

Bayesian Data-Analysis Toolbox  
Release 4.23, Manual Version 3

G. Larry Bretthorst  
Biomedical MR Laboratory  
Washington University School Of Medicine,  
Campus Box 8227  
Room 2313, East Bldg.,  
4525 Scott Ave.  
St. Louis MO 63110  
<http://bayes.wustl.edu>  
Email: [gbretthorst@wustl.edu](mailto:gbretthorst@wustl.edu)

September 18, 2018



# Contents

<b>Manual Status</b>	<b>16</b>
<b>1 An Overview Of The Bayesian Analysis Software</b>	<b>19</b>
1.1 The Server Software	19
1.2 The Client Interface	22
1.2.1 The Global Pull Down Menus	24
1.2.2 The Package Interface	24
1.2.3 The Viewers	27
<b>2 Installing the Software</b>	<b>29</b>
<b>3 the Client Interface</b>	<b>33</b>
3.1 The Global Pull Down Menus	35
3.1.1 the Files menu	35
3.1.2 the Packages menu	40
3.1.3 the WorkDir menu	45
3.1.4 the Settings menu	46
3.1.5 the Utilities menu	50
3.1.6 the Help menu	50
3.2 The Submit Job To Server area	51
3.3 The Server area	52
3.4 Interface Viewers	52
3.4.1 the Ascii Data Viewer	53
3.4.2 the fid Data Viewer	53
3.4.3 Image Viewer	59
3.4.3.1 the Image List area	59
3.4.3.2 the Set Image area	62
3.4.3.3 the Image Viewing area	62
3.4.3.4 the Grayscale area on the bottom	63
3.4.3.5 the Pixel Info area	63
3.4.3.6 the Image Statistics area	64
3.4.4 Prior Viewer	65
3.4.5 Fid Model Viewer	68
3.4.5.1 The fid Model Format	70

3.4.5.2	The Fid Model Reports . . . . .	71
3.4.6	Plot Results Viewer . . . . .	71
3.4.7	Text Results Viewer . . . . .	74
3.4.8	Files Viewer . . . . .	80
3.5	Common Interface Plots . . . . .	80
3.5.1	Data, Model And Residual Plot . . . . .	81
3.5.2	Posterior Probability For A Parameter . . . . .	82
3.5.3	Maximum Entropy Histograms . . . . .	83
3.5.4	Markov Monte Carlo Samples . . . . .	83
3.5.5	Probability Vs Parameter Samples plot . . . . .	86
3.5.6	Expected Log Likelihood Plot . . . . .	88
3.5.7	Scatter Plots . . . . .	88
3.5.8	Logarithm of the Posterior Probability Plot . . . . .	91
3.5.9	Fortran/C Code Viewer . . . . .	93
3.5.9.1	Fortran/C Model Viewer Popup Editor . . . . .	93
<b>4</b>	<b>An Introduction to Bayesian Probability Theory</b>	<b>99</b>
4.1	The Rules of Probability Theory . . . . .	99
4.2	Assigning Probabilities . . . . .	102
4.3	Example: Parameter Estimation . . . . .	109
4.3.1	Define The Problem . . . . .	110
4.3.1.1	The Discrete Fourier Transform . . . . .	110
4.3.1.2	Aliases . . . . .	113
4.3.2	State The Model—Single-Frequency Estimation . . . . .	114
4.3.3	Apply Probability Theory . . . . .	115
4.3.4	Assign The Probabilities . . . . .	118
4.3.5	Evaluate The Sums and Integrals . . . . .	120
4.3.6	How Probability Generalizes The Discrete Fourier Transform . . . . .	123
4.3.7	Aliasing . . . . .	126
4.3.8	Parameter Estimates . . . . .	132
4.4	Summary and Conclusions . . . . .	136
<b>5</b>	<b>Given Exponential Model</b>	<b>137</b>
5.1	The Bayesian Calculation . . . . .	139
5.2	Outputs From The Given Exponential Package . . . . .	141
<b>6</b>	<b>Unknown Number of Exponentials</b>	<b>143</b>
6.1	The Bayesian Calculations . . . . .	145
6.2	Outputs From The Unknown Number of Exponentials Package . . . . .	148
<b>7</b>	<b>Inversion Recovery</b>	<b>151</b>
7.1	The Bayesian Calculation . . . . .	153
7.2	Outputs From The Inversion Recovery Package . . . . .	154

<b>8</b>	<b>Bayes Analyze</b>	<b>155</b>
8.1	Bayes Model	159
8.2	The Bayes Analyze Model Equation	161
8.3	The Bayesian Calculations	167
8.4	Levenberg-Marquardt And Newton-Raphson	171
8.5	Outputs From The Bayes Analyze Package	176
8.5.1	The “bayes.params.nnnn” Files	177
8.5.1.1	The Bayes Analyze File Header	178
8.5.1.2	The Global Parameters	182
8.5.1.3	The Model Components	184
8.5.2	The “bayes.model.nnnn” Files	185
8.5.3	The “bayes.output.nnnn” File	186
8.5.4	The “bayes.probabilities.nnnn” File	190
8.5.5	The “bayes.log.nnnn” File	193
8.5.6	The “bayes.status.nnnn” and “bayes.accepted.nnnn” Files	196
8.5.7	The “bayes.model.nnnn” File	197
8.5.8	The “bayes.summary1.nnnn” File	198
8.5.9	The “bayes.summary2.nnnn” File	199
8.5.10	The “bayes.summary3.nnnn” File	200
8.6	Bayes Analyze Error Messages	200
<b>9</b>	<b>Big Peak/Little Peak</b>	<b>207</b>
9.1	The Bayesian Calculation	209
9.2	Outputs From The Big Peak/Little Peak Package	216
<b>10</b>	<b>Metabolic Analysis</b>	<b>219</b>
10.1	The Metabolic Model	223
10.2	The Bayesian Calculation	225
10.3	The Metabolite Models	228
10.3.1	The IPGD_D2O Metabolite	228
10.3.2	The Glutamate.2.0 Metabolite	232
10.3.3	The Glutamate.3.0 Metabolite	235
10.4	The Example Metabolite	236
10.5	Outputs From The Bayes Metabolite Package	238
<b>11</b>	<b>Find Resonances</b>	<b>239</b>
11.1	The Bayesian Calculations	241
11.2	Outputs From The Bayes Find Resonances Package	246
<b>12</b>	<b>Diffusion Tensor Analysis</b>	<b>247</b>
12.1	The Bayesian Calculation	249
12.2	Using The Package	254
<b>13</b>	<b>Big Magnetization Transfer</b>	<b>259</b>
13.1	The Bayesian Calculation	259
13.2	Outputs From The Big Magnetization Transfer Package	262

<b>14 Magnetization Transfer</b>	<b>265</b>
14.1 The Bayesian Calculation	267
14.2 Using The Package	271
<b>15 Magnetization Transfer Kinetics</b>	<b>275</b>
15.1 The Bayesian Calculation	277
15.2 Using The Package	281
<b>16 Given Polynomial Order</b>	<b>285</b>
16.1 The Bayesian Calculation	287
16.1.1 Gram-Schmidt	287
16.1.2 The Bayesian Calculation	288
16.2 Outputs From the Given Polynomial Order Package	290
<b>17 Unknown Polynomial Order</b>	<b>293</b>
17.1 Bayesian Calculations	295
17.1.1 Assigning Priors	296
17.1.2 Assigning The Joint Posterior Probability	297
17.2 Outputs From the Unknown Polynomial Order Package	299
<b>18 Errors In Variables</b>	<b>303</b>
18.1 The Bayesian Calculation	305
18.2 Outputs From The Errors In Variables Package	308
<b>19 Behrens-Fisher</b>	<b>311</b>
19.1 Bayesian Calculation	311
19.1.1 The Four Model Selection Probabilities	314
19.1.1.1 The Means And Variances Are The Same	315
19.1.1.2 The Mean Are The Same And The Variances Differ	317
19.1.1.3 The Means Differ And The Variances Are The Same	318
19.1.1.4 The Means And Variances Differ	319
19.1.2 The Derived Probabilities	320
19.1.3 Parameter Estimation	321
19.2 Outputs From Behrens-Fisher Package	322
<b>20 Enter Ascii Model</b>	<b>329</b>
20.1 The Bayesian Calculation	331
20.1.1 The Bayesian Calculations Using Eq. (20.1)	331
20.1.2 The Bayesian Calculations Using Eq. (20.2)	332
20.2 Outputs Form The Enter Ascii Model Package	335
<b>21 Test Ascii Model</b>	<b>337</b>

<b>22 Enter Ascii Model Selection</b>	<b>341</b>
22.1 The Bayesian Calculations	343
22.1.1 The Direct Probability With No Amplitude Marginalization	344
22.1.2 The Direct Probability With Amplitude Marginalization	346
22.1.2.1 Marginalizing the Amplitudes	347
22.1.2.2 Marginalizing The Noise Standard Deviation	352
22.2 Outputs Form The Enter Ascii Model Package	353
<b>23 Binned Density Function Estimation</b>	<b>355</b>
23.1 Using The Package	355
23.2 The Bayesian Calculations	357
<b>24 Kernel Density Function Estimation</b>	<b>361</b>
24.1 Using The Package	361
24.2 The Output Plots	364
24.3 The Bayesian Calculations	369
24.4 The Output Reports	372
<b>25 MaxEnt Density Function Estimation</b>	<b>373</b>
25.1 Using The Package	373
25.2 Introduction	377
25.3 Review of The Maximum Entropy Method Of Moments	381
25.4 The Bayesian Calculations	387
25.5 Summary and Conclusions	392
<b>26 Phasing An Image</b>	<b>395</b>
26.1 The Bayesian Calculation	396
26.2 Using The Package	402
<b>27 Phasing An Image Using Non-Linear Phases</b>	<b>405</b>
27.1 The Model Equation	405
27.2 The Bayesian Calculations	407
27.3 The Interfaces To The Nonlinear Phasing Routine	409
<b>28 Analyze Image Pixel</b>	<b>411</b>
28.1 Modification History	413
<b>29 The Image Model Selection Package</b>	<b>415</b>
29.1 The Bayesian Calculations	417
29.2 Outputs Form The Image Model Selection Package	418
<b>30 Analyze Image Pixel Unique</b>	<b>423</b>
<b>31 Bayes Test Data</b>	<b>427</b>
31.1 Running the Package	427
31.2 Outputs From the Bayes Test Data Package	429

<b>A Ascii Data File Formats</b>	<b>435</b>
A.1 Ascii Input Data Files . . . . .	435
A.2 Ascii Image File Formats . . . . .	436
A.3 The Abscissa File Format . . . . .	437
<b>B Markov chain Monte Carlo With Simulated Annealing</b>	<b>439</b>
B.1 Metropolis-Hastings Algorithm . . . . .	440
B.2 Multiple Simulations . . . . .	441
B.3 Simulated Annealing . . . . .	442
B.4 The Annealing Schedule . . . . .	442
B.5 Killing Simulations . . . . .	443
B.6 the Proposal . . . . .	444
<b>C Thermodynamic Integration</b>	<b>445</b>
<b>D McMC Values Report</b>	<b>449</b>
<b>E Writing Fortran/C Models</b>	<b>455</b>
E.1 Model Subroutines, No Marginalization . . . . .	455
E.2 The Parameter File . . . . .	458
E.3 The Subroutine Interface . . . . .	460
E.4 The Subroutine Declarations . . . . .	462
E.5 The Subroutine Body . . . . .	463
E.6 Model Subroutines With Marginalization . . . . .	464
<b>F the Bayes Directory Organization</b>	<b>469</b>
<b>G 4dfp Overview</b>	<b>471</b>
<b>H Outlier Detection</b>	<b>475</b>
<b>Bibliography</b>	<b>479</b>
<b>Index</b>	<b>484</b>

# List of Figures

1.1	The Start Up Window . . . . .	23
1.2	Example Package Exponential Interface . . . . .	25
2.1	Installation Kit For The Bayesian Analysis Software . . . . .	31
3.1	The Start Up Window . . . . .	34
3.2	The Files Menu . . . . .	35
3.3	The Files/Load Image Submenu . . . . .	37
3.4	The Packages Menu . . . . .	41
3.5	The Working Directory Menu . . . . .	46
3.6	The Working Directory Information Popup . . . . .	47
3.7	The Settings Pull Down Menu . . . . .	47
3.8	The McMC Parameters Popup . . . . .	48
3.9	The Edit Server Popup . . . . .	49
3.10	The Submit Job Widgets . . . . .	51
3.11	The Server Widgets Group . . . . .	52
3.12	The Ascii Data Viewer . . . . .	54
3.13	The Fid Data Viewer . . . . .	55
3.14	Fid Data Display Type . . . . .	56
3.15	Fid Data Options Menu . . . . .	58
3.16	The Image Viewer . . . . .	60
3.17	The Image Viewer Right Mouse Popup Menu . . . . .	61
3.18	The Prior Probability Viewer . . . . .	66
3.19	The Fid Model Viewer . . . . .	69
3.20	The Plot Results Viewer . . . . .	72
3.21	Plot Information Popup . . . . .	73
3.22	The Text Results Viewer . . . . .	75
3.23	The Bayes Condensed File . . . . .	78
3.24	Data, Model, And Resid Plot . . . . .	81
3.25	The Parameter Posterior Probabilities . . . . .	82
3.26	The Maximum Entropy Histograms . . . . .	84
3.27	The Parameter Samples Plot . . . . .	85
3.28	Posterior Probability Vs Parameter Value . . . . .	86
3.29	Posterior Probability Vs Parameter Value, A Skewed Example . . . . .	87
3.30	The Expected Value Of The Logarithm Of The Likelihood . . . . .	89

3.31	The Scatter Plots . . . . .	90
3.32	The Logarithm Of The Posterior Probability By Repeat Plot . . . . .	92
3.33	The Fortran/C Model Viewer . . . . .	94
3.34	The Fortran/C Code Editor . . . . .	95
4.1	Frequency Estimation Using The DFT . . . . .	112
4.2	Aliases . . . . .	113
4.3	Nonuniformly Nonsimultaneously Sampled Sinusoid . . . . .	127
4.4	Alias Spacing . . . . .	128
4.5	Which Is The Critical Time . . . . .	130
4.6	Example, Frequency Estimation . . . . .	131
4.7	Estimating The Sinusoids Parameters . . . . .	133
5.1	The Given And Unknown Number Of Exponential Package Interface . . . . .	138
6.1	The Unknown Exponential Interface . . . . .	144
6.2	The Distribution Of Models . . . . .	149
6.3	The Posterior Probability For Exponential Model . . . . .	150
7.1	The Inversion Recovery Interface . . . . .	152
8.1	Bayes Analyze Interface . . . . .	156
8.2	Bayes Analyze Fid Model Viewer . . . . .	160
8.3	The Bayes Analyze File Header . . . . .	179
8.4	The bayes.noise File . . . . .	180
8.5	Bayes Analyze Global Parameters . . . . .	183
8.6	The Third Section Of The Parameter File . . . . .	184
8.7	Example Of An Initial Model In The Output File . . . . .	187
8.8	Base 10 Logarithm Of The Odds . . . . .	187
8.9	A Small Sample Of The Output Report . . . . .	188
8.10	Bayes Analyze Uncorrelated Output . . . . .	189
8.11	The bayes.proBABILITIES.nnnn File . . . . .	191
8.12	The bayes.log.nnnn File . . . . .	193
8.13	The bayes.status.nnnn File . . . . .	196
8.14	The bayes.model.nnnn File . . . . .	197
8.15	The bayes.model.nnnn File Uncorrelated Resonances . . . . .	198
8.16	Bayes Analyze Summary Header . . . . .	198
8.17	The Summary2 (Best Summary) . . . . .	199
8.18	The Summary3 Report . . . . .	201
9.1	The Big Peak/Little Peak Interface . . . . .	208
9.2	The Time Dependent Parameters . . . . .	218
10.1	The Bayes Metabolite Interface . . . . .	220
10.2	The Bayes Metabolite Viewer . . . . .	222
10.3	Bayes Metabolite Parameters And Probabilities List . . . . .	227
10.4	The IPGD_D20 Metabolite . . . . .	229

10.5	Bayes Metabolite IPGD_D20 Spectrum . . . . .	230
10.6	Bayes Metabolite, The Fraction of Glucose . . . . .	231
10.7	Glutamate Example Spectrum . . . . .	233
10.8	Estimating The $F_{c0}$ , $y$ and $F_{a0}$ Parameters . . . . .	236
10.9	Bayes Metabolite, The Ethyl Ether Example . . . . .	237
11.1	The Find Resonances Interface With The Ethyl Ether Spectrum . . . . .	240
12.1	The Diffusion Tensor Package Interface . . . . .	248
12.2	Diffusion Tensor Parameter Estimates . . . . .	256
12.3	Diffusion Tensor Posterior Probability For The Model . . . . .	257
13.1	The Big Magnetization Package Interface . . . . .	260
13.2	Big Magnetization Transfer Example Fid . . . . .	263
13.3	Big Magnetization Transfer Expansion . . . . .	263
13.4	Big Magnetization Transfer Peak Pick . . . . .	264
14.1	The Magnetization Transfer Package Interface . . . . .	266
14.2	Magnetization Transfer Package Peak Picking . . . . .	272
14.3	Magnetization Transfer Example Data . . . . .	273
14.4	Magnetization Transfer Example Spectrum . . . . .	274
15.1	Magnetization Transfer Kinetics Package Interface . . . . .	276
15.2	Magnetization Transfer Kinetics Package Arrhenius Plot . . . . .	282
15.3	Magnetization Transfer Kinetics Water Viscosity Table . . . . .	283
16.1	Given Polynomial Order Package Interface . . . . .	286
16.2	Given Polynomial Order Scatter Plot . . . . .	291
17.1	Unknown Polynomial Order Package Interface . . . . .	294
17.2	The Distribution of Models On The Console Log . . . . .	298
17.3	The Posterior Probability For The Polynomial Order . . . . .	300
18.1	The Errors In Variables Package Interface . . . . .	304
18.2	The McMC Values File Produced By The Errors In Variables Package . . . . .	310
19.1	The Behrens-Fisher Interface . . . . .	312
19.2	Behrens-Fisher Hypotheses Tested . . . . .	313
19.3	Behrens-Fisher Console Log . . . . .	323
19.4	Behrens-Fisher Status Listing . . . . .	324
19.5	Behrens-Fisher McMC Values File, The Preamble . . . . .	325
19.6	Behrens-Fisher McMC Values File, The Middle . . . . .	326
19.7	Behrens-Fisher McMC Values File, The End . . . . .	327
20.1	Enter Ascii Model Package Interface . . . . .	330
21.1	The Test Ascii Model Package Interface . . . . .	338

21.2	The Mcmc Values Report For The Test Ascii Model Package . . . . .	340
22.1	The Enter Ascii Model Selection Package Interface . . . . .	342
23.1	Binned Histograms Density Estimation Package Interface . . . . .	356
23.2	Binned Histograms Package Example Data . . . . .	358
23.3	Binned Histograms Given Number Of Bins . . . . .	359
23.4	Binned Histograms With Automatic Determination Of The Number Of Bins . . . . .	360
24.1	The Nine Common Kernels . . . . .	362
24.2	Kernel Density Estimation Interface . . . . .	363
24.3	Posterior Probability For The Kernel Type . . . . .	365
24.4	Posterior Probability For The Number Of Kernels . . . . .	366
24.5	The Model Averaged Kernel Density Function . . . . .	367
24.6	Kernel Density Function And Samples . . . . .	368
24.7	A Binned Histogram, The Model Averaged Density Function And The Difference . . . . .	369
25.1	MaxEnt Density Function Estimation Package Interface . . . . .	374
25.2	The MaxEnt Method Of Moments McMC Values Report . . . . .	376
25.3	The MaxEnt Density Function Estimation Package Console Log . . . . .	377
25.4	The Posterior Probability For The Number Of Multipliers . . . . .	378
25.5	The Model Averaged Density Function And Samples . . . . .	379
25.6	The Data, Model And Residuals . . . . .	380
25.7	Density Function Sample Data . . . . .	382
25.8	The First 10 Power And Central Moments . . . . .	384
25.9	Maximum Entropy Distributions As A Function Of The Number of Multipliers . . . . .	385
25.10	The Posterior Probability For The Number Of Multipliers . . . . .	390
25.11	Posterior Probability For The Lagrange Multipliers . . . . .	391
25.12	Maximum Entropy Model Averaged Density Function . . . . .	392
26.1	Absorption Model Images . . . . .	396
26.2	The Interface To The Image Phasing Package . . . . .	397
26.3	Linear Phasing Package The Console Log . . . . .	403
27.1	Nonlinear Phasing Example . . . . .	406
27.2	The Interface To The Nonlinear Phasing Package . . . . .	410
28.1	The Interface To The Analyze Image Pixels Package . . . . .	412
29.1	The Interface To The Image Model Selection Package . . . . .	416
29.2	Single Exponential Example Image . . . . .	419
29.3	Single Exponential Example Data . . . . .	420
29.4	Posterior Probability For The ExpOneNoConst Model . . . . .	421
30.1	The Analyze Image Pixel Unique Abscissa Package Interface . . . . .	424
31.1	The Bayes Test Data Package Interface . . . . .	428

31.2	The Output Image Test Data . . . . .	429
31.3	Example Exponential Test Data . . . . .	430
31.4	The Front of the McMC Values Report For Bayes Test Data Package . . . . .	432
31.5	The Bottom Part of the Mcmc Values Report For The Bayes Test Data Package . . . . .	433
A.1	Ascii Data File Format . . . . .	436
D.1	The McMC Values Report Header . . . . .	450
D.2	McMC Values Report, The Middle . . . . .	451
D.3	The McMC Values Report, The End . . . . .	452
E.1	Writing Models A Fortran Example . . . . .	456
E.2	Writing Models A C Example . . . . .	457
E.3	Writing Models, The Parameter File . . . . .	459
E.4	Writing Models Fortran Declarations . . . . .	463
E.5	Writing Models Fortran Example . . . . .	466
E.6	Writing Models The Parameter File . . . . .	467
G.1	Example FDF File Header . . . . .	473
H.1	The Posterior Probability For The Number of Outliers . . . . .	476
H.2	The Data, Model and Residual Plot With Outliers . . . . .	478



# List of Tables

8.1	Multiplet Relative Amplitudes . . . . .	165
8.2	Bayes Analyze Models . . . . .	181
8.3	Bayes Analyze Short Descriptions . . . . .	195

## Manual Version 3

# Manual Status, Software Release 4.23

Date	Status	Package or Chapter
09-13-2018	Changed	I Corrected a plot in the Chapter <a href="#">25</a>
07-13-2018	Changed	A number of typos were corrected in both the interface and manual
11-10-2015	Changed	All Java is now distributed with a certified signing certificate
02-18-2015	Added	Added IO routines for Loading Bruker data
07-08-2014	Added	First draft of the Bayes Test Data Chapter <a href="#">31</a>
08-29-2013	Added	First draft of the Kernel Density Function Chapter <a href="#">24</a>
08-29-2013	Added	First draft of the MaxEnt Density Function Chapter <a href="#">25</a>
08-20-2013	Added	Description of Param Vs Prob Plots , Chapter <a href="#">3.5.5</a>
02-20-2012	First Draft	Enter Ascii Model Package, Chapter <a href="#">20</a>
02-14-2012	Edited	Behrens-Fisher Package, Chapter <a href="#">19</a>
02-10-2012	Edited	Errors In Variables Package, Chapter <a href="#">18</a> I also fixed a problem in the list of figures related to how I was handling captions
02-08-2012	Edited	Unknown Polynomial Model Package, Chapter <a href="#">17</a>
02-02-2012	Edited	Given Polynomial Model Package, Chapter <a href="#">16</a>
02-01-2012	Edited	Kinetic magnetization Transfer Package, Chapter <a href="#">15</a>
01-31-2012	Edited	Magnetization Transfer Package, Chapter <a href="#">14</a>
01-27-2012	Edited	Big Magnetization Transfer Package, Chapter <a href="#">13</a>
01-24-2012	Edited	Bayes Diffusion Tensor Package, Chapter <a href="#">12</a>
01-22-2012	Edited	I worked on the Bibliography and added a number of hyperlinks to the Bibliography entries.
01-19-2012	First Draft	Bayes Find Resonance Package, Chapter <a href="#">11</a>
01-12-2012	Edited	Metabolite Package, Chapter <a href="#">10</a>
01-10-2012	Edited	Big Peak/Little Peak, Chapter <a href="#">9</a> I also edited the Bayes Analyze section and updated the index, Chapter <a href="#">8</a>
01-09-2012	Edited	Inversion Recovery, Chapter <a href="#">7</a>
01-06-2012	Edited	Unknown Number of Exponential, Chapter <a href="#">6</a>
01-06-2012	Edited	Given Exponential, Chapter <a href="#">5</a>
01-05-2012	Edited	Installation the software, Chapter <a href="#">2</a>
01-04-2012	First Draft	Bayes Analyze, Chapter <a href="#">8</a>
12-19-2011	Partial Draft	Bayes Analyze, Chapter <a href="#">8</a>
Nov	First Draft	4dfp, Appendix <a href="#">G</a>
Nov	First Draft	Given Exponential model, Chapter <a href="#">5</a>
Nov	First Draft	Outlier detection, Appendix <a href="#">H</a>
Nov	First Draft	Ascii File Formats, Appendix <a href="#">A</a>

Date	Status	Package or Chapter
Oct	First Draft	Thermo dynamic integration, Appendix <b>C</b>
Oct	First Draft	Fortran/C code viewer, Appendix <b>E</b>
Oct	First Draft	Directory organization, Appendix <b>F</b>
Oct	First Draft	Markov Chain Monte Carlo, Appendix <b>B</b>
Oct	First Draft	Update the Index
Sept	First Draft	Software overview, Chapter <b>1</b>
Sept	First Draft	Introduction to probability theory, Chapter <b>4</b>
Sept	First Draft	Unknown Number of Exponentials model, Chapter <b>6</b>
Sept	First Draft	The Software Interface, Chapter <b>3</b>
Sept	First Draft	Installing the software, Chapter <b>2</b>



# Chapter 1

## An Overview Of The Bayesian Analysis Software

The Bayesian Analysis Software developed at Washington University is a client/server based software package that analyzes common problems in the sciences using Bayesian Probability theory. The Software is a client/server software package consisting of three distinct sets of software: The Server software, the Client software and the Installation software. The Server software actually runs the Bayesian analysis. The Client software is an interface that functions as a buffer between the user and the server software. Finally, there is an Installation procedure that downloads and installs software.

The software is loosely divided into a series of programs which we refer to as packages. Each package addresses a specific kind of problem. For example, the exponential package estimates the parameters associated with exponential models. All of the calculations presented in this manual use Bayesian probability [1, 36] theory to estimate the parameters or to perform model selection. For those unfamiliar with Bayesian Probability theory Chapter 4 contains a tutorial, and there are a number of excellent tutorials [31, 40, 3, 11] and books [33, 64, 66, 61, 32] in the literature. Most but not all of the packages described in this manual use Markov chain Monte Carlo to approximate the Bayesian posterior probabilities. For those unfamiliar with Markov chain see [24, 46] and Section B gives a description of how the various packages implement the Markov chain Monte Carlo calculations.

### 1.1 The Server Software

Before we describe the interface, we briefly describe the server software and how the client software interfaces to it. The server, the machine that actually runs the Bayesian Analysis, can be any multi-core LinuxPC, either 32 or 64, bit running GNU/Linux (CentOS 4.7 or higher) or a Sun system running Solaris 9 or 10. For all servers it is assumed that an httpd server is installed and functional on the server machine. When the software is installed on the server, the installation procedure downloads the latest version of the software from Washington University and installs it on the server, see Chapter 2 for instructions on how to install the software. The server software consists of three parts: a web server, a set of scripts that are used by the web server, and the programs that implement the Bayesian probability theory calculations. The web server handles the

communications between the client and the server applications. The clients send requests to the servers and the servers use a set of scripts to handle these requests. These scripts do things as simple as listing the process currently running on the server; to things as complicated as unpacking an analysis and then running the appropriate software. In the following Chapters we will describe each of these software packages.

The server software contains the programs that run the Bayesian analysis packages, while the Client Interface allows one too easily access these programs. Here is a list of the packages with a brief description of each. The Client Interface Chapter, Chapter 3, contains a more extensive description of the packages, and the later Chapters in this manual contain detail information about each package.

- The Exponential package estimates the decay rate constants and amplitudes of signals known to be decaying exponentially.
- The Unknown Exponential package estimates the decay rate constants and amplitudes of signals known to be decaying exponentially when the number of exponential components are unknown.
- The Inversion Recovery package is a special type of exponential analysis that is very common in NMR. In this problem the NMR signal starts at a negative value and decays to a positive value.
- The Diffusion Tensor package analyzes NMR diffusion measurements using one, two or three diffusion tensor models with or without a constant.
- The Enter Ascii Model package allows the user to define a model and then use Bayesian Probability theory to analyze data using that model.
- The Enter Ascii Model Selection package utilizes the models generated for Enter Ascii to do model selection.
- The Test Ascii Model model package supports the other packages that use Ascii Models by giving the user a means of testing models.
- The Magnetization Transfer (two sites) package solves the Bloch-McConnell equations to obtain the exchange rate constants for two site magnetization exchange.
- The Magnetization Transfer Kinetics package is a magnetization transfer package that solves the Bloch-McConnell equations at multiple temperatures and concentrations to derive the entropy and enthalpies of the the exchange process.
- The Big Magnetization Transfer package solves the magnetization transfer problem when one of the sites can be considered infinite compared to the other.
- The Bayes Analyze package is a time domain frequency estimation package that is fully capable of determining the number of resonances in an FID and estimating the resonance parameters.
- The Big Peak/Little Peak package analyzes time domain FID data in which there is a single big peak that may be many orders of magnitude larger in intensity (the big peak) than the metabolic peaks (the little peaks) of interest.

- The Find Resonances package analyzes NMR FID data looking for resonances. The program is a model selection program that is attempting to determine the number of resonances in the data and estimate the parameters associated with those resonances.
- The Metabolite package analyzes FID data from a number of known samples, for example a C13 FID of Glutamate. The intensity of the Glutamate resonances are related to each other through a metabolic model. This model can be very simple or very complex. Metabolic models can be added to the library of models, but there are no facilities for building these models within the interface.
- The Behrens-Fisher package solves the classical medical testing problem: given two experiments that consist of repeated measurements of the same quantity where in the second measurement one has change some experiential parameter determine if the experiments are the same or if they differ.
- The Errors in Variables package solves the problem of straight-line fitting when there are errors in both the measured data and in the measured time, or abscissa value. The implementation in this package allows the user to set the order of the polynomial to be fit, so its a little more general than just straight-line fitting.
- The Polynomial Models package fits polynomials of either a given or an unknown order to the input data. When the unknown model is selected the programs that implement the calculation compute the posterior probability for the order of the polynomial needed to fit the data down to the noise.
- The Maximum Entropy Histograms density estimation package is a ASCII package that takes as its input a sample drawn from an unknown density function. It then computes the posterior probability for the number of nontrivial moments in the data, i.e., the number of Lagrange multipliers need by the Maximum Entropy density function. Its output is the estimated density function with error bars on the estimated density function.
- The Binned Histogram package estimates a binned density function with error bars. In the near future we will be enhancing this package to perform model selection. That is to say the binned histogram package will automatically determine both the number of bins and smoothing need to describe the density function.
- The Linear Phasing package produces linearly phased images. In NMR the complex image data have phases that vary across the image in a linear fashion. These linear phases are present because of the gradients that are used to generate an MR image. The linear phasing package estimates the value of the zero and first order phases in the phase encode and readout domains and then unwraps this phase so that the image can be displayed in absorption mode.
- The Non-Linear phasing package phases images that have phases that are varying in a Non-Linear fashion. In this package the phases are estimated on a pixel by pixel basis and the estimated phase is used to generate an absorption mode image.
- The Image Pixels package loads a one of the predefined Ascii models and then uses that model to analyze images on a pixel by pixel basis. The loaded models can be generated by the users or they can be loaded from a system library that we provide.

- The Image Pixels Model Selection package extends the concepts in Analyze Image Pixels to model selection. In this package the user can load a number of different models that describe the signal in a pixel and then the program will compute the posterior probability for the model. Outputs include the posterior probability for the model indicator as well as parameter maps of the parameters.

## 1.2 The Client Interface

The interface to the Bayesian Analysis software is a Java interface that runs on any machine having Java 1.8 update 112 or higher. Assuming the Bayesian Analysis software has been installed on a server at your site, for arguments sake lets call this machine “your.server.net,” then you can bring up the interface, the client software, by issuing:

```
javaws http://your.server.net:8080/Bayes/launch.jnlp
```

where “javaws” is the Java web-start utility and comes with most Java installations, “your.server.net” should be replaced by your server name or IP address, and you should replace “8080” by the port number used by your installation, see Chapter 2 for a description of how to install the software.

If you do not have the software installed on your local machines, you can download the interface directly from Washington University:

```
javaws http://bayes.wustl.edu/Bayes/launch.jnlp
```

This version of the interface, will allow you to view the packages and to determine what is available. However, because the software has not been installed on one of your machines, you will not be able to run an analysis.

Assuming you use one of these two methods to start the interface, it will display the default startup page shown in Fig. 1.1. The purpose of the startup page is to allow you to restart an analysis. When you exit the interface or change working directories, the interface saves the current settings in a special Java properties file. When the interface starts, it consults this file and determines what your last WorkDir was and how to restart that analysis. If an analysis was saved, the interface displays the messages shown in Fig. 1.1, the lines starting “To restore analysis”. This line contains the name of the package that was being processed, in this case the package name was “AnalyzeImagePixels” and the analysis was saved in a WorkDir named “Given”. If the **Restore Analysis** button is activated then the “Given/AnalyzeImagePixels” analysis will be restored to its previous status. When the interface finishes restoring the analysis, it will function exactly like you never exited the WorkDir or interface.

If you do not want to restore an analysis then changing the package will delete the contents of the current WorkDir and configure the WorkDir for the new package. If you do not want to change packages, but want to check on another analysis then changing the current working directory using the **WorkDir** menu will cause the interface to switch to the new WorkDir and assuming that WorkDir contains a previous analysis that analysis will be restored to its previous status.

Finally, if you wish to start a completely new analysis then selecting **WorkDir/Edit** will bring up a popup that will allow you to create a new WorkDir. After you create and join a new WorkDir the first thing you must do is to select the package you wish to use.

The global pull down menus along the top of the startup page are always present on all package interfaces, not just the startup page. They allow the user to load files, select packages, configure

Figure 1.1: The Start Up Window

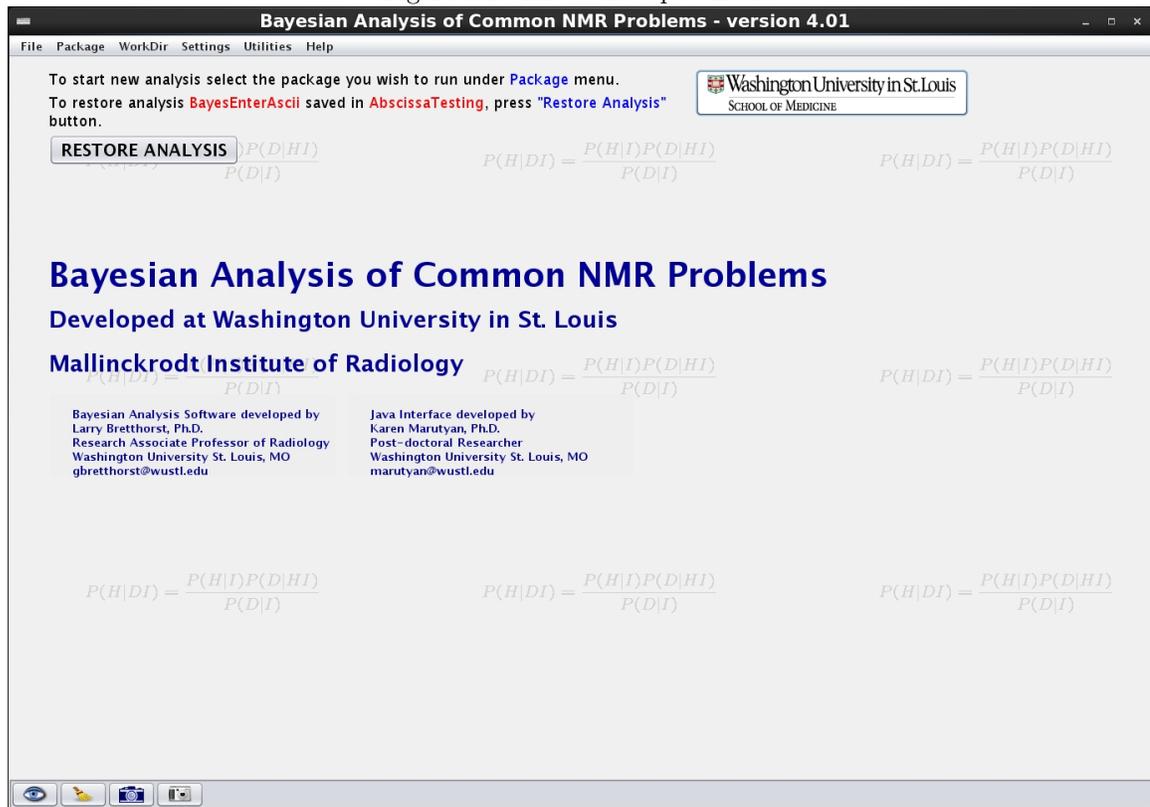


Figure 1.1: The Bayesian Analysis Startup Page allows you to select what functions you wish to perform. For example you might restore an old analysis, change a setting, run one of the utility programs or select a new WorkDir or a new Bayesian Analysis package.

servers, change working directories, set options, etc. Each pull down menu has multiple functions and the following Sections explain these menus and how to go about using them.

### 1.2.1 The Global Pull Down Menus

The global pull down menus at the top of the interface are always present. They allow you to select Bayesian Analysis applications, configure servers, change WorkDir, etc. Each item across the top is a pull down menu and each menu has multiple functions. These functions are explained in detail in Section 3 Here we give a brief summary of these menus:

**Files** is pull down menu that allows you to perform various tasks involving files. For example, you can load Ascii data, FID spectral or image data and images. Additionally, you can save the current WorkDir, and you can restore a previously saved experiment. See Section 3.1.1 for more on the files submenu.

**Packages** is a pull down menu that allows you to select the Bayesian Analysis package you wish to use. Each of the packages is described in more detail in the upcoming Chapters. See Section 3.1.2 for a more extensive discussion of the packages pull down menu.

**WorkDir** is a pull down menu that allows you to select, create or delete a WorkDir. Working directories are contained within the “Bayes” directory in your home account. These directories are scratch areas used to contain the loaded data, configuration files, and the results of running an analysis. See Section 3.1.3 for a more extensive discussion of working directories.

**Settings** is a pull down menu that allows you to configure the Bayesian Analysis packages. The various menu items allow you to configure the Markov chain Monte Carlo simulations, see Section 3.1.4; add, delete and modify server settings, See Section 3.1.4; and it allows you to configure some optional features of the software.

**Utilities** is a pull down menu that allows you to start a memory monitor, get information on the system you are running, and allows you to determine if there is an updated version of the Bayesian Analysis software. See Section 3.1.5 for more on the utilities.

**Help** is a pull down menu that allows you to view information about the current installation of the Bayesian Analysis software, and it allows you to visit the Bayesian Analysis Software home page.

### 1.2.2 The Package Interface

When one of the packages is selected the interface displays that package interface. For example if the Exponential package is selected, the interface shown in Fig. 1.2 is displayed. This interface is very similar to the interface of many other packages and we will use it to illustrate some of the general features of the Interface.

First, note that the global menus that were present on the Bayesian Analysis Home Page are present on all package interfaces. Second, below the global menus is an area that is used to configure a package. Each set of widgets are enclosed in a highlighted box. We are going to call these enclosed widgets, widget groups and we will name them based on the name above each group. So on the Exponential interface there are five widget groups. The first two, Submit Job to Server and Server

Figure 1.2: Example Package Exponential Interface

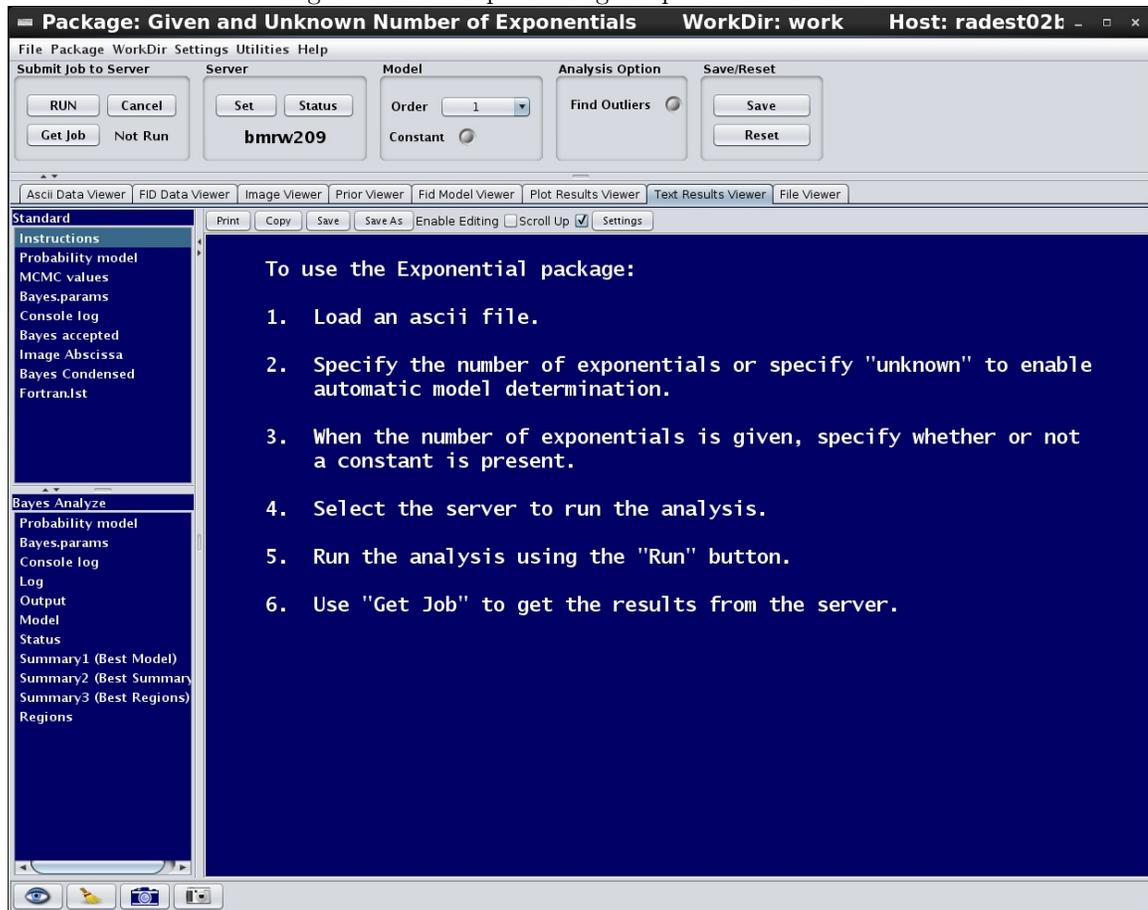


Figure 1.2: When one of the Bayesian Analysis packages is selected from the “Packages” pull down menu, the appropriate interface is displayed; here the interface to the exponential package is displayed. A package interface consists of three parts: the global pull down menus along the top, the package setup widgets just below the global pull down menus, and the viewing area, the dark blue area, at the bottom.

widget groups are common to all packages. However, most packages have some variation of the five seen in the Exponential package, but some packages have more and some have less. For the exponential package here is a brief description of these widget groups:

**Submit Job to Server** is a widget group that has three buttons and one text area. This widget group is responsible for submitting jobs, checking on their status and, when necessary canceling jobs.

- The **Run** button is used to submit a job to a server. If the currently selected server is named Server1, then the Run button will submit the job to Server1 and it will change the Run Status text area to Active or Submitted depending on whether the server uses a queuing facility. When the run button is activated most of the widgets on the interface are disabled. This is to prevent the user from making changes to the configuration while a job is running.
- The **Get Job** button sends a request to the currently selected server requesting the status of the current job. If the status is other than “Run” the Run Status text area is updated with the current status and nothing else happens. If the current status is Run, the job is fetched from the server and the appropriate files are updated. Finally the Run status text area is set to Run. If for some reason the job failed, the Run Status text area is set to Error.
- The **Cancel** button will send a request to the server to cancel a job. When the server receives this request, it will determine if the job is running and if so the job is killed and the temporary work directories containing the job are removed. If the job has already finished, the temporary work directories are removed.
- The **Run Status** text area on the bottom right of the Submit Job widget group is used to display the current status of a job.

**Server** is a global widget group that has two buttons and one text area. In general terms this widget group allows you to set the current server.

- The server **Set** button allows you to set the current server. When you click on this button, a pull down menu appears containing a list of all of the servers that you have configured on the interface. Note there may be other servers, but if you have not told the interface about them, they will not appear in this pull down menu. Clicking on a server, will cause it to be set as the current server. The current server is displayed in the server name text area under this button. At the bottom of pull down menu is an item **Edit Servers** that can be used to modify your list of servers. Activating this widget will bring up a popup, Chapter 3.1.4, that allows you to modify your current servers and to add new ones if desired. This Server Edit popup is also available under the **Settings/Server Setup** menu.
- The server **Status** button will send a request for a list of jobs currently running on the server. On Linux and Sun systems this request is a simple “ps”. The results of this request are displayed in the Text Viewer at the bottom of the interface.
- The current Server is displayed in the **Server Name** text area under the two button in the Server widget group.

**Model** is a widget group that is specific to the Exponential package. In the exponential package the Model widget group serves three purposes: to set the order of the exponential model to be processed, to indicate if a constant offset is present, and indicate if the number of exponentials is unknown. For a more detailed description of these widgets see the chapters on the exponential packages, Chapters 5 and 6.

**Analysis Options** is a widget group that shows up on many packages. The exact content of this widget group is specific to each package. Here there is a single widget that indicates whether or not the program is to attempt outlier detection. For more on the outlier model and how it is handled in the calculations see Chapter H.

**Reset** will reset all optional settings back to their default values.

**Save** will bring up a popup that allows you to navigate to the location where you want to save the current WorkDir and then to Save the current WorkDir. The **Set** button will save a WorkDir.

### 1.2.3 The Viewers

After a job has been run and retrieved by the interface, the interface unpacks the result of the analysis. After unpacking the run status is set to “Run” and the various viewers located at the bottom of the interface can be used to look at the results of an analysis. These viewers are used to display various kinds of data.

The buttons along the center of the interface activate the various viewers. These Viewers are



used by the interface to display different kinds of data. Because the display requirements for different types of data are very different there are many different viewers. Not all viewers show up on all packages. On the Exponential package, the viewers shown above, there are seven of these viewers, and this is pretty typical of all packages. For more information on these viewers see Chapter 3.4. Here we are just going to briefly list the viewers and note their primary function:

- The **Ascii Data Viewer** is used to display Ascii data. For more information on this viewer see Section 3.4.1.
- The **FID Data Viewer** allows you to look at both the time and frequency domain FID data. Here FID data means spectroscopic FID data. For more information on this viewer see Section 3.4.2.
- The **Image Viewer** is used to display 4dfp images. For more information on this viewer see Section 3.4.2.
- The **Prior Viewer** is used to display and set the prior probabilities used in the Bayesian calculations. For more information on this viewer see Section 3.4.4.
- The **FID Model Viewer** is used to display FID models generated by packages that process FID data. For more on this viewer see Section 3.4.5.

- The **Plot Results Viewer** is used to display the plots associated with an analysis and is the primary method for viewing the results of an analysis. For more on this viewer see Section 3.4.6.
- The **Text Results Viewer** is used to display and print the Ascii files that result from an analysis. For more on the Text Results viewer, see Section 3.4.7.
- Finally the **File Viewer** is used to view the all the files generated by analysis. For more on the Text Results viewer, see Section 3.4.8.

The overview given in this Chapter should give you some indication of what the software can do. The Java interface provides a simple user friendly way of setting up a Bayesian Analysis. After the analysis is set up the interface will automatically ship the analysis to the selected server. The Bayesian Analysis software on that server can run many different types of analysis relevant to NMR in parallel. The interface allows the user to leave an analysis while its running and then come back to that analysis at a later time and simply pick up the analysis from the point they left off. The user can determine the status of a job while its running and then fetch the job when its completed. The interface provides a convenient way of displaying the results of the analysis in graphical form and, finally, allows the user to view and print the outputs from an analysis.

## Chapter 2

# Installing the Software

This Chapter will walk you through the process of setting up a Bayesian software server. The server contains all of the software needed to run the various analysis and once installed, your server is an independent standalone installation that does not communicate with Washington university, except when you download or check for updates. Here are the steps needed to setup a server:

**Pick the machine** that is to act as the server. Lets call this machine “your.server.net.” The server(s) can be a multi-core LinuxPC, either 32 or 64 bit, running CentOS 4.1 or higher, or a Sun system running Solaris 9 or 10.

**Both** Java (6.0 or higher) and javaws must be accessible to “your.server.net.” Note for the LinuxPC uses, the Java shipped with CentOS is OpenJDK and this Java version does not come with javaws. To fix this problem, download and install Sun Java.

**The Fortran and C** compilers are not strictly required to run this software, but the software is much more usable when these are installed. For this reason, I would strongly suggest users install both Fortran and C compilers. The software supports Intel, Gnu and Sun compilers. The Intel compilers cost a few hundred dollars, the Gnu and Sun compilers are free. Note that while the software supports Sun compilers on Solaris, it does not support these compilers on Linux systems.

**Create an account** on “your.server.net” that is to be configured to run the software. This account should be dedicated to running the Bayesian Analysis software and should not be used by anyone for any other purpose. For arguments sake, lets assume this account is named “bayes”, although it can have any name.

**Login** to the “bayes” account on “your.server.net” and configure the path variable so that both Java and javaws are in your path.

**Run** the following command:

```
javaws http://bayes.wustl.edu/ServerSoftware/launch.jnlp
```

When this command executes, you will get the popup shown in Fig. 2.1. This Java application will bring up a configuration window with the software's best guess as to what is needed in the various configuration parameters. Review these settings and correct any that are not set correctly.

**Clicking** the install button will cause the install kit to download the software from bayes.wustl.edu and then install that software in the top level directory of the bayes account. The software runs as user bayes and does not have write permission outside of the bayes account. The compressed tar file that is downloaded is between 35 and 45 megabytes depending your server hardware, so downloading this file could take a few minutes.

**After** the server installation kit completes, it displays a list of commands that must be executed as root. These commands will copy the Apache server configuration files to the appropriate system directory, make the server's home directory and install the start/stop files for the server. These commands can be cut and pasted, so you do not have to retype them.

Your sever should be up and running and you should have a fully functional installation of the Bayesian Analysis software. To check, entering "http://your.server.net:8080" in a web browser should bring up the default apache server page; where your.sever.net is the name of your server, and you should replace 8080 by the port number used in the installation. If the apache server home page appears, everything is working, if not something has gone wrong in the installation and you should contact us for assistance.

You should be able to run the client interface on any machine that has Java and JavaWS installed. We routinely run the interface on Sun Solaris, MacOS, CentOS and WindowsPCs. To run the interface, issue the command:

```
javaws http://your.server.net:8080/Bayes/launch.jnlp
```

from a command prompt, where you must supply your server name and port number. Additionally, this command can be placed in a shortcut to facilitate starting the interface. Running this command should result in the interface being displayed on your client machine. If you wish to create multiple servers, repeat all of the steps including copying the setup commands as root for each of your servers.

Finally, the Bayesian Analysis software does not have an update feature. That is to say, when we distribute new software, one simply reruns the installation procedure to install the updates. So, for example, if you installed the software but did not configure either Fortran or C compilers, then to change this you would simply rerun the installation procedure and select the appropriate compilers. When the installation is done, you system should then support compilers. If you have any questions or comment please contact me: [Email: larry@bayes.wustl.edu](mailto:larry@bayes.wustl.edu)

Figure 2.1: Installation Kit For The Bayesian Analysis Software

**Bayesian Server Software Installation/Update**

File Help

Installation Info

**Bayesian Analysis Of Common NMR Problems  
version 4.10 is being installed**

Host Configuration

Hostname: 192.168.122.1

Host IP: 192.168.122.1

Host OS: CentOS release 6.4 (Final)

Hardware: x86\_64

User: larry

Group: larry

Shell: /bin/csh

Home Dir: /home/larry

Apache Server

CentOS/Sun Ubuntu/Debian

Server Root: /etc/httpd [Set]

Doc. Root: /var/www/html [Set]

HTTPD: /usr/sbin/httpd [Set]

PID File: /etc/httpd/run/httpd8080.pid [Set]

Log Dir: /etc/httpd/logs [Set]

Config File: /etc/httpd/conf/httpd.conf [Set]

Start/Stop File: /etc/rc5.d/K15httpd [Set]

Compilers

Fortran: /opt/intel/composerxe-2011/bin/ifort [Set]

C: /usr/bin/cc [Set]

Server Parameters

Port: 8080

Queue: None

Email:

Use Server Passwords

Email Update Notifications

Setup Apache Server

QUIT INSTALL

Figure 2.1: When the `javaws` command executes, it looks over your computer and makes its best guesses as to how to configure the software. After determining these guesses, this window is displayed so that you can correct and fill in some additional information. After adding compilers, hit the “Install” button to load the software. The install program will download and install the software. When it completes it pops up a window that contains a number of commands that must be run as root. These commands copy the configuration files to their proper location.

ub

# Chapter 3

## the Client Interface

The interface to the Bayesian Analysis software is a Java interface that runs on any machine having Java 6 or higher. Assuming the Bayesian Analysis software has been installed on a server at your site, for arguments sake lets call this machine “your.server.net,” then the client interface can be displayed by issuing:

```
javaws http://your.server.net:8080/Bayes/launch.jnlp
```

where “javaws” is the Java web-start utility and comes with most Java installations, “your.server.net” should be replaced by the server name or IP address, and “8080” should be replaced by the port number used during installation, see Chapter 2 for a description of how to install the software.

When the interface starts it will displays the default start up page shown in Fig. 3.1. This figure is a repeat of the figure shown in Chapter 3, Fig. 1.1 and is repeated here for convenience. The purpose of the start up page is to allow an analysis to be restarted. When the interface exits or changes working directories, the interface saves the current settings in a special Java properties file. When the interface start, it consults this file to determines what the last WorkDir was and how to restart that analysis. If an analysis was saved, the interface displays the messages shown in Fig. 3.1, the lines starting “To restore analysis”. This line contains the name of the package that was being processed, in this case the package name was “AnalyzeImagePixels” and the analysis was saved in a WorkDir named “Given”. If the “Restore Analysis” button is activated then the “Given/AnalyzeImagePixels” analysis will be restored to its previous status. When the interface finishes restoring the analysis, it will function exactly like WorkDir or interface was never exited.

If an analysis is not to be restored, then changing the package will delete the contents of the current WorkDir and configure the WorkDir for the new package. If another analysis is running in a different WorkDir, then changing the current working directory using the “WorkDir” menu will cause the interface to switch to the new WorkDir and the previous analysis will be restored.

Finally, if a completely new WorkDir is needed, then selecting “WorkDir/Edit” will bring up a popup that can create a new WorkDir. After the WorkDir is create, the first thing that must be done is to select a package.

Figure 3.1: The Start Up Window

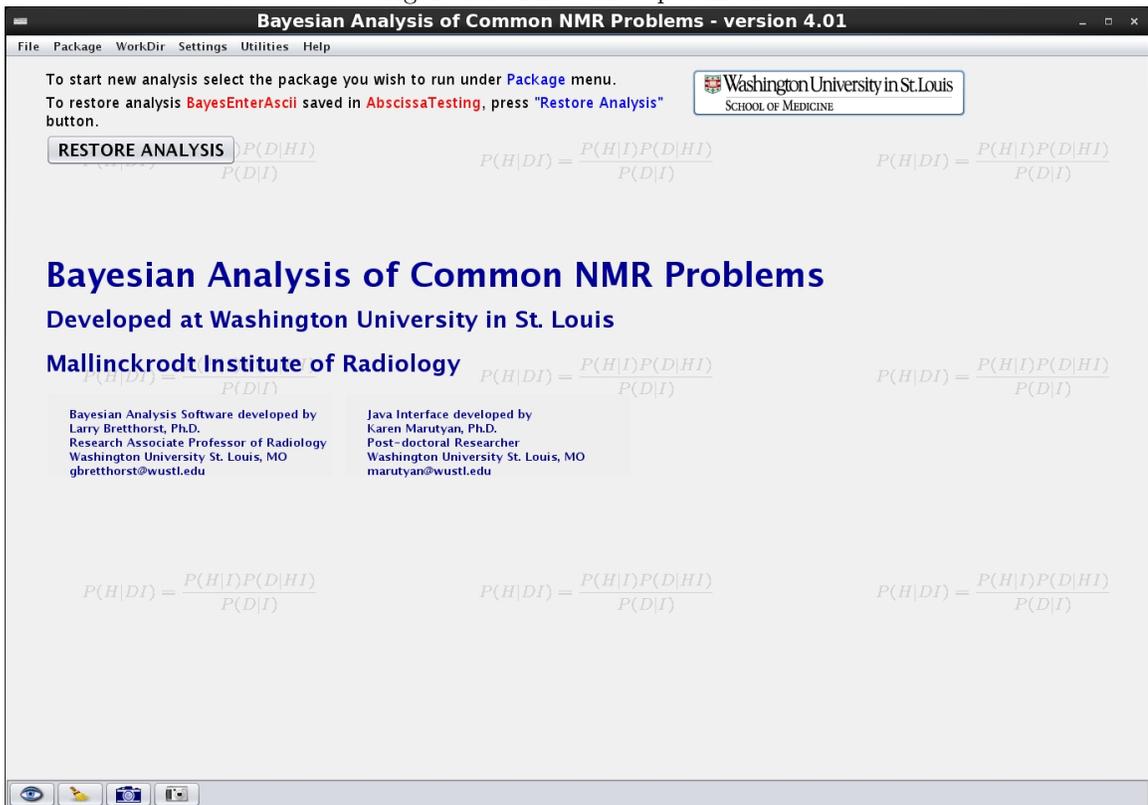


Figure 1.1: The Bayesian Analysis Start up Page allows you to select what functions you wish to perform. For example you might restore an old analysis, change a setting, run one of the utility programs or select a new WorkDir or a new Bayesian Analysis package.

Figure 3.2: The Files Menu

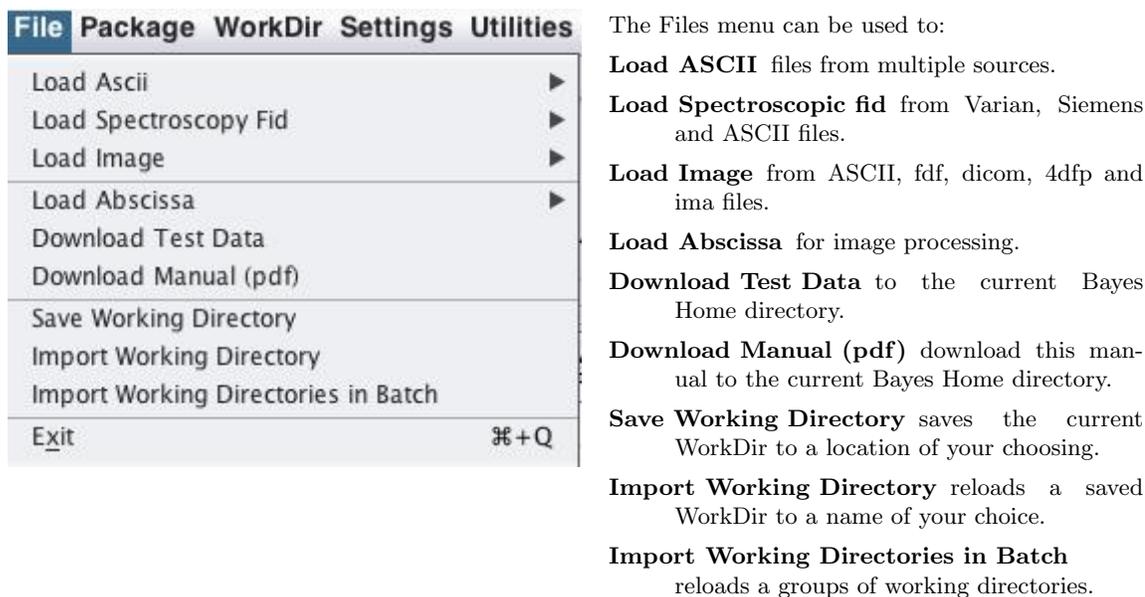


Figure 3.2: When the Files menu is selected, this pull down menu is displayed. Use this menu to load data, download updates to this manual and save and import working directories.

## 3.1 The Global Pull Down Menus

The global pull down menus along the top of the start up page are always present on all package interfaces, not just the start up page. They allow the user to load files, select packages, configure servers, change working directories, set options, etc. Each pull down menu has multiple functions and the following subsections explain these menus in detail and how to go about using them. We will take the menu from left to right starting with perhaps the most complicated menu, the Files menu.

### 3.1.1 the Files menu

The Files menu is a general purpose menu that handle most functions concerning loading images, ASCII data and other types of files into the Bayesian Analysis software. Figure 3.2 shows what this menu looks like when activated. Here is what each selection menu on the Files menu does:

**Load ASCII** selection menu loads ASCII data from either an ASCII file or a Bayes Analyze file. In either case the data is copied/reformatted and saved in the BayesOtherAnalysis directory of the current WorkDir. When multiple ASCII files are loaded the currently selected data file is plotted. The file format for an ASCII file is package specific, i.e., the number of data and abscissa columns required varies with the package. The file format for each package is addressed in the Chapter describing the package. However, the general file format used by

Bayesian ASCII software is described in Section A. After the data is loaded into the WorkDir, the ASCII Data Viewer is then activated and the ASCII data is plotted in the viewer.

The Load ASCII menu has two submenus, one to load a plain ASCII file and one that extracts amplitudes from a Bayes Analyze file. Here is a description of these selection menus:

**File** brings up a navigation popup that navigates to the appropriate ASCII file and then load that file. Note that file is parsed by the interface to determine if it has the correct number of data columns and that it contains only ASCII data. The loaded ASCII file is copied into the current WorkDir and the file is assigned a name using a sequential number. These numbers are unique so multiple ASCII files having the same name can be loaded without conflict. The loaded file is then displayed in the ASCII Data Viewer.

**Bayes Analyze File** is a selection menu that will load the amplitudes of a resonance as a function of some arrayed parameter. These amplitudes are read from the previous run of the Bayes Analyze package. A prompt for the resonances number who's amplitudes are to be loaded will appear. The Bayes Analyze analysis must use a joint analysis, i.e., all fid's are analyzed jointly looking for common frequencies with fid dependent amplitudes. The amplitudes are combined with the arrayed variable from the propar in the "fid" subdirectory in the current WorkDir.

**Load Spectroscopic fid** loads time domain spectroscopic fid data from several sources: Varian fid data, Siemens rda, Siemens Raw data and ASCII Text fid data. When any of these selection menus are activated, they brings up a popup that allows navigation to the appropriate file and then load it. When data are loaded, the data are copied to the "fid" Subdirectory of the current WorkDir and a Varian fid, propar and text file are written into this Subdirectory. If the loaded fid is a Varian fid, the propar is copied from the source directory, otherwise a propar is generated and modified to reflect the number of data values, acquisition time and sweep width of the current data. The data are then Fourier transformed, and the real part of the discrete Fourier transform is displayed using the phasing parameters in the propar.

**Load Image** menu handles the task of loading various kinds of image data into the Bayesian Analysis package. The menu will loads both k-space and image data. The first two selection menu items will read k-space data and then convert the k-space data into images. The remaining menu items all read various types of images and convert them into our internal format. Images are stored in the "Images" Subdirectory of the current WorkDir. No attempt is made to change the name of an image so loading images with the same name will result in replacing the old image. We will briefly mention the various types of data that can be loaded and give a brief description of the file formats.

**Varian k-space fid** will bring up a popup that allows navigation loading of standard Varian k-space image fid. The selected file must be suffixed with ".fid" and a Varian binary file is expected. The binary is copied into the "image.fid" Subdirectory in the current WorkDir. It is then Fourier transformed, phased and three ".img" file are written into the "images" Subdirectory. These three files are named "LoadedImage-Abs.4dfp.img," "LoadedImage-Real.4dfp.img," and "LoadedImage-Imag.4dfp.img" and contain the absolute value, imaginary and real images.

**Text k-space fid** will bring up a popup that allows navigating to and load a text k-space image fid. The selected file must be suffixed with ".fid" and a Varian binary file is

Figure 3.3: The Files/Load Image Submenu

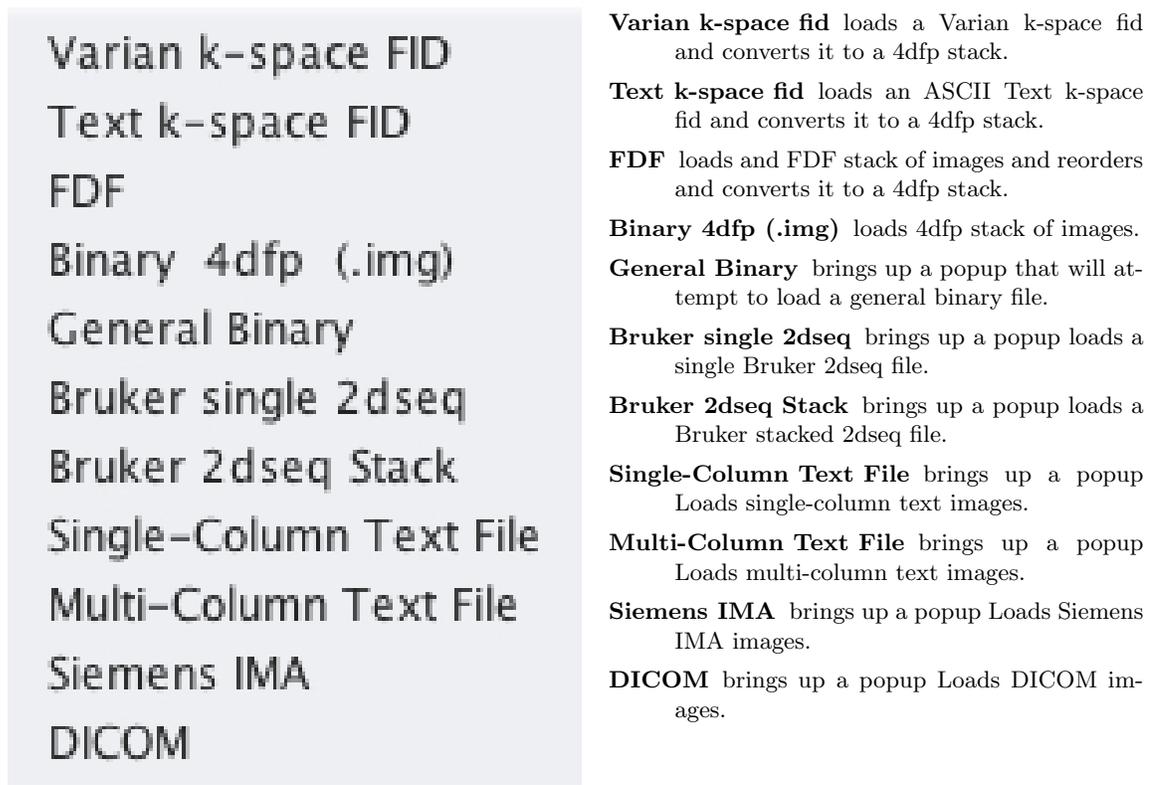


Figure 3.3 When the Files/Load Image selection menu is selected this pull down menu is displayed. It is used to select the type of image data to be loaded. After selecting a data type, a popup will be displayed that will allow navigation to the appropriate loading of the data. In most cases the popup will have a number of configuration parameters that have to enter.

expected. The binary is copied into the “image.fid” Subdirectory in the current WorkDir. It is then Fourier transformed, phased and a three “.img” file are written into the “images” Subdirectory. The image files images are names as discussed in the previous item.

**FDF** will bring up a popup that allows navigating and loading of Varian FDF images. One or more of the FDF files are loaded by the open button. This will copy and reformat the FDF images into a single 4dfp file located in the “images” Subdirectory of the current WorkDir. The input images are ordered so that the displayed images are in the proper slice and array order.

**Binary 4dfp (.img)** will bring up a popup that allows navigating and loading of 4dfp file. The 4dfp file is copied into the images Subdirectory. The 4dfp file type is the internal standard in which all images are stored in the Bayesian Analysis software, so no additional processing is needed. See Appendix G for a description of 4dfp files.

### General Binary

#### Bruker single 2dseq

#### Bruker 2dseq Stack

**Single-Column Text File** will bring up a popup that allows one to navigate to a single-column text file and load it into the images Subdirectory. The single-column text is read by the interface and then parsed into images with the assistance of a popup window. The popup will display the total number lines in the text file and the row, column, slice and array dimensions must be set so that the total pixels is equal to the total lines. Until these dimensions are set correctly the interface will not load the image. Multiple images can be stacked in the file. If so, they can be stacked either in slice or array order and specify which order is used when the data are loaded. The default outer loop, the most slowly varying loop, is the array dimension with slice as the inner loop.

**Multi-Column Text File** will bring up a popup that allows navigating and loading of a Multi-Column text file. Multi-Column text files (images) are read by the interface as series of lines each containing multiple pixels. Each line in the file corresponds to one horizontal line in the displayed image. For MRI data each line corresponds to the phase encode direction. The number of phase encode pixels depends on the number of phase encodes and the zero padd level of the Fourier transform. So if the image has a total of 64 phase encode pixels (including zero padding), then each line in the text file must have 64 phase encode pixels. Additionally, if there are 96 pixels in the vertical (readout) direction, then there must be 96 total lines for each image in the text file. Like single-column text files, multiple images can be stacked in one file and the images can be ordered by either by slice or array element. Also, like single-column text files, when an image is selected, a popup will be displayed and the number of slices, array dimension, and image sizes must be specified.

**Siemens IMA** will bring up a popup that allows navigating to loading of of Siemens IMA images. One or more of the IMA files can be selected and opened. This will copy and reformat the IMA images into a single 4dfp file located in the “images” Subdirectory of the current WorkDir. The order of the images is alphabetical, if the images must be ordered in some special way they must be renamed appropriately.

**DICOM** will bring up a popup that allows navigating to and loading of DICOM images. This will copy and reformat the DICOM images into a single 4dfp file located in the

“images” Subdirectory of the current WorkDir. The interface attempts to order the images internally so that the displayed images are in the proper slice and array order.

**Load Abscissa** is submenus on the Files menu. When selected, it brings up a popup that allows the user to navigate to an to a file and then load it as an Abscissa file. Abscissa files are stored in the “images” Subdirectory and are named “Abscissa.” Abscissa files can be multicolumn ASCII files. The abscissa is used for several purposes, for example it is used to generate ASCII files from image pixel data. Additionally, when an image is processed on a pixel by pixel basis, the model that does the processing must know the abscissa values. For example if one is processing diffusion tensor data, one must know the “B” values. The “B” values are the abscissa, and in this example the abscissa file would be a three column ASCII file. The number of columns in the selected Abscissa file must match the requirements of the current package before the interface will load the file. When the Load Abscissa button is activated, one can select one of two options:

**From File** will bring up a popup that allows navigating to and loading of an Abscissa file.

**From Procpar** will bring up a popup that allows navigating to and selecting a procpar file. The interface will then read the procpar and find the arrayed variable and attempt to construct the abscissa from procpar. If multiple variables are arrayed in the procpar, then a multicolumn Abscissa file is constructed.

**Download Test Data** to the Bayes Home directory. The Bayesian Analysis software ships with a directory containing data that can be used to test the various packages. This data is contained in a file on the server. That file is located in the Bayes user account in a directory named Bayes. The file is named Bayes.test.data.tar.gz. However, the data is not generally accessible to users because the file is a gzip compressed tar file. When selected, the “Download Test Data” submenu, downloads a copy of this file and then uncompress and untar the files. The directory, Bayes.test.data, containing the test data is placed in the Bayes directory in the bayes user account. The downloaded test data, can then be load and used to test the various packages. Inside the Bayes.test.data directory the test data is organized by package.

**Download Manual (pdf)** will download a copy of the user manual to the Bayes Subdirectory in the current Bayes Home directory. This manual is named BayesManual.pdf and is the version of the manual issued with the Bayesian Analysis software installed on the server. Additionally, a web browser can be pointed to

<http://bayes.wustl.edu/Manual/BayesManual.pdf>

and the most recent copy of the manual can be downloaded from the Bayesian Analysis’ home page. Finally, the above link can be activated and the acrobat reader will download the most recent version of the manual from bayes.wustl.edu.

**Save Working Directory** widget allows a WorkDir to be saved. When this widget is activated, a popup is displayed. This popup allows selection of both the WorkDir and the location where the WorkDir is to be saved. Finally, the WorkDir is copied to the specified location. When the WorkDir is copied, the entire contents of the WorkDir are copied, all ASCII files, images, fid’s etc. are copied and saved in the specified location.

**Import Working Directory** allows a saved WorkDir to be reloaded. When the Load Working Directory widget is activated a popup is displayed that allows the select the WorkDir to be reloaded. In a copy statement this is the source location of the files to be copied. After selecting the source WorkDir, you are prompted to enter the name of the WorkDir where the files are to be copied will appear. The source directory is then copied to the Bayes directory in the current Bayes Home using the new WorkDir name as the “to” location in the copy. After reloading the WorkDir, plots, text reports, fid’s, images and ASCII files will have been restored to the same status they were in when the WorkDir was saved.

**Import Working Directories in Batch** imports multiple saved working directories. For example while working on a project where model selection was needed on about 100 different samples. We created a working directory for each sample, ran the model selection and then save the resulting working directories. At a latter time it was necessary to reload all 100 working directories so that we could review and in some cases rerun the analysis.

### 3.1.2 the Packages menu

The packages menu, shown in Fig. 3.4, is used to select a package, a set of programs used to solve some particular problem. The software contains roughly 20 packages and these packages implement various calculations using Bayesian probability theory. The various packages implemented by this software are briefly describe here and a detail description of each package is given in later Chapters.

**Exponential** The Exponential package estimates the decay rate constants and amplitudes of signals known to be decaying exponentially. It does this when the number of the exponentials is known or unknown. In both cases the input to this package can come from ASCII files, from a peak pick or from Bayes Analyze files. In all cases one or more input data sets can be processed. When multiple input data sets are processed, the package looks for exponentially decaying signals that are common to the various data sets, but allowing each exponential to have differing initial conditions in each data set. See Chapter 5 for more on the exponential problem when the model is given, and Chapter 6 when the number of exponentials or the present of a constant offset are unknown.

**Inversion Recovery** The Inversion Recovery package is a special type of exponential analysis that is very common in NMR. In this problem the NMR signal starts at a negative value and decays to a positive value. The inversion recover model differs from an exponential plus a constant model only in that the model is typically formulated so that the two amplitudes represent the initial, time equal to zero, and equilibrium amplitude; thus the amplitudes are linear combinations of the amplitudes that would be estimated by an exponential plus a constant model. As a side note, this package is really a special case of the Enter ASCII package described below. We call these special cases preloaded enter ASCII models because the interface preloads the inversion recover model from the system model directory and thus simplifies what the user must do to run this inversion recovery model. This package can analyze multiple data jointly to look for a common parameters. See Chapter 7 for a description of the inversion recovery package.

**Diffusion Tensor** The Diffusion Tensor package analyzes NMR diffusion measurements using one, two or three diffusion tensor models with or without a constant. These tensor can use either “b”

Figure 3.4: The Packages Menu

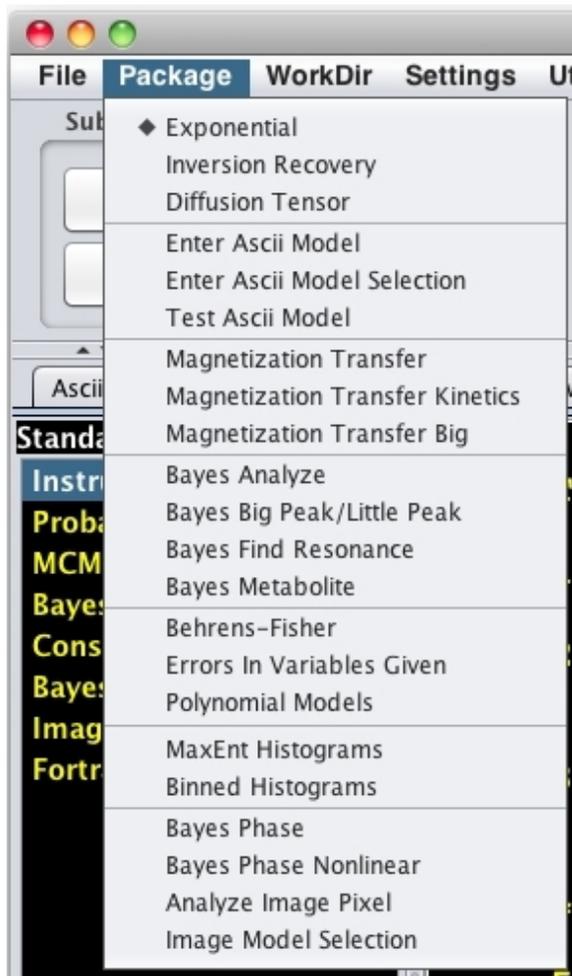


Figure 3.4: When the Package menu is selected this pull down menu is displayed. It is populated with a complete list of all of the packages supported by the Bayesian Analysis software. Selecting a package will cause the interface to display the interface to the selected package and the interface will configure the current working directory for that package. The packages are grouped more or less by the type of data and model processed. For example, Exponential, Inversion Recovery and Diffusion Tensor all process ASCII data and they all process models that are exponential in nature; while things like Bayes Analyze, Big Peak/Little Peak, Find Resonance, and Metabolite analysis all analyze fid data and they all estimate parameters associated with resonances. A brief description of each package is given in this section and a Chapter is devoted to describing the models and in some cases the Bayesian calculations done in each package.

values or “g” (gradient) values for the abscissa and the “b” values can be either 3D vectors or “b” matrices. Thus this package process 18 different diffusion tensor models. Because McMC packages compute the probability for the model using thermodynamic integration, this package has the ability to do some simple model selection. As with most packages multiple ASCII data sets can be analyzed jointly to look for common diffusion tensor parameters. process ASCII Diffusion tensor models, similarly for image model selection. See Chapter 12 for a description of the diffusion tensor model.

**Enter ASCII Model** The Enter ASCII Model package allows the user to define a model and then use Bayesian Probability theory to analyze data using that model. To create a simple model, activate the Fortran/C Code Viewer and then activate the “Edit/Create New Model” button. When this button is activated it will make a copy of the Example.f model, and open it in an editor. This model can be changed, compiled, tested, save and run as needed. In addition to creating a Fortran/C model, the users must create a file that describes the model parameters and the prior probabilities for those parameters. This process is done simiautomatically when the Fortran/C model editor is used. However, If models are edited manually then this file must be created manually. See Chapter 20 for a description of the Fortran/C models and their “params” file.

**Enter ASCII Model Selection** The Enter ASCII Model Selection package utilizes the models generated for Enter ASCII to do model selection. After setting up a number of rival models using Enter ASCII, one can then proceed to this package. Here one can load up to 10 different models and then use this package to compute the posterior probability for the models. The only requirement between the models, is that they must process the same data, so all models must have the same number of data columns, and because ASCII data has the abscissa in the file all models must use the same abscissa. See Chapter 22 for a description of the ASCII Model Selection package.

**Test ASCII Model** The Test ASCII Model model package supports the other packages that use ASCII Models. This package gives one a facility for testing models to ensure they are doing their calculations correctly. This package allows load a model and data associated with that model, and then the Test ASCII Model package will thoroughly test the model by evaluating the model 10,000 times using parameter sampled from the priors. In the process of evaluating the model, the package will catch any arithmetic errors that occur and it will show the abscissa value where the invalid arithmetic occurred. The outputs form the model include a peak posterior probability estimate of the model and plots of the model signal as a function of the parameter samples and plots of the residuals, the difference between the data and the model. See Chapter 21 for a description of the test ASCII model package.

**Magnetization Transfer** The Magnetization Transfer (two sites) package solves the Bloch-McConnell equations to obtain the exchange rate constants for two site magnetization exchange. Input to this package is usually the peak amplitudes or intensities from two inversion recovery time courses where the exchanging peaks in are selectively inverted. The ASCII file used by this package is three column ASCII, one abscissa and the amplitudes of the two exchanging peaks. See Chapter 14 for a description of the Magnetization Transfer package.

**Magnetization Transfer Kinetics** The Magnetization Transfer Kinetics package is a magnetization transfer package that solves the Bloch-McConnell equations at multiple temperatures and

concentrations to derive the entropy and enthalpies of the the exchange process. Input to this package is also three column ASCII, with multiple data sets taken at differing temperature and concentrations. See Chapter 15 for a description of the Magnetization Transfer Kinetics package.

**Big Magnetization Transfer** The Big Magnetization Transfer package solves the magnetization transfer problem when one of the sites can be considered infinite compared to the other. See Chapter 13 for a description of the Big Magnetization Transfer package.

**Bayes Analyze** The Bayes Analyze package is a time domain frequency estimation package that is fully capable of determining the number of resonances in an fid and estimating the resonance parameters. This package can analyze single fid's, or it can run multiple fid's and look for frequencies common to these fid's. Input to this package can come from different sources and appropriate data conversions are carried out when the data are loaded. See Chapter 8.1 for a description of the Bayes Analyze package.

**Big Peak/Little Peak** The Big Peak/Little Peak package analyzes time domain fid data in which there is a single big peak that may be many orders of magnitude larger in intensity (the big peak) than the metabolic peaks (the little peaks) of interest. The Big Peak/Little Peak package solves this problem by treating the big peak as a nuisance and then uses Bayesian probability theory to account for the big peak while simultaneously estimating the frequencies, decay rate constants and amplitudes of the resonances of interest. See Chapter 9 for a description of the Big Peak/Little Peak package.

**Find Resonances** The Find Resonances package analyzes NMR fid data looking for resonances. The program is a model selection program that is attempting to determine the number of resonances in the data and estimate the parameters associated with those resonances. This package uses Markov chain Monte Carlo simulations to determine the posterior probability for the number of resonances in the data. This package essentially solves the same problem as the Bayes Analyze package described above. However, because it uses McMC the calculations are much slower than those in Bayes Analyze, but they are much more thorough; often having much better resolution than Bayes Analyze. See Chapter 11 for a description of the Find Resonance package.

**Metabolite** The Metabolite package analyzes fid data from a number of known samples, for example a C13 fid of Glutamate. The intensity of the Glutamate resonances are related to each other through a metabolic model. This model can be very simple or very complex. Metabolic models can be added to the library of models, but there are no facilities for building these models within the interface. Metabolic models relate the intensity of the resonances in the model to a series of metabolic parameters, typically fractional rates that relates how much of a compound went through a certain chemical reaction. The resonances in a metabolic models are described in a metabolite file and the metabolic model itself is encoded in a FORTRAN or C routine. The metabolic package reads the resonance and the metabolic models and then uses Bayesian probability theory to estimate the metabolic parameters as well as the parameters associated with the resonances, i.e., the frequencies and decay rate constants. See Chapter 10 for a description of the Metabolite package.

**Behrens-Fisher** The Behrens-Fisher package solves the classical medical testing problem: given two experiments that consist of repeated measurements of the same quantity where in the second measurement one has change some experiential parameter determine if the experiments are the same or if they differ. See Chapter 19 for a description of the Behrens-Fisher package and see [On the Difference in Means](#) for a detailed description of the calculations.

**Errors in Variables** The Errors in Variables package solves the errors in variables problem. In this problem one has a data set that has uncertainty in both the X and Y variables. These errors may be know or unknown, so this package solves four different errors in variables problems. In the name the “given” refers to the fact that the program solves this problem given the order of the polynomial to fit. See Chapter 18 for a description of the Errors in Variables package.

**Polynomial** The Polynomial Models package fits polynomials of either a given or an unknown order to the input data. When the order is specified then a polynomial of that order is analyzed using Bayesian probability theory to determine the appropriate coefficients. When the order is specified as unknown, the Bayesian probability theory is used to compute the posterior probability for the order of the polynomials. The input data is two column ASCII and this package do not process multiple data sets. See Chapter 16 for a description of the Polynomial package when the order of the polynomial is given and see Chapter 17 for the calculations when the order of the polynomial is unknown.

**MaxEnt Histograms** The Maximum Entropy Method Of Moments density estimation package, is a ASCII package that takes as its input a two column ASCII file. Column one is just a data point number and column two is a sample from the unknown density function. The program models the density function as a Maximum Entropy moment distribution having an unknown number of Lagrange multipliers. So the parameters are Lagrange multipliers and the unknown number of them. The program does a Markov chain Monte Carlo simulation with simulated annealing where the number of multipliers is one more parameter in the simulation. Outputs include the posterior probability for the number of multipliers, the posterior probabilities for the multipliers, scatter plots and the polynomials used in the calculations. See Chapter 25 for a description of the Maximum Entropy Histograms package.

**Binned Histograms** The Binned Histogram package is a new histogram package. In the previous release of the software, there was a MaxEnt histogram package that infers histograms that are functionally Maximum Entropy moment distributions. As such the program is inferring the moments and the number of moments needed to represent the input samples from unknown density. This procedure works well for compact distribution, but fails badly when the distribution of samples is multimodal. In order to estimate density functions when the samples are multimodal we added a histogram package that infers what can only be called binned histograms. These histograms can represent any distribution, they have error bars on the number of counts in the bins, and the user can indicate if the histograms are to be smoothed or not. See Chapter 23.1 for a description of the Binned Histograms package.

**Linear Phasing** The Linear Phasing package produces linearly phased images. In spin echo MRI most images can be phased (absorption mode images) by calculating two first order phases and one zero order phase. Bayes Phase computes these phases and then applies them to the images. The resulting images are then available for further processing by the Analyze Image

Pixels package. See [Automatic phasing of MR images. Part I: Linearly varying phase](#) for more on this calculation and for a more detailed description of this package see Chapter 26.

**Non-Linear Image Phasing** The Non-Linear phasing package phases images that are varying in a Non-Linear fashion. This package takes as its input the output from the Linear Phasing package. This package can be used to produce absorption mode images for gradient echo MR images or any other image in which the phase is varying in an unpredictable fashion. For more information on this calculation see [Automatic phasing of MR images. Part II: Voxel-wise phase estimation](#) and for a more detailed description of this package see Chapter 27.

**Image Pixels** The Image Pixels package loads a predefined model and then uses that model to analyze images on a pixel by pixel basis. Model can be loaded from the system directory and these predefined models perform a number of common calculations in MRI such as exponential analysis with one or more exponentials with or without a constant, diffusion tensor, Additionally, the users can copy and the edit an example model to create models of his own. These models can be loaded from the users home directory and then used to analyze the image.

The Image Pixels package includes an option for finding the peak of the posterior probability. When this option is selected, a different program is actually run by the package. This program is a searching algorithm that looks for the peak in the posterior probability for the parameters in the model. These peak parameter estimates are then used to generate maps of the various parameters appearing in the model. Because this program is a searching routine rather than an MCMC routine, it is very fast and can give good results using any ASCII model in a fraction of the time needed to run the Markov chain Monte Carlo simulations. See Chapter 28 for a description of the Image Pixel package.

**Image Pixel Model Selection** The Image Pixels Model Selection package extends the concepts in Analyze Image Pixels to model selection. In this package one can load a number of different models and then use Bayesian probability theory to determine which model best accounts for the data. The models in use here are the same models mentioned in both Analyze Image Pixels and the Enter ASCII packages. However, here because the models can have different parameterizations, the output images are constructed from the derived parameters. See Chapter 29 for a description of the Image Pixel Model Selection package.

### 3.1.3 the WorkDir menu

Working directories are directories that are used to run, configure and store analysis in the Bayesian Analysis Software package. They are physically located in the current Bayes Home directory. The default location of the Bayes Home directory is the user home directory and the default name of the Bayes Home directory is “Bayes”. The name and location of the Bayes Home directory can be configured using the Settings/preferences popup and multiple Home directories are allowed. The working directory menu, called WorkDir, is generated on the fly and contains a list of the working directories in the current Bayes Home directory. Additionally, the WorkDir menu contains one fixed menu item named “Edit”. The WorkDir menu is a pull down menu that allows the user to manage working directories, Fig. 3.1.3. The top part of this menu will list all of the working directories in the current Bayes Home directory. The name of the working directories are assigned by the user and the name can be descriptive of the type of analysis being done in that directory. For example, in Fig. 3.1.3 all of the working directories have names that indicate of project the user was working on.

Figure 3.5: The Working Directory Menu

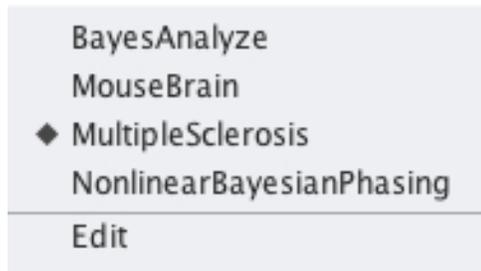


Figure 3.5: When the WorkDir menu is selected this pull down menu is displayed. It contains a list of all of the current Working directories. The contents of this menu varies depending on the number of working directories and the names that have assigned. By selecting one of menu items, the interface will join that working directory and if an analysis is present it will restore that analysis to its previous status.

When a working directory is selected, the interface will save the status of the current WorkDir and then change into the selected working directory and restore that working directory to its previous status.

The last entry on this menu is the “Edit” button. When activated, the edit button will bring up the popup shown in Fig. 3.6. that allows one to modify and manage working directories. Along the left-hand side of this menu is a list of all of the working directories. As noted, these are the names of the working directories as defined by the user. By clicking on these items, the status of the working directory is displayed. Status information includes, the package that is loaded, whether or not the package has been run, and information about the files, ASCII, fid, and Images that have been loaded. Additionally, information about the server that is selected in this WorkDir is displayed. Finally, using the buttons along the button of the WorkDir manager, a new WorkDir can be created. To create a WorkDir, simply enter the name of the directory in the text area in front of the “New” button. and click the new button to create the directory. To delete a WorkDir, select the directory to be delete, and activate the delete directory button. Finally, to load a WorkDir, select the directory to be loaded, and activate the load directory button.

### 3.1.4 the Settings menu

The settings menu is shown in Fig. 3.7. This menu allows one to configure the interface to make use of the operating environment and to control the Markov chain Monte Carlo simulations. Set the window size and to set so user preferences. Here is a description of the this Settings menu:

**the McMC Parameters submenu** Many, indeed, most of the Bayesian Analysis packages use Markov chains to approximate the joint posterior probability for the parameters appearing in the model. This is done by using the Markov chain to draw samples from the joint posterior probability for of the parameters. From these samples, Monte Carlo integration can be used to approximate the posterior probability for each parameter appearing in the model. The number of Markov chain, the number of samples gathered and the annealing schedule are things that the user can control, i.e., configure. Activating the McMC parameters widget brings up the popup shown in Fig. 3.8. This popup allows number of Markov chain Monte Carlo simulations that are to be run concurrently or in parallel to be set: Concurrently if only a single processor is available, and in parallel if multiple processors are available. Additionally, the number of McMC repetitions can be set, and thus the number or samples gathered for use in computing mean and standard deviations parameter estimates. The number of samples = number of

Figure 3.6: The Working Directory Information Popup

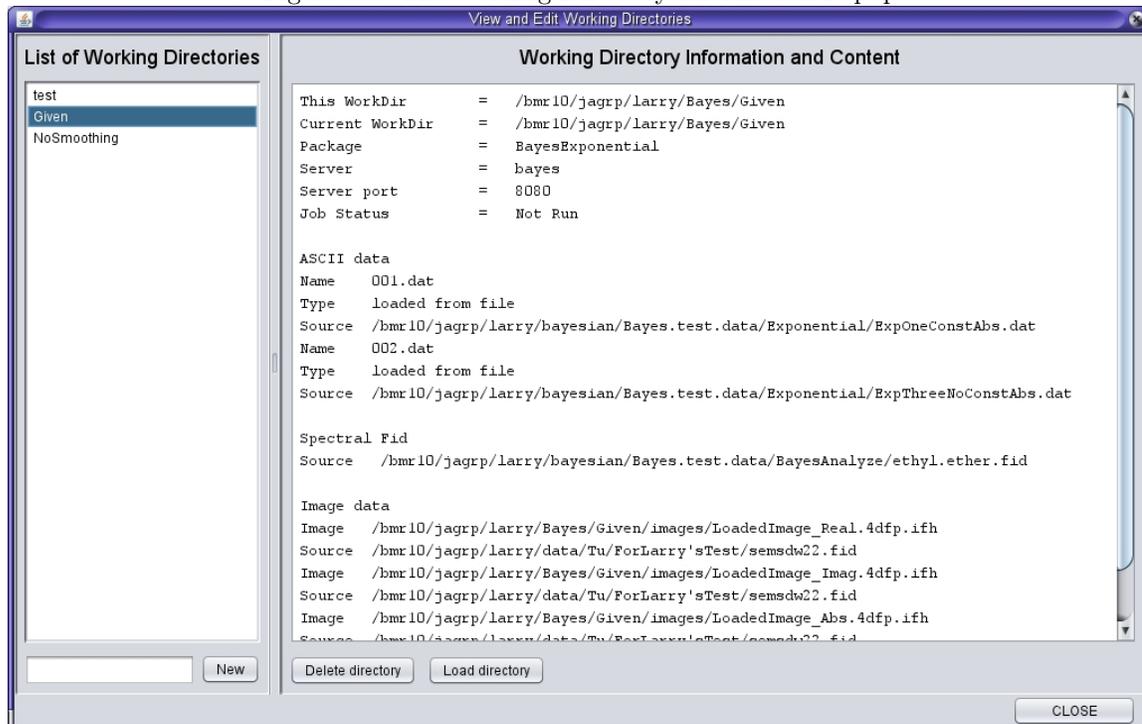


Figure 3.6: When the WorkDir Manager is selected this popup window is displayed. Along the left-hand side is the list of the working directories. By clicking on these working directories, the current status of each directory can be viewed and by using the buttons at the bottom working directories can be created, loaded or deleted.

Figure 3.7: The Settings Pull Down Menu

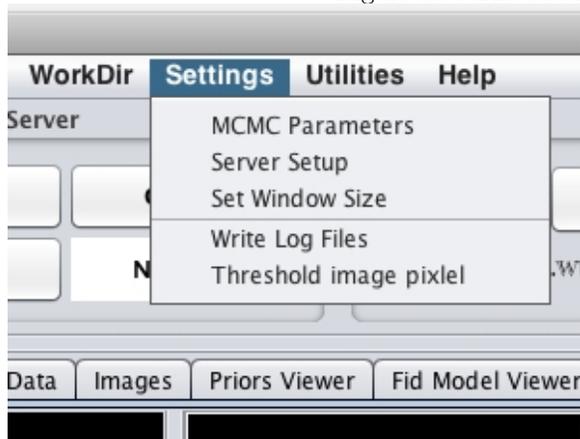


Figure 3.7: The Settings menu is a pull down that allows one to configure and control a number of important features of the interface. The two most important features concern the MCMC parameters and the Server Setup. When the MCMC Parameters menu is activated, it brings up a popup that allows the number of simulations, repeats and the minimum number of annealing steps used in the Markov chain Monte Carlo simulations to be set. When the Server Setup is activated, the resulting popup allows one to add, remove and configure servers. Finally, the preferences button will bring up a popup that allows the configuration some interface parameters, like for example the location of the Bayes Home directory.

Figure 3.8: The McMC Parameters Popup



Figure 3.8 The McMC Settings menu is a popup that set the number of Markov chain Monte Carlo simulations that run concurrently or in parallel: Concurrently if only a single processor is available, and in parallel if multiple processors are available. Additionally, the number of McMC repetitions can be set, and thus the number or samples gathered for use in computing mean and standard deviations parameter estimates. The number of samples = number of repeats times number of simulations. Finally, the minimum number of annealing steps to take during the the simulated annealing phase can be set. For more on how the McMC is used in the Bayesian Analysis software, see Section B.

repeats times number of simulations. Finally, the minimum number of annealing steps to take during the the simulated annealing phase can be set. For more on how the McMC simulations are run see Section B.

**the Server Setup submenu** Activating, the Settings/Server Setup menu will bring up the popup shown in Fig. 3.9. The Settings “Server Setup” menu is a popup that allows servers to be add, delete and modify. To select a server simply click on the server name. After selecting a server, that server becomes the current server and any job submitted will be sent to the selected server. Note that servers can also be selected by activating Server/Set button on all package interfaces. When servers are selected on the Settings/Server Setup popup, the server name, port, Bayes user account, etc. are displayed. Any field except the server name and port number can be modified. However, modifying anything other then the user name and email preferences is not advised. Indeed, modifying the processing account, password settings or queue name will likely result in nonfunctional server configuration.

The Settings “Server Setup” popup has four buttons that can be used to add, delete, configure and display server information.

**Add Server** will bring up a popup in which the server name and port number of the server can be entered. After entering the server name and port, most of the server information can be configured using the Auto Config Server button. However, email preferences and user name may have to be set manually.

**Remove Server** will delete the currently selected server from the list of servers. Note there is no prompt to ask if a server is to be removed.

**View Server Installation Info** will bring up a popup that shows more information about the server. In will show the date the software was installed, the software version, the Bayes user account, port, compiler information including the path to the compilers, number of

Figure 3.9: The Edit Server Popup



Figure 3.9: The Server/Server Setup menu is a popup that allows one to add, delete and modify server settings. To select a server simply click on the server name. Any field except the server name and port number can be modified. Servers can be added using the Add Server button. Servers can be removed by activating the remove server button. Finally, the View Server Installation Info button will bring up a popup that lists all of the installation information available on the server.

CPU, whether or not passwords are in use and information about who the Bayesian Analysis administrator is.

**Auto Configure Server** fetches a configuration file from the server. It then uses this configuration to set the processing account, passwords and number of CPU on the Server Setup popup.

**Set Window Size** will bring up a popup that allows the size of the interface windows to be set. Normally one would set the size of the interface by using a cursor to stretch/contract the window as needed. However, sometimes use of a cursor is difficult depending on the type of terminal in use. This menu allows the size of the window to be set without using a mouse.

**Preferences** The preference widget will bring up a popup that allows you to configure some preferences. The main settings are for the location of the Bayes Home directory. You can uninstall the Java interface from you server. You can tell the interface what output format to write screen captures in. Finally, there are a couple of widgets that indicate if jobs are to be deleted from the servers. This last function is mostly used by us in debugging software. It could happen that while trying to diagnosis a problem, we need for the job to remain on the server so we can see what happened to it. Normally, completed jobs, i.e., any job that is no longer active on the server are removed as soon as the get job widget is activated.

### 3.1.5 the Utilities menu

The utilities menu allows the user to run a number of utilities. These utilities can monitor the memory usage, display some information about Java and its installation and finally determine if a new version of the software is available. Here is a more detailed description of these utilities: There are three utilities that can be run by the user:

**Memory Monitor** will activate a Java Memory monitor. Java, for whatever reason, normally can only access 1GB of memory. Sometimes, we have found that this is not enough to hold large images and applications. The memory monitor can be used to monitor memory usage. If 1GB of memory is not enough, it is possible to get 2GB of memory. However, the launch.jnlp files on the server must be modified to do this. If assistance is needed in modifying these files, please contact.

**System Info** will display information about the current installation on the client machine. This information could be important to us when trying to help diagnosis problems. The displayed information includes: the user name, the user home directory, the architecture, OS, Os version, file separator and the version of Java in use. In case of problems all of this information could be useful in figuring out what is going on.

**Software Update** will bring up a popup showing all of the servers currently have defined and it will indicate if each server has the most recent version of the software installed on it. So the Software Update utility check with us here at Washington University to find out what the current version of the software is and it then checks each of the servers to see if they are up-to-date. If the software is not up-to-date log into each out-of-date server as the bayes user and then update, i.e. reinstall, the software.

### 3.1.6 the Help menu

The Help menu, not shown, will provide help concerning the current release of the software. There are four different types of help available:

**Release Notes** will activate a web browser and download a page that describes the changes in the current version of the Bayesian Analysis Software. Those release notes contain links to the main BayesianAnalysis home page and a chain of links that will describe in the previous releases.

**Online Manual** will download the current version of the manual and the display that manual in the Acrobat reader.

**Bayes Analysis Home Page** will load the home page from the BayesianAnalysis.wustl.edu web site and bring that page up in the default web browser. That page contains a description of the software as well as the release notes for the current and previous versions of the software.

**Contact Us** bring up an email client with my email address in it so that questions can be directly addressed to me.

Figure 3.10: The Submit Job Widgets



**Run** will submit a job to the indicated server.

**Cancel** will cancel a submitted job and remove all files pertaining to a job.

**Get Job** will fetch the current job or fetch the status of the current job.

**Status Label** contains the status of the current job.

Figure 3.10: The Submit Job To Server widget group is used to control the expectation of a job. You can use it to Run a job, fetch jobs from the server (Get Job), cancel a job and the box showing “Not Run” is a status widget that gives you the current status of a job.

## 3.2 The Submit Job To Server area

Just below the global pull down menus, there are a number of widget groups that are used to configure a package. These widget groups are different for different packages. And details on a package specific widget group, consult the chapter on that package. However, there are two widget groups that are global in the sense that they occur on all packages. In the next two Subsections we are going to describe these widget groups and their function.

The first of these widget groups is called the “Submit Job To Server” widget group and this widget group is shown in Fig. 3.10. Below the global menu is an area that is used to configure a package.

**Run** will first check to see if all the required elements are set for this particular job. If everything is properly configured it will create a tar file of the current WorkDir and send it to the server shown in the server widget group. When the job arrives at the server, it is untared and then the requested programs are run. After the job arrives at the server, interface set the status to either Active or Queued and it will lock most of the widgets on a package. The user is free to join another analysis or wait for the analysis to finish.

**Cancel** will cancel a submitted job and remove all files pertaining to that job from the server even if the job is completed. When finished canceling the job, the interface will set the current status to either “Not Run” or “error” if the job could not be canceled and or removed from the server.

**Get Job** performs two main tasks, when activated it first checks the current status of a job and depending on the status it either fetches the job or it fetches and displays the accepted report.

**Text Label** contains the status of the current job. Note this field is only updated any when one of the three Submit Job to Server buttons are activated. To obtain the current status of a job, activate the “Get Job” button.

Figure 3.11: The Server Widgets Group



**Set** will bring up a popup window containing a selection list of the current servers. Selecting one of these servers will change the current server. The Server Name will be changed to reflect the selection. Note, selecting the “Edit” button in the server list will bring up the Server Configuration popup discussed in Subsection 3.1.4.

**Status** will send a request to the selected server asking for its current CPU load. This system load is displayed using “ps” on most Linux systems.

**Server Name** contains the name of the currently selected server.

Figure 3.11: The Server widget group consists of two buttons and one text string. The widgets Set the current server, display the Status of the current server and the text string contains the name of the current server.

### 3.3 The Server area

The “Server” widgets group configures and controls server. This widget group is shown in Fig. 3.11. In general terms this widget group allows the select and configuration of servers. The server widgets group

- The server “Set” button allows the current server to be selected. When this button is activated, a pull down menu appears containing a list of all of the servers. Clicking on a server, will cause it to be set as the current server. The current server is displayed in the server name text area under this button. At the bottom of pull down menu is an item “Edit Servers” that can be used to modify the list of servers. Activating this widget will bring up a popup, Chapter 3.1.4, that allows servers to be add, deleted and modified as desired. This Server Edit popup is also available under the “Settings/Server Setup” menu.
- The server “Status” button will send a request for a list of jobs currently running on the server. On Linux and Sun systems this request is a simple “ps”. The results of this request are displayed in the Text Viewer at the bottom of the interface.
- The current Server is displayed in the “Server Name” text area under the two button in the Server widget group.

### 3.4 Interface Viewers

Just below the widget groups is a set of buttons that activate various viewers. These buttons start on the left in Fig. 1.2 with Ascii Data Viewer and end on the right with File Data Viewer. Each

viewer is used to look at a given type of data. On the Exponential package there are seven of these viewers, and this is pretty typical of all packages.

### 3.4.1 the Ascii Data Viewer

Ascii data can be loaded and viewed in all packages, even packages that analyze fid and image data. The Ascii Data Viewer, shown in Fig. 3.12, is used to display this data. To load an Ascii data select the “File/Load Ascii/File” submenu item. When activated this widget will bring up a popup file loading widget. Navigate to the desired file and then select and load the file. Please note that there are some rather complicated rules concerning what Ascii files can be loaded in a given package. To give one example in the exponential package only load two-column Ascii files can be loaded, while in the magnetization transfer packages only load three column ASCII file can be loaded. Additionally, files suitable for the the magnetization transfer packages will not load in the exponential packages and vice versa, i.e., the file loading popup knows how many abscissa and data columns are required for a given package/user define model. For the exact requirements for each package, see the Chapter on that package. When an Ascii data set is loaded the data is copied into the BayesOtherAnalysis directory of the current WorkDir and the data set is renamed as 001.dat, 002.dat, etc. where the assigned number is just the number of the loaded Ascii data set. The data sets are renamed with unique names to prevent name collision problems. The original name, size, number of columns are stored in a file named 001.afh, 002.aft, etc. These Ascii file header files, the “afh” files, are used to display information about a Ascii file. When an Ascii data set is successfully loaded the Ascii Data Viewer, Fig. 3.12, is automatically activated. Additionally, it can be activated by clicking the Ascii Data Viewer button. When activated a list of all loaded Ascii data files is shown on the left-hand side of the viewer. When a left-hand file name is selected by clicking on it, the selected data set is plotted on the right-hand side viewer. If the data are multicolumn data, for example complex data, the plot will have multiple traces on it.

This viewer responds to a right-mouse click in both parts of the viewer. In the file list area, the left-hand side, the right mouse click shows a submenu that allows file deletion, displays information about the selected file and allows one to view the Ascii file as text. In the plotting area, a right mouse click brings up a submenu that allows one to configure the plot. For example a right-mouse click to can be used to change the headings, axis labels and to save or print the plot.

On the left-hand side at the bottom there are two widgets, one labeled “Delete” and one labeled “?” that delete the selected file or show information about the selected file. These functions are redundant with the functions available using a right-mouse click.

Finally, on the plotted area, a left-mouse click and dragging the cursor down and to the right will select, highlight, a region. When the cursor is released, the highlighted are is expanded. To return to the full plot, left-mouse click and drag the cursor up and to the left will restore the plot.

### 3.4.2 the fid Data Viewer

The Fid Data Viewer, shown in Fig. 3.13, is activated whenever fid data is loaded from the Files menu, or when the Fid Data Viewer is selected. This viewer allows one to look at both the time and frequency domain fid data. When an fid is loaded, the fid is copied into the “fid” directory in the current WorkDir and written in Varian format. The data is then Fourier transformed, the Fid Data Viewer is activated, the spectral data is phased and plotted. Additionally, like the “afh” file

Figure 3.12: The Ascii Data Viewer

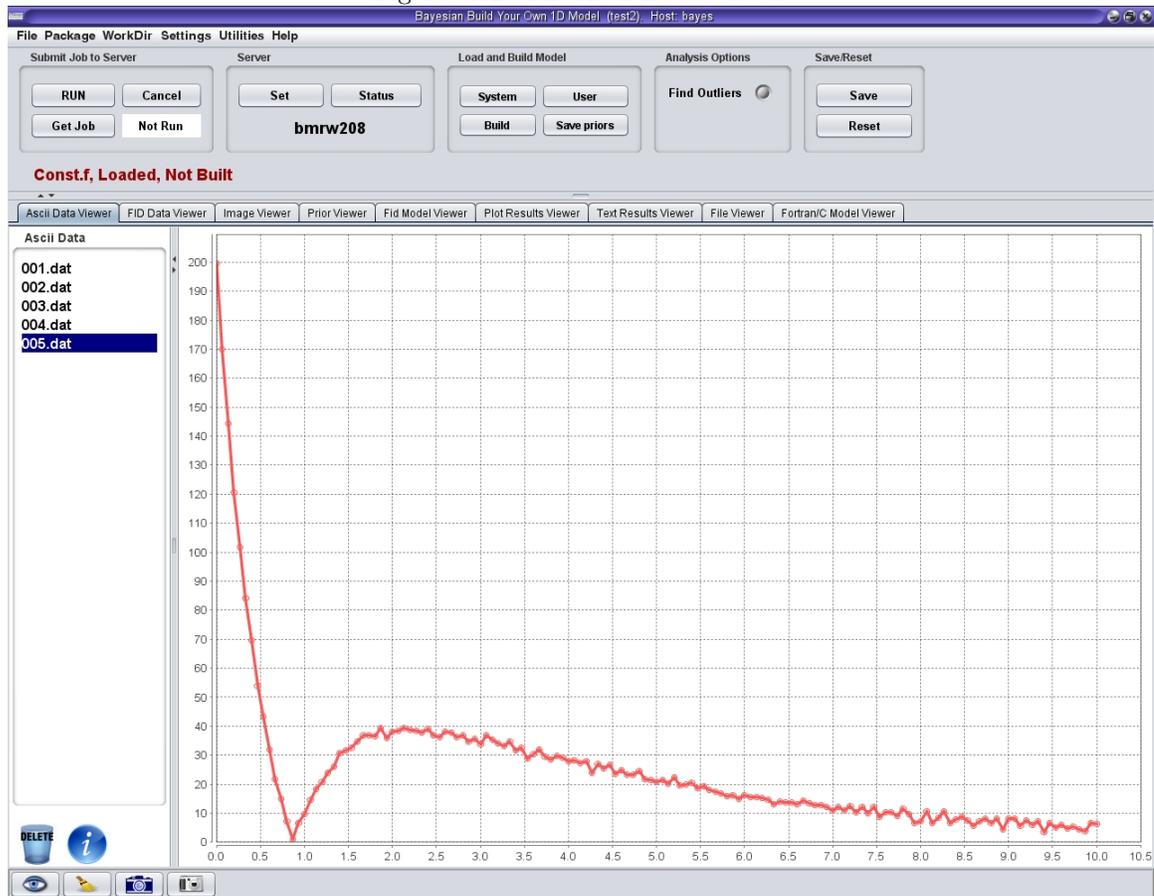


Figure 3.12: The Ascii Data Viewer is used to display Ascii files. To change files one clicks on the name of the file to be viewed. The “Delete” button will delete the selected file and the italics “*i*” button in the circle will display everything known about the file. This button is redundant with a right mouse and selecting show info.

Figure 3.13: The Fid Data Viewer

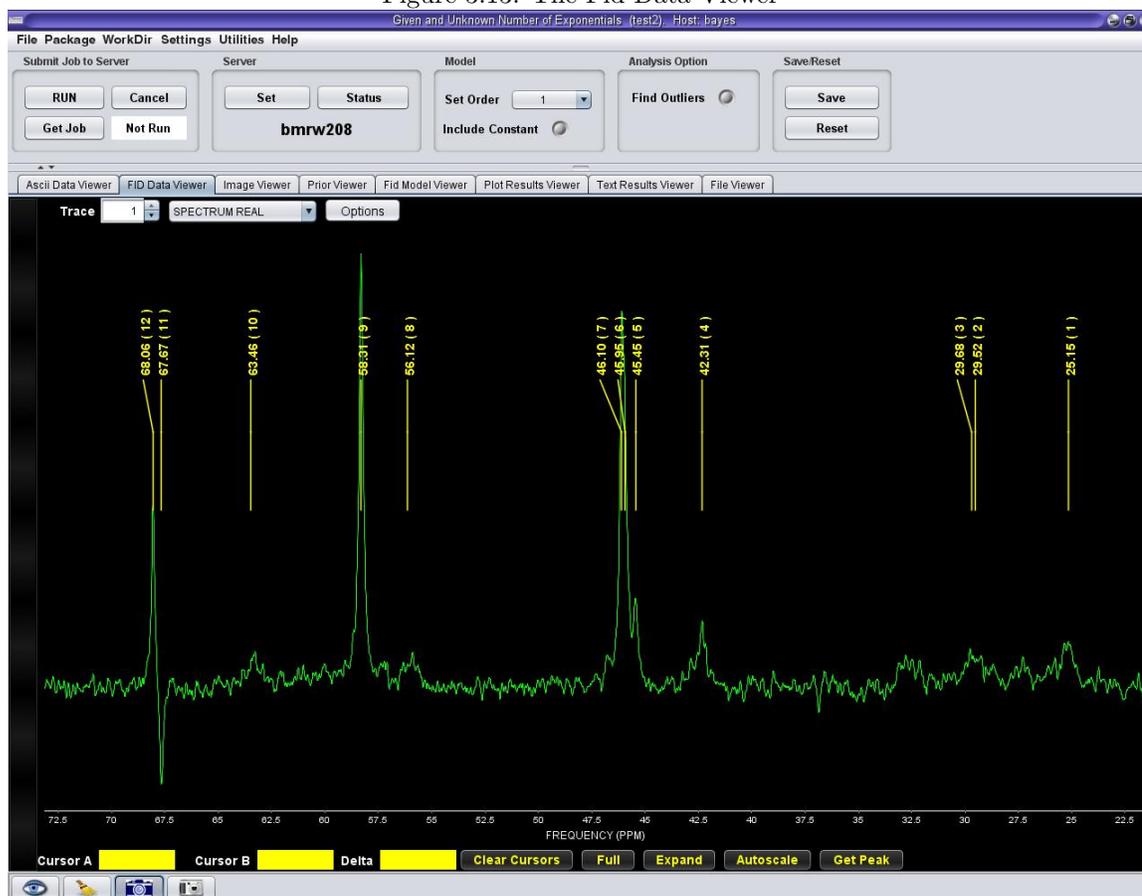


Figure 3.13: The Fid Data Viewer is used to display fid files. When activated the spectrum of the currently loaded fid is displayed. If no fid data are loaded an empty viewer is display. This viewer can be used to zoom in on resonances, change the scale, phase and many other functions. See the text for an extensive discussion of this viewer.

Figure 3.14: Fid Data Display Type

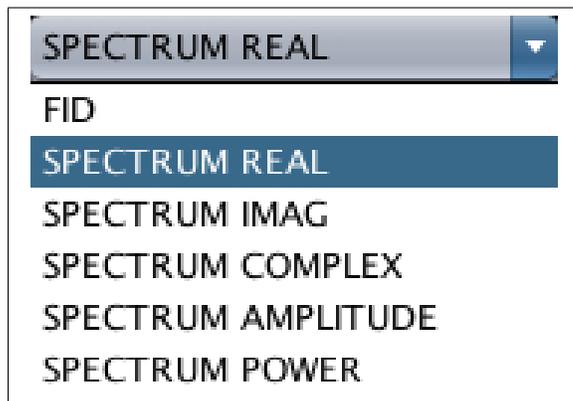


Figure 3.14: The Fid Data Display Type is a pull down menu which displays the current setting in the widget. The default setting is “SPECTRUM REAL,” however there are a number of other types of display’s: for example, absolute value, power spectrum, etc. The real or absorption mode spectrum is the default.

an “ffh” file is written that contains information about the fid. This “ffh” file is displayed whenever information about the currently loaded fid is requested.

The Fid Data Viewer uses left, right and center mouse clicks. The left and right mouse clicks are used to set the locations of a left and right cursor. These cursors are displayed in Fig 3.13 as the two vertical red lines. When set the frequencies of these cursors are shown in in at the bottom of the viewer as “Cursor A” the left cursor and “Cursor B” the right cursor. The difference between these cursors is shown in the “Delta” display. Note the units used in these three display areas are the same as the units on the displayed frequency axis. When the left and right cursors are displayed, the “Expand” button can be used to expand the selected area. The “Full” button will display the full spectrum and the “Clear Cursors” will remove the cursors from the display.

The vertical scale on the display can be adjusted automatically using the “Autoscale” button or it can be adjusted manually. The center mouse button is used to adjust the vertical scale manually. If cursor is placed anywhere in spectrum display above the axis and hit the center mouse button, the display vertical scale is adjusted upward. If cursor is placed below the axis and hit the center mouse button the vertical scale is reduced.

In addition to adjusting the vertical scale, the position of the spectrum in the viewer can be changed. The left-hand part of this viewer has a vertical strip that is shaded a slightly different color than the black of the viewing area. This area can be used to adjust the position of the display vertically. So if the base-line of the spectrum is too high or too low place the cursor in this shaded region and then hold down the center mouse button and the display can be moved up or down.

The type of display can be changed on the Fid Data Viewer. The top-center pull down menu that usually reads “SPECTRUM REAL” can be used to change the type of display. When this pull down menu is extended the menu shown in Fig. 3.14 is displayed. Most of the options are self explanatory and we give no further explanations of this viewer here except to note that all of the cursor and mouse functions work on the different data types.

If a peak is expanded and the “Get Peak” button is activated, the amplitudes of the peaks will be extracted from the current cursor position. The interface will attempt to combine these peak amplitudes with any arrayed variable from the fid procpair to produce an Ascii data set that has the appropriate axis. This data set is assigned a new Ascii data set number, and the Ascii Data Viewer is activated to display the data.

The Options widget is the most complicated widget on the Fid Data Viewer. Its functions are extremely varied and describing all of the functions hidden under the “Options” menu would be very difficult. Here we are going to list these options and give some pointers on what they do. First the expanded options menu is shown in Fig. 3.15. Here is a brief explanation of the various options:

**Data Info** Brings up a popup window that displays information about the currently loaded fid. This information includes things like the original source fid name. The current weighting, the units in use, the phase, the reference and a number of other data items. Its basically a dump of all of the relevant parameters concerning this fid.

**Save As Varian fid** this functions saves the currently loaded fid as a Varian binary fid file and it does this regardless of the input format of the data. So an input data set could have been an Ascii fid and this save command will save the data as a Varian fid. Effectively translating the Ascii data to an fid.

**Save As Text** Saves the currently loaded fid as a two column Text file: (Real, Imaginary).

**Show Plotted Data** brings up the current plot in a popup window that can be viewed, printed and saved.

**Clear Data** will remove the current fid, that is to say the files: fid, text and procar will be removed from the current WorkDir/fid directory.

**Apply Phasing** will bring up a popup window that allows the phase of the currently displayed fid to be set. This popup contains two sliders that allow the zero and first order phase to be set. The displayed fid can be zoomed using the procedures described earlier in Section 3.4.1, so while the displayed fid is not initially zoomed before phasing it can zoom as needed. When the phased has been set, hitting the “Phase fid” button will set the phase on all traces in the data. The popup contains a “Trace” button that allows the displayed trace to be changed and it allows the type of data shown to be changed. So for example one could phase the imaginary part of the spectrum if desired. Finally the popup also allows the phase to be “Reset” to the loaded values.

**Set Regions** Brings up a popup that displays the currently set regions. Regions, low to high frequency intervals, are used in the Bayes Analyze Regions report to calculate the total intensity in a given set of regions. To use this popup simply set a low-high region using the left and right cursors. When a region is set, hit the “Mark” button to add the region to the regions file. Note the “Delete” button can be used to remove a region and the “Close” button will save the regions file. The regions file is save in the current WorkDir in the BayesAnalyzeFiles Subdirectory. This newly created regions file will be used in the regions report the next time Bayes Analyze package is run.

**Set Fn** is a pull down menu that allows the size of the Fourier transform to be set. The pull down menu contains powers of two from 1K up to 256K that can be used in the calculations.

**Set Reference** brings up a popup that allows the current reference frequency to be set. At the bottom of this popup are three buttons, “Set Left Most Frequency to Zero”, “Set Right Most Frequency to Zero”, and “Set Center Frequency to Zero” that are preprogrammed common reference schemes. To set the reference to an arbitrary value, first position the cursor at the

Figure 3.15: Fid Data Options Menu

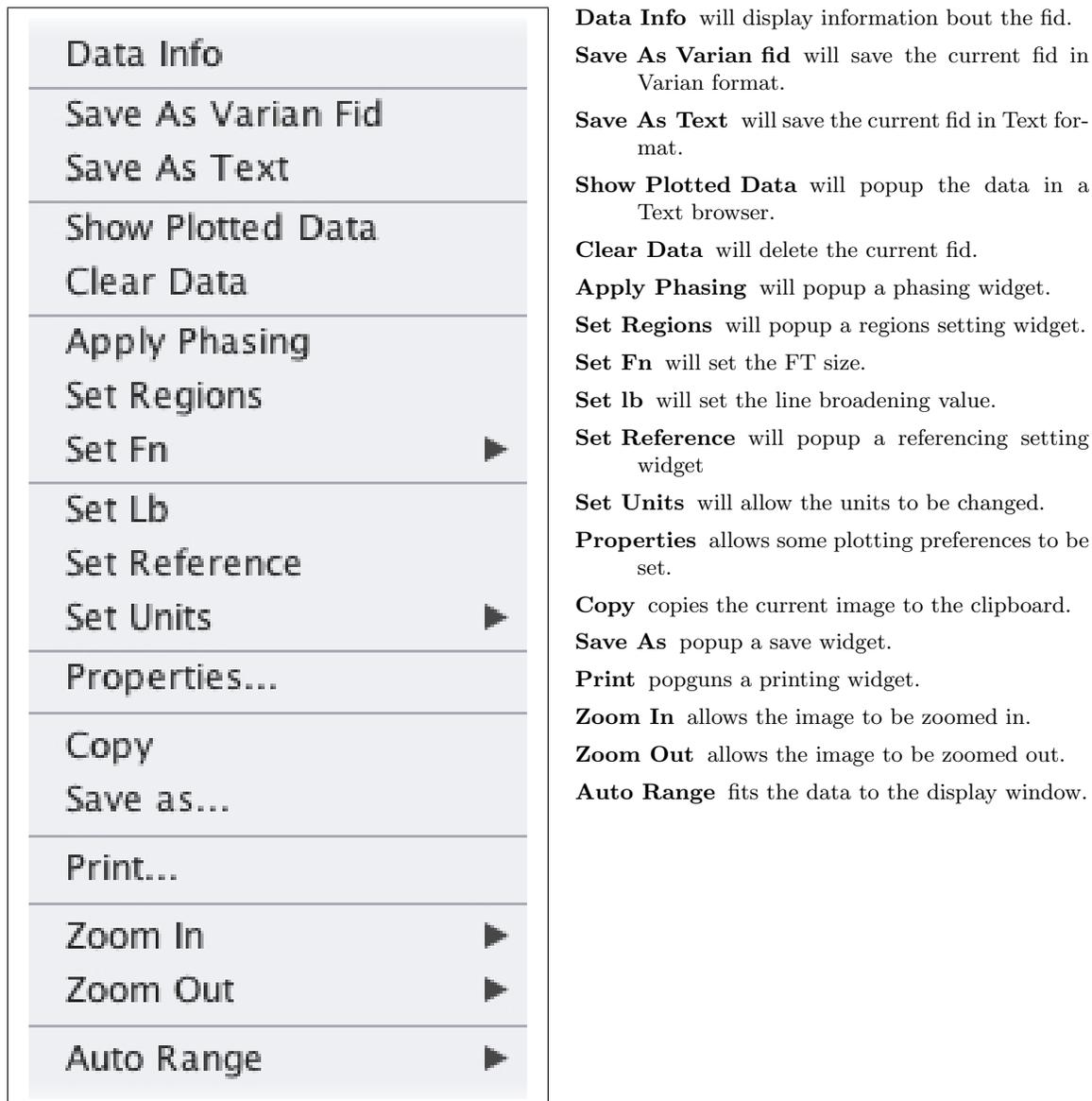


Figure 3.15: When the Options menu is selected from the fid display, this pull down menu is displayed. It performs various optional tasks concerning the Fid Data Viewer. For example the size of the Fourier transform, the weighting or reference on the Fourier transform can be set, etc.

point in the spectrum where the reference is to be assigned. Second type into the “New Value” entry box the value of the reference to. Finally, hit the “Set” button to set the reference.

**Units** is a selection menu that allows the units, “Hertz” or “PPM”, to be set.

**Properties** allows the axis and labels on the Fid Data Viewer to be set. This can be useful when making a graphic that is to be used in a publication.

**Copy** places a copy of the current graphics window and places it on the clip-board. On Windows and Mac machines this makes the graphics available for plotting and use in papers etc.

**Save As** will bring up a popup that allows navigation to the directory where the current graphics is to be saved. Enter the name of the file in the “File Name” box. Activate the “Save” button to save a “png” copy of the graphics.

**Print** will bring up a popup that configures the print job. After configuring the printer, the “OK” button will print the graphics.

**Zoom In** will zoom the axis in. There are options for zooming either both axis or either axis separately.

**Zoom Out** will zoom the axis out. There are options for zooming either both axis or either axis separately.

**Auto Range** will automatically set the range to view the entire spectrum in both the horizontal and vertical domains.

### 3.4.3 Image Viewer

Image data can be loaded into any package using the “File/Load Image” menu. As explained in Section 3.1.1, many different types of image can be loaded and these images are converted into 4dfp images and stored in the image Subdirectory. A 4dfp image consists of multiple binary images that are stacked by slice and element number. Multiple images can be loaded provided they have unique names. If the names are not unique then the current image replaces the previously loaded image. Images are used as input to several packages and can be viewed using the Image Viewer. The Image Viewer is shown in Fig. 3.16. In general terms this viewer consists of four parts: a list of images under the “Image List” label, two sliders that allow a particular image in a stack to be selected.

#### 3.4.3.1 the Image List area

The image list area is used to control what image, a 4dfp stack, is currently being displayed. The image list is just as its name implies, a list of currently loaded 4dfp image stacks. Clicking on a 4dfp stack will cause the stack to be selected and the image indicated by the slice and element number to be displayed. The image within the stack is displayed can be controlled using the sliders at the bottom left of the Image Viewer. The currently selected image stack is highlighted in red and the check box will be checked. Multiple images can be selected either one at a time or in blocks. To select a single image use the control-left mouse click and the image will be selected. Multiple images can be selected by first selecting a single image and then using a shift-left mouse click on another image to select all images between the first and second selected image. Clicking on a single image

Figure 3.16: The Image Viewer

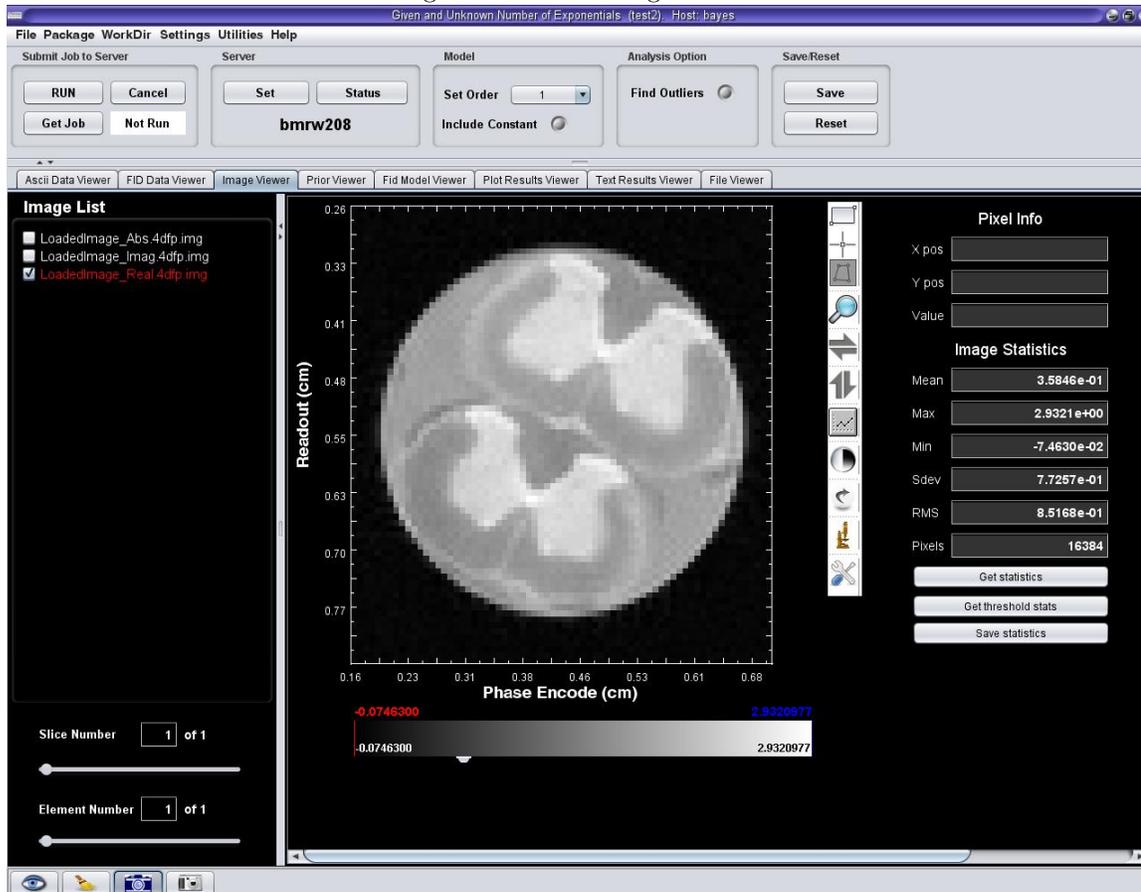


Figure 3.16: When the Image Viewer is selected, this window is displayed. In general terms, this viewer consists of four parts: the image selection widgets on the left-hand side of the viewer. The widgets used to select a slice and element on the lower left-hand side. The image viewing area in the center, and the pixel information area on the right-hand side of the image. For more information on each widget, hold the cursor over a widget and read the tool tip.

Figure 3.17: The Image Viewer Right Mouse Popup Menu

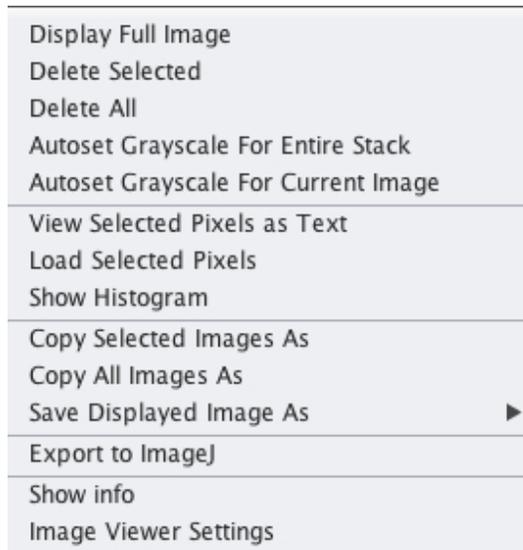


Figure 3.17: The right-mouse popup menu, activated on an image, has many functions: images may be delete, ROI pixels may be loaded into the the Ascii Data Viewer, the gray scale can be adjusted, pixel information can be displayed, pixels can be viewed as text, look at a histogram. Finally, the image stack can be exported to ImageJ. This feature allows one to use all of the facilities in ImageJ to analyze and display images.

will deselect all but the single image clicked on. Selected images are used in some of the packages to indicate which images are to be processed by a package.

The Image List and Image Viewing areas responds to a right mouse click. When activated a menu of options appears. this menu is shown in Fig. 3.17. Here is a list of these menu options and the function they perform:

**Display Full Image** will zoom the display out to show the full image.

**Delete Selected** will delete the currently displayed image.

**Delete All** will delete the all images contained in the image Subdirectory.

**Autoset Grayscale For Entire Stack** will attempt to set a grayscale that can be used do display all images.

**Autoset Grayscale For Current Image** will attempt to set a grayscale that can be used do display the current images.

**View Selected Pixels as Text** will display the pixel values contained in an ROI in a popup text window.

**Load Selected Pixels** will load the pixel values contained in an ROI as an Ascii data set. The abscissa values in this data set are pixel numbers. Loading pixel values will fail if a model is selected that requires more than a two column Ascii Data set.

**Show Histogram** will load the pixel values as a histogram.

**Copy Selected Image As** will copy a selected image stack to a new name and location.

**Copy All Images As** will copy all image stacks to a new directory.

**Save Displayed Image As** will save the currently displayed image as a jpg, png, tif or bmp image.

**Export to ImageJ** will export and open the image to in ImageJ.

**Show info** will display everything know about an image.

**Image Viewer Settings** will bring up a popup that configures the Image Viewer.

### 3.4.3.2 the Set Image area

The bottom-left area shown in Fig. 3.16 is used to set the image that is currently being displayed. This is done by simply moving either the slice or element number slider to the desired image. Alternately, the slice or element number can be entered and the viewer will display the desired image.

### 3.4.3.3 the Image Viewing area

The area in the center of the Image Viewer is used to display images. The Image Viewing area responds to a right-mouse click and the widgets on this submenu are the same as those shown in Fig. 3.17 and we urge people to read the previous Subsection to determine their function. However, the image Viewing area has two menus that can be used to manipulate images. The first of these menus is at the top of the image viewing area. Here is a close up of this menu: Each of the buttons on



this menu performs various tasks associated with images. When activated, clicked on, the functions of these buttons are:

-  when activated a square ROI can be drawn on the image. Put the cursor in the image where an ROI is desired. Hold down the left-mouse and drag the cursor diagonally to draw the ROI. When an ROI is present the buttons on the right-hand side of the Image Viewer can be used to compute some statistics about values of the pixels. We will have more to say about this in Section 3.4.3.6
-  the star will create a point ROI. Simply place the cursor on the pixel to be capture and click the left-mouse button.
-  when activated the polygon widget draws a polygon ROI. To do this place the cursor where the ROI is to start and click the left-mouse button. A dot should appear at this vertex. Now move the cursor to the place where Move to the next polygon vertex to be and click the left-mouse button. Continue this for as long as needed. However, to close and end drawing the ROI end the drawing by placing the cursor on the starting vertex and left-mouse click a second time.



when activated this button will expand a square ROI. This button only functions when a square ROI is present.



when activated the image is flipped left to right.



when activated the image is flipped top to bottom.



when activated the pixel values contained within an ROI are averaged, written to an Ascii file and the Ascii file viewer is activated displaying the extracted pixels. This widget only functions when an ROI is present.



when activated the interface will attempt to determine a gray scale appropriate for the image stack. When the calculation is completed the images are displayed using the new gray scale.



when activated the original image is displayed using the original gray scale.



when activated the image stack is copied and exported to ImageJ.



when activated the user can set user preferences. This is the same popup that is available on the Settings/Preferences menu.

#### 3.4.3.4 the Grayscale area on the bottom

The area at the bottom of the Image Viewing area is a grayscale area that can be used to manually set the grayscale. Here setting the grayscale means setting a lower and upper threshold on the pixel values that are displayed. To adjust the lower threshold, position the cursor on the grayscale bar at the value to be thresholded and do a left mouse click. This will move the red hatch mark to the cursor position, it will set the minimum threshold and finally it will redisplay the image using this threshold. Similarly, to set the upper threshold, position the cursor on the grayscale bar at the value threshold and do a right mouse click. This will move the blue hatch mark to the cursor position, it will set the maximum threshold and finally it will redisplay the image using this threshold. The threshold hatch marks, the blue and red vertical bars, can also be dragged. That is to say position the cursor on either the red or blue hatch mark and then drag the hatch mark to higher or lower values whichever is appropriate. Also the selected thresholds can be moved left or right on the grayscale bar. To move the threshold, located small tab in the center of the grayscale bar. This tab can be dragged left or right to vary the grayscale window that is displayed. This tab cannot be dragged until the left or right mouse clicks are used to raise or lower the displayed grayscale.

#### 3.4.3.5 the Pixel Info area

The Pixel Information area is the area on the top right-hand side of the Image Viewer, see Fig. 3.16. It is used to display information about the image. The three widgets contained in this area have the following functions:

**X Pos** is the pixel number in the X (horizontal) direction. The first number is the pixel number in the raw image. Note that the graphic display in the Image Viewer is a  $512 \times 512$  pixel display. The second number, the one in parentheses, is the pixel number in the graphics area and ranges from 0 to 511.

**Y Pos** is the pixel number in the Y (vertical) direction.

**Value** is the intensity of the pixel from the raw image at the current cursor position.

### 3.4.3.6 the Image Statistics area

is located on the right at the bottom of the Image Viewer. These widgets remain empty until one hits the “Get Statistics” button. When this button is activated the interface will compute a few basic statistics about the ROI if present, and about the entire image if no ROI is present. Here is a brief description of the information displayed in these widgets:

**Mean** contains the mean value of the images pixels in the selected region.

**Max** contains the maximum value of the image pixels in the selected region.

**Min** contains the minimum value of the image pixels in the selected region.

**Sdev** contains the standard deviation of the selected region. If  $p_i$  stands for the  $i$ th pixel in the selected ROI containing  $N$  pixels, then the “Sdev” value is calculated as:

$$\text{Sdev} = \sqrt{\frac{1}{N} \sum_{i \in \text{ROI}} [p_i - \text{Mean}]^2} \quad (3.1)$$

and Mean is the mean pixel value in the ROI and is given by

$$\text{Mean} = \frac{1}{N} \sum_{i \in \text{ROI}} p_i. \quad (3.2)$$

Note, for absolute value images this calculation should not be used as an estimate of the noise standard deviation. In a ROI containing only noise the Mean value is part of the noise and should not be subtracted when calculating the noise standard deviation. Additionally, even in an Absorption mode image this calculation may not give a good estimate of the noise standard deviation again because the deviation from the mean is not an estimate of the noise value.

**RMS** contains the **R**oot **M**ean **S**quare deviation of the selected region. Again if  $p_i$  stands for the  $i$ th pixel in the selected ROI containing  $N$  pixels, then the “RMS” value is calculated as:

$$\text{RMS} = \sqrt{\frac{1}{N} \sum_{i \in \text{ROI}} p_i^2}. \quad (3.3)$$

Note, for absolute value images this calculation should give a better estimate of the noise standard deviation and in absorption mode images this is the correct way to calculate the noise standard deviation.

**Pixels** contains the total number of pixels in the ROI.

Finally, there are three buttons at the bottom of the “Image Statics” area that are used to get and save various statistics computed by the interface:

**Get Statistics** will generate the statistics from the selected ROI. If no ROI is present then the statistics are generated for the entire image. Note this widget does not use the minimum or maximum pixel values set by the “Get Threshold Statistics” button, it simply computes the statistics for all pixels contained in the ROI.

**Get Threshold Statistics** will generate statistics for the selected ROI using a minimum and maximum pixel values. When this button is activated it will popup a widget in which a minimum and maximum pixel values can be entered. After entering these values selecting OK will cause the statistics to be calculated and updated. These updated statistics will ignore all pixels below the minimum or above the maximum. If no ROI is present then the statistics are generated for the entire image.

**Save Statistics** will bring up a popup that allows navigation and saving of the statistics. Here is an example of the saved statistics

Min	=	9.0002e+00
Mean	=	9.4447e+00
Max	=	9.9981e+00
SDev	=	2.8362e-01
RMS	=	2.8362e-01
Pixels	=	1292

and this output is left justified in the saved file.

Note all statistics generated are for a single image, there are no capabilities in the interface for applying the statistics calculations across multiple images.

### 3.4.4 Prior Viewer

The Prior Viewer is used by almost every package and, as its name implies, it is used to view, modify and generally set up the prior probabilities used in the Bayesian calculations. The viewer is shown in Fig. 3.18. In its appearance and function it is very similar to the Ascii Data Viewer, Fig. 3.12. Along the left-hand side is a list of priors that can be modified and on the right is a plotting area where the priors are plotted. In the example shown, it is the prior probabilities for the an inversion recovery model that are shown. Inversion recover models are single exponential plus a constant so there are three prior probabilities: the prior probability for the decay rate constant, and the initial and final intensities. When an item in list of parameters is activated with a mouse click, the prior probability for the selected parameter is displayed in the viewing area and the parameters that describe the prior probability are shown in the entry boxes just above the viewing area. The values in the entry boxes can be changed as needed. Selecting a different type of prior will cause the entry boxes to change to something appropriate for the selected prior. In the example shown, the prior is described by three parameters, a low, a peak and a high. The user can set these parameters to any

Figure 3.18: The Prior Probability Viewer

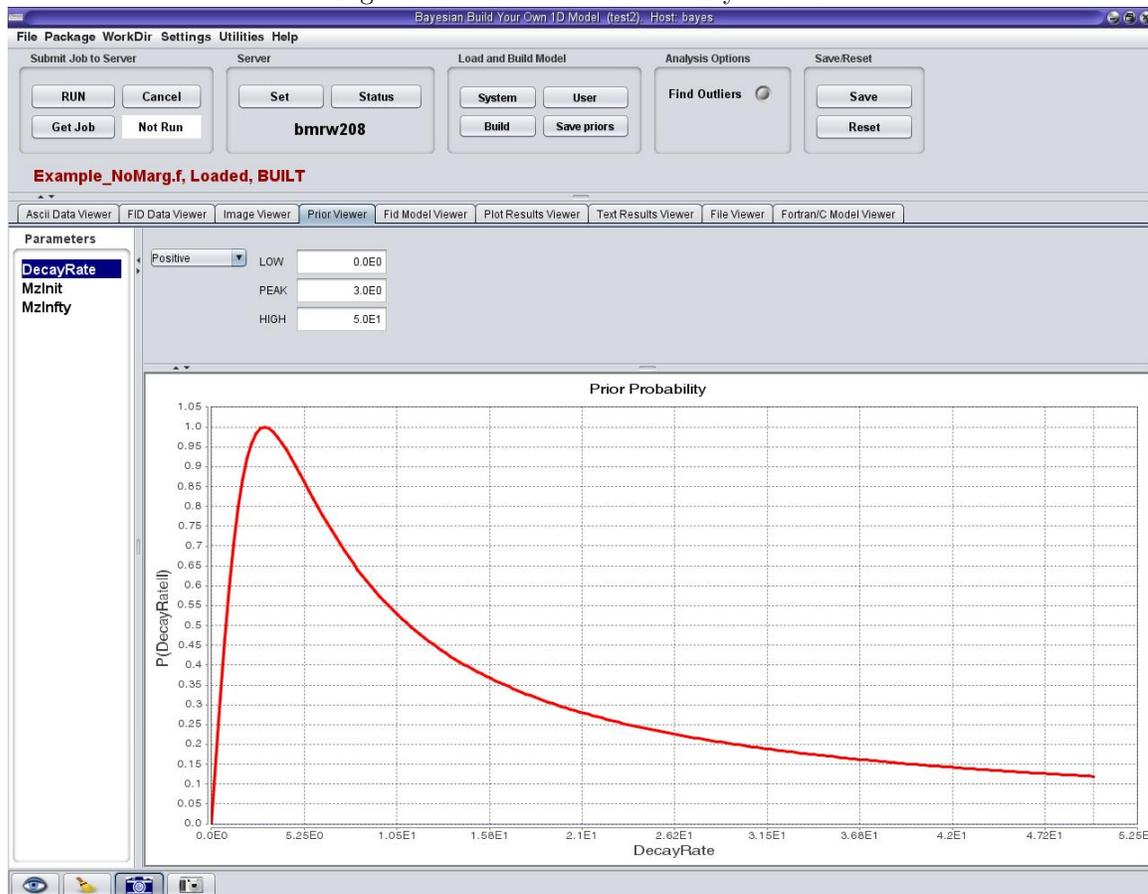


Figure 3.18: The Prior Viewer is used on most packages and it allows one to view and modify the prior probabilities used in the calculation. Here, the viewer is displaying a prior we call a positive prior. The positive prior has a number of very useful characteristics for scale parameters: It goes to zero at zero, thus preventing scale parameters from going negative. It is asymptotically Jeffreys' so allows the upper bound to be very large and it has a peak at a user specified location.

values they wish and the interface will redisplay the prior as to reflect the changes. The interface will allow invalid values in these entry boxes, for example setting High less than Low, but it will not let the analysis to be run with invalid settings. If the parameters are invalid and the run button is activated, the interface will popup an error message and the error must be corrected before the analysis can be run.

When an analysis is run a list of the priors selected by the user is sent to the analysis package. The package normalizes the priors in such a way as to ensure the prior probability is always between zero and one and then proceeds to use the selected prior in the calculation. The normalization for the prior is set discretely, that is to say the prior probability is discretizing and the discrete prior is normalized so that the sum over the discrete samples is one. The prior is discretized on a 101 step inclusive interval. So for example, if the prior is an exponential of decay rate constant 1, with a range given by Low and High, then the normalization constant is given by

$$dX = \frac{[\text{High} - \text{Low}]}{100} \quad (3.4)$$

$$\text{Norm} = \sum_{i=1}^{101} \exp[-\{\text{Low} + dX(i - 1)\}] \quad (3.5)$$

so the prior probability is given by

$$P(X|\text{High Low}) = \frac{\exp[-X]}{\text{Norm}}. \quad (3.6)$$

Plugging in zero for the low and 10 for the high, the normalization constant for the exponential prior is roughly 10.5. The prior starts at about 0.1 and goes down exponentially by about a factor of 22000.

The prior selection box allows the user to select one of five different prior probabilities for a parameter. Once selected, the user can then specify the various elements that determine the prior shape. In all cases the user must supply a prior range, i.e., low and high parameter value. However, each prior type can have other characteristics that must be supplied by the user, for example a Gaussian requires a mean and standard deviation. Here is a description of the five priors and input requirements needed to generate a normalized prior:

**Uniform** selects a uniform prior probability having the user specified low and high parameter range. Uniform prior probabilities are not typically used for continuous parameters, although the user can certainly use them. Rather the interface typically uses a uniform prior probability for expressing an interference between models. For example, the Ascii Model Selection program can load up to 10 models. The prior probability for the model is uniform, i.e., one over the number of models.

**Gaussian** selects a Gaussian prior probability having the mean and standard deviations set by the user. Gaussian prior probabilities are the most common prior probability used in the software. Its a natural prior for amplitudes and it is typically used for any other parameter that can take on both positive and negative values.

**Exponential** selects a exponential prior probability having the decay rate constant set by the user. Exponential priors are typically used in the Bayesian Analysis software for discrete parameters. For example, the prior probability for the number of sinusoids in a model would typically be assigned a exponential prior.

**Positive** selects a positive prior probability having a peak value set by the user. An example of this prior is shown in Fig. 3.18. This prior is meant for scale parameters, and scale parameters can't be negative and this prior will not allow parameters to go to zero, let alone negative. This is clearly illustrated in Fig. 3.18, because the probability goes to zero as the value of the parameter goes to zero. This prior's behavior is asymptotically Jeffreys' [33], again a characteristic desirable for scale parameters. This is also illustrated in Fig. 3.18, because for large parameter, the prior is dropping off very slowly; indeed like  $1/X$ . Finally, given that this prior goes to zero at zero and is asymptotically Jeffreys', the prior must have a peak value. This peak is also seen in Fig. 3.18, and as it turns out, the prior is essentially characterized by the location of this peak. The functional form of the prior is given by:

$$P(X|\text{Peak}) = \begin{cases} \left(\frac{1}{\text{Norm}}\right) \frac{X}{X^2 + \text{Peak}^2} & \text{if } \text{Low} \leq X \leq \text{High} \\ 0 & \text{otherwise} \end{cases} \quad (3.7)$$

where  $X$  is the parameter, Low is the low parameter range, usually zero in this prior. High is the largest parameter value and Peak is the location of the peak. In the numerical calculations, Norm is set using the procedures described above.

**Parameter** is a prior used in the Bayesian Software package when a parameter is to be set to a constant. This prior is essentially a delta function and allows no variation in a parameter. As a computational note, the Markov chain Monte Carlo simulations do not vary any parameter that has a prior type set to parameter. There are a few packages that make use of this prior. For example, the enter Ascii Model package makes use of it as a means of passing the calculation routines parameters. For example, a diffusion tensor model typically needs a conversion constant computed from some spectrometer settings. These spectrometer settings are passed to these modes as parameters and the models use them to compute the conversion factors.

As explained above, when these priors are used they are discretized and normalized to ensure that they sum to one. However, the Prior Viewer does not normalize these priors; rather the viewer sets the largest value in the prior to one. So care must be taken when comparing one prior to another because the prior scales shown in the viewer are not the same as those used in the calculation.

### 3.4.5 Fid Model Viewer

Free induction decay data are time-domain data. However, most people look at fid data in the frequency domain. This presents a unique problem for frequency finding packages, because they work in the time-domain, but their outputs will almost certainly be viewed in the frequency domain. The frequency finding packages either generate time-domain models of the fid data directly or they generate a series of Ascii Model Files that can be used to generate a time-domain fid model of the data. Either way, the Fid Model Viewer must sometimes generate a time-domain fid model and then convert the time-domain fid model in the frequency domain for viewing. The Fid Model Viewer, shown in Fig. 3.19, is used to display the results from the analysis of an time-domain fid in the frequency domain. This viewer is activated when the Fid Model Viewer button is activated. If a time-domain model exists in the package, it is Fourier transformed, phased and the absorption mode spectrum is displayed. If no time-domain model exists, rather if Ascii Model Files exist, then

Figure 3.19: The Fid Model Viewer

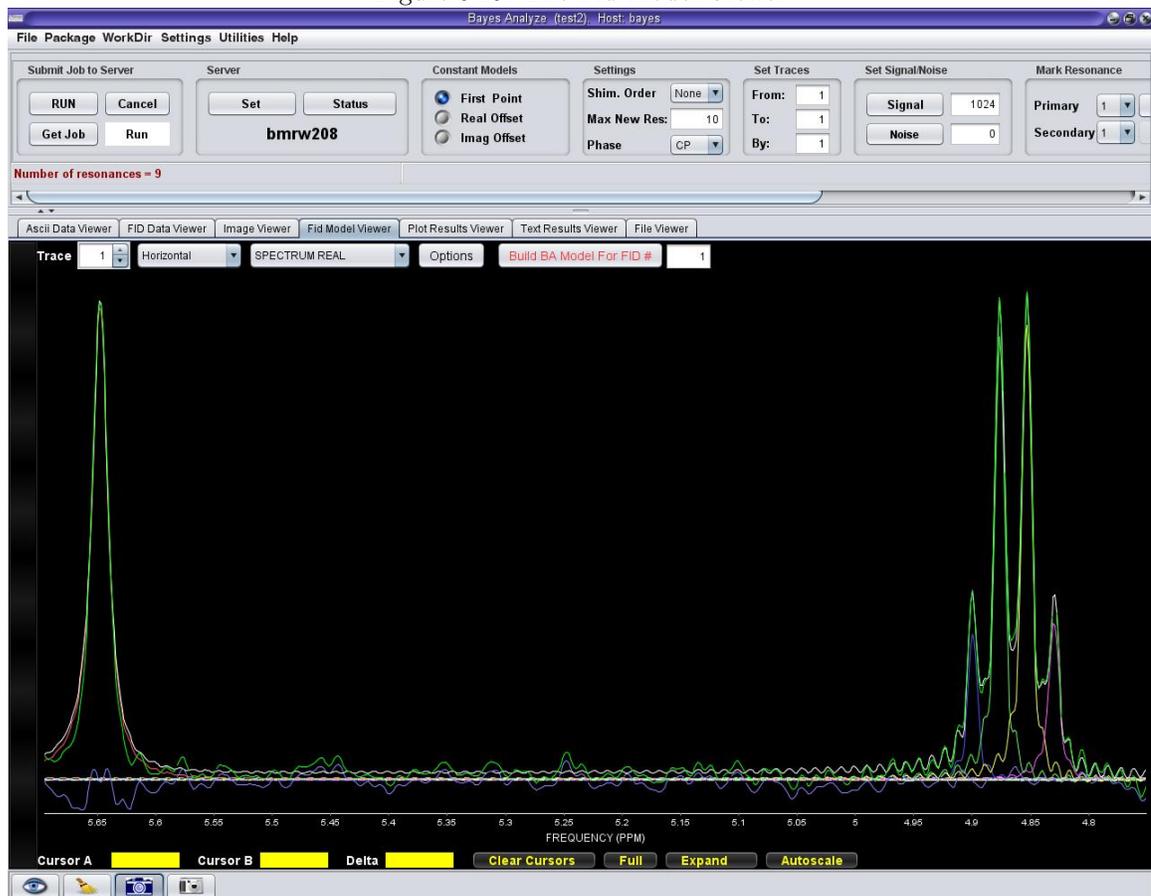


Figure 3.19: Frequency finding programs have the ability to display their outputs overlaid by the original fid. These outputs can be in the time domain or in the frequency domain. Additionally, because fid data can be arrayed displaying these outputs requires a special Viewer to generate and display the results. This is the job of the Fid Model Viewer shown here.

the user must generate a time-domain fid model using the “Build BA Model For fid #” button. When activated the interface will dynamically generate a time-domain fid Model and then Fourier transform, phase and display the model in this viewer.

This viewer is very similar to the Fid Data Viewer discussed in Section 3.13. Indeed the underlying Java class that defines this viewer is the same as the Fid Data Viewer, here it has had a few additional functions added to it. In Section 3.13 we discussed the widgets along the bottom of the viewer and three of the widgets on the top part of the viewer, the “Trace” spinner, the “Data Type” selection and the “Options” widgets. The functions of these widgets is identical to their function on the Fid Data Viewer and, consequently, we refer the reader to Section 3.13 for those discussions. There are three main differences from Fid Data Viewer: first the presence of the pull down menu containing the word “Trace”. This widget allows different types of displays to be selected and we will give a description of the displays shortly. Second is the presence of the “Build BA Model For fid #” button. This button allows one to build and view a model fid and again we will have more to say about this shortly. Finally, the entry box just to the right of this button specifies the trace that is to be modeled.

#### 3.4.5.1 The fid Model Format

Some packages generate fid models when they run: Big Peak/Little Peak and the Metabolite package both do this and some packages, Bayes Analyze and Find Resonances, generate Ascii Model Files that must be further processed to produce a fid model. The “Build BA Model” button builds these fid models from the Ascii Model Files when needed. The Fid Model Viewer button is present on all packages and can be used to generate Bayes Analyze models from previously analyzed spectroscopic fid data. The interface uses the Ascii Model Files to generate a fid Model. Assuming Bayes Analyze Model files are present, when one clicks on the “Build BA Model fid #” button or one enters a fid trace number in the entry box then the interface sends a request to the current server to build a fid Model. The interface waits for this job to run. If needed the user is prompted for a password. When the job completes it is automatically retrieved by the interface and unpacked. The model fid is Fourier Transform and displayed in the Fid Model Viewer, see Fig. 3.19 for an example of the output. The Fid Model Viewer displays a number of traces, one each for each trace in the model field. The model fid is a time-domain arrayed fid contains the following traces:

**Trace 1** is the complex time-domain fid data from the original input fid. i.e., the spectroscopic fid data that is being modeled.

**Trace 2** is the complex time-domain fid model of the input fid.

**Trace 3** is the complex time-domain difference between the model and the fid. This difference is usually called the residuals.

**The remaining traces** are the complex time-domain models of each individual resonance in the model. The resonances are in increasing frequency order, so trace 4 is the most negative frequency and the last resonance is the most positive. If the total number of resonances is  $N$  then the Model fid file contains  $3 + N$  traces.

### 3.4.5.2 The Fid Model Reports

Next to the “Trace” spinner on the Model Fid Viewer, there is a widget that is initialized with the work “Trace” in it. This widget is the “Report Type” widget and the word “Trace” refers to the fact that the default report is to display the model traces. The report type widget is actually a selection menu and when activated it will display a selection of different kinds of reports that can be viewer/printed etc. This report type selection menu has the following selection items:

**Trace** is a display of the individual traces in the model fid. The trace spinner just to the left of the report type selection menu can be used to change the trace number. The number can also be changed by typing in the trace number to be viewed.

**Data** will display the original fid data. As noted the format of the fid data is also controlled by a selection menu. The default data type is to display the spectrum of the data.

**Model** will display the model fid data.

**Residual** will display the difference between the complex fid data and the model.

**Vertical** will display three plots stacked one above the other, the bottom plot is the original fid data, the middle plot is the model fid and the top plot is the residuals. Again the format of these displays is controlled by the data selection menu menu.

**Overlay** also displays the data in three stacked plots. Here the lower plot is the original fid data overlaid by the model. The middle plot is the residuals and the top plot is a plot of each resonance in the data.

**Horizontal** is the plot shown in Fig. 3.19. In this plot the data, model, residuals and the individual resonances are all plotted on the same scale. If examine Fig. 3.19 is examined, it is almost impossible to distinguish the data and the model because they overlap each other almost perfectly. The residuals are the small randomly varying trace in purple. Note that these individual resonances do not actually reach the top of the spectrum in any of the peaks. None-the-less sum of these resonances fit the data perfectly.

**Stacked** is a plot of all traces in the model fid, one above the other. So the lower three traces are the data, model and residuals and traces 4 through the top are the individual resonances in increasing frequency order.

The type of data used in the displayed report is controlled by the data type widget. So for example by setting the data type to “fid” the report type widget will display all of its reports in the time-domain. Similarly, if the data type is set to “Spectrum Real” then the reports use the real part of the spectrum of the model.

### 3.4.6 Plot Results Viewer

The Plot Results Viewer is for all practical purposes the Ascii Data Viewer described in Section 3.12 and everything said about the Ascii Data Viewer applies here. In this Subsection we are going to discuss the differences between the Ascii Data Viewer, explain briefly how to use this viewer and give a general introductions to the kinds of plots this viewer displays. However, each package may

Figure 3.20: The Plot Results Viewer

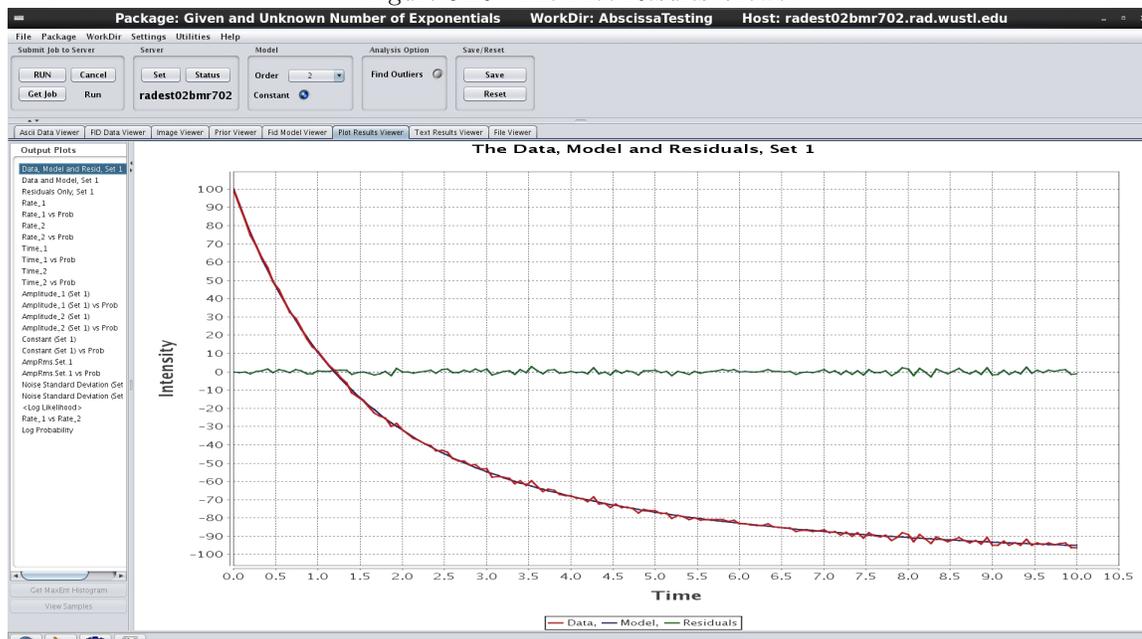


Figure 3.20: Ascii packages and some fid packages output an Ascii Plot file. This file contains a list of all of the Ascii plot files that can be viewed. These plots are displayed by the Plot Results Viewer. Roughly speaking there are six different types of these plots: plots involving the data, model and residuals; plots involving the posterior probabilities for parameters; plots showing correlations between parameters and two types of plots that are used to determine if the analysis ran/converged correctly.

have plots unique and if a plot is discovered that is not explained here, then consult the Chapter on the package being used.

The Plot Results Viewer is shown in Fig. 3.20 and if Ascii Data Viewer is compared to the Plot Results Viewer, one will find that about the only difference is the label on top of the plot files list. In the Ascii Data Viewer this label reads “Ascii Data” while here it reads “Output Plots.” In the Ascii Data Viewer the area under this label contains a listing of all of the files that have been loaded into the experiment, while here it contains a listing of all of the output plots generated by the package. The example shown in Fig. 3.20 is the result of running a biexponential model with a constant on biexponential data. In the output plots list there are 24 different plots. When a plot is activated, the output plot list entry is highlighted and the plot is displayed. The plots are organized roughly as three groups of plots, in the top part of the output plot list are plots showing how the model fits the data. In the middle section are the posterior probabilities for the various parameters appearing in the model. Finally the bottom contains a number of plots that are meant as aids in understanding the outputs from the simulations. A general description of the plots that are common across all Markov chain Monte Carlo packages is given in Section 3.5.

Like the Ascii Data Viewer, this viewer also responds to a right mouse click. This is true for the plotting area and the Output Plots area. However, the menu shown in the plotting area is the

Figure 3.21: Plot Information Popup

```

01 Line 1 2
02 1
03 BayesExpGiven.Rate_1
04 Prob for the Rate_1, Given a 2 Exp Model and a Const
05 Rate_1
06 Rate_1
07 Probability Density
08 Mean: 0.26448; Sd: 2.15E-02; Peak: 0.26487

```

Figure 3.21: The plot information widget will popup up a window containing information about the currently displayed plot. We have numbered the lines in this display to make referencing them easier. Line 01 indicates that this is a line plot and that column 1 vs 2 from the file named in Line 03 is to be plotted. Line 02 is an internal indicator and it tell us that column 1 of the Bayes.Mcmc.Samples file is to be displayed should the user activate the “View Samples” button. Line 03 is the name of the file being plotted. The file is located in the “Bayes Home director/WorkDir/BayesOtherAnalysis” directory. Line 04 is the title of the plot. Line 05 is a long abscissa label. Line 06 is a short abscissa label. Line 07 is the Y axis label. Finally, Line 08 on probability plots, contains the parameter estimates.

same as in the Ascii Data Viewer, but the menu shown in the Output Plots differs. Here is a brief description of this submenu and what it does:

**Plot Information** will display all of the information available to the interface about this plot.

Figure 3.21 is an example of what is displayed when the plot information menu item is selected. We have numbered the lines in this figure to make referencing them easier. Line 01 indicates that this is a line plot and we are to plot column 1 vs 2 from the file named in Line 03. Line 02 is an internal indicator and it tell us that column 1 of the Bayes.Mcmc.Samples file is to be displayed should the user activate the “View Samples” button. Line 03 is the name of the file being plotted. The file is located in the “Bayes Home director/WorkDir/BayesOtherAnalysis” directory. Line 04 is the title of the plot. Line 05 is a long abscissa label. Line 06 is a short abscissa label. Line 07 is the Y axis label. Finally, Line 08 on probability plots, contains the parameter estimates information displayed as part of the title.

**Show Data From Plot** will display the data actually plotted in a popup. This display is usually two column, but it will be multicolumn when plots having more than a single trace are displayed.

**Show Source File for Plot** will popup a window containing the original source file. This source file can be different from that displayed by the “Show Data From Plot” widget. For example, the file containing the data, model and residuals is a four column file; while a plot of only the data and the model would contain only three columns even though the source file is four columns.

### 3.4.7 Text Results Viewer

After an analysis has been run the viewers are used to look at the outputs from the analysis. The Text Results view show in Fig. 3.22 is a typical example of the Ascii outputs. We are going to describe each of these shortly but note that the Text Results Viewer shows the output from two different packages: The current package, labeled standard, and Bayes Analyze. The reason the Bayes Analyze package is shown is because that package is unique in that some of the outputs from that package can server as inputs to the current Ascii package. Consequently, the current working directory can contain outputs from the current package in BayesOtherAnalysis and outputs from Bayes Analyze in BayesAnalyzeFiles. Activating any of the items on these selection menus will cause the appropriate file from the selected package to be listed in the Viewing area. If the requested file does not exists then an appropriate message is displayed. You can also switch reports by selecting them or by using the up and down arrows on your keyboard. However, you cannot jump between the standard output and the Bayes Analyze outputs using the up and down arrows. The entries in both text results viewers, standard and Bayes Analyze, are fixed and do not vary from package to package. Along the top of the viewing area are a number of buttons that are specific to the Text Results Viewer. Here is a description of these buttons:

**Print** will direct the currently selected file to the printer. the button will popup a widget that allows you to select a few print options including the printer.

**Copy** will copy the currently selected file to your clipboard.

**Save** will save the current copy of the file on top of the original file located in BayesHome/CurrentWorkDir/BayesAnalyzeFiles.

**Save As** will popup a navigation window that will allow you to navigate to the location you wish the file to be saved and then it will save the file.

**Enable Editing** will allow you to change the contents of the viewing window. For example, you might want to specialize a title or some text in the window as a reminder of what you did in the analysis prior to saving the results. When activated the check box just to the right will toggle on and off. You can also just check the box to enable editing.

**Scroll Up** will cause files to be positioned at either the beginning or ending of the file.

**Settings** will popup a window that allows you to configure the Text Results Viewer. Mostly it allows you to set the fonts and font point sizes.

The Standard selection list has a number of widgets associated with it and these reports are briefly described here. See the individual packages for a more detailed description of these reports. In all cases when a widget in the standard selection list displays a file it is located in your BayesHome/CurrentWorkDir/BayesOtherAnalysis directory.

**Instructions** will redisplay the instructions that are shown whenever a package is started. These instructions are typically terse, but they will indicate the types of things that must be done to successfully run a package. For more information on the individual packages consult the appropriate Chapter.

Figure 3.22: The Text Results Viewer

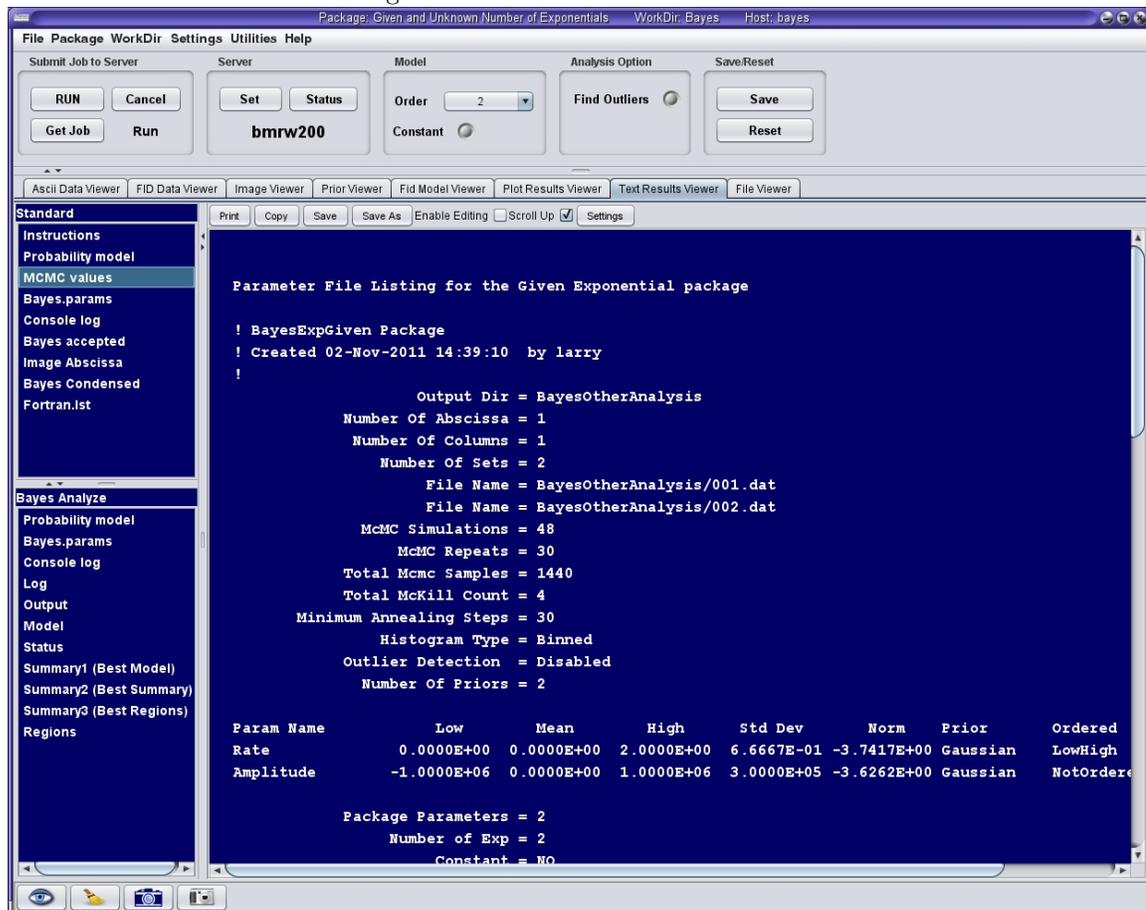


Figure 3.22: After a package has been run, the various outputs can be viewed using the Viewers. The Text Results Viewer is used to view the various reports that are output from the packages. The area along the left-hand side lists the various reports available from the current package, top, and from the last run of Bayes Analyze, bottom. The text has a brief description of the output and you must consult the package Chapter to find out about package specific reports.

**Probability model** will display the contents of the Bayes.prob.model file. This file usually contains one record for each time an analysis package has been run and it contains the results of a thermodynamic calculation for the probability for the model. See [61, 37] for more on thermodynamic integration.

**Mcmc Values** will display the contents of the PackageName.mcmc.values file, where “Package-Name” is our internal name of a package. If you are looking for this file, there is only a single file in the BayesOtherAnalysis directory having the suffix “mcmc.values”. For a much more detailed description of this report, see Subsection D

**Bayes.params** will display the contents of the Bayes.params file. The Bayes.params file contains a complete description of the package setup including the prior settings.

**Console.log** will contain all outputs that went to console when a job is running. Usually this is not useful, but when something goes wrong the console log would be consulted.

**Bayes.accepted** is the file that is displayed when the “Get Job” widget is activated and the job has not yet finished. The exact contents of this report are specific to a package but in general terms it consists of two parts: A header:

Accepted Report given the ExpTwoConst\_Marg model

McMC Phase:	Sampling
McMC Simulations:	50
McMC Repeats:	30
Number Killed Per Cycle:	5
Min Annealing Steps:	51
Current Step:	30
Fraction Samples Gathered:	1.00000
Average Log Posterior Prob:	-3.66274138E+02
StdDev Log Posterior Prob:	1.05944
Average Log Prior:	-19.245
Average Log Likelihood:	-347.029
StdDev Log Likelihood:	1.055

which is reasonably standardized. In general terms most of this header is either setup information concerning the package, or its a set of current statistics about the current status of the package. For example the Simulations, Repeats and Minimum Steps are all setup parameters; while the others are current status information. For example the various average probabilities are the mean value of the given probability averaged across the number of simulations, in this case 50. Similarly, the StdDev values are the standard deviation of the given probability. In general terms the likelihood should increase as a function of the annealing step; while the posterior probability will decrease. In both case the standard deviations of these quantities will be large for low values of the annealing parameters and will become smaller as the annealing parameter increases.

The lower part of the Bayes.accepted file is package specific and the number of entries in this part of the report can be highly variable. In general terms the lower part of the report will

contain some statistics about how often each parameter in the Markov chain Monte Carlo simulation is being accepted and rejected. Here is an example taken from the Enter Ascii Model package with the marginalized two exponential plus a constant model

Param Desc	Avg. Param.	Param Sd.	Proposal	Prior Contrib	Rate
DecayRate1	2.682716E-01	2.179858E-02	4.367748E-02	-4.525808E+00	0.2393
DecayRate2	9.382104E-01	4.349060E-02	7.788657E-02	-3.835734E+00	0.2485

Because this is a marginal probability density function only the decay rates are varied in the McMC simulations and consequently only the decay rates are shown in the accepted report. The entries are the average parameter value, its current standard deviation, the current proposal used in the McMC simulation, the contribution of this parameter to the logarithm of the prior probability and finally the acceptance rate. The acceptance rate, labeled Rate, is the number of times a parameter was accepted divided by the total number of times a new value was proposed. The packages try to keep this acceptance rate between 20 and 30%. For more on how this is done see Section B and for more about the Bayes.accepted report generated by a specific package consult the appropriate Chapter.

**Image Abscissa** will display the file BayesHome/WorkDir/images/Abcissa if its available. This file is used in image processing to tell the Ascii model files what the abscissa values are for the Ascii model. Consult the Section A for more about this file.

**Bayes Condensed** displays the file BayesHome/WorkDir/BayesOtherAnalysis/Bayes.Condensed.File an example of this file is shown in Fig. 3.23. The file consists of one line per parameter in the model. Each line contains the parameter name, the mean and standard deviation of the parameter computed from the simulations and the parameter value taken from the simulation that had maximum posterior probability. The header shown in this figure is not present in the actual condensed file, that header was put in the figure only as an aid in identifying the various fields.

The names shown in this plot are generally the names assigned to the parameter by the user or by us when we wrote the code. These names should be simple and self explanatory. As illustrated in Fig. 3.23, the interface will modify the names of the amplitudes and noise standard deviations by appending a data set number to them when multiple data sets are used. In this example the data set numbers are the “.01” and “.02” suffixes. So, for example, “BayesExpGiven.Amplitude\_2.02” is the amplitude assigned to decay rate 2 in data set number “02”. Similarly, “BayesExpGiven.Amplitude\_2.01” is the amplitude number 2 in data set “01”.

**Fortran.lst** will display the file BayesHome/WorkDir/model.compile/CurrentModel.lst where “CurrentModel” is the name of the Fortran or C model loaded. Note that this listing will contain any errors issued by the Fortran or C compilers when the “Build” button is activated.

Note that if you were to use the File Viewer to look at the contents of a WorkDir after it has been successfully run, you will find that it contains many more files than mentioned here. Those other files contain the probability density functions and the other reports mentioned earlier in this Section.

As noted, the Text Viewing area allows you to view the output from the current model, but it also lets you view the Bayes Analyze output from the previous run of Bayes Analyze. This selection menu also contains a fixed number of entries. Activating each entry will do the following:

Figure 3.23: The Bayes Condensed File

Parameter Name	Mean	StdDev	Peak
BayesExpGiven.Rate_1	1.63255E-03	1.47866E-03	2.22598E-05
BayesExpGiven.Rate_2	2.95738E-01	6.03008E-03	3.01355E-01
BayesExpGiven.Time_1	4.44296E+03	5.02856E+04	4.49241E+04
BayesExpGiven.Time_2	3.38280E+00	7.03289E-02	3.31834E+00
BayesExpGiven.Amplitude_1.01	-1.01643E+02	1.84909E+00	-9.97170E+01
BayesExpGiven.Amplitude_2.01	1.01287E+02	1.72691E+00	9.94602E+01
BayesExpGiven.AmpRms.Set.1	1.43493E+02	2.52340E+00	1.40840E+02
BayesExpGiven.NoiseStdDev.01	1.08257E+00	8.47944E-03	1.07244E+00
BayesExpGiven.Amplitude_1.02	-1.00316E+01	1.80558E-01	-9.84339E+00
BayesExpGiven.Amplitude_2.02	9.70180E+00	1.69333E-01	9.52261E+00
BayesExpGiven.AmpRms.Set.2	1.39556E+01	2.46988E-01	1.36957E+01
BayesExpGiven.NoiseStdDev.02	1.07327E+00	5.41963E-04	1.07274E+00
BayesExpGiven.RmsAmplitude_1	7.22218E+01	1.31372E+00	7.08533E+01
BayesExpGiven.RmsAmplitude_2	7.19484E+01	1.22696E+00	7.06506E+01
BayesExpGiven.RmsAmpTotal	1.01944E+02	1.79284E+00	1.00058E+02

Figure 3.23: The Bayes.Condensed.File is shown here. This file is a condensed version of the outputs found in the Mcmc.Values report. The file consists of one output line for each output parameter including derived parameters. Each output line consists of the parameter name, the mean value computed and standard deviation of the samples gathered in the Markov chain Monte Carlo simulation. The peak parameter value is the value of the parameter in the simulation that had peak posterior probability. The heading line shown in this Figure is not present in the condensed file, it is here only to aid in describing the parameters

**probability model** is a file containing the probability for the model. Its function is similar to that displayed in the Standard output but the way the file is produced and the format of the file are completely different. In the Bayes Analyze outputs the Probability model file contains one line for each resonance added to the model. These lines contain a description of the model, the logarithm of the posterior probability, the probability gain, and the date and time the model was added. For a complete description of this file, see Subsection 8.5.4.

**Bayes.params** The bayes.params file is written by the interface and serves as the input parameter file to Bayes Analyze. It contains various parameter settings and the initial model to be processed. The parameter file is divided into three general sections, a header, the global parameters, and the resonance parameters. Each of these three sections is describe in detail in Subsections 8.3, 8.5.1.2 and 8.5.1.3.

**Console.log** is a running history of what model is being analyzed at the current time. Here is a small snippet of this file:

```
bayes_analyze (V01.20-00)
Developed by Washington University School of Chemistry and
Monsanto St. Louis NMR Center
Base 10 Log Evidence for The First Resonance is: 268.3

Beginning a 1 resonance model
Base 10 Log Of The Probability for 1 Resonance =-3.75974811E+03
Base 10 Log Evidence for The Next Resonance is: 59.2

Beginning a 2 resonance model
Base 10 Log Of The Probability for 2 Resonances =-3.71837136E+03
Base 10 Log Evidence for The Next Resonance is: 65.6
```

As you can see from the above list, the console log is just an indication of the current model, the logarithm of the posterior probability for that model, and finally the log of the evidence that there is another resonance in the data.

**log** is a complete list of all of the steps taken in the Levenberg-Marquardt searching algorithm. For a complete description of the log file, see Subsection 8.5.5.

**Output** the output file is a detailed output from every model processed by Bayes Analyze. For a complete description of the output file, see Subsections 8.5.3

**Model** the Bayes Analyze model file is used as input to Bayes Model. Bayes Model takes the parameters in the model file and generated an fid model of the Marquardt fid. For a complete description of the model file, see Subsections 8.5.7

**Status** While Bayes Analyze is running it updates a status file with some information about what it is currently doing. This information is written into a status file that is fetched by the status button. For a complete list of the various status messages see Subsections 8.6

**Summary 1** When the summary 1 report is run, it goes into the Bayes Output file and locates the model which had the highest posterior probability and then writes that model into the summary1 file. For a complete description of this file, see Sections 8.5.8.

**Summary 2** When the summary 2 report is run, it goes into the Bayes Output and Bayes model files and locates the model that had maximum posterior probability and then produces a summary of the report. For a complete description of the summary 2 file, see Sections 8.5.9.

**Summary 3** or the regions report is produced whenever a regions file is present in the BayesAnalysisFiles directory. When present the scripts will run the summary 3 report. For a description of how to generate a regions file, see Subsections 3.4.2. For a complete description of the summary3 report, see Sections 8.5.10.

**Regions** will list the regions file if it exists. For a description of how to generate a regions file, see Subsections 3.4.2. For a complete description of the regions file, see Sections 8.5.10.

### 3.4.8 Files Viewer

The Files Viewer is a tool provided to assist you in finding files. When activated the viewer opens in your current Bayes Home directory and it shows you a listing of all files and directories in Bayes Home. Using a single mouse click on a file will display that file; while using a double mouse click on a directory will expand that directory. You can use this tool to quickly locate and display files. Note that as of this writing a variable length font is in use by this viewer and consequently, it does not preserve line spacing when a file is displayed.

## 3.5 Common Interface Plots

All of the packages that use Markov chain Monte Carlo to approximate the posterior probability for the parameters share a number of common plots. In this section we are going to describe those plots and exactly what you can use them for. When a Markov chain Monte Carlo package runs it outputs a list of plots that are available to the user. This list of plots is shown in the Plot Viewer on the left-hand side under the “Output Plots” heading. The first three plots are: the data, the model and the residuals (the difference between the data and the model), the data and the model; and finally just the residuals. However, if more than one data set was analyzed jointly, these plots are repeated for each data set. These plots are followed by plots of the posterior probabilities for each parameter appearing in the model. These plots are paired in groups of two plots, one of the posterior probability for the parameter and one scatter plot of parameter value versus the posterior probability. Plots of the posterior probability for the noise standard deviation in each data set follow the parameters used in the model. These noise standard deviation plots are followed by a plot of the expected logarithm of the likelihood, we will have more to say about this plot shortly. The expected log likelihood plot is followed by a set of scatter plots and finally the logarithm of the posterior probability in each simulation is plotted as a function of repeat number. We are going to describe each of these general plot types in the following subsections.

Figure 3.24: Data, Model, And Resid Plot

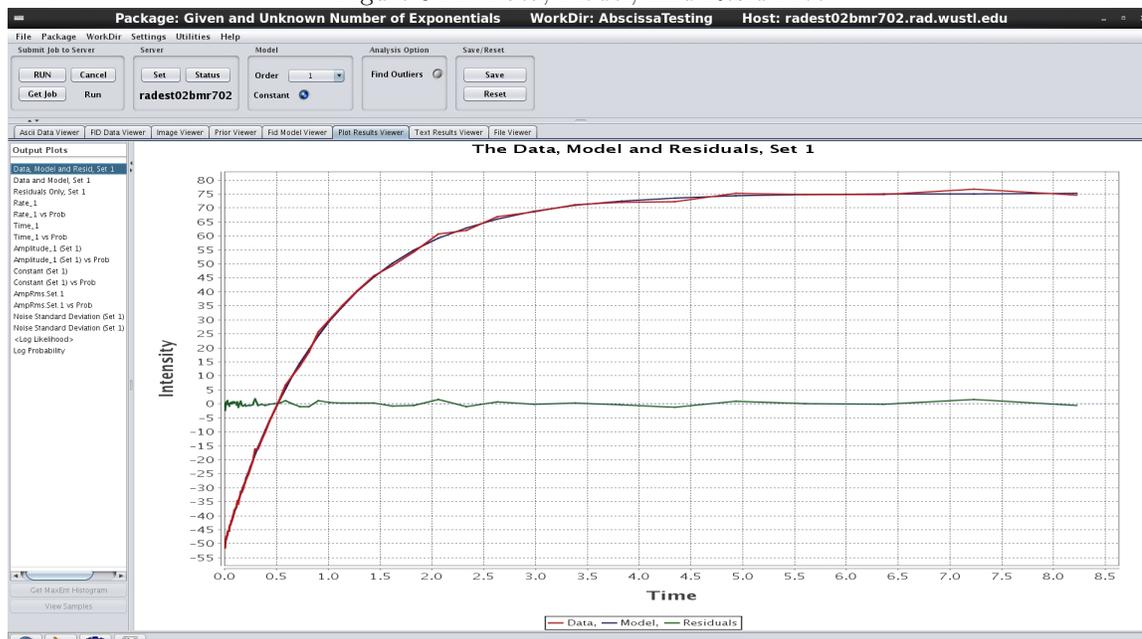


Figure 3.24: The data, model and residuals plot is generated from the simulation that had maximum posterior probability. The three lines are the shown in red, the model of the data generated from the parameters taken from the simulation that had maximum posterior probability are shown in blue. The residuals, the difference between the data and the model, and shown in green.

### 3.5.1 Data, Model And Residual Plot

In all packages that used Markov chain Monte Carlo to approximate the Bayesian Posterior probability, the very first output plot is of the data, the model and the residuals. An example of this plot is shown in Fig. 3.24. In this figure the data are shown as the red line. The model of the data generated from the simulation that maximized the posterior probability is shown as the blue line. Finally, the residual, the difference between the data and the model, is shown as the green line. In most cases these plots and residuals all exist on the same scale and consequently, will all show up and be displayed properly. However, it is possible for the residuals to be so different in scale that displaying the data, model and residual plot is a problem. An example of this occurs in certain types of radiology data where the average signal value is set to a large number but because the data are almost constant, the residuals are always essentially zero and plots, while correct, are scaled very badly. To avoid this kind of problem we generate two additional plots, not show. One is of the data and the model. Because we do not include the residual, its possible to scale this plot much better than when the residuals is present. The third plot is just of the residuals. And again because the model and data are not present, its possible to scale this residual plot and see what is happening.

If multiple data sets are being analyzed jointly, then these three plots are repeated one time for each data set. The Output Plot label will have the data set number in it so you can tell by looking at the output plot label or the title of the plots also contain the data set number.

Figure 3.25: The Parameter Posterior Probabilities

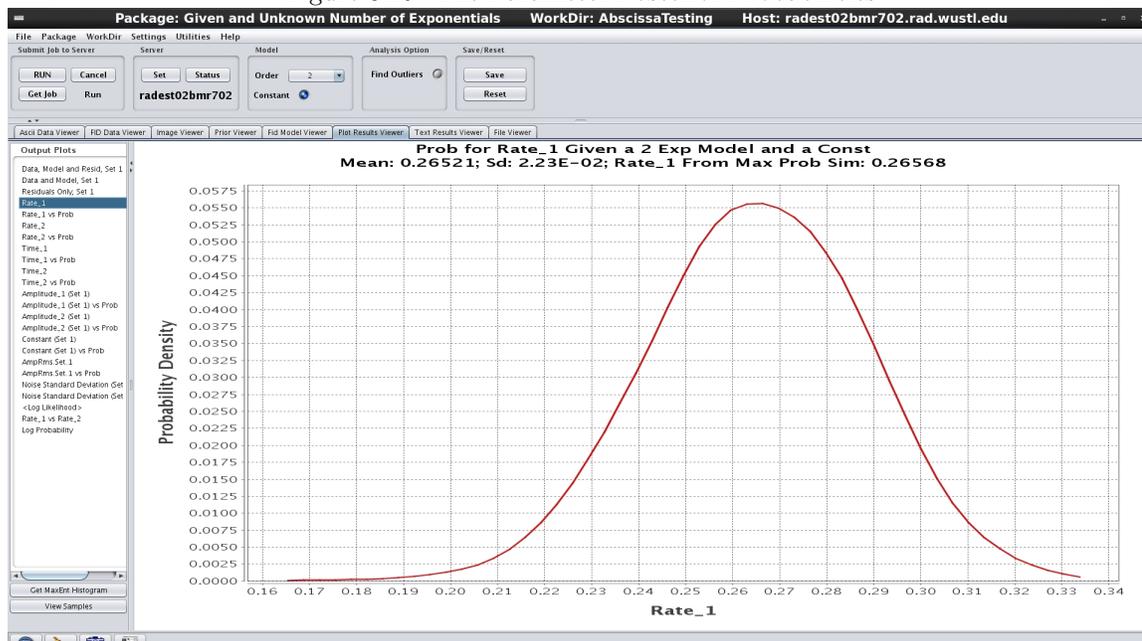


Figure 3.25: The posterior probability for a parameter is a kernel density histogram created out of the samples from the Markov Chain Monte Carlo simulation. In this plot, on the far left, the posterior starts out low, essentially zero, and increases as the parameter increases. Because probabilities represent degrees of belief, this is saying that values as low as 0.975 are not very probable, however a value of the parameter of 0.995 is very probable, and as the parameter is increased, the probability decreases and goes to zero around 1.020. Consequently, in Bayesian Probability the width of a posterior probability distribution is a natural measure of how uncertain one is of the estimated parameter value.

### 3.5.2 Posterior Probability For A Parameter

In Bayesian probability theory everything known about a parameter is summarized in a posterior probability. These posterior probabilities are written  $P(\alpha|DI)$ , where this should be read as the posterior probability the parameter had value  $\alpha$  given the data  $D$  and the prior information  $I$ . Figure 3.25 is a plot of the posterior probability for a decay rate constant using a biexponential plus a constant model. The data used in this analysis is just the biexponential plus a constant data that resides in the Exponential directory of the Bayesian Analysis test data.

The posterior probability for a parameter plot is a kernel density histogram created out of the samples from the Markov Chain Monte Carlo simulation. It is the standard plot used when generating posterior probabilities for parameters. The plot is strictly a visual representation of the parameters drawn from Markov chain Monte Carlo simulation. Monte Carlo samples and is not used in any calculations. It is their to aid the user in visualizing the sample distribution. In Fig. 3.25, on the far left the posterior probability starts out low, essentially zero, and for this parameter increases as the parameter increases. Because probabilities represent degrees of belief, this is saying that values as low as 0.16 are not very probable, however 0.265 is very probable, and as the parameter is

increased further the probability decreases and goes to zero around 0.33. Consequently, in Bayesian Probability theory the width of a posterior probability distribution is a natural measure of how uncertain one is of the estimated parameter value. Along the top of this plot is a title that gives the name of the parameter being plotted, in this case “Rate\_1” which is indicating this is the decay rate 1 in the biexponential model. The “Given” clause describes the model, so this is a 2 exponential model plus a constant. This title line is used on all posterior probability plots. The second line of this title contains the estimated mean and standard deviation of the Markov chain Monte Carlo samples. Note this mean and standard deviation is *not* the mean and standard deviation of the histogram, it is the mean and standard deviation of the Monte Carlo samples. In this case, the mean is 0.26521 and the standard deviation is 0.0223, so Bayesian Probability theory thinks the mean is uncertain in the second decimal place. Finally the header contains the value of the parameter, in this case “Rate\_1”, from the simulation that had maximum posterior probability. Again note that this is not the same as the peak in the histogram.

### 3.5.3 Maximum Entropy Histograms

When a posterior probability for a parameter is displayed the two buttons at the bottom of the “Output Plots” panel are enabled. These two buttons allow one to view the output samples and they allow one to pass the Markov chain Samples to the Maximum Entropy Histogram package. When this option is selected the interface runs an analysis that will infer the number of Lagrange multipliers necessary to generate a fully Bayesian density function representation of the samples. From these samples of the Lagrange multipliers the Maximum Entropy Histogram package will then be used to generate an output histogram with error bars, see Chapter 25 for more on the Maximum Entropy Histogram package. When the “Get MaxEnt Histogram” button is activated the samples obtained from the Markov chain Monte Carlo simulation are shipped to the server in background and the MaxEnt Histogram package is run. When the analysis finishes the interface automatically fetches the results and replaces the current kernel density histogram with the maximum entropy histogram. An example of this type of histogram is shown in Fig. 3.26. The Maximum Entropy histogram is distinguished from the kernel density histogram by the gray background, the kernel density histogram background is white, and the histogram has a blue region around the red line. The red line is the mean value of the density function averaged over all of the Markov chain Monte Carlo simulations used in the analysis. Again, this analysis is run in background and the user never sees anything of it but the resulting histogram. The shaded blue region is a standard deviation computed from the Markov chain Monte Carlo samples for a given parameter.

### 3.5.4 Markov Monte Carlo Samples

Finally, the actual samples generated by the Markov chain Monte Carlo simulations can be displayed by activating the “View Samples” button. When this button is activated the samples are displayed as a scatter plot of the sample value versus the sample number, Fig. 3.27. In this plot the horizontal is just sample number and are essentially arbitrary. However, if you ran 50 Markov chain Monte Carlo simulations, then the first 50 dots shown are the samples drawn from one repetition of the simulations. After these simulations are saved, the Markov chain Monte Carlo simulations are processed long enough for the simulations to “forget” where they started, and another 50 samples are drawn and displayed. So if there are patterns in these samples it would be strong evidence that the Markov chain Monte Carlo simulations failed to converge.

Figure 3.26: The Maximum Entropy Histograms

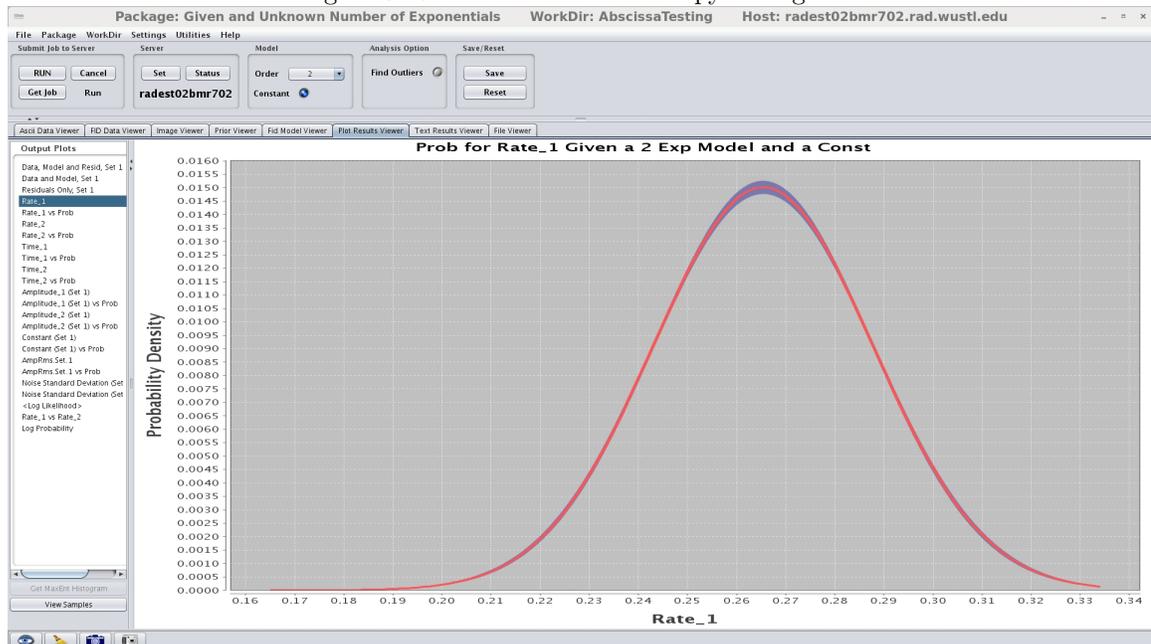


Figure 3.26: When the “Get MaxEnt Histogram” button is activated, the interface runs an analysis that computes the Maximum Entropy Histogram using Bayesian Probability theory. This histogram is automatically fetched by the interface and replaces the kernel density binned histogram. The plot is distinguished from the kernel density plot because the background is gray and the histogram has error bars shown as the blue region. This blue region is a one standard deviation on the histogram mean. Fig. (25).

Figure 3.27: The Parameter Samples Plot

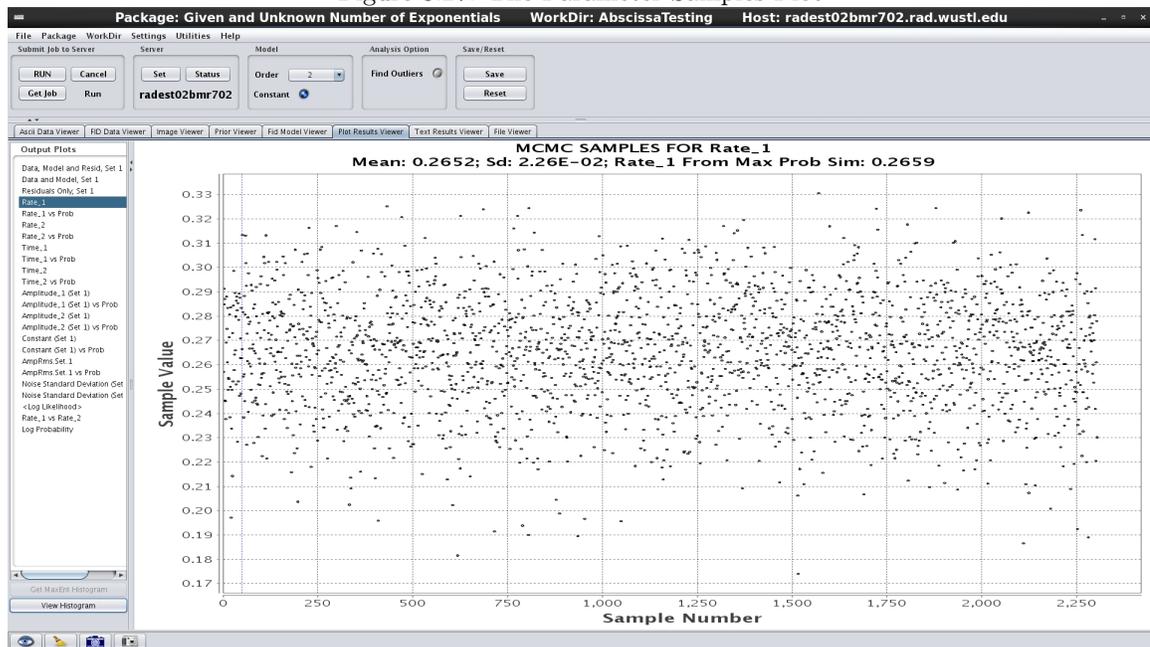


Figure 3.27: When the “View Samples” button is activated the parameter sample values drawn from the Markov chain Monte Carlo simulation are shown as a function of sample number. If the Markov chain Monte Carlo simulation has worked properly these samples will appear random with no visible patterns. Usually the sample will have a high density area that thins out as you go away from the mean value. The mean and standard deviations shown in the headers and in the MCMC Values report are computed from the samples.

Figure 3.28: Posterior Probability Vs Parameter Value

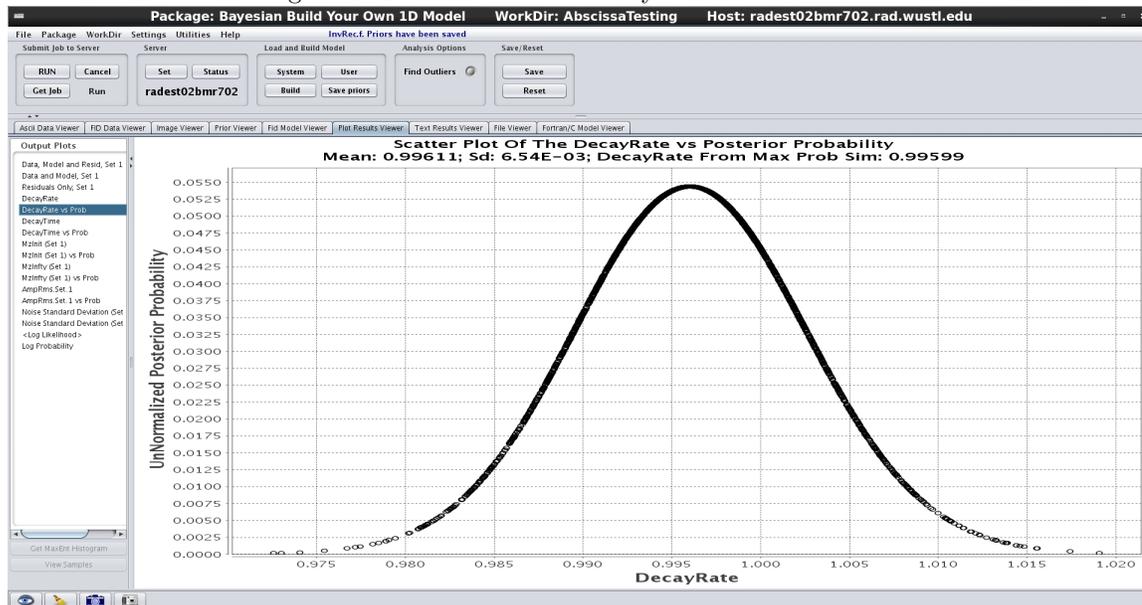


Figure 3.28: In addition to plotting the posterior probability for the parameter the unnormalized posterior probability for the parameter is plotted against the parameter value used in evaluating the posterior probability. As illustrated here, for well peaked posterior probabilities, these samples will look almost exactly like the binned histogram, Fig. (3.25).

### 3.5.5 Probability Vs Parameter Samples plot

In release 4.10 a new type of plot was added, see Fig. (3.28). For each plot of the posterior probability for a parameter, there is an addition scatter plot of the parameter versus the posterior probability. If the posterior probability for a given parameter is highly symmetric, having a well defined maximum, then this new plot, Fig. (3.28), will look almost exactly like the plot shown in Fig. (3.25). In this new plot, the horizontal axis is the value of the parameter, while the vertical axis is the unnormalized posterior probability. When the Markov chain Monte Carlo simulations are running, it is the logarithm of the posterior probability for the parameters that is computed. To obtain the unnormalized posterior probability shown in this plot, we locate the Markov chain Monte Carlo simulation that had maximum log posterior probability. We renormalize the log posterior probability in each Markov chain Monte Carlo simulation by subtracting this maximum log posterior probability from the log posterior probability in each simulation. This renormalization cancels when we normalize the posterior probabilities. It ensures that when we exponentiate these log posterior probabilities the result is always between zero and one. Finally, the samples for a given parameter are rescaled so that the sum over all samples is one, Figure 3.25. We also generate a scatter plot of the parameter vs the posterior probability from the Markov chain Monte Carlo samples. When the posterior probability for a parameter is well peaked, this plot is almost identical to the plot shown in Fig. (3.25). However, when the posterior probability is not well peaked, these plots can be very different from Fig. (3.25).

One of the reasons this plot was added was to help eliminate a common confusion. In the

Figure 3.29: Posterior Probability Vs Parameter Value, A Skewed Example

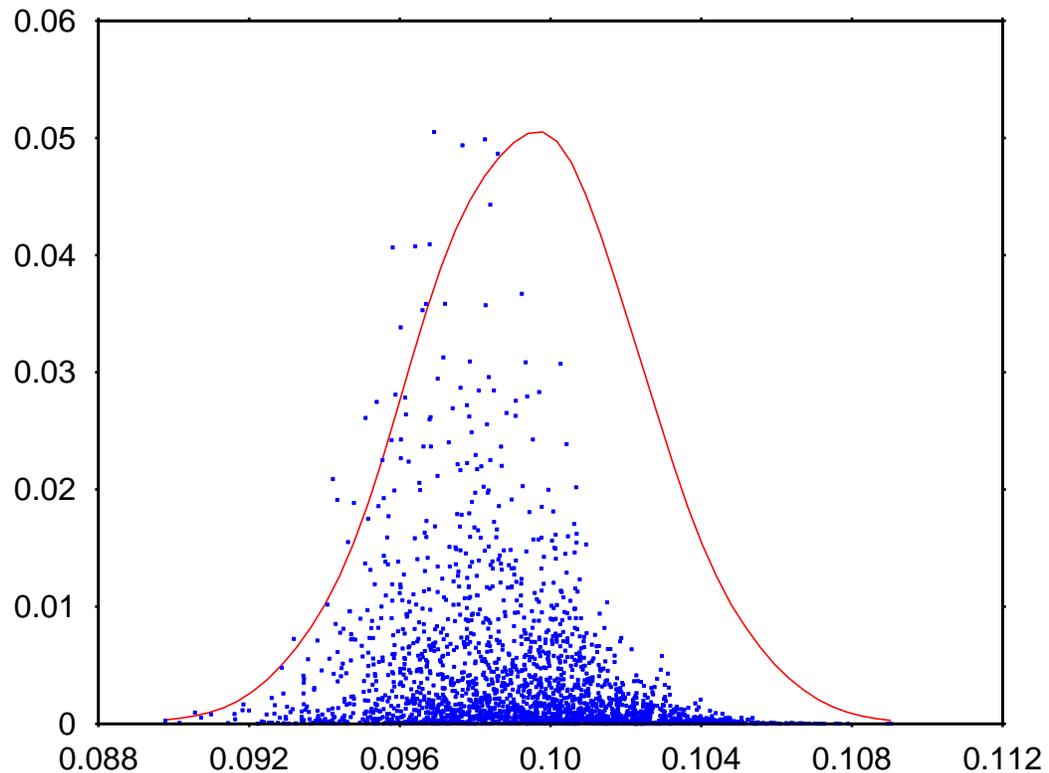


Figure 3.29: When the peak of the posterior probability is not symmetric or only weakly a function of a given parameter, then it is possible for the parameters of the simulation that had maximum posterior probability to be significantly different from the mean of the Markov chain Monte Carlo samples. This is illustrated here, the peak of the histogram is rather different from the sample that had maximum posterior probability.

heading of the plots of the posterior probability, part of the heading reads “"Parameter Name" From Max Prob Sim: "xxxx"” where "Parameter Name" is the name of the parameter being plotted, for example in the exponential package it would read "Rate.1" and "xxxx" is the value of the parameter for which the posterior probability was maximum, in the example shown here, this value is 0.99599. For plots like the one shown, the peak in the histogram and the parameter that maximized the posterior probability are essentially the same. However, it is possible for the parameter value at the peak of the histogram, and the parameters from the simulation having maximum posterior probability to be very different. This is illustrated in Fig. 3.29. The resulting smoothed binned histogram and the samples have been plotted on the same scale. The histogram is shown as the line in this plot and the samples are the individual dots. In this example, the peak of the histogram is about 0.1, while the sample that had maximum posterior probability is somewhere around 0.094. So be warned, the peak in the histogram and the sample that had maximum posterior probability can be very different and using this new plot, along with Fig. (3.25), you will be able to see both

the peak in the histogram and the sample that had peak posterior probability.

### 3.5.6 Expected Log Likelihood Plot

A plot of the expected logarithm of the likelihood as a function of the annealing parameters is shown in Fig. 3.30. When the Markov chain Monte Carlo simulations start, the annealing parameter, labeled Beta, starts at zero and is increased to one according to some annealing schedule. The Markov chain Monte Carlo simulations are being done using simulated annealing. In simulated annealing one raises the likelihood to a power  $\beta$ . If the parameters of interest are designated by  $M$ , and the data as  $D$ , then logarithm of the posterior probability is given by

$$\log P(M|DI) = \log P(M|I) + \beta \log P(D|MI) \quad (3.8)$$

where  $P(M|DI)$  is the posterior probability for the parameters given the data and the prior information,  $P(M|I)$  is the prior probability for the parameters,  $\beta$  is the annealing parameter and  $P(D|MI)$  is the direct probability for the data given the parameters  $M$  and the prior information  $I$  and in this discussion it is called a likelihood. When the annealing parameter is zero, only the logarithm of the prior probability contributes to the calculation and the simulations explore the prior probability. Because the data are not being used, the fit to the data is very bad and, consequently, the expected logarithm of the likelihood starts out at a very low number. As the annealing parameter is increased, the logarithm of the likelihood contributes more and more to the calculation and so the expected logarithm of the likelihood increases coming to a maximum when the model has fit the data as well as possible. All the packages that run Markov chain Monte Carlo simulations run multiple simulations in parallel. The expected logarithm of the likelihood shown in Fig. 3.30 is the average of logarithm of the likelihood over all of these simulations. See Subsection B for a more detailed discussion of simulated annealing and Markov chain Monte Carlo.

### 3.5.7 Scatter Plots

There are two additional plot types that need to be discussed: scatter plots and the logarithm of the posterior probability plot. Scatter plots are generally toward the bottom of the output plot list and have names of the form “parameter 1 vs parameter 2”, where parameter 1&2 are any two parameters from the simulations. An example scatter plot is shown in Fig. 3.31. This particular example was generated using the exponential package with a biexponential model plus a constant. Many samples are drawn from the joint posterior probability for the parameters. In the example shown here, there were 50 simulations and 30 samples were drawn from each simulation, so there are a total of 1500 samples. The scatter plot shown in Fig. 3.31 is of Rate\_1 vs Rate\_2. Each point in this plot is one of the 1500 simulations. The coordinates of each point are the values of the decay rate constants in a given simulation.

Scatter plots can look like an ellipsoid that is aligned with the axes, in which case the parameters are uncorrelated. Scatter plots can also be tilted, like the one shown in Fig. 3.31 in which case the parameters are correlated. Finally, scatter plots can take on highly irregular shapes and can even have sharp cutoffs when parameters have natural bounds. The scatter plot shown in Fig. 3.31 is an almost classical example of a scatter plot of two correlated parameters. Note the almost linear like upward scatter in this plot. This type of feature is indicative of a correlation between the sum and difference of the decay rate constants. The density of points in a scatter plot is a sample from the

Figure 3.30: The Expected Value Of The Logarithm Of The Likelihood

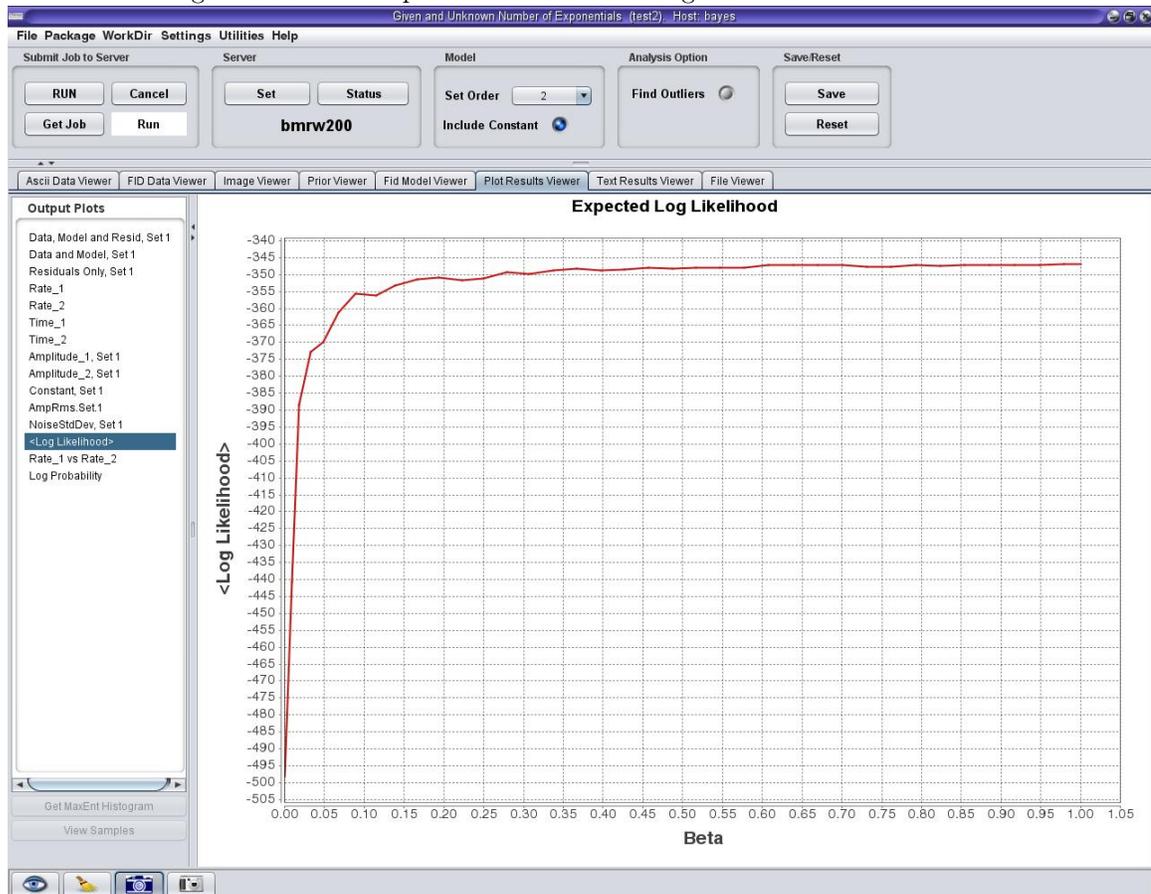


Figure 3.30: When the Markov chain Monte Carlo simulations run the intermediate results can be used to compute the logarithm of the posterior probability. Additionally, a plot of the expected logarithm of the likelihood can be produced. This plot can be used to aid one in determining if the Markov chain Monte Carlo simulations have converged.

Figure 3.31: The Scatter Plots

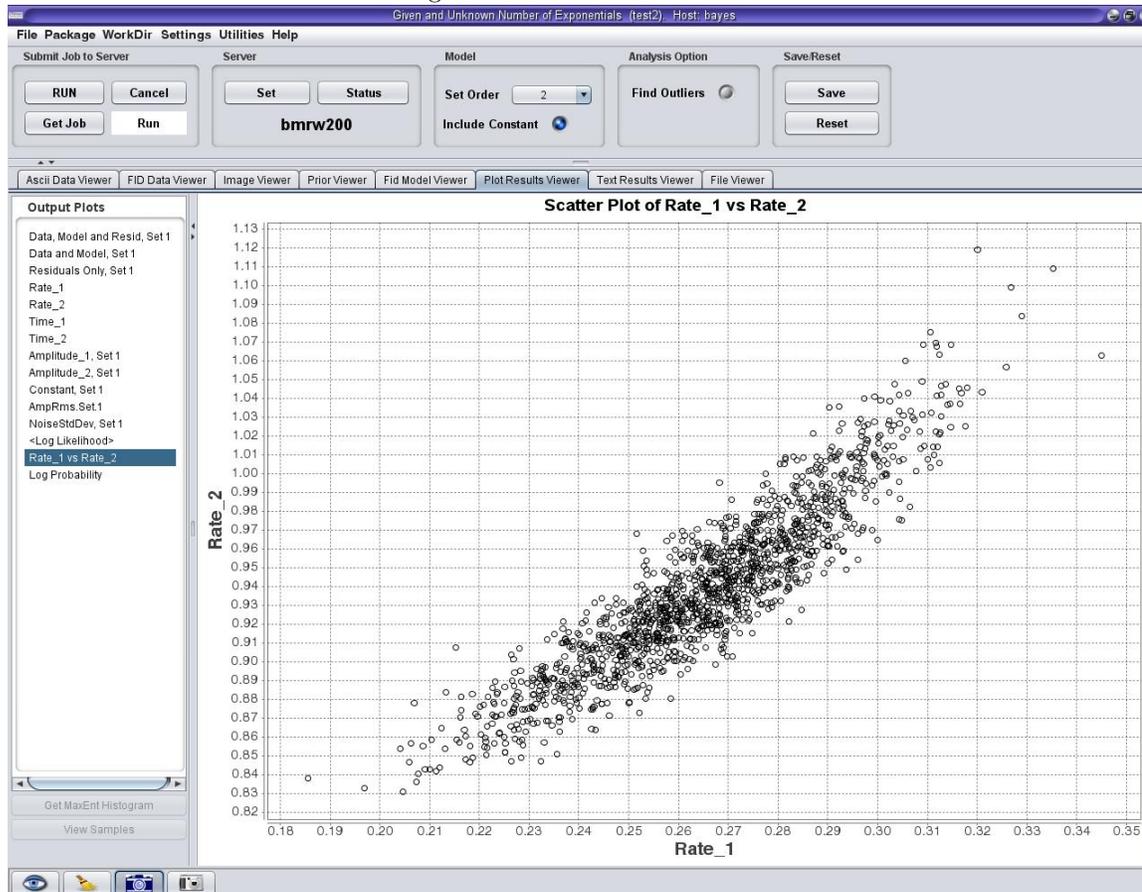


Figure 3.31: When the Markov chain Monte Carlo simulations run, many of the packages output scatter plots that are used to determine if the parameters are correlated and to check on the convergence of the simulations. Here is an example scatter plot created using the exponential package with biexponential data with a constant. The plot is of the two decay rate constants one versus the other. In this case there is a strong correlation between decay rate constants, but the simulations are otherwise well converged.

joint posterior probability for the decay rate constants. In Fig. 3.31 the scatter plot is elongated along the vector sum of the decay rate constants and it is contracted by about a factor of 3 along the vector difference. This implies that the sum of the decay rate constants is not well determined compared to the difference. This type of correlation is very common among models containing multiple exponentials.

Not all packages produce scatter plots and even packages that do generate scatter plots do not always generate scatter plots of all of the parameters. For example here, there are three amplitudes and two decay rate constants in the model, but only a scatter plot of the decay rate constants was output. That's because the exponential package uses a marginal posterior probability and the amplitudes are not varied by the Markov chain Monte Carlo simulations. Scatter plots are only produced for parameters that are varied by the simulations. Additionally, some packages have a very large number of parameters and because the number of scatter plots increases like the square of the number of parameters, scatter plots are typically not output when the number of these plots would be very large.

### 3.5.8 Logarithm of the Posterior Probability Plot

The last plot that we are going to discuss is shown in Fig. 3.32. This plot is a bit of a mess, and generally speaking, the messier it is, the better. In the Markov chain Monte Carlo simulations used by the Bayesian Analysis software multiple chains are run in parallel. In the annealing phase, the simulations are not in equilibrium. However, when one reaches the sample gathering phase, the simulations are should be in equilibrium. Here, equilibrium means that as the Markov chain Monte Carlo simulations are run the details of each simulation change, but the expected values of the parameters, probabilities, likelihoods etc. remain constant, i.e., the system is at a static equilibrium. In the literature this condition is often called detailed balance.

In the sample gathering phase, the simulations are run through a predetermined number of steps and then each simulation is save. This process is repeated until all of the user specified repeats have been gathered. Each line in the plot shown in Fig. 3.32 is the logarithm of the posterior probability for one of the simulations as a function of repeat number. As the simulations are run, the posterior probability for a given simulation is increasing and decreasing as a function of the repeat number. When all of the simulations are doing this, the simulations are often said to be mixing. Indeed in this plot, the simulations are mixing so well that it is impossible to follow the trajectory of a single simulation. However, note there is a fairly sharp maximum that bounds the posterior probability. This bound reflects the fact that the model can only fit the data so well. Individual simulations come up to the maximum and then move away from it. Here, the size of the deviations from this maximum are on the order of a few e-folding, with the maximum deviation being about 10 and a typical deviation being more like one or two. The size of the deviations and the presence of a sharp boundary are both data and model dependent. However, a good rule of thumb concerning the size of these deviations is that each parameter in the model can, on average, cause a deviation of one and perhaps two e-folding and if there are 2 parameters then an average deviation would be 2 to 4 e-folding. Models with more parameters will deviate from the maximum by roughly the number of parameters times one or two. Large deviations caused by a single parameter are hard to obtain simply because such a large deviation in the posterior probability is rejected by the Markov chain.

Figure 3.32: The Logarithm Of The Posterior Probability By Repeat Plot

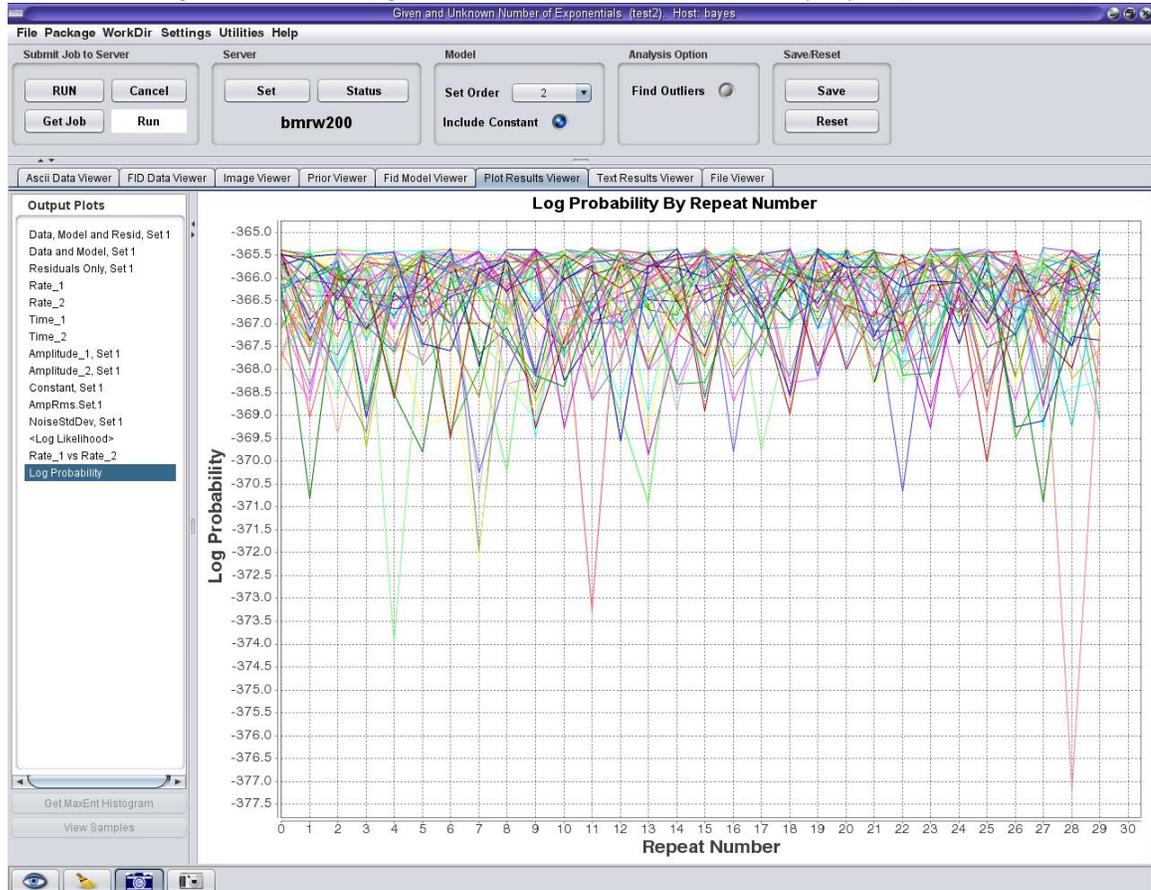


Figure 3.32: When the simulated annealing phase is finished for all of the simulations, the program that run the simulations begin gathering samples of each of these simulations. The program tries to gather uncorrelated, i.e., independent samples. If the user specified 50 simulations and 30 samples, the the program tries to gather 30 independent samples of the 50 simulations. The shown here is the logarithm of the posterior probability for each simulating plotted by sample number.

### 3.5.9 Fortran/C Code Viewer

The Fortran/C Code Viewer is shown in Fig. 3.33, it is used to view and or modify the Ascii models you have loaded. When a model is loaded using either the “System” or “User” buttons a local copy of the model is stored in the BayesAsciiModels subdirectory of your current Bayes home directory and it is displayed in in the Fortran/C Model Viewer along with a list of all of the currently loaded models, left panel. This viewer contains 5 total widgets, and here is a description of them:

**Ascii Models** is a list of all of the currently loaded Models. This list is a selection menu and by left mouse clicking on a model, the model will be displayed by the Fortran/C Model Viewer. If multiple models are loaded, then clicking on each model displays that model.

**Remove Selected Model** will delete the model from you Bayes Home/BayesAsciiModels Subdirectory. Note that when system models are loaded, they are copied to the BayesAsciiModels Subdirectory and removing them will remove them from the Subdirectory, it will not remove them from your system directory.

**Edit/Create New Model** will open the current model in an editor and allow you to make changes to the model. The modified model can be saved using the current name or it can be renamed. For a description of how to write Fortran/C models, see Section E.

**Code** will display the source code of the currently selected model. Note in this viewer neither the model nor the parameters is editable. However, the priors can be changed on the Prior Viewer and if you activate the Edit/Create New Models button both the model and the parameters can be modified in the popup window. For a description of how to write Fortran/C models, see Section E.

**Parameters** will display the parameters file associated with this model. See the above comment on the code.

#### 3.5.9.1 Fortran/C Model Viewer Popup Editor

To edit a model one activates the “Edit/Create New Model” button in the lower left-hand part of the Fortran/C model viewer. When activated the code is copied into a work file and that file is displayed in the Fortran/C model editor shown in Fig 3.5.9.1. On this edit window you can modify the number of parameters, delete/add derived and change the number of model vectors. Here is a rough description of how to do these things:

**Create/Edit Model** contains the name of the current model. If you wish to create a new model, simply change the name of the model in this field. To save this model you must hit the “Save and Load” button.

**Abscissa** can be used to set the number of abscissa columns. Remember if you change the number of abscissa columns the code will probably need to be modified to accommodate this change.

**Data Columns** can be used to change the number of data columns. Remember if you change the number of data columns the code will need to be modified to accommodate this change.

**Model Vectors** can only be changed indirectly. This number is a count of the number of parameters having a parameter type of “Amplitude”. Consequently, to change the number of model vectors you must change the number of amplitudes.

Figure 3.33: The Fortran/C Model Viewer

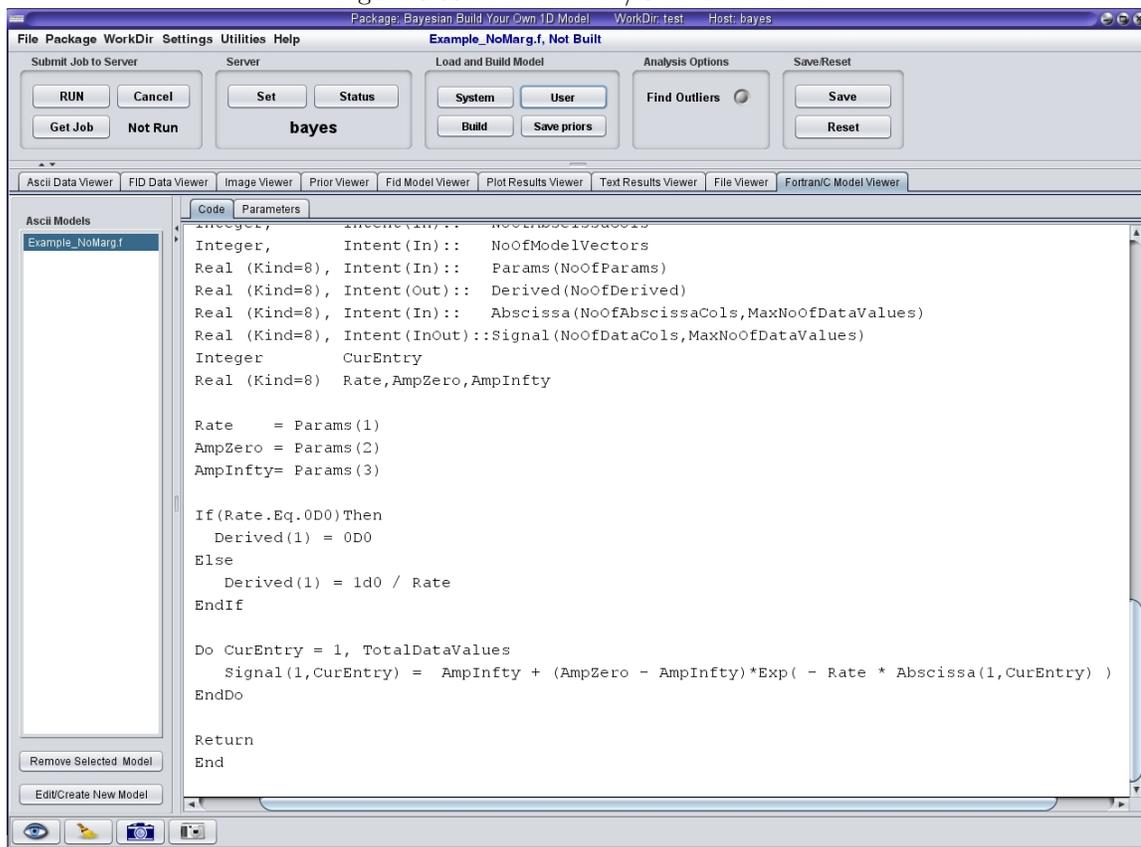


Figure 3.33: The Fortran/C Model Viewer shown here is used to view the models you have currently loaded. You can change a model by selecting it from the “Ascii models” list on the left. The two buttons “Code” and “Parameters” will display the code or the parameters respectively. As a reminder the code is either the Fortran or C code used to implement a model and the parameters file contains essentially a description of the prior probabilities for the model. The display produced by these two buttons are simple displays and neither the code nor the parameters are changeable on this display. However, when the “Edit/Create New Model” button is activated The code is displayed in a popup in a popup editing window. Finally, the “Remove Selected Model” can be used to delete the model from the your BayesAsciiModels subdirectory. Note that activating this button removes your local copy of the model, it does not remove the model from the system directory.

Figure 3.34: The Fortran/C Code Editor

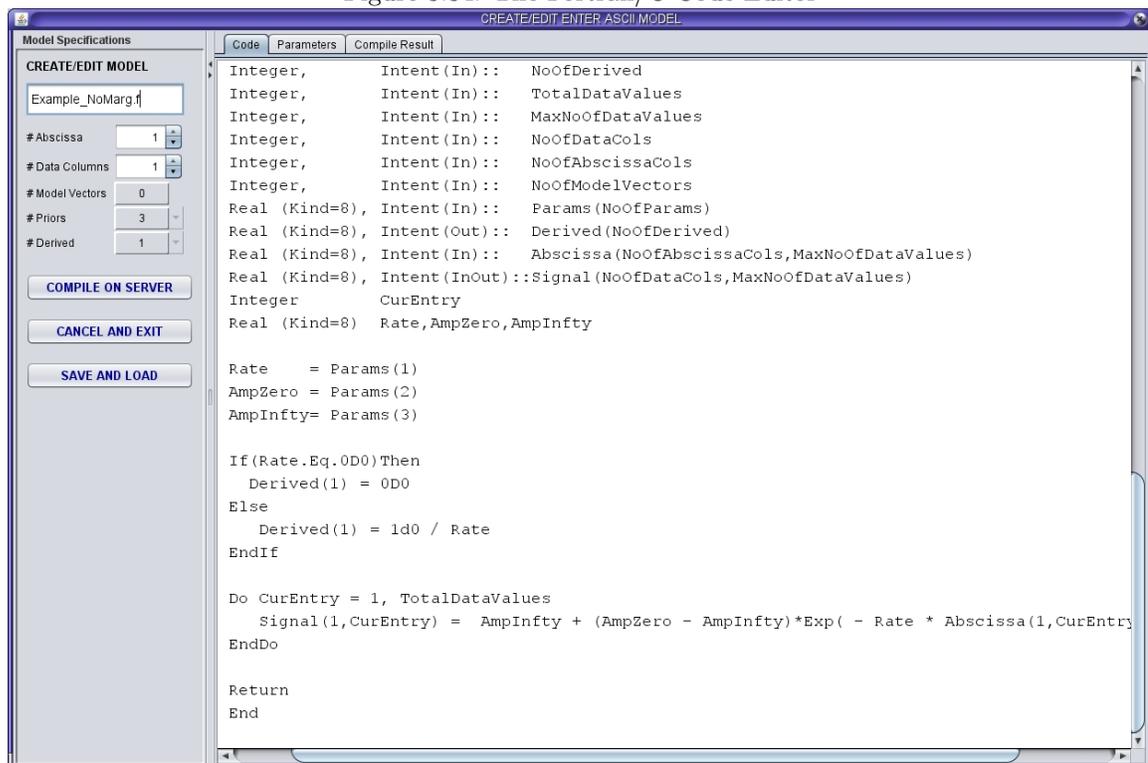


Figure 3.34: The Fortran/C Code Editor shown here is used to edit your currently selected model. In this popup window the text can be modified in any way you please. You can add parameters, change the number of model vectors as well as edit the parameters and the model.

**Priors** is a display text that indicates the current number of priors. To change the number of priors activate the grayed out down arrow on the right side of the prior count label. When activated this pull down menu has three options:

**Add New Parameter** when activated will popup a window asking you for the name of the new parameter. Enter the name of the parameter in the popup and hit OK. The parameter will be added to the list of parameters in the model. However, the prior will be filled in with zero and you must set the prior accordingly. To set the prior simply click on the prior name in the list of priors, this will display the prior in the area on the top of this window. Change the values as you see fit. Note the values are updated as you make the changes.

**Remove All** will remove all of the priors from this model. Note you are not prompted to prevent you from making a mistake, the priors are simply removed. If this is an error, then hit "Cancel and Exit" to abort your efforts and try again.

**Remove parameter** will display a selection list and you may pick out the individual prior to be removed.

**Derived** widget is identical in its behavior as the "Priors" widget. The derived label contains the current number of derived parameters and the grayed out down arrow on the right side allows you to add, remove and remove derived parameters.

**Compile On Server** will send the current version of the model to the server to compile. Note that this is a simple compile and it does not save the model in your BayesAsciiModels subdirectory.

**Cancel And Exit** will cancel you current modifications and the Create Enter Ascii model window will exit. Note that if you had activated the "Save and Load" button those changes will still be in effect when cancel is activated.

**Save and Load** will save your current modifications and the reload the model so that when you exit the popup model editing window your changes are ready to be run.

Along the top of the popup Edit/Create Ascii model are three additional widgets. here is a description of their function

**Code** will display the current code including any modifications you have made to the code. The window displaying the code is a simple editor and you can make changes to the code as you see fit.

**Parameters** will display the current parameter file any modifications you have made to the parameters.. The window displaying the code is a simple editor and you can make changes to the code as you see fit. The window displaying the parameters is not quite a edit window, however you can use the parameter window to edit the parameters. First, you can change the parameter being displayed along the top in two ways. You can simply left-mouse click on the parameter you wish to view or you the down arrow on the right-hand side of the displayed parameter name can be used to select a parameter. Regardless, after selecting a parameter, you can use change any value displayed in the top line. As you change these values the parameter file is automatically updated. There are about nine different fields displayed for each parameter, here is a brief list of their functions:

**Name** is the name of the current parameter. You can simply enter another name if you wish to change this. Note that names must be unique within a model.

**Low** is the lowest value the parameter can take on. Simply enter any value you wish to change the low value. Note that you can enter any value in this low field, including something larger than the high and the popup will not complain. However, such errors must be corrected before you will be able to save the prior.

**Mean** is the mean value of the Gaussian prior, and this field is also used as the peak value in a positive prior. Note that this editor does not change the labels on the prior fields when the prior types are changed.

**High** is the highest value this parameter can take on and is used by all priors.

**Sdev** is an abbreviation for Standard Deviation and is used on Gaussian Prior probabilities.

**Prior Type** is a pull down menu that allows you to select the prior type.

**Edit (type)** sets whether or not the prior type can be edited by the user. The default allows the field to be edited.

**Order** indicates if this parameter must be ordered or not. The default is not to order the parameters. If the parameters are to be ordered, they can be ordered from low to high or high to low.

**Edit (order)** indicates if the order parameter can be modified or not. Usually when a parameter is ordered, this cannot be safely change. Safely change in the sense that the model will continue to work correctly. Consequently, we default this to not editable, but allow the user to change this field at his own peril.

**Non-Linear** is the default parameter type, this selection menu can be used to set the parameter to amplitude or parameter.

**Compile Results** will redisplay the results from the previous compile.



## Chapter 4

# An Introduction to Bayesian Probability Theory

This Chapter is a tutorial on Bayesian probability theory. In it the procedures and principles needed to apply probability theory as extended logic will be discussed in detail. Primarily these procedures and principles will be illustrated using an example taken from NMR, the single frequency estimation problem. In this example we illustrate the assignment of probabilities and the use of uninformative prior probabilities. While we attempt to explain all of the steps in this calculation in detail, some familiarity with higher mathematics and Bayesian probability theory is assumed. For an introduction to probability theory see [32, 64, 66, 40]; for a derivation of the rules of probability theory see Jaynes [32, 31], and for an introduction to parameter estimation using probability theory see Bretthorst [3]. In this tutorial the sum and product rules of probability theory will be given and no attempt will be made to derive them. However, if one wishes to represent degrees of belief as real numbers, reason consistently, and have probability theory reduce to Aristotelian logic when the truth of the hypotheses are known, then the sum and product rules are the unique rules for conducting inference. For an extensive discussion of these points and much more, see Jaynes [32].

### 4.1 The Rules of Probability Theory

There are two basic rules for manipulating probabilities, the product rule and the sum rule; all other rules may be derived from these. If  $A$ ,  $B$ , and  $C$  stand for three hypotheses, then the product rule states

$$P(AB|C) = P(A|C)P(B|AC), \quad (4.1)$$

where  $P(AB|C)$  is the joint probability that “ $A$  and  $B$  are true given that  $C$  is true,”  $P(A|C)$  is the probability that “ $A$  is true given  $C$  is true,” and  $P(B|AC)$  is the probability that “ $B$  is true given that both  $A$  and  $C$  are true.” The notation “ $|C$ ” means conditional on the truth of hypothesis  $C$ . In probability theory *all* probabilities are conditional. The notation  $P(A)$  is not used to stand for the probability for a hypothesis, because it does not make sense until the evidence on which it is based is given. Anyone using such notation either does not understand that all knowledge is conditional, i.e., contextual, or is being extremely careless with notation. In either case, one should

be careful when interpreting such material. For more on this point see Jeffreys [33] and Jaynes [32].

In Aristotelian logic, the hypothesis “ $A$  and  $B$ ” is the same as “ $B$  and  $A$ ,” so the numerical value assigned to the probabilities for these hypotheses must be the same. The order may be rearranged in the product rule, Eq. (4.1), to obtain:

$$P(BA|C) = P(B|C)P(A|BC), \quad (4.2)$$

which may be combined with Eq. (4.1) to obtain a seemingly trivial result

$$P(A|BC) = \frac{P(A|C)P(B|AC)}{P(B|C)}. \quad (4.3)$$

This is Bayes’ theorem. It is named after Rev. Thomas Bayes, an 18th century mathematician who derived a special case of this theorem. Bayes’ calculations [1] were published in 1763, two years after his death. Exactly what Bayes intended to do with the calculation, if anything, still remains a mystery today. However, this theorem, as generalized by Laplace [36], is the basic starting point for inference problems using probability theory as logic.

The second rule of probability theory, the sum rule, relates to the probability for “ $A$  or  $B$ .” The operation “or” is indicated by a “+” inside a probability symbol. The sum rule states that given three hypotheses  $A$ ,  $B$ , and  $C$ , the probability for “ $A$  or  $B$  given  $C$ ” is

$$P(A + B|C) = P(A|C) + P(B|C) - P(AB|C). \quad (4.4)$$

If the hypotheses  $A$  and  $B$  are mutually exclusive, that is the probability  $P(AB|C)$  is zero, the sum rule becomes:

$$P(A + B|C) = P(A|C) + P(B|C). \quad (4.5)$$

The sum rule is especially useful because it allows one to investigate an interesting hypothesis while removing an uninteresting or nuisance hypothesis from consideration.

To illustrate how to use the sum rule to eliminate nuisance hypotheses, suppose  $D$  stands for the data,  $\omega$  the hypothesis “the frequency of a sinusoidal oscillation was  $\omega$ ,” and  $B$  the hypothesis “the amplitude of the sinusoid was  $B$ .” Now suppose one wishes to compute the probability for the frequency given the data,  $P(\omega|D)$ , but the amplitude  $B$  is present and must be dealt with. The way to proceed is to compute the joint probability for the frequency and the amplitude given the data, and then use the sum rule to eliminate the amplitude from consideration. Suppose, for argument’s sake, the amplitude  $B$  could take on only one of two mutually exclusive values  $B \in \{B_1, B_2\}$ . If one computes the probability for the frequency and ( $B_1$  or  $B_2$ ) given the data one has

$$P(\omega|D) \equiv P(\omega[B_1 + B_2]|D) = P(\omega B_1|D) + P(\omega B_2|D). \quad (4.6)$$

This probability distribution summarizes all of the information in the data relevant to the estimation of the frequency  $\omega$ . The probability  $P(\omega|D)$  is called the marginal probability for the frequency  $\omega$  given the data  $D$ .

The marginal probability  $P(\omega|D)$  does not depend on the amplitudes at all. To see this, the product rule is applied to the right-hand side of Eq. (4.6) to obtain

$$P(\omega|D) = P(B_1|D)P(\omega|B_1D) + P(B_2|D)P(\omega|B_2D) \quad (4.7)$$

but

$$P(B_1|D) + P(B_2|D) = 1 \quad (4.8)$$

because the hypotheses are exhaustive. So the probability for the frequency  $\omega$  is a weighted average of the probability for the frequency given that one knows the various amplitudes. The weights are just the probability that each of the amplitudes is the correct one. Of course, the amplitude could take on more than two values; for example if  $B \in \{B_1, \dots, B_m\}$ , then the marginal probability distribution becomes

$$P(\omega|D) = \sum_{j=1}^m P(\omega B_j|D), \quad (4.9)$$

provided the amplitudes are mutually exclusive and exhaustive. In many problems, the hypotheses  $B$  could take on a continuum of values, but *as long as only one value of  $B$  is realized when the data were taken* the sum rule becomes

$$P(\omega|D) = \int dB P(\omega B|D) \quad (4.10)$$

where probabilities that have had one or more parameters removed by integration are frequently called marginal probabilities. Note that the  $B$  inside the probability symbols refers to the hypothesis; while the  $B$  appearing outside of the probability symbols is a number or index. A notation could be developed to stress this distinction, but in most cases the meaning is apparent from the context.

The sum and integral appearing in Eqs. (4.9,4.10) are over a set of mutually exclusive and exhaustive hypotheses. If the hypotheses are not mutually exclusive, one simply uses Eq. (4.4). However, if the hypotheses are *not* exhaustive, the sum rule *cannot* be used to eliminate nuisance hypotheses. To illustrate this, suppose the hypotheses,  $B \in \{B, \dots, B_m\}$ , are mutually exclusive, but not exhaustive. The hypotheses  $B$  could represent various explanations of some experiment, but it is always possible that there is something else operating in the experiment that the hypotheses  $B$  do not account for. Let us designate this as

$$\text{SE} \equiv \text{“Something Else not yet thought of.”} \quad (4.11)$$

The set of hypotheses  $\{B, \text{SE}\}$  is now complete, so the sum rule may be applied. Computing the probability for the hypothesis  $B_i$  conditional on some data  $D$  and the information  $I$ , where  $I$  stands for the knowledge that amplitudes  $B$  are not exhaustive, one obtains

$$P(B_i|DI) = \frac{P(B_i|I)P(D|B_iI)}{P(D|I)} \quad (4.12)$$

and for SE

$$P(\text{SE}|DI) = \frac{P(\text{SE}|I)P(D|\text{SE}I)}{P(D|I)}. \quad (4.13)$$

The denominator is the same in both these equations and is given by

$$\begin{aligned} P(D|I) &= \sum_{i=1}^m P(DB_i|I) + P(D\text{SE}|I) \\ &= \sum_{i=1}^m P(B_i|I)P(D|B_iI) + P(\text{SE}|I)P(D|\text{SE}I). \end{aligned} \quad (4.14)$$

But this is indeterminate because SE has not been specified, and therefore the likelihood,  $P(D|\text{SE}I)$ , is indeterminate even if the prior probability  $P(\text{SE}|I)$ , is known. However, the relative probabilities

$P(B_i|DI)/P(B_j|DI)$  are well defined because the indeterminacy cancels out. So there are two choices: either *ignore* SE and thereby assume the hypotheses  $B$  are complete or *specify* SE, thereby completing the set of hypotheses.

## 4.2 Assigning Probabilities

The product rule and the sum rule are used to indicate relationships between probabilities. These rules are not sufficient to conduct inference because, ultimately, the “numerical values” of the probabilities must be known. Thus the rules for manipulating probabilities must be supplemented by rules for assigning numerical values to probabilities. The historical lack of these supplementary rules is one of the major reasons why probability theory, as formulated by Laplace, was rejected in the late part of the 19th century. To assign any probability there is ultimately only one way, logical analysis, i.e., non-self-contradictory analysis of the available information. The difficulty is to incorporate only the information one actually possesses without making gratuitous assumptions about things one does not know. A number of procedures have been developed that accomplish this task: Logical analysis may be applied directly to the sum and product rules to yield probabilities [32]. Logical analysis may be used to exploit the group invariance of a problem [27]. Logical analysis may be used to ensure consistency when uninteresting or nuisance parameter are marginalized from probability distributions [29]. And last, logical analysis may be applied in the form of the principle of maximum entropy to yield probabilities [66, 27, 28, 60, 59]. Of these techniques the principle of maximum entropy is probably the most powerful, and in this tutorial it will be used to assign most probabilities.

In this tutorial there are three different types of information that must be incorporated into probability assignments: parameter ranges, knowledge of the mean and standard deviation estimates of several quantities, and some properties of the noise. Their assignment differs only in the types of information available. In the first case, the principle of maximum entropy leads to a bounded uniform prior probability. In the second and third cases, it leads to a Gaussian probability distribution. To understand the principle of maximum entropy and how these probability assignments come about, suppose one must assign a probability distribution for the  $i$ th value of a parameter given the information  $I$ . This probability is denoted  $P(i|I)$  ( $1 \leq i \leq m$ ). The Shannon entropy, defined as

$$H \equiv - \sum_{i=1}^m P(i|I) \log P(i|I), \quad (4.15)$$

is a measure of the amount of ignorance (uncertainty) in this probability distribution [58]. Shannon’s entropy is based on a qualitative requirement, the entropy should be monotonically increasing for increasing ignorance, plus the requirement that the measure be consistent. The principle of maximum entropy then states that if one has some information  $I$ , one can assign the probability distribution,  $P(i|I)$ , that contains only the information  $I$  by maximizing  $H$  subject to the information (constraints) represented by  $I$ . Because  $H$  measures the amount of ignorance in the probability distribution, assigning a probability distribution that has maximum entropy yields a distribution that is least informative (maximally ignorant) while remaining consistent with the information  $I$ : the probability distribution,  $P(i|I)$ , contains only the information  $I$ , and does not contain any additional information not already implicit in  $I$  [60, 59].

To demonstrate its use, suppose that one must assign  $P(i|I)$  and nothing is known except that

the set of hypotheses is mutually exclusive and exhaustive. Applying the sum rule one obtains

$$\sum_{i=1}^m P(i|I) = 1. \quad (4.16)$$

This equation may be written

$$\sum_{i=1}^m P(i|I) - 1 = 0 \quad (4.17)$$

and because this equation sums to zero, any multiple of it may be added to the entropy of  $P(i|I)$  without changing its value:

$$H = - \sum_{i=1}^m P(i|I) \log P(i|I) + \beta \left[ 1 - \sum_{i=1}^m P(i|I) \right]. \quad (4.18)$$

The constant  $\beta$  is called a Lagrange multiplier. But the probabilities  $P(i|I)$  and the Lagrange multiplier  $\beta$  are not known; they must be assigned. To assign them,  $H$  is constrained to be a maximum with respect to variations in all the unknown quantities. This maximum is located by differentiating  $H$  with respect to both  $P(k|I)$  and  $\beta$ , and then setting the derivatives equal to zero. Here there are  $m$  unknown probabilities and one unknown Lagrange multiplier. But when the derivatives are taken, there will be  $m + 1$  equations; thus all of the unknowns may be determined. Taking the derivative with respect to  $P(k|I)$ , one obtains

$$\log P(k|I) + 1 + \beta = 0, \quad (4.19)$$

and taking the derivative with respect to  $\beta$  returns the constraint equation

$$1 - \sum_{i=1}^m P(i|I) = 0. \quad (4.20)$$

Solving this system of equations, one finds

$$P(i|I) = \frac{1}{m} \quad \text{and} \quad \beta = \log m - 1. \quad (4.21)$$

When nothing is known except the specification of the hypotheses, the principle of maximum entropy reduces to Laplace's principle of indifference [36]. But the principle of maximum entropy is much more general because it allows one to incorporate many different types of information.

As noted earlier, in the inference problem addressed in this chapter, there are three different types of information to be incorporated into probability assignments. The specification of parameter ranges occurs when the prior probabilities for various location parameters appearing in the calculation must be assigned. (A location parameter is a parameter that appears linearly in the model equation.) For these location parameters, the principle of maximum entropy leads to the assignment of a bounded uniform prior probability. However, care must be taken because most of these parameters are continuous and *the rules and procedures given in this tutorial are strictly valid only for finite, discrete probability distributions*. The concept of a probability for a hypothesis containing a continuous parameter, a probability density function, only makes sense when thought of as a limit. If the

preceding calculations are repeated and the number of hypotheses are allowed to grow infinitely, one will automatically arrive at a valid result as long as all probabilities remain finite and normalized. Additionally, the direct introduction of an infinity into any mathematical calculation is ill-advised under any conditions. Such an introduction presupposes the limit already accomplished and this procedure will cause problems whenever any question is asked that depends on how the limit was taken. For more on the types of problems this can cause see Jaynes [29], and for a much more extensive discussion of this point see Jaynes [32]. As it turns out, continuous parameters are not usually a problem, provided one always uses normalized probabilities. In this tutorial, continuous parameters will be used, but their prior probabilities will be normalized and the prior ranges will never be allowed to go to infinity without taking a limit.

The second type of information that must be incorporated into a probability assignment is knowledge of the mean and standard deviation of a parameter estimate. It is a straightforward exercise to show that, in this case, the principle of maximum entropy leads to a Gaussian distribution.

The third type of information that must be incorporated into a probability assignment is information about the errors or noise in the data. The probability that must be assigned is denoted  $P(D|LI)$ , the probability for the data given that the signal is  $L$ , where the data,  $D$ , is a joint hypothesis of the form,  $D \equiv \{d_1 \dots d_N\}$ ;  $d_j$  are the individual data items, and  $N$  is the number of data values. If the signal  $L$  is given time  $t_j$ , then

$$d_j - L(t_j) = n_j \quad (4.22)$$

assuming that the noise is additive, and  $n_j$  is the misfit between the data and the model and is usually called the noise. Thus the probability for the data can be assigned if one can assign a probability for the noise.

To assign a probability for the noise, the question one must ask is, *what properties of the noise are to be used in the calculations?* For example, should the results of the calculations depend on correlations? If so, which of the many different types of correlations should the results depend on? There are second order correlations of the form

$$\rho'_s = \frac{1}{N-s} \sum_{j=1}^{N-s} n_j n_{j+s}, \quad (4.23)$$

where  $s$  is a measure of the correlation distance, as well as third, fourth, and higher order correlations. In addition to correlations, should the results depend on the moments of the noise? If so, on which moments should they depend? There are many different types of moments. There are power law moments of the form

$$\sigma'_s = \frac{1}{N} \sum_{j=1}^N n_j^s, \quad (4.24)$$

as well as moments of arbitrary functions, and a host of others.

The probability that must be assigned is the probability that one should obtain the data  $D$ , but from Eq. (4.22) this is just the probability for noise  $P(e_1 \dots e_N|I')$ , where  $e_j$  stands for a hypothesis of the form “the value of the noise at time  $t_j$  was  $e_j$ , when the data were taken.” The quantity  $e_j$  is an index that ranges over all valid values of the errors; while the probability for the noise,  $P(e_1 \dots e_N|I')$ , assigns a reasonable degree of belief to a particular set of noise values. For the

probability for the noise to be consistent with correlations it must have the property that

$$\rho_s = \langle e_j e_{j+s} \rangle \equiv \frac{1}{N-s} \sum_{j=1}^{N-s} \int de_1 \cdots de_N e_j e_{j+s} P(e_1 \cdots e_N | I') \quad (4.25)$$

and for it to be consistent with the power law moments it must have the additional property that

$$\sigma_s = \langle e^s \rangle \equiv \frac{1}{N} \sum_{j=1}^N \int de_1 \cdots de_N e_j^s P(e_1 \cdots e_N | I') \quad (4.26)$$

where the notation  $\langle \rangle$  denote mean averages over the probability density function.

In Eq. (4.23) and Eq. (4.24), the symbols  $\rho'_s$  and  $\sigma'_s$  were used to denote means or averages over the sample noise. These averages are the sample correlation coefficients and moments and they represent states of nature. In Eq. (4.25) and Eq. (4.26), the symbols  $\rho_s$  and  $\sigma_s$  are used to denote mean averages over the probability for the noise, and they represent states of knowledge. To use information in a maximum entropy calculation, that information must be known.

Assuming that none of these quantities are known, how can the principle of maximum entropy be used? Its use requires known information, and unless at least some of the  $\rho'_s$  and  $\sigma'_s$  are known, it would appear that maximum entropy cannot be used. However, this description of the problem is not what probability theory asks us to do. Probability theory asks us to assign  $P(e_1 \cdots e_N | I')$ , where  $I'$  represents the information on which this probability is based. Suppose for the sake of argument that that information is a mean,  $\nu$ , and standard deviation,  $\sigma$ , then what probability theory asks us to assign is  $P(e_1 \cdots e_N | \nu \sigma)$ . This expression should be read as the joint probability for all the errors given that the mean and standard deviation of the errors. According to probability theory, in the process of assigning the probability for the errors, we are to assume that both  $\nu$  and  $\sigma$  are known or given values. This is a very different state of knowledge from knowing that the mean and standard deviation of the sampling distribution are  $\nu$  and  $\sigma$ . If we happen to actually know these values, then there is less work to do when applying the rules of probability theory. However, if their values are unknown, we still seek the least informative probability density function that is consistent with a fixed or given mean and standard deviation. The rules of probability theory are then used to eliminate these unknown nuisance hypotheses from the final probability density functions.

But which of these constraints should be used? The answer was implied earlier by the way the question was originally posed: what *properties* of the errors are to be used in the calculations? The class of maximum entropy probability distributions is the class of all probability density functions for which sufficient statistics exist. A sufficient statistic is a function of the data that summarizes all of the information in the data relevant to the problem being solved. These sufficient statistics are the sample moments that correspond to the constraints that were used in the maximum entropy calculation. For example, suppose we used the first three correlation coefficients,  $\rho_1$ ,  $\rho_2$ , and  $\rho_3$ , as defined by Eq. (4.25) in a maximum entropy calculation, then the parameter estimates will depend only on the first three correlation coefficients of the data and our uncertainty in those estimates will depend on  $\rho_1$ ,  $\rho_2$ , and  $\rho_3$  if they are known, and on the first three correlation coefficients of the errors if  $\rho_1$ ,  $\rho_2$ , and  $\rho_3$  are not known. *All* other properties of the errors have been made irrelevant by the use of maximum entropy. So the real question becomes, what does one know about the errors before seeing the data? If there is information that suggests the errors may be correlated, then by all means a correlation constraint should be included. Additionally, if one has information that suggests the higher moments of the noise can deviate significantly from what one would expect from

a Gaussian distribution, then again a constraint on the higher moments should be included. But if one has no information about higher moments and correlations, then one is always better off to leave those constraints out of the maximum entropy calculation, because the resulting probability density function will have higher entropy. Higher entropy distributions are by definition less informative and therefore make more conservative estimates of the parameters. Consequently, these higher entropy probability density functions are applicable under a much wider variety of circumstances, and typically they are simpler and easier to use than distributions having lower entropy.

In assigning the probability density function for the errors, it will be assumed that our parameter estimates are to depend only on the mean and variance of the errors in the data. The appropriate constraints necessary are on the first and second moments of the probability density function. The constraint on the first moment is given by

$$\nu = \frac{1}{N} \sum_{j=1}^N \int de_1 \cdots de_N e_j P(e_1 \cdots e_N | I') \quad (4.27)$$

and by

$$\sigma^2 + \nu^2 = \frac{1}{N} \sum_{j=1}^N \int de_1 \cdots de_N e_j^2 P(e_1 \cdots e_N | I') \quad (4.28)$$

for the second moment, where  $\nu$  and  $\sigma^2$  are the fixed or given values of the mean and variance. Note the second moment of the probability distribution, Eq. (4.28), is written as  $\sigma^2 + \nu^2$ , to make the resulting probability density function come out in standard notation.

We seek the probability density function that has highest entropy for a fixed or given value of  $\sigma^2$  and  $\nu$ . To find this distribution Eq. (4.27) and Eq. (4.28) are rewritten so they sum to zero:

$$\nu - \frac{1}{N} \sum_{j=1}^N \int de_1 \cdots de_N e_j P(e_1 \cdots e_N | I') = 0, \quad (4.29)$$

and

$$\sigma^2 + \nu^2 - \frac{1}{N} \sum_{j=1}^N \int de_1 \cdots de_N e_j^2 P(e_1 \cdots e_N | I') = 0. \quad (4.30)$$

Additionally, the probability for finding the noise values somewhere in the valid range of values is one:

$$1 - \int de_1 \cdots de_N P(e_1 \cdots e_N | I') = 0. \quad (4.31)$$

Because Eq. (4.29) through Eq. (4.31), sum to zero, each may be multiplied by a constant and added

to the entropy of this probability density function without changing its value, one obtains

$$\begin{aligned}
 H &= - \int de_1 \cdots de_N P(e_1 \cdots e_N | I') \log P(e_1 \cdots e_N | I') \\
 &+ \beta \left[ 1 - \int de_1 \cdots de_N P(e_1 \cdots e_N | I') \right] \\
 &+ \delta \left[ \nu - \frac{1}{N} \sum_{j=1}^N \int de_1 \cdots de_N e_j P(e_1 \cdots e_N | I') \right] \\
 &+ \lambda \left[ \sigma^2 + \nu^2 - \frac{1}{N} \sum_{j=1}^N \int de_1 \cdots de_N e_j^2 P(e_1 \cdots e_N | I') \right]
 \end{aligned} \tag{4.32}$$

where  $\beta$ ,  $\delta$ , and  $\lambda$  are Lagrange multipliers. To obtain the maximum entropy distribution, this expression is maximized with respect to variations in  $\beta$ ,  $\delta$ ,  $\lambda$ , and  $P(e'_1 \cdots e'_N | I')$ . After a little algebra, one obtains

$$P(e_1 \cdots e_N | \nu\sigma) = (2\pi\sigma^2)^{-\frac{N}{2}} \exp \left\{ - \sum_{j=1}^N \frac{(e_j - \nu)^2}{2\sigma^2} \right\}, \tag{4.33}$$

where we have replaced  $I'$  by the information actually used in assigning this probability density function,

$$\lambda = \frac{N}{2\sigma^2}, \quad \delta = -\frac{N\nu}{\sigma^2}, \quad \text{and} \quad \beta = \frac{N}{2} \left[ \log(2\pi\sigma^2) + \frac{\nu^2}{\sigma^2} \right] - 1. \tag{4.34}$$

There are several interesting points to note about this probability density function. First, this is a Gaussian distribution. However, the fact that the prior probability for the errors has been assigned a Gaussian makes no statement about the sampling distribution of the errors; rather it says only that for a fixed value of the mean and variance the probability density function for the errors should be maximally uninformative and that maximally uninformative distribution happens to be a Gaussian. Second, this probability assignment apparently does not contain correlations. The reason for this is that a constraint on correlations must lower the entropy. By definition a probability assignment with lower entropy is more informative, and so must make more precise estimates of the parameters. Instead of saying this probability density function does not contain correlations, it would be more correct to say that this probability density function makes allowances for *every possible correlation* that could be present and so is less informative than correlated distributions. Third, if one computes the expected mean value of the moments, one finds

$$\langle e^s \rangle = \exp \left\{ -\frac{\nu^2}{2\sigma^2} \right\} \sigma^{2s} \frac{\partial^s}{\partial \nu^s} \exp \left\{ \frac{\nu^2}{2\sigma^2} \right\} \quad (s \geq 0) \tag{4.35}$$

which reduces to

$$\langle e^0 \rangle = 1, \quad \langle e^1 \rangle = \nu, \quad \text{and} \quad \langle e^2 \rangle = \sigma^2 + \nu^2 \tag{4.36}$$

for  $s = 0$ ,  $s = 1$ , and  $s = 2$ , just the constraints used to assign the probability density function. Fourth, for a fixed value of the mean and variance this prior probability has highest entropy. Consequently, when parameters are marginalized from probability distributions or when any operation

is performed on them that preserves mean and variance while discarding other information, those probability densities necessarily will move closer and closer to this Gaussian distribution regardless of the initial probability assignment. The Central Limit Theorem is one special case of this phenomenon—see Jaynes [32].

Earlier it was asserted that maximum entropy distributions are the only distributions that have sufficient statistics and that these sufficient statistics are the only properties of the data, and therefore the errors, that are used in estimating parameters. We would like to demonstrate this property explicitly for the Gaussian distribution [32, 12]. Suppose the value of a location parameter is  $\nu_0$  and one has a measurement such that

$$d_j = \nu_0 + n_j. \quad (4.37)$$

The hypothesis about which inferences are to be made is of the form “the true value of the mean is  $\nu$  given the data,  $D$ .” Assigning a Gaussian as the prior probability for the errors, the likelihood function is then given by

$$P(D|\nu\sigma I) = (2\pi\sigma^2)^{-\frac{N}{2}} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{j=1}^N (d_j - \nu)^2 \right\}. \quad (4.38)$$

The posterior probability for  $\nu$  may be written as

$$P(\nu|D\sigma I) \propto (2\pi\sigma^2)^{-\frac{N}{2}} \exp \left\{ -\frac{N}{2\sigma^2} ([\bar{d} - \nu]^2 + s^2) \right\} \quad (4.39)$$

where a uniform prior probability was assigned for  $\nu$ . The mean data value,  $\bar{d}$ , is given by

$$\bar{d} = \frac{1}{N} \sum_{j=1}^N d_j = \nu_0 + \bar{n} \quad (4.40)$$

where  $\bar{n}$  is the mean value of the errors. And  $s^2$  is given by

$$s^2 = \overline{d^2} - (\bar{d})^2 = \frac{1}{N} \sum_{j=1}^N d_j^2 - \left( \frac{1}{N} \sum_{j=1}^N d_j \right)^2 = \overline{n^2} - (\bar{n})^2 \quad (4.41)$$

where  $(\bar{n})^2$  is the mean square of the noise values. From which one obtains

$$(\nu)_{est} = \begin{cases} \bar{d} \pm \sigma/\sqrt{N} & \sigma \text{ known} \\ \bar{d} \pm s/\sqrt{N-3} & \sigma \text{ unknown} \end{cases} \quad (4.42)$$

as the estimate for  $\nu$ . The actual error,  $\Delta$ , is given by

$$\Delta = \bar{d} - \nu_0 = \bar{n} \quad (4.43)$$

which depends only on the *mean of the noise values*; while our accuracy estimate depends only on  $\sigma$  if the standard deviation of the noise is known, and *only on the mean and mean-square* of the noise values when the standard deviation of the noise is not known. Thus the underlying

sampling distribution of the noise has completely canceled out and the only property of the errors that survives is the actual mean and mean-square of the noise values. *All* other properties of the errors have been made irrelevant. Exactly the same parameter estimates will result if the underlying sampling distribution of the noise is changed, provided the mean and mean-square of the new sampling distribution is the same, just the properties needed to represent what is actually known about the noise, and to render what is *not* known about it irrelevant. For more on the subject of how maximum entropy probability assignments render the underlying sampling distribution nearly irrelevant see [32] and [12].

In Section 4.1 the sum and product rules of probability theory were given. In Section 4.2 the principle of maximum entropy was used to demonstrate how to assign probabilities that are maximally uninformative while remaining consistent with the given prior information. In the following section a nontrivial parameter estimation problem is given. Each step in the calculation is explained in detail. The example is complex enough to illustrate all of the points of principle that must be faced in more complicated problems, yet sufficiently simple that anyone with a background in calculus should be able to follow the mathematics. In addition, the problem, single frequency estimation, is of interest to the general NMR community and in the process of solving the problem we will uncover a number of new and interesting features about the discrete Fourier transform and the bandwidth of nonuniformly sampled data.

### 4.3 Example: Parameter Estimation

Probability theory tells one what to believe about a hypothesis  $C$  given all of the available information or evidence  $E_1 \cdots E_n$ . This is done by computing the posterior probability for hypothesis  $C$  conditional on all of the evidence  $E_1 \cdots E_n$ . This posterior probability is represented symbolically by

$$P(C|E_1 \cdots E_n). \quad (43)$$

It is computed from the rules of probability theory by repeated application of the sum and product rules and by assigning the probabilities so indicated. This is a general rule and there are no exceptions to it: *ad hoc devices have no place in probability theory*. Given the statement of a problem, the rules of probability theory take over and will lead every person to the same unique solution, provided each person has exactly the same evidence.

To someone unfamiliar with probability theory, how this is done is not obvious; nor is it obvious what must be done to obtain a problem that is sufficiently well defined to permit the application of probability theory as logic. Consequently, in what follows all of the steps in computing  $P(C|E_1 \cdots E_n)$  will be described in detail. To compute the probability for any hypothesis  $C$  given some evidence  $E_1 \cdots E_n$ , there are five basic steps, which are not necessarily independent:

1. *Define The Problem:* State in nonambiguous terms exactly what hypothesis you wish to make inferences about.
2. *State The Model:* Relate the hypothesis of interest to the available evidence  $E_1 \cdots E_n$ .
3. *Apply Probability Theory:* The probability for hypothesis  $C$  conditional on all the available evidence  $E_1 \cdots E_n$  is computed from Bayes theorem. The sum rule is then applied to eliminate nuisance hypotheses. The product rule is then repeatedly applied to factor joint probabilities to obtain terms which cannot be further simplified.

4. *Assign The Probabilities:* Using the appropriate procedures, translate the available evidence into numerical values for the indicated probabilities.
5. *Evaluate The Integrals and Sums:* Evaluate the integrals and sums indicated by probability theory. If the indicated calculations cannot be done analytically, then implement the necessary computer codes to evaluate them numerically.

Each of these steps will be systematically illustrated in the following example, the single frequency estimation problem when the data are nonsimultaneously nonsimultaneously sampled.

### 4.3.1 Define The Problem

The problem to be solved is to estimate the frequency of a decaying sinusoid given some data and whatever prior information we have. The Bayesian calculations presented in this chapter will be for quadrature NMR data that has been sampled at differing times and with differing numbers of data values in each channel. We will derive the solution to the frequency estimation problem given an exponentially decaying sinusoidal model. Then, through a series of simplifications, we will reduce this calculation to the case of frequency estimation for a stationary sinusoid given real (nonquadrature) data. In the process of making these simplifications, we will encounter the Lomb-Scargle periodogram [39, 54, 55, 56], the Schuster periodogram [57] and a weighted power spectrum as sufficient statistics for frequency estimation. Because each sufficient statistic will have been derived from the rules of probability theory we will see the exact conditions under which each is an optimal frequency estimator. Thus, by making these simplifications, we will see how probability theory generalizes the discrete Fourier transform to handle nonuniformly nonsimultaneously sampled data and what is happening to the aliasing phenomenon in these data. Before doing this, we need to understand the discrete Fourier transform and the phenomena of aliasing.

#### 4.3.1.1 The Discrete Fourier Transform

When the data consist of uniformly sampled time domain data containing some type of harmonic oscillations, the discrete Fourier transform is almost universally used as the frequency estimation technique. This is done for a number of reasons, but primarily because the technique is fast and experience has shown that the frequency estimates obtained from it are often very good. The discrete Fourier transform,  $\mathcal{F}(f_k)$ , is defined as:

$$\mathcal{F}(f_k) = \sum_{j=0}^{N-1} \mathbf{d}(t_j) \exp\{2\pi f_k t_j \mathbf{i}\} \quad (4.44)$$

where  $\mathbf{i} = \sqrt{-1}$ ,  $\mathbf{d}(t_j)$  is the complex discretely sampled data,

$$\mathbf{d}(t_j) = d_R(t_j) + \mathbf{i}d_I(t_j), \quad (4.45)$$

and is composed of real,  $d_R(t_j)$ , and imaginary,  $d_I(t_j)$ , data samples;  $N$  is the total number of complex data samples and  $f_k$  is the frequency. For uniformly sampled data, the times are given by

$$t_j = j\Delta T, \quad j \in \{0, 1 \dots N - 1\}, \quad (4.46)$$

where  $\Delta T$  is the dwell time, the time interval between data samples, and the frequencies are given by

$$f_k = \frac{k}{N\Delta T} \quad k \in \left\{ -\frac{N}{2}, -\frac{N}{2} + 1, \dots, \frac{N}{2} \right\}. \quad (4.47)$$

These frequencies are the ones at which a discrete Fourier transform is exactly equal to the continuous Fourier transform of a bandlimited function [49, 48]. The largest frequency interval free of aliases for a bandlimited function is given by

$$-f_{Nc} \leq f \leq f_{Nc} \quad (4.48)$$

and is called the bandwidth. The frequency  $f_{Nc}$  is called the Nyquist critical frequency and is given by

$$f_{Nc} = \frac{1}{2\Delta T}. \quad (4.49)$$

Nothing would prohibit one from taking  $f_k$  as a continuous variable and evaluating Eq. (4.44) at different frequencies. Indeed, this is exactly what the common practice of zero-padding<sup>1</sup> does. After all, adding zero to a sum does not change that sum, so for any given frequency zero padding has no effect in Eq. (4.44). The only effect is in Eq. (4.47): changing  $N$  changes the frequencies at which Eq. (4.44) is evaluated.

If we expand the right-hand side of Eq. (4.44), we obtain

$$\mathcal{F}(f_k) = R(f_k) + \mathbf{i}I(f_k) \quad (4.50)$$

where

$$R(f_k) = \sum_{j=0}^{N-1} [d_R(t_j) \cos(2\pi f_k t_j) - d_I \sin(2\pi f_k t_j)] \quad (4.51)$$

and

$$I(f_k) = \sum_{j=0}^{N-1} [d_R(t_j) \sin(2\pi f_k t_j) + d_I \cos(2\pi f_k t_j)] \quad (4.52)$$

are the real and imaginary parts of the discrete Fourier transform.

Three different ways of viewing the results of the discrete Fourier transform are common: the absorption spectrum, the power spectrum and the absolute-value spectrum. The absorption spectrum, the real part of an appropriately phased discrete Fourier transform, is commonly used in NMR. In NMR the sinusoids usually have the same phase; consequently if one multiplies Eq. (4.50) by  $\exp\{\mathbf{i}(\theta + T_0 f_k)\}$ , the phase of the sinusoids can be made to cancel from the discrete Fourier transform. The two parameters,  $\theta$  and  $T_0$ , are the zero- and first-order phase corrections, and must be estimated from the discrete Fourier transform. An absorption spectrum's usefulness is limited to problems in which the sinusoids have the same phase parameters. This is common in NMR, but not with other physical phenomena, consequently, we will not discuss the absorption spectrum further.

The power spectrum is defined as

$$\text{Power}(f_k) = \frac{R(f_k)^2 + I(f_k)^2}{N} \quad (4.53)$$

---

<sup>1</sup> To zero pad a data set, one adds zeros to the end of a data set, sets  $N$  to the length of this new zero padded data set and then runs a fast discrete Fourier transform on the zero padded data.

Figure 4.1: Frequency Estimation Using The DFT

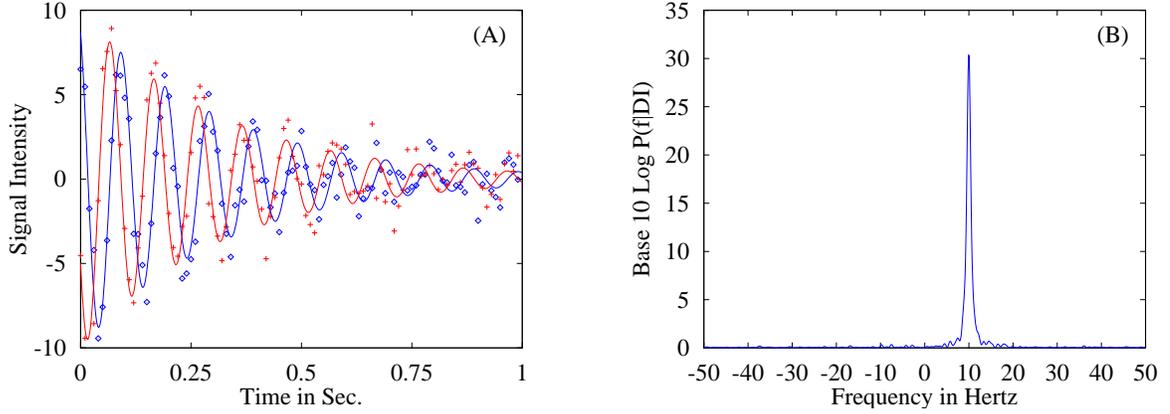


Figure 4.1: Panel (A) is computer simulated data. It contains a single exponentially decaying sinusoidal signal plus noise. The lines represent the real and imaginary parts of the sinusoid. The locations of the data values are denoted by the isolated characters. The Nyquist critical frequency for this data set is  $f_{Nc} = 50$  Hz. Panel (B) is a plot of the base 10 logarithm of the posterior probability for the frequency of a stationary sinusoid given these data.

and is the square of the absolute-value spectrum. It has been shown by Bretthorst [4, 5, 6] and Woodward [65], and as we will demonstrate shortly, the power spectrum is the sufficient statistic in a Bayesian calculation for the posterior probability for the frequency given a single stationary sinusoidal model with simultaneously sampled quadrature data.

In this chapter we will make several plots of the discrete Fourier transform and its generalizations to nonuniformly nonsimultaneously sampled data. To allow direct comparison of these plots we will always plot the same function of the data, the base 10 logarithm of the posterior probability for the frequency of a stationary sinusoid independent of the phase, amplitude and variance of the noise, Eq. (4.101) below. For uniformly sampled quadrature data, this probability is a simple function of the power spectrum, see Bretthorst [2] for a more extended discussion of the relationship between the discrete Fourier transform and the posterior probability for a stationary frequency.

To illustrate the use of the discrete Fourier transform as a frequency estimation tool, suppose we had the data shown in Fig. 4.1(A). The signal in this simulated data is an exponentially decaying sinusoid of amplitude 10 plus Gaussian noise of zero mean and standard deviation one. These data were generated with a dwell time of  $\Delta T = 0.01$  Sec. One hundred complex data values were generated at times ranging from 0 to 0.99 seconds. The frequency is 10 Hz, and the decay rate constant is  $3 \text{ Sec.}^{-1}$ . The real and imaginary data values are represented by the isolated characters in Fig. 4.1(A). The lines represent the real and imaginary parts of the true sinusoid. The Nyquist critical frequency for these data is one-half the inverse of the dwell time:

$$f_{Nc} = \frac{1}{2\Delta T} = \frac{1}{2(0.01 \text{ Sec.})} = 50 \text{ Hz.} \quad (4.54)$$

Figure 4.1(B) is a plot of the base 10 logarithm of the posterior probability over the bandwidth ( $-50 \text{ Hz} \leq f \leq 50 \text{ Hz}$ ). This base 10 logarithm starts at essentially zero and then increases some 30

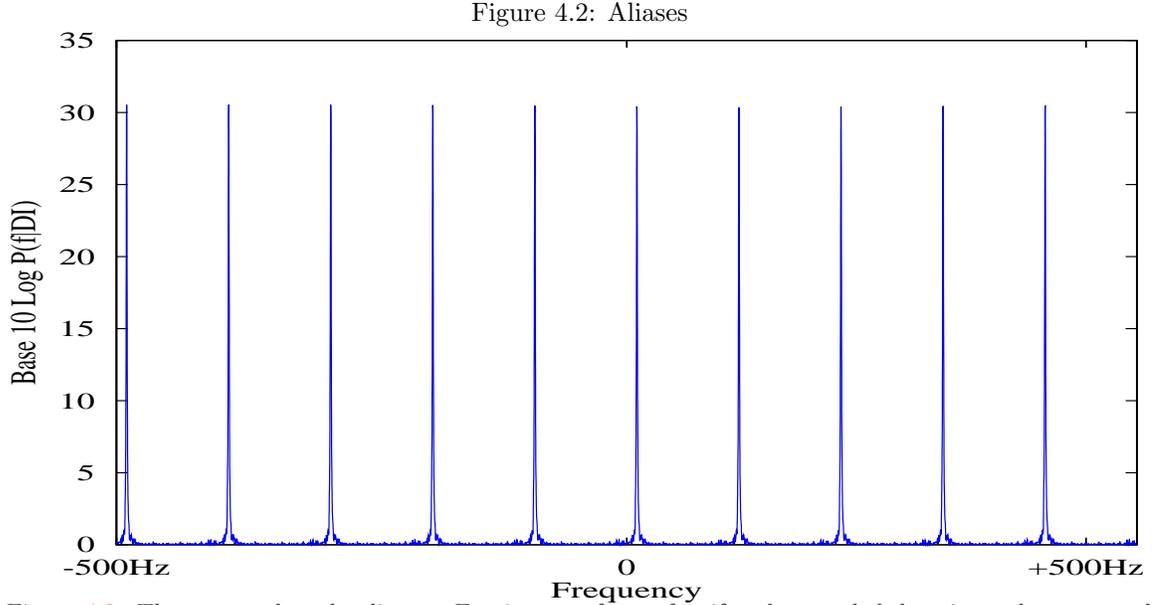


Figure 4.2: The reason that the discrete Fourier transform of uniformly sampled data is rarely computed at frequencies greater than the Nyquist critical frequency is simply that the discrete Fourier transform is a periodic function with a period equal to the bandwidth  $2f_{Nc}$ .

orders of magnitude, coming to a very sharp peak at 10 Hz.

#### 4.3.1.2 Aliases

We would like to investigate the aliasing phenomenon. To do this we must evaluate the discrete Fourier transform outside the bandwidth and this cannot be done using the fast discrete Fourier transform. Because we are plotting the base 10 logarithm of the posterior probability for the frequency of a stationary sinusoid, we can simply evaluate the posterior probability for any frequency and the sufficient statistic will be the power spectrum evaluated at that frequency; we are not restricted to the frequencies  $f_k$  specified by Eq. (4.47). The resulting plot is shown in Fig. 4.2. Outside of the interval  $(-f_{Nc} \leq f \leq f_{Nc})$ , the base 10 logarithm of the posterior probability, and thus the power spectrum and the discrete Fourier transform, are periodic functions of  $f$  with a period equal to the frequency interval spanned by the bandwidth. In Fig. 4.2, the frequency interval plotted is  $(-10f_{Nc} \leq f \leq 10f_{Nc})$ , so there should be 10 peaks in this range as Fig. 4.2 shows.

To understand why the discrete Fourier transform is a periodic function of frequency, suppose we wish to evaluate the discrete Fourier transform at the frequencies

$$f_k = \frac{k}{N\Delta T}, \quad k = mN + k', \quad k' \in \left\{ -\frac{N}{2}, -\frac{N}{2} + 1, \dots, \frac{N}{2} \right\}. \quad (4.55)$$

By itself the index  $k'$  would specify the normal frequency interval, Eq. (4.47), of a discrete Fourier transform. However, the integer  $m$  shifts this frequency interval up or down by an integer multiple

of the total bandwidth. If  $m = 0$ , we are in the interval  $(-f_{Nc} \leq f_k \leq f_{Nc})$ ; if  $m = 1$ , we are in the interval  $(f_{Nc} \leq f_k \leq 3f_{Nc})$ , etc. If we now substitute Eqs. (4.55) and (4.46) into the discrete Fourier transform, Eq. (4.44), the reason the discrete Fourier transform is periodic becomes readily apparent

$$\mathcal{F}(f_{k'}) \equiv \sum_{j=0}^{N-1} \mathbf{d}(t_j) \exp \left\{ \frac{2\pi \mathbf{i}(mN + k')j}{N} \right\}, \quad (4.56)$$

$$= \sum_{j=0}^{N-1} \mathbf{d}(t_j) \exp \{2\pi \mathbf{i}mj\} \exp \left\{ \frac{2\pi \mathbf{i}k'j}{N} \right\}, \quad (4.57)$$

$$= \sum_{j=0}^{N-1} \mathbf{d}(t_j) \exp \left\{ \frac{2\pi \mathbf{i}k'j}{N} \right\}, \quad (4.58)$$

$$= \sum_{j=0}^{N-1} \mathbf{d}(t_j) \exp \{2\pi \mathbf{i}f_{k'}t_j\}. \quad (4.59)$$

In going from Eq. (4.57) to (4.58) a factor,  $\exp\{\mathbf{i}(2\pi mj)\}$ , was dropped because both  $m$  and  $j$  are integers, so  $(2\pi mj)$  is an integer multiple of  $2\pi$ , and the complex exponential is one. Aliases occur because the complex exponential canceled leaving behind a discrete Fourier transform on the interval  $(-f_{Nc} \leq f_k \leq f_{Nc})$ . The integer  $m$  specifies which integer multiple of the bandwidth is being evaluated and will always be an integer no matter how the data are collected. However, the integer  $j$  came about because the data were uniformly sampled. If the data had not been uniformly sampled the relationship,  $t_j = j\Delta T$ , would not hold, the complex exponential would not have canceled, and aliases would not have been present.

Because frequency estimation using the discrete Fourier transform was not derived using the rules of probability theory, there is no way to be certain that the estimates we obtain are the best we could do. What is needed is the solution to the single frequency estimation problem using Bayesian probability theory. Consequently, in the next section we analyze this problem, then in the following sections we will derive the conditions under which the discrete Fourier transform power spectrum is a sufficient statistic for single frequency estimation and we will see how probability theory generalizes the discrete Fourier transform to nonuniformly nonsimultaneously sampled data, and we will see the effect of these generalizations on the aliasing phenomenon.

### 4.3.2 State The Model—Single-Frequency Estimation

The problem is the estimation of the frequency of a single exponentially decaying sinusoid independent of the amplitude and phase of the sinusoid, given nonuniformly nonsimultaneously sampled quadrature data. First, what do we mean by nonuniformly nonsimultaneously sampled quadrature data? “Quadrature” simply means we have a measurement of the real and imaginary parts of a complex signal. So nonuniformly nonsimultaneously sampled quadrature data are measurements of the real and imaginary parts of a complex signal for which the measurements of the real and imaginary parts of the signal occur at different times. But if we have differing numbers of data samples with differing sample times, we really have two data sets: a real and an imaginary data set.<sup>2</sup> The

<sup>2</sup>Of course, when we say an “imaginary data set” we mean only that the data are a measurement of the imaginary part of the signal; not that the data are imaginary numbers.

real data set will be designated as  $D_R \equiv \{d_R(t_1) \cdots d_R(t_{N_R})\}$ , where  $d_R$  means a real data sample,  $t_i$  is the time the data sample was acquired, and  $N_R$  is the total number of data samples in the real data set. Similarly, the imaginary data set will be denoted by  $D_I \equiv \{d_I(t_1) \cdots d_I(t_{N_I})\}$ . We impose no restrictions on the number of data samples or the acquisition times in either channel. They could be the same or different, depending on the limit we investigate. Note that the times do not carry a channel indication so the context within the equations will have to establish which times we are referring to. If the equation context fails, we will clarify it in the text.

To perform any calculation using probability theory, the hypothesis of interest must be related to the information we actually possess. For the problem of estimating the frequency of a complex sinusoid, this means relating the frequency to the quadrature data through a model. If the complex data are given by Eq. (4.45), then the data and the sinusoid are related by

$$\mathbf{d}(t_j) = \mathbf{A} \exp\{-\mathbf{f}t_j\} + \mathbf{n}(t_j). \quad (4.60)$$

The complex amplitude is given by  $\mathbf{A} = A_1 - \mathbf{i}A_2$ , and is equivalent to the amplitude and phase of the sinusoid. The complex frequency,  $\mathbf{f} = \alpha + 2\pi\mathbf{i}f$ , contains two parameters: the decay rate constant,  $\alpha$ , and the frequency,  $f$ . Note the minus signs in the definition of the complex amplitude and the one in Eq. (4.60). These signs correspond to a convention establishing what is meant by a positive frequency. The signs were chosen to model the data produced by a Varian NMR spectrometer. Other vendors use different conventions. Changing these conventions will change none of the conclusions that follow and very few of the actual details of the calculations. The decay rate constant,  $\alpha$ , has units of inverse seconds, the frequency,  $f$ , Hertz, and the times,  $t_j$ , seconds. The quantity  $\mathbf{n}(t_j)$  represents the complex noise at time  $t_j$ . Note that in this equation the times,  $t_j$ , simply designate the times at which we actually have data. If the datum happen to be a measurement of the real part of the signal, then the time would be associated with the real data set, and similarly for the imaginary part of the signal.

If we separate Eq. (4.60) into its real and imaginary parts, we have for the real part

$$d_R(t_i) = M_R(t_i) + n_R(t_i) \quad (4.61)$$

$$M_R(t_i) \equiv [A_1 \cos(2\pi ft_i) - A_2 \sin(2\pi ft_i)] \exp\{-\alpha t_i\} \quad (4.62)$$

and for the imaginary part we have

$$d_I(t_j) = M_I(t_j) + n_I(t_j) \quad (4.63)$$

$$M_I(t_j) \equiv -[A_1 \sin(2\pi ft_j) + A_2 \cos(2\pi ft_j)] \exp\{-\alpha t_j\}, \quad (4.64)$$

where  $n_R(t_i)$  and  $n_I(t_j)$  represent the noise in the real and imaginary data at times  $t_i$  and  $t_j$ . The quantity that we would like to estimate is the frequency,  $f$ , and we would like to estimate it independent of the amplitudes and variance of the noise. The decay rate constant,  $\alpha$ , will sometimes be treated as a nuisance parameter, sometimes estimated, and sometimes taken as a given, depending of our purpose at the time. For now we will estimate it.

### 4.3.3 Apply Probability Theory

In probability theory as logic, all of the information about a hypothesis is summarized in a probability density function. For this problem, estimating the frequency of an exponentially decaying sinusoid,

the probability density function is designated as  $P(f|D_R D_I I)$ , which should be read as the posterior probability for the frequency given the real and imaginary data sets and the information  $I$ . The information  $I$  is all of the other information we have about the parameters appearing in the problem. Note that  $f$  appearing in this equation is not a parameter in any real sense; rather it is a hypotheses of the form “at the time the data were taken the frequency of the sinusoid was  $f$ .” So by computing the posterior probability for various values of  $f$  we are ranking an entire series of hypotheses about the values of the frequency.

There are many different ways to compute the posterior probability for the frequency,  $P(f|D_R D_I I)$ , but they all lead to the same final result. Here we will simply apply Bayes’ theorem:

$$P(f|D_R D_I I) = \frac{P(f|I)P(D_R D_I |fI)}{P(D_R D_I |I)} \quad (4.65)$$

The three terms on the right-hand side of this equation are the prior probability for the frequency,  $P(f|I)$  and it represents what was known about the frequency before seeing the data. The second term,  $P(D_R D_I |fI)$ , is the direct probability for the data given the frequency, and it represents what was learned about the frequency from the data. This term is often called the likelihood. Finally, the denominator,  $P(D_R D_I |I)$ , is a normalization constant and is computed using the product and sum rules of probability theory as

$$P(D_R D_I |I) = \int df P(D_R D_I |fI) = \int df P(f|I)P(D_R D_I |fI). \quad (4.66)$$

This term has many names, it is a direct prior probability, direct because it is a probability for the data; and prior because it depends only on our prior information. You will often hear the logarithm of this term called the evidence, and this probability is often called a Bayes factor. None of these terms really convey the importance of this term in model selection problems. However, our goal is parameter estimation, not model selection, so this direct probability is just a constant that we don’t care about. If we agree to normalize the posterior probability for the frequency at the end of the calculation, then we can drop this normalization constant and one obtains:

$$P(f|D_R D_I I) \propto P(f|I)P(D_R D_I |fI). \quad (4.67)$$

The prior probability for the frequency is simplified enough that it could be assigned, however the direct probability for the data given the frequency,  $P(D_R D_I |fI)$ , is not nearly simplified enough to assign.

To see how to continue factoring this probability, note that  $P(D_R D_I |fI)$  does not depend on the amplitude, phase or decay rate constant of the sinusoid, nor does it depend on the any properties of the errors. So this probability must be a marginal probability where the effects of the hypotheses indexed by these other parameters have been removed using the rules of probability theory. This marginal probability is computed from the joint probability for the data and these hypotheses given the frequency and prior information  $I$ , one can write

$$P(f|D_R D_I I) \propto P(f|I) \int dA_1 dA_2 d\alpha d\sigma P(A_1 A_2 \alpha \sigma D_R D_I |fI) \quad (4.68)$$

where  $\sigma$  is the standard deviation of the noise prior probability. Applying the product rule one obtains

$$P(f|D_R D_I I) \propto P(f|I) \int dA_1 dA_2 d\alpha d\sigma P(A_1 A_2 \alpha \sigma |I) P(D_R D_I |f A_1 A_2 \alpha \sigma I) \quad (4.69)$$

where the assumption was made that if the frequency is given to you, then it doesn't change the prior probabilities you would assign to the other parameters.

To compute the posterior probability for the frequency, we must perform a multidimensional integral over all of the parameters we are not interested in. As it will turn out, all of the integrals except the one over the decay rate may be done in close form. Consequently in much of what follows, the decay rate will be assumed known, in the sense of given and we will ignore it. The exception to this is the section on parameter estimation, and there we will marginalize over the decay rate numerically. However, before we can do this we must continue simplifying these probabilities, and before we can do the integrals we must assign numerical values to them.

First, we will simplify the joint posterior probability for the parameters. This probability,  $P(\alpha A_1 A_2 \sigma | I)$ , has four hypotheses as its arguments and one given. If we designate any one of these hypotheses as  $a$ , and all of the others as  $b$ , then this prior probability may be factored as  $p(ab|I) = P(a|I)P(b|aI)$ , where  $p(a|I)$  is the prior probability for hypotheses  $a$  given  $I$ , and  $P(b|aI)$  is the joint prior probability for all of the other hypotheses given knowledge of both  $a$  and  $I$ . Now suppose  $P(b|aI) = P(b|I)$ , that is to say, knowledge of  $a$  does nothing to change our prior probability assignment for  $b$ . This condition, known as logical independence, allows the prior probability to be factored into a set of independent prior for each parameter

$$P(\alpha A_1 A_2 \sigma | I) = P(\alpha | I)P(A_1 | I)P(A_2 | I)P(\sigma | I). \quad (4.70)$$

There are no additional simplifications that can be made to reduce these prior probabilities, and we will soon be forced to assign them numerical values. However, before doing that we must simplify the direct probability or likelihood.

We expect the two data sets to have the same noise standard deviation, because in NMR they are acquired by projecting the same signal onto orthogonal functions, a sine and a cosine. The fact that the two data sets are orthogonal projections, also means that each of the two data sets should contain unique and different information, i.e., they should be logically independent of each other. So the joint direct probability,  $P(D_R D_I | f \alpha A_1 A_2 \sigma I)$ , will factor as

$$P(D_R D_I | f \alpha A_1 A_2 \sigma I) = P(D_R | f \alpha A_1 A_2 \sigma I)P(D_I | f \alpha A_1 A_2 \sigma I) \quad (4.71)$$

a product of likelihoods, one for the real data and one for the imaginary data. Indeed, logical independence of these two data sets is almost forced upon us because we don't even know if the imaginary data exists, the number of imaginary data values could be zero.

If we now collect the prior probabilities from Eq. (4.70) and the likelihoods from Eq. (4.71) the posterior probability for the frequency becomes

$$\begin{aligned} P(f | D_R D_I I) &= \int dA_1 dA_2 d\alpha d\sigma \\ &\times P(f | I)P(\alpha | I)P(A_1 | I)P(A_2 | I)P(\sigma | I) \\ &\times P(D_R | f \alpha A_1 A_2 \sigma I)P(D_I | f \alpha A_1 A_2 \sigma I). \end{aligned} \quad (4.72)$$

None of the prior probabilities may be further simplified, however, the likelihoods could be further simplified, but if we are not taking correlations into account, need not be.

### 4.3.4 Assign The Probabilities

We have now reached the point where we need to assign numerical values to each of the probabilities appearing in Eq. (4.72). In all cases we are going to assign numerical values that represent what we actually know about the parameters. In most of these cases that may not be very much. For example, the prior probabilities for the amplitudes  $P(A_1|I)$  or  $P(A_2|I)$ . We know they can be either positive or negative, and we know that their value is an upper,  $A_{max}$ , and lower bound,  $A_{min}$ . If that is all we know about these parameters then the principle of maximum entropy will lead us to assign a uniform prior probability:

$$P(A_x|I) = \begin{cases} \frac{1}{A_{max} - A_{min}} & A_{min} \leq A_x \leq A_{max} \\ 0 & \text{otherwise} \end{cases} \quad (4.73)$$

where  $A_x$  represents either  $A_1$  or  $A_2$ .

In a similar vein, we will assign a uniform prior probability for the frequency. For uniformly sampled data, the range of frequencies that are not aliases is simply related to the sampling rate. However, for nonuniformly sampled data, exactly what this range is, is an all together more interesting question, a question we will answer shortly, but not until we see how probability theory generalizes the discrete Fourier transform. Nonetheless we will assume that one can specify a range of values,  $f_{min}$  to  $f_{max}$  and that our prior expectation of the frequency is uniform over this region.

The standard deviation,  $\sigma$ , is a scale parameter. The completely uninformative prior probability for a scale parameter is the Jeffreys' prior [33],

$$P(\sigma|I) \propto \frac{1}{\sigma} \quad 0 \leq \sigma < \infty. \quad (4.74)$$

However, this is not strictly speaking a probability at all, because it cannot be normalized. To make this a proper, i.e., normalized, probability one must introduce an upper,  $\sigma_{max}$ , and lower bound,  $\sigma_{min}$ , and compute the normalization constant:

$$P(\sigma|I) = \begin{cases} \frac{1}{\log(\sigma_{max}/\sigma_{min})\sigma} & \sigma_{min} \leq \sigma \leq \sigma_{max} \\ 0 & \text{otherwise} \end{cases} \quad (4.75)$$

and then perform the integral over the valid range. If a Jeffreys' prior was desired, then at the end of the calculation one can pass to the limits  $\sigma_{min} \rightarrow 0$ , and  $\sigma_{max} \rightarrow \infty$ . It is only by following this safe, cautious procedures, that one can be sure of not inadvertently introducing singular mathematics into the calculations. However, in the process of doing the integrals over  $A_1$ ,  $A_2$  and  $\sigma$  we will assume that upper and lower bounds are very wide, much much wider than the maximum likelihood estimates of these parameters, and pass to the limit. We do this because it simplifies mathematics without introducing any complicating factors into this calculation.

The only remaining prior probability to be assigned is for the decay rate constant,  $P(\alpha|I)$ . Now the decay rate constant is a scale parameter and a bounded Jeffreys' prior would be appropriate for most conditions. However, one of the intentions of this chapter is to illustrate how probability theory generalizes the discrete Fourier transform to the case of nonuniformly nonsimultaneously sampled

data and for that demonstration a bounded Jeffreys' prior is inconvenient. Consequently, we will use one of two priors' for the decay rate constant, either a bounded Jeffreys' prior:

$$P(\alpha|I) = \begin{cases} \frac{1}{\log(\alpha_{max}/\alpha_{min})\alpha} & \alpha_{min} \leq \alpha \leq \alpha_{max} \\ 0 & \text{otherwise} \end{cases} \quad (4.76)$$

when we are interested in estimating the decay rate constant, or we will assume that  $\alpha$  is a given value  $\hat{\alpha}$  and assign

$$P(\alpha|I) = \delta(\hat{\alpha} - \alpha) \quad (4.77)$$

when we are interested in illustrating the relationships between the discrete Fourier transform.

The only other prior probabilities to be assigned are the two direct probabilities  $P(D_R|f\alpha A_1 A_2 \sigma I)$  and  $P(D_I|f\alpha A_1 A_2 \sigma I)$ . We will concentrate on assigning the direct probability for the real data,  $P(D_R|f\alpha A_1 A_2 \sigma I)$ , and after assigning it, how to assign  $P(D_I|f\alpha A_1 A_2 \sigma I)$  will be obvious.

Probabilities are designated by the notation  $P(X|I)$ . The hypotheses appearing on the left-hand side of the vertical bar “|” are the hypotheses about which we are making inferences; while the hypotheses appearing on the right are given as true, *i.e.*, they specify the facts on which this probability is based. With this in mind, look at  $P(D_R|f\alpha A_1 A_2 \sigma I)$ . This term is the direct probability for the real data given the truth of all of the parameters in the model. But if the parameters are given, then from Eq. (4.61)

$$d_R(t_i) - M_R(t_i) = n_R(t_i). \quad (4.78)$$

The left-hand side of this equation contains the given parameter values, consequently the right-hand side contains the given error values. These given error values are thus hypotheses about which we must make inferences. These hypotheses are of the form “the true noise value was  $n_R(t_i)$  at time  $t_i$ ,” where  $n_R(t_i)$  is a running index and its numerical value would range over all valid noise amplitudes. Equation (4.78) is used in probability theory by introducing the joint probability for the data and the errors, and using the product and sum rules of probability theory to remove the dependence on the unknown error values:

$$\begin{aligned} P(D_R|f\alpha A_1 A_2 \sigma I) &= \int dn_R(t_1) \cdots dn_R(t_{N_R}) P(D_R\{n_R\}|f\alpha A_1 A_2 \sigma I) \\ &\propto \int dn_R(t_1) \cdots dn_R(t_{N_R}) P(\{n_R\}|\sigma) P(D_R|\{n_R\}f A_1 A_2 \alpha I) \end{aligned} \quad (4.79)$$

where  $\{n_R\} \equiv \{n_R(t_1), n_R(t_2), \dots, n_R(t_{N_R})\}$ ,  $P(D_R|\{n_R\}f A_1 A_2 \alpha I)$ , is the direct probability for the data given the errors and the parameters. The final term,  $P(\{n_R\}|\sigma)$ , is the prior probability for the errors given  $\sigma$ .

Most of the section on maximum entropy was devoted to deriving the Gaussian distribution as the prior probability appropriate to represent what is actually known about the errors. Using what was derived in Section 4.2 and assigning a Gaussian prior probability distribution for the errors one has

$$P(n_R(t_1) \cdots n_R(t_N)|\sigma) = (2\pi\sigma^2)^{-\frac{N_R}{2}} \exp\left\{-\frac{1}{2\sigma^2} \sum_{i=1}^{N_R} n_R(t_i)^2\right\} \quad (4.80)$$

That leaves one final term that must be assigned,  $P(D_R|\{n_R\}f A_1 A_2 \alpha I)$ , the probability for the data given the errors and the parameters. But if we know the parameter values and we know the

error values, the by Eq. (4.78),  $P(D_R|\{n_R\}fA_1A_2\alpha I)$ , must be a delta function

$$P(D_R|\{n_R\}fA_1A_2\alpha I) = \delta [d_R(t_i) - M_R(t_i)]. \quad (4.81)$$

If we substitute this delta function, Eq. (4.81) and the prior probability for the errors, Eq.(4.80), into Eq. (4.79) one obtains

$$\begin{aligned} P(D_R|f\alpha A_1A_2\sigma I) &= \int dn_R(t_1) \cdots dn_R(t_{N_R}) \\ &\times (2\pi\sigma^2)^{-\frac{N_R}{2}} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{N_R} n_R(t_i)^2 \right\} \\ &\times \delta [d_R(t_i) - M_R(t_i)]. \end{aligned} \quad (4.82)$$

Finally, performing the integral over all of the  $n_R$  one obtains

$$P(D_R|f\alpha A_1A_2\sigma I) = (2\pi\sigma^2)^{-\frac{N_R}{2}} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{N_R} [d_R(t_i) - M_R(t_i)]^2 \right\} \quad (4.83)$$

as the direct probability for the data. In most expositions using probability one simply postulates a Gaussian probability density function and dispenses with all of the intermediate steps that tie everything together. Indeed in the original version of this paper, [13], that is exactly what the author did. However, here it was thought necessary to show how the rules of probability theory interact with maximum entropy to give both uninformative and informative prior probabilities.

Now having seen this derivation, we will simply skip all of the intermediate steps and make a Gaussian assignment for  $P(D_I|f\alpha A_1A_2\sigma I)$ . If we now substitute all of the probabilities into the posterior probability for the frequency one obtains

$$\begin{aligned} P(f|D_R D_I I) &\propto \int dA_1 dA_2 d\alpha \frac{d\sigma}{\sigma} P(\alpha|I) \\ &\times \sigma^{-N_R} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=0}^{N_R-1} [d_R(t_i) - M_R(t_i)]^2 \right\} \\ &\times \sigma^{-N_I} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{j=0}^{N_I-1} [d_I(t_j) - M_I(t_j)]^2 \right\} \end{aligned} \quad (4.84)$$

where we have dropped some constants, as they will disappear when this probability is normalized and we have left the prior probability for the decay rate constant unspecified.

### 4.3.5 Evaluate The Sums and Integrals

Substituting Eqs. (4.62) and (4.64) for  $M_R(t_i)$  and  $M_I(t_j)$  into the joint posterior probability for the frequency and decay rate constant, Eq. (4.71), one obtains

$$P(f|D_R D_I I) \propto \int dA_1 dA_2 d\alpha \frac{d\sigma}{\sigma} P(\alpha|I) \sigma^{-(N_R+N_I)} \exp \left\{ -\frac{Q}{2\sigma^2} \right\} \quad (4.85)$$

where

$$Q \equiv (N_R + N_I)\bar{d}^2 - 2\sum_{l=1}^2 A_l T_l + \sum_{k,l=1}^2 g_{kl} A_k A_l. \quad (4.86)$$

The mean-squared data value is defined as

$$\bar{d}^2 = \frac{1}{N_R + N_I} \left[ \sum_{i=0}^{N_R-1} d_R(t_i)^2 + \sum_{j=0}^{N_I-1} d_I(t_j)^2 \right]. \quad (4.87)$$

The projection of the data onto the  $T_l$  model vector is given by

$$\begin{aligned} T_1 &\equiv \sum_{i=0}^{N_R-1} d_R(t_i) \cos(2\pi f t_i) \exp\{-\alpha t_i\} \\ &\quad - \sum_{j=0}^{N_I-1} d_I(t_j) \sin(2\pi f t_j) \exp\{-\alpha t_j\} \end{aligned} \quad (4.88)$$

for  $l = 1$  and

$$\begin{aligned} T_2 &\equiv - \sum_{i=0}^{N_R-1} d_R(t_i) \sin(2\pi f t_i) \exp\{-\alpha t_i\} \\ &\quad - \sum_{j=0}^{N_I-1} d_I(t_j) \cos(2\pi f t_j) \exp\{-\alpha t_j\} \end{aligned} \quad (4.89)$$

for  $l = 2$ . The matrix  $g_{kl}$  is defined as

$$g_{kl} \equiv \begin{pmatrix} a & c \\ c & b \end{pmatrix}, \quad (4.90)$$

with

$$a = \sum_{i=0}^{N_R-1} \cos^2(2\pi f t_i) \exp\{-2\alpha t_i\} + \sum_{j=0}^{N_I-1} \sin^2(2\pi f t_j) \exp\{-2\alpha t_j\} \quad (4.91)$$

where the sum over the cosine uses the  $t_i$  associated with the real data set, while the sum over the sine uses the  $t_j$  associated with the imaginary data set. Similarly,  $b$  is defined as

$$b = \sum_{i=0}^{N_R-1} \sin^2(2\pi f t_i) \exp\{-2\alpha t_i\} + \sum_{j=0}^{N_I-1} \cos^2(2\pi f t_j) \exp\{-2\alpha t_j\}, \quad (4.92)$$

where the sum over the sine uses the  $t_i$  associated with the real data set, while the sum over the cosine uses the  $t_j$  associated with the imaginary data set. Finally,  $c$  is defined as

$$\begin{aligned} c &= - \sum_{i=0}^{N_R-1} \cos(2\pi f t_i) \sin(2\pi f t_i) \exp\{-2\alpha t_i\} \\ &\quad + \sum_{j=0}^{N_I-1} \sin(2\pi f t_j) \cos(2\pi f t_j) \exp\{-2\alpha t_j\}, \end{aligned} \quad (4.93)$$

where the sum over the cosine-sine product uses the  $t_i$  associated with the real data set, while the sum over the sine-cosine product uses the  $t_j$  associated with the imaginary data set.

The integrals over  $A_1$  and  $A_2$  are both Gaussian integrals and will be evaluated first. The way we will do this is to make a simple observation about Gaussian quadrature integrals and then use that observation to evaluate the integrals. The observation is simply that Gaussians are symmetric in the amplitudes. Because they are symmetric, all integrating with respect to the amplitudes does is to constrain the amplitudes to their maximum posterior probability estimates and it introduces a volume factor, the determinate. Consequently, the maximum of the posterior probability as a function of the amplitudes is given by the solution to

$$\sum_{l=1}^2 g_{kl} \hat{A}_l = T_k. \quad (4.94)$$

For this simple  $2 \times 2$  matrix, the solution is given by

$$\hat{A}_1 = \frac{bT_1 - cT_2}{ab - c^2} \quad (4.95)$$

and

$$\hat{A}_2 = \frac{aT_2 - cT_1}{ab - c^2}. \quad (4.96)$$

The sufficient statistic is thus given by

$$\overline{h^2} = \sum_{j=1}^2 T_j \hat{A}_j. \quad (4.97)$$

This sufficient statistic can be rewritten as

$$\overline{h^2} \equiv \frac{bT_1^2 + aT_2^2 - 2cT_1T_2}{ab - c^2}. \quad (4.98)$$

While not obvious, it is this statistic that generalizes the discrete Fourier transform to nonuniformly nonsimultaneously sampled data. This statistic will reduce to the Lomb-Scargle periodogram, a weighted normalized power spectrum and the Schuster periodogram under appropriate conditions. finally the posterior probability for the frequency is given by

$$P(f|\sigma D_R D_I I) \propto \int d\alpha d\sigma \frac{P(\alpha|I)}{\sqrt{ab - c^2}} \sigma^{-(N_R + N_I - 2) - 1} \exp \left\{ \frac{(N_R + N_I) \overline{d^2} - \overline{h^2}}{2\sigma^2} \right\} \quad (4.99)$$

where we have assumed that the upper and lower amplitude bounds on the amplitude integrals may be extended to plus and minus infinity without introducing an appreciable error into the integral.

The integral over the standard deviation of the noise prior probability,  $\sigma$ , can be transformed into a Gamma function:

$$\int_0^\infty \sigma^{-N-1} \exp \left\{ -\frac{Q}{\sigma^2} \right\} = \frac{1}{2} \Gamma \left( \frac{N}{2} \right) Q^{-\frac{N}{2}} \quad (4.100)$$

from which one obtains

$$P(f|D_R D_I I) \propto \int d\alpha \frac{P(\alpha|I)}{\sqrt{ab - c^2}} \left[ (N_R + N_I) \overline{d^2} - h^2 \right]^{-\frac{N_R + N_I - 2}{2}} \quad (4.101)$$

which is of the form of Student's  $t$ -distribution, and we have dropped some constants that cancel when this probability density function is normalized. If the decay rate constant is a given, the prior,  $P(\alpha|I)$ , should be taken as a delta function and the integral in Eq. (4.101) evaluated analytically. Otherwise, the integral must be evaluated numerically.

### 4.3.6 How Probability Generalizes The Discrete Fourier Transform

We are now in a position to demonstrate how probability theory generalizes the discrete Fourier transform and what the effect of this generalization is on the aliasing phenomenon. We mentioned earlier that the sufficient statistic for frequency estimation is related to a power spectrum, and we demonstrate that next. Several simplifications must be made to reduce Eq. (4.98) to a power spectrum. First, we must be estimating the frequency of a stationary sinusoid. A stationary sinusoid has no decay, so  $\alpha = 0$ . Second, the data must be uniformly and simultaneously sampled. With these assumptions the matrix  $g_{kl}$ , Eq. (4.90), simplifies because  $c = 0$  and  $a = b = N_R = N_I = N$ , where  $N$  is the total complex data values. The sufficient statistic, Eq. (4.98), reduces to

$$\overline{h^2} = \frac{R(f)^2 + I(f)^2}{N} \quad (4.102)$$

and is the power spectrum defined in Eq. (4.53). The functions  $R(f)$  and  $I(f)$  were defined earlier, Eqs. (4.51) and (4.52), and are the real and imaginary parts of the discrete Fourier transform.

The Schuster periodogram, or power spectrum, is the sufficient statistic for frequency estimation in uniformly simultaneously sampled quadrature data given a stationary sinusoidal model, but we already have two generalizations to the discrete Fourier transform that were not contained in the definition, Eq. (4.44). First, the frequency appearing in Eq. (4.102) is a continuous parameter; it is not in any way restricted to discrete values. Probability theory indicates that there is information in the data at frequencies between the  $f_k$  in the discrete Fourier transform. Second, the frequency,  $f$ , is bounded only by our prior information. Probability theory does not indicate that the frequency must be less than the Nyquist critical frequency. Consequently, probability theory is telling one that aliases are real indications of the presence of a frequency and absolutely nothing in the data can tell one which is the correct frequency, *only* prior information can do that.

The Schuster periodogram is an optimal frequency estimator for uniformly simultaneously sampled quadrature data. However, this is not true for real data where the statistic was originally proposed. To see this suppose we have a real data set, so that either  $N_I = 0$  or  $N_R = 0$ . The  $g_{kl}$  matrix does simplify—many of the sums appearing in Eqs. (4.91)-(4.93) are zero—but the matrix remains nondiagonal and so Eq. (4.98) is the sufficient statistic for this problem, and is numerically equal to the Lomb-Scargle periodogram. The Schuster periodogram, however, is never a sufficient statistic for either real uniformly or nonuniformly sampled data. It can only be derived as an approximation to the sufficient statistic for this problem. To derive it two approximations must be made in the  $g_{kl}$  matrix. First, the off-diagonal element must be much smaller than the diagonal and so can be approximated as zero. The second approximation assumes the diagonal elements may be approximated by  $a = b = N/2$ . Both approximations ignore terms on the order of  $\sqrt{N}$  and are good

approximations when one has large amounts of data. In this discussion we have implicitly assumed that the times appearing in the cosine and sine transforms of the data making up the Schuster periodogram were evaluated at the times one actually has data. Trying to interpolate the data onto a uniform grid and then using a power spectrum is not justified under any circumstances.

Suppose now the signal is exponentially decaying with time, but otherwise the data remain uniformly and simultaneously sampled. Examining Eq. (4.90), we see that  $c = 0$  and  $a = b = N_{\text{eff}}$ , where  $N_{\text{eff}}$  is an effective number of complex data values and is given by

$$N_{\text{eff}} = \sum_{i=0}^{N-1} \exp\{-2\alpha t_i\}. \quad (4.103)$$

The sufficient statistic becomes

$$\overline{h^2} = \frac{R(f, \alpha)^2 + I(f, \alpha)^2}{N_{\text{eff}}} \quad (4.104)$$

which is a weighted power spectrum. The effective number of complex data values is equal to the number of complex data values,  $N$ , when the decay rate constant,  $\alpha$ , is zero, and is approximately,  $1/2\alpha$ , for densely sampled signals which decay into the noise. The reason for this behavior should be obvious: as the decay rate constant increases, for a fixed dwell time, fewer and fewer data values contribute to the estimation process. When the decay rate constant is large, the effective number of data values goes to zero and the data are uninformative about either the frequency or the decay rate constant.

The functions  $R(f, \alpha)$  and  $I(f, \alpha)$  are the real and imaginary parts of the discrete Fourier transform of the complex data weighted by an exponential of decay rate constant  $\alpha$ . In a weighted discrete Fourier transform, one multiplies the complex data by a weighting function

$$\text{Complex Weighted Data} = \mathbf{d}(t_i) \exp\{-\alpha t_i\} \quad (4.105)$$

and then performs the discrete Fourier transform on the weighted data.

In spectroscopic applications many different weighting functions are used. Indeed Varian NMR software comes with exponential, Gaussian, sine-bell and a number of others. In all of these cases, use of the weighting function in conjunction with a discrete Fourier transform amounts to estimating the frequency of a single sinusoid having a decay envelope described by the weighting function. If the weighting function does not mimic the decay envelope of the signal, then these procedures are, from the standpoint of parameter estimation, less than optimal. Of course, most of these weighting functions were developed with very different ideas in mind than parameter estimation. For example, the sine-bell is intended to increase resolution of multiple close lines, just as a Gaussian is used to transform the line shape of the resonances from Lorentzian to Gaussian in the hope that the Gaussian will be narrower in the frequency domain. Nonetheless, probability theory indicates that all of these procedures are estimating the frequency of a single sinusoid having a decay envelope described by the weighting function. The better this weighting function describes the true decay of the signal, the better the parameter estimates will be.

For exponential weighting, the spectroscopist must choose a value of  $\alpha$ . This is typically done so that the decay envelope of the weighting function matches the decay of the signal. This is equivalent to trying to locate the maximum of the joint posterior probability for the frequency and decay rate constant, Eq.(4.85). If one makes a contour plot of this joint posterior probability, this plot will have nearly elliptical contours with the major axis of the ellipse nearly parallel to the decay rate

constant axis (decay rate constants are less precisely estimated than frequencies), while the minor axis will be nearly parallel to the frequency axis. Thus, estimates of the frequency and decay rate constant are nearly independent of each other. Consequently, if the spectroscopist can guess the decay rate constant, even approximately, the frequency estimate he obtains will be almost as good as that obtained by doing a thorough search for the location of the joint maximum.

It is commonly believed that matched weighting functions (matching the shape of the weighting function to the observed decay of the data) increases the signal-to-noise ratio in the resulting power spectrum at the expense of broadening the peak, thereby decreasing the frequency resolution of the discrete Fourier transform. This is correct if the discrete Fourier transform is used as a spectral estimation procedure and one then tries to estimate multiple frequencies from this spectrum. However, from the standpoint of probability theory, it is only the single largest peak in the weighted discrete Fourier transform power spectrum that is relevant to frequency estimation, and then it is only a very small region around the location of the maximum that is of interest. All of the details in the wings around that peak are irrelevant to the estimation problem. If there are multiple peaks in the power spectrum, probability theory will systematically ignore them: *the weighted power spectrum is the sufficient statistic for single frequency estimation*; it does not estimate multiple frequencies, although one can show that under many conditions the sufficient statistic for the multiple frequency estimation problem is related to the multiple peaks in a power spectrum [2]. Given a multiple-frequency model, probability theory will lead one to other statistics that take into account the nonorthogonal nature of the sinusoidal model functions. These multiple-frequency models always result in parameter estimates that are either better than or essentially identical to those obtained from a power spectrum. They will be essentially identical to the power spectrum results when multiple, very well separated sinusoids are present, and they will always be better when overlapping resonances are present—see Bretthorst [4, 5, 6, 2, 7, 8, 9] for details.

Suppose we have nonuniformly but simultaneously sampled data. What will happen to the resulting Bayesian calculations? When we repeat the Bayesian calculation we find that absolutely nothing has changed. The number of effective data values,  $N_{\text{eff}}$ , is given by Eq. (4.103), the matrix  $g_{kl}$  is given by Eq. (4.90), just as the sufficient statistic is given by Eq. (4.104), and the posterior probability for the frequency is given by Eq. (4.101). Here the generalization to the discrete Fourier transform accounts for the fact that the times must be evaluated explicitly, the formula,  $t_j = j\Delta T$ , does not hold and one must substitute the actual time,  $t_i$  and  $t_j$ , into the equations. Missing observations, make no difference whatever to probability theory. Probability theory analyzes the data you actually obtain, regardless of whether the data are uniformly sampled or not. Indeed, in Bayesian probability theory there is no such thing as a missing data problem.

Having allowed the data to be nonuniformly sampled, we are in a position to see what is happening to the aliasing phenomenon. However, before addressing this, there is one last generalization that we would like to make. This generalization allows the data to be nonsimultaneously sampled. Because the samples are nonsimultaneous, the sums appearing in the discrete Fourier transform, Eqs. (4.51) and (4.52), are no longer correct. The terms appearing in these equations are the sine and cosine transforms of the real and imaginary data sets. When the data were simultaneously sampled, the sine and cosine transforms could be combined into a single sum. For nonsimultaneous time samples, this cannot be done. Each sine and cosine transform must have an independent summation index. If you examine the projections of the model onto the nonuniformly nonsimultaneously sampled data, Eqs. (4.88) and (4.89), you will find this is exactly what probability theory has done. The function  $T_1$  corresponds to the real part of the discrete Fourier transform and is the cosine transform of the real data minus the sine transform of the imaginary data; however, now it also accounts for the

nonuniform nonsimultaneous times. Similarly, up to a minus sign,<sup>3</sup>  $T_2$  corresponds to the imaginary part of the discrete Fourier transform. So the discrete Fourier transform has been generalized in the sense that the sine and cosine transforms now have separate summation indices. However, there is more to this generalization than using separate summation indices.

In the previous examples the matrix  $g_{kl}$ , Eq. (4.90), was diagonal with diagonal elements equal to the effective number of data values in each channel. For simultaneously sampled data these diagonal elements are equal. For nonsimultaneously sampled data, the diagonal elements remain the effective number of data values in each channel, but these are no longer equal. For simultaneous data samples, the zero off-diagonal element means that the integrals over the amplitudes are completely independent of each other. In the model, the function multiplying each amplitude may be thought of as an  $N$  dimensional vector. With nonsimultaneous samples these vectors are linear combinations of each other. The magnitude of the off-diagonal element is a measure of how far from orthogonal these vectors are. Consequently, the sufficient statistic, Eq. (4.98), is now taking into account the fact that these vectors are not orthogonal, the real and imaginary data can have different numbers of data values and that the effective number of data values is a function of both frequency and decay rate constant.

### 4.3.7 Aliasing

Now that we have finished discussing the generalizations of the discrete Fourier transform to nonuniformly nonsimultaneously sampled data, we would like to investigate some of the properties of these calculations to show what has happened to the aliasing phenomenon. Earlier, when we investigated the discrete Fourier transform of uniformly sampled data, we showed that, for frequencies outside the bandwidth, the power spectrum was a periodic function of frequency. What will happen to these aliases when we use nonuniformly nonsimultaneously sampled data? Are the aliases still there? If not, where did they go? One thing that should be obvious is that in nonuniformly nonsimultaneously sampled data the Nyquist critical frequency does not apply, at least not as previously defined, because the times are nonuniformly sampled. Consequently, nonuniformly nonsimultaneously sampled data will not suffer from aliases in the same way that uniformly simultaneously sampled data does.

To demonstrate this, we will again use simulated data. The simulated data will have exactly the same signal as the data shown in Fig. 4.1(A). The only difference between these two data sets will be the times at which the data were acquired. The nonuniformly nonsimultaneously sampled simulated data are shown in Fig. 4.3(A). In generating the times at which we have data we had to choose a sampling scheme. On some spectrometers it is possible to sample data exponentially, so we choose exponential sampling. In an exponentially sampled data set, a histogram of the time samples would follow an exponential distribution. Thus, there are more data samples at short times, and exponentially fewer at longer times. However, the discussion here pertains to all nonuniformly nonsimultaneously sampled data, not just to data with times that are exponentially sampled. The base 10 logarithm of the posterior probability for the frequency of a stationary sinusoid is shown in Fig. 4.3(B). This plot spans a frequency interval that is ten thousand times larger than the corresponding plot shown in Fig. 4.1(B). In Fig. 4.2(B), when we extended the region of interest to  $\pm 10f_{Nc} = \pm 500$  Hz we had 10 aliases. Yet here we have gone to  $\pm 50$  kHz and there are no aliases—where did the aliases go? Why do these data seem to have a bandwidth that is at least 10,000 times larger than in the first example?

<sup>3</sup> The minus sign comes about because of the use of Varian sign conventions in Eq. (4.60).

Figure 4.3: Nonuniformly Nonsimultaneously Sampled Sinusoid

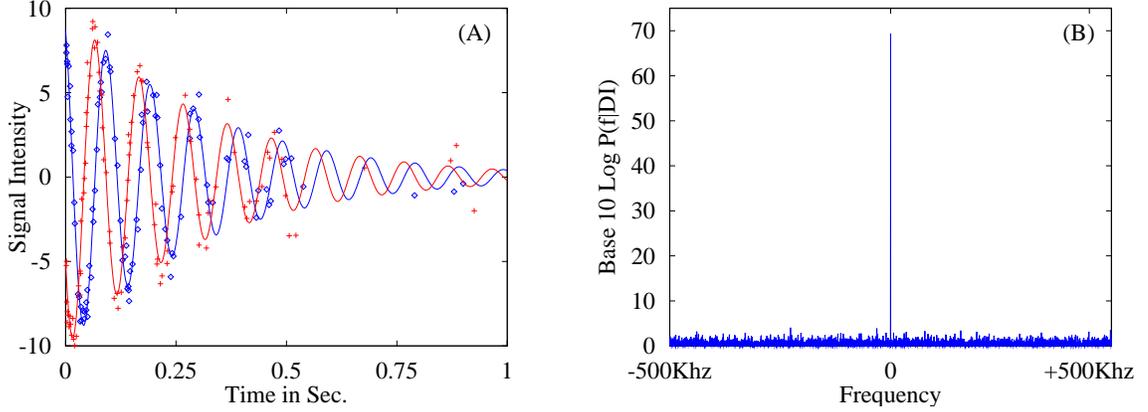


Figure 4.3: A Nonuniformly Nonsimultaneously Sampled Exponentially Decaying Sinusoid Panel (A) is computer simulated data. It contains the same exponentially decaying sinusoidal signal as shown in Fig. 4.1(A) plus noise of the same standard deviation. The lines represent the real and imaginary parts of the sinusoid. The location of the nonuniformly nonsimultaneously sampled data values are denoted by the isolated characters. Panel (B) is the base 10 logarithm of the posterior probability for the frequency given a stationary sinusoid model using these data. Note that this plot spans 10,000 times the Nyquist critical frequency for the uniformly sampled version of this data set shown in Fig. 4.1(A).

We showed earlier in Eqs. (4.56)-(4.59) that aliases come about because of the integer  $j$  used in specifying the uniformly sampled times,  $t_j = j\Delta T$ , in the discrete Fourier transform. In the present problem, nonuniformly nonsimultaneously sampled data, there is no  $\Delta T$  such that all of the acquisition times are integer multiples of this time; not if the times are truly sampled randomly. However, all data and times must be recorded to finite accuracy. This is true even of the simulated data shown in Fig. 4.3(A). Consequently, there must be a largest effective dwell time,  $\Delta T'$ , such that all of the times (both the real and imaginary) must satisfy

$$t_l = k_l \Delta T' \quad t_l \in \{\text{Real } t_i \text{ or Imaginary } t_j\} \quad (4.106)$$

where  $k_l$  is an integer. The subscript  $l$  was added to  $k$  to indicate that each of the times  $t_l$  requires a different integer  $k_l$  to make this relationship true. Of course, this was also true for uniformly sampled data: its just that for uniformly sampled data the integers were consecutive,  $k_l = 0, 1, \dots, N - 1$ . The effective dwell time is always less than or equal to the smallest time interval between data items, and is the least common denominator for all of the times. Additionally,  $\Delta T'$  is the dwell time one would have had to acquire data at in order to obtain a uniformly sampled data set with data items at each of the times  $t_i$  and  $t_j$ . The effective dwell time,  $\Delta T'$ , can be used to define a Nyquist critical frequency

$$f_{Nc} = \frac{1}{2\Delta T'}. \quad (4.107)$$

Aliases *must* appear for frequencies outside the bandwidth defined from  $\Delta T'$ .

The reason that aliases must appear for frequencies outside this bandwidth can be made apparent in the following way. Suppose we have a hypothetical data set that is sampled at  $\Delta T'$ . Suppose

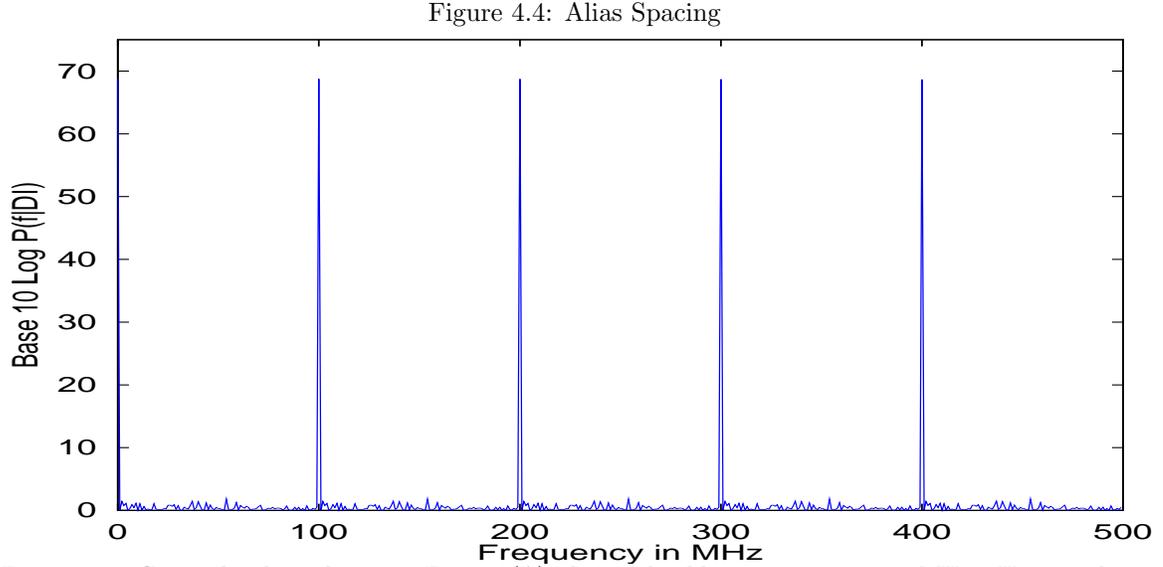


Figure 4.4: Given the data shown in Fig. 4.3(A) aliases should appear every 100 MHz. Here we have evaluated the logarithm of the posterior probability for the frequency of a stationary sinusoid at frequencies given by  $10 + n \times 10^6$  Hz. Aliases should appear at  $n = 100, 200, 300$ , etc.

further, the hypothetical data are zero everywhere except at the times we actually have data, and there the data are equal to the appropriate  $d_R(t_i)$  or  $d_I(t_j)$ . If we now compute the discrete Fourier transform of this hypothetical data set, then by the analysis done in Eqs. (4.56)-(4.59) the Nyquist critical frequency of this data set is  $1/2\Delta T'$  and frequencies outside this bandwidth are aliased. Now look at the definitions of  $T_1$  and  $T_2$ , Eqs. (4.88) and (4.89), for the data set we actually have. You will find that these quantities are just the real and imaginary parts of the discrete Fourier transform of our hypothetical data set. The zeros in the hypothetical data set cannot contribute to the sums in the discrete Fourier transform: they act only as place holders, and so the only part of the sums that survive are just where we have data. By construction that is just what Eqs. (4.88) and (4.89) are computing. So aliases *must* appear at frequencies greater than this Nyquist critical frequency.

For the data shown in Fig. 4.3, the times and the data were recorded to 8 decimal places in an ASCII file. This file was then used by another program to compute the posterior probability for the frequency, Eq. (4.98). Because the data were recorded to 8 decimal places, a good guess for the effective dwell time would be  $\Delta T' = 10^{-8}$  Sec. This would correspond to a Nyquist critical frequency of  $f_{Nc} = 5 \times 10^7$  Hz. The first alias of the 10 Hz frequency should appear at 100,000,010 Hz. If we evaluate the base 10 logarithm of the posterior probability at frequencies given  $(10 + n \times 10^6)$  Hertz, we should see peaks at  $n = 100, 200, 300$  etc. This plot is shown in Fig. 4.4. Note that the aliases are right at the expected frequencies. An extensive search from zero up to 100 MHz uncovered no aliases prior to the one just above 100 MHz. This suggests that the effective bandwidth of the 100 complex data values shown in Fig. 4.3(A) is 100 MHz! That is one million times larger than the bandwidth of the data shown in Fig. 4.1. Indeed, the effective dwell time,  $\Delta T'$ , was defined as the maximum time for which all of the  $t_i$  and  $t_j$  are integer multiples of  $\Delta T'$ . The Nyquist critical frequency computed from  $\Delta T'$  is the *smallest* frequency for which the argument of the complex exponential in Eq. (4.57)

is an integer multiple of  $2\pi$ . Consequently,  $1/2\Delta T'$  is the Nyquist critical frequency for these data and there are no aliases within this implied bandwidth.

The fact that the data was recorded to 8 decimal places and that the bandwidth of this simulated data set is  $10^8$  Hz brings up another curious thing about aliases. Aliasing is primarily a phenomenon that concerns the times in the discretely sampled data, the data itself are almost irrelevant to aliasing. In the example we are discussing the times were recorded to 8 decimal places. If we had recorded the times to only 7 decimal places the aliases would have been in different places. If the last significant digit in the times is truncated, the bandwidth will be at least a factor of 10 lower. Of course, we must qualify this somewhat, in the case we are discussing, a change in the 8'th decimal place changes the sines and cosines so little that the only noticeable effect is on the location of the aliases. However, if we were to continue truncating decimal places we will eventually reach the point where the times are so far from the correct values that we are essentially computing nonsense.

So far we have shown that the data have an effective bandwidth of  $1/\Delta T'$  but that is not quite the same thing as showing that frequencies anywhere in this 100 MHz interval can be correctly estimated. There is another time, the minimum time between data values,  $\Delta T_M$ , which might be of some importance. It might be thought that the data can show no evidence for frequencies outside a band of width  $1/\Delta T_M$ . In the data set shown in Fig. 4.3(A),  $\Delta T_M = 0.00000488$  Sec. This would correspond to a bandwidth of roughly 200 kHz, a tremendous frequency range, but smaller than the effective bandwidth of 100 MHz by a factor of roughly 500. Which is correct?

The arguments given earlier prove that it is the effective dwell time,  $\Delta T'$ , and not the minimum interval between data values,  $\Delta T_M$ , that is the important quantity. However, to illustrate that  $\Delta T'$  is the critical time, it is a simple matter to generate data that contain a frequency in the range ( $[1/\Delta T_M] < f < [1/\Delta T']$ ) and then compute the base 10 logarithm of posterior probability for the frequency of a stationary sinusoid over the entire range ( $0 \leq f \leq 1/\Delta T'$ ). From a practical standpoint this is nearly impossible for data with a Nyquist critical frequency of 50 MHz; this frequency will have to be lowered. This can be done by truncating the exponentially sampled times to 5 decimal places, and then generating the simulated signal using these truncated times. Truncating the times lowers the Nyquist critical frequency to 50 kHz. Additionally, when these simulated data shown in Fig. 4.5(A) were generated, no times were generated closer together than 0.0001 Sec. This time corresponds to a Nyquist critical frequency of 5 kHz; so there is a factor of 10 difference between the Nyquist critical frequency and the frequency calculated from  $\Delta T_M$ . The simulated acquisition parameters used in generating these data are similar to those used previously, *i.e.*,  $N_R = N_I = 100$ ,  $\alpha = 3 \text{ Sec.}^{-1}$ , Amplitude = 10,  $\sigma = 1$ . The only difference is that the simulated frequency is 50 kHz, a full factor of 5 higher than the minimum sampling time would give as the highest resolvable frequency, and equal to the Nyquist critical frequency for these data. However, the region plotted ( $0 \leq f \leq 100 \text{ kHz}$ ), has been shifted upward by 50 kHz, so the 50 kHz frequency is in the middle of the plotted region, and this region should be free of aliases. The base 10 logarithm of the posterior probability for the frequency of a stationary sinusoid is shown in Fig. 4.5(B). It was evaluated at every 1 Hz from 0 to 100,000 Hz. This frequency resolution is more than enough to ensure that if aliases exist, multiple peaks would be present in this plot. Note that there is a single peak and it is located at 50 kHz, the frequency of the simulated resonance. So the critical time is indeed  $\Delta T'$  and the bandwidth of these data is 100 kHz.

Having shown that the critical time is the effective dwell time  $\Delta T'$  and that there are no aliases in the full bandwidth implied by  $\Delta T'$ , one might be tempted to think that that is the end of the story. However, that is not quite correct. While there are no aliases in the bandwidth implied by  $\Delta T'$ , it is possible for there to be multiple peaks which are artifacts related to the effective dwell

Figure 4.5: Which Is The Critical Time

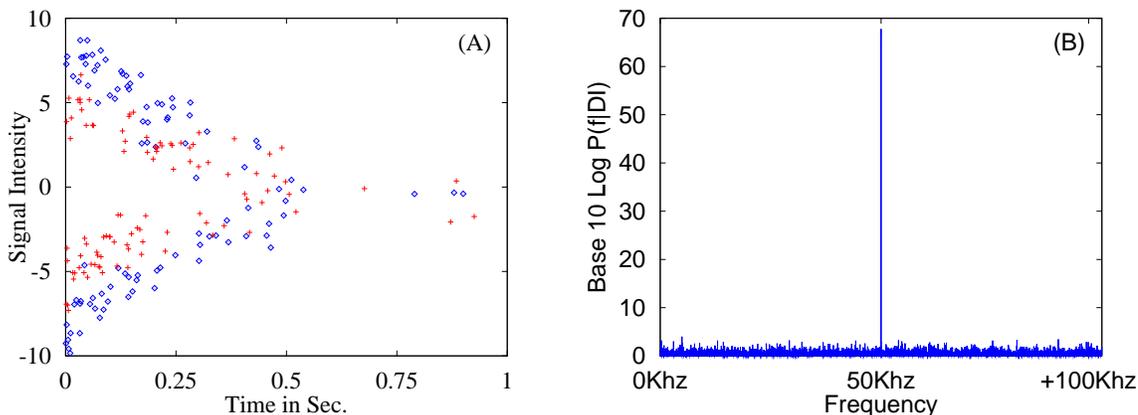


Figure 4.5: Which Is The Critical Time:  $\Delta T'$  Or  $\Delta T_M$ ? Panel (A) is computer simulated data. Except for the frequency, it contains the same signal as that shown in Fig. 4.1(A). The frequency in these data is 50 kHz. The positions of the nonuniformly nonsimultaneously sampled data values are denoted by the isolated characters. Panel (B) is the base 10 logarithm of the posterior probability for the frequency of a stationary sinusoid given these data.

time. These artifacts are not aliases in the sense that they are not exact replicas of the main peak; rather they are evidence for the resonance, and their height is directly related to how strongly the data indicate the presence of a resonance at that frequency. To see how multiple peaks might occur, suppose we have the data shown in Fig. 4.1(A); with 4 additional nonuniformly but simultaneously sampled complex data values at 0.001, 0.015, 0.025, and 0.037 seconds respectively.<sup>4</sup> These 4 complex data values were generated by sampling the same signal plus noise as shown in Fig 4.1(A), but at the four nonuniformly sampled times. Now, according to the analysis done in this chapter, the Nyquist critical frequency for the combined data is  $1/(2 \times 0.001 \text{ Sec.}) = 500 \text{ Hz}$ ; this bandwidth is exactly the same as the frequency interval shown in Fig. 4.2 where we had 10 aliases. In principle, these 4 complex data values should increase the bandwidth by a factor of 10 and should destroy the aliasing phenomenon in the  $\pm 500 \text{ Hz}$  frequency band. A plot of the base 10 logarithm of the posterior probability for the frequency of a stationary sinusoid given the combined data is shown in Fig. 4.6(A). Note that there are 10 peaks in Fig. 4.6(A), just as there are 10 peaks in Fig. 4.2, but the peaks are not of the same height, so they are not true aliases. To determine which peak corresponds to the true frequency one must assign a bounded prior probability for the frequency (to eliminate the true aliases at frequencies above 500 Hz and below -500 Hz) and then normalize the posterior probability. The fully normalized posterior probability for the frequency of a stationary sinusoid is shown in Fig. 4.6(B). The fully normalized posterior probability density function has a single peak at 10 Hz, the true frequency. In this example, the 4 extra nonuniformly sampled complex data values were enough to raise the probability for the true frequency several orders of magnitude, and when the posterior probability for the frequency was normalized, the vast majority of the weight in the

<sup>4</sup>The fact that the 4 complex data values are simultaneously sampled is irrelevant. The results of this analysis would be the same regardless if these 8 total data items were simultaneously sampled or not.

Figure 4.6: Example, Frequency Estimation

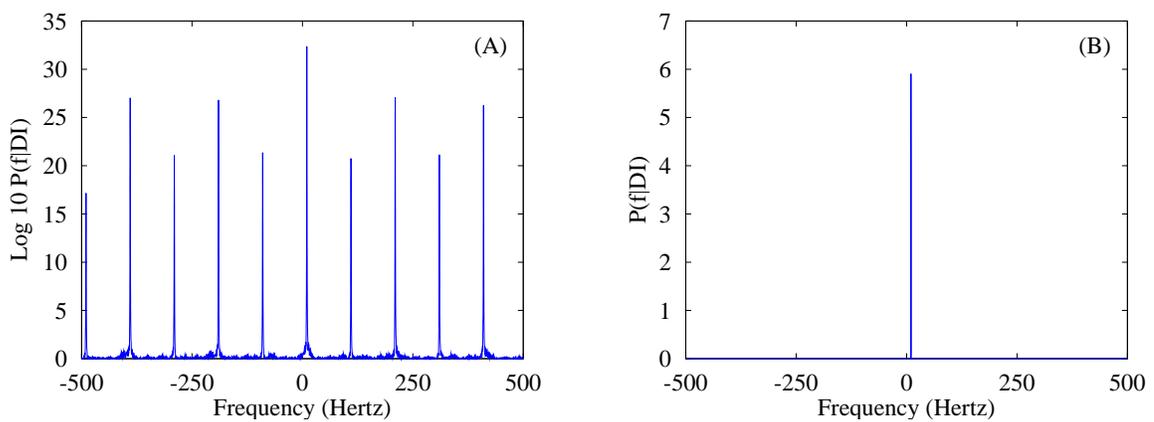


Figure 4.6: Panel (A) is the base 10 logarithm of the posterior probability for the frequency of a stationary sinusoid using the data shown in Fig. 4.1(A) with four additional data values with sample times given by 0.001, 0.015, 0.025, and 0.037 seconds respectively. These four data values were generated using the same signal and signal-to-noise ratio as those shown in Fig. 4.1(A). The only difference is the nonuniformly sampled times. Note that we now have multiple peaks, but they are not true aliases because they are not of the same height. Panel (B) is the fully normalized posterior probability for the frequency of a stationary sinusoid given these data, note that all of the peaks except the one at the true frequency, 10 Hz, have been exponentially suppressed.

posterior distribution was concentrated around the true frequency, consequently all of the spurious peaks seen in Fig. 4.6(A) were exponentially suppressed.

In preparing this example the number of nonuniformly sampled data values had to be chosen. Initially 6 complex nonuniformly sampled data values were generated. But this was abandoned because, when the base 10 logarithm of the posterior probability for a stationary frequency was plotted, the spurious peaks were only about 5 percent of the main peak and so did not illustrate the points we wanted to make. However, it does indicate that it does not take many nonuniformly sampled data values to eliminate these spurious peaks. As few as 10 percent should completely eliminate them. Nonetheless, it is possible for the fully normalized posterior probability for the frequency to have multiple peaks in data that contain only a single frequency. If this happens, it indicates that the data simply cannot distinguish which of the possibilities are the true frequency. The only recourse will be to obtain more measurements, preferably at nonuniformly nonsimultaneously sampled times.

The preceding example reiterates what has been said several times: the discrete Fourier transform power spectrum, the Schuster periodogram, a weighted power spectrum, the Lomb-Scargle periodogram, and the generalizations to these presented in this chapter are sufficient statistics for frequency estimation given a *single* frequency model. Multiple peaks in the discrete Fourier transform and its generalizations are not necessarily evidence for multiple frequencies. The only way to be certain that multiple frequencies are present is to postulate models containing one, two, etc. frequencies and to then compute the posterior probability for these models. Depending on the outcome of that calculation, one can then estimate the frequencies from the appropriate model.

### 4.3.8 Parameter Estimates

So far the discussions have concentrated on the discrete Fourier transform, how probability theory generalizes it to nonuniformly nonsimultaneously sampled data, and how these generalizations affect aliases. Now we are going to discuss the effect of nonuniform nonsimultaneous sampling on the parameter estimates. In this discussion we are going to estimate the parameters using the data shown in Fig. 4.1(A) and in Fig. 4.3(A). These two data sets contain exactly the same signal and each data set contains Gaussian white noise drawn from a Gaussian random number generator of unit standard deviation. However, the noise realizations in each data set are different, and this will result in slightly different parameter estimates for each data set. Nonetheless these two data sets provide an excellent opportunity to demonstrate how nonuniform nonsimultaneous sampling affects the parameter estimates.

We will discuss estimation of the frequency, decay rate constant and the amplitude. We will not discuss estimation of the phase and standard deviation of the noise as these are of less importance. The posterior probability for the frequency, decay rate constant, and amplitude are shown in panels (A), (B) and (D) of Fig. 4.7 respectively. Each of these plots is the fully normalized marginal posterior probability for the parameter of interest independent of all of the other parameters appearing in the model. Panel (C) contains the absolute-value spectra computed from these two data sets and will be used to compare Fourier transform estimation procedures to the Bayesian calculations. The solid lines in these plots were computed from the nonuniformly nonsimultaneously sampled data shown in Fig. 4.3(A); while the curves drawn with open characters were computed using the uniformly sampled data shown in Fig. 4.1(A).

A Markov chain Monte Carlo simulation was used to compute the marginal posterior probability for each parameter. All of the parameters appearing in the model were simulated simultaneously, thus the Markov chain Monte Carlo simulation simulated the joint posterior probability for all

Figure 4.7: Estimating The Sinusoids Parameters

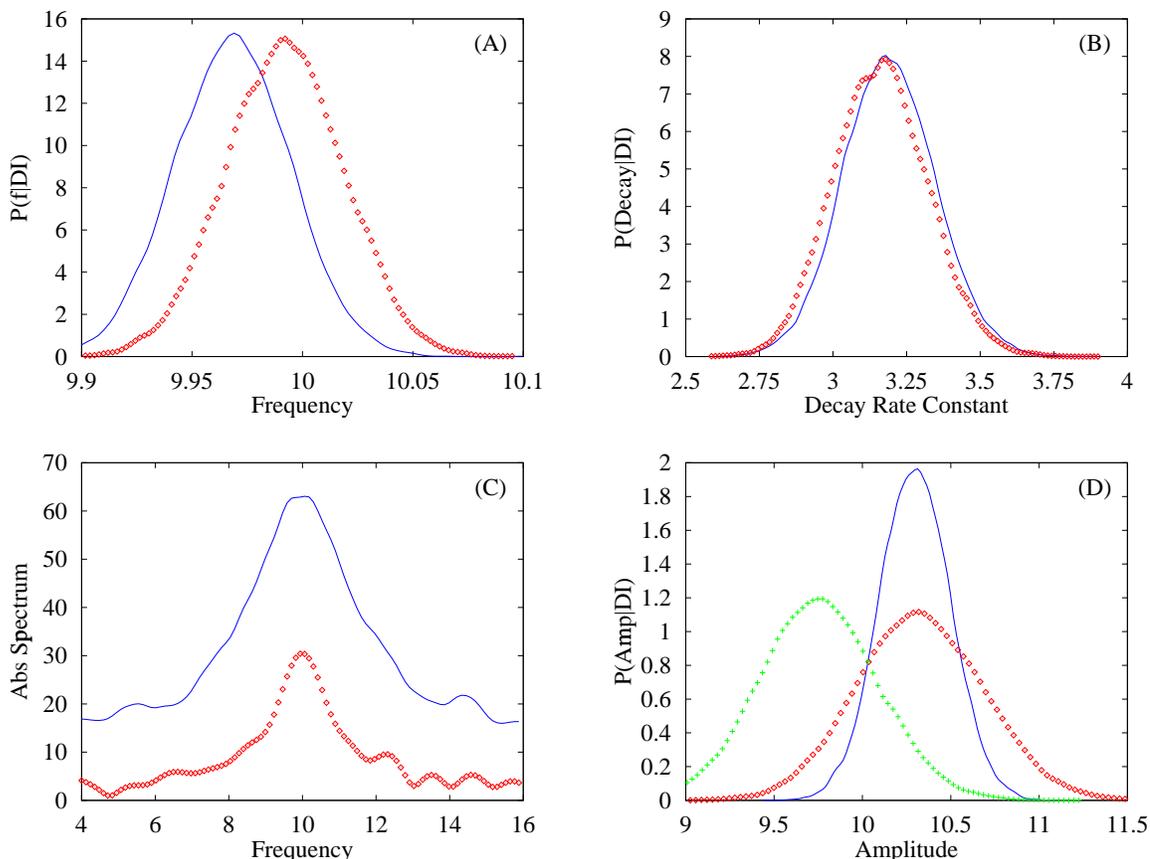


Figure 4.7: Estimating The Parameters The posterior probability of the parameter of interest was computed given the nonuniformly nonsimultaneously sampled data, solid lines, and was computed given the uniformly sampled data, open characters. Panel (A) is the posterior probability for the frequency, (B) the posterior probability of the decay rate constant, (D) the posterior probability for the amplitude. Panel (C) is the absolute-value spectrum computed for the uniformly sampled data and for the nonuniformly nonsimultaneously sampled data. The extra curve in panel (D), the plus signs, is the posterior probability for the amplitude computed from a nonuniformly nonsimultaneously sample data set having exactly the same signal with exactly the same signal-to-noise level but with times that were generated from a uniform random number generator.

the parameters. This was done for computational convenience; *i.e.*, it was easier to do a single Markov chain Monte Carlo simulation than to do five separate calculations, one for each parameter appearing in the model. Because the probability density functions shown in panels (A), (B) and (D) were formed by computing a histogram of the Markov chain Monte Carlo samples there are small irrelevant artifacts in these plots that are related to the number of samples drawn from the simulation. For more on Markov chain Monte Carlo methods and how these can be used to implement Bayesian calculations see Gilks [24] and Radford [46].

The marginal posterior probability for the frequency is shown in panel (A). This is the fully normalized marginal posterior probability for the frequency independent of all of the other parameters, Eq. (4.101). Note that the true frequency, 10 Hz, is well covered by the posterior probability computed from both the uniformly (open characters) and nonuniformly nonsimultaneously (solid line) sampled data. Also note that these distributions are almost identical in height and width. Consequently, both the uniform and nonuniformly nonsimultaneously sampled data have given the same parameter estimates to within the uncertainty in these estimates. Of course, the details for each estimated differ, because the noise realizations in each data set differ. Consequently, the frequency estimate is not strongly dependent on the sampling scheme. Indeed this can be derived from the rules of probability theory with the following proviso: the two sampling schemes must cover the *same* total sampling time and must sample the signal in a reasonably dense fashion so that sums may be approximated by integrals, [2, 8]. Having said this, we must reemphasize that this is only true for frequency estimates using data having sampling schemes covering the *same* total sampling time; it is not true if the sampling times differ nor is it necessarily true of the other parameters appearing in the model. Indeed one can show that for a given number of data values, the precision of the frequency estimate for a stationary sinusoid is inversely proportional to the total sampling time, provided one samples the signal in a reasonably dense fashion. Thus, sampling 10 times longer will result in frequency estimates that are 10 times more precise. As noted in Bretthorst [2] this is equivalent to saying that for frequency estimation data values at the front and back of the data are most important in determining the frequency, because it is in these data that small phase differences are most highly magnified by the time variable. This could be very useful in some applications; but of course, it will not help in NMR applications where the signal decays away.

We have also plotted the absolute-value spectra computed from these two data sets, Fig. 4.7(C). Note that the peaks of these two absolute-value spectra are at essentially the same frequency as the corresponding peaks in panel (A); although they are plotted on differing scales. If the absolute value spectrum is used to estimate the frequency, one would typically use the frequency of the peak as the estimate and then claim roughly the half-width-at-half-height as the uncertainty in this estimate. For these two data sets that is about 10 plus or minus 2 Hz. The two fully normalized posterior probabilities shown in panel (A) span a frequency interval of only 0.2 Hz. This frequency interval is roughly 6 standard deviations. Thus the frequency has been estimated to roughly 10 Hz with an uncertainty of  $0.2/6 \approx 0.03$  Hz; a 60 fold reduction in the uncertainty in the frequency estimate.

One last note before we begin the discussion of estimating the decay rate constant, we reiterate that all of the details in the wings of the absolute-value spectrum shown in panel (C) are irrelevant to the frequency estimation process. The posterior probability for the frequency has peaked in a region that is very small compared to the scale of these wings, all of the information about the frequency estimate is contained in a very small region around the largest peak in the absolute-value spectrum.

The marginal posterior probability for the decay rate constant is shown in Fig. 4.7(B). Here we again find that the parameter estimates from both data sets are essentially identical in all of there

relevant details. Both probabilities peak at nearly the same value of the decay rate constant, both have nearly the same width, and therefore the same standard deviation; thus like frequency estimates, the estimates for the decay rate constants do not strongly depend on the sampling scheme. In principle the accuracy of the estimates for the decay rate constants scale with time just like the frequency estimates, of course, with decaying signals this is of little practical importance. Note that the decay rate constant has been estimated to be about  $3.2 \pm 0.3 \text{ Sec.}^{-1}$  at one standard deviation. The true value is  $3 \text{ Sec.}^{-1}$ , so both sampling schemes give reasonable estimates of the decay rate. If one were to try and estimate the decay rate constant from the absolute-values spectrum, the half-width-at-half-height would normally be used, here that is about  $2 \text{ Sec.}^{-1}$  and no claim about the accuracy of the estimate would be made.

The marginal posterior probability for the amplitude of the sinusoid is shown in Fig. 4.7(D). The amplitude,  $A$ , was defined in Eq. (4.60). In this chapter we did not directly talk about amplitude estimation (see Bretthorst [8] for a discussion of this subject), rather we treated the amplitudes of the sine and cosine model functions as nuisance parameters and removed them from the posterior probability for the other parameters. We did this because we wished to explore the relationships between frequency estimation using Bayesian probability theory and the discrete Fourier transform. However, the Markov chain Monte Carlo simulation used  $A \cos(2\pi ft + \theta) \exp\{-\alpha t\}$  as the model for the real data, so it was a trivial matter to compute the posterior probability for the amplitude. If you examine Fig. 4.7(D) you will note that now we do have a real difference between the uniform (open characters) and the nonuniformly nonsimultaneously sampled data (solid lines). The amplitude estimates from the nonuniformly nonsimultaneously sampled data are a good factor of 2 more precise than the estimates from the uniformly sampled data. One might think that this is caused by the nonuniform nonsimultaneous sampling and this would be correct, but not for the obvious reasons. If you examine panel (D) you will note that we have plotted a third curve (plus signs). This curve is the posterior probability for the amplitude computed from data with the exact same signal and signal-to-noise ratio, but having times that are nonuniformly nonsimultaneously sampled where the times were generated from a uniform random number generator. We will call this data set the uniform-randomly sampled data. Note that the height and width of the posterior probabilities computed from both the uniformly and the uniform-randomly sampled data are essentially the same, so by itself the nonuniform nonsimultaneous sampling did not cause the amplitude estimates to improve. The amplitude estimate improved because exponential sampling gathered more data where the signal was large. The accuracy of the amplitude estimate is proportional to the standard deviation of the noise and inversely proportional to square root of the effective number of data values. Because exponential sampling gathered more data where the signal was large, its effective number of data values was larger and so the amplitude estimate improved. In this case, the improvement was about a factor of 2, so the exponential sampling had an effective number of data values that was about a factor of 4 larger than for the uniformly or uniform-randomly sampled data. This fact is also reflected in differing heights of the absolute value spectra plotted in Fig. 4.7(C). The peak height of an absolute value spectrum is proportional to the square root of the effective number of data values. In panel (C) the spectra computed from the uniformly sampled data set, open characters, is roughly a factor of 2 lower than the height of the spectrum computed from the exponentially sampled data set, solid line.

## 4.4 Summary and Conclusions

Probability theory when interpreted as logic is a quantitative theory of inference, just as mathematics is a quantitative theory of deduction. Unlike the axioms of mathematics, the *desiderata* of probability theory do not assert that something is true; rather they assert that certain features of the theory are desirable. Stated broadly these *desiderata* are degrees of belief are represented by real numbers; probability theory when interpreted as logic must qualitatively correspond to common sense; and when the rules for manipulating probabilities allow the evidence to be combined in more than one way, one must reach the same conclusions, i.e., the theory must be self consistent. These qualitative requirements are enough to uniquely determine the content of the theory [31, 28, 30, 20]. The rules for manipulating probabilities in this theory are just the standard rules of statistics plus Bayes' theorem. Thus Bayesian probability theory reinterprets the rules for manipulating probabilities. In this theory a probability represents a state of knowledge, not a state of nature.

Because a probability represents a reasonable degree of belief, a Bayesian can assign probabilities to hypotheses which have no frequency interpretation. Thus, problems such as: “What is the probability of a frequency  $\omega$ , independent of the amplitude and phase of the sinusoid, given the data?” or “Given several possible models of the data, which model is most probable?” make perfect sense. The first question is a parameter estimation problem and assumes the model to be correct. The second question is more general; it is a model selection problem and does not assume the model to be correct. In the following Chapters we will have need for both types of calculations. Most of the problems we will deal with are parameter estimation problems. However, whenever a model is selected in which an “unknown” options is invoked, a model selection calculation is be done. In addition, some of the calculations, such as the big peak/little peak calculation have a model selection calculation built into a parameter estimation problem.

## Chapter 5

# Given Exponential Model

The Given Exponential Package estimates amplitudes and decay rate constants in data that are known to contain signals which are sums of exponentials. This signal may or may not contain a constant offset, and the number of exponentials in the data need not be known. The calculations presented in this Chapter describe the given model, i.e., given the number of exponentials and whether a constant is or is not present. We describe the calculations for the unknown number of exponentials in Chapter 6. The input data files analyzed by this package are Ascii and may be input from Ascii files, a peak pick, a Bayes Analyze file or they may be loaded from an image pixel. When “Exponential” package button is activated, the interface window shown in Fig. 5.1 is displayed. This is the interface for both “Given” and “Unknown” number of exponentials. To use this package, you must do the following:

**Select** the exponential package from the Package menu.

**Load** one or more Ascii data sets using the Files menu. When a data set is successfully loaded the data is plotted in the Ascii Data viewer.

**Set** the number of exponentials in the model using the Model/Order selection menu.

**Check** the Model/Constant box if the data contains an offset.

**Check** the Analysis Options/Find Outliers box if you suspect outliers are present in the data.

**Review** the prior probabilities for the decay rate constant using the Prior Viewer.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

Figure 5.1: The Given And Unknown Number Of Exponential Package Interface

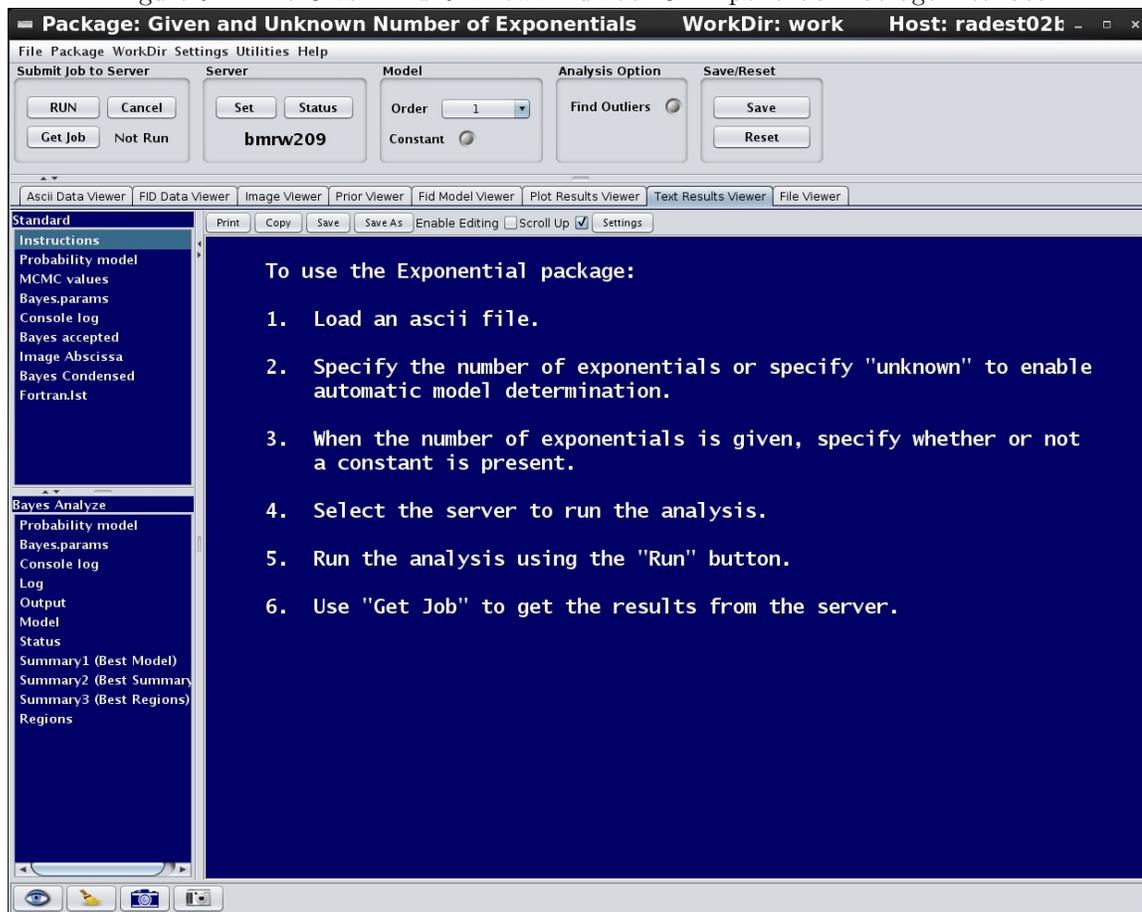


Figure 5.1: When the Exponential package is selected, this is the displayed interface. The package setup widgets, in this case labeled “Model,” allow you to select the number of exponentials in the data and they allow you to specify whether or not a constant offset is present. Additionally, the prior viewer can be used to set the prior probability for the decay rate constants. Note in this package the posterior probabilities are marginal posterior probabilities, so there are no adjustable prior probabilities for the amplitudes.

## 5.1 The Bayesian Calculation

The sums of exponential package process data that are known to contain exponential signals of the form:

$$d_{ik} = C_k + \sum_{j=1}^m A_{jk} \exp \{-\alpha_j t_{ik}\} + n_{ik} \quad (5.1)$$

where  $d_{ik}$  is the  $i$ th data value in the  $k$ th data set,  $C_k$  is the constant offset in the  $k$ th data set,  $m$  is the number of exponentials,  $A_{jk}$  is the amplitude or intensity of the  $j$ th exponential in the  $k$ th data set,  $\alpha_j$  is the  $j$ th exponential decay rate constant,  $t_{ik}$  is the  $i$ th abscissa value in the  $k$ th data set, and as this equation implies, the abscissa values and the sampling times need not be the same from one data set to the next. In Fig. 5.1, the model widgets, allow you to set the number of exponentials and allow you to indicate if a constant is present. Additionally, the prior viewer can be used to set the prior probability for the decay rate constants. Finally, the number of data sets  $n$  is determined by the number of Ascii data sets loaded into the analysis. If 5 Ascii data sets are loaded, the  $n = 5$ .

When the number of exponentials are given, the problem is one of parameter estimation and the program that implements this Bayesian calculation computes the marginal posterior probability for each of the parameters appearing in the model. For example, the marginal posterior probability for the decay rate constant,  $\alpha_j$ , is computed from the joint posterior probability for all of the parameters using the sum rule:

$$P(\alpha_j|DI) = \int d\{A\}d\{C\}d\{\sigma\}d\alpha_1 \dots d\alpha_{j-1}d\alpha_{j+1} \dots d\alpha_m P(\{A\}\{C\}\{\sigma\}\{\alpha\}|DI) \quad (5.2)$$

where the integral is over all of the parameters except the parameter of interest, in this case  $\alpha_j$ . The notation,  $\{\cdot\}$  is being used to stand for all of the enclosed quantities. So for example if there are three exponentials and 5 data sets, there are 15 total amplitudes represented by  $\{A\}$ . Similarly, because there is one constant per data set, then there would be 5 total constants represented by  $\{C\}$ . The right-hand side of this equation was factored using the rules of probability theory and Bayes Theorems' [1]

$$\begin{aligned} P(\{A\}\{C\}\{\sigma\}\{\alpha\}|DI) &\propto \left[ \prod_{l=1}^m P(\alpha_l|I) \right] \\ &\times \left[ \prod_{k=1}^n P(C_k|I)P(\sigma_k|I) \right] \\ &\times \left[ \prod_{k=1}^n \prod_{j=1}^m P(A_{jk}|I) \right] \\ &\times \left[ \prod_{k=1}^n P(D_k|\{A\}_k\{\alpha\}C_k\sigma_k I) \right] \end{aligned} \quad (5.3)$$

where  $m$  is the given number of exponentials,  $n$  is the number of data sets,  $P(\alpha_l|I)$  is the prior probability for the  $l$ th decay rate constants,  $P(C_k|I)$  is the prior probability for the constant in the  $k$ th data set,  $P(\sigma_k|I)$  is the prior probability for the standard deviation of the noise,  $P(A_{jk}|I)$  is the prior probability for the amplitudes of the  $j$ th exponential in the  $k$ th data set, and  $P(D_k|\{A\}_k\{\alpha\}C_k\sigma_k I)$

is the direct probability or likelihood of data set  $D_k$  given the amplitudes,  $\{A\}_k$ , in the  $k$ th data set, the constant offset,  $C_k$ , and the standard deviation of the noise,  $\sigma_k$ .

The various probabilities are assigned as follows. The prior probability for the decay rate constant,  $P(\alpha_l|I)$  are user defined and the functional form of this prior probability can be any of the following: a uniform prior probability, a bounded Gaussian, an exponential prior probability or a prior positive. When the interface initializes the exponential package, a default prior probability for the decay rate constant is defined using the maximum value of the abscissa. This default prior probability is a bounded Gaussian prior probability that goes through 4.5 e-foldings over the decay rate constant ranges shown in the interface. If the maximum decay rate constant is  $\alpha_{\text{Max}}$ , then the default prior probability for the decay rate constant is given by

$$P(\alpha_l|I) = \begin{cases} \frac{1}{N_l} \exp\{-\frac{\alpha_l^2}{2\sigma_l^2}\} & \text{if } (0 \leq \alpha_l \leq \alpha_{\text{Max}}) \\ 0 & \text{otherwise} \end{cases} \quad (5.4)$$

where  $N_l$  is a normalization constant. If we make the assumption that the exponential signal components decay to no more than 20 e-foldings over the run of the data, then we can define a maximum decay rate constant:

$$\alpha_{\text{Max}} T_{\text{Max}} \equiv 20, \quad (5.5)$$

$T_{\text{Max}}$  is the maximum abscissa value, so

$$\alpha_{\text{Max}} = \frac{20}{T_{\text{Max}}}, \quad (5.6)$$

and  $\sigma_l$  is set so that the prior goes through 4.5 e-foldings:

$$\frac{(\alpha_{\text{Max}})^2}{2\sigma_l^2} = 4.5. \quad (5.7)$$

Consequently,

$$\sigma_l \approx \frac{6.666666}{T_{\text{Max}}}. \quad (5.8)$$

The user assigns only a single prior probability for the decay rate constants and the interface uses this prior for all decay rate constants.

The prior probabilities for the amplitudes, the  $P(A_{jk}|I)$ , are assigned using broad Gaussian that range from  $-\infty$  to  $+\infty$ ,

$$P(A_{jk}|I) = \left(\frac{2\pi\sigma_k^2}{\delta^2}\right)^{-\frac{1}{2}} \exp\left\{-\frac{\delta^2}{2\sigma_k^2} A_{jk}^2\right\} \quad (5.9)$$

where  $\delta = 0.01$  in the given and unknown number of exponentials. In these package you cannot change the prior probability for the amplitudes. Similarly, the prior probability for the constant offset in each data set,  $P(C_k|I)$ , is also assigned as a Gaussian prior probability using

$$P(C_k|I) = \left(\frac{2\pi\sigma_k^2}{\delta^2}\right)^{-\frac{1}{2}} \exp\left\{-\frac{\delta^2}{2\sigma_k^2} C_k^2\right\} \quad (5.10)$$

the same functional form with  $\delta$  also equal to 0.01. Consequently, it is possible for this package to estimate either the amplitudes or the constant offsets to be negative when the prior information available to the user would constrain it to be positive. If this is unacceptable, in the Enter Ascii Model package there is a full suite of exponential models that allow you to control the prior range on the amplitudes as well as the decay rate constants, Chapter 20. The prior probability for the standard deviation of the noise, the  $\sigma_k$ , were assigned using Jeffreys' priors [33],

$$P(\sigma_k|I) \propto \frac{1}{\sigma_k}. \quad (5.11)$$

Finally, the direct probability for the data was assigned using a Gaussian likelihood function. This Gaussian had a standard deviation given by  $\sigma_k$  that is specific to each data set.

The exponential model equation, Eq. (5.1) is symmetric under relabeling of the amplitudes and decay rate constants. This symmetry causes the joint posterior probability for the decay rate constants to be symmetric in the sense that if there is a peak at  $\alpha_1 = \beta$  and  $\alpha_2 = \gamma$ , then there is also a peak at  $\alpha_1 = \gamma$  and  $\alpha_2 = \beta$ ; this symmetry is caused because the model does not tell us which exponential signal corresponds to to which model component. Consequently, a convention must be introduced which brakes this symmetry. In the calculations implemented here, we break this symmetry by ordering the rate constants:  $\{\alpha_1 < \alpha_2 < \alpha_3, \text{etc.}\}$ .

The full Bayesian calculation and the assignment of the prior probabilities is discussed in reference [15] and this paper is available in pdf by activating [this link](#). Additionally, much more about exponential parameter estimation is contained in [16, 17]. The [first](#) paper describes the problem of determining the number of exponentials in a given sample of data, while the [second](#) paper discusses how the accuracy of the parameter estimates depends on the number of data values, signal-to-noise level and the rate of decay of the sample.

## 5.2 Outputs From The Given Exponential Package

The Text outputs files from the exponential packages consist of: “Bayes.prob.model,” “BayesExp-Given.mcmc.values,” “Bayes.params,” “Console.log,” “Bayes.accepted” and a “Bayes.Condensed.File.” These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the mcmc.values report is discussed in Appendix D and the other reports are discussed in Chapter 3. Additionally, the “Plot Results Viewer” can be used to view the output probability density functions. In addition to the standard data, model and residual plots there are probability density functions for the decay rate constants, decay times, the amplitudes for each data set for each exponential and finally there are probability density functions for the standard deviation of the noise in each data set.



## Chapter 6

# Unknown Number of Exponentials

The unknown number of exponentials package estimates amplitudes and decay rate constants of an unknown number of exponential signal components in data that are known to contain sums of exponential signals. The data analyzed by this package are Ascii data files and may be input as Ascii files, peak picks, the amplitudes from a Bayes Analyze analysis run or extracted from an image stack. The unknown number of exponentials package is accessed by selecting the “Exponential” package on the package menu. When the Exponential package is selected, the interface shown in Fig. 6.1 is displayed. This interface is used in both the given and unknown number of exponentials packages. The unknown number of exponential package is selected by setting the model order to “Unknown”. Note when the model order is set to unknown, the “Constant” widget is grayed out, i.e., becomes inactive because the unknown number of exponentials package computes, among other things, the posterior probability a constant is present. To use the unknown number of exponentials package, you must do the following:

**Select** the exponential package from the Package menu.

**Load** one or more Ascii data sets using the Files menu.

**Set** the “Unknown” model order in the model order selection menu.

**Check** the Find Outliers check box if you suspect outliers are present in the data.

**Review** the prior probabilities assignment for the decay rate constants using the Prior Viewer.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

Figure 6.1: The Unknown Exponential Interface

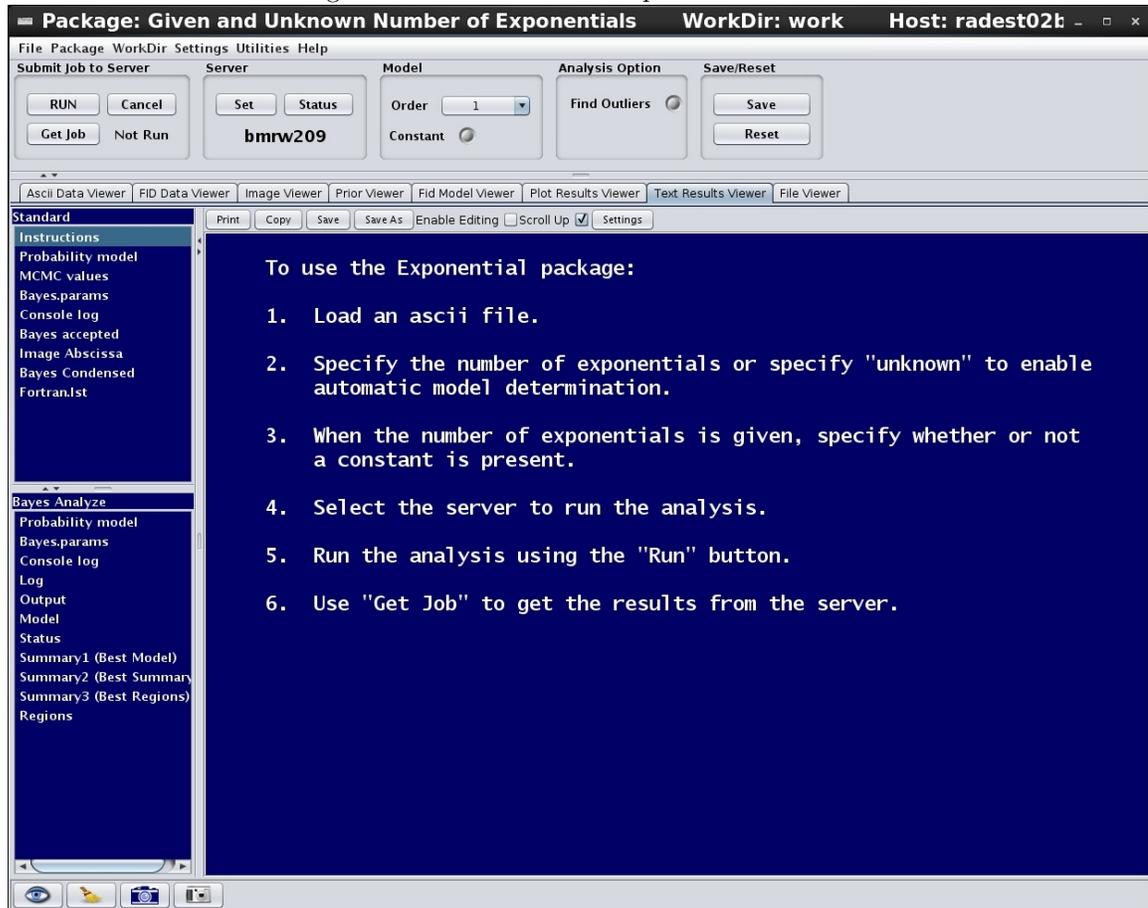


Figure 6.1: When the Exponential package is selected, the Exponentials interface is displayed. The unknown number of exponentials package is selected by setting the model order to “Unknown”. The prior viewer can be used to set the prior probability for the decay rate constants. Note in this package the posterior probabilities are marginal posterior probabilities, so there are no adjustable prior probabilities for the amplitudes.

## 6.1 The Bayesian Calculations

The Bayesian calculation is for the joint posterior probability for the number of exponentials and whether or not a constant offset is present in the data. As with all Bayesian calculations, one begins by relating the data to the parameters of interest. In this case the model which relates the unknown number of exponentials to the data is of the form

$$d_{ij} = C_j + \sum_{l=1}^m A_{lj} \exp\{-\alpha_l t_i\} + n_{ij} \quad (6.1)$$

where  $d_{ij}$  is the  $i$ th data value in the  $j$ th data set containing  $N_j$  data values. Note that the number of data values is data set dependent and may differ from one data set to the next,  $C_j$  is a constant offset in the  $j$ th data set and part of the calculation is to determine if this constant is present,  $m$  is the unknown number of exponentials,  $A_{lj}$  is the amplitude or intensity of the  $l$ th exponential in the  $j$ th data set,  $\alpha_l$  is the  $l$ th exponential decay rate constant, and  $n_{ij}$  is noise in the  $i$ th data value in the  $j$ th data set. Except for the lack of prior information about the number of exponentials  $m$ , and whether or not the constant  $C$  is present, this is the given number of exponentials model, Eq. (5.1). However, now the Bayesian calculations are more substantial because both  $m$  and whether or not  $C_j$  are present must be part of the Bayesian calculation.

The calculation begins by first ranked the models from the simplest to the most complicated and then numbering them from 1 to the total number of models. In the program that implements this calculation the maximum number of exponentials is set to 4, consequently there are a total of 10 models. The model number is designated as  $u$  and when  $u = 1$ , the model is no exponential and no constant,  $u = 2$  is no exponential and a constant,  $u = 3$  is one exponential and no constant, etc., up to model 10, 4 exponentials and a constant. The model indicator,  $u$ , is a discrete hypotheses that must be incorporated into the Bayesian calculation and its numerical value indicates both the number of exponentials and whether or not a constant offset is present.

To determine the model, i.e., estimates  $u$ , one computes the posterior probability for the model indicator given the data and the prior information,  $P(u|DI)$ . This posterior probability is computed by an application of Bayes' theorem [1],

$$P(u|DI) = \frac{P(u|I)P(D|uI)}{P(D|I)} \quad (6.2)$$

where  $P(u|I)$  is the prior probability for the model indicator,  $P(D|uI)$  is the marginal direct probability for the data given the model indicator, and the denominator is

$$P(D|I) = \sum_u P(Du|I) = \sum_u P(u|I)P(D|uI) \quad (6.3)$$

is the normalization constant needed to ensure the total probability for the model indicator sums to one. If this posterior probability is normalized at the end of the calculation, then

$$P(u|DI) \propto P(u|I)P(D|uI). \quad (6.4)$$

The two terms on the right-hand side of this equation are the prior probability for the model indicator,  $P(u|I)$ , which can be assigned using a uniform prior probability, and the direct probability for all of the data given the model indicator,  $P(D|uI)$ , which is much too complicated to see how it

should be assigned. However, this direct probability can be computed from the joint probability for all of the data and the parameters given the model indicator,  $P(D\{A\}\{\alpha\}\{C\}\{\sigma\}|uI)$ , by application of the sum rule:

$$P(u|DI) \propto P(u|I) \int d\{A\}d\{C\}d\{\alpha\}d\{\sigma\}P(D\{A\}\{C\}\{\alpha\}\{\sigma\}|uI) \quad (6.5)$$

where all of the parameters except the model indicator have been removed by marginalization. The notation  $\{\cdot\}$ , means all of the parameters of the indicated type. So for example  $\{A\}$  means all of the amplitudes appearing in the model. If there are 3 exponentials and 5 data sets there would be 15 total amplitudes. Also note that the parameters represented by  $\{C\}$  may or may not occur in any given model depending on the model indicator. So if the model indicator is even then  $\{C\}$  is present, otherwise its not. The right-hand side of this equation may be factored, one obtains

$$P(u|DI) \propto P(u|I) \int d\{A\}d\{\alpha\}d\{C\}d\{\sigma\}P(\{A\}\{\alpha\}\{C\}\{\sigma\}|uI)P(D|u\{A\}\{\alpha\}\{C\}\{\sigma\}I) \quad (6.6)$$

where  $P(\{A\}\{\alpha\}\{C\}\{\sigma\}|uI)$  is the joint prior probability for all of the parameters appearing in model  $u$  and  $P(D|u\{A\}\{\alpha\}\{C\}\{\sigma\}I)$  is the direct probability for all of the data given all of the parameters and the model indicator. Both of these probabilities are too complicated to see how they might be assigned, but the rules of probability theory can be used to factor these probabilities into increasingly simple terms: Assuming logical independence, i.e., changing the prior information about the amplitudes would not change the prior information about the decay rate constants, standard deviations, etc., then the joint prior probability for all of the parameters,  $P(\{A\}\{\alpha\}\{C\}\{\sigma\}|I)$  can be factored into a series is into a series of independent prior probabilities for each parameter:

$$P(\{A\}\{\alpha\}\{C\}\{\sigma\}|I) = \left[ \prod_{j=1}^m \prod_{k=1}^n P(A_{kj}|I) \right] \left[ \prod_{j=1}^m P(\alpha_j|I) \right] \left[ \prod_{k=1}^n P(C_k|I) \right] \left[ \prod_{k=1}^n P(\sigma_k|I) \right] \quad (6.7)$$

where  $m$  is the number of exponentials in the model indicated by  $u$ ,  $n$  is the number of data sets,  $P(A_{kj}|I)$  is the prior probability for amplitude  $k$  in data set  $j$ ,  $P(\alpha_j|I)$  is the prior probability for the  $j$ th decay rate constant,  $P(C_k|I)$  is the prior probability for the constant offset in the  $k$ th data set, and  $P(\sigma_k|I)$  is the prior probability for the standard deviation of the noise in the  $k$ th data set. Note, that the prior probability for the constant offset,  $P(C_k|I)$ , is only present when the model indicator is even. Also note that the dependence on the model indicator has been dropped in these priors, because knowing the number of exponentials and whether or not a constant is present would not change the prior information about the amplitudes and decay rate constants. Substituting the

prior, Eq. (6.7) back into the posterior, Eq. (6.6) to obtain

$$\begin{aligned}
 P(u|DI) &\propto P(u|I) \int d\{A\}d\{\alpha\}d\{C\}d\{\sigma\} \\
 &\times \left[ \prod_{j=1}^m \prod_{k=1}^n P(A_{kj}|I) \right] \\
 &\times \left[ \prod_{j=1}^m P(\alpha_j|I) \right] \\
 &\times \left[ \prod_{k=1}^n P(C_k|I) \right] \\
 &\times \left[ \prod_{k=1}^n P(\sigma_k|I) \right] \\
 &\times \left[ \prod_{k=1}^n P(D_k|uA_{1k} \dots A_{mk}\{\alpha\}C_k\sigma_k I) \right]
 \end{aligned} \tag{6.8}$$

as the posterior probability for the model indicator, where  $P(D_k|uA_{1k} \dots A_{mk}\{\alpha\}C_k\sigma_k I)$  is the direct probability for data set  $D_k$  given both the parameters that occur in all data sets, the  $\{\alpha\}$ , and the parameters specific to data set  $D_k$ . These data set specific parameters are the constant offset  $C_k$  and the amplitudes of the  $m$  exponentials in the  $k$ th data set. Here these amplitudes have been designated as  $A_{1k} \dots A_{mk}$ .

If you examine the integrand of this posterior, you will discover that it is the joint posterior probability for the parameters given the model indicator. Consequently, this integrand is exactly the same as the integrand in the joint posterior probability given the model order, Eq. (5.3), and in what follows the same prior probabilities will be assigned in both problems. This will have the effect of making the parameter estimates from the unknown number of exponentials and the given number of exponentials essentially identical when the model indicators are the same.

Equation. (6.8) cannot be further simplified and the only recourse is to assign the various prior probabilities and evaluate the integrals. The prior probability for the model indicator,  $P(u|I)$ , was assigned using a uniform prior probability:

$$P(u|I) \propto \begin{cases} \frac{1}{10} & \text{if } 1 \leq u \leq 10 \\ 0 & \text{otherwise} \end{cases} . \tag{6.9}$$

The prior probability for the noise standard deviation,  $P(\sigma_k|I)$ , was assigned a Jeffreys' prior

$$P(\sigma_k|I) \propto \frac{1}{\sigma_k}. \tag{6.10}$$

Note that the Jeffreys' prior has not been bounded and normalized because the exact same normalization constant appears in all models tested by this calculation. The prior probabilities for the

amplitudes,  $P(A_{jk}|I)$ , are assigned broad Gaussian prior probability:

$$P(A_{jk}|I) = \left(\frac{2\pi\sigma_k^2}{\delta^2}\right)^{-\frac{1}{2}} \exp\left\{-\frac{\delta^2}{2\sigma^2}A_{jk}^2\right\} \quad (6.11)$$

where  $\delta/\sigma_k$  plays the part of the standard deviation of this prior probability and cannot be changed by the user. In the program that implements this calculation  $\delta = 0.01$ .

The prior probabilities for the decay rate constants are under user control and the user may select the type of prior used. However, if the user does not set the prior probability for the decay rate constants, then the same default prior probability described in Chapter 5 Eq. (5.4) is used.

The model equation is symmetric under relabeling of the amplitudes and decay rate constants. This symmetry causes the joint posterior probability for the decay rate constants to be symmetric in the sense that if there is a peak at  $\alpha_1 = \beta$  and  $\alpha_2 = \gamma$ , then there is also a peak at  $\alpha_1 = \gamma$  and  $\alpha_2 = \beta$ ; this symmetry is caused because the model does not tell us which exponential signal corresponds to to which model component. Consequently, a convention must be introduced which breaks this symmetry by identifying a model component with a signal component. This is done by forcing the decay rate constants to be ordered.

The full Bayesian calculation and the assignment of the prior probabilities is discussed in reference [16] and this paper is available in pdf by activating [this link](#). Additionally, much more about exponential parameter estimation is contained in [15, 17]. The [first](#) paper describes the problem when the exponential model is given and is a simpler version of this problem, while the [second](#) paper discusses how the accuracy of the parameter estimates depends on the number of data values, signal-to-noise level and the rate of decay of the sample.

## 6.2 Outputs From The Unknown Number of Exponentials Package

The Text outputs from the unknown number of exponentials package consist of: “Bayes.prob.model,” “BayesExpGiven.mcmc.values,” “Bayes.params,” “Console.log,” “Bayes.accepted” and a condensed output file “Bayes.Condensed.File.” These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the mcmc.values report is discussed in Appendix D and the other reports are discussed in Chapter 3.

When the unknown number of exponentials package is running, the distribution of simulations having a given model indicator  $u$  is in constant flux. The unknown number of exponentials program writes the number of simulations having  $u = 1$ ,  $u = 2$ , etc. to the console log after it completes each annealing step. This display serves as a kind of visual picture of the calculation of the posterior probability for the model indicator. A sample of this output is shown in Fig. 6.2. This listing was generated from a data set that contains three exponential plus a constant. When the annealing parameter is zero, the distribution of simulations reflects only the prior probability for the model. Because that prior probability is uniform, the simulations are essentially evenly distributed across all models. As the program runs, the annealing parameter is increased and the distribution of simulations begins to change to reflect the increasing importance of the likelihood. Typically the simulations begin to cluster around one or two models. This clustering continues until of the simulations have settled into the posterior probability for the model indicator.

Figure 6.2: The Distribution Of Models

Phase	Annl Param	<Likelihood>	<StdDevLike>	0	0+C	1	1+C	2	2+C	3	3+C	4	4+C	
Annealing	0.000	-7.2157E+02	1.5195E+02	12	6	7	8	12	13	9	11	10	12	
	2	0.005	-5.6842E+02	1.6292E+02	7	5	4	4	2	21	4	24	3	26
	3	0.011	-4.8783E+02	1.1564E+02	1	0	0	2	0	14	13	28	16	26
	4	0.018	-4.3429E+02	8.3983E+01	0	0	1	0	0	18	2	29	17	33
	5	0.026	-3.9971E+02	5.2703E+01	0	0	0	0	0	9	4	26	21	40
	6	0.038	-3.8265E+02	3.4063E+01	0	0	0	0	0	8	2	37	21	32
	7	0.054	-3.7660E+02	3.6781E+01	0	0	0	0	0	7	2	34	24	33
	8	0.069	-3.6507E+02	1.7327E+01	0	0	0	0	0	4	0	39	24	33
	9	0.090	-3.6031E+02	1.3435E+01	0	0	0	0	0	1	0	44	24	31
	10	0.113	-3.5629E+02	8.9340E+00	0	0	0	0	0	1	0	54	21	24
	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Phase	Annl Param	<Likelihood>	<StdDevLike>	0	0+C	1	1+C	2	2+C	3	3+C	4	4+C	
Annealing	0.745	-3.4957E+02	1.5147E+00	0	0	0	0	0	0	0	97	2	1	
	32	0.777	-3.4935E+02	1.2221E+00	0	0	0	0	0	0	94	1	5	
	33	0.809	-3.4936E+02	1.1754E+00	0	0	0	0	0	0	99	0	1	
	34	0.841	-3.4917E+02	9.8822E-01	0	0	0	0	0	0	94	1	5	
	35	0.873	-3.4893E+02	8.6573E-01	0	0	0	0	0	0	99	1	0	
	36	0.906	-3.4926E+02	1.3015E+00	0	0	0	0	0	0	97	1	2	
	37	0.938	-3.4910E+02	9.9978E-01	0	0	0	0	0	0	97	0	3	
	38	0.970	-3.4902E+02	9.6721E-01	0	0	0	0	0	0	97	2	1	
	39	1.000	-3.4903E+02	9.9140E-01	0	0	0	0	0	0	100	0	0	

Figure 6.2: While the unknown number of exponentials package is running, it prints a listing that shows the distribution of the model indicator as a function of the annealing parameter. The 10 columns to the right are the number of simulations in model 0, 1, etc. As the annealing parameter increases these should cluster into one or two columns and the distribution of these simulations is the posterior probability for the model order. In this example the data were three exponentials plus a constant. Notice that as soon as the annealing parameter begins to increase the simulations quickly move to high order models and eventually they all end up in the three exponential plus a constant model.

In addition to these text reports there is one new plot result, the posterior probability for the model order, Fig. 6.3. The abscissa is labeled with the model order. These model orders, 1, 2, 3, etc. are the number of exponentials in the model. When a constant is present this, the model order is changed to 1+C, 2+C, 3+C etc. to indicate the presence of one, two or three exponentials plus a constant. The vertical axis, is the posterior probability for the model indicator. Each probability is computed by counting the number of simulations having the indicated model divided by the total number of simulations. The “Plot Results Viewer” can be used to view the probability for the model indicator and the output probability density functions for the parameters in a given model order. In addition to the standard data, model and residual plots there are probability density functions for the decay rate constants, decay times, the amplitudes for each data set for each exponential and finally there are probability density functions for the standard deviation of the noise in each data set.

Figure 6.3: The Posterior Probability For Exponential Model

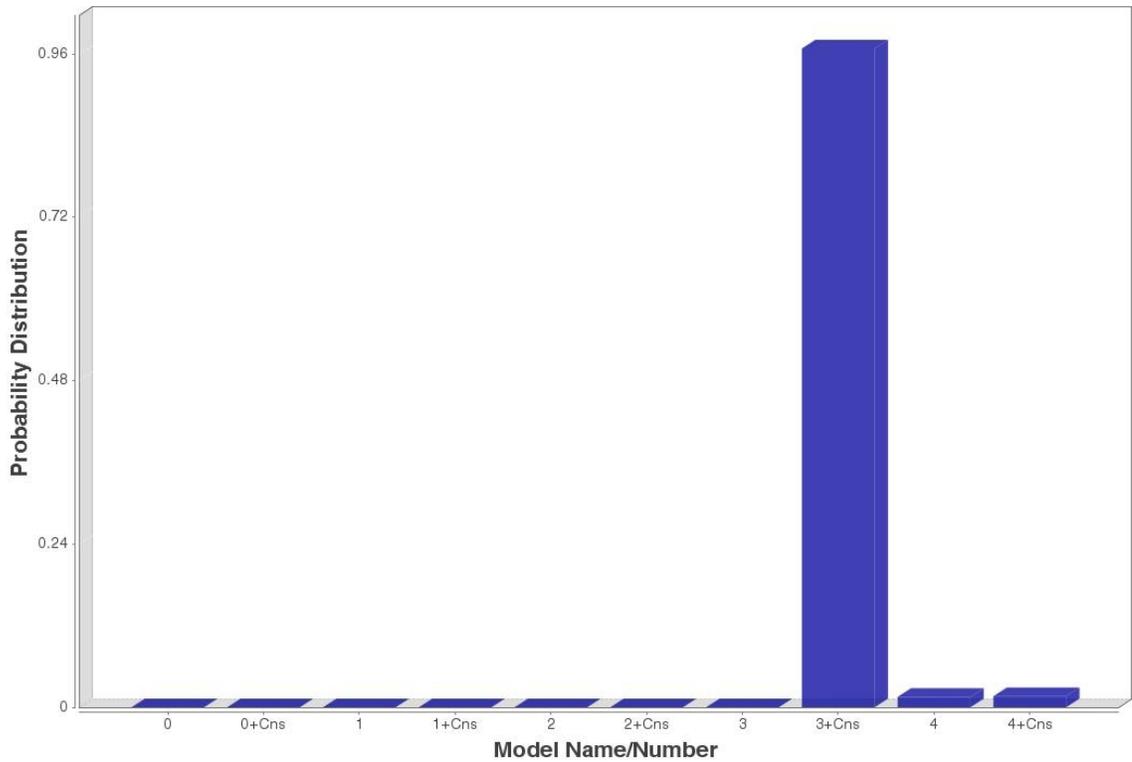


Figure 6.3: When the Unknown number of Exponentials package runs a new output plot is produced. The plot is of the posterior probability for the model order. The abscissa is labeled with the model order. These model orders are arbitrated as 1, 2, 3, etc. when the model is one, two or three exponentials without a constant. When a constant is present this label is changed to 1+C, 2+C, 3+C etc. to indicate the presence of one, two or three exponentials plus a constant.

## Chapter 7

# Inversion Recovery

The inversion recovery package is an example of a preloaded package. Preloaded packages are special cases of the Enter Ascii package, where the model is loaded at the time the package is started. In this example, the model is a single exponential plus a constant with the two amplitudes expressed as an initial and final amplitude. The input data files analyzed by this package are Ascii and may be input from Ascii files, a peak pick, a Bayes Analyze file or they may be loaded from an image pixel. When “Inversion Recovery” package button is activated, the interface window shown in Fig. 7.1 is displayed. To use this package, you must do the following:

**Select** the inversion recovery package from the package menu.

**Load** one or more Ascii data sets using the Files menu.

**Check** the find outliers box if you suspect outliers are present in the data.

**Review** the prior probability for the decay rate constant to make sure the prior assignment is correct.

**Select** the name of the server that is to process the request.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run.

The inversion recovery package has only one package specific button, the Find Outliers button and it is should only be used when you think there might be an outlier in the data. Additionally, the prior viewer can be used to setup the prior probability for the decay rate constant.

Figure 7.1: The Inversion Recovery Interface

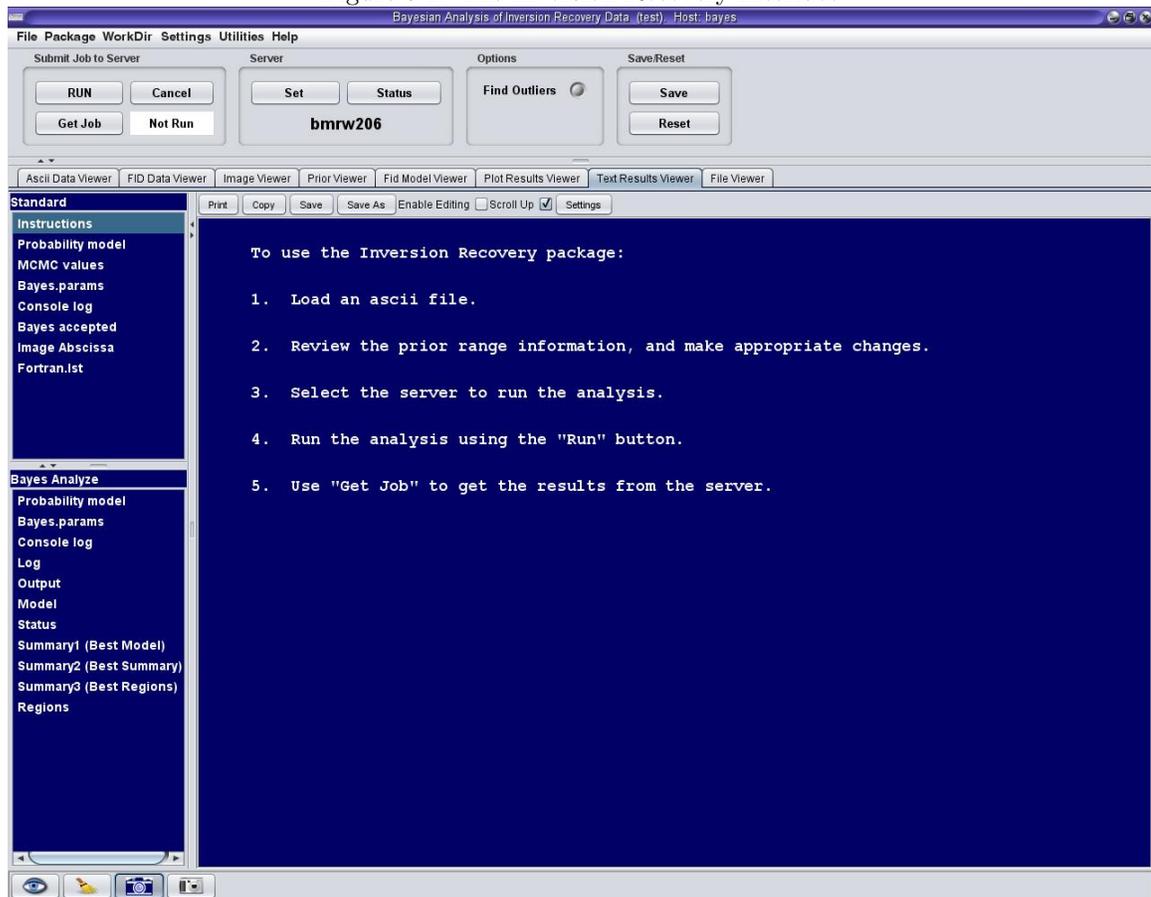


Figure 7.1: When the Inversion Recovery package is selected the interface shown here is displayed. The prior viewer can be used to set the prior probability for the decay rate constants. Note in this package the posterior probabilities are marginal posterior probabilities, so there is no prior for the amplitudes.

## 7.1 The Bayesian Calculation

The inversion recovery model is a single exponential model when the amplitudes are expressed as an initial amplitude in the  $j$ th data set,  $M_{0j}$  and a final amplitude  $M_{\infty j}$  in the  $j$ th data set:

$$d_{ij} = M_{\infty j} + (M_{0j} - M_{\infty j}) \exp\{-\alpha t_i\} + n_{ij} \quad (7.1)$$

where  $\alpha$  is the decay rate constant of the magnetization. It is easy to verify that  $M_{0j}$ , is the magnetization at  $t = 0$  by simply substituting  $t = 0$ , the model reduces to  $M_{0j}$  and one is estimating the initial magnetization. When  $t = \infty$ , the exponential goes to zero and one is estimating the ending magnetization,  $M_{\infty j}$  and the magnetization relaxes from  $M_{0j}$  to  $M_{\infty j}$ . Because of the way this model is written,  $M_{0j}$  is not forced to be negative,  $M_{0j}$  simply represents the beginning magnetization, while  $M_{\infty j}$  represents the ending magnetization. This package uses a marginal posterior probability for the decay rate constant, and consequently the integrals over the amplitudes are from minus to plus infinity.

The calculations are very standard, and proceeds by computing the marginal posterior probability for each parameter from the joint posterior probability for all of the parameters. So, for example, the posterior probability for the decay rate constant is given by

$$P(\alpha|DI) = \int P(\alpha\{M_0\}\{M_\infty\}\{\sigma\}|DI)d\{M_0\}d\{M_\infty\}d\{\sigma\} \quad (7.2)$$

where  $\{M_0\}$  is all of the initial amplitudes given the  $n$  input data sets, similarly  $\{M_\infty\}$  are all of the final amplitudes and  $\{\sigma\}$  are the standard deviations of the noise prior probabilities for the data sets. We assume that there is a single initial and final amplitude associated with each data set and that the data sets are independent, consequently, this equation can be factored using Bayes' Theorem [1] and the rules of probability theory

$$P(\alpha|DI) = \prod_{j=1}^n \int P(\alpha M_{0j} M_{\infty j} \sigma_j | D_j I) dM_{0j} dM_{\infty j} d\sigma_j \quad (7.3)$$

and factoring the joint posterior probability for all of the parameters one obtains:

$$P(\alpha|DI) = P(\alpha|I) \prod_{j=1}^n \int P(M_{0j}|I) P(M_{\infty j}|I) P(\sigma_j|I) P(D_j|\alpha M_{0j} M_{\infty j} \sigma_j I) dM_{0j} dM_{\infty j} d\sigma_j. \quad (7.4)$$

The prior probabilities for the amplitudes,  $P(M_{0j}|I)$  and  $P(M_{\infty j}|I)$  are assigned using broad Gaussian prior probability and to facilitate marginalization, these priors cannot be changed by the user. The prior probability for the standard deviations,  $P(\sigma_j|I)$ , are assigned using a Jeffreys' prior, [33]. The prior probabilities for the decay rate constant,  $P(\alpha|I)$ , can be set by the user. The integrals over both the magnetizations and the standard deviations are evaluated analytically using the techniques described in [2].

The posterior probability for the decay rate constant computed using this inversion recovery model is a marginal posterior probability. Consequently, only the decay rate constant occurs in the posterior probability. However, because the initial and final magnetizations were removed using marginalization, and the corresponding marginalization integrals were from minus to plus infinity, it is possible for this package to estimate an amplitude to be either positive or negative. If this is

in contradiction to the information available to the user, then using the Enter Ascii Model package with an inversion recovery model without marginalization will enable the user to impose any prior ranges needed on the initial and final magnetizations.

The full Bayesian calculation and the assignment of the prior probabilities is discussed in reference [15] and this paper is available in pdf by activating [this link](#). Additionally, much more about exponential parameter estimation is contained in [16, 17]. The [first](#) paper describes the problem of determining the number of exponentials in a given sample of data, while the [second](#) paper discusses how the accuracy of the parameter estimates depends on the number of data values, signal-to-noise level and the rate of decay of the sample.

## 7.2 Outputs From The Inversion Recovery Package

The Text outputs reports from the Inversion Recovery packages are very simple, consisting of a probability for the model file “Bayes.prob.model.” A file containing the results of the Markov chain Monte Carlo simulations “BayesEnterAscii.mcmc.Values” is written. This file contains the mean and standard deviation parameter estimates. A parameter file “Bayes.params” is generated. An output console log “Console.log” and an accepted report “Bayes.accepted,” are both written while the package is running. Finally a condensed file, “Bayes.Condensed.File,” is written that contains the parameter estimates in a condensed form that is more usable by a computer program. The format of the mcmc.values report is discussed in Appendix D and the other reports are discussed in Chapter 3. The “Plot Results Viewer” can be used to view the output probability density functions. In addition to the standard data, model and residual plots there are probability density functions for the decay rate constants, decay times, the initial and final magnetizations for each data set and finally there are probability density functions for the noise standard deviation in each data set.

## Chapter 8

# Bayes Analyze

The Bayes Analyze package analyzes one dimensional and one dimensional arrayed free inductions decays (Fid) using Bayesian probability theory [1]. The package is an implementation of the procedures described in [4, 5, 6, 2, 8]. These articles and books describe in detail the types of calculations that must be done using probability theory to detect the presence of resonances, estimate the associated parameters, and determine the number of resonances. For a tutorial on parameter estimation see Chapter 4 or reference [3]. For a tutorial on model selection see reference [11] and for a full exposition of the Bayesian Probability theory see references [33, 32]. Last, for a comparison of Bayesian and Fourier techniques for frequency estimation see reference [35, 47].

The interface to the Bayes Analyze package is shown in Fig. 8.1. The package specific widgets in the top portion of this interface have been shifted to the right. Consequently, the submit job and the server widgets are not shown. To use the Bayes Analyze package, you must do the following:

**Select** the Bayes Analyze package from the Package menu.

**Load** a spectroscopic Fid using the Files menu.

**Check** the various package specific widgets to make sure the default settings are correct. When the Bayes Analysis interface starts, the widgets on the interface are initialized using a set of defaults. These defaults automatically indicate that there is a first point problem, that a maximum of 10 resonances can be found on each run, that a correlated phase model is assumed, that all Fids should be analyzed jointly, that the entire Fid should be analyzed and finally that only singlets should be estimated.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

Figure 8.1: Bayes Analyze Interface

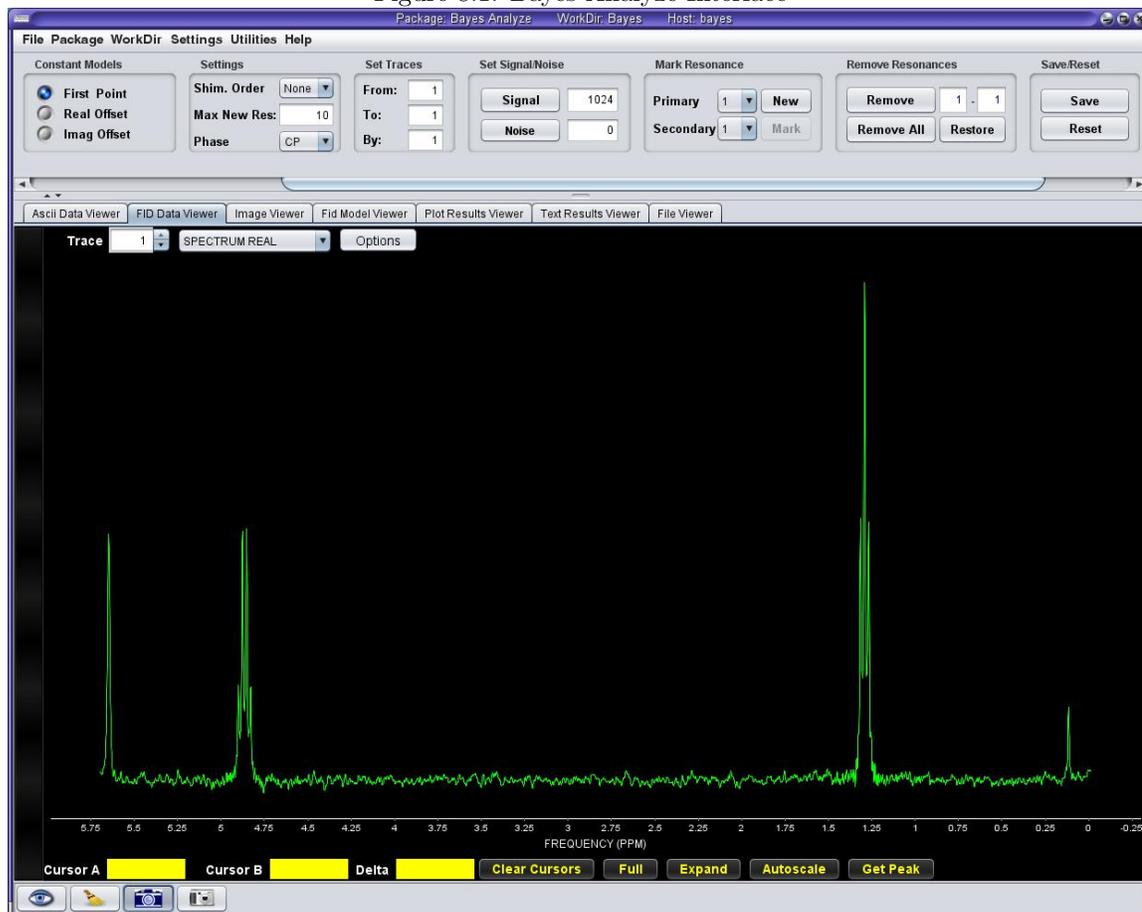


Figure 8.1: This window is used to control the execution of Bayes Analyze. On this window you may load Fids using the Files Menu, set all necessary Bayes Analyze parameters, mark multiplets, run the analysis, etc. Finally, after running the analysis, by switching to the Fid Model Viewer you can generate and view models of the loaded Fid.

Unlike, some of the other packages, Bayes Analyze has a number of package specific widgets. The reason for this is simply that Bayes Analyze is a searching routine and does not make use of Markov chain Monte Carlo. Additionally, Bayes Analyze has a number of different ways that it can model data and defining these models increases the complexity of the interface. Here we give a brief descriptions of the package specific widgets. We will divide this description into sections one section for each grouping of buttons in the interface. There are seven such groupings: Constant Models, Settings, Set Traces, Set Signal/Noise, Mark Resonances, Remove Resonances and Save/Reset. We begin by describing the constant models:

**Constant Models** NMR FID data often contain constant offsets. This is especially true when the time domain FID data are not acquired using phase cycling. Regardless of the cause of the constants, in order to analyze the FID data these offsets must be accounted for in the model. Here is a brief description of the offsets implemented in Bayes Analyze:

**First Point** is a model of the first complex data value in the Fid. This model is included by default, because the first point in an Fid is often misplaced and on some spectrometers, intentionally misplaced. The first point model allows Bayes Analyze to take into account this possibly misplaced data value.

**Real Offset** indicates that there is a constant offset in the real channel. The default is off, and checking this box causes the real offset model to be included.

**Imag Offset** indicates that there is a constant offset in the imaginary channel. The default is off, and checking this box causes the imaginary offset model to be included.

**Settings** allow the users to set some optional features of the model and the search:

**Shim. Order** allows the user to specify the shimming model used in Bayes Analyze. Loosely, the shimming model expands the line-shape of each resonance in a super position of Lorentzian's. The shimming order allows the user to set the number of Lorentzian's used. The number of Lorentzian's is forced to be odd and the valid entries are none, 1, 3, 5 and 7, where 1 is no shimming model, i.e., the lineshape is pure Lorentzian, 3 allows for lineshapes that have a single bulge on one or both sides of the line, and 5 and 7 allows for very badly shimmed lines.

**Max New Res** is the maximum number of resonances that can be added to the resonance model the next time Bayes Analyze is run. The default value is 10, but any number greater than zero can be entered.

**Phase** allows the user to specify the phase of the resonance model. There are two choices, one in which the phase of each resonance is linearly related to the phase of all other resonances. This phase model is called "CP" for correlated phase. In the correlated phase model, there are two phase parameters, a zero and first order phase. The other phase model is called "UP" for uncorrelated phase. In the uncorrelated phase model, each resonance has its own phase parameter. When marking resonances, the "Phase" widgets indicates the phase model of the marked resonances, i.e., when the phase model is uncorrelated a marked resonance has its own phase. Similarly, when the phase model is correlated phase, marked resonances have shared zero and first order phase parameters. Both correlated and uncorrelated phase resonance can be marked in the same analysis.

**Set Traces** is an area consisting of three buttons that allows the user to specify how the arrayed Fid data is to be processed.

**From** tells Bayes Analyze to start analyzing the Fid data using Fid number “From”. For example, if there are 10 Fids and you set “From” to 3, then the first Fid to be process is number 3.

**To** tells Bayes Analyze that the last Fid number to analyze is “To”. If “To” is less than the total number of Fid data, then Fids numbers greater than “To” are not analyzed. Continuing the example started in the previous item, if “To” is set to 7, then Fido’s 8 through 10 would not be processed; but Fido’s 3, 4, 5, 6 and 7 would be.

**By** tells Bayes Analyze to process the Fids in blocks of Fids of size “By”. If “By” is not an integer divisor of the total number of traces in the Fid, then the last block of traces processed by Bayes Analyze will have fewer than “By” Fids. So if “From” is 3, “To” is 7 and if “By” is 2, then Bayes Analyze will process Fido’s 3, and 4 together in the first analysis. These outputs files will have a 0003 suffix. It will then process 5 and 6 together and these outputs will be written into output files having a suffix of 0005. Finally, Fid 7 will be processed separately and the outputs from this analysis will have output files suffixed with 0007.

**Set Signal/Noise** is an area used to define both signal and noise regions in the Fid. A signal region is used by Bayes Analyze to speed up the analysis by essentially telling Bayes Analyze to only analyze the signal region. A Noise region is used by the interface to estimate the noise standard deviation in the data and there by increase Bayes Analyze’s sensitive to small resonances.

**Signal** tells Bayes Analyze to analyze the Fids using the data from the first point up to the data value entered in the “Signal” box. You can set this value by simply typing in the text entry box or you can place a single cursor on the Fid data and then activate the Signal button. Activating the signal button will set the text field to the number of the current data value.

**Noise** is used to set a noise region. The noise region is defined to that part of the Fid starting with “Noise” up to the end of the Fid. This region is used by the interface to calculate the estimated standard deviation of the noise. These values are written in to a file named bayes.noise located in Bayes/WorkDir/BayesAnalyzeFiles/bayes.noise where WorkDir must be replaced by the name of your current WorkDir. Bayes Analyze uses these noise standard deviations to assign a Gaussian likelihood rather than a Student’s  $t$ -distribution. You can set the noise region by simply typing in the noise entry box, or you can position a single cursor at the location you wish the noise region to start. Activating the noise button will set the entry box.

**Mark Resonance** is an area used to mark resonances. These resonances may be singlets, spin one half multiplets, or spin one half multiplets of multiplets.

**Primary** allows you to set the order of the primary multiplet. The primary multiplet is the multiplet having the largest  $J$ -coupling constant. The current value of the primary spinner is use to set the primary multiplet order.

**Secondary** allows you to set the order of a secondary multiplet. A secondary multiplet is the multiplet, in a multiplet of multiplets, that has the smallest  $J$ -coupling constant. For example, a triplet of doublets might have a primary coupling constant of 10 Hertz, and the secondary coupling constant could be from 1 to something less than 10 Hertz.

**New** tells the interface you are about to mark resonance. Based on the Primary and Secondary settings, the interface will display a message that indicates what must be marked and in what order. For example, if the primary coupling is 2, the secondary is 3 and you hit the “New” button:

Mark center. Mark primary J-coupling. Mark secondary J-coupling.

will be displayed and you must first mark the center of the multiplet. Second, you must mark the primary  $J$ -coupling constant. The primary coupling constant is the largest coupling constant in the multiplet. Third, you must mark the secondary  $J$ -coupling constant. The secondary  $J$ -coupling this coupling constant is the smallest coupling constants in the multiplet. In order for marking a multiple of multiplets to be successful, you must mark all three of these items in the specified order.

**Mark** tells the interface to use the current cursor position(s) for the item currently being marked. When marking coupling constants a double cursor must be used. However, a double or single cursor can be used to mark the center of a multiplet. If a double cursor is used, the multiplet center is assumed to be the average value of the two cursors. This is very useful when marking even numbered multiplets because there is no resonance at the center of things like doublets, and quartets.

**Remove Resonances** is an area used to remove resonances. You can remove individual resonances or you can remove all resonances from the current model.

**Remove** will remove the resonance number displayed in the entry box from the current model.

**Remove All** will remove *all* of the resonances in the current model.

**Restore** will restore a previously delete resonance provided the interface still has the resonance from the current model in memory.

**Save/Reset** is an area used copy an analysis back to the original Fid directory, or it can reset the analysis back to its default settings.

**Save** will copy the Bayes Analyze files from the current analysis back to the Fid directory that is current loaded. If this Fid directory contains Bayes Analyze files they are overwritten by the new analysis.

**Reset** will reset all of the parameters to their default values.

## 8.1 Bayes Model

After Bayes Analyze has been run, the output from the analysis can be used to generate a model of the time domain Fid data. This model is displayed in a special Fid Model Viewer, shown in Fig. 8.2, the bottom three quarters of the figure. To generate a Fid Model one first activated the Fid Model Viewer, and then use the “Build BA Model of FID #” button to build the model. When this button

Figure 8.2: Bayes Analyze Fid Model Viewer

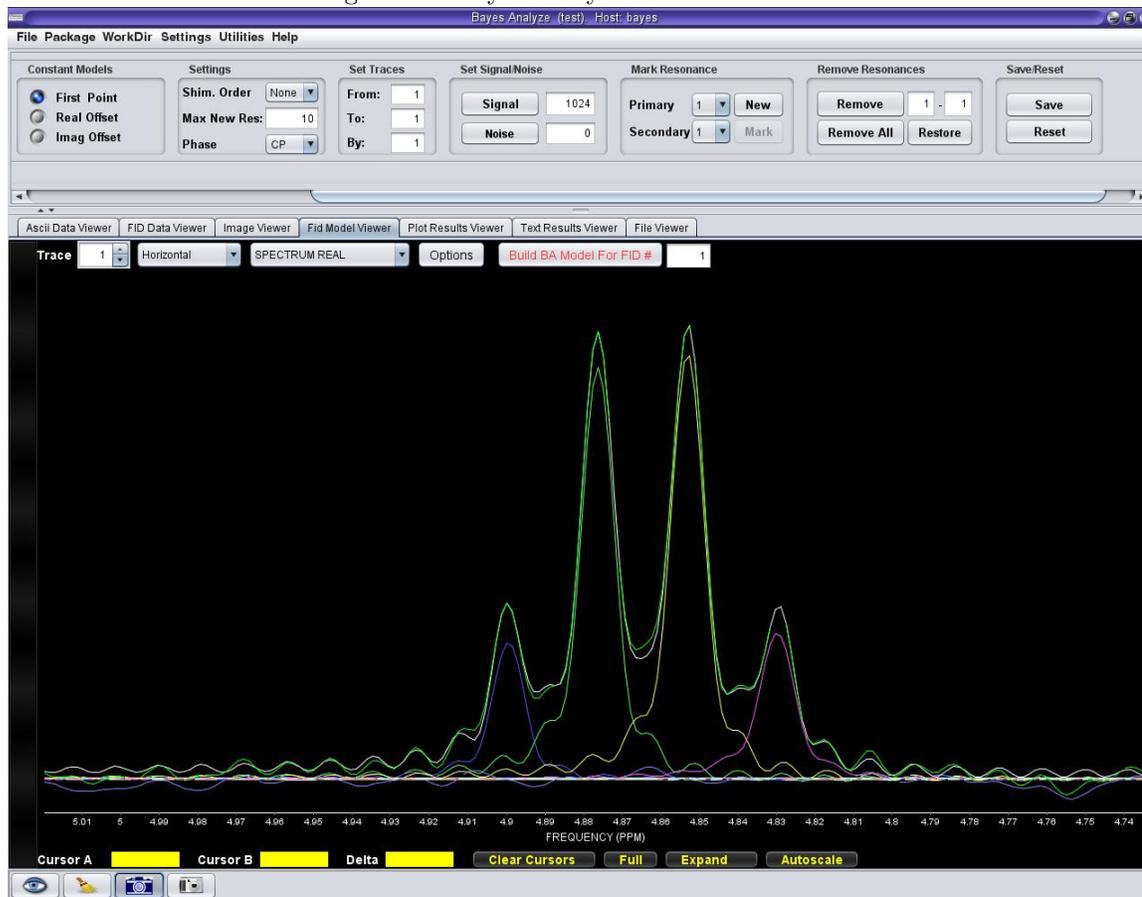


Figure 8.2: After the Bayes Analysis program has been run, the Fid Model Viewer can be used to view the results of the analysis in a graphically. Here we show the region around the quartet. This result was generated from an automatic analysis of this data. In the region shown there are a total of 4 model resonances and we have overlaid the spectrum of the time domain Fid model. This overlaid data matches the spectrum so well that the data and the model are almost indistinguishable. The other four resonances on this data are the four model components in this region. Note that because this is a time domain model, it even models the Gibbs ringing artifacts, sometimes referred to as sinc wiggles.

is activated the appropriate Fid data, and the estimated Fid parameters are sent to the server. The server unpacks the data and runs a job to build the model. During this time, the interface waits for the job to complete. When completed, the job is fetched, the model Fid is Fourier Transformed, and displayed using the Fid Model Viewer. The displayed plot in Fig. 8.2 is called a Horizontal display or plot. Here we shown the spectrum of the Ethyl Ether Fid model in the region of the quartet. This result was generated from an automatic run of Bayes Analyze using this Ethyl Ether data. In the region shown there are a total of 4 model resonances. The Horizontal plot overlays the spectrum of the time domain Fid data and the spectrum of the time domain model. This overlaid spectrum matches the spectrum so well that the data and the model are almost indistinguishable. If you look closely in the valleys between resonances, you can see a few small differences. In addition to displaying the spectrum of the data and the model, the spectrum of the individual resonances in the model are also displayed. These are the other resonances on this plot. Note that because this is a time domain model, it even models the Gibbs ringing artifacts, sometimes referred to as sinc wiggles.

## 8.2 The Bayes Analyze Model Equation

To apply probability theory as logic, one always computes the probability for some hypothesis given what one knows. In NMR one knows a great deal about the types of signals that can appear in the detector. For free induction decays, the signals are sinusoids that decay in time. In NMR if a sample loses half of its magnetization in one second, then in the second second it will lose half of its magnetization. Thus, the decay of the signal will be exponential. In the frequency domain, exponentials decay have a Lorentzian lineshape. In the time domain, the equation that relates an exponentially decaying sinusoidal signal to the data is given by

$$d_{Ri} = \sum_{j=1}^m B_j \cos(\omega_j t_i + \theta_j) e^{-\alpha_j t_i} + n_R(t_i) \quad (8.1)$$

for the real channel, where  $d_{Ri}$  is the  $i$ th data value sampled at time  $t_i$ ,  $\omega_j$  is the resonance frequency for the  $j$ th resonance,  $\alpha_j$  is its associated exponential decay rate constants,  $B_j$  is the resonance amplitude,  $\theta_j$  is the phase of the resonance,  $m$  is the number of resonances, and  $n_R(t_i)$  is the error or noise in the data. The imaginary channel is  $90^\circ$  phase shifted, and so the sinusoids are given by:

$$d_{Ii} = - \sum_{j=1}^m B_j \sin(\omega_j t_i + \theta_j) e^{-\alpha_j t_i} + n_I(t_i) \quad (8.2)$$

where  $n_I(t_i)$  are the errors or noise in the imaginary channel. Note, that the frequency  $\omega_j$  does not have a factor of  $2\pi$  in it and internally the program that implements this package uses dimensionless times: consequently,  $t_i \in \{0, 1, 2, \dots, N-1\}$ , that is to say,  $t_i$  is a simple integer. In these units the frequencies and decay rate constants have units of radians. In this package when the user marks a frequency in Hertz, or PPM, the package converts the frequency into radians. This conversion to Hertz is given by:

$$\omega_j = \frac{(f_j + f_r)2\pi\Delta t}{N} \quad (8.3)$$

where  $\omega_j$  is the internal frequency in radians,  $f_j$  is the frequency in Hertz,  $f_r$  is the reference frequency in Hertz,  $\Delta t$  is the time interval between data values in seconds, and  $N$  is the total

number of complex data values acquired in the free induction decay. Similarly, the decay rates  $\alpha_j$  are also in radians. The conversion of the decay rates to these internal units is given by:

$$\alpha_j = \frac{\beta_j \pi \Delta t}{N} \quad (8.4)$$

where  $\beta_j$  is the full width at half-maximum of the resonance in the absorption spectrum. Note that there are two differences between the frequency and decay rate conversions: we don't reference decay rates and there is no factor of 2.

Equations (8.1 and 8.2) are the conventions used by Varian/Agilent and Siemens spectrometers. Bruker spectrometers use different conventions. The different conventions come about because the data are very high frequency and must be mixed down to a set of difference frequencies that can be properly digitized. In the process of mixing down, the reference sines and cosines may have either positive or negative frequencies, and the two channel could correspond to either a cosine or sine mixing. Consequently, there are four different sets of model equations corresponding to different sign conventions. These different sign conventions can effectively aliases either the positive or negative frequencies in the spectrum depending on which convention is used. Additionally, some NMR spectrometers use only a single digitizer and collect the real and imaginary channels at different times. This time difference is not accounted for in Eqs. (8.1 and 8.2). Varian/Agilent and Siemens use a cosine and minus sine combination as shown in Eqs. (8.1 and 8.2). Bruker, uses a different convention and because of this Bruker spectroscopic data cannot be correctly processed by the Bayes Analyze package. At least at the present time the "File/Load Spectroscopic Fid" menu does not load Bruker spectroscopic data. If one wished to analyze such data, an Ascii Model could be written using the proper sine and cosine conventions and then run using either the Enter Ascii Model package or the Enter Ascii Model Selection package. However, because these packages read Ascii data, the Bruker data would have to be converted to an appropriate Ascii file.

Model Eqs. (8.1 and 8.2) leave out a great deal of what is often known about the NMR signal. For example, these equations ignore constant offsets, baseline artifacts, multiplets, multiplets of multiplets, linear phase errors in the frequency domain, and non-Lorentzian lineshapes. In spite of this, a program that implemented this complex model would still be useful, because this model would allow for the estimation of frequencies, amplitudes, phases, decay rate constants, and the number of resonances in most NMR free induction decays.

The actual model signal used in the Bayes Analyze package is more sophisticated than Eqs.(8.1 and 8.2). For example, the way Eqs. (8.1 and 8.2) were written implies that the phase of the sinusoids is different for each resonance and this is not necessarily true. Indeed the model used by the Bayes Analyze package allows the user to specify whether the phases of the resonances are the same ("correlated") or different ("uncorrelated"). If the resonances are correlated the model includes a common "zero order" phase and a linear frequency-dependent "first order" phase shift. These two parameters are in exact analogue of the left and right phase parameters that VnmrJ uses. Additionally, the model includes spin  $\frac{1}{2}$  "AX" multiplets and multiplets of multiplets. If these modifications were the only additional prior information built into the model, the model would be considerably more sophisticated than Eqs. (8.1 and 8.2) would indicate. However, the actual model in use also takes into account two other types of information.

The first type of additional information is that there may be a constant offset in the data. No amplifier is perfect and, as a result, NMR free induction decays often have constant offsets. This is particularly true if phase cycling is not used. If these offsets are not accounted for in the model, they would introduce artifacts into the analysis. Three different types of constant models are allowed:

a first point model, a constant in the real channel and a constant in the imaginary channel. The first point model is just what its name implies: it is a model of the first time-domain point in the data. In essence this model assumes that the first time-domain data value is not correct. The types of things that could cause the first point to be off are the presence of “probe or filter ringing” or an extremely rapidly decaying sinusoid in the data. Including the first point model is the default and you must uncheck the “Constant Models/First Point” to turn this model off. The second and third constant models are a constant offsets in the real and imaginary channel. These offset models are different from the first point. The first point model is a constant that extends only one point in time, while the offset models extends all the way through the acquisition. The use of these constant models is selected by the user by toggling the “Constant Models/Real Offset.” These three constant models may be used either separately or in any combination.

The second type of additional information that was incorporated into the model equation was that the lineshapes are not pure Lorentzian. One of the problems that had to be faced early in the development of this package was what decay model to use. In the development of the Bayes Analyze package various models of the decay were postulated and probability theory was used to test these models against the Lorentzian model. It was found that no simple models of the physics were better than the Lorentzian model. To give just a few examples of the model tested, we tested Gaussian, Lorentzian plus Gaussian, linear drifts in the  $H_0$  field (which imply a  $J_0$  Bessel function corrections to the decay), Taylor expansions of the lineshape, multi-exponential decay, and a host of others. With one exception, none of these models worked better than Lorentzian.

The model finally implemented in the Bayes Analyze program takes into account one simple observation: badly shimmed resonances in the frequency domain look like a superposition of several frequencies. Indeed this is precisely what a shimming artifacts is. If the magnetic field is not homogeneous then different parts of the sample have different resonance frequencies. The model implemented in Bayes Analyze treats each resonance in the frequency domain as superposition of Lorentzian’s. The Lorentzian’s are equally spaced and, for reasons that will become apparent, the order of the expansion is always odd. The amplitude of each Lorentzian is chosen so that the lineshape of the modeled resonances match the common lineshape of the spectrum. The total relative amplitude of the Lorentzian’s sums to one, so that the amplitude of the modeled resonance retains its normal meaning. The default model used in Bayes Analyze is no shimming. That is to say the lineshapes are not expanded in a super position of Lorentzian’s. To select the shimming model, one must activate the “Settings/Shim Order” and select the expansion order that is to be used. Bayes Analyze does not automatically select this setting.

We are now in a position to state the model that describes a resonance in the Bayes Analyze package. Note that this is just the resonance portion of the model, it does not include the constants. In the Bayes Analyze package, a resonance is any group of related frequencies that are treated as a single unit by the package. Thus, a singlet, a multiplet, and a multiplet of multiplets are all examples of resonances even though the Fourier transform of some of these resonances could have multiple peaks. With this terminology out of the way, the model for a resonance in the Bayes Analyze package is given by

$$M_{Rl}(t) = \sum_{j=1}^p \sum_{k=1}^s \sum_{h=1}^o P_j S_k R_h \cos(\omega_{h_j k l} [t + t_0] + \phi_l) e^{-\alpha_l t} \quad (8.5)$$

where  $M_{Rl}(t)$  is the model of the  $l$ th resonance in the real channel, and

$$M_{Rl}(t) = - \sum_{j=1}^p \sum_{k=1}^s \sum_{h=1}^o P_j S_k R_h \sin(\omega_{h_j k l} [t + t_0] + \phi_l) e^{-\alpha_l t} \quad (8.6)$$

is the model of the  $l$ th resonance in the imaginary channel, where  $\alpha_l$  is the decay rate constant of the resonance, the frequencies,  $\omega_{h_j k l}$ , the three relative amplitudes,  $P_j$ ,  $S_k$ ,  $R_h$ , and the multiplet orders,  $p$ ,  $s$  and  $o$ , are defined shortly. The time delay,  $t_0$ , is a first order phase correction and is only present in correlated resonances. As noted earlier, correlated resonances are ones that can be phased using a common zero and first order phase. The zero order phase of the  $l$ th resonance is represented by  $\phi_l$  and it also is only present for correlated resonances. In NMR the phase of a resonance can be common to two or more correlated resonances. When the phase is specific to a single resonance, the model is called an uncorrelated resonance model. When the resonance model is uncorrelated, the model can be rewritten as a sum of sinusoids containing a real and imaginary amplitude with no constant phase and this rewritten model is actually used in Bayes Analyze for uncorrelated resonances. However, for the current discussion we do not introduce this additional complication, we are going to discuss the Bayes Analyze resonance model as if it is given by Eq. (8.5) and Eq. (8.6) respectively. Given this resonance model, the data and the resonance models are then related by:

$$d_{Ri}(t) = C_{Ri} + F_{Ri} \delta(t-1) + \sum_{l=1}^m B_{li} M_{Rl}(t) + n_{Ri}(t) \quad (8.7)$$

for the real channel, and

$$d_{Ii}(t) = C_{Ii} + F_{Ii} \delta(t-1) + \sum_{l=1}^m B_{li} M_{Ii}(t) + n_{Ii}(t) \quad (8.8)$$

for the imaginary channel, where  $C_{Ri}$  and  $C_{Ii}$  are real and imaginary constant offsets in the  $i$ th data set,  $F_{Ri}$  and  $F_{Ii}$  are the magnitude of the first point in the real and imaginary channel in the  $i$ th data set,  $\delta(t-1)$  is a Kronecker delta function,  $B_{li}$  is the time domain amplitude of the  $l$ th resonance in the  $i$ th data set, and  $n_{Ri}(t)$  and  $n_{Ii}(t)$  represents the noise in the real and imaginary channels in the  $i$ th data set. The time domain amplitude,  $B_{li}$ , is proportional to the integral over the resonance peak in the frequency domain in the  $i$ th data set. Consequently, when we use the word amplitude, for all practical purposes it is synonymous with integral. For multiplets the amplitude  $B_{li}$  is the total amplitude of the multiplet in a given data set. In the frequency domain this is the total integral over all peaks in the multiplet. The order of the primary multiplet (defined to be the multiplet with the largest coupling constant) has been designated as  $p$ . Similarly,  $s$  is the order of the secondary multiplet. The expansion order of the shimming model is  $o$ . The normalized relative amplitude of the  $j$ th peak frequency in a primary spin one half multiplet is designated by  $P_j$ . Similarly,  $S_k$  is the normalized relative amplitude for the  $k$ th peak frequency in the secondary multiplet. Up to the normalization constant,  $P_j$  and  $S_k$  are given by Pascal's triangle, Table 8.1. The normalized relative amplitudes of each Lorentzian used in the lineshape expansion are designated as  $R_h$ . Finally,  $\omega_{h_j k l}$  is the frequency of the various peaks in the spectrum of a spin one-half multiple of multiplets where the Lorentzian expansion of the lineshape is, for all practical purposes, a third multiplet. The frequencies  $\omega_{h_j k l}$  are given by

$$\omega_{h_j k l} = \omega_l - \frac{J_o(o+1-2h)}{2} - \frac{J_{pl}(p+1-2j)}{2} - \frac{J_{sl}(s+1-2k)}{2} \quad (8.9)$$

Table 8.1: Multiplet Relative Amplitudes

Multiplet Name	Relative Amplitudes											
Singlet	1											
Doublet	1	1										
Triplet	1	2	1									
Quartet	1	3	3	1								
Pentet	1	4	6	4	1							
Heptet	1	5	10	10	5	1						
Septet	1	6	15	20	15	6	1					
Octet	1	7	21	35	35	21	7	1				
Nonatet	1	8	28	56	70	56	28	8	1			
Decatet	1	9	36	84	126	126	84	36	9	1		
Undecatet	1	10	45	120	210	252	210	120	45	10	1	
Dodecatet	1	11	55	165	330	462	462	330	165	55	11	1

Table 8.1: The relative amplitudes of the peaks in a multiplet are given by Pascal's Triangle. Here we lists the names and these relative amplitudes of the primary and secondary multiplet orders. Note that before the model is actually run the Bayes\_Analyze program normalizes this table so that the total area of a multiplet is one. Consequently, the amplitude reported for a multiplet is the total amplitude for all of the peaks within the multiplet.

where the location of the center of the  $l$ th resonance is given by  $\omega_l$  and is the resonance frequency. For a simple triplet,  $\omega_l$  would be the frequency of the center of the triplet and for a doublet  $\omega_l$  would be the average of the two doublet resonance,  $J_o$  is the spacing of the frequency components in the Lorentzian expansion of the lineshape. Note that this spacing is common to all resonances and is not specific to a given resonance. This ensures that the lineshape for all resonance is the same. The  $J$ -coupling constant for the primary multiplet is designated as  $J_{pl}$ , and  $J_{sl}$  is  $J$ -coupling constant for the secondary multiplet. Note that if  $o = p = s = 1$ , Eqs. (8.7,8.8) reduce to the single exponentially decaying sinusoidal model, Eqs. (8.1 and 8.2). Also note that so long as the shimming order is one,  $s = 1$ , this model is just the exponentially decaying sinusoidal model of multiplets of multiplets described earlier. When the shimming order is not one then this model is of multiplets of multiplets where the lineshape is determined by a superposition of  $o$  Lorentzian's of relative amplitudes  $R_h$ .

The actual resonance model used in the Bayes Analyze package is a superposition of multiple resonances plus whatever constant models have been selected by the user. The relative amplitudes  $R_h$  are common to every resonance in every data set. The resonance frequencies and decay rate constants are common for every data set. Correlated resonances have a common phase  $\phi$ , while uncorrelated resonances each have a unique phase in each data set. This unique phase is actually implemented as a cosine and sine amplitude in the programs; not as an amplitude plus a phase as illustrated in Eqs. (8.1 and 8.2). The first order phase correction  $t_0$  appears only in correlated resonances. Finally, the amplitudes  $B_{li}$  is the amplitude of the  $l$ th resonance in the  $i$ th data set.

For notational simplicity, we are going to consolidate the sum in Eq. (8.7) and Eq. (8.8) into a

single summation:

$$d_{Ri}(t) = \sum_{l=1}^{m'} B'_{li} M'_{Rl}(t) + n_{Ri}(t) \quad (8.10)$$

$$d_{Ii}(t) = \sum_{l=1}^{m'} B'_{li} M'_{Il}(t) + n_{Ii}(t) \quad (8.11)$$

where constant terms were made a part of the sum, the amplitudes have been redefined to include the constant offsets and the first point problem, the functions  $M'_{Rl}$  and  $M'_{Il}$  also include the constant and first point models, and the model count,  $m'$ , is a count of the total models including the constants and first point models.

The Bayes Analyze program searches for the most probable value of these parameters. The search is done in the logarithm of the joint posterior probability for the parameters independent of the amplitudes of the resonances. The search algorithm is a modified version of a Levenberg-Marquardt algorithm. When the shimming model is activated, the shimming delta  $J_o$  and the relative shimming amplitudes  $R_h$  are common to all resonance models. Because, these parameters are the same for every resonance model, the program is looking for effects that are common to all resonances, that is why we tend to call this model a shimming artifact model, not a lineshape model. A true lineshape model would allow each resonance to have a different lineshape, as sometimes happens, and this is not what the shimming model does. The shimming model is multiplied by the exponentially decaying part of the signal. Thus the width of the Lorentzian's that make up the lineshape expansion in the frequency domain are different for every resonance. When all resonance have comparable linewidth, the shape of the line in the discrete Fourier transform is highly effected by the shimming model. When some lines are narrow and some are broad, the narrow line are primarily effected by the shimming model and appear highly non-Lorentzian. Broad lines, however, are minimally affected by this model and appear highly Lorentzian. In either case the estimated decay rate constant is decreased, not increases. The lines do decay faster, but it is interference that causes the decay, not an increase in the decay rate constant.

There are three sets of relative amplitudes:  $P_j$ ,  $S_j$ , and  $R_j$ . Each of these relative amplitude are normalized. The relative amplitude of a resonance is a weighted average over these three, so the total relative amplitude of a resonance is one. Consequently, the amplitude  $B$  is the total amplitude of all components that makes up a resonance. In the frequency domain this means that the amplitude  $B$  is the proportional to the integral over all peaks that make up a resonance. The Bayes Analyze package takes special steps to make sure that the total relative amplitude is one. For  $P_j$  and  $S_j$  this means normalizing the amplitudes from Pascal's triangle. However, for the relative shimming amplitudes, the  $R_j$ , the problem is more complex. Imposing the condition that the total relative shimming amplitude be one is accomplished by assigning the amplitude of the center component such that the total is one. However, for this model to have physical significance all of the  $R_j$  must be positive, including the center one. But imposing these two conditions, assigning the center amplitude and positively, is not enough to ensure a physically meaningful result, because these conditions allow the resonance frequency to be different from the center of a peak in the discrete Fourier transform. To avoid this problem, three additional conditions are imposed: the shimming expansion order is always odd, the center component is required to have the largest relative amplitude, and the relative shimming amplitudes must increase up to the center and then decrease beyond. With these conditions in place the search algorithm places the center of the shimming expansion in the center of

the line in the frequency domain. Consequently, the resonance frequency corresponds to the center of a resonance, and the total amplitude is proportional to the total integral over the lines in the frequency domain.

All of these conditions are imposed using prior probabilities. These prior are such that they have almost no effect on the result so long as the appropriate conditions are met, but when the conditions are about to be violated they prevent the parameters from moving to physically meaningless values. The various priors make sure that  $J$ -coupling constants and decay rate constants never go negative. They ensure that the total shimming amplitudes are ordered, positive, and sum to one. Last, they ensure that resonances marked by the user stay near where the user marked them. All of these conditions are informative and aid in making sure the output from Bayes Analyze is physically meaningful.

### 8.3 The Bayesian Calculations

The program that implements this frequency finding package is called Bayes Analyze. Unlike other packages in the Bayesian Analysis this program does not implement the calculations using Markov chain Monte Carlo; rather the program that implements this calculation uses a series of approximations and a Levenberg-Marquardt searching algorithm to locate the peak of the joint posterior probability for the parameters. At the maximum of the posterior probability, Bayes Analyze then uses a Taylor expansion about the peak to approximate the joint posterior probability for the parameters. Integrating over this approximation gives an estimate of the parameters and the uncertainty in those parameter estimates and it gives an estimate of the posterior probability for the model. Using Bayes' theorem [1] and the sum rule of probability theory, the joint posterior probability for the nonlinear parameters is given symbolically by

$$P(\Omega|D_R D_I I) = P(\Omega|I) \int P(\{B'\}\{\sigma\}|I) P(D_R D_I | \Omega \{B'\}\{\sigma\} I) d\{B'\} d\{\sigma\} \quad (8.12)$$

where we are using the notation  $\{\cdot\}$  to stand for all of the parameters of the dot kind. For example  $\{B'\}$  means all of the amplitudes in all of the data sets. The integrals are over all of the amplitudes and the standard deviations of the noise in all of the data sets. Assuming that the priors for the amplitudes are independent, the integrand can be factored to obtain

$$P(\Omega|D_R D_I I) = P(\Omega|I) \prod_{i=1}^n \int P(\{B'\}_i | I) P(\sigma_i | I) P(D_{Ri} D_{Ii} | \Omega \{B'\}_i \sigma_i I) d\{B'\}_i d\sigma_i \quad (8.13)$$

where  $P(\{B'\}_i | I)$  is the prior probability for the  $m'$  amplitudes in the  $i$ th data set,  $P(\sigma_i | I)$  is the prior probability for the standard deviation of the noise in the  $i$ th data set,  $n$  is the number of Fids being analyzed jointly, and  $P(D_{Ri} D_{Ii} | \Omega \{B'\}_i \sigma_i I)$  is the joint likelihood of the real and imaginary data in the  $i$ th data set. If we further assume that the real and imaginary measurements are independent, then likelihood can be factored into a product of two likelihoods, one for the real and one for the imaginary data:

$$P(\Omega|D_R D_I I) = P(\Omega|I) \prod_{i=1}^n \int P(\{B'\}_i | I) P(\sigma_i | I) P(D_{Ri} | \Omega \{B'\}_i \sigma_i I) P(D_{Ii} | \Omega \{B'\}_i \sigma_i I) d\{B'\}_i d\sigma_i \quad (8.14)$$

where  $P(D_{Ri}|\Omega\{B'\}_i\sigma_i I)$  and  $P(D_{Ii}|\Omega\{B'\}_i\sigma_i I)$  are the likelihoods of the real data and the imaginary data in the  $i$ th data set. Assuming logical independence, the prior probability for the amplitudes in data set  $i$ ,  $P(\{B'\}_i|I)$ , can be factored to obtain:

$$P(\{B'\}_i|I) = \prod_{j=1}^{m'} P(B'_{ji}|I). \quad (8.15)$$

The prior probability for each amplitude in including the first point and constant offset amplitudes, is represented by  $P(B'_{il}|I)$  and we will assign this prior probability using a Gaussian prior:

$$P(B'_{il}) = \left(\frac{2\pi\sigma_i^2}{\gamma^2}\right)^{-\frac{1}{2}} \exp\left\{-\frac{\gamma^2 B'_{li}{}^2}{2\sigma_i^2}\right\} \quad (8.16)$$

where  $\gamma$  indicates how strongly we believe the amplitude is small rather than large and we have included the factor of  $\sigma_i$  for computational convenience. The direct probability for the real data is represented by  $P(D_{Ri}|\Omega\{B'\}_i\sigma_i I)$  and  $P(D_{Ii}|\Omega\{B'\}_i\sigma_i I)$  is the direct probability for the imaginary data. We will assign these likelihoods for each data set using a Gaussian likelihood:

$$P(D_{Ri}|\Omega\{B'\}_i\sigma_i I) = (2\pi\sigma_i^2)^{-\frac{N}{2}} \exp\left\{-\sum_{k=1}^N \frac{[d_{Ri}(t_k) - \sum_{l=1}^{m'} B'_{li} M'_{Rl}(t_k)]^2}{2\sigma_i^2}\right\} \quad (8.17)$$

and similarly for the direct probability for the imaginary data:

$$P(D_{Ii}|\Omega\{B'\}_i\sigma_i I) = (2\pi\sigma_i^2)^{-\frac{N}{2}} \exp\left\{-\sum_{k=1}^N \frac{[d_{Ii}(t_k) - \sum_{l=1}^{m'} B'_{li} M'_{Il}(t_k)]^2}{2\sigma_i^2}\right\} \quad (8.18)$$

where it is assumed that the average noise power in the two channels is the same and that each data set contains exactly  $N$  data values.

To compute the posterior probability for one of the parameters,  $P(\omega_j|D_R D_I I)$ , two sets of integrals must be done: one over the amplitudes  $\{B'\}$  and one over all of the nonlinear parameters except the parameter being estimated. In the program that implements this calculation, the integral over the amplitudes is done exactly while the integrals over the nonlinear parameters are done approximately using a Lorentzian approximation. We briefly sketch how the integrals over the amplitudes are evaluated. If we assign a Jeffreys' prior probability [33] for the noise standard deviations, Eq. (8.16) for the prior probability for the amplitudes and assign Gaussian likelihood functions, then the joint posterior probability for the  $\Omega$  parameters independent of the amplitudes becomes:

$$\begin{aligned} P(\Omega|D_R D_I I) &= P(\Omega|I) \prod_{i=1}^n \int \frac{1}{\sigma_i} \int \left(\frac{2\pi\sigma_i^2}{\gamma^2}\right)^{-\frac{m'}{2}} \exp\left\{-\sum_{l=1}^{m'} \frac{\gamma^2 B'_{li}{}^2}{2\sigma_i^2}\right\} \\ &\times (2\pi\sigma_i^2)^{-\frac{N}{2}} \exp\left\{-\sum_{j=1}^N \frac{[d_{Ri}(t_j) - \sum_{l=1}^{m'} B'_{li} M'_{Rl}(t_j)]^2}{2\sigma_i^2}\right\} \\ &\times (2\pi\sigma_i^2)^{-\frac{N}{2}} \exp\left\{-\sum_{j=1}^N \frac{[d_{Ii}(t_j) - \sum_{l=1}^{m'} B'_{li} M'_{Il}(t_j)]^2}{2\sigma_i^2}\right\} d\{B'\}_i d\sigma_i \end{aligned} \quad (8.19)$$

where  $d_{Ri}(t_j)$  is the  $j$ th real data value of the  $i$ th data set and similarly  $d_{Ii}(t_j)$  is the corresponding imaginary data value. There are  $n$  sets of integrals over the amplitudes. Each set of amplitude integrals are  $m'$  dimensional Gaussian quadrature integrals having bounds ranging from minus to plus infinity. One evaluates these amplitude integrals by first simplifying the exponent of the integrand, and the joint posterior probability is reduced to:

$$P(\Omega|D_R D_I I) = P(\Omega|I) \prod_{i=1}^n \int \frac{1}{\sigma_i} \int \left( \frac{2\pi\sigma_i^2}{\gamma^2} \right)^{-\frac{m'}{2}} (2\pi\sigma_i^2)^{-N} \exp \left\{ -\frac{Q_i}{2\sigma_i^2} \right\} d\{B'\}_i d\sigma_i \quad (8.20)$$

where  $Q_i$  is defined to be

$$Q_i \equiv \sum_{l=1}^{m'} \gamma^2 B_{li}'^2 + \sum_{j=1}^N [d_{Ri}(t_j) - \sum_{l=1}^{m'} B_{li}' M_{Ri}'(t_j)]^2 + \sum_{j=1}^N [d_{Ii}(t_j) - \sum_{l=1}^{m'} B_{li}' M_{Ii}'(t_j)]^2. \quad (8.21)$$

To proceed,  $Q_i$  must be simplified:

$$\begin{aligned} Q_i &= d_{Ri} \cdot d_{Ri} + d_{Ii} \cdot d_{Ii} \\ &\quad - 2 \sum_{j=1}^{m'} B_{ji}' [d_{Ri} \cdot M_{Rj} + d_{Ii} \cdot M_{Ij}] \\ &\quad + \sum_{j=1}^{m'} \sum_{l=1}^{m'} B_{ji}' B_{li}' [M_{Rj} \cdot M_{Rl} + M_{Ij} \cdot M_{Il} + \gamma^2 \delta_{jl}] \end{aligned} \quad (8.22)$$

where “ $\cdot$ ” means sum over time and  $\delta_{jl}$  Kronecker delta function. If the quadrature data are thought of as a complex vector of length  $N$ , then the three lines making up  $Q_i$  are the total squared data value, the length of the projection of the data onto the model, and, if not for the prior probability for the amplitudes, the third term is the total squared length of the model projected onto itself. In this equation, the prior acts as a safeguard and prevents the total projection of the model onto the data from being equal to the model projected onto itself. When this calculation is implemented, the projection of the model onto the data should never be equal to the projection of the model onto itself, the only what this could happen is if the data had no noise. Regardless, the prior prevents this singular arithmetic and, consequently, the amplitudes that maximize the function  $Q_i$  are always real numbers, i.e., positive or negative, but never complex numbers.

Defining an interaction matrix  $g_{jk}$ :

$$g_{jl} \equiv M_{Rj} \cdot M_{Rl} + M_{Ij} \cdot M_{Il} + \gamma^2 \delta_{jl}, \quad (8.23)$$

$Q_i$  can be rewritten as:

$$\begin{aligned} Q_i &= d_{Ri} \cdot d_{Ri} + d_{Ii} \cdot d_{Ii} \\ &\quad - 2 \sum_{j=1}^{m'} B_{ji}' [d_{Ri} \cdot M_{Rj} + d_{Ii} \cdot M_{Ij}] \\ &\quad + \sum_{j=1}^{m'} \sum_{l=1}^{m'} B_{ji}' B_{li}' g_{jl}. \end{aligned} \quad (8.24)$$

These integrals are multivariate Gaussian integrals and any multivariate integral of this form may be done analytically. These integrals are exactly the same for each data set, and here we do these

integrals for an arbitrary data set and then use that result to generate the appropriate product. First, a change of variables is introduced:

$$B'_{k'i} = \sum_{j=1}^{m'} \frac{A_{ji}e_{jk}}{\sqrt{\lambda_j}}, \quad A_{ki} = \sqrt{\lambda_k} \sum_{j=1}^{m'} B'_{ji}e_{kj}, \quad (8.25)$$

with

$$dB'_{1i} \cdots dB'_{m'i} = \lambda_1^{-\frac{1}{2}} \cdots \lambda_{m'}^{-\frac{1}{2}} dA_{1i} \cdots dA_{m'i}. \quad (8.26)$$

Next we introduce a change of function:

$$R_k = \sum_{j=1}^{m'} \frac{M_{Rj}e_{kj}}{\sqrt{\lambda_k}}, \quad \text{with} \quad \sum_{j=1}^{m'} B'_{ji}M_{Rj} = \sum_{j=1}^{m'} A_{ji}R_j(t), \quad (8.27)$$

$$I_k = \sum_{j=1}^{m'} \frac{M_{Ij}e_{kj}}{\sqrt{\lambda_k}}, \quad \text{with} \quad \sum_{j=1}^{m'} B'_{ji}M_{Ij} = \sum_{j=1}^{m'} A_{ji}I_j(t), \quad (8.28)$$

where  $e_{jk}$  is the  $k$ th component of the  $j$ th eigenvector of the interaction matrix, Eq. (8.23), and  $\lambda_j$  is its  $j$ th eigenvalue. Note the  $g_{jk}$  matrix is not data set dependent, and consequently, these eigenvalues and the corresponding eigenvectors are not data set dependent. Using the property

$$\sum_{i=1}^N R_j(t_i)R_k(t_i) + I_j(t_i)I_k(t_i) = \delta_{jk}, \quad (8.29)$$

the joint posterior probability of the nonlinear  $\Omega$  parameters Eq. (20) becomes

$$\begin{aligned} P(\Omega|D_R D_I I) &= P(\Omega|I) \prod_{i=1}^n \int \frac{d\sigma_i}{\sigma_i} (2\pi\sigma_i^2)^{-2N} \left( \frac{2\pi\sigma_i^2}{\gamma^2} \right)^{-\frac{m'}{2}} \lambda_1^{-\frac{1}{2}} \cdots \lambda_{m'}^{-\frac{1}{2}} \\ &\times \int_{-\infty}^{\infty} dA_{1i} \cdots dA_{m'i} \exp \left\{ -\frac{Q'_i}{2\sigma_i^2} \right\}, \end{aligned} \quad (8.30)$$

where

$$Q'_i = d_{Ri} \cdot d_{Ri} + d_{Ii} \cdot d_{Ii} - 2 \sum_{j=1}^{m'} A_{ji} h_{ji} + \sum_{j=1}^{m'} A_{ji}^2 \quad (8.31)$$

and

$$h_{ji} \equiv d_{Ri} \cdot R_{ji} + d_{Ii} \cdot I_{ji}, \quad (8.32)$$

after completing the square and evaluating the  $m'$  integrals for each data set, one obtains

$$P(\Omega|D_R D_I I) = P(\Omega|I) \prod_{i=1}^n \int d\sigma_i \lambda_1^{-\frac{1}{2}} \cdots \lambda_{m'}^{-\frac{1}{2}} \sigma_i^{-2N-1} \gamma^{m'} \exp \left\{ -\frac{Q''_i}{2\sigma_i^2} \right\}, \quad (8.33)$$

where

$$Q''_i = d_{Ri} \cdot d_{Ri} + d_{Ii} \cdot d_{Ii} - \sum_{j=1}^{m'} h_{ji}^2. \quad (8.34)$$

Bayes Analyze uses this joint posterior probability in its calculation. However, it does not use this functional form of the calculation. This functional form is the result of doing the multivariate Gaussian integrals but it is not computationally convenient because of the presence of the eigenvalues and eigenvectors of the  $g_{kl}$  matrix. If you look at Eq. (8.31) and Eq. (8.34), it's clear that evaluating the integrals has just constrained the amplitudes  $A_{ji}$  to the values that maximized the joint posterior probability. If we go back to Eq. (8.24) and constraining  $Q_i$  to the amplitudes that maximize  $Q_i$ , then Eq. (8.24) and Eq. (8.34) must be identical. However, the maximum of Eq. (8.24) can be computed with a matrix inverse while Eq. (8.34) requires a full eigenvalue decomposition.

To locate the maximum of Eq. (8.24), one takes the derivative of Eq. (8.24) with respect to an arbitrary amplitude  $B'_{ki}$  and sets the resulting derivative to zero

$$\begin{aligned} \frac{dQ_i}{dB'_{ki}} &= -2 \sum_{j=1}^{m'} \frac{d}{dB'_{ki}} \hat{B}'_{ji} [d_{Ri} \cdot M_{Rj} + d_{Ii} \cdot M_{Ij}] + \sum_{j=1}^{m'} \sum_{l=1}^{m'} \frac{d}{dB'_{ki}} \hat{B}'_{ji} \hat{B}'_{li} g_{jl} \\ &= -2 [d_{Ri} \cdot M_{Rk} + d_{Ii} \cdot M_{Ik}] + \sum_{l=1}^{m'} \hat{B}'_{li} g_{kl} + \sum_{j=1}^{m'} \hat{B}'_{ji} g_{jk} \end{aligned} \quad (8.35)$$

but the matrix  $g_{jk}$  is a symmetric matrix so the two sums are identical and the maximum is given by the solution of:

$$0 = -2 [d_{Ri} \cdot M_{Rk} + d_{Ii} \cdot M_{Ik}] + 2 \sum_{l=1}^{m'} \hat{B}'_{li} g_{kl} \quad (8.36)$$

where we have designated the amplitudes that maximize the joint posterior probability by  $\hat{B}'_{li}$ . The  $\hat{B}'_{li}$  are given by

$$\hat{B}'_{li} = \sum_{v=1}^{m'} (g_{kl})_{vl}^{-1} T_{vi} \quad (8.37)$$

where  $T_{vi}$  is given by

$$T_{vi} = d_{Ri} \cdot M_{Rv} + d_{Ii} \cdot M_{Iv} \quad (8.38)$$

and  $(g_{kl})_{vl}^{-1}$  is the  $l$ th element of the  $v$ th column of the inverse of the  $g_{kl}$  matrix. If we were to substitute  $\hat{B}'_{ui}$  into Eq. (8.24), then  $Q_i$  becomes

$$Q_i = d_{Ri} \cdot d_{Ri} + d_{Ii} \cdot d_{Ii} - \sum_{u=1}^{m'} \hat{B}'_{ui} T_{ui} \quad (8.39)$$

where the quadratic term exactly cancels one of the linear terms.

## 8.4 Levenberg-Marquardt And Newton-Raphson

The search algorithm used by Bayes Analyze is a modified Levenberg-Marquardt algorithm that combines features of Levenberg-Marquardt and Newton-Raphson. Levenberg-Marquardt algorithms are themselves modified versions of the Newton-Raphson algorithm, where an approximation has been used in calculating the second derivatives of  $\chi^2$ . Newton-Raphson is a technique used in

finding the roots of a function. For example to find a root of a function  $f(x)$ , one computes the first derivative of  $f(x)$  with respect to  $x$  at some value of  $x$  which we will call  $x_i$ :

$$\frac{d}{dx_i} f(x_i) \equiv \dot{f}(x_i). \quad (8.40)$$

The Newton-Raphson estimate of the correction to  $x_i$  is then given by

$$\delta_i = -\frac{f(x_i)}{\dot{f}(x_i)} \quad (8.41)$$

and one updates the value of  $x_i$ :

$$x_{i+1} = x_i + \delta_i. \quad (8.42)$$

One then takes this new value of  $x_{i+1}$  and uses it as an initial guess in Eq. (8.40). The convergence of Newton-Raphson is quadratic, so the iteration procedure will usually converge to a root very quickly. The above procedure is discussed [50] and we refer you that discussion for more on Newton-Raphson and how to extend it to multiple variables.

Levenberg-Marquardt algorithms minimize  $\chi^2$ ,

$$\chi^2 \equiv \sum_{i=1}^N \frac{[d_i - y(t_i, \Omega)]^2}{\sigma^2} \quad (8.43)$$

they do not find roots; where in this equation,  $d_i$ , represents a single data item sampled at time  $t_i$ ,  $y(t_i, \Omega)$  is a model as a function of a set of  $\Omega$  parameters and  $\sigma$  is the standard deviation of the noise in the current data set. To minimize  $\chi^2$  one must find the values of the  $\Omega$  parameters for which the first derivative is zero. Consequently, in a Levenberg-Marquardt algorithm it is first derivative of  $\chi^2$  that plays the part of  $f(x)$  in a Newton-Raphson algorithm. The basic ideal is essentially the same as the Newton-Raphson algorithm. One has a function,  $\chi^2$ , that one wishes to maximize. Taking the first partial derivative of  $\chi^2$  with respect  $\omega_j$ , one obtains:

$$\beta_j \equiv -\frac{1}{2} \frac{\partial}{\partial \omega_j} \chi^2 = \frac{1}{\sigma^2} \sum_{i=1}^N [d_i - y(t_i, \Omega)] \frac{\partial y(t_i, \Omega)}{\partial \omega_j} \quad (8.44)$$

where this is a definition of  $\beta_j$ . Next one defines the second derivatives:

$$\alpha_{jk} \equiv \frac{1}{2} \frac{\partial^2}{\partial \omega_j \partial \omega_k} \chi^2 = \frac{1}{\sigma^2} \sum_{i=1}^N \left[ \frac{\partial y(t_i, \Omega)}{\partial \omega_j} \frac{\partial y(t_i, \Omega)}{\partial \omega_k} - \left\{ [d_i - y(t_i, \Omega)] \frac{\partial^2 y(t_i, \Omega)}{\partial \omega_j \partial \omega_k} \right\} \right] \quad (8.45)$$

which is a symmetric matrix of second derivatives. So far nothing in this calculation distinguishes a Levenberg-Marquardt algorithm from a Newton-Raphson algorithm. There are two distinguishing characteristic. First, in a Levenberg-Marquardt algorithm the term in curly brackets is assumed small when compared to the other terms in this equation, consequently,

$$\alpha_{jk} \approx \frac{1}{\sigma^2} \sum_{i=1}^N \left[ \frac{\partial y(t_i, \Omega)}{\partial \omega_j} \frac{\partial y(t_i, \Omega)}{\partial \omega_k} \right]. \quad (8.46)$$

To motivate this approximation note that the term,  $d_i - y(t_i, \Omega)$ , is essentially the residual at time  $t_i$  and this residual should alternate between positive and negative values. Consequently, the sum over time should on average be small. The approximation made in Eq. (8.46) has profound implications from a computational stand point. Note that the approximation does not make use of any second derivatives; rather it approximates the second derivative by a product of two first derivatives. Computationally, this means that the program that implements this calculation does not have to be programmed with second derivative information and this can be a major simplification in the calculations.

The second distinguishing characteristic in a Levenberg-Marquardt algorithm is  $\alpha_{jk}$  is modified by multiplying the diagonal of this matrix by a control parameter:

$$\alpha'_{jk} = \alpha_{jk}(1 + \lambda\delta_{jk}) \quad (8.47)$$

where  $\delta_{jk}$  is the Kronecker delta function, and  $\lambda$  controls the algorithm by making it possible to switch back and forth between a steepest descents algorithm and a Newton-Raphson algorithm. When  $\lambda$  is zero the algorithm is a Newton-Raphson algorithm and when  $\lambda$  is large the algorithm simply follows the gradients, i.e., steepest descents. The corrections to the  $\Omega$  parameters are then given by

$$\sum_{k=1}^{\nu} \alpha'_{jk} \delta_k = \beta_j \quad (8.48)$$

where  $\delta_k$  is the correction for the  $k$ th parameter. This simple set of linear algebraic equations can be solved for the  $\delta_k$  and by iterating the procedure the global minimum of  $\chi^2$  can be located. As noted, the parameter  $\lambda$  is used to control the algorithm. If  $\lambda$  is large the  $\alpha'_{jk}$  matrix is diagonally dominate and the Levenberg-Marquardt algorithm just follows the gradients. However, if  $\lambda$  is small the algorithm follows the full curvature of the space, i.e., Newton-Raphson. In Bayes Analyze, if a step increases the joint posterior probability,  $\lambda$  is decreased by about 10%, and smaller  $\lambda$  means the convergence is speeds up. However, if a step decreases the posterior probability, the value of  $\lambda$  is increased by 10% and the corrections to the  $\Omega$  parameters are recomputed. Using these new corrections, the joint posterior probability is computed and if the posterior probability increases, the new values of the  $\Omega$  parameters are keep and the iteration cycle is repeated until the first derivatives are zero.

Bayes Analyze searches in the logarithm of the joint posterior probability for the  $\Omega$  parameters, it does not search in  $\chi^2$ , consequently Bayes Analyze is not a Levenberg-Marquardt algorithm. None the less, the above discussion can be directly applied to the Bayesian calculations done because the logarithm of the joint posterior probability has components that can be written in the form of a  $\chi^2$ . Because integrating out the amplitudes just constrains the amplitude to the value that maximizes the joint posterior probability, the  $Q_i$  function can also be written as:

$$Q_i \equiv \sum_{l=1}^{m'} \gamma^2 \hat{B}'_{li}{}^2 + \sum_{j=1}^N [d_{Ri}(t_j) - \sum_{l=1}^{m'} \hat{B}'_{li} M'_{Ri}(t_j)]^2 + \sum_{j=1}^N [d_{Ii}(t_j) - \sum_{l=1}^{m'} \hat{B}'_{li} M'_{Ii}(t_j)]^2 \quad (8.49)$$

where this is just Eq. (8.20) with the amplitudes replaced by there peak joint posterior probability estimates. As explained, a Levenberg-Marquardt algorithm is essentially a Newton-Raphson method specialized to search in  $\chi^2$ . The reason we rewrote the statistic this last time, Eq. (8.49), was to express  $Q_i$  in a  $\chi^2$  like form. Now  $Q_i$  is not strictly  $\chi^2$  and the statistic used in the search isn't either.

None the less we are going to define a  $\chi^2$  term and then discuss how Bayes Analyze implements it's search using this definition:

$$\chi^2 \equiv \log P(\Omega|I) + \sum_{i=1}^n \frac{Q_i}{\sigma_i^2} \quad (8.50)$$

where  $P(\Omega|I)$  is the prior probability for the nonlinear parameters and  $Q_i$  is given by Eq. (8.49). This definition of  $\chi^2$  is essentially the logarithm of the joint posterior probability for the  $\Omega$  parameters independent of the amplitudes where a some constants have been dropped.

To implement a Levenberg-Marquardt algorithm, one must define both an  $\beta_j$  and an  $\alpha_{jk}$ , in the Bayes Analyze program these are defined as:

$$\begin{aligned} \beta_j &\equiv -\frac{1}{2} \frac{\partial}{\partial \omega_j} \chi^2 \\ &= -\frac{1}{2} \frac{\partial}{\partial \omega_j} \log P(\Omega|I) - \frac{1}{2} \gamma^2 \sum_{l=1}^{m'} \frac{\partial}{\partial \omega_j} \hat{B}'_{li}{}^2 \\ &\quad + \sum_{i=1}^n \frac{1}{\sigma_i^2} \sum_{j=1}^N \left[ d_{Ri}(t_j) - \sum_{l=1}^{m'} \hat{B}'_{li} M'_{Ri}(t_j) \right] \left[ \frac{\partial}{\partial \omega_j} \sum_{l=1}^{m'} \hat{B}'_{li} M'_{Ri}(t_j) \right] \\ &\quad + \sum_{i=1}^n \frac{1}{\sigma_i^2} \sum_{j=1}^N \left[ d_{Ii}(t_j) - \sum_{l=1}^{m'} \hat{B}'_{li} M'_{Ii}(t_j) \right] \left[ \frac{\partial}{\partial \omega_j} \sum_{l=1}^{m'} \hat{B}'_{li} M'_{Ii}(t_j) \right] \end{aligned} \quad (8.51)$$

and  $\alpha_{jk}$  is defined using the same approximation used in a Levenberg-Marquardt algorithm:

$$\begin{aligned} \alpha_{jk} &\equiv \frac{1}{2} \frac{\partial^2}{\partial \omega_j \partial \omega_k} \chi^2 \\ &\approx \frac{1}{2} \frac{\partial^2}{\partial \omega_j \partial \omega_k} \log P(\Omega|I) + \frac{1}{2} \gamma^2 \sum_{l=1}^{m'} \frac{\partial^2}{\partial \omega_j \partial \omega_k} \hat{B}'_{li}{}^2 \\ &\quad + \sum_{i=1}^n \frac{1}{\sigma_i^2} \sum_{\nu=1}^N \left[ \frac{\partial}{\partial \omega_j} \sum_{l=1}^{m'} \hat{B}'_{li} M'_{Ri}(t_\nu) \right] \left[ \frac{\partial}{\partial \omega_k} \sum_{l=1}^{m'} \hat{B}'_{li} M'_{Ii}(t_\nu) \right]. \end{aligned} \quad (8.52)$$

With this definition of  $\beta_j$  and  $\alpha_{jk}$  it is now possible to implement a Levenberg-Marquardt algorithm. This algorithm has a number of interesting properties not present in a standard Levenberg-Marquardt algorithm. For example, by including the prior probabilities for the nonlinear parameters,  $P(\Omega|I)$ , one can effectively bound a parameter to positive values and we do this using a positive prior, Eq. (3.7). Second, by taking into account the presence of multiple data sets with differing standard deviations, this statistic allows many different kinds of data to be combined. Finally, by marginalizing out the amplitudes, the Levenberg-Marquardt algorithm does not search on the amplitudes. This turns out to be very significant when multiple resonances and data sets are present. In NMR experiments, one often takes multiple data sets. These data sets measure the same quantity, but with some parameter varied, often a delay time or a gradient. The amplitudes of the

resonances from these multiple data sets are then used in additional analysis, such as inversion recovery experiments. However, the number of these data sets can be large, in the hundreds or even thousands. If we had not marginalized out the amplitudes, a 10 resonance problem, looking for common frequencies in one thousand free induction decays would have 10 frequencies, 10 decay rate constants, perhaps one or two phase parameters, and 10,000 or 20,000 amplitudes depending on how the problem is parameterized. For all practical purposes, the problem would be intractable because all of the frequencies, decay rate constants and amplitudes are correlated, i.e., one must search on all of these parameters simultaneously. However, by marginalizing out the amplitude the resulting search only involves 20 or so parameters. Consequently, marginalization is a key feature in jointly analyzing arrayed experiments.

Of course, implementing this calculation is a tricky problem. We have not, for example, talked about how one computes the derivatives of the amplitudes, nor how to estimate the noise standard deviations. Its enough to know that the derivatives of the amplitudes and the standard deviations can be computed at each step in the calculation. Consequently, when the search completes, one has the peak value of the joint posterior probability, the zero first derivatives, a very good approximation to the second derivatives, and estimates of the standard deviation of the noise in each data set. Together these can be used to write an Gaussian approximation to the joint posterior probability for the parameters. The approximate joint posterior probability for the parameters can be used to estimate the parameters using mean  $\pm$  standard deviation estimates. Additionally, the posterior probability for the model can be approximated as the integral over the approximate joint posterior probability for the parameters and this approximate joint posterior probability for the model can be used to determine when the algorithm should stop. So a general description of the Bayes Analyze algorithm is:

1. Using the Fourier transform of the residuals [7], compute the odds ratio [5] that any given peak in the spectrum is a resonance.
2. If no peak in the odds ratio rises above the noise threshold, exit the analysis. Otherwise, using the single most probable resonance frequency from the signal detection algorithm, postulate a resonance model containing one more resonance.
3. Using the Levenberg-Marquardt search algorithm optimize the parameters in this model.
4. Using the peak, the covariance matrix and the estimated noise standard deviations, approximate the joint posterior probability for the parameters, then using this approximation estimate the parameters and the posterior probability for the model.
5. Output these parameter estimates to the various Bayes Analyze output files.
6. If the posterior probability for the model decreases or the maximum number of resonances has been added to the model, finish up the analysis and exit Bayes Analyze.
7. If the posterior probability for the model increased, go back to the first item and repeat this procedure until either no additional candidate resonances can be found or the posterior probability for the model decreases.

## 8.5 Outputs From The Bayes Analyze Package

Unlike the other packages, the Bayes Analyze package is not a single program, rather it is 4 different programs that are run one right after the other and a fifth that is run on demand. The main program is called Bayes Analyze and it is the frequency finding program. There are three programs, called Bayes Summary 1, 2 and 3, that take the outputs from Bayes Analyze and reformat them into three summary reports. Finally, there is a modeling program, called Bayes Model, that takes the output from Bayes Analyze and creates a model of the input Fid. The Bayes Analyze program does not use Markov chain Monte Carlo to do its calculations. Consequently, the outputs from the Bayes Analyze package are completely different from the outputs from MCMC packages. Thus to view these outputs, a different set of output widgets are needed. The interface has two different output areas on the text results viewer, one for MCMC packages and one for the Bayes Analyze package.

The Bayes Analyze output files and the summary reports are numbered with a suffix. For example the main Bayes Analyze output file is named “bayes.output.nnnn” where the “nnnn” is related to the From, To, and By fields described earlier, see page 158. When Bayes Analyze process a set of Fids, it begins with Fid number “From” and analyzes “By” number of Fids. The output from this analysis is written into a series of files labeled “name.From” where “name” is one Bayes Analyze output files and “From” is the “From” Fid number. After Bayes Analyze finishes this analysis, it increments the “From” Fid number by “By” Fid number and then checks to see if it has reached Fid “To”. If not it repeats the analysis using this new “From” Fid number, again writing out files named “name.From.” This loop continues until all of the Fids have been processed. Some of these outputs are available through the interface, and some only by looking at them within the WorkDir. Here is a brief description of the more important input and output files used by the Bayes Analyze package:

**bayes.params** file is written by the interface and serves as the input parameter file to the Bayes Analyze program. It contains various parameter settings and the initial model to be processed.

The parameter file is divided into three general sections, a header, the global parameters, and the resonance parameters. Each of these three sections is describe in detail in Subsections 8.5.1.1, 8.5.1.2 and 8.5.1.3.

**bayes.params.nnnn** are files written by Bayes Analyze for each individual sub-analysis. They are updated version of bayes.params, updated in the sense of containing the maximum joint posterior probability estimates of the frequencies, decay rate constants, etc. Additionally, the “From,” “To,” and “By” numbers are updated to reflect this particular sub-analysis. For example, if the current analysis is running Fids 5 through 10, then a parameter file, bayes.params.0005, is written and used by the interface to mark the location of the various resonances. After Bayes Analyze completes the interface reads the bayes.params.nnnn file and the updated model is then available for further processing within the interface. Note that which bayes.params.nnnn file is read by the interface depends on which Fid is to be modeled. That is to say if you are planning to model Fid 25 from an array, the interface will figure out which bayes.params.nnnn file contains the parameters for Fid 25 and then read those parameters. Again, each of these three sections is describe in detail in Subsections 8.5.1.1, 8.5.1.2 and 8.5.1.3.

**bayes.output.nnnn** are the main output files from Bayes Analyze and they contain detailed information about each model Bayes Analyze processed. The description of this file may be found in Subsection 8.5.3.

**bayes.proBABILITIES.nnnn** contain a one line entry for each model processed by Bayes Analyze. These lines contain the base 10 logarithm of the posterior probability for each model processed by Bayes Analyze. Note the file is cumulative and results from multiple runs of Bayes Analyze may be viewed. For a complete description of this file, see Subsection 8.5.4.

**bayes.log.nnnn** is a detailed log of each step taken in the searching algorithm for each model processed by Bayes Analyze, see Subsection 8.5.5 for a complete description of this file.

**Bayes.accepted** is used by the interface to show the current status of an analysis. For example, if you are running Bayes Analyze when the “Status” button is activated, the interface will display the current contents of the Bayes.accepted file. This file is a recent addition to Bayes Analyze and is a simple copy of the bayes.status.nnnn file. It was added to Bayes Analyze so the new interface could determine the status of the current analysis. For a complete description of this file see Subsection 8.5.6.

**bayes.status.nnnn** contains the status of analysis nnnn. Usually by the time this file can be viewed, it simply indicates the analysis is completed. As noted above this file is also written into Bayes.accepted and the Bayes.accepted file may view while Bayes Analyze is running. For a complete description of this file see Subsection 8.5.6.

**bayes.model.nnnn** is used by Bayes Model to produce a maximum joint posterior probability model of an Fid. As one of its outputs Bayes Analyze writes the parameters that maximize the joint posterior probability into the model file. The model file, unlike the bayes.params.nnnn files, contain both the frequency and amplitude estimates. For a complete description of this file see Subsection 8.5.7.

**bayes.summary1.nnnn** The summary files, bayes.summary1.nnnn, bayes.summary2.nnnn and bayes.summary3.nnnn, are produced by Bayes Analyze package whenever you run the analysis. The summary1 or “Best Model” report contains parameters from the most probable model. For a complete description of the summary1 report see Subsection 8.5.8.

**bayes.summary2.nnnn** The summary2 or “Best Summary” report is a highly condensed version of the summary1 report. However, because each line displays the information for a single resonance, the report is available only for nonarrayed free induction decays. For a complete description of the summary2 report see Subsection 8.5.9.

**bayes.summary3.nnnn** Finally, the summary3 or “Best Regions” report displays the intensity (normalized and total), contained within a series of regions. These regions are defined by using the “Options/Set Regions” button. Use of this report and how to set the regions is described further in Subsection 8.5.10.

### 8.5.1 The “bayes.params.nnnn” Files

Bayes Analyze reads the parameter file and uses the information in the parameter file to direct its operations. The parameter file is used to communicate between the interface and the Bayes Analyze program. The name of the parameter files may be either “bayes.params” or “bayes.params.nnnn.” If the number is not present the file was output from the interface and is a Bayes Analyze input file. If the number is present the file was output from Bayes Analyze package and serves as input to the

interface. The parameter file consists of three general sections, a header, the global parameters, and the resonance model. We describe each of these sections in the following.

### 8.5.1.1 The Bayes Analyze File Header

The Bayes Analyze package writes a file header into many of its output files. This header describes the analysis, and serves as a record of the analysis. An example header is shown in Fig. 8.3. In the Bayes Analyze package, blank and comments lines (lines starting with “!”) are ignored and may appear anywhere within the parameter or model files. As you can see, the first three lines in the file header are comments. Comments are ignored by Bayes Analyze and Bayes Model. Also, file names appearing in Fig. 8.3 are referenced from the directory where Bayes Analyze is to be run. In the interface when Bayes Analyze is run, the program runs in the current WorkDir so the file and directory names appearing on Lines 05–08 are referenced from that working directory. Here is a description of the parameter file header:

**Line 01** is a comment and indicates the name of the Fid loaded into the current WorkDir.

**Line 02** is a comment containing the name of the input file type, in this case a “Parameter File.”

**Line 03** is a comment containing name of the user who created the file and the date and time.

**Line 04** indicates the number of parameters that are to follow, in this example there are 32. Each line contains a description followed by the value of the parameter. Some of the parameters are text strings and some of them are numbers. Text strings start in column 22 and extend to the right. Numerical values may appear anywhere from column 22 through 32. Bayes Analyze recognizes parameters by the labels that appear in column 5, these labels must appear as shown, including capitalization. The ordering of the parameters is generally not critical, although keeping them in the order shown is advisable. The numbers on the left and the column headings were added for reference and do not appear in the file.

**Line 05** is the file version number and, as Fig. 8.3 implies. This parameter must follow the configuration parameters. The programs have the ability to read old file versions. We have included this parameter because output files from a Bayesian Analysis may be saved – sometimes for long periods of time and we did not want to lose the ability to read and model older versions of these files.

**Line 06** is the fully qualified name of the input Fid file. As noted earlier, when Bayes Analyze is run it is run in the current WorkDir so the fid file must fully qualified from WorkDir. Consequently, when the parameter file is written by the interface this field will always be fid/fid, however any fully qualified name is allowed.

**Line 07** is the fully qualified name of the input procpar file.

**Line 08** is the name of the current analysis directory. This is the fully qualified name of the directory in which the output from Bayes Analyze is to be written. When the parameter file is written by the interface, the analysis directory is “BayesAnalyzeFiles”.

**Line 09** is the name of the directory where Bayes Model is to create the model. This is Bayes.model.fid when the interface generates this file, but it need not be that way and may be set to any value if Bayes Analyze is run manually.

Figure 8.3: The Bayes Analyze File Header

```

          1          2          3          4          5          6
123456789012345678901234567890123456789012345678901234567890123

01! File: "~larry/bayesian/Bayes.test.data/BayesAnalyze/MTz_C13_irradR_20C.fid"
02! Bayesian Analysis Input Parameter File
03! Created 07-May-2001 15:14:21 by Larry

04 31 Configuration Parameters

05 File Version = 3.000
06 Fid File = fid/fid
07 Procpa File = fid/procpa
08 Analysis Dir = BayesAnalyzeFile
09 Model Dir = Bayes.model.fid
10 Model Dir Org = DATA
11 Units = PPM
12 Activate Shims = YES
13 Activate Delay = YES
14 Data Type = VNMR
15 Noise = NO
16 Output = FULL
17 Default Model = (CP)
18 First Fid = 1
19 Last Fid = 1
20 No Fids = 1
21 Total Points = 1024
22 Complex Points = 512
23 Noise Start = 0
24 Model Points = 1024
25 Model Fid = 1
26 Prior Odds = 0.00000
27 Sampling Time = 0.29900
28 Spec Freq = 300.00000
29 User Reference = -2.84349
30 True Reference = -2.84349
31 Total Models = 3
32 Shim Order = 7
33 Max Freqs = 10
34 Max Candidates = 1
35 Default Lb = 1.08755

```

Figure 8.3: The Bayes Analyze File Header is present in most output files from the Bayes Analyze package. This header contains all of the parameters used to run Bayes Analyze. Among other things, it contains the name of the Fid file, the procpa and the output directory where the model is to be written. It also specifies the directory organization, the units and many other parameters needed to run Bayes Analyze. See text for a more extensive description of the file header.

Figure 8.4: The bayes.noise File

```
1 3.22299585E+01
2 3.19859327E+01
3 3.22769104E+01
4 3.18850559E+01
5 3.21451727E+01
6 3.19790225E+01
7 3.20911816E+01
8 3.19975486E+01
9 3.19348404E+01
```

Figure 8.4: The file bayes.noise is a two column Ascii file generated by the interface when a noise region is set. The first column is the number of the Fid and the second is the root mean-square data value in the noise region, and is an estimate of the standard deviation of the noise in the noise region. Note that the bayes.noise contains one entry for each trace in the Fid, i.e., the “From,” “To” and “By” fields are not used when generating the bayes.noise file.

**Line 10** specifies the organization of the directories. The new interface sets this to “DATA” meaning that the output directory is a standard Fid directory. However, this can also be set to “VNMR,” in which case the Bayes Analyze expects the structure of the analysis directory to be in VnmrJ format.

**Line 11** specifies the units to use in the analysis. The valid entries are “PPM,” or “HERTZ”.

**Line 12** specifies whether the lineshape model is to be used. The valid values are “YES” or “NO” and these mean yes the lineshape is to be expanded, or no it is not. If lineshape, or shimming model is activated then the shimming expansion order, shown on Line 32, must be specified as greater than one.

**Line 13** is used to activate the first order phase correction. Note that the new interface does not allow you to modify this parameter, it is automatically activated by the interface.

**Line 14** indicates the type of input data to be processed. The new interface sets this to “VNMR” indicating that the input file is a VNMR Fid.

**Line 15** indicates whether an estimate of the root mean-square estimate of noise standard deviation is available. The valid values for this indicator are “YES” or “NO.” These indicate yes a noise measurement is available or no a noise measurement is not available. If a noise measurement is available the Bayes Analyze program will attempt to read the “bayes.noise” file. An example of this file is shown in Fig. 8.4. This noise file is two columns, the first being the number of the free induction decay and the second being the estimated root mean-square noise value. This file must be present if the noise used indicator is “YES.” If used, there must be one entry in this file for every free induction decay in the input data. This file is built by the menus using the Bayes Noise program.

**Line 16** indicates the format of output written to standard out. This entry may be be “FULL,” “BRIEF” or “NONE.” This option is set to “BRIEF” by the new interface.

Table 8.2: Bayes Analyze Models

Model Name	Description	Resonance
Real Constant	A constant in the real channel	no
Imaginary Constant	A constant in the imaginary channel	no
DC Offset In Both	The same constant in both channels	no
First Point Problem	Model the data as if the first point were in error	no
(CP)	The Correlated Phase model	yes
(UP)	The Uncorrelated Phase model	yes

Table 8.2: The Bayes Analyze package currently supports the 6 models listed here. The first column is the model name. The second column is a longer description of the model. Finally, the third column indicates whether the model is a resonance model or not.

**Line 17** is used to set the resonance model used in the automatic mode of Bayes Analyze. Table 8.2 is a list of all the model names currently in use in the Bayesian Analysis package. There are two resonance models in this table, these are the Correlated Phase model, “(CP)” and the Uncorrelated Phase model, “(UP)”.

**Lines 18** through 20 are the first, last, and number of Fids respectively. Bayes Analyze will begin processing the data starting with the first Fid. It processes number of Fids at a time looking for common resonances. After processing a given block of free induction decays, the program will increment first Fid by number of Fids and then proceed to process this block of Fids. This process continues until all indicated Fids have been processed. As noted in Section 8.5.5 the number of the first Fid processed in a cycle is appended to all of the output files from Bayes Analyze.

**Lines 21** is the total and number of data values to be processed by Bayes Analyze. Note this is the sum of the total real data values plus the number of complex data values, i.e., 2 times the total complex data values.

**Lines 22** are the total number of complex data values to be processed by Bayes Analyze. The complex number of data values is redundant in that it is just half the total number of data values.

**Line 23** is the starting position of the noise region. This field is not used by Bayes Analyze. This field was placed in the parameter file so that it could be saved when an analysis is saved.

**Line 24** is the number of model points (real plus complex) in the output modeled Fid. When the model program, Bayes Model, creates a model of a Fid it creates an arrayed Fid in the model experiment. The first element of this array is a copy of your data, if number of model points is less than “np” only part of your data is copied to the output Fid. If it is greater than “np”, the data are copied and then zero padded. The default is for this value to be equal to “np”.

**Line 25** is the number of the free induction decay to model. If the user does not specify the free induction decay to model it defaults to the free induction decay that was last displayed in the experiment. If no free induction decay has been displayed it defaults to the first free induction decay processed by the user.

- Line 26** the prior odds is automatically set to zero by the interface and should not be changed.
- Line 27** is the sampling time of the data being processed. This field is the same as the VNMR parameter “at” unless you are using the signal region, then its the sampling time for the data being processed. If you are processing 25% of the free induction decay then this field is 25% of “at” etc.
- Line 28** is the spectrometer frequency (the “reffrq” parameter) and is used in the conversion to and from ppm. Internally the Bayes Analyze program uses radian as the frequency units. When either Bayes Analyze or Bayes Model reads the input parameter file the programs determine what units are being used and makes the appropriate conversions to internal units.
- Line 29** is the reference frequency use by the Bayes Analyze. This parameter has no exact correspondence to any VNMR parameter. If the number on line 29 is called “ref,” then it is computed as:
- $$\text{“ref”} = -([\text{“rfp”} - \text{“rfl”}] + [\text{“sw”}/2.0]) \quad (8.53)$$
- where “rfp,” “rfl” and “sw” are the reference point, the reference line, and the sweep width respectively. For ppm, “ref” is then divided by the reference frequency “reffrq.”
- Line 30** is the true reference. The meaning of the field is exactly the same as the user reference.
- Line 31** is a count of the total model components in the input model to be analyzed in slave mode. A model component might be a singlet, a multiplet, or a constant.
- Line 32** is the order of the shimming expansion being done on the lineshape. This field is ignored unless the activate shims indicator is “YES”. An example of setting the activate shims indicator is shown on Line 11 in Fig. 8.3. When this indicator is set, the expansion order must be 3, 5, or 7.
- Line 33** is the maximum number of new resonances that may be incorporated into the model on this run of Bayes Analyze.
- Line 34** is the maximum number of resonances that may be incorporated from one run of the signal detection algorithm. This number is set to 1 by the interface and should not be changed.
- Line 35** is current value of “lb.”

### 8.5.1.2 The Global Parameters

Immediately following the configuration parameters are the global parameters. An example of the global parameter section is shown in Fig. 8.5. The numbers on the left as well as the column indicators on the top have been added for reference purposes. In the example shown there are 10 global parameters, the maximum. When these parameters appear in the parameter file, they must appear in the order shown.

**Line 01** is a count of the number of global parameters.

**Line 03** is the constant phase used by Bayes Analyze.

Figure 8.5: Bayes Analyze Global Parameters

```

0          1          2          3          4          5
1234567890123456789012345678901234567890123456789012
01  10 Global Model Parameters
02
03  Center Phase = 219.17041042
04  Time Delay   = 0.00830983
05  Shim Delta   = 36.923702254267 2.50000000000000
06  R Minus 3   = 0.0121083700262
07  R Minus 2   = 0.0123818187190
08  R Minus 1   = 0.2878006370120
09  R Center    = 0.2881405188680
10  R Plus 1    = 0.1340506287029
11  R Plus 2    = 0.1334510175742
12  R Plus 3    = 0.1320670090977

```

Figure 8.5: The global parameter immediately follow the header. There are at most 10 global parameters. Here we have exhibited all 10 of these. These parameters include the two phasing parameters, the spacing of the Lorentzian's in the shimming model (Shim Delta), and the relative amplitudes of the Lorentzian's in the expansion.

**Line 04** is the first order phase parameters used for correlated resonances. VNMR allows spectra to have both a constant and a linear frequency dependent phase. These are specified by the left and right phases. These parameters are natural for spectra but are clumsy in the time domain. The Bayesian Analysis package expresses this phase variation as a constant phase with a time delay.

**Lines 05** through 12 concern the expansion of the lineshape in Lorentzian's. These parameters are not present unless the shimming model is turned on and Bayes Analyze has been run previously. Line 05 is the shim delta: it is the separation frequency between the Lorentzian's in the shimming model expansion of the lineshape. It is the exact analogue of the  $J$ -coupling constants in multiplets. There are two entries present on line 05: its current value and its initial value. Its initial value is set to "lb"/(expansion order).

**Lines 06** through 12 are the relative amplitudes assigned to the Lorentzian's that make up the lineshape expansion. If the shimming order is set to three, then there are three Lorentzian's for each peak in the spectrum. The total area under these peaks is required to add up to one, so there are only two independent amplitudes; the third is redundant. By convention we take the redundant amplitude to be the center amplitude. In Fig. 8.5 there are 7 relative amplitudes shown, i.e., a seventh order expansion of the lineshape is being done. Each relative amplitude is labeled "R" plus or minus something. For a seventh order expansion the largest relative amplitude is the center relative amplitude, Line 04. Line 06 is labeled "R Minus 3" and this means the third and smallest relative amplitude whose frequency is given by  $\omega - 3\delta_{\text{shim}}$  where  $\omega$  is the resonance frequency. Similarly, "R Plus 2" means the relative amplitude of the shimming frequency given by  $\omega + 2\delta_{\text{shim}}$ .

Figure 8.6: The Third Section Of The Parameter File

```

01! # Name
02 1 Real Constant
03
04! # Name
05 2 Imaginary Constant
06
07! # Name          Order      Frequency      Init Value
08 3 (CP) Triplet  (3,1) 0      1.2913619868959 1.2900933000000
09
10                      Decay Rate      Init Value
11                      1.0405457695713 1.0875500000000
12
13                      Primary J      Init Value
14                      6.9823931646449 6.6850000000000
15
16                      Secondary J      Init Value
17                      0.0000000000000 0.0000000000000

```

Figure 8.6: The third section of the parameter file contains the model to be used in the analysis. The first part of this model is the constant models. Here there are two constant models, Line 02, and Line 05. Lines 08 through line 17 is an example of a triplet resonance model. In the actual parameter file all the parameters for a resonance are on a single line, so lines 10 through 17 should be part of line 08 and each field should occupy exactly 15 positions with an additional space separator. However, I broke that line apart so the entire contents of the the line would be visible.

### 8.5.1.3 The Model Components

The parameter file consists of the header, the global parameters, and then the individual resonance models. Figure 8.6 is an example containing some constant models as well as a resonance model. In this example there are three model components. Each model component consists of a blank line, a comment line used as a header, here is a description of these lines:

**Line 01** is the comment line for the first model.

**Line 02** is the description of the model. In this case the model is a real constant model and constant models do not have parameters, so there is nothing else on this line except the model number and the model name.

**Line 03** is a blank line.

**Line 04** is the model header.

**Line 05** is the description of the imaginary constant model. The two fields are simply the model number and the model name.

**Line 06** is a blank line.

**Line 07** 10, 13 and 16 is the model header for the third model. This header is for a multiplet and when this header was generated it labels the most complicated model used by Bayes Analyze, a multiplets of multiplets. Also note, that lines 10, 13 and 16 should be part of line 07 with each field on the header taking 15 spaces with an additional space separator.

**Line 08** 11, 14 and 17 are the parameters associated with a multiplet. In the real parameter file lines 08, 11, 14 and 17 are on the same line with each item taking exactly 15 spaces and is space separated from its neighbor. Here we have separated them so that they have not dropped off the right-hand margin.

**3** is the model number.

**(CP)** is the resonance type, in this case the resonance is a Correlated Phase resonance. See Table 8.2 for a list of the valid model type.

**triplet** is the resonance name, i.e., single, doublet, triplet, etc.

**(3,1)** is the order of the primary and secondary multiplets. In this case a triplet of singlets.

**0** the zero following the multiplets order is the relative phase of the multiplet. The relative phase may have values 0, 90, 180, and 270. The new interface sets this phase to zero. Here zero means that the resonance is in phase with the other resonances.

**1.2913619868959** is the frequency of the multiplet in currently selected units, in this case PPM.

**1.2900933000000** is the initial value of this frequency prior to optimization.

**Line 09** is a blank line.

**Line 10** is header for the decay rate constant.

**Line 11** is the decay rate constant and its initial value.

**Line 12** is a blank line.

**Line 13** is header for the primary J-coupling constant.

**Line 14** is the primary J-coupling constant.

**Line 15** is a blank line.

**Line 16** is header for the secondary J-coupling constant.

**Line 17** is the secondary J-coupling constant. Note these entries for the secondary coupling constant are not present unless you have a multiplet of multiplets.

### 8.5.2 The “bayes.model.nnnn” Files

The bayes.model.nnnn file is a copy of the bayes.params file that contains the output amplitudes. These amplitudes follow each frequency component and are written in this form:

```
!                               Fid           Cos Amplitude   Sine Amplitude
                               1             13413.806578806  411.29599022536
```

when uncorrelated resonances are modeled and in this form:

!	Fid	Amplitude
	1	4537.7682653001
	2	4571.1890384819
	3	4450.1289475121
	4	4577.9827364511
	5	4537.7818733113

when correlated resonances are modeled. The Fid number on this line indicates that there were 5 different fid's or traces in the data set analyzed. When Bayes Model is activated, it finds the resonances and amplitudes for the trace being modeled and then generates an output Varian Fid that contains the original trace as trace 1. Trace 2 is the model built by Bayes Model and is a sum of the sinusoids in the Bayes Model file. Trace 3 is the residual, the difference between the data and the model Fid. As Fid data are complex all three of these three traces are complex time domain Fid data. Finally, if Bayes Analyze found 10 resonances then following each of these first 3 traces there will be 10 additional traces, one containing a model of each of the 10 sinusoids. The output Fid model is used by the interface whenever the Fid Model viewer is activated.

### 8.5.3 The “bayes.output.nnnn” File

The output file is the main output report from the Bayes Analyze program. The output file is name “bayes.output.nnnn” where, as explained in Section 8.5.5, the number nnnn is the number of the first free induction decay who's analysis is contained in this output file. This file contains the detailed output from each model processed by Bayes Analyze. Because it contains so much detail, it can be rather difficult to follow and should be printed only when this information is needed. When the detail is not needed, the summary report should be used. The summary report is essentially the section out of the output file pertaining to the model with maximum probability. For more information on the summary report and the difference between these two reports see Sections 8.5.8 and 8.5.9.

When Bayes Analyze runs in its automatic mode many different models may be analyzed. At the end of the analysis on each model, the optimized values of the model parameters are written to the output report. In this section we describe this report in detail. The output reports starts with a listing of the configuration parameters associated with the analysis. An example of these configuration parameters is given in Fig. 8.3 and an explanation of these configuration parameters is given in Section 8.5.1.1. There are two small differences between what is shown in Fig. 8.3 and what is printed in the output report. These difference consist of a header a the beginning of the report. This header merely indicates that the initial configuration parameters follow. The other difference is at the end of the configuration parameters. The output file contains a list of the resonances processed in the slave mode. Figure 8.7 is a small sample of what this listing might look like. There are two multiplets, one triplet and one quartet in the initial model. The notation used to indicate the triplet and quartet is the same as that explained in Section 8.5.1.3. This is followed by the resonance frequency and the primary and secondary coupling constants. These coupling constants are zero if they are not used.

One or more sets of detailed information follows the initial model. The detailed information consists of the output from the signal detection calculation, output from the optimization of the joint probability for the parameters, the output from the model selection calculation, and the normalized

Figure 8.7: Example Of An Initial Model In The Output File

```
Initial Model:
# Order      Frequency      Decay Rate      Primary J      Secondary J
1 (3,1) 0    1.2913982000000 0.8574675300000 6.9821668000000 0.0000000000000
2 (4,1) 0    4.8648472000000 0.8881744100000 6.9795215000000 0.0000000000000
```

Figure 8.7: This example shows two resonances, one triplet of singlets, the (3,1), and one quartet. The parameters, other than the amplitudes, follow. Here there are nontrivial values of the frequencies, the decay rate constants and the primary  $J$  coupling constant.

Figure 8.8: Base 10 Logarithm Of The Odds

```
Base 10 Log Evidence for The First Frequency is: 49.6
-----
```

Figure 8.8: After the signal detection algorithm runs, the base 10 logarithm of the odds in favor of a resonance are printed to the output file. Here the odds are  $10^{49.6} : 1$  in favor of a sinusoidal signal component.

probability for the models analyzed so far.

The signal detection output consists of only single line preceding the dashed line, see Fig. 8.8 for an example of this output. As the message indicates, this is a base 10 logarithm. In this particular case the evidence indicates that it is a bet of approximately  $10^{49.6} : 1$  that a resonance is present.

After the signal detection calculation is performed, and assuming a resonance was detected, Bayes Analyze builds a model having one new resonance and then optimize the parameters associated with this model. The resonance added corresponds to the frequencies having maximum signal detection probability. The optimization step locates the maximum of the joint posterior probability for the parameters. The second section of the output report contains the values of the parameters for which the joint posterior probability is maximum. An example of this output is shown in Fig. 8.9. Figure 8.9 consists of the global parameters, here the two phase parameters, and this is followed by the model parameters. In the parameter file this model information was on one very long line. In the output report each model component is on several lines and each component has one or more amplitudes associated with it. The individual model components all start with a header. This header begins “#Name” and is always present. There are five headers shown in Fig. 8.9, so there are five model components. Each model component consists of a model name, the estimated resonance parameters, and the estimated amplitudes. Both constants and resonances have amplitudes; however, only resonances have frequencies, decay rate constants and  $J$ -coupling constants. The model name, multiplet order, relative phase, and other parameters are the same as those described in Section 8.5.1 and we will not repeat that here. The resonance frequency is printed on the same line with the model name and is followed by up to three additional lines containing the decay rate, the primary  $J$ -coupling constant, and the secondary  $J$ -coupling constant. Each of these parameters has an associated uncertainty printed with it.

Following the model component name and the resonance parameter, if present, are the amplitudes associated with each free induction decay processed. If there were 10 free induction decays, there will be 10 amplitudes. The number of the free induction decay that the amplitude is associated with appears on each line. In addition to printing the amplitudes, the uncertainty in the estimated

Figure 8.9: A Small Sample Of The Output Report

```

Base 10 Log Evidence for The Next Frequency is: 49.6
-----
Number of Models In This Step 6
Center Phase      Uncertainty      Time Delay      Uncertainty
-5.7941E+01      7.32E-01          8.322E-03      6.28E-03

# Name
1 Real Constant
      Fid  Amplitude  Uncertainty
      1  5.106E+01  4.41E+01

# Name
2 Imaginary Constant
      Fid  Amplitude  Uncertainty
      1  1.16E+01  4.41E+01

# Name      Order      Type      Parameter      Uncertainty
3 (CP) Singlet      (1,1) 0      Freq  1.16733E-01  3.54E-04
      Decay  1.007E+00  3.25E-01
      Fid  Amplitude  Uncertainty
      1  7.997E+02  6.58E+01

# Name      Order      Type      Parameter      Uncertainty
4 (CP) Triplet      (3,1) 0      Freq  1.2913983E+00  6.07E-05
      Decay  8.577E-01  3.73E-02
      Jp    6.9822E+00  2.31E-02
      Fid  Amplitude  Uncertainty
      1  9.381E+03  9.97E+01

# Name      Order      Type      Parameter      Uncertainty
5 (CP) Quartet      (4,1) 0      Freq  4.8648472E+00  8.50E-05
      Decay  8.879E-01  5.96E-02
      Jp    6.9795E+00  3.26E-02
      Fid  Amplitude  Uncertainty
      1  6.345E+03  1.10E+02

Base 10 Log Of The Probability of 4 Resonances =-3.17799725E+03

          Probability For The Model
----- Model ---- Probability ---- Prob/I --- Prob/F ----- Ended -----
  1 Resonances  3.149126-312      431.0      431.1 Fri May 10 10:41:53 1996
  2 Resonances  3.274689E-50      626.3      707.6 Fri May 10 10:41:54 1996
  3 Resonances  1.000000E+00      764.3      768.3 Fri May 10 10:41:55 1996

Bayesian Analysis Ended Fri May 10 10:42:11 1996
And Took: 15 Sec. (0.25 Min.)

```

Figure 8.9: The top of the reports indicates the number of models and outputs the global parameters used in the analysis. The numbered entries are the individual models and the parameters estimates for those models. Finally the bottom shows the base 10 logarithm of the posterior probability for the models and how much the probability increased during the search.

Figure 8.10: Bayes Analyze Uncorrelated Output

#	Name	Order	Type	Parameter	Uncertainty
3	(UP) Triplet	(3,1) 0	Freq	1.291424E+00	1.28E-04
			Decay	8.688E-01	7.77E-02
			Jp	6.9828E+00	4.84E-02
	Fid	Cos Amplitude	Uncertainty	Sine Amplitude	Uncertainty
	1	4.826E+03	2.10E+02	-8.118E+03	2.10E+02

Figure 8.10: The output is different for uncorrelated resonance. In uncorrelated resonance models, there is a phase parameter for each resonance which, for computational reasons, is written as two amplitudes. If the sine amplitude is represented by  $S$  and the cosine amplitude by  $C$ , then the amplitude of the sinusoid  $A$  is related to  $S$  and  $C$  by  $A = \sqrt{C^2 + S^2}$ .

amplitude is also printed. These uncertainties are different for each Fid. The amplitudes shown in Fig. 8.9 are all for correlated phase models components so there is only a single amplitude. However, uncorrelated phase models have two amplitudes associated with them: a cosine amplitude and a sine amplitude. Under these conditions the output amplitudes have two entries, see Fig. 8.10. If you compare the frequency, decay rate constant and  $J$ -coupling constants you will find them to be essentially identical. The amplitude has been replaced by the corresponding cosine and sine amplitudes along with an estimate of how uncertain one is of the true value. In the summary reports these two amplitudes are reported as an amplitude and a phase. For more on the summary reports see Sections 8.5.8 and 8.5.9.

After the output associated with the parameter estimates comes the output from the model selection calculation. This output is in two forms, one is a logarithm of the posterior probability for the model. In Fig. 8.9 this is given by the following line:

```
Base 10 Log Of The Probability of 4 Resonances ==-3.17799725E+03
```

This is a logarithm of an unnormalized probability distribution; consequently, only differences between this number and the logarithm of the probability for some other model are meaningful. Roughly speaking a difference of one means that the new model is fitting one additional data value. Because the number of data can be large it is not unusual to see large difference in the probability for the models.

The second form of this output is the normalized probability for the model. This cumulative probability distribution consists of one entry for every model tested. Here are these lines reproduced from Fig. 8.9:

```

                          Probability For The Model
----- Model ----- Probability ----- Prob/I --- Prob/F ----- Ended -----
  1 Resonances  3.149126-312      431.0      431.1 Fri May 10 10:41:53 1996
  2 Resonances  3.274689E-50      626.3      707.6 Fri May 10 10:41:54 1996
  3 Resonances  1.000000E+00      764.3      768.3 Fri May 10 10:41:55 1996

```

They indicate several things about the models that were processed. First, three models were tested on this run of Bayes Analyze. If Bayes Analyze is run multiple times, all of the models tested are written into the probabilities file and a blank line separates the output from successive runs. Each model written into the output file gets a short description. This description is under the

labeled, “Model,” in fig. 8.9. The description is usually just the number of resonance components in the model. The model description is followed by the normalized posterior probability for the model. These are labeled “Probability.” When this probability distribution is normalized, one is essentially using probability theory to answer the following question: Given the one, two, and three exponentially decaying resonances models which of these three models best accounts for the data. In order to normalized the probability distribution, we had to tell probability theory to pick the best model from this set and only from this set. This is not equivalent to saying that the three exponential model is the best, only that it is the best of the three tested. Indeed the statement that some model is the “best” is indeterminate until the alternatives against which it is being tested are given.

In this example, the one resonance model has a probability that is essentially zero, the two resonance model has a probability 272 orders of magnitude higher; but, it is still 50 orders of magnitude lower than the probability for the three resonance model. These numbers are computed using a Student-*t* distribution. The logarithm of the Student-*t* distributions is multiplied by essentially the number of data values. A change of one order of magnitude in these probabilities means, at least very approximately, that the model has fit one more data value – so a change of 272 means that the two resonance model fits roughly 272 more data points than the one resonance model, etc. This rule – one point equal one order of magnitude – is only approximate and should be used only as a guide in ones thinking. Usually the changes from one model to the next are so large that little thought need go into which to accept. However, sometimes these probabilities can be very close, and when that happens one must carefully evaluate the different models to determine which is appropriate.

After the probabilities comes the initial (Prob/I) and the final (Prob/F) base 10 logarithm of the joint posterior probability for the parameters in the model. The joint posterior probability for the parameters is the probability density function used in optimizing the model parameters. The initial and final probabilities listed here are the initial and final probabilities from that optimization.

Finally, the last entry is the date and time the processing occurred. When Bayes Analyze begins a step it gets the date and time from the system clock. The numbers displayed are those times. Their primary function is to supply the user with one more indication of which model one is dealing with. When the user tests many different models it is possible for this set of probabilities to become fairly long. These dates and times along with the initial and final search probabilities can help determine which models one is dealing with.

The list of probabilities printed in this section are taken from the probabilities files. Bayes Analyze will never delete the probabilities files. If they are present, it will simply append the results of any new calculations to the end of these files. We maintain this information across multiple runs precisely so that one may test different models. To reset this list of probabilities one must clear the experiment, load new data, or manually delete the probabilities files.

After Bayes Analyze completes a run, it will compute the total computer time used and print this number at the end of the output file. This number is purely informational.

#### 8.5.4 The “bayes.probabilities.nnnn” File

The probabilities file is perhaps one of the simplest and most important files in the Bayesian Analysis package. These files are named “bayes.probabilities.nnnn” where the number of the first Fid analyzed in this file. An example of the probabilities file is shown in Fig. 8.11. This file is particularly simple in its format. It consists of a description of the model, i.e., “A Constant,” “1 Resonance,” etc., the

Figure 8.11: The bayes.proBABILITIES.nnnn File

Model Desc	Model Prob	Search/I	Search/F	Date-Time
1 Resonance	0.000	102.0	124.9	Mon May 20 08:54:44 1996
2 Resonances	40.823	169.4	174.9	Mon May 20 08:54:45 1996
3 Resonances	93.923	228.5	237.7	Mon May 20 08:54:45 1996
4 Resonances	165.324	298.0	318.8	Mon May 20 08:54:46 1996
5 Resonances	261.385	396.5	426.5	Mon May 20 08:54:47 1996
6 Resonances	443.048	536.2	622.7	Mon May 20 08:54:48 1996
7 Resonances	478.148	666.7	668.5	Mon May 20 08:54:49 1996
8 Resonances	515.355	713.7	716.9	Mon May 20 08:54:51 1996
9 Resonances	576.265	770.8	790.1	Mon May 20 08:54:52 1996
10 Resonances	576.079	797.5	798.3	Mon May 20 08:54:55 1996
1 Resonance	90.088	220.1	220.2	Mon May 20 08:55:09 1996
2 Resonances	156.099	286.7	296.1	Mon May 20 08:55:09 1996
3 Resonances	260.609	384.5	410.9	Mon May 20 08:55:10 1996
4 Resonances	450.966	531.9	614.9	Mon May 20 08:55:11 1996
5 Resonances	486.696	659.5	661.2	Mon May 20 08:55:12 1996
6 Resonances	524.166	706.5	709.6	Mon May 20 08:55:13 1996
7 Resonances	584.155	762.5	781.5	Mon May 20 08:55:14 1996
8 Resonances	583.867	788.8	789.5	Mon May 20 08:55:16 1996
2 Resonances	284.782	431.2	431.4	Mon May 20 08:55:30 1996
3 Resonances	546.734	624.3	707.2	Mon May 20 08:55:31 1996
4 Resonances	596.811	763.7	768.4	Mon May 20 08:55:32 1996
5 Resonances	596.499	775.7	776.4	Mon May 20 08:55:33 1996
2 Resonances	281.672	428.1	430.4	Mon May 20 08:56:02 1996
3 Resonances	542.561	624.6	705.9	Mon May 20 08:56:05 1996
4 Resonances	592.327	762.4	767.0	Mon May 20 08:56:08 1996
5 Resonances	592.005	774.4	775.1	Mon May 20 08:56:11 1996

Figure 8.11: Unlike the other files in the Bayesian Analysis package the bayes.proBABILITIES.nnnn file accumulates across multiple runs of Bayes Analyze. Indeed the only way to get rid of this file is either clear the experiment, load a new Fid, or issue a Unix “rm” on this file. The probabilities file contains the probability for the various models that were tested by Bayes Analyze. The first line in this figure, the header, is here only to label the columns; it is not normally present in this file. The file consists of an indication of the type of model tested, the base 10 logarithm of the probability for that model, the initial and final search probabilities and the date and time the model was tested. For more on this example see the text.

base 10 logarithm of the probability for the model, the beginning and ending search probability (if applicable), and the date and time the model was analyzed.

To compare several different models using probability theory, one computes the probability for each model. The model with highest probability is the best. When Bayes Analyze computes the probability for a model, that probability is appended to the probabilities file. Because information is appended to this file it accumulates across multiple runs of Bayes Analyze and so it may be used to compare many different models to see which model is best. To illustrate how one can use this file to do model selection see Fig. 8.11. This figure is an example of the probabilities file for four different runs of Bayes Analyze. Note that Bayes Analyze has separated these four different runs with a blank. In this figure, we have changed the normalization so that the logarithm of the probability for the one resonance model is zero. The probabilities for the models are stored as unnormalized logarithm, so any constant may be added or subtracted from every value in the file without changing the normalized probabilities.

The probabilities file illustrated in Fig 8.11 are from an analysis of a H1 Fid of ethyl ether. Part of the spectrum of the Fid is shown in Fig 8.1. Bayes Analyze was run on this data four different times using different initial models to see which model was the best description of this data. In the first of the four runs Bayes Analyze was allowed to find correlated resonances in its automatic mode. The maximum resonances was set to 20 resonances, many more than the 9 present. In that run the program found 9 resonances, three associated with the triplet, four with the quartet, and the two nuisance resonances. When Bayes Analyze tried to add a tenth resonance the probability for the model went down thus indicating that this last candidate resonance was probably noise. This run is represented in Fig. 8.11 by the first 10 lines. Notice that the probability for the model starts at zero and then rises steadily until it reaches a maximum of 576.265 and then it decreases slightly.

The first test indicated that a 9 resonance model was the best. However, we know that there is a triplet and a quartet in this data. In the next test, we marked the triplet, and let Bayes Analyze add resonances until the probability for the model decreased. Bayes Analyze added resonances until it reaches an 8 resonance model and then the logarithm of the model probability decreases; so Bayes Analyze has indicated that a 7 resonance model (1 triplet and 6 singlets) is the best model tested so far. The logarithm of the probability for 9 singlets was 576.265; while the logarithm of the probability for the 6 singlet plus one triplet is 584.155, an increase of 7.89 orders of magnitude. So probability theory prefers this model to the 9 singlet model, as it should.

In the third test, we marked the triplet and quartet and then allowed Bayes Analyze to add resonances in its automatic mode. Bayes Analyze tested three additional resonances and on the third resonance the logarithm of the probability for the model decreased. So the model preferred in this test was a triplet, a quartet and two singlets for a total of 9 peaks. The base 10 logarithm of the probability for this model is 596.811. This should be compared to the best model in the previous two test: 584.15 and 576.265. Note that this model is 12.656 orders of magnitude more probable than the best previous model.

Just for illustrative purposes we ran one additional test. We turned on third order shimming and allowed Bayes Analyze to run using the triplet and quartet model as a starting point. In this case Bayes Analyze added three resonances and on the third singlet the probability decreased. But the probability for this shimming model has decreased from 596.811 to 592.327, a decrease of 4.484 orders of magnitude – so the shimming model does not help. The shimming model did nothing to improve the fit and the prior probability for this model has decreased enough to rule this model out.

The bayes.probabilities.mmm files may be safely deleted anytime, even while Bayes Analyze is running. If a probabilities file is not found, Bayes Analyze creates it when it begins analyzing a

new model. There are three ways to reset the probabilities file. Here “reset” means removing the content of the file. Resetting the file may be desirable when you wish to compare a given set of models. First, loading a new free induction decay will clear all of the files out of the current WorkDir. Second, hitting the “Reset” button on the Bayes Analyze interface will remove all of the current Bayes Analyze output files from the current WorkDir, including the probabilities file. Finally, you can manually delete the file by navigating to Bayes/WorkDir/BayesAnalyzesFiles and deleting the bayes.probabilities.mnnn files.

### 8.5.5 The “bayes.log.mnnn” File

A record of what the search algorithm is doing is written to the bayes.log.mnnn file in the format shown in Fig. 8.12. Each search, with the possible exception of the first one, begins with a line

Figure 8.12: The bayes.log.mnnn File

```

Base 10 Log Evidence for The Next Resonance is: 0.3
----- 5 Resonance Model -----
Log Prob      Relax      L      Parameter      Parameter      Parameter
776.5         I 1.000  0.010  0.1167(f)      0.9950(d)      -57.9401(Ph)
              I 1.000  0.010  1.2914(f)      0.8575(d)      6.9822(Jp)
              I 1.000  0.010  0.0083(T0)     4.8648(f)      0.8882(d)
              I 1.000  0.010  6.9795(Jp)     5.6478(f)      3.2275(d)
              I 1.000  0.010  4.4656(f)      1.0876(d)
777.1         A 1.000  0.000  0.1167(f)      0.9822(d)      -57.8867(Ph)
              A 1.000  0.000  1.2914(f)      0.8571(d)      6.9821(Jp)
              A 1.000  0.000  0.0088(T0)     4.8648(f)      0.8869(d)
              A 1.000  0.000  6.9794(Jp)     5.6477(f)      3.2289(d)
              A 1.000  0.000  4.4648(f)      2.5345(d)
777.3         Z 1.000  0.000  0.1167(f)      0.9806(d)      -57.8711(Ph)
              Z 1.000  0.000  1.2914(f)      0.8569(d)      6.9821(Jp)
              Z 1.000  0.000  0.0090(T0)     4.8648(f)      0.8868(d)
              Z 1.000  0.000  6.9795(Jp)     5.6477(f)      3.2284(d)
              Z 1.000  0.000  4.4645(f)      3.5147(d)

Base 10 Log Of The Probability of 5 Resonances =-3.17473576E+03

              Probability For The Model
----- Model ---- Probability ---- Prob/I --- Prob/F ----- Ended -----
  A Constant    4.358829-431      12.7           Wed May 1 13:14:23 1996
  2 Resonances  5.610831-313      123.8          431.4 Wed May 1 13:14:25 1996
  3 Resonances  5.051696E-51      624.3          707.2 Wed May 1 13:14:25 1996
  4 Resonances  6.026667E-01      763.7          768.4 Wed May 1 13:14:26 1996
  5 Resonances  2.940744E-01      775.7          776.4 Wed May 1 13:14:28 1996

Bayesian Analysis Ended Fri May 3 14:23:44 1996
And Took: 1 Sec. (0.02 Min.)

```

Figure 8.5.5: The “bayes.log.mnnn” file is a history of what Bayes Analyze did during an analysis. This file generally starts with the base 10 logarithm of the odds ratio for another resonance in the data. Bayes Analyzes uses the frequency associated with that logarithm of the evidence to postulate a new model. It optimizes that model. Each step shown here is a single step in the Levenberg-Marquardt algorithm. The Bayes Analyze program and accepts or rejects the model depending on this posterior probability.

containing the logarithm of the evidence in favor of another resonance and ends with a line containing the logarithm of the posterior probability for the current model. Both these log probabilities are illustrated in Fig. 8.12, the evidence in favor of a resonance the first line in Fig. 8.12 and the logarithm of the posterior probability for the model is the line beginning “Bayes 10 Log Of The Probability of”. We indicated that the first search may not contain the the logarithm of the evidence in favor of another resonance, that’s because the first search may be done on a user defined model and in this case the evidence calculation is not performed.

If the evidence in favor of another resonance is greater than zero, i.e., odds greater than 1:1, Bayes Analyze will create a model containing the frequency with the highest evidence and perform a search for the peak value of the joint posterior for the parameters in this model. As Bayes Analyze is searching, it write out a set of lines showing the current status of the search. In Fig. 8.12 these search status lines are essentially the center of the figure. The Levenberg-Marquardt search algorithm has several parameters associated with it, these are: the logarithm of the joint probability for the parameters, a relaxation parameter and a Levenberg-Marquardt parameter. These parameters are identified by the headings: “Log Prob”, “Relax”, and ‘L’ shown in Fig. 8.12. The single character in front of the relaxation parameter indicates whether the search step is the **I**nitial step, a step in which the posterior probability increased and the step was **A**ccepted, or whether or not the posterior probability decreased and the step was therefore **R**ejected. Finally, **Z**, indicate the covariance matrix is being computed.

The remaining three fields in Fig. 8.12, under the “Parameter” label, are the parameters that appear in the model. There is no particular order to these parameters except to note that resonances are listed in increasing resonance order. However, the resonances are ordered before a search begins – not while it is going on, so it is possible for the resonance frequencies to become disordered while the search is in progress. Each parameter is listed to four decimal places, although, the number of decimal places varies depending on the size of the parameter. The parameter is followed by a short description in parentheses – See Table 8.3 for a list of these descriptions and their meaning. This short description is only used in the log file.

After Bayes Analyze locates the maximum of the joint posterior probability for the parameters, the approximate covariances matrix is computed with the Levenberg-Marquardt parameter set to zero. This covariance matrix is used to compute a Gaussian approximation of the joint posterior probability for the parameters. Finally, by integrating this joint posterior probability for the parameters, one is able to compute an approximation to the posterior probability for the model. The base 10 logarithm of the posterior probability for the model is printed at the end of each search. Here is an example of this line:

```
Base 10 Log Of The Probability of 5 Resonances ==-3.17473576E+03
```

Because it is the logarithm of the posterior probability for the model that is printed, it is only differences between between the logarithm of the model probabilities that are meaningful. For example if the previous model probability was -128 and the current model has probability of -100, then this new model is  $10^{28}$  times more probably than the previous model.

After printing out the base 10 logarithm of the probability for the model, the output files are updated and the calculations, starting with the signal detection calculation, are repeated. The log file contains a record of what occurs as Bayes Analyze optimizes and test each model. These calculations continue until one of three things happen: the logarithm of the posterior probability for the model decreases, the signal detection routine failed to find a candidate resonance, or the maximum number of new resonances has been reached. After completing the calculation on a given

Table 8.3: Bayes Analyze Short Descriptions

Short Desc.	Full Description	Units
(Ph)	The center phase	Degrees
(T0)	The starting time of the fid	Points
(f)	The Resonance frequency	Hertz or PPM
(d)	The decay rate constant	Hertz
(Jp)	The primary $J$ -coupling constant	Hertz
(Js)	The secondary $J$ coupling constant	Hertz
(Sd)	Lorentzian spacing in lineshape expansion	Hertz
(1)	The first relative shim amplitude	No units
(2)	The second relative shim amplitude	No units
⋮		
(7)	The seventh shim amplitude	No units

Table 8.3: Bayes Analyze prints the parameters along with a short description of them to the bayes.log.mnmn file. These descriptions are shown here along with an explanation and the units used.

model, Bayes Analyze prints the the normalized posterior probability for the models to the log file. This part of the printout is shown at the bottom of Fig. 8.12 and begins with header reading “Probability For The Model” This list of model probabilities is cumulative and accumulates across multiple runs of the Bayes Analyze program. Different runs are separated by a blank line.

In Fig. 8.12 the lines that make up the normalized probability for the model,

```

                Probability For The Model
----- Model ----- Probability ----- Prob/I --- Prob/F ----- Ended -----
  A Constant    4.358829-431      12.7           Wed May  1 13:14:23 1996
  2 Resonances  5.610831-313      123.8          431.4 Wed May  1 13:14:25 1996
  3 Resonances  5.051696E-51      624.3          707.2 Wed May  1 13:14:25 1996
  4 Resonances  6.026667E-01      763.7          768.4 Wed May  1 13:14:26 1996
  5 Resonances  2.940744E-01      775.7          776.4 Wed May  1 13:14:28 1996

```

consists of a description of the model, the normalized probability for the model, the logarithm of the joint posterior probability for the parameters at the beginning of the search, under the heading “Prob/I,” followed by the logarithm of the joint posterior probability for the parameters at the completion of the search under the heading “Prob/F” and finally the time this particular model was run. If constant models are present they are shown at the top of the entries for a given run of Bayes Analyze.

After this table of model probabilities is printed, Bayes Analyze determines the amount of computer time it took to run this job and prints this information along with the data and time the job started and ended. After printing this last entry, Bayes Analyze terminates.

Figure 8.13: The bayes.status.nnnn File

```

Bayesian Analysis started: Tue May 14 13:11:01 1996
Status last updated: Fri May 17 10:27:21 1996
Current Fids: 15 Through 25
Status: Analysis completed

```

The “bayes.status.nnnn” file indicates the current status of a run. This file is also written using the name Bayes.accepted and is displayed when the “Get Job” button is pressed if the job has not completed.

### 8.5.6 The “bayes.status.nnnn” and “bayes.accepted.nnnn” Files

The status file is a three or four line file that is written by Bayes Analyze and Bayes Model. The accepted file is just the current version of the status file. Both Bayes Model and Bayes Analyze will write and or update the status and the accepted files. Bayes Model places the number of the last Fid modeled in the status and accepted files, while Bayes Analyze continuously continuously updates both the status and accepted files with the current status of the Bayes Analyze run. An example of this file is shown in Fig. 8.13. The first line of the file is the time the analysis started. The second is the last time this file was updated. Optionally the free induction decays being processed is given on the third line. This line is present when multiple Fids are being processed in blocks. The last line indicates what action happen at the time of the last update. In the example shown, this line indicates that the run had completed. The status file is also used to communicate setup problems, and program errors. A complete listing of all of the messages that may be written to the status file is given in subsection 8.6. In this section, informational messages are numbered 1-8; setup messages are numbered 9-29; and errors messages are numbered 30-49. The informational messages are written into the status file for no other purpose than to allow you to determine the current status of a run. The status display contains the status file as part of that display. The informational messages might be as simple as indicating that the input data is being read (message number 1) or they might indicate that the analysis is completed as message 7 indicates. Message numbers 9-29 are setup errors. These indicate that the Bayes Analyze detected a problem in the setup. They include such things as: not finding the input free induction decay (message 15), exceeding to the capabilities of the programs (message 27), and a number of others. All of these messages require one to correct one or more problems in the parameter file. The remaining messages, messages 30-49, are program errors in one form or another. These message consists of the name of the routine in which the error occurred and a message that indicates you should contact *me*. In almost all of these cases these message could not occur without something being seriously wrong with the program. Indeed most of these messages can occur only while the program is being tested. However, there are two messages, message number 37 and 38, that are not terribly serious, and when they occur are probably related to how the analysis was setup.

Message 37 occurs when a matrix inversion routine detects a singular matrix. But singular matrices can only occur when one or more of the model functions are nearly identical and even then this is rare because the prior probabilities used in the calculations are suppose to prevent this. You should check to make sure that the analysis is makes physical sense. For example, did you mark a peak and then Bayes Analyze tried to put a second peak at nearly the same location. This type of problem can occur whenever you mark a peak too far from the “true” location of the resonance.

Figure 8.14: The bayes.model.nnnn File

```

! # Name                Order      Frequency      Init Value
  3 (CP) Singlet        (1,1) 0      24.305874354538 24.172984999931

!                               Fid          Amplitude
                               1           76499.464296592
                               2           75736.815735717
                               3           75008.870579233
                               4           74424.926696927
                               5           73761.523349848
                               6           73730.846586329
                               7           72942.021502301
                               8           72735.215054879
                               9           72103.722022270
                              10          72022.172495999

```

Figure 8.14: The bayes.model.nnnn file is a modified version of the bayes.params file. In the bayes.model.nnnn file each model is followed by the amplitudes, one amplitude for each data set. Otherwise, the format of the Bayes Model and Bayes Params file are the same.

It can also occur when you mark a peak thinking it is a single line and it is really multiple lines. The solution to this type of problem is to remove the marked peaks and allow Bayes Analyze to put the peak in by itself. There are other reasons that this problem can occur, such as when all of the constant models are used and Bayes Analyze attempts to model a baseline artifact. Baseline artifacts are modeled with vary rapid decaying sinusoids, and so are very nearly the same as the first point models – so they can be come redundant. This can cause the matrix inversion to fail. The pattern should be clear: this problem occurs when several model components are nearly identical. The solution is to remove the redundant models and rerun the analysis. If the problem persists and you can make no progress on solving the problem, contact [me](#). The other error, message number 38, is essentially the same as the previous one. In this case the matrix inversion routine failed but did not detect the failure. One or more of the model functions are redundant and that redundancy is causing either convergence or roundoff problems in the matrix inversion routine. The solution is again to remove the redundant vectors and try the analysis again. The remaining errors in Subsection 8.6 are true program errors and should never occur. Indeed if they do occur, you should save the WorkDir in which the problem occurred (making sure you get the Fid and not a pointer to it) and then contact [me](#).

### 8.5.7 The “bayes.model.nnnn” File

The model file is a modified version of the parameter file. In addition to all of the information contained in the parameter file, the model file also contains the estimated amplitudes of each model component. An example of the amplitude information is shown in Fig. 8.14. The resonance shown is the same triplet shown in Fig. 8.6. The amplitude section has a comment header, followed by the estimated amplitude of the resonance. This line is repeated for each free induction decay processed. Note that this is the output for a correlated phase model. Some models, uncorrelated phase resonances, have multiple amplitudes per resonance. Fig. 8.15 is an example of the output amplitude for an uncorrelated resonance. The uncorrelated resonance has both a cosine and a sine

Figure 8.15: The bayes.model.nnnn File Uncorrelated Resonances

```
!
          Fid      Cos Amplitude  Sine Amplitude
          1      4848.0043462323 -8028.116298822
```

Figure 8.15: Uncorrelated resonance models have two amplitudes, effective an amplitude and a phase. Otherwise the format of the an uncorrelated resonances is exactly the same as for a correlated resonance model.

Figure 8.16: Bayes Analyze Summary Header

```
Summary1: /home/akgrp/glb/vnmrsys/exp3/bayes.output.0001
Date/time: Fri May 10 10:42:13 1996
Directory: /home/akgrp/glb/vnmrsys/exp3
File:      "/home/akgrp/glb/data/p13.fid"
Text File: This is test data
```

Figure 8.16: The summary report header contains a few items particular the the analysis that you are running. In particular it contains the name of the output file being summarized, the data and time, the analysis directory and the text file from the Fid. For more on this report, see the text.

amplitude. Together these specify the amplitude and phase of the resonance.

### 8.5.8 The “bayes.summary1.nnnn” File

The “bayes.summary1.nnnn” file is produced when the **Best** button is activated. This button calls a program name Bayes Summary1. In the process of analyzing a Fid, Bayes Analyze develops a hierarchy of models. These models usually begin with one resonance, followed by two, etc., until either the signal detection algorithm fails to find evidence for an additional signal or the probability for the model goes down. The output file written by Bayes Analyze contains all of this information and can get fairly long. The purpose of the summary report is to take the information in the output files and to print out only that part of it that is most important to the analysis. In particular it extracts that part of the output file that corresponds to the model having maximum probability. In this section we describe the output from Bayes Summary1 and point out where the information has been reformatted.

First, the preceding discussion indicates that the summary report is built out of the output file; this is not totally correct. The summary report is built out of pieces of the output, model, probabilities, text, and procar files. When Bayes Summary1 is run the first thing the program does is to determine how many output files there are in the experiment. Each of these output files are summarized one at time – so if there are 10 output files, 10 different summary reports are written.

Much of the summary report is identical to other outputs and we will not repeat those descriptions here. We will simply reference you to the correct descriptions. The summary output starts with a header that is unique to the summary report. The header is shown in Fig. 8.16. This header is made up of the name of the output file that was summarized, the date the summary was done, the directory in which the output files were located, the name of the Fid that was processed, and last the contents of the text file. In building this header Bayes Summary1 had to process both the text file and the procar file. However, this program also gathers information from the model file, the

Figure 8.17: The Summary2 (Best Summary)

```

Summary2: ./bayes.output.0001
Date/time: Fri Sep 6 08:57:32 1996
Directory: .
File: "/home/jagrp/glb/vnmrsys/data/ethyl.ether"
Text File: This is test data

      Freq      Error      Decay      Error  Amplitude  Error  Type Resonance
      ppm      ppm      Hertz      Hertz  arbitrary
0.116750  1.10E-04  0.604000  0.110000  648.0000  27.00000 (CP) Singlet
1.291439  7.90E-05  0.792000  0.023000  9033.000  47.00000 (CP) Triplet (See 1)
4.864975  2.50E-05  0.772000  0.026000  5926.000  51.00000 (CP) Quartet (See 2)
5.648083  6.30E-05  2.991000  0.057000  3999.000  50.00000 (CP) Singlet

No.      Jp      Err      Sp      Err
      Hertz  Hertz  Hertz  Hertz
  1  6.978000  0.008100
  2  6.972500  0.009600

Bayesian Analysis Ended Thu Dec 1 15:35:19 1994
And Took: 0 Sec. (0.00 Min.)

```

Figure 8.17: The output from the Summary2 program is condensed listing of the frequencies, decay rate constants, amplitudes, and the uncertainties in these parameters.

output file, and the probabilities file. These files are used in building the main body of the report. This header is followed by the configuration parameters and these have already been explained – see Section 8.5.1 for a description of these parameters.

The remaining part of the summary report is essentially identical to the output report and we will not repeat it here. There is one difference that is worth noting, in the summary report uncorrelated resonances have their amplitudes reported as an amplitude and a phase, this is different from the output report where there are listed as a cosine and sine amplitude. This difference is mostly for convenience, sometimes the amplitude of an uncorrelated resonances is needed.

### 8.5.9 The “bayes.summary2.nnnn” File

The “bayes.summary2.nnnn” file is produced when the **Freq** button is activated the Bayes Analyze interface. This button calls a program name Bayes Summary2. Like the first summary report, this report is only on the section of the output file most important to the user. Because of its extremely compressed format this report can only be run on non-arrayed free induction decays. The report is essentially a column listing of the estimated frequencies, decay rate constants, the amplitudes and the uncertainties in these parameters. The report can be run on either correlated or uncorrelated phase models but when it is run on uncorrelated phase models the sine and cosine amplitudes are reported as the square root of the sum of the squares of these amplitudes. This quantity is the amplitude of the resonance.

An example of this report is shown in Fig. 8.17. Note that the first few lines of this report are essentially the same as in the Summary1 report. The main difference is in the detailed output. In this report the detailed output is essentially a column listing of the most probable model in the output

report. In this column listing the frequency, decay rate constant, amplitude, and the estimated uncertainty in our knowledge of these parameters are given. Additionally, the type of resonance and the resonance name are given. When the resonance is a multiplet the coupling constants are given at the end of the report. When no multiplets are present this section of the report does not appear. When uncorrelated resonances models are encountered the sine and cosine amplitudes are reported in the form of a total amplitude for the resonances. The phase of these resonances are not reported. If this information is needed it is reported in the Summary1 report. Last, the resonances are reported in the units that were in use at the time the analysis was run.

### 8.5.10 The “bayes.summary3.nnnn” File

The summary3 file and report are generated whenever a regions file is present in the current Working Directory. The regions/summary3 report takes as its input a “bayes.regions” input file. This file is produced automatically whenever the regions button is activated. It contains a series of lines, each line containing a region of the spectrum to be summarized. These regions are written into the file as low-high frequency pairs. As a user you set them by displaying the spectrum, and activating the “Options/set regions” button.

The regions/summary3 report contains the standard header, the regions information, followed by the probabilities information. Figure 8.18 shows a copy of the regions report. First, the actual regions are shown. For this report, these regions were specified by “Options/Set Regions” and when they are defined the regions report is automatically run when Bayes Analyze is run. Each region contains the total intensity in that region. This information is displayed in two forms, as a percent of intensity and as the total intensity. Additionally, each of these regions contain an estimate of the uncertainty in the intensity estimates.

## 8.6 Bayes Analyze Error Messages

This appendix contains a complete listing of all of the messages that Bayes Analyze and Bayes Model write to the status file. In the following list we have numbered the messages, the message is in bold, and this is followed by an explanation of the message. These messages are divided into three general categories: informational, setup errors, and program errors. The categories are not organized in any particular order. There are some additional comments about each category at the end of each category.

When Bayes Analyze is running in either its automatic or slave mode the status file is updated with an information message whenever the program changes from one general function to another. The informational messages follow:

1. **Reading input data** – The input free induction decays are being read by Bayes Analyze.
2. **Completed Baseline Artifact Model** – The baseline model has been processed. Bayes Analyze will attempt to model a baseline artifact whenever the baseline routines are activated in the parameter file. These routines can only be activated manually at present.
3. **Probability For A Constant Completed** – Computing the probability for the constant model is now completed.

Figure 8.18: The Summary3 Report

```

----- Regions -----
#      From      To
1      2.47      2.73
2      2.99      3.18
3      3.18      3.33
4      3.33      3.46
5      3.46      3.62
6      3.66      3.86
7      3.93      4.28
8      5.21      5.64

Fid:  1 of  1 Total Intensity In The Region
Region 1      Err      Region 2      Err      Region 3      Err
3.3930E+03  2.55E+02  3.9720E+03  3.00E+02  3.8620E+03  3.37E+02

Region 4      Err      Region 5      Err      Region 6      Err
4.8270E+03  2.98E+02  5.5800E+03  3.67E+02  4.2380E+03  3.52E+02

Region 7      Err      Region 8      Err
6.8470E+03  3.38E+02  5.9030E+03  3.39E+02

Fid:  1 of  1 Percent of the Intensity In The Region
Region 1      Region 2      Region 3      Region 4      Region 5      Region 6
8.79 0.66  10.28 0.78  10.00 0.87  12.50 0.77  14.45 0.95  10.97 0.91

Region 7      Region 8
17.73 0.88  15.28 0.88

```

Figure 8.18: The Regions/Summary3 Report is output by the Bayes Analyze package whenever regions are defined. Briefly, this report is the total resonance intensity in each of the defined rations.

4. **Performing  $n$  Resonance Search** – Bayes Analyze is optimizing the parameters for an  $n$  resonance model where  $n$  is the number of resonances components in the model.
5. **Completing  $n$  Resonance Model** – The maximum of the joint posterior probability for the  $n$  resonance model has been located and Bayes Analyze is now writing the output files and computing the probability for the model.
6. **Completed  $n$  Resonance** – Bayes Analyze has finished the  $n$  resonance model but has not yet started the  $n + 1$  model.
7. **Analysis completed** – Bayes Analyze has completed running in both the slave and automatic mode. You may now print, model, or modify the analysis in anyway deemed appropriate.
8. **Fid Number  $n$  Modeled** – The analysis has been completed.

The proceeding messages are informational and do not indicate any problem. They are written to the “`bayes.status.nnnn`” file to indicate the current status of the analysis. However, the following messages indicate some type of a problem. If you receive one of these messages you must take action to correct the problem.

9. **Configuration error - Did not recognize (text)** – There is an error in the configuration parameters in the parameter file. The word “(text)” is replaced by the text that was not recognized.
10. **Memory Allocation Failed – Rerun Later** – Both Bayes Analyze and Bayes Model dynamically allocates virtual memory for the data and for the models. If there is not enough memory available you will receive this message. If you have sufficient virtual memory, you can try again later. If you do not have enough virtual memory to run the program you can increase the swap space and then rerun the analysis. Alternately, you can try analyzing fewer free induction decays. Along the same line, you can try reducing the amount of data analyzed by setting a signal region. Reducing either of these reduces the amount of virtual memory requested.
11. **Fid File Not Found – Correct Model File** – The input Fid file was not found. Bayes Model is looking for the Fid specified in the model file. Either that Fid has been deleted, the model file is wrong, or Bayes Model is being run on a machine that does not have access to the Fid.
12. **Initial Model In Error - Fix** – When Bayes Analyze read the input parameter file it detected a problem in the file. If you created the file manually, you need to correct the problem. Before Bayes Analyze proceeds to process the input model, Bayes Analyze will call the routine that computes the prior probability for all of the parameters. If this model is valid, in the sense that all of the parameters are physically reasonable, the routine that evaluates the prior probability will return a zero status. If the parameters are not physically reasonable, the routine will set a nonzero status and this is what happened. The types of problems that can cause this are: negative decay rate constants, the secondary coupling constant is greater than the primary coupling constant, one of the coupling constants or shimming amplitudes is negative. The number of things that can be wrong here is extensive, but it will be some type of logical problem. Getting an input field in the wrong place could also cause this problem.

13. **The input file is in an old format Vers('nnn')**– The input file either contains an bad file version or this is a file version that is no longer recognized by the Bayesian Analysis package.
14. **ASCII Fid File Error** – Either the input ASCII Fid file was not found, or it contained too few total data values.
15. **The Input Fid File Was Not Found** – The open for the input Fid file failed. The input file name is probably incorrect. This error can occur in either the Bayes Model or Bayes Analyze.
16. **The ASCII Parm File Was Not Found** – Either the file name in the parameters file is in error, or the input ASCII parameter file is not present. Either way Bayes Analyze could not find the file.
17. **The ASCII Parm File Is In Error** – The ASCII input parameter file is in error. Check the file and make the appropriate corrections. See Section 8.5.1 for more on this file.
18. **Input Procpa File Not Found – See log file** – The open failed on the procpa file. Either the name in the parameter or model file is incorrect, or the file has been deleted. The log file contains the name of the file the program was trying to read.
19. **Input Fid File Was Not Found – See log File** – The open failed for the Varian Fid file. Either the name in the parameter or model file is incorrect, or the file has been deleted. The log file contains the name of the file the program was trying to read.
20. **Adding Freq Would Cause Max Freq To Be Exceeded** – The requested analysis has exceeded the capabilities set at the time Bayes Analyze was compiled. You can determine the limits by running the command “bayes analyze limits” This will cause Bayes Analyze to print its hard coded limits.
21. **Adding Freq Would Cause MaxRes To Be Exceeded** – Same as above.
22. **Adding Freq Would Cause Rmax To Be Exceeded** – Same as above.
23. **Adding Freq Would Cause MGMAX To Be Exceeded** Same as above.
24. **Adding Resonance Would Cause RMAX To Be Exceeded** – Same as above.
25. **No of Fids exceeded max – see log file** – The requested number of Fids exceeded the maximum allowed for this version of Bayes Analyze. Either reduce the number of Fids requested or contact Varian for a version of the programs with larger limits. The log file contains the current limits.
26. **No of Fids exceeded actual – see log file** – The requested number of Fids exceeded the actual number in the array.
27. **Total data values exceeded max – see log file** – The requested number of data values exceeds the maximum allowed for this version of Bayes Analyze. You can determine the maximum allowed for the parameters by running the Unix command “bayes analyze limits” This will cause Bayes Analyze to print its hard coded limits.

28. **Total data values requested exceeded actual – see log file** – The number of data values that are suppose to be used is greater than the actual number of data values in the Fid.
29. **Requested Fid Is Not In Model File** – The input parameter file does not contain the parameters for the Fid you wanted to model.

Except for messages 37 and 38, the following list of errors are true program errors. If you receive one of them it means that Bayes Analyze has been pretty thoroughly messed up and you should contact Varian immediately. To illustrate what is meant by a “true” program error, consider message number 39. The routine SETMODEL has the function of taking a particular model and generating the time domain model. The model to generate is requested by the calling routine. The SETMODEL routine consists of a series of IF statements that determine the type of model to be processed and then calls the appropriate model dependent code. Message number 39 can only be obtained if these IF statements fail to recognize the model, the only way the model could be unrecognized is if we forgot to modify this routine when a new model was added; so this message represents a very significant problem in Bayes Analyze that should never be seen outside of testing. With the exception of messages 37 and 38 the other messages have similar implications. Here is the list of program errors:

30. **CONSTGIJ – Program Error Contact Varian**
31. **DEVZERO – Program Error Contact Varian**
32. **OUTMODEL – Program Error Contact Varian**
33. **PMSG – Program Error Contact Varian**
34. **PUTMODEL – Program Error Contact Varian**
35. **SETGIJ – Program Error Contact Varian**
36. **TRANS – Program Error Contact Varian**
37. **ALOGP – Possible Program Error Investigate Further**
38. **ALOGP2 – Possible Program Error Investigate Further**
39. **SETMODEL – Program Error Contact Varian**
40. **SCCETRANS – Program Error Contact Varian**
41. **SCUETRANS – Program Error Contact Varian**
42. **GETMODEL – Program Error Contact Varian**
43. **APPEND\_TEXT – Program Error Contact Varian**
44. **READHDR – Program Error Contact Varian**
45. **WRITEDATA – Program Error Contact Varian**
46. **WRITEMODEL – Program Error Contact Varian**

47. **WRITERESID** – Program Error Contact Varian

48. **COPY\_PROCPAR** – Program Error Contact Varian

49. **LIST\_MODEL** – Program Error Contact Varian

The two exceptions that we were discussing are messages numbers 37 and 38. These two messages are program errors, but they may be caused by the model. For example if the estimated frequencies become redundant, then the covariance matrix will be singular and these errors are issued. Message number 37 indicates that this matrix is singular. When this matrix becomes singular it is almost always because one or more of the resonances have converged to the same resonance frequency. This behavior may occur under several circumstances: first it is possible that a peak was marked and this peak was so far from the value indicated by probability theory that Bayes Analyze added one or more additional resonances near the same location. These resonances are nearly identical and this can cause the error. The easiest thing to try under these conditions is to remove the offending resonances and allow Bayes Analyze to run in its automatic model. There is one other condition where this problem has been known to occur. NMR lines never decay in a perfectly exponential manner because the magnetic fields are never perfectly uniform. When the signal-to-noise is very high Bayes Analyze may try to fit so many resonances at nearly the same resonance frequency that the matrix it is trying to invert becomes singular. Unfortunately, in this case, there is very little that can be done. You can try turning on the shimming models and rerunning the analysis, but if the problem re-occurs then you may not be able to analyze this data.

So far we have been discussing error number 37, but error number 38 is almost the same and it is caused by essentially the same conditions. The previous error occurs when the matrix inversion routine detects that the matrix is singular. Here the inversion routine did not detect failure and it returned a zero status. After the inverse is computed, Bayes Analyze computes the projection of the data onto the model. What it found was that the projection of the data onto the model was greater than the projection of the data onto itself. This is simply not possible, and when it happens it means that the matrix inverse failed.



## Chapter 9

# Big Peak/Little Peak

The Big Peak/Little peak package is designed to analyze a single FID containing a solvent resonance, the big peak, and a few small resonances. The solvent resonance is treated as a nuisance signal and marginalized from the problem; the frequencies, decay rate constants and amplitudes of the small resonances are the primary output from this analysis. The interface to this package is shown in Fig. 9.1. To use this package, you must do the following:

**Select** the “Big Peak/Little Peak” package from the Package menu.

**Load** the spectroscopic FID you wish to analyze.

**Display** the trace you wish to analyze. Big Peak/Little peak analyzes a single trace and this trace must be displayed in the FID viewer. Consequently, you must select the trace you wish to analyze by displaying it.

**Use** a double cursor to bracket the location of the solvent and hit the “Solvent” button. This will result in the location of the solvent being bracketed by a double set of lines on the display. Additionally, the solvent frequency and decay rate constant will be displayed in the parameter list. Clicking on these parameters will cause their associated prior probabilities to be displayed and you can adjust these values if desired.

**Use** a double cursor to bracket the locations of each metabolite to be included in the model and hit the “Metabolite” button. When the metabolite button is activated the frequency and decay rate constant for this metabolite are added to the list of parameters and as with the solvent parameters clicking on a parameter will display the parameter prior and you can adjust these prior probabilities as needed.

**If** needed a right click on a parameter can be used to remove a frequency and decay rate pair.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

Figure 9.1: The Big Peak/Little Peak Interface

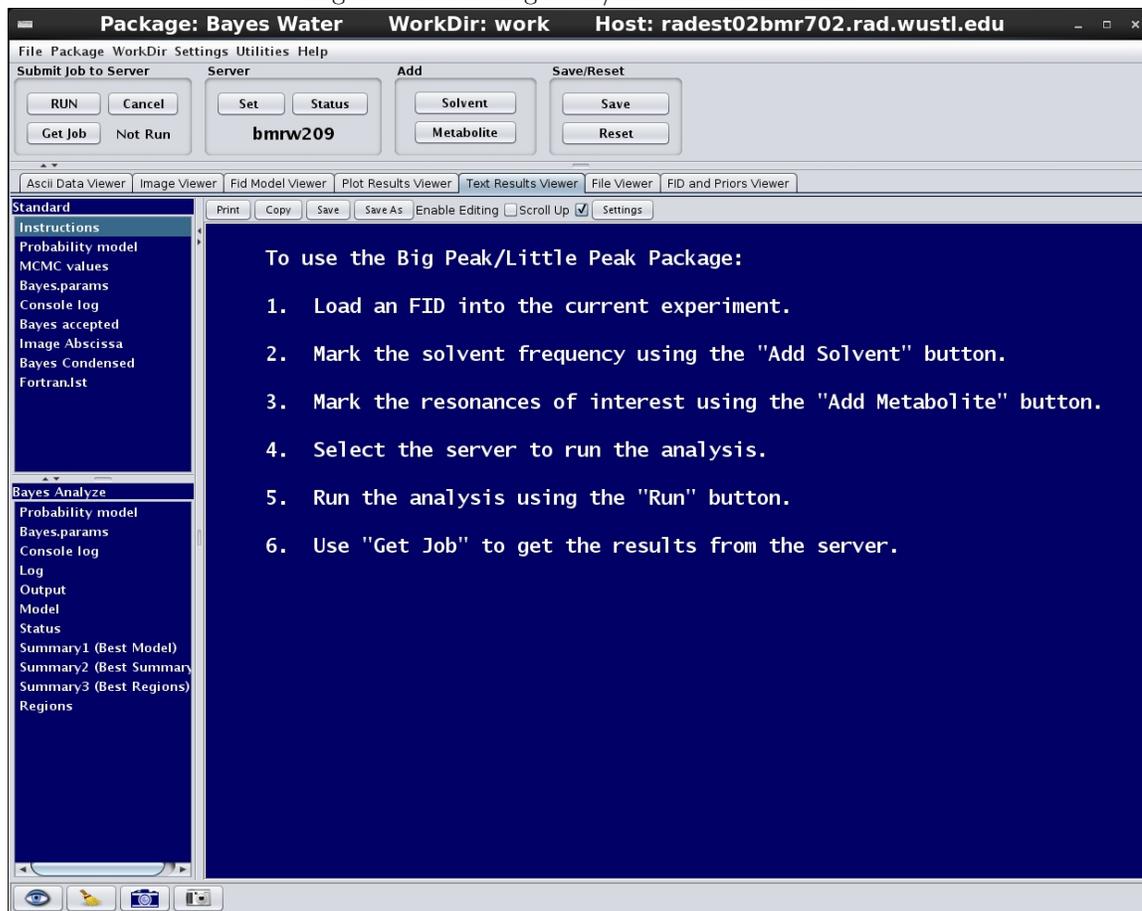


Figure 9.1: The Big Peak/Little Peak Interface This panel relevant parts of interface to the Big Peak/Little Peak package. To use the Big Peak/Little Peak package you must specify the frequency of the solvent, the big peak, and you must mark the locations of all of the small peaks you wish to analyze. The output from the program includes the estimated frequencies, decay rate constants, and amplitudes of these small resonances.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

**View** the results and the model using the Model Viewer.

## 9.1 The Bayesian Calculation

The Big Peak/Little Peak package analyzes a single Fid that contains a large solvent resonance or solvent suppression artifact and one or more smaller resonances of interest. In this problem we will assume these small resonances are of relatively low signal-to-noise and may be modeled as exponentially decaying sinusoids:

$$\text{Model of the Interesting Resonances} = \sum_{k=1}^m A_k \cos(2\pi f_k [t_i + t_0] + \theta) \exp\{-\alpha_k t_i\} \quad (9.1)$$

where  $m$  is the number of interesting resonances,  $A_k$  is the amplitude of the  $k$ th interesting resonance,  $f_k$  is its frequency,  $\alpha_k$  is its decay rate constant,  $t_0$  is a time offset and may be thought of as a frequency dependent phase correction, and  $\theta$  is the constant or zero order phase.

For most purposes this is the model used in the Bayes Analyze package when it is modeling singlets. Unfortunately, the solvent peak cannot be modeled this way because of its dynamic range. Exponentially decaying sinusoidal models work very well for low to moderate signal-to-noise levels. However, when the signal-to-noise levels become larger than a few hundred, this model becomes inadequate. Inadequate in the sense that the difference between the data and the best fit model, the residual, contains a systematic artifact. The discrete Fourier transform of this artifact is often much larger than peaks associated with the resonances of interest. Consequently, programs that fit exponentially decaying sinusoidal to solvent peaks put more and more resonances in the location of the solvent and often never see the resonances of interest. Indeed if you run the Bayes Analyze program on the test data set “Bayes.test.data/BayesWater\_test.fid” you will quickly discover that Bayes Analyze cannot really analyze this FID. It places multiple frequencies in the location of the solvent, and completely misses the small resonances on the right of the solvent. In order to fix this problem a better model of the solvent resonances must be used in the analysis.

Because we are using Bayesian probability theory and because we don't care about the solvent resonance we are free to model this resonance in any way that will fit it down to the noise level. The solvent model employed in this calculation is of an exponentially decaying sinusoid having an amplitude and phase that are slowly varying functions of time. If  $S(t_i)$  represents the complex solvent signal, then the model of the solvent is given by

$$S(t_i) = A(t_i) \exp\{2\pi i f_s t_i - \alpha_s t_i - i\phi(t_i)\} \quad (9.2)$$

where  $f_s$  and  $\alpha_s$  are the solvent frequency and decay rate constant and  $A(t_i)$  and  $\phi(t_i)$  are the unknown amplitude and phase modulation. Note that if the solvent frequency and decay rate constant are set to zero, then this model is just a complex trend.

This solvent model is a complex function and before it may be used in a Bayesian calculation it must be separated into its real and imaginary parts:

$$\begin{aligned} S(t_i) &= [A_c(t_i) \cos(2\pi f_s t_i) - A_s(t_i) \sin(2\pi f_s t_i)] \exp\{-\alpha_s t_i\} \\ &\quad - i[A_c(t_i) \sin(2\pi f_s t_i) + A_s(t_i) \cos(2\pi f_s t_i)] \exp\{-\alpha_s t_i\} \end{aligned} \quad (9.3)$$

where we have written the amplitude,  $A(t_i)$ , and phase,  $\phi(t_i)$ , as a cosine and sine amplitude,  $A_c(t_i)$  and  $A_s(t_i)$ .

The two functions  $A_c(t_i)$  and  $A_s(t_i)$  must have a few simple properties if they are to be physically meaningful. First, when the shimming is good or the signal-to-noise is low, the expansion order will go to 1, and the polynomials reduce to constants. Second, the solvent resonance does not usually deviate all that much from an exponentially decaying sinusoid, so we do not expect the functions  $A_c(t_i)$  and  $A_s(t_i)$  to be rapidly varying. Indeed if these two functions were rapidly varying we would reject their use because they would be able to represent the small resonances of interest. The simplest functions that have the desired properties are polynomial expansions:

$$A_c(t_i) = \sum_{j=1}^{n_c} B_j \mathcal{L}_j(t_i) \quad \text{and} \quad A_s(t_i) = \sum_{k=1}^{n_s} C_k \mathcal{L}_k(t_i) \quad (9.4)$$

where  $n_c$  and  $n_s$  are the number of the polynomials in the cosine and sine expansions respectively. The number of polynomials,  $n_c$  and  $n_s$ , and the amplitudes,  $B_i$  and  $C_i$ , are unknowns and must be inferred as part of the Bayesian calculations.

The polynomials  $\mathcal{L}_j(t_i)$  will be chosen so that they are orthogonal on the discretely sampled times:

$$\sum_{i=1}^N \mathcal{L}_j(t_i) \mathcal{L}_k(t_i) = \delta_{jk} \quad (9.5)$$

where  $N$  is the total number of complex data values, and  $\delta_{jk} = 0$  if  $j \neq k$  and it is equal to one if  $j = k$ . We make this choice for computational reasons because it helps stabilize the numerical computation.

Having specified the model for the small resonances of interest, Eq. (9.1) and the model for the solvent, Eq. (9.3), we are now in a position to relate the FID data to the model signal. The model for the real data will be written as

$$d_R(t_i) = M_R(t_i) + \text{Noise in the real data} \quad (9.6)$$

and similarly for the imaginary data

$$d_I(t_i) = M_I(t_i) + \text{Noise in the imaginary data.} \quad (9.7)$$

The model of the real part of the FID data,  $M_R(t_i)$ , will be written as

$$M_R(t_i) = \sum_{j=1}^{n_A} \mathcal{A}_j R_j(t_i) \quad (9.8)$$

and similarly the model for the quadrature, or imaginary, part of the FID will be written as

$$M_I(t_i) = \sum_{j=1}^{n_A} \mathcal{A}_j I_j(t_i) \quad (9.9)$$

where the total number of amplitudes,  $n_A$ , is given by

$$n_A = m + n_c + n_s. \quad (9.10)$$

In these equations we have relabeled the amplitudes,  $A_l$ ,  $B_j$  and  $C_k$ , as  $\mathcal{A}_j$ :

$$\mathcal{A} \equiv \{A_1, \dots, A_m, B_1, \dots, B_{n_c}, C_1, \dots, C_{n_s}\}. \quad (9.11)$$

The functions  $R_j(t_i)$  and  $I_j(t_i)$  are defined as

$$R_j(t_i) = \begin{cases} \cos(2\pi f_j[t_i + t_0] + \theta) \exp\{-\alpha_j t_i\} & \text{if } 1 \leq j \leq m \\ \mathcal{L}_{(j-m)}(t_i) \cos(2\pi f_s t_i) \exp\{-\alpha_s t_i\} & \text{if } m+1 \leq j \leq m+n_c \\ -\mathcal{L}_{(j-m-n_c)}(t_i) \sin(2\pi f_s t_i) \exp\{-\alpha_s t_i\} & \text{if } m+n_c+1 \leq j \leq m+n_c+n_s \end{cases} \quad (9.12)$$

and

$$I_j(t_i) = \begin{cases} -\sin(2\pi f_j[t_i + t_0] + \theta) \exp\{-\alpha_j t_i\} & \text{if } 1 \leq j \leq m \\ -\mathcal{L}_{(j-m)}(t_i) \sin(2\pi f_s t_i) \exp\{-\alpha_s t_i\} & \text{if } m+1 \leq j \leq m+n_c \\ -\mathcal{L}_{(j-m-n_c)}(t_i) \cos(2\pi f_s t_i) \exp\{-\alpha_s t_i\} & \text{if } m+n_c+1 \leq j \leq m+n_c+n_s \end{cases}. \quad (9.13)$$

The calculation implemented in the Markov chain Monte Carlo simulation is a combined parameter estimation and model selection calculation. The target distribution of the Monte Carlo simulation is the joint posterior probability for the nonlinear parameters and the expansion orders,  $P(f_s \alpha_s f_1 \alpha_1 \dots f_m \alpha_m t_0 \theta n_c n_s | D_R D_I I)$ . This is a marginal posterior probability where the amplitudes,  $\mathcal{A} \equiv \{\mathcal{A}_1, \dots, \mathcal{A}_{n_A}\}$ , and the standard deviation of the noise prior probability,  $\sigma$ , were removed using the rules of probability theory. We could have left these parameters in the joint posterior probability for all of the parameters and then targeted this distribution in the Monte Carlo simulations. However, we choose not to do this because when the order of one of the expansions is changed, the amplitudes that maximize the joint posterior probability for the parameters tend to change rather abruptly. Consequently, sampling them is very difficult; it was easier to remove them using marginalization. By removing them, we simply allowed probability to determine them for us automatically.

The joint posterior probability for the nonlinear parameters is computed from the joint posterior probability for all of the parameters,  $P(f_s \alpha_s f_1 \alpha_1 \dots f_m \alpha_m t_0 \theta n_c n_s \sigma \mathcal{A} | D_R D_I I)$ , by application of Bayes' theorem and the sum rule,

$$P(f_s \alpha_s f_1 \alpha_1 \dots f_m \alpha_m t_0 \theta n_c n_s | D_R D_I I) \propto \int d\mathcal{A} d\sigma P(f_s \alpha_s f_1 \alpha_1 \dots f_m \alpha_m t_0 \theta n_c n_s \sigma \mathcal{A} | D_R D_I I). \quad (9.14)$$

By repeatedly applying the product rule, the right-hand side of this equation may be factored to obtain

$$\begin{aligned} P(f_s \alpha_s f_1 \alpha_1 \dots f_m \alpha_m t_0 \theta n_c n_s | D_R D_I I) &\propto P(\theta | I) P(t_0 | I) P(f_s | I) P(\alpha_s | I) \\ &\times P(n_c | I) P(n_s | I) \left[ \prod_{l=1}^m P(f_l | I) P(\alpha_l | I) \right] \\ &\times \int d\mathcal{A} d\sigma P(\sigma | I) \left[ \prod_{j=1}^{n_A} P(\mathcal{A}_j | I) \right] \\ &\times P(D_R | f_s \alpha_s f_1 \alpha_1 \dots f_m \alpha_m t_0 \theta n_c n_s \mathcal{A} I) \\ &\times P(D_I | f_s \alpha_s f_1 \alpha_1 \dots f_m \alpha_m t_0 \theta n_c n_s \mathcal{A} I) \end{aligned} \quad (9.15)$$

where  $P(\theta|I)$  is the prior probability for the phase,  $P(t_0|I)$  is the prior probability for the time offset,  $P(f_s|I)$  and  $P(\alpha_s|I)$  are the prior probabilities for the solvent frequency and decay rate constant,  $P(n_c|I)$  and  $P(n_s|I)$  are the prior probability for the cosine and sine expansion orders and  $P(f_l|I)$  and  $P(\alpha_l|I)$  are the prior probability for the small peak frequencies and decay rate constants. These prior probabilities do not depend on the amplitudes or the standard deviation of the noise; consequently, we have removed them from the integral. However,  $P(\sigma|I)$ ,  $P(\mathcal{A}_j|I)$ ,  $P(D_R|f_s\alpha_s f_1\alpha_1 \dots f_m\alpha_m t_0\theta n_c n_s \mathcal{A}I)$ , and  $P(D_I|f_s\alpha_s f_1\alpha_1 \dots f_m\alpha_m t_0\theta n_c n_s \mathcal{A}I)$ , do participate in the integral.

All of the terms in Eq. (9.15) have been simplified to the point that they may now be assigned. The prior probability for the phase,  $P(\theta|I)$ , was assigned a uniform prior probability:

$$P(\theta|I) = \begin{cases} \frac{1}{360} & \text{If } 0 \leq \theta \leq 360, \\ 0 & \text{Otherwise} \end{cases} . \quad (9.16)$$

The prior probability for the delay time,  $P(t_0|I)$ , was assigned as a Gaussian:

$$P(t_0|I) = (2\pi\sigma_{t_0}^2)^{-\frac{1}{2}} \exp\left\{-\frac{t_0^2}{2\sigma_{t_0}^2}\right\}, \quad (9.17)$$

where  $\sigma_{t_0} = 3\Delta T$  and  $\Delta T$  is the sampling rate. In words we are saying that at one standard deviation we think we could miss the first three time points, but we think it would be very unlikely to miss the first 10.

The prior probability for the frequency of the solvent resonance,  $P(f_s|I)$ , is assigned as a Gaussian using input from the user. Using a double cursor the user is required to mark the low,  $f_{low}$ , and high,  $f_{high}$ , solvent frequency and the prior then constrains the solvent frequency to be within these marked values. The mean of this Gaussian is the mean of the low and high, so  $\hat{f}_s = (f_{low} + f_{high})/2$ , while the interval high minus low is taken to be a three standard deviation interval, so  $\hat{\sigma}_f = (f_{high} - f_{low})/3$ . The prior probability for the solvent resonance is then given by

$$P(f_s|I) \propto \begin{cases} \exp\left\{-\frac{(\hat{f}_s - f_s)^2}{2\hat{\sigma}_f^2}\right\} & \text{if } f_{low} \leq f_s \leq f_{high} \\ 0 & \text{otherwise} \end{cases} \quad (9.18)$$

Because of its width and because it is centered at the peak in the Fourier transform, this prior ensures that the Monte Carlo simulations do not wander into a nonphysical region of the parameter space. Additionally, the user has the option of not specifying a solvent resonance. When this is done the solvent frequency and decay rate constant,  $f_s$  and  $\alpha_s$ , are set equal to zero and the priors are not used in the calculation. Under these conditions the solvent model reduces to a trend in the real and imaginary FID data.

In a similar vane the prior probability for the solvent decay rate constant is also assigned as a Gaussian:

$$P(\alpha_s|I) \propto \begin{cases} \exp\left\{-\frac{(\hat{\alpha}_s - \alpha_s)^2}{2\hat{\sigma}_\alpha^2}\right\} & \text{if } 0 \leq \alpha_s \leq \alpha_{high} \\ 0 & \text{otherwise} \end{cases} \quad (9.19)$$

where  $\alpha_{high} = 10(f_{high} - f_{low})$ ,  $\hat{\alpha}_s = (f_{high} - f_{low})/2$  and  $\sigma_\alpha = f_{high} - f_{low}$ . This prior also serves little purpose other than to keep the Monte Carlo simulations from wandering into a nonphysical region of parameter space.

The prior probabilities for the expansion orders,  $P(n_c|I)$  and  $P(n_s|I)$ , were assigned using an exponential prior probability:

$$P(n_c|I) \propto \exp\{-n_c\} \quad \text{and} \quad P(n_s|I) \propto \exp\{-n_s\}. \quad (9.20)$$

Note that we have indicated that these priors are proportional to an exponential. Normally in model selection problems one must use fully normalized prior probabilities. However, for this problem, the normalization constant for these priors cancel so we do not bother deriving it.

The prior probability for the standard deviation of the noise prior probability,  $P(\sigma|I)$  will be assigned as a Jeffreys' prior:

$$P(\sigma|I) \propto \frac{1}{\sigma} \quad (9.21)$$

and we will not bound this prior. Normally this is a bad idea because it effectively introduces an infinity into the Bayesian calculation. We can get away with it here, because that same infinity is introduced into every model we consider in exactly the same way and so cancels out of the calculation when the distributions are normalized. However in many cases this cannot be done, and one must explicitly bound the above Jeffreys' prior and then integrate only over that bound.

The target distribution of the Monte Carlo simulation is a marginal distribution, Eq. (9.15), where the amplitudes have been removed by integration. When we assigned the prior probabilities for these amplitudes we assigned a broad unbounded Gaussian of the form:

$$P(\mathcal{A}_j|I) = \left(\frac{2\pi\sigma^2}{\beta^2 G_{jj}}\right)^{-\frac{1}{2}} \exp\left\{-\frac{\beta^2 G_{jj} \mathcal{A}_j^2}{2\sigma^2}\right\} \quad (9.22)$$

where  $\sigma$  is the standard deviation of the noise prior probability. The quantity  $G_{jj}$ , defined in Eq. (9.34) below, is the squared length of a model signal and serves to ensure that the prior information about a particular amplitude remains roughly the same. The hyperparameter  $\beta$  was set at  $\beta = 0.01$ , so when the marginalization occurs this prior affects the estimated amplitudes in the fourth decimal place: this prior stabilizes the matrix inversion that occurs when these amplitudes are marginalized.

The prior probabilities for the frequencies of the resonances are also generated from the input low and high frequency range. The interface computes a mean frequency,  $\hat{f}_l$ , as the average of the low and high frequency. And it takes the interval, high-low, as a 3 standard deviation interval. The prior probability for the frequency of the  $l$ th resonance of interest are then assigned as

$$P(f_l|I) \propto \exp\left\{-\frac{(\hat{f}_l - f_l)^2}{2\hat{\sigma}_l^2}\right\}, \quad (9.23)$$

subject to the condition  $f_{low} \leq f_l \leq f_{high}$ , where  $f_{low}$  and  $f_{high}$  are the input low and high parameter ranges.

Similarly, the prior probability for the decay rate constants are also assigned as Gaussians using the input linewidth. If we designate this linewidth as  $lb$  then the program generates

$$P(\alpha_l|I) \propto \exp\left\{-\frac{\alpha_l^2}{2 \times (3lb)^2}\right\} \quad (9.24)$$

subject to the condition  $0 \leq \alpha_l \leq 10lb$ . So this prior keeps the decay rate small, and allows the program to explore a wide range of decay rate constants, while keeping the rate constants near the input  $lb$  value.

Finally, Gaussian noise prior probabilities were used to assign the two likelihoods. The likelihood for the real part of the FID data was assigned as

$$P(D_R|f_s\alpha_s f_1\alpha_1 \dots f_m\alpha_m t_0\theta n_c n_s \sigma \mathcal{A}I) = (2\pi\sigma^2)^{-\frac{N}{2}} \exp \left\{ -\sum_{i=1}^N \frac{[d_R(t_i) - M_R(t_i)]^2}{2\sigma^2} \right\}. \quad (9.25)$$

Similarly, the likelihood for the imaginary part of the FID data was assigned as

$$P(D_I|f_s\alpha_s f_1\alpha_1 \dots f_m\alpha_m t_0\theta n_c n_s \sigma \mathcal{A}I) = (2\pi\sigma^2)^{-\frac{N}{2}} \exp \left\{ -\sum_{i=1}^N \frac{[d_I(t_i) - M_I(t_i)]^2}{2\sigma^2} \right\}. \quad (9.26)$$

We have now assigned all of the probabilities appearing in Eq. (9.15), and so are now able to evaluate the integrals over the amplitudes and the standard deviation of the noise prior probability. To begin this evaluation we substitute Eqs. (9.16,9.17,9.18,9.19, 9.20,9.21,9.22,9.23,9.24, 9.25,9.26) into Eq. (9.15) to obtain

$$\begin{aligned} P(f_s\alpha_s f_1\alpha_1 \dots f_m\alpha_m t_0\theta n_c n_s | D_R D_I I) &\propto \frac{1}{360} \\ &\times (2\pi\sigma_{t_0}^2)^{-\frac{1}{2}} \exp \left\{ -\frac{t_0^2}{2\sigma_{t_0}^2} \right\} \\ &\times \exp \left\{ -\frac{(\hat{f}_s - f_s)^2}{2\hat{\sigma}_f^2} \right\} \\ &\times \exp \left\{ -\frac{(\hat{\alpha}_s - \alpha_s)^2}{2\hat{\sigma}_\alpha^2} \right\} \\ &\times \exp \{-n_c\} \\ &\times \exp \{-n_s\} \\ &\times \prod_{l=1}^m (2\pi\hat{\alpha}_l^2)^{-\frac{1}{2}} \exp \left\{ -\frac{(\hat{f}_l - f_l)^2}{2\hat{\alpha}_l^2} \right\} \\ &\times \prod_{l=1}^m (2\pi\hat{\alpha}_l^2)^{-\frac{1}{2}} \exp \left\{ -\frac{(\hat{\alpha}_l - \alpha_l)^2}{2\hat{\alpha}_l^2} \right\} \\ &\times \int d\mathcal{A}d\sigma \\ &\times \frac{1}{\sigma} \\ &\times \prod_{j=1}^{n_A} \left( \frac{2\pi\sigma^2}{\beta^2 G_{jj}} \right)^{-\frac{1}{2}} \exp \left\{ -\frac{\beta^2 G_{jj} \mathcal{A}_j^2}{2\sigma^2} \right\} \\ &\times (2\pi\sigma^2)^{-\frac{N}{2}} \exp \left\{ -\sum_{i=1}^N \frac{[d_R(t_i) - M_R(t_i)]^2}{2\sigma^2} \right\} \\ &\times (2\pi\sigma^2)^{-\frac{N}{2}} \exp \left\{ -\sum_{i=1}^N \frac{[d_I(t_i) - M_I(t_i)]^2}{2\sigma^2} \right\} \end{aligned} \quad (9.27)$$

where each of the prior probabilities have been written on a separate line for easy identification.

Now in preparation of evaluating the integrals over the amplitudes and standard deviation of the noise prior probability, we are going to drop all constants that are common to each model, one obtains

$$\begin{aligned}
P(f_s \alpha_s f_1 \alpha_1 \dots f_m \alpha_m t_0 \theta n_c n_s | D_R D_I I) &\propto \exp \left\{ -\frac{t_0^2}{2\sigma^2} - \frac{(\hat{f}_s - f_s)^2 + (\hat{\alpha}_s - \alpha_s)^2}{2\hat{\alpha}_s^2} - n_c - n_s \right\} \\
&\times \exp \left\{ -\sum_{l=1}^m \frac{(f_l - \hat{f}_l)^2 + (\hat{\alpha}_l - \alpha_l)^2}{2\hat{\alpha}_l^2} \right\} \\
&\times \int dA d\sigma \sigma^{-2N-1} \prod_{j=1}^{n_A} \left( \frac{2\pi\sigma^2}{\beta^2 G_{jj}} \right)^{-\frac{1}{2}} \exp \left\{ -\frac{\beta^2 \mathcal{A}_j^2 G_{jj}}{2\sigma^2} \right\} \\
&\times \exp \left\{ -\sum_{i=1}^N \frac{[d_R(t_i) - M_R(t_i)]^2 + [d_I(t_i) - M_I(t_i)]^2}{2\sigma^2} \right\}.
\end{aligned} \tag{9.28}$$

Now designating the integral as  $\mathcal{I}$  and working only on that integral by first substituting Eqs. (9.12) and (9.13) into  $\mathcal{I}$  one obtains

$$\mathcal{I} = \prod_{j=1}^{n_A} \left( \frac{2\pi}{\beta^2 G_{jj}} \right)^{-\frac{1}{2}} \int dA d\sigma \sigma^{-2N-1-n_A} \exp \left\{ -\frac{Q}{2\sigma^2} \right\} \tag{9.29}$$

where

$$Q \equiv 2N\bar{d}^2 - 2 \sum_{l=1}^{n_A} \mathcal{A}_l T_l + \sum_{j=1}^{n_A} \sum_{k=1}^{n_A} \mathcal{A}_j \mathcal{A}_k g_{jk}. \tag{9.30}$$

The mean-square data value,  $\bar{d}^2$ , is given by

$$\bar{d}^2 = \frac{1}{2N} \sum_{i=1}^{2N} [d_R(t_i)^2 + d_I(t_i)^2]. \tag{9.31}$$

The projection of the data onto the  $l$ th model function,  $T_l$ , is given by

$$T_l = \sum_{i=1}^N [d_R(t_i) R_l(t_i) + d_I(t_i) I_l(t_i)]. \tag{9.32}$$

The matrix  $g_{jk}$  is given by

$$g_{jk} = (1 + \beta^2 \delta_{jk}) G_{jk} \tag{9.33}$$

where  $\delta_{jk}$  is the Kronecker delta function and

$$G_{jk} \equiv \sum_{i=1}^N [R_j(t_i) R_k(t_i) + I_j(t_i) I_k(t_i)]. \tag{9.34}$$

The amplitude integrals are multivariate Gaussian integrals and evaluating such integrals is straight-forward, one obtains

$$\mathcal{I} = |g_{jk}|^{-\frac{1}{2}} \prod_{j=1}^{n_A} (\beta^2 G_{jj})^{\frac{1}{2}} \int d\sigma \sigma^{-2N-1} \exp \left\{ -\frac{2N\bar{d}^2 - n_A \bar{h}^2}{2\sigma^2} \right\} \tag{9.35}$$

where the factors involving  $2\pi$  have canceled,

$$\overline{h^2} \equiv \frac{1}{n_{\mathcal{A}}} \sum_{j=1}^{n_{\mathcal{A}}} T_j \hat{\mathcal{A}}_j \quad (9.36)$$

and the  $\hat{\mathcal{A}}_j$  are given by the solution to

$$\sum_{j=1}^{n_{\mathcal{A}}} g_{jk} \hat{\mathcal{A}}_j = T_k. \quad (9.37)$$

The amplitudes,  $\hat{\mathcal{A}}$ , are the maximum posterior probability estimates given the parameter values.

The integral over the standard deviation of the noise prior probability may be transformed into a Gamma function and the integral is then easily evaluated. Evaluating the integral, one obtains

$$\mathcal{I} = |g_{jk}|^{-\frac{1}{2}} \prod_{j=1}^{n_{\mathcal{A}}} (\beta^2 G_{jj})^{\frac{1}{2}} \left[ \frac{2N\overline{d^2} - n_{\mathcal{A}}\overline{h^2}}{2} \right]^{-N} \quad (9.38)$$

where a factor of  $2\Gamma(N)$  was dropped because it cancels when the posterior probability for the nonlinear parameters is normalized.

Finally inserting  $\mathcal{I}$  back into Eq. (9.28), one obtains

$$\begin{aligned} P(f_s \alpha_s f_1 \alpha_1 \dots f_m \alpha_m t_0 \theta n_c n_s | D_R D_I I) &\propto \exp \left\{ -\frac{t_0^2}{2\sigma_{t_0}^2} - \frac{(\hat{f}_s - f_s)^2}{2\sigma_f^2} - \frac{(\hat{\alpha}_s - \alpha_s)^2}{2\sigma_\alpha^2} - n_c - n_s \right\} \\ &\times \exp \left\{ -\sum_{l=1}^m \frac{(\hat{f}_l - f_l)^2 + (\hat{\alpha}_l - \alpha_l)^2}{2\hat{\alpha}_l^2} \right\} \\ &\times |g_{jk}|^{-\frac{1}{2}} \prod_{j=1}^{n_{\mathcal{A}}} (\beta^2 G_{jj})^{\frac{1}{2}} \left[ \frac{2N\overline{d^2} - n_{\mathcal{A}}\overline{h^2}}{2} \right]^{-N} \end{aligned} \quad (9.39)$$

as the posterior probability for the nonlinear parameters. It is this posterior probability that is targeted by the Monte Carlo simulation. For a bit more on this topic, see [18]. There you will find some examples of this calculation and more on how the various prior probabilities were assigned.

## 9.2 Outputs From The Big Peak/Little Peak Package

The text outputs files from the Big Peak/Little Peak packages consist of: “Bayes.prob.model,” “mcmc.values,” “Bayes.params,” “Console.log,” “Bayes.accepted” and a “Bayes.Condensed.File.” These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the mcmc.values report is discussed in Appendix D and the other reports are discussed in Chapter 3. Additionally, the “Plot Results Viewer” can be used to view the output probability density functions. In addition to the standard data, model and residual plots there are probability density functions for each of the marked frequencies including the solvent frequency. These probability density plots include plots for the frequency, decay rate constant and the amplitudes. In the case of the solvent, there are a number of additional plots that help one determine how the water resonance is changing as a function of time. These plots are shown in Fig. 9.2. Here is a description of these additional plots:

- Fig. 9.2 Panel **a** is a two line plot. The red line is a plot of the logarithm of the estimated solvent envelope as a function of acquisition time. The solvent envelope is defined in Eq. (9.2) as the function  $A(t) \exp\{-\alpha_s t\}$ . If a resonance is a pure exponentially decaying sinusoid, this function should be a straight line whose slope would be estimated to be  $-\alpha'_s$ , the blue line, where  $\alpha'_s$  is the estimated slope given an exponentially decaying sinusoidal model.
- Fig. 9.2 panel **b** is the difference between the two lines shown in panel **a**. The smaller this difference is, the more exponential the decay is.
- Panel **c** is the phase of the solvent frequency as a function of the time. The phase of the solvent is defined in Eq. (9.2) as the function  $\phi(t_i)$  and this plot starts at zero because the constant phase has been removed.
- Panel **d** in Fig. 9.2 is the accumulated phase of the water resonance. If the water frequency is a constant, this plot would be the water frequency times time.

Figure 9.2: The Time Dependent Parameters

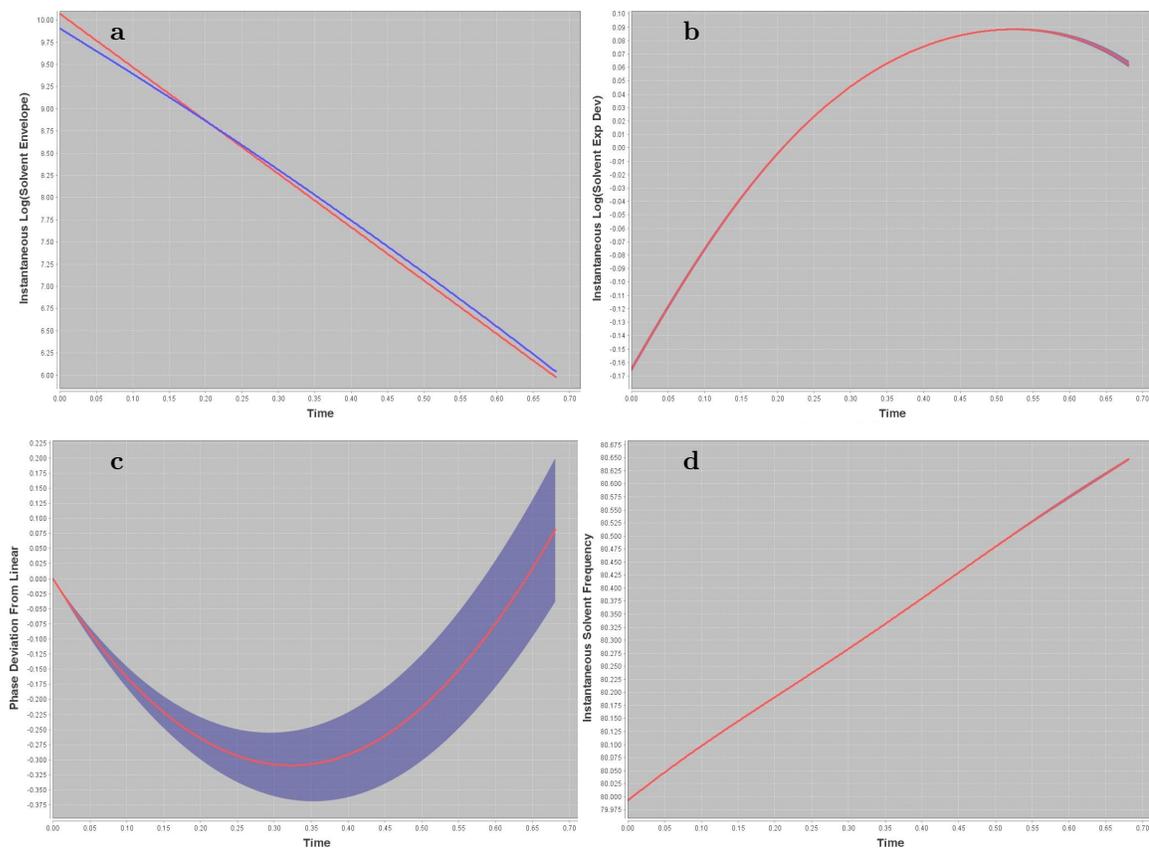


Figure 9.2: Panel **a** shows the logarithm of the estimated instantaneous decay envelope of the water (blue) as a function of time. The red line is the logarithm of the best fit decay envelope assuming an exponentially decaying sinusoid. Note that the instantaneous water frequency starts low, and then gets faster. Eventually the difference between these two lines goes to a constant. Panel **b** is the difference between the two curves shown in Panel **a**, i.e., the deviation from linear. Panel **c** is a deviation in the phase from a constant. Panel **d** is the accumulated solvent frequency as a function of time. As one can see, this curve is highly linear indicating that these phase variations are small.

## Chapter 10

# Metabolic Analysis

The metabolic analysis package is designed to analyze Fids of a known type where one relates the resonance intensities to some metabolic parameters. For example, in the glutamate package we have a number of metabolic parameters that relate the rates at which carbon enters the metabolic cycles to the intensities of the various carbon resonances. The program, Bayes Metabolite, infers these metabolic parameters using a metabolic model to calculate the intensities of the metabolic resonances. The program takes as its input a nonarrayed Fid and a series of files that describe the metabolite and the nuisance resonances. A metabolite is the set of resonances whose intensities are predicted by the metabolic model. The nuisance resonances are all the other resonances in the data. Metabolites are described in a file having an “.ISO” suffix, and nuisance resonances are described in files suffixed with “.Res”. For example, the example metabolite file is named “Example.ISO” while the example nuisance resonance file is named “Example.Res”. The file format is the same for both metabolites and nuisance resonances. However, nuisance resonances use only the part of the file that describes the resonances, those parts of the file describing the metabolites are ignored. Consequently, a metabolic file may be used as a nuisance resonance file, but a nuisance resonance file may not be as used as a metabolite file. The metabolite files identify the subroutine that processes the metabolite, they describe the metabolic parameters, the coupling constants, and the resonances. Here, by describe we mean, these files specify the names of the parameters, their valid ranges, and the prior probability to be used for each parameter.

Using the Bayes Metabolite package involves several steps: First, the Fid must be loaded. Second, one must specify the model by loading the metabolite models and the nuisance resonance model. Third, the spectrum must be properly referenced so that all of the resonances in the model overlap the peaks in the spectrum of the Fid. Fourth, the metabolic parameters and the resonance frequencies must be reviewed to make sure that everything is correct. Finally, one runs the analysis and views the results using the standard widgets. We discuss each of these steps in more detail in the following paragraphs.

**Select** the Metabolite package from the Package menu.

**Load** the Fid to be analyzed using the Files menu. The loaded Fid will be displayed in the Fid Data Viewer.

**Load the Metabolite** file using the “Metabolite System” or the “Metabolite User” see Fig. 10.1.

Figure 10.1: The Bayes Metabolite Interface

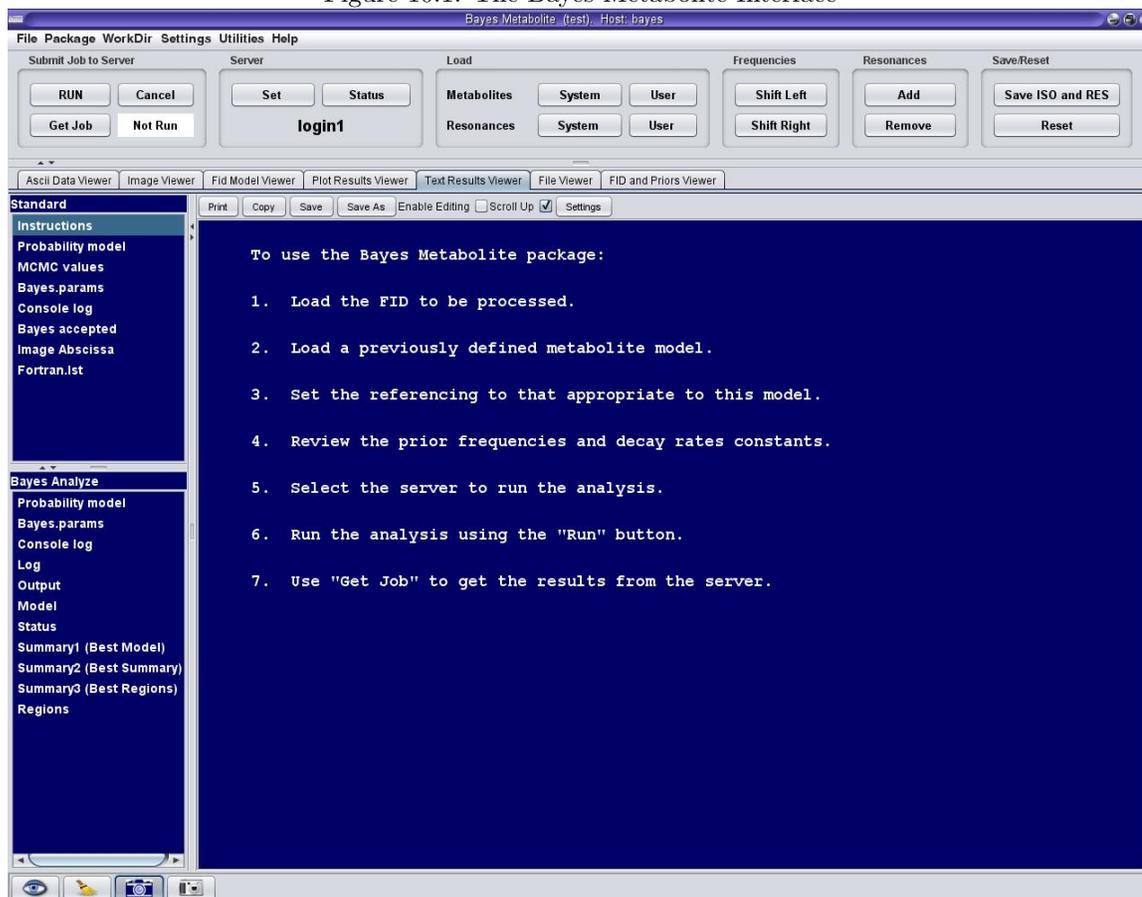


Figure 10.1: The Bayes Metabolite package is used to analyze Fids that are described by a metabolite model. To use the package, one loads the Fid, Loads the Metabolite and resonance models, shifts the resonance if necessary, and finally, runs the analysis.

Activating the “Metabolite System” button will bring up a list of Metabolite files on your specified server, while activating the “Metabolite User” button, will bring up a list of Metabolite files contained in your BayesHome directory: “BayesHome/Bayes.Predefined.Spec,” where the BayesHome directory is where the Bayesian Analysis files are located in your home directory. This directory will be Bayes, unless you have changed it. In either case, select the metabolite file you wish to load and hit the “Open” button. In addition to loading the resonances from the metabolite file, opening the metabolite file will also copy the metabolite file to BayesHome/Bayes.Predefined.Spec directory. After loading the metabolite, the interface displays the Metabolite in a special Metabolite viewer, see Fig. 10.2. The Metabolite viewer can be used to shift blocks of resonances and it can be used to modify the prior ranges.

**Load the Resonance** file by activating either the “Resonance System” or the “Resonance User” buttons, see Fig. 10.1. These buttons function exactly the same as “Metabolite System” and User buttons, but these load resonance files, not metabolite files.

**Examine** each section of the spectrum and the model to make sure that the Metabolic and Resonance models are properly aligned. When aligning resonances, all displayed resonances are shifted in a block. To shift a block of resonances, use a double cursor. Place one cursor on the center of one of the peaks, and place the other cursor at the location you wish to shift that peak to. The “Shift Left” button assumes the right-hand cursor is the peak and it shifts the peak to the location of the left-hand cursor. Similarly, the “Shift Right” button assumes the left cursor is on the peak and shifts it to the location of the right cursor.

**Review** the metabolic and resonance parameters by clicking on their names on prior viewer. You can change any parameter range in a metabolite.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

**View** the Bayes Metabolite model by activating “Fid Model Viewer.” The Bayes Metabolite model Fid contains multiple traces. The first trace is the original data, the second is a model of the Fid produced from the parameter values that maximized the joint posterior probability for the parameters, and the third is the residuals (the difference between the data and the model). Each metabolite also outputs one trace for each site in each metabolite. For example, the Glutamate.2.0 metabolite model has 4 sites, so four additional output site traces are written. Each site trace contains all of the resonances in that site. When metabolite models were loaded all of the nuisance resonances are output in a single final trace. So the Glutamate.2.0 metabolite would output a total of  $3 + 4 + 1 = 8$  traces. When no metabolite models are loaded, the nuisance resonances are output in individual traces.

Figure 10.2: The Bayes Metabolite Viewer

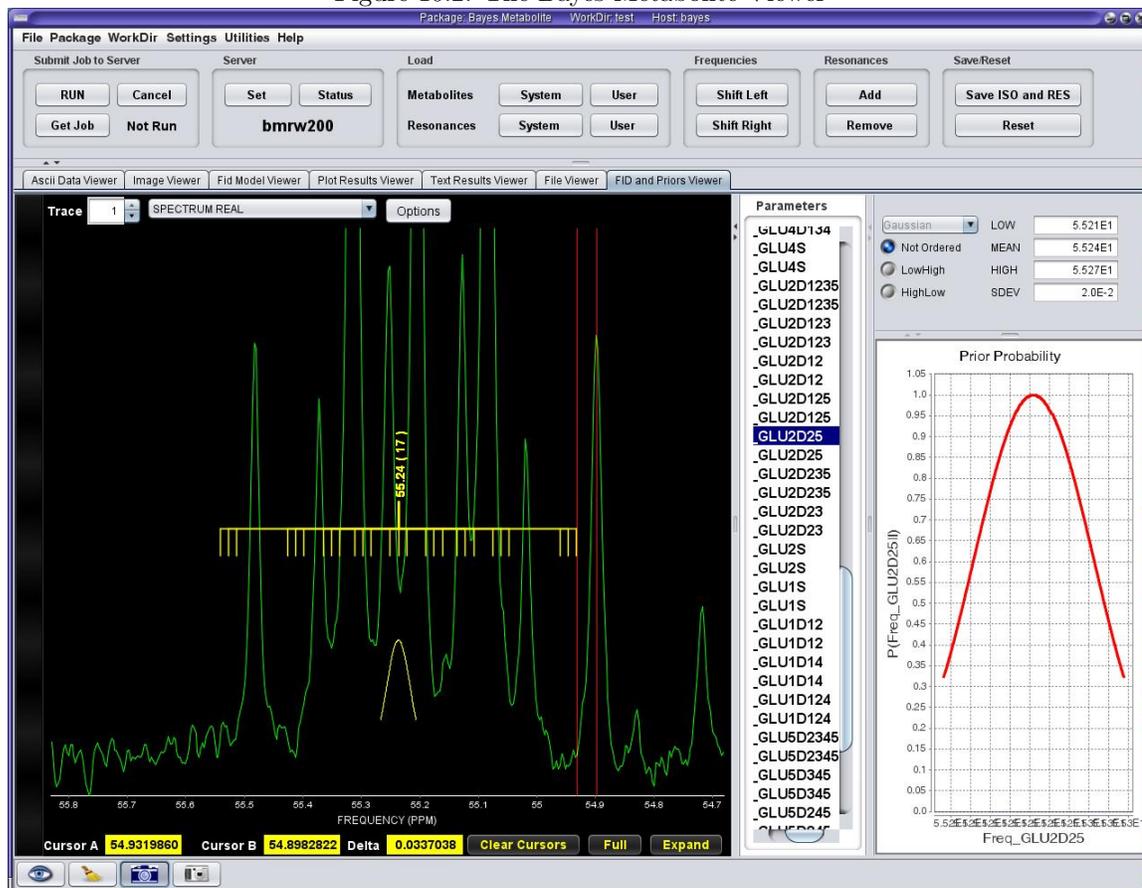


Figure 10.2: This is an example of the metabolite viewer when the “Glutamate.3.0.ISO” file is loaded. The left-hand portion of the viewer is the standard file viewer while the right-hand portion is an instance of the prior viewer. Using the file viewer one can align the spectrum and the model using the shift buttons. These buttons shift blocks of resonances either left or right by the amount of the double cursor. The prior viewer can be used to adjust and change the ranges of the various priors.

## 10.1 The Metabolic Model

The model used in Bayes Metabolite is of two parts, a metabolic model and a nuisance resonance model. These two models differ from each other in two important respects, nuisance resonance models have an amplitude associated with each resonance and they have no metabolic parameters; while metabolic models predict the fractional intensity of each metabolic resonance from the metabolic parameters. Thus metabolic models contain the chemistry, while nuisance resonances complete the spectral model.

Bayes Metabolite process a single non-arrayed Fid. Free induction decays consists of two data sets, a real data set and a quadrature or imaginary data set. These data sets are related to each other in the time domain model by a 90° phase shift. The time domain model of real data used in Bayes Metabolite is

$$d_R(t_i) = G_R(t_i) + n_i \quad (10.1)$$

where  $d_R(t_i)$  represents the data acquired at time  $t_i$ ,  $G_R(t_i)$  represents the model of the data and  $n_i$  represents the misfit between the data and the model. The real data model is a sum of four terms:

$$G_R(t_i) \equiv F_R\delta(t_1 - t_i) + C_R + \sum_{j=1}^m M_{Rj}(t_i) + N_R(t_i), \quad (10.2)$$

these four terms are the first point model  $F_R\delta(t_1 - t_i)$ , a constant model  $C_R$ , the metabolic model represented by the sum, and finally the nuisance model  $N_R(t_i)$ . In NMR Fid data, it is not unusual for the first values to be in error by as much as a factor of two. Rather than discarding this data, this effect is included in the Fid model using a first point model. This is represented symbolically by  $F_R\delta(t_1 - t_i)$  where  $F_R$  is related to the true intensity of the first point, and the delta function ensures this model is zero everywhere except the first data value. In addition to first point problems, it is also not unusual for the data to contain a constant offset. This is modeled by a constant  $C_R$ . Bayes Metabolite can process up to 4 different metabolic models at one time. The number of metabolic models being processed is  $m$ , where the  $j$ th metabolic model is represented by symbolically  $M_{Rj}(t_i)$ , and the nuisance resonance are represented by  $N_R(t_i)$ .

The metabolite model for the real data,  $M_R(t_i)$ , is a sum over each resonance in each isotopomer in the metabolic model,

$$M_R(t_i) = \sum_{\ell=1}^{N_B} B_{\ell} \sum_{k=1}^{n_{\ell}} F_{\ell k}(\Theta) \text{R}(2\pi f_{\ell k}[t_i + t_0] + \phi_{\ell k}) \exp\{-\alpha_{\ell k} t_i\} \quad (10.3)$$

where we have represented the resonances symbolically by “RealRes”. Metabolic resonances are grouped together by isotopomers or sites. For example, the  $k$ th resonance of the  $\ell$ th site might be a C2 doublet coming about from the  $J_{12}$  coupling in glutamate. The number of sites is designated as  $N_B$  and the number of resonances within the  $\ell$ th site is  $n_{\ell}$ . The amplitude of each resonance is calculated from the total amplitude of the site,  $B_{\ell}$ , and the fractional intensity,  $F_{\ell k}$ , of each site resonance. These fractional intensities are computed from the metabolite parameters  $\Theta$ . The total fractional intensity of a given site sums to one, so

$$\sum_{k=1}^{n_{\ell}} F_{\ell k}(\Theta) = 1. \quad (10.4)$$

The frequency of the  $k$ th resonance of the  $\ell$ th site is denoted as  $f_{\ell k}$ ,  $t_0$  may be thought of as a linear or first order phase correction, and  $\phi_{\ell k}$  is the constant part of the resonance phase. In this model each resonances decays exponentially. The exponential decay rate constant is represented by  $\alpha_{\ell k}$ . Note that if a resonances is a multiplet then all lines in a multiplet decay with the same exponential decay rate constant.

The resonance model, “RealRes,” is a multiplet of multiplets of multiplets. We will use this resonance model for both metabolic and nuisance resonances. Consequently we will represent the frequency of the resonance as  $f_x$  where the  $x$  subscript could be both a site and resonances subscript as in the metabolic models, or it could be only a resonance subscript. The time domain model of a multiplets of multiplet of multiplets is given by

$$\text{RealRes}(2\pi f_x[t_i + t_0] + \phi_x) \equiv \sum_{u=1}^{O_P} R_u^{O_P} \sum_{v=1}^{O_S} R_v^{O_S} \sum_{w=1}^{O_T} R_w^{O_T} \cos(2\pi \mathcal{F}_{uvw}[t_i + t_0] + \phi_x) \quad (10.5)$$

where the primary multiplet order is  $O_P$ , the secondary order is  $O_S$ , and  $O_T$  is the tertiary order. The fractional intensity of the  $u$ th line of the primary multiplet is  $R_u^{O_P}$ . Similarly,  $R_v^{O_S}$  and  $R_w^{O_T}$  are the fractional intensities of the various lines in the secondary and tertiary multiplets. The fractional intensities of for each multiplet sums to one, so

$$\sum_{u=1}^{O_P} R_u^{O_P} = 1, \quad \sum_{v=1}^{O_S} R_v^{O_S} = 1 \quad \text{and} \quad \sum_{w=1}^{O_T} R_w^{O_T} = 1; \quad (10.6)$$

then by definition the fractional intensity of the various lines in the multiplet of multiplets of multiplets must sum to one:

$$\sum_{u=1}^{O_P} \sum_{v=1}^{O_S} \sum_{w=1}^{O_T} R_u^{O_P} R_v^{O_S} R_w^{O_T} = 1. \quad (10.7)$$

The interface defaults these fractional intensities to spin one half multiplets. However, these fractional intensities are input from the interface and may be changed by the user. Finally, the frequency,  $\mathcal{F}_{uvw}$ , is given by

$$\begin{aligned} \mathcal{F}_{uvw} &= \mathcal{F}_{uv} - \frac{J_T(O_T + 1 - 2w)}{2}, \\ \mathcal{F}_{uv} &= \mathcal{F}_u - \frac{J_S(O_S + 1 - 2v)}{2}, \\ \mathcal{F}_u &= f_x - \frac{J_P(O_P + 1 - 2u)}{2} \end{aligned} \quad (10.8)$$

where  $J_P$ ,  $J_S$  and  $J_T$  are the primary, secondary and tertiary coupling constants in Hertz and  $f_x$  is the center frequency of the multiplet in Hertz. Note that the package actually uses PPM as its units for the center frequency, so the program converts the frequency,  $f_x$ , to Hertz prior to using these formula.

The nuisances resonance model,  $N_R(t_i)$ , is defined as

$$N_R(t_i) \equiv \sum_{k=1}^n B_k \text{RealRes}(2\pi f_k[t_i + t_0] + \phi_k) \exp\{-\alpha_k t_i\} \quad (10.9)$$

where  $B_k$  is the amplitude of the  $k$ th resonance. The other quantities appearing in this equation have the same meaning as those defined in Eq. (10.5), except here they refer to the nuisance resonances.

The time domain the data consists of a real and a quadrature or imaginary data. The imaginary data are modeled as

$$d_I(t_i) = G_I(t_i) + \text{Noise} \quad (10.10)$$

with

$$G_I(t_i) \equiv F_I \delta(t_1 - t_i) + C_I + \sum_{j=1}^m M_{Ij}(t_i) + N_I(t_i) \quad (10.11)$$

where  $d_I(t_i)$  represents the imaginary data at time  $t_i$ ,  $F_I$  is the first point model,  $C_I$  is the constant offset,  $M_{Ij}(t_i)$  is the  $j$ th metabolic model,  $N_I(t_i)$  are the nuisance resonances, and “Noise” represents the misfit between the data and the model in the imaginary data. Note that “Noise” is only a symbolic representation of this misfit and should not be taken to mean that the noise is the same in the real and imaginary data.

The imaginary metabolic model,  $M_I(t_i)$ , is just a  $90^\circ$  phase shifted version of the real metabolic model and is given by

$$M_I(t_i) = \sum_{\ell=1}^{N_B} B_\ell \sum_{k=1}^{n_\ell} F_{\ell k}(\Theta) \text{ImagRes}(2\pi f_{\ell k}[t_i + t_0] + \phi_{\ell k}) \exp\{-\alpha_{\ell k} t_i\} \quad (10.12)$$

where

$$\text{ImagRes}(2\pi f_k[t_i + t_0] + \phi_k) \equiv - \sum_{u=1}^{O_P} R_u^{O_P} \sum_{v=1}^{O_S} R_v^{O_S} \sum_{w=1}^{O_T} R_w^{O_T} \sin(2\pi \mathcal{F}_{uvw}[t_i + t_0] + \phi_x). \quad (10.13)$$

All of the quantities within these equations are the same as those defined for the real channel. Similarly, The nuisances resonance model for the imaginary data,  $N_I(t_i)$ , is defined as

$$N_I(t_i) \equiv \sum_{k=1}^n B_k \text{ImagRes}(2\pi f_k[t_i + t_0] + \phi_k) \exp\{-\alpha_k t_i\}. \quad (10.14)$$

## 10.2 The Bayesian Calculation

The calculation performed by Bayes Metabolite is for the marginal posterior probability for each parameter appearing in the model. If we designate all of the parameters appearing in the model, except the standard deviation of the noise prior probability, as  $\chi$ , then the marginal probability for  $\chi_j$  is given by

$$P(\chi_j|DI) \propto \int d\chi_1 \dots d\chi_{j-1} d\chi_{j+1} \dots d\chi_z P(\chi|DI) \quad (10.15)$$

where all of these integrals are evaluated numerically. The posterior probability for each of the parameters may be computed from joint posterior probability,  $P(\chi|DI)$ , and it is this distribution that is targeted by the Markov chain Monte Carlo simulations.

The joint posterior probability for the parameters is a marginal probability where the standard deviation of the noise has been removed analytically:

$$P(\chi|DI) \propto \int d\sigma P(\chi\sigma|DI) \quad (10.16)$$

where  $P(\chi\sigma|D\sigma I)$  is the joint posterior probability for all of the parameters. To compute this probability one applies Bayes' theorem to obtain

$$P(\chi|DI) \propto \int d\sigma P(\chi\sigma|I)P(D|\chi\sigma I). \quad (10.17)$$

In the preceding we are using  $D$  to stand for both the real and imaginary data. If we designate  $D_R$  and  $D_I$  as the real and imaginary data, then the joint posterior probability is given by

$$P(\chi|DI) \propto \int d\sigma P(\chi|I)P(\sigma|I)P(D_R|\chi\sigma I)P(D_I|\chi\sigma I). \quad (10.18)$$

We used logical independence to factor both the joint prior probability and the likelihood. The joint prior probability was factored into a prior probability for the standard deviation of the noise,  $P(\sigma|I)$ , and the joint prior probability for all of the other parameters  $P(\chi|I)$ . The joint likelihood for the data,  $P(D_R D_I|\chi\sigma I)$ , was factored into independent likelihood for each data set separately.

If we assign the prior probability for the noise standard deviation using a Jeffreys' prior, and assign the two likelihoods,  $P(D_R|\chi\sigma I)$  and  $P(D_I|\chi\sigma I)$ , using a Gaussian prior probability, one obtains

$$P(\chi|DI) \propto \int P(\chi|I) d\sigma \frac{1}{\sigma} (2\pi\sigma^2)^{-N} \exp\left\{-\frac{Q_R + Q_I}{2\sigma^2}\right\} \quad (10.19)$$

with

$$Q_R = \sum_{i=1}^N [d_R(t_i) - G_R(t_i)]^2 \quad (10.20)$$

where  $N$  is the total complex data values, and

$$Q_I = \sum_{i=1}^N [d_I(t_i) - G_I(t_i)]^2. \quad (10.21)$$

Evaluating the integral over the standard deviation, one obtains

$$P(\chi|DI) \propto P(\chi|I) [Q_R + Q_I]^{-N} \quad (10.22)$$

where we have dropped some constants that cancel on normalization.

The joint prior probability for the other parameters is factored into independent prior probability for each parameter

$$\begin{aligned} P(\chi|I) &= P(F_R|I)P(F_I|I)P(C_R|I)P(C_I|I)P(t_0|I)P(\phi|I) \\ &\times \left[ \prod_{k=1}^{N_J} P(J_k|I) \right] \left[ \prod_{k=1}^n P(B_k|I)P(f_k|I)P(\alpha_k|I)P(\phi_k|I) \right] \\ &\times \prod_{j=1}^m \left\{ \left[ \prod_{k=1}^{N_{\Theta_j}} P(\Theta_{jk}|I) \right] \left[ \prod_{k=1}^{N_{J_j}} P(J_{jk}|I) \right] \left[ \prod_{k=1}^{N_{B_j}} P(B_{jk}|I) \right] \right. \\ &\times \left. \left( \prod_{\ell=1}^{N_{B_j}} \left[ \prod_{k=1}^{n_{j\ell}} P(f_{j\ell k}|I)P(\alpha_{j\ell k}|I)P(\phi_{j\ell k}|I) \right] \right) \right\} \end{aligned} \quad (10.23)$$

Figure 10.3: Bayes Metabolite Parameters And Probabilities List

Parameter	Description	
$N_J$ :	The number of $J$ coupling constants in the nuisance model	
$n$ :	The number of resonances in the nuisance model	
$N_{\Theta_j}$ :	The number of metabolic parameter in the $j$ th metabolite	
$N_{J_j}$ :	The number of $J$ coupling constants in the $j$ th metabolite	
$N_{B_j}$ :	The number of sites/amplitudes in the $j$ th metabolite	
$n_{\ell_j}$ :	The number of resonances in the $\ell$ th site of the $j$ th metabolite	
Prior Probability	Description of the Global Parameters	Assigned
$P(F_R I)$ :	the real first point offset	Gaussian
$P(F_I I)$ :	the imaginary first point offset	Gaussian
$P(C_R I)$ :	the real constant offset	Gaussian
$P(C_I I)$ :	the imaginary constant offset	Gaussian
$P(\phi I)$ :	the constant part of the resonance phase	uniform
$P(t_0 I)$ :	the time offset, i.e., linear frequency dependent phase	Gaussian
Prior Probability	Description of the Nuisance Resonances Parameters	Assigned
$P(J_k I)$ :	the $k$ th $J$ coupling constant	user input
$P(B_k I)$ :	the amplitude of the $k$ th resonance	Gaussian
$P(f_k I)$ :	the frequency of the $k$ th resonance	user input
$P(\alpha_k I)$ :	the decay rate constant of the $k$ th resonance	user input
$P(\phi_k I)$ :	the constant phase of the $k$ th resonance	user input
Prior Probability	Description of the Metabolic Parameters	Assigned
$P(\Theta_{jk} I)$ :	the $k$ th metabolic parameter in the $j$ th metabolite	user input
$P(J_{jk} I)$ :	the $k$ th coupling constant in the $j$ th metabolite	user input
$P(B_{jk} I)$ :	the $k$ th amplitude in the $j$ th metabolite	Gaussian
$P(f_{j\ell k} I)$ :	the frequency of the $k$ th resonance, $\ell$ th site, $j$ th metabolite	user input
$P(\alpha_{\ell j k} I)$ :	the decay rate of the $k$ th resonance, $\ell$ th site, $j$ th metabolite	user input
$P(\phi_{jk} I)$ :	the phase of the $k$ th resonance, $\ell$ th site, $j$ th metabolite	user input

where we define the symbols and probabilities appearing in this equation in Table 10.3. In this table “user input” means that the information needed to assign the prior is read from the input parameter files. Prior probabilities assigned in this way are either bounded uniform or Gaussian prior probabilities. In both cases the user must specify all of the information necessary to assign the prior. Prior probabilities noted as being “Gaussians” are bounded zero mean Gaussian prior probabilities and the bound and standard deviations of these Gaussians are set by Bayes Metabolite. For example, the range for the amplitudes extends from zero to a few times the maximum data value and this range represents a 3 standard deviations interval; so the prior keeps the amplitudes positive and in a range that is reasonable for the data. The prior probability for the phases, the  $P(\phi_{jk}|I)$  and the  $P(\phi_k|I)$ , are assigned as delta functions:

$$P(\phi_{jk}|I) = \delta(\phi - \phi_{jk}), \quad \text{and} \quad P(\phi_k|I) = \delta(\phi - \phi_k), \quad (10.24)$$

if the resonances has a common constant phase, and they are assigned a uniform prior probabilities if the resonance has a unique phase. Whether or not the resonance has a common phase is defined in the parameter files.

Markov chain Monte Carlo using simulated annealing is used to draw samples from Eq. (10.22). From these samples the marginal posterior probability for each parameters is generated. These posterior probabilities are written to an output files contained within the experiment. Additionally, the mean and standard deviation estimates of the parameters are written to `mcmc.values` file contained in the experiment. Finally, the parameters that maximized the joint posterior probability for the parameters are used to generate a model of the original Fid. This model may then be viewed using the appropriate interface widgets.

## 10.3 The Metabolite Models

Metabolic models consist of three parts: First, there must be a physical model of some chemical process. This physical model must lead to a prediction of the fractional intensities of the resonances in the spectrum. Second, a subroutine must use this physical model to compute the fractional intensities of each resonance from the metabolic parameters. Finally, the metabolic and resonance parameters must be described to the program that processes the metabolite. In the following subsections we will describe the current metabolites, and then in the next Section we will give detailed instructions on how to build and test your own metabolic models.

### 10.3.1 The IPGD\_D2O Metabolite

The methodology used for the estimates of the contribution of glycogenolysis, glycerol, and PEP to serum glucose was first proposed by Landau [19], with analysis of  $^2\text{H}$  enrichment carried out by mass spectroscopy. How this labeling is carried out is illustrated symbolically in Fig. 10.4. Peak intensities from the H2, H5, H6S and H6R resonances are used to calculate the contribution of Glycogen, Glycerol and the Kreb's cycle to the creation of Glucose. The three contributions are computed from the H2, H5 and H6 intensities. The H6 intensity is either H6R or H6S. Ratios are computed using H6R intensities to make sure that nothing has gone wrong in the experiment; it is the ratios computed using the H6S intensities that are of metabolic importance.

Data typical of this metabolic model are shown in Fig. 10.5. For this work, the samples of serum glucose were prepared according to Jones, et. al. [34]. Deuterium NMR spectroscopy was carried out at 92Mhz. The 90 degree pulse width was 7.9us. The spectral width was set to 10ppm and 928 data points were collected in the Fid to give a total acquisition time of 1s with no further relaxation delay. The longest relaxation time for a deuteron in the IPG derivative is 230ms.

The four resonances used in the metabolic calculations are labeled H2, H5, H6R and H6S respectively. The remaining resonances, including the solvent peaks are described by the `IPGD_D2O.Res` file. In this simple metabolite, the metabolic parameters *are* the fractional intensities of the H2, H5, H6R and H6S resonances. Because the total fractional intensity must add up to one, we impose the condition

$$1 = \text{H2} + \text{H5} + \text{H6}_R + \text{H6}_S. \quad (10.25)$$

From the fractional intensities there are six derived parameters that are output from the `IPGD_D2O`

Figure 10.4: The IPGD\_D20 Metabolite

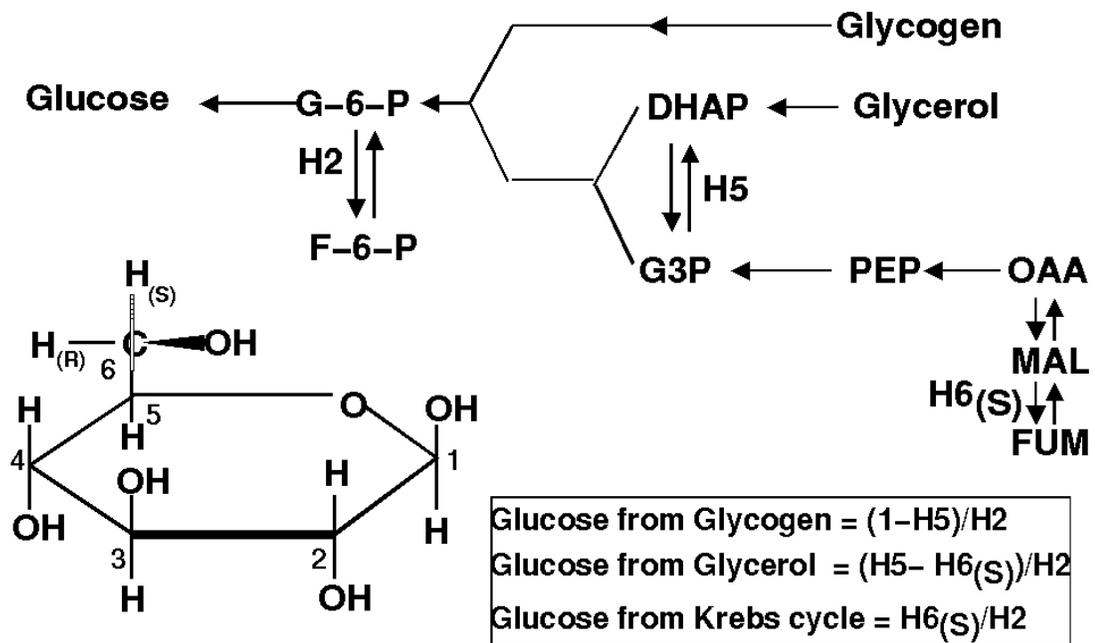


Figure 10.4: Peak intensities from H2, H5 and H6S resonances are used to calculate the contribution of Glycogen, Glycerol and the Krebs cycle to the creation of Glucose, see [34]. This figure courtesy of John G. Jones, South Western Medical Center.

Figure 10.5: Bayes Metabolite IPGD\_D20 Spectrum

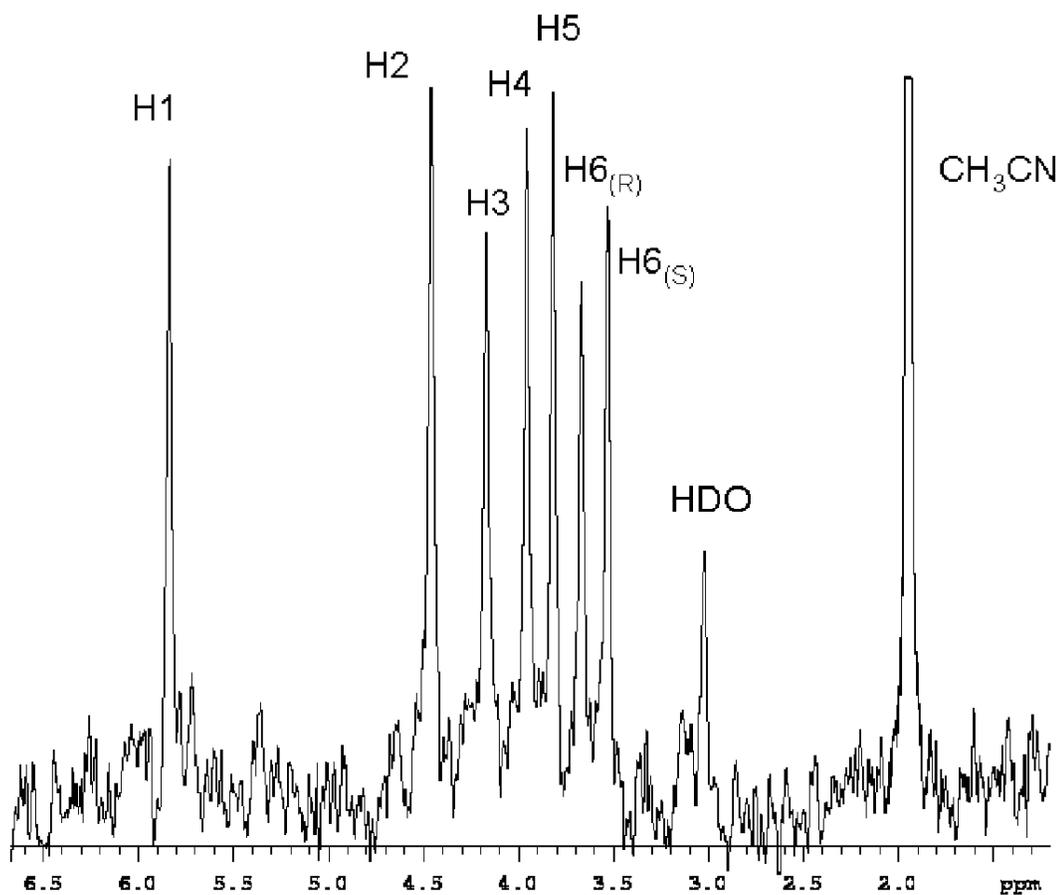


Figure 10.5: This spectrum is a typical example of IPGD\_D20 data. It is the ratios, as described in the text, of the labeled four peak that are computed by the IPGD\_D20 metabolic package. Note the four metabolic peaks of interest have been annotated. The remaining resonances, including the solvent, are described by the IPGD\_D20.Res file and are the intensities of these peaks are not used in the Metabolic analysis.

Figure 10.6: The Fraction Of Glucose From Glycogenolysis, Glycerol And The Krebs Cycle

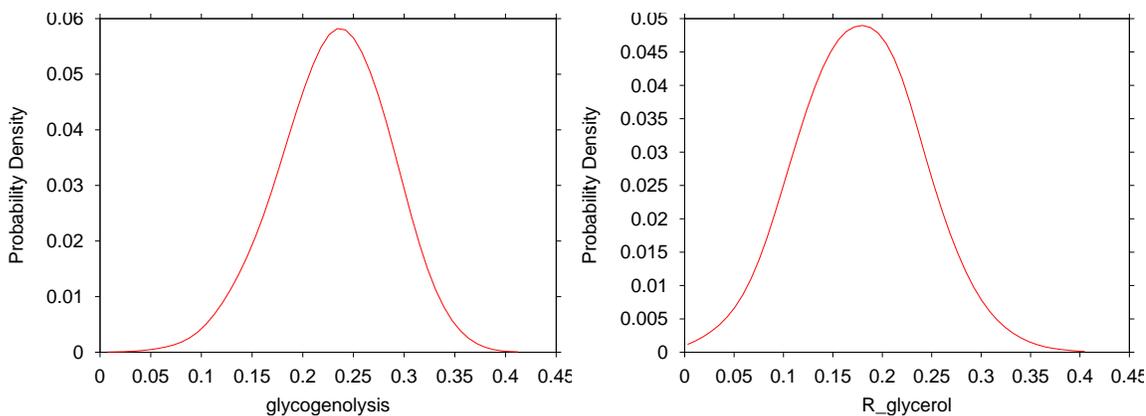
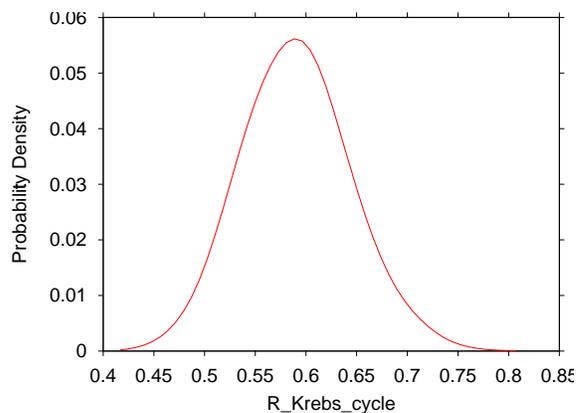


Fig. 10.6. These three plots are of the posterior probabilities for the fraction of glucose from glycogenolysis, above, from glycerol above right, and from the Krebs cycle, right. The widths of these plots are a natural measure of how uncertain one is of a particular parameter. For example the glycogenolysis fraction has been estimated to be roughly  $0.25 \pm 0.05$ .



metabolite analysis:

$$\begin{aligned}
 \text{Spins\_H2} &= \text{H2}, \\
 \text{glycogenolysis} &= 1 - \frac{\text{H5}}{\text{H2}}, \\
 \text{R\_glycerol} &= \frac{\text{H5} - \text{H6}_{(R)}}{\text{H2}}, \\
 \text{R\_Krebs\_cycle} &= \frac{\text{H6}_{(R)}}{\text{H2}}, \\
 \text{S\_glycerol} &= \frac{\text{H5} - \text{H6}_{(S)}}{\text{H2}}, \\
 \text{S\_Krebs\_cycle} &= \frac{\text{H6}_{(S)}}{\text{H2}}.
 \end{aligned}
 \tag{10.26}$$

Using the data shown in Fig. 10.5, and the IPGD\_D2O.ISO and IPGD\_D2O.Res files we ran this analysis and have displayed probability density functions for the three ratios of interest, the “S”

ratios, are shown in Fig. 10.6. For more on these calculations and why they are important see [34].

### 10.3.2 The Glutamate.2.0 Metabolite

The Glutamate 2.0 metabolic model is a model of the TCA cycle in hearts. This model was first proposed by Jeffrey, et. al. [42]. The assumption included in this model necessitate that only the perfusion described in Jeffrey, et. al. be utilized. Carbon-13 spectroscopic analysis of the glutamate isotopomer can be carried out on the unpurified abstract of the freeze clamped heart, with two caveats. The spectroscopy should be at 100 MHz carbon frequency or above to minimize second order effects in the multiplets, and the relaxation delay should be 2.5s with a 90 degree excitation pulse so that different resonances are not weighted by relaxation effects. A spectrum typical of this experiment is displayed in Fig. 10.7. The four panels in this figure are the resonances that each of the four carbon isotopomer gives rise to. The function of the Glutamate metabolic model is to predict the relative intensity of each of the resonances within each isotopomer.

For a heart perfused with [2-<sup>13</sup>C] acetate plus [3-<sup>13</sup>C] propionate, there are three basic metabolite parameters:

- $F_{c2}$ : The fraction of acetyl-CoA enriched in C2.
- $y$ : Anaplerosis, Carbon skeleton entry into the TCA acid cycle relative to citrate synthase; by definition  $y \geq 0$ .
- $F_{a1}$ : Anaplerotic substrate enriched in a single aliphatic carbon.

From these three parameters, there are two derived parameters:

- $F_{c0}$ : By definition if  $F_{c2}$  is the fraction of acetyl-CoA enriched in C2, and  $F_{c0}$  is the fraction not enriched, then

$$F_{c0} + F_{c2} = 1. \quad (10.27)$$

- $F_{a0}$ : Similarly if  $F_{a1}$  is the Anaplerotic substrate enriched in aniphatic carbon, and  $F_{a0}$  is the fraction not enriched, then

$$F_{a0} + F_{a1} = 1. \quad (10.28)$$

These five metabolic parameters are used by the Glutamate.2.0 subroutine to predict the relative intensity of each resonance. In the process of computing the joint posterior probability for the model, the simulations will propose a set of metabolic parameters. These are passed to the Glutamate.2.0 subroutine, and that subroutine passes back the predicted relative amplitudes of each resonance in the model.

The Glutamate.2.0 model contains 11 resonances which interact through three coupling constants:

- $J_{12}$ : The coupling constant between the C1 and C2 carbons,
- $J_{23}$ : The coupling constant between the C2 and C3 carbons,
- $J_{34}$ : The coupling constant between the C3 and C4 carbons.

Figure 10.7: Glutamate Example Spectrum

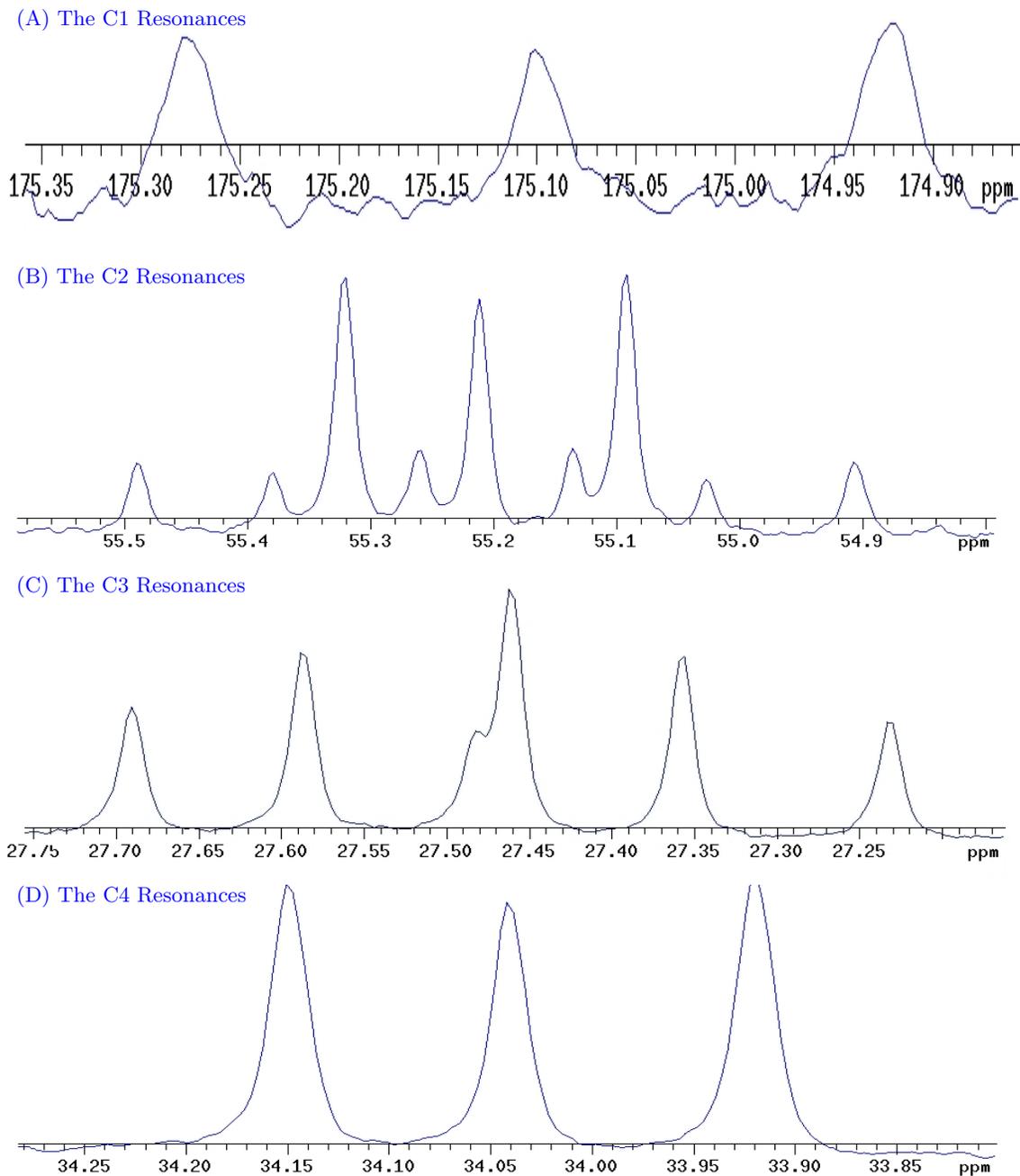


Figure 10.7: This spectrum is a typical example of Glutamate data: C1 is a singlet and a doublet; C2 is a doublet of doublets, two doublets and a singlet; C3 is a doublet of doublets, a doublet, and a singlet; finally, C4 is a doublet and a singlet.

The prior ranges, means, and standard deviations of these coupling constants are read from the “Glutamate.2.0.ISO” file. For example the  $J_{12}$  coupling constant is  $54 \pm 1$  Hertz, with a low and high of 52 and 56 Hertz respectively. Typically these coupling constants are determined to better than 0.01 Hertz, so the priors have little effect on the numerical simulations; they serve only to make sure that the Markov chain keeps these coupling constants near the values supported by the data.

In the Glutamate.2.0 model there are four sites and so four amplitudes that must be estimated. The Glutamate.2.0 subroutine computes the normalized intensity of each resonance within each sites from the metabolic parameters. The four sites are named C1, C2, C3 and C4. So when we say that the Glutamate.2.0 subroutine must compute the fractional intensity of a sites we mean that each carbon gives rise to a set of related lines and total amplitude of an sites times the fractional intensity of a given resonance is equal to the intensity of that resonance.

These four carbons give rise to a total of 11 resonances. Figure 10.7 shows the resonances from each of the four carbons. These resonances are named  $Cn_x$  where  $Cn$  is the carbon giving rise to the resonance, and  $x$  designates which resonance. For example,  $C2_{d_{23}}$ , is the C2 doublet caused by the  $J_{23}$  coupling. As noted, the Glutamate.2.0 subroutine must compute the fractional intensity of each resonance from the metabolic parameters. If we define

$$a = \frac{1}{y+1}, \quad (10.29)$$

$$b = \frac{1}{2(y+1)^2}, \quad (10.30)$$

and

$$c = bF_{c2} \quad (10.31)$$

then the fractional intensities for the two C1 resonances, shown in Fig. 10.7(A), are given by:

$$\begin{aligned} C1_d &= aF_{c2}, \\ C1_s &= 1 - C1_d \end{aligned} \quad (10.32)$$

where  $C1_d$  is the fractional intensity of the C1 doublet due to  $J_{12}$  coupling and  $C1_s$  is the fractional intensity of the C1 singlet. Note, by definition  $C1_d + C1_s = 1$ .

Similarly, the fractional intensities of the four C2 resonances, shown in Fig. 10.7(B), are given by

$$\begin{aligned} C2_{d_{23}} &= c(2y + F_{c0} + 1), \\ C2_{d_{12}} &= cF_{c0}, \\ C2_q &= cF_{c2}, \\ C2_s &= 1 - C2_{d_{12}} - C2_{d_{23}} - C2_q \end{aligned} \quad (10.33)$$

where  $C2_{d_{23}}$  is the fractional intensity of the C2 doublet due to  $J_{23}$  coupling,  $C2_{d_{12}}$  is the fractional intensity of the C2 doublet due to  $J_{12}$  coupling,  $C2_q$  is the fractional intensity of the C2 doublet of doublets due to  $J_{12}$  and  $J_{23}$  couplings, Finally,  $C2_s$  is the fractional intensity of the C2 singlet; and, as with the C1 resonances, the fractional intensity of the C2 singlet is computed in such a way as to insure  $1 = C2_{d_{12}} + C2_{d_{23}} + C2_q + C2_s$ .

The fractional intensities of the three C3 resonances, shown in Fig. 10.7(C), are given by

$$\begin{aligned} C3_d &= C1_d(y + 2F_{c0}), \\ C3_t &= C1_d F_{c2}, \\ C3_s &= 1 - C3_d - C3_t \end{aligned} \tag{10.34}$$

where  $C3_d$  is the fractional intensity of the C3 doublet due to  $J_{23}$  coupling,  $C3_t$  is the fractional intensity of the C3 doublet of doublets due to  $J_{23}$  and  $J_{34}$  couplings. Note this doublet of doublet is often called a triplet because the coupling is such that the resonance pattern looks like a 1:2:1 triplet. Finally,  $C3_s$  is the fractional intensity of the C3 singlet, and as with the C1 and C2 resonances its intensity is computed in such a way as to insure  $1 = C3_d + C3_t + C3_s$ .

The fractional intensities of the two C4 resonances, shown in Fig. 10.7(D), are given by

$$\begin{aligned} C4_{d_{34}} &= (F_{c2} + yF_{A1})/(2y + 1) \\ C4_s &= 1 - C4_{d_{34}} \end{aligned} \tag{10.35}$$

where  $C4_{d_{34}}$  is the fractional intensity of the C4 doublet due to  $J_{34}$  coupling, and  $C4_s$  is the the fractional intensity of the C4 singlet.

To use the glutamate model, load the “Glutamate.2.0.ISO” isotopomer file and reference the C1 singlet to 175.1 ppm. Next hit the display spectrum button and the interface will note the locations of all of the prior probabilities on the spectrum. These priors should now overlap the centers of the C1 resonances. If they don’t then you must not have gotten the referencing correct or something is wrong with the spectrum. Next expand the various regions of the spectrum and make sure that the priors overlap the centers of each of the carbon resonances. If they do not overlap, and you have referenced the spectrum, you will have to use the edit metabolic resonances button to correct the frequency shifts. Note that you must be careful in doing this to make sure that the resonances remain in the given order. This can be difficult, because the interface orders the priors by frequency, and if you change a frequency, you can change the order. If the order changes from what is coded into the glutamate subroutine, the Metabolite program will not even try to run. After making sure the resonances all line up, load the nuisance resonances. These nuisance resonances are in the “Glutamate.2.o.Res” file. This particular file was set up for the “sb33.90A.fid” located in the Bayes.test.data directory downloaded from the interface. If the nuisance resonances in your data set are different, unload the resonance file and use the edit parameters button under the processing type resonance label to enter the nuisance resonances. The easiest way to do this is simply to mark them using the buttons on the edit resonances window. Finally, run the analysis using the “Run Analysis” button.

### 10.3.3 The Glutamate.3.0 Metabolite

The newest metabolite to be added to the metabolite package is an extension of the glutamate 2.0 metabolite to account for the C5 couplings. This is an extensive addition to the metabolite package having some 19 metabolic parameters, 6 coupling constants and 27 sets of resonances. These resonances range from singlets to doublets of doublets of doublets. If you need more information on this metabolite consult the paper [41] and if this does not answer you questions contact its creator, [Mark Jeffrey](#), at the Southwest medical center in Dallas Texas.

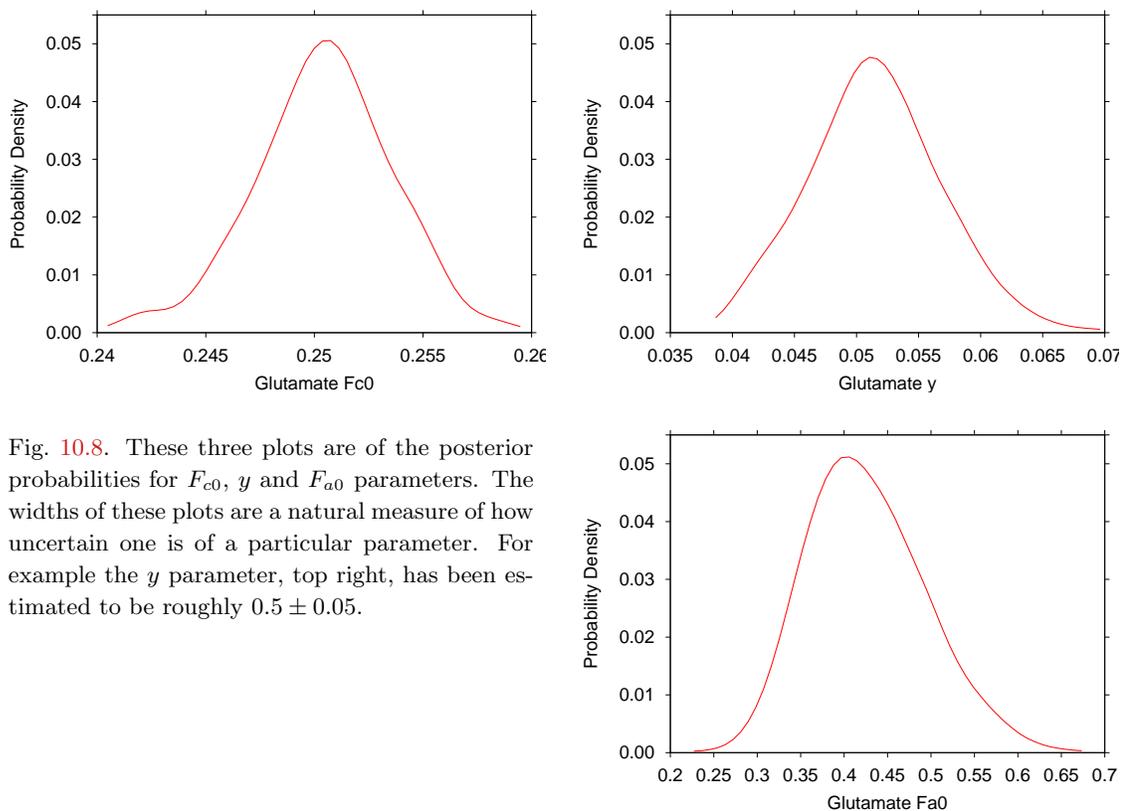
Figure 10.8: Estimating The  $F_{c0}$ ,  $y$  and  $F_{a0}$  Parameters

Fig. 10.8. These three plots are of the posterior probabilities for  $F_{c0}$ ,  $y$  and  $F_{a0}$  parameters. The widths of these plots are a natural measure of how uncertain one is of a particular parameter. For example the  $y$  parameter, top right, has been estimated to be roughly  $0.5 \pm 0.05$ .

## 10.4 The Example Metabolite

The example metabolite is just what its name implies, its a simple example of a metabolite used to demonstration how to build metabolite model. This metabolite imposes only a single condition on the data, in this case it forces the fractional amplitudes of two multiplets to be in a fixed ration. The data used in the example is the “ethyl.ether.fid” and may be loaded from the Bayes.test.data directory. The spectrum of this data is shown in Fig. 10.9(A). To run the example metabolite, load the Fid, the “Example.ISO” file and the nuisance resonance file “Example.Res” and run the analysis.

The example metabolite only runs a few minutes. It has one metabolite and one derived parameter. In this case it is the fractional intensity of the triplet that is the metabolite parameters, the derived parameter is the fractional intensity of the quartet. The output from this metabolite is the two fractional intensities, the total intensity, the coupling constant, and the parameters needed to describe both the metabolite and nuisance resonances. These outputs may be viewed using the standard widgets. Additionally, the metabolite model generated from the maximum posterior probability estimate of the parameters may be viewed by activating the Fid model viewer. see Section 3.19 for more on how to use the model experiment.

Figure 10.9: Bayes Metabolite, The Ethyl Ether Example

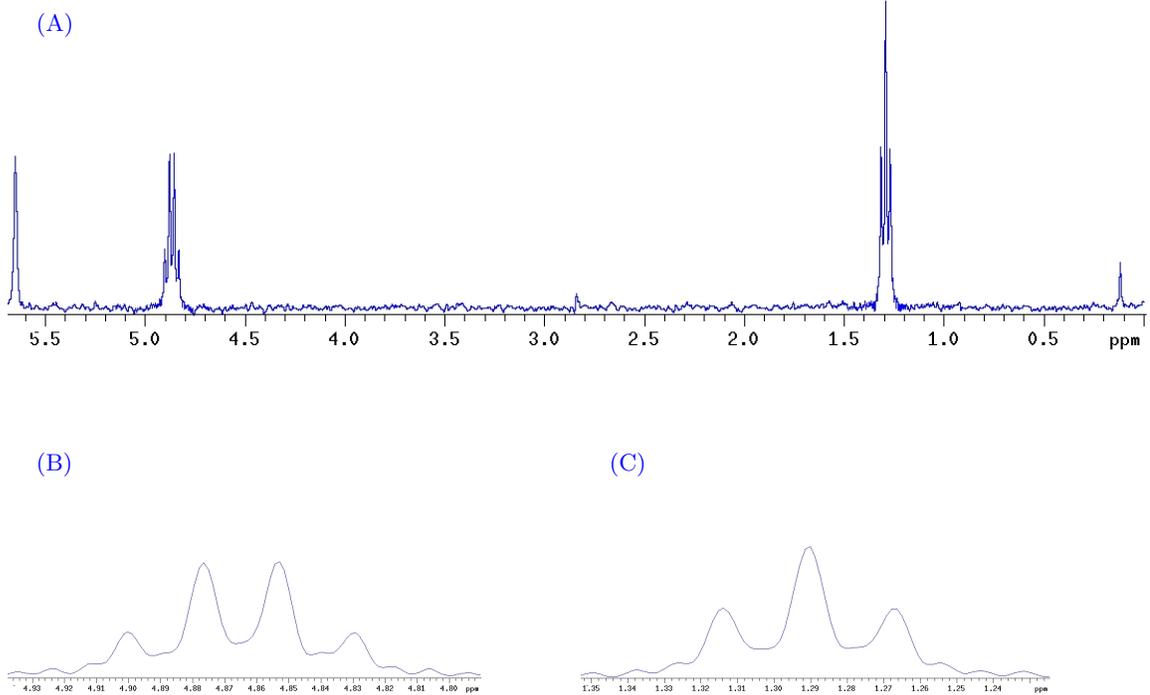


Figure 10.9: Panel (A) shows the full spectrum of the example metabolite, ethyl ether. The metabolite resonance, Panels (B) and (C), are described in the in the Example.ISO file. Metabolites must predict the fractional intensities of the metabolite resonances. Here that means the example subroutine imposes the condition that the area of the quartet should be  $2/5$  while the area of the triplet should be  $3/5$ .

## 10.5 Outputs From The Bayes Metabolite Package

The Text outputs files from the Metabolite packages consist of: “Bayes.prob.model,” “BayesMetabolite.mcmc.values,” “Bayes.params,” “Console.log,” “Bayes.accepted” and a “Bayes.Condensed.File.” These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the mcmc.values report is discussed in Appendix D and the other reports are discussed in Chapter 3. Additionally, the “Plot Results Viewer” can be used to view the output probability density functions. In addition to the standard data, model and residual plots there are probability density functions for each metabolic parameter and for each frequency, J coupling constant, and decay rate constant in the model. Finally there is probability density functions for the standard deviation of the noise in the time domain F<sub>id</sub> data.

## Chapter 11

# Find Resonances

There are two frequency finding programs in the Bayesian Analysis Software: Bayes Analyze, Chapter 8, and Bayes Find Resonance. Bayes Analyze is a searching algorithm that uses the residuals from the current fit, to determine if there is evidence in the data for and additional resonance. While this procedure is implemented using Bayesian probability theory, it is still an approximation to computing the full Bayesian posterior probability for the number of resonances. We implemented Bayes Analyze in this way, so that it would be very fast. However, under some conditions, Bayes Analyze will miss resonances when they are either very close together or the signal-to-noise of the resonance is very low. To solve these problems, we implemented a frequency finding program that uses Markov chain Monte Carlo to compute the posterior probability for the number of resonances. The interface to the find resonance package is shown in Fig. 11.1 To use this package, you must do the following:

**Select** the Bayes Find Resonances package from the Package menu.

**Load** the Fid data that is to be analyzed. If the Fid is arrayed, select the trace that is to be analyzed. The trace analyzed is the currently displayed trace. At the present time only a single Fid is processed by Bayes Find Resonances package.

**Select** the phase model, the choices are correlated, uncorrelated and automatic.

- If the phase model is “Common,” the all resonances have the same phase.
- If the phase model is “Independent,” the all resonances have a different phase.
- If the phase model is “Automatic,” then the Bayes Find Resonances package computes the posterior probability for the phase of each resonance.

**Check** the “Constant” box if the data contains an offset.

**Set** the first and last Fids that are to be analyzed. Note that these Fids are analyzed separately, not jointly, so you will get an analysis for each selected Fid.

**Set** the maximum number of resonances that can be included in a model.

**Select** the server that is to process the analysis.

Figure 11.1: The Find Resonances Interface With The Ethyl Ether Spectrum

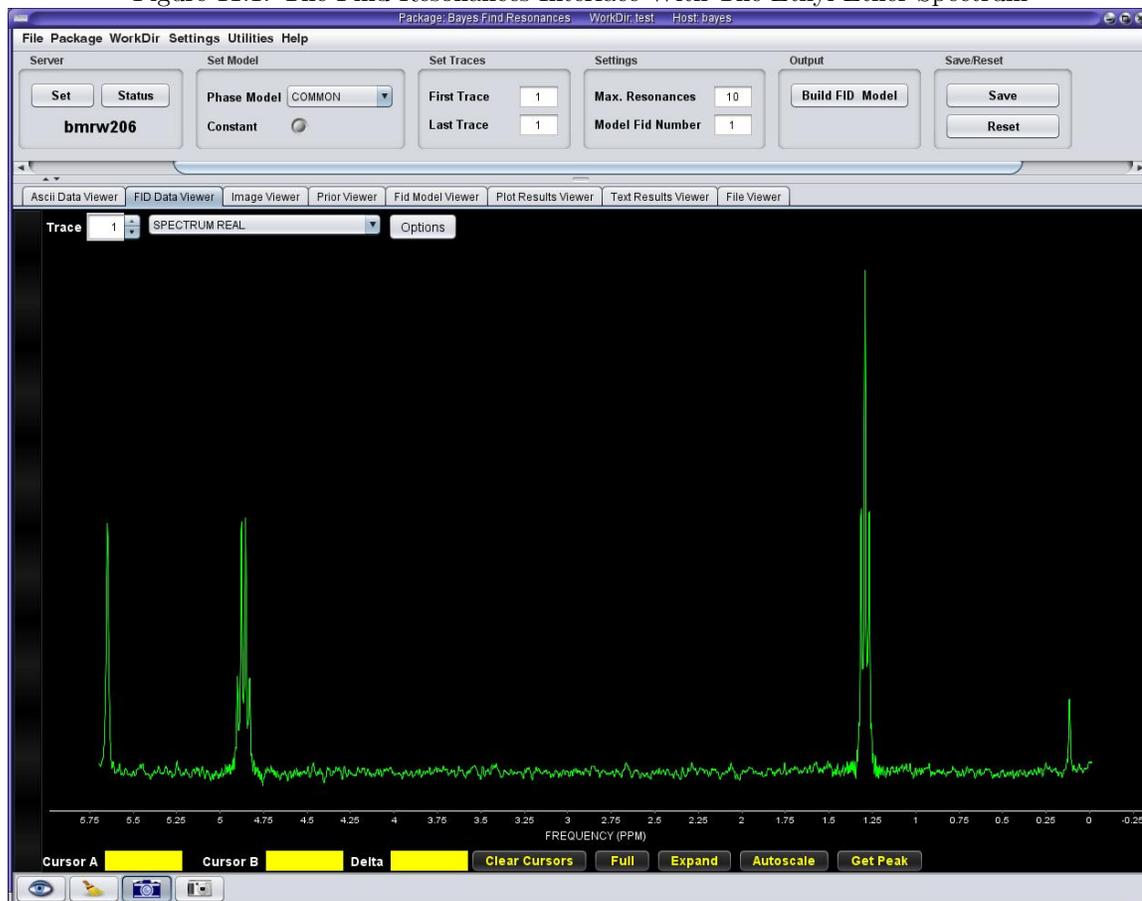


Figure 11.1: When the Find Resonances package is selected, this is the displayed interface. To use this package, load the Fid you wish to analyze. The spectrum of this Fid will be displayed in the Fid Data Viewer. Select the Fid you wish to analyze and display that Fid. At the present time only a single Fid may be processed at one time. Set the various optional feature of the model you wish to use and run the analysis. When the analysis finishes use the “Build Model” button to select and build a model of the Fid. The Fid Model Viewer can then be used to view this model.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the analysis on the selected server by activating the “Run” button.

**Get** the results of the analysis by activating the “Get Job” button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

Unlike Bayes Analyze which outputs parameter estimates computed using the values that maximized the joint posterior probability for the parameters, Bayes Find Resonances outputs mean and standard deviation parameter estimates computed from all high probability models. That is to say if the probability for the number of resonances was 50% for 9 and 50% for 10 resonances, then there will be mean and standard deviation parameter estimates for the frequencies and decay rate constants from both of these models. Additionally, Bayes Find Resonances runs multiple Fids one right after the other. Consequently, when a Fid model of the time domain data is generated, you must specify both the resonance model and the number of the Fid to model. The Fid number to model is indicated using the “Model Fid Number” entry box. If there are multiples high probability resonance models, clicking on the “Build FID Model” button will show you a list of these models and you can select which one you wish to use in generating a time domain Fid model.

## 11.1 The Bayesian Calculations

The first step in all Bayesian calculations is to define the problem. Here, the problem is essentially a parameter estimation calculations where one is estimating the frequencies, amplitudes, decay rate constants and phases of multiple exponentially decaying sinusoidal. The model will designated as  $\mathbf{M}(\mathbf{t}_i)$  where complex quantities will be in bold. Then symbolically the model which relates the complex data, model and noise is given by:

$$\mathbf{d}_i = \mathbf{M}(\mathbf{t}_i) + \mathbf{n}_i \quad (i \in \{1, \dots, N\}), \quad (11.1)$$

where  $N$  is the total number of complex data values,  $\mathbf{d}_i$  is a complex data values sampled at time  $t_i$ , and  $\mathbf{n}_i$  is a complex noise value at time  $t_i$ . The complex model  $\mathbf{M}(\mathbf{t}_i)$  is given by:

$$\mathbf{M}(\mathbf{t}_i) = [\mathbf{F}\delta(t_i) + \mathbf{C}] \delta(\nu) + \sum_{j=1}^m A_j \exp \{2\pi i f_j(t_i - t_0) - \alpha_j t_i + i\phi\delta(\xi_j) + i\phi_j[1 - \delta(\xi_j)]\} \quad (11.2)$$

where  $\mathbf{F}$  is a model of the first data value, the function  $\delta(\cdot)$  is defined below,  $\mathbf{C}$  is a complex offset,  $m$  is the unknown number of sinusoids,  $A_j$  is the amplitude of the  $j$ th sinusoid,  $f_j$  is the frequency of the  $j$ th sinusoid,  $t_0$  is a first order phase correction,  $\alpha_j$  is the decay rate constant of the  $j$ th sinusoid. The quantity “ $\phi\delta(\xi_j) + \phi_j[1 - \delta(\xi_j)]$ ” is the phase of the  $j$  sinusoid and is either a common zero order phase,  $\phi$ , or a unique phase specific to the  $j$ th sinusoid,  $\phi_j$ . Whether or not the phase is common or unique depends on the value of the indicator function  $\delta(\xi_j)$ . The indicator function  $\delta(\cdot)$  is defined as

$$\delta(\nu) = \begin{cases} 1 & \text{If } \nu = 0 \\ 0 & \text{Otherwise} \end{cases} \quad (11.3)$$

so, for example, the quantity  $\mathbf{F}\delta(t_i)$  is present only when  $t_i = 0$ . Similarly, the common phase  $\phi$  is present only when  $\xi_j = 0$ , where  $\xi_j$  is a two value binary variable defined as:

$$\xi_j = \begin{cases} 1 & \text{If the } j\text{th sinusoid has a common zero order phase} \\ 0 & \text{Otherwise} \end{cases} . \quad (11.4)$$

Here common zero phase means that several sinusoids share the same zero order phase parameter.

The value of  $\xi_j$  is under user control. If the user selects the ‘‘Common’’ phase model, then  $\xi_j = 0$  for all sinusoids and all sinusoids share the same common zero order phase parameter. If the user selects the ‘‘Independent’’ phase model, then  $\xi_j = 1$  for all sinusoids and all sinusoids have a unique zero order phase parameter  $\phi_j$ . Finally, if the user selects the phase model as ‘‘Independent’’ then the parameters  $\xi_j$  are binary variables that are simulated in the Markov chain Monte Carlo simulation, i.e., the Bayes Find Resonances package automatically determines which resonances have common zero order phase and which have a unique zero order phase.

Whether or not the constant models are present is also under user control. If the ‘‘Constant’’ check box is activated, then  $\nu$  is set equal to zero by the package and  $\delta(\nu) = 1$  and in Eq (11.2) the constant models are present. If the constant check box is not active then  $\nu = 1$  and no constants are present in Eq (11.2). However, unlike the phase model, the Bayes Find Resonances package does not simulate the binary variable  $\nu$ , this value is set by the user and the package uses the indicated value.

The Bayes Find Resonances package is a hybrid parameter estimation and model selection package. It is model selection in that it must determine how many resonances are present, and when the phase model is selected as ‘‘Independent’’ it must also estimate the binary parameters,  $\xi_j$ . So, the set of parameters estimated by Bayes Find Resonances package when all parameters are active, is:

**F** is the complex first point model, it contains two constants  $F_R$  and  $F_I$ , the real and imaginary first point parameters.

**C** is the complex constant offset model, it contains two constants  $C_R$  and  $C_I$ , which are the real and imaginary offset parameters.

$m$  is the unknown number of sinusoids in the data.

$A$  is the collection of amplitudes in the  $m$  sinusoids, so  $A \equiv \{A_1, \dots, A_m\}$ .

$f$  is the collection of frequencies in the  $m$  sinusoids, so  $f \equiv \{f_1, \dots, f_m\}$ .

$\alpha$  is the collection of decay rate constants in the  $m$  sinusoids, so  $\alpha \equiv \{\alpha_1, \dots, \alpha_m\}$ .

$t_0$  is the first order phase in in the  $m$  sinusoids.

$\xi$  is the collection of phase model indicators in the  $m$  sinusoids, so  $\xi \equiv \{\xi_1, \dots, \xi_m\}$ .

$\phi$  is the common zero order phase in the  $j$ th sinusoids when  $\delta(\xi_j) = 1$ .

$\phi_j$  is the zero order phase in the  $j$ th sinusoids when  $\delta(\xi_j) = 0$ .

We are going to designate this collection of parameters as  $\Phi$  and then proceed with the Bayesian calculations.

The Bayesian calculations are for the posterior probability for the number of resonances in the data set. This posterior probability is designated as  $P(m|DI)$ , where  $D$  represents all of the data and  $I$  stands for all of the prior information. This posterior probability is computed by application of Bayes' Theorem:

$$P(m|DI) = \frac{P(m|I)P(D|mI)}{P(D|I)} \quad (11.5)$$

where  $p(m|I)$  is the prior probability for the number of resonances,  $P(D|mI)$  is the marginal direct probability for the data given the model order and  $P(D|I)$  is a marginal direct probability for the data given only the prior information  $I$ . The direct probability for the data given only the prior information,  $P(D|I)$ , is a normalization constant and is given by:

$$\begin{aligned} P(D|I) &= \sum_{m=1}^{\text{Max}} P(Dm|I) \\ &= \sum_{m=1}^{\text{Max}} P(m|I)P(D|mI) \end{aligned} \quad (11.6)$$

where the maximum number of resonances is designated as "Max." Comparing, Eq. (11.6) to Eq. (11.5) it is easy to see that  $P(D|I)$  is a normalization constant. If we normalize the posterior probability for the number of resonances,  $P(m|DI)$ , at the end of the calculations, then Eq (11.5) becomes:

$$P(m|DI) \propto P(m|I)P(D|mI). \quad (11.7)$$

The prior probability for the number of resonances,  $P(m|I)$ , is sufficiently simplified that we could assign it a numerical value. For now we will simply leave it in this symbolic form. However, the direct probability for the data given the number of resonances and the prior information,  $P(D|mI)$ , is not yet sufficiently simplified so that its value can be assigned. To proceed with the calculation, one introduces the collection of parameters  $\Phi$  into this probability, the the direct probability for the data,  $P(D|mI)$ , the posterior probability for the number of resonances becomes

$$P(m|DI) \propto P(m|I) \int P(D\Phi|mI)d\Phi. \quad (11.8)$$

One proceeds by applying the product rule to the right-hand side of this equation:

$$P(m|DI) \propto P(m|I) \int P(\Phi|I)P(D|\Phi mI)d\Phi. \quad (11.9)$$

Factoring the prior probability for all of the parameters,  $P(\Phi|I)$  into individual prior probabilities

for each parameter, one obtains:

$$\begin{aligned}
P(m|DI) &\propto P(m|I) \int P(F_R|I)P(F_I|I)P(C_R|I)P(C_I|I) \\
&\times P(\phi|I)P(t_0|I)P(D|\Phi mI) \\
&\times \left[ \prod_{j=1}^m P(A_j|I) \right] \\
&\times \left[ \prod_{j=1}^m P(f_j|I) \right] \\
&\times \left[ \prod_{j=1}^m P(\alpha_j|I) \right] \\
&\times \left[ \prod_{j=1}^m P(\xi_j|I) \right] \\
&\times \left[ \prod_{j=1}^m P(\phi_j|I)^{\delta(\xi_j)} \right] d\Phi.
\end{aligned} \tag{11.10}$$

However, this is the marginal posterior probability for the number of resonances, and is the main output from the Bayes Find Resonances package. But the quantity sampled in the Markov chain Monte Carlo simulation is the joint posterior probability for all of the parameters. While this posterior probability is very similar, its not quite the same. The joint posterior probability for all of the parameters is given by:

$$\begin{aligned}
P(m\Phi|DI) &\propto P(m|I)P(F_R|I)P(F_I|I)P(C_R|I)P(C_I|I) \\
&\times P(\phi|I)P(t_0|I)P(D|\Phi mI) \\
&\times \left[ \prod_{j=1}^m P(A_j|I) \right] \\
&\times \left[ \prod_{j=1}^m P(f_j|I) \right] \\
&\times \left[ \prod_{j=1}^m P(\alpha_j|I) \right] \\
&\times \left[ \prod_{j=1}^m P(\xi_j|I) \right] \\
&\times \left[ \prod_{j=1}^m P(\phi_j|I)^{\delta(\xi_j)} \right]
\end{aligned} \tag{11.11}$$

which is Eq. (11.10) without the integrations. The prior probabilities are assigned as follows:

$P(m|I)$  is assigned as an Exponential prior with  $(0 \leq m \leq 50)$  where a model containing 50 resonances is a hard-coded maximum.

$P(F_R|I)$  is assigned as a bounded Gaussian prior whose range is  $\pm 6 \times$  the amplitude of the first data value and whose standard deviation is three times the magnitude of the first data value.

$P(F_I|I)$  is assigned the same as  $P(F_R|I)$  was assigned.

$P(C_R|I)$  is assigned as a bounded Gaussian whose mean is equal to the average of the last 10 real data values and whose upper and lower bounds is 5 times the mean. Finally, the standard deviation of this prior is one fifth of the prior range.

$P(C_I|I)$  is assigned like  $P(C_R|I)$  except the means and bounds are taken from the imaginary channel.

$P(\phi|I)$  is assigned a uniform prior probability ranging from zero to  $2\pi$ .

$P(t_0|I)$  essentially sets a time when the phases of all the sinusoids are the same. The prior probability for this time offset has a mean of zero, meaning that the phases are all expected to be the same at the start of the acquisition. However, we allow this parameter to range over  $\pm 5$  dwell or sampling times, i.e., the zero of time could occur the equivalent of 5 data values before the start of the acquisition and up to 5 data value past the start of acquisition. However, the standard deviation of this Gaussian prior probability is only 0.3 data values. Indicating, that while  $T_0$  is allowed to have a large range, in fact it is strongly suspected that its value is near zero.

$P(A_j|I)$  is assigned a bounded Gaussian prior probability for one of the amplitudes. Its mean is zero, its upper and lower bounds are 3 times the average magnitude of the first 10 complex data values and its standard deviation is two times that average. This prior is used for the prior probability for each resonance amplitude in the model.

$P(f_j|I)$  is assigned a uniform prior probability ranging over the entire sweep width of the data. This prior is used for the prior probability for each resonance frequency in the model.

$P(\alpha_j|I)$  is assigned a positive prior probability whose low is 0.001 in dimensionless units. The peak value is set to the current value of the “lb” parameter. The maximum value is set so that if the signal were decaying at this maximum, it will go through roughly 3 e-foldings in the first 9 data values. So the maximum value is strongly dependent on the sampling rate.

$P(\xi_j|I)$  is assigned a discrete uniform prior probability having two possible values, zero or one with zero indicating an independent phase and one indicating a common phase.

$P(\phi_j|I)$  is assigned a uniform prior probability ranging from zero to  $2\pi$ . In the above equations, [11.10](#) and [11.11](#), this probability is written as  $P(\phi_j|I)^{\delta(\xi_j)}$ , which is just a notational mechanism to indicate that the prior is either present or not. When  $\delta(\xi_j) = 1$ ,  $\xi_j = 0$ , the phase model is independent and the prior is present. Similarly, when  $\xi_j = 1$  the phase model is common and this prior is not present because the prior is raised to the zero power.

$P(D|\Phi m I)$  is the direct probability for the data given all of the parameters and the prior information and is assigned using a Gaussian prior probability for the noise. This probability is often called a likelihood or likelihood function.

The Markov chain simulation that implements this calculation targets the joint posterior probability for all of the parameters in the model, Eq. [\(11.11\)](#). It then uses Monte Carlo integration to obtain samples from the marginal posterior probability for each parameter appearing in the model. The samples from the marginal posterior probability for each parameter are then used to generate mean and standard deviation estimates of each parameter appearing in the model. Additionally, the samples are used to generate histograms. These histograms are crude estimates of the posterior probability for each parameter. In addition to outputting the histograms, the samples are also output and these samples can be used to generate Maximum Entropy histograms of the samples, see

Section (25). Finally, these samples are used to compute the posterior probability for the number of resonances, Eq. (11.10).

If there are multiple high probability models in this posterior, i.e., the posterior probability for the number of resonances,  $P(m|DI)$ , has significant weight on several values of  $m$ , then samples are drawn for each high probability model and these samples are used to generate histograms and parameter estimates given each high probability model. Consequently, then the algorithm finishes it is possible to have not one set of outputs but several. These outputs are viewed using the standard widgets. Additionally, it is possible to generate time domain FID models for each high probability model.

## 11.2 Outputs From The Bayes Find Resonances Package

The Text outputs files from the Find Resonance packages consist of: “Bayes.prob.model,” “BayesFind-Res.mcmc.values,” “Bayes.params,” “Console.log,” “Bayes.accepted” and a “Bayes.Condensed.File.” These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the mcmc.values report is discussed in Appendix D and the other reports are discussed in Chapter 3. Additionally, the “Plot Results Viewer” can be used to view the output probability density functions. In addition to the standard data, model and residual plots there are probability density functions for the frequencies and decay rate constants, and the amplitudes for each resonance for each high probability model.

Finally, for each high probability model, there is an output “bayes.params.nnnn” and “bayes.model.nnnn” file where this file has exactly the same format as the Bayes Analyze Model file, see Chapter 8.5.7. These files are used in conjunction with the “Build FID Model” button to generate a time domain model of the input FID data. When this button is activated the FID and the selected bayes.model.nnnn file are sent to the server and the Bayes Model program, see Chapter 8.1 for a description of this program, is run. During this time the interface waits for the Bayes Model program to finish. When the interface detects that the model has been built, the interface fetches the model from the server, Fourier transforms the model and then displays the model using the FID Model Viewer.

## Chapter 12

# Diffusion Tensor Analysis

The diffusion tensor package estimates the parameters associated with diffusion tensor models. The data analyzed by this package are Ascii and may be input from an Ascii file, they can be loaded from the amplitudes resulting from a Bayes Analyze run or they may be input from an image pixel. The diffusion tensor package is accessed by selecting the “Packages/Diffusion Tensor” menu. When this button is activated the interface window shown in Fig. 12.1 is displayed. To use the Diffusion Tensor package, you must do the following:

**Check** the abscissa options appropriate for your input data.

**Load** one or more Ascii data sets using the Files menu. This data must have three abscissa for both “b” and “g” vector data. The format of a diffusion tensor data set is: data point number, data value,  $A_r$ ,  $A_p$ ,  $A_s$ , where “ $A_r$ ” is either a “B” or “G” value in the readout direction. Similarly, “ $A_p$ ” is a “B” or “G” value in the phase encode direction “ $A_s$ ” is a “B” or “G” value in the slice select direction. When a data set is successfully loaded it will be displayed in the Files Viewer.

**Set** the number of of tensors you wish to analyze. This number can be 1, 2 or 3 and is set using “Tensor Number” pull down menu. selection menu.

**Check** the “Include Constant” box if the data contains an offset.

**Check** the Analysis Options/Find Outliers box if you suspect outliers are present in the data.

**Review** the prior probabilities for the the various parameters in the model using the Prior Viewer.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

Figure 12.1: The Diffusion Tensor Package Interface

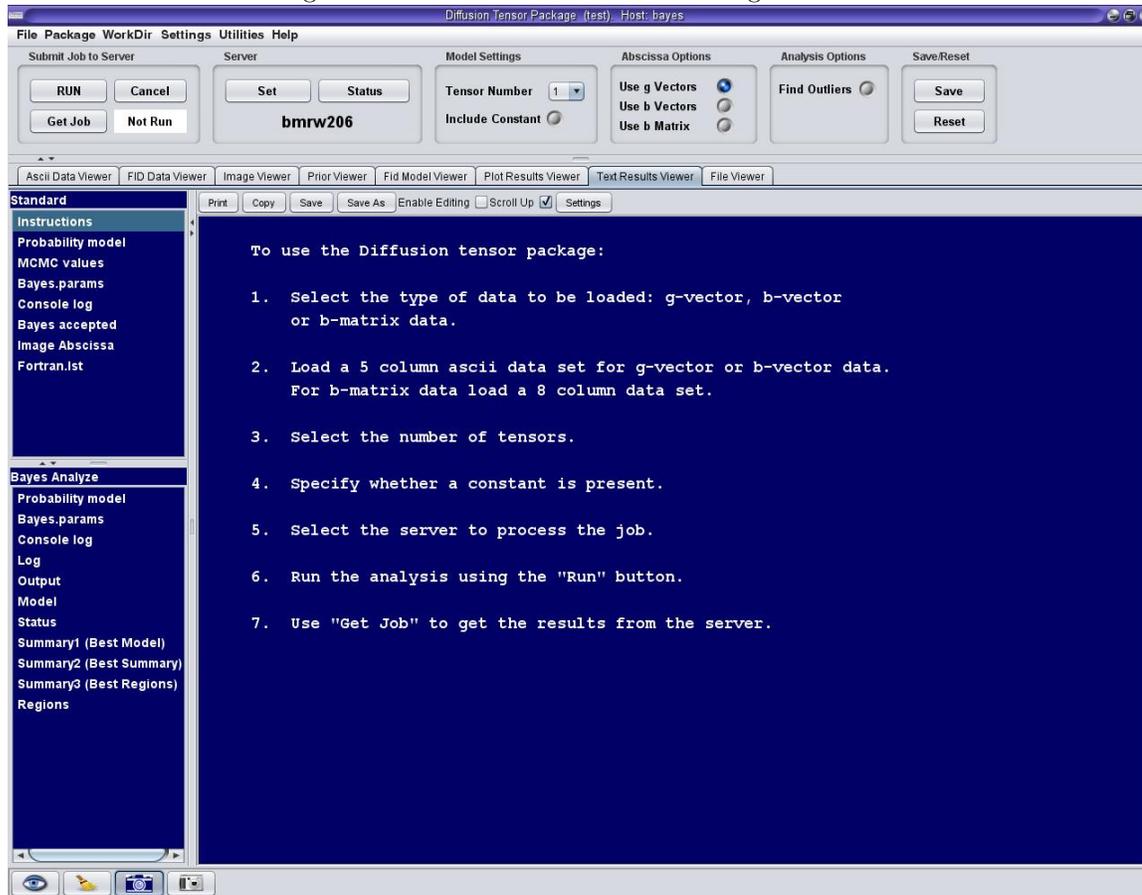


Figure 12.1: Diffusion Tensor Interface Fig. 12.1. The interface to the diffusion tensor package is shown here. To use this package one first selects the type diffusion tensor data to be analyzed, using the “Abcissa Options.” Depending on what check box is activated, the input data will have either 3 or 6 abcissa values. After the Abcissa Options are set, the data may be loaded, the number of tensors can then be selected and whether or not a constant offset is present. Finally, the analysis can be run using the “Run” button.

## 12.1 The Bayesian Calculation

A diffusion Tensor  $\mathbf{D}$  is a  $3 \times 3$  symmetric matrix having six independent elements:

$$\mathbf{D} = \begin{pmatrix} D_{rr} & D_{pr} & D_{sr} \\ D_{rp} & D_{pp} & D_{sp} \\ D_{rs} & D_{ps} & D_{ss} \end{pmatrix} \quad (12.1)$$

where there are three redundant elements,  $D_{rp} = D_{pr}$ ,  $D_{ps} = D_{sp}$  and  $D_{rs} = D_{sr}$ . Using this diffusion tensor, the data and the model are related to each other by

$$d_i = A \exp \left\{ -\text{Conv} \sum_{j=1}^3 \sum_{k=1}^3 g_{ij} D_{jk} g_{ik} \right\} + n_i \quad (12.2)$$

where  $d_i$  is the  $i$ th data value acquired using gradients ( $g_{i1}$ ,  $g_{i2}$ , and  $g_{i3}$ ), where the first, second and third directions are  $x$ ,  $y$  and  $z$  respectively, “Conv” is a conversion constant that ensures the argument of the exponential is unitless, and  $n_i$  represents the noise. For a rectangular gradient this conversion factor is given by:

$$\text{Conv} = \Gamma^2 \delta^2 (\Delta - \delta/3) \quad (12.3)$$

where  $\Gamma$  is the gyromagnetic ratio, usually of protons,  $\delta$  is the duration of the diffusion gradient pulses,  $\Delta$  is the time between the diffusion gradient pulses.

The diffusion tensor model, Eq. (12.2) without the noise, is what is used in most diffusion tensor analyzes. To estimate the diffusion matrix, the experimenter takes a number of diffusion weighted data values, typically 6, plus a data value having no diffusion gradients. This nondiffusion weighted data value is used as an estimate of the amplitude  $A$ , and the data are divided by this amplitude estimate. If the measurement is noiseless, and there are no other effects in the data, the amplitude is typically estimated from the first point of the data, and then divided out of the right-hand side of Eq. (12.2). Thus if one takes the logarithm, the resulting equations are linear in the elements of the diffusion matrix. One then solves the resulting set of linear equations for the diffusion matrix. Finally, an eigenvalue decomposition of the diffusion matrix results in an estimate of the magnitude of the diffusion along three primary directions. Here we are going to call these principal diffusion magnitudes  $\lambda_1$ ,  $\lambda_2$  and  $\lambda_3$  respectively. The eigenvectors are a rotation matrix  $\mathbf{R}$  characterized by three Euler angles and these Euler angles can also be estimated from the eigenvectors. Unfortunately, this procedure has many problems including, but not limited to the fact that the amplitude does not cancel because the measured amplitude is not noiseless. In some samples the diffusion data does not go to zero and, more importantly, this procedure can generate negative eigenvalues, a completely unphysical result because a real symmetric matrix must have three positive eigenvalues. Consequently, a different approach is needed to resolve these problems.

The negative eigenvalue problem can be solved by introducing the eigenvalues directly into the problem, and then using Bayesian probability theory and prior probabilities to prohibit negative eigenvalues. Here is how this is done. First, one introduces a diagonal diffusion tensor  $\mathbf{U}$  defined as:

$$\mathbf{U} = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \quad (12.4)$$

where  $\lambda_1$ ,  $\lambda_2$  and  $\lambda_3$  are the unknown magnitudes of the diffusion along an as yet unknown set of directions. Next one introduces the rotation matrix  $\mathbf{R}$  mentioned earlier. In this matrix the three Euler angles,  $\theta$ ,  $\phi$  and  $\psi$ , are unknowns that must be inferred:

$$\mathbf{R} = \begin{pmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{pmatrix} \times \begin{pmatrix} \cos \phi & \sin \phi & 0 \\ -\sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (12.5)$$

where we have expressed this rotation matrix as a product of the three rotations need rotate a diagonal matrix into any  $3 \times 3$  symmetric matrix. Multiplying the three matrices, one obtains:

$$\mathbf{R} = \begin{pmatrix} R_{rr} & R_{pr} & R_{sr} \\ R_{rp} & R_{pp} & R_{sp} \\ R_{rs} & R_{ps} & R_{ss} \end{pmatrix}, \quad (12.6)$$

with

$$R_{rr} = \cos(\psi) \cos(\theta) \cos(\phi) - \sin(\psi) \sin(\phi), \quad (12.7)$$

$$R_{pr} = \cos(\psi) \cos(\theta) \sin(\phi) + \sin(\psi) \cos(\phi), \quad (12.8)$$

$$R_{sr} = -\cos(\psi) \sin(\theta), \quad (12.9)$$

$$R_{rp} = -\sin(\psi) \cos(\theta) \cos(\phi) - \cos(\psi) \sin(\phi), \quad (12.10)$$

$$R_{pp} = -\sin(\psi) \cos(\theta) \sin(\phi) + \cos(\psi) \cos(\phi), \quad (12.11)$$

$$R_{sp} = \sin(\theta) \sin(\psi), \quad (12.12)$$

$$R_{rs} = \sin(\theta) \cos(\phi), \quad (12.13)$$

$$R_{ps} = \sin(\theta) \sin(\phi), \quad (12.14)$$

and

$$R_{ss} = \cos(\theta). \quad (12.15)$$

$$(12.16)$$

If we now rotate the diagonal tensor of eigenvalues, Eq. (12.4), into a nondiagonal space, one obtains

$$\mathbf{D} = \mathbf{R}^T \times \mathbf{U} \times \mathbf{R} \quad (12.17)$$

where  $\mathbf{D}$  is the same tensor defined in Eq. (12.1). Explicitly working out this rotation results in:

$$D_{rr} = R_{rr}\lambda_1 R_{rr} + R_{pr}\lambda_2 R_{pr} + R_{sr}\lambda_3 R_{sr}, \quad (12.18)$$

$$D_{rp} = R_{rr}\lambda_1 R_{rp} + R_{pr}\lambda_2 R_{pp} + R_{sr}\lambda_3 R_{sp}, \quad (12.19)$$

$$D_{rs} = R_{rr}\lambda_1 R_{rs} + R_{pr}\lambda_2 R_{ps} + R_{sr}\lambda_3 R_{ss}, \quad (12.20)$$

$$D_{pr} = D_{rp}, \quad (12.21)$$

$$D_{pp} = R_{rp}\lambda_1 R_{rp} + R_{pp}\lambda_2 R_{pp} + R_{sp}\lambda_3 R_{sp}, \quad (12.22)$$

$$D_{ps} = R_{rp}\lambda_1 R_{rs} + R_{pp}\lambda_2 R_{ps} + R_{sp}\lambda_3 R_{ss}, \quad (12.23)$$

$$D_{sr} = D_{rs}, \quad (12.24)$$

$$D_{sp} = D_{ps}, \quad (12.25)$$

and

$$D_{ss} = R_{rs}\lambda_1 R_{rs} + R_{ps}\lambda_2 R_{ps} + R_{ss}\lambda_3 R_{ss}, \quad (12.26)$$

$$(12.27)$$

which means that we can apply Bayesian probability theory to infer the eigenvalues and Euler angles directly from the data without taking any logarithms and without attempting to divide the amplitude of the tensor from the problem. Additionally, because we can use the eigenvalues and Euler angles directly, we will be able to infer these quantities even when the data contain multiple diffusion tensors; something not possible with the procedures described earlier.

In the Bayesian calculations that follow, it will be assumed that there are multiple diffusion tensors in a given data set. Consequently, the notation used up to now must be modified. Here we introduce an index that represents the  $\ell$ th diffusion tensor. Additionally, we are going to include an constant offset in the diffusion tensor model. The presence of this constant will be under user control, so it may be included or excluded at the user's discretion. Equation (12.2) will be rewritten as:

$$d_i = C\delta(\nu) + \sum_{\ell=1}^m A_{\ell} \exp \left\{ \text{Conv} \sum_{j=1}^3 \sum_{k=1}^3 g_{ij} D_{jkl} g_{ik} \right\} + n_i \quad (12.28)$$

where  $m$  is the number of diffusion tensors, "Conv" is the conversion factor defined in Eq. (12.3),  $D_{jkl}$  is the  $j$ th column, the  $k$  row, of the  $\ell$ th diffusion tensor,  $g_{ik}$  is the  $k$ th component of the  $i$ th gradient,  $A_{\ell}$  is the amplitude of the  $\ell$ th tensor,  $C$  is the constant offset, and  $\delta(\nu)$  is an indicator function defined as:

$$\delta(\nu) = \begin{cases} 1 & \text{If the user included a constant} \\ 0 & \text{Otherwise} \end{cases} . \quad (12.29)$$

We will combine the constant offset and the amplitudes into a single set of amplitudes represented by:  $B_{\ell} \in \{C, A_1, \dots, A_m\}$ . The number of  $B$  amplitudes is  $n$  with  $n = m + \delta(\nu)$ . Similarly, we will define a new model function  $G_{\ell}$  that combines the diffusion tensors and the constant:

$$G_{\ell}(\Theta, i) \equiv \begin{cases} i^0 & \text{If } \ell = 1 \\ \exp \left[ \text{Conv} \sum_{j=1}^3 \sum_{k=1}^3 g_{ij} D_{jkm} g_{ik} \right] & \text{otherwise, with } m = \ell - 1 \end{cases} , \quad (12.30)$$

where we intentionally wrote the constant model as  $i^0$  as a reminder to the reader that the constant model is 1 for all  $i$ . Additionally, we have included  $i$  in the argument list of the function  $G$  as a reminder that  $G$  is a function of the data point number. In the above we are using  $\Theta$  to stand for all of the diffusion tensor parameters, so

$$\Theta \equiv \{\lambda_{11}, \lambda_{12}, \lambda_{13}, \theta_1, \phi_1, \psi_1, \dots, \lambda_{m1}, \lambda_{m2}, \lambda_{m3}, \theta_m, \phi_m, \psi_m\}. \quad (12.31)$$

In this notation, the Eq. (12.2) can be written as

$$d_i = \sum_{\ell=1}^n B_{\ell} G_{\ell}(\Theta, i) + n_i \quad (12.32)$$

and it is in this form that it will be used in the Bayesian calculations for the posterior probability for the  $\Theta$  parameters.

The Diffusion Tensor package computes the marginal posterior probability for each of the parameters appearing in the model. For computational convenience we are going to marginalize out the amplitudes, the constant offset and the standard deviation of the noise. Consequently, the target distribution for the Markov chain will be the joint marginal posterior probability for all of the diffusion parameters  $\Theta$ . This joint posterior probability is computed from the joint posterior probability for all of the parameters, including the amplitudes and the standard deviation of the noise:

$$P(\Theta|DI) = \int P(B_1 \dots B_n \sigma \Theta|DI) dB_1 \dots dB_n d\sigma. \quad (12.33)$$

The right-hand side of this equation is factored using Bayes' theorem

$$P(\Theta|DI) \propto \int P(B_1 \dots B_n \sigma \Theta|I) P(D|B_1 \dots B_n \sigma \Theta I) dB_1 \dots dB_n d\sigma \quad (12.34)$$

where  $P(B_1 \dots B_n \sigma \Theta|I)$  is the joint prior probability for all of the parameters and the other term, the direct probability for the data,  $P(D|B_1 \dots B_n \sigma \Theta I)$ , is often called the likelihood function. Next the product rule of probability theory is used to factor the joint prior probability for the parameters into a series of independent prior probabilities, one for each parameter,

$$\begin{aligned} P(B_1 \dots B_n \sigma \Theta|I) &\propto P(\sigma|I) \prod_{\ell=1}^n P(B_\ell|I) \\ &\times \prod_{j=1}^m [P(\lambda_{j1}|I) P(\lambda_{j2}|I) P(\lambda_{j3}|I)] \\ &\times \prod_{j=1}^m [P(\theta_j|I) P(\phi_j|I) P(\psi_j|I)] \end{aligned} \quad (12.35)$$

where to make this factorization we assumed logical independents, i.e., the prior probability we would assign to one parameter does not depend on any other parameters. Substituting, the prior probability for the parameters, Eq. (12.35), into the marginal posterior probability for the  $\Theta$  parameters, Eq. (12.34), one obtains:

$$\begin{aligned} P(\Theta|DI) &\propto \int \prod_{\ell=1}^n [P(B_\ell|I)] \\ &\times \prod_{j=1}^m [P(\lambda_{j1}|I) P(\lambda_{j2}|I) P(\lambda_{j3}|I)] \\ &\times \prod_{j=1}^m [P(\theta_j|I) P(\phi_j|I) P(\psi_j|I)] \\ &\times P(D|B_1 \dots B_n \sigma \Theta I) dB_1 \dots dB_n d\sigma \end{aligned} \quad (12.36)$$

We have reached the point in the calculation where we must assign probabilities to represent the various terms in this equation. The prior probability for the amplitudes  $B_\ell$  will be assigned as a Gaussians given by

$$P(B_\ell|I) = \left( \frac{2\pi\sigma^2}{g_{\ell\ell}\beta^2} \right)^{-\frac{1}{2}} \exp \left\{ -\frac{g_{\ell\ell}\beta^2 B_\ell^2}{2\sigma^2} \right\} \quad (12.37)$$

where  $\sigma$  is the standard deviation of the prior probability for the errors,  $\beta$  is hyperparameter and expresses how strongly we think we can guess the value of the amplitudes relative to how well it can be determined from the data and  $g_{\ell\ell}$ , the squared-length of the function  $G_\ell(\Theta, i)$ , is defined in Eq. (12.44) below. The prior probability for the noise standard deviation,  $P(\sigma|I)$ , was assigned a Jeffreys' prior

$$P(\sigma|I) \propto \frac{1}{\sigma}. \quad (12.38)$$

The prior probabilities for the eigenvalues are defined by the user. When the user loads a set of data, the interface will make its best guess for the prior probabilities and set a default Gaussian prior. The user, using the prior viewer, can modify or completely replace these default prior probabilities.

The model equation is symmetric under relabeling of the amplitudes and tensors parameters and each tensor is symmetric under reordering the rows or columns of the tensors. These symmetry cause the joint posterior probability for the tensor parameters to be symmetric in the sense that if there is a peak at  $\lambda_{11} = \beta$  and  $\lambda_{12} = \gamma$ , then there is also a peak at  $\lambda_{11} = \gamma$  and  $\lambda_{12} = \beta$ ; this symmetry occurs because the model does not tell us which eigenvalue corresponds to to which model component. Consequently, a convention must be introduced which brakes this symmetry by identifying a model component with a signal component. In the calculations implemented here, we break this symmetry by ordering the eigenvalues within and across tensors. By ordering the eigenvalues this manner, we effectively tell probability theory what we mean when we say, "eigenvalue one," we mean the largest eigenvalue. We do impose one additional condition on these eigenvalues, if there are multiple tensors, then we impose the additional condition:  $0 < \lambda_{11} < \lambda_{21} < \dots < \lambda_{m1} < \text{High}$ , where "High" is the upper bound in the prior probability for the eigenvalues. In effect telling probability theory that when we say tensor 1, we mean the tensor having the smallest principle eigenvalue, etc. Finally, we will assign the prior probabilities for the angles as uniform prior probabilities of the form:

$$P(\text{angle}|I) = \begin{cases} 1 & \text{If } 0 \leq \text{angle} \leq \pi \\ 0 & \text{Otherwise} \end{cases}, \quad (12.39)$$

where "angle" is any of the Euler angles in any of the tensors. The direct probability for the data was assigned using a Gaussian likelihood. Gathering up the prior probabilities and assigning the likelihood, the joint posterior probability for the parameters, Eq. (12.34), is given by:

$$P(\Theta|DI) \propto \int (2\pi\sigma^2)^{-\frac{N+m}{2}} \prod_{j=1}^m \sqrt{g_{jj}} \beta \prod_{k=1}^3 P(\lambda_{jk}|I) \exp\left\{-\frac{Q}{2\sigma^2}\right\} dB_1 \dots dB_n d\sigma \quad (12.40)$$

where we have left the prior probabilities for the eigenvalues in symbolic form because they are user defined. The quantity  $Q$  is given by

$$Q \equiv N\bar{d}^2 - 2N \sum_{j=1}^m B_j T_j(\Theta) + N \sum_{j=1}^m \sum_{k=1}^m B_j B_k g_{jk} (1 + \beta^2 \delta_{jk}) \quad (12.41)$$

and is essentially Chi-squared where

$$\bar{d}^2 = \frac{1}{N} \sum_{i=1}^N d_i^2 \quad (12.42)$$

is the mean-square data value. The  $T_j$

$$T_j = \frac{1}{N} \sum_{i=1}^N d_i G_j(\Theta, i) \quad (12.43)$$

are the mean projection of the data onto the model. Finally, the  $g_{jk}$  are defined as

$$g_{jk} = \frac{1}{N} \sum_{i=1}^N G_j(\Theta, i) G_k(\Theta, i). \quad (12.44)$$

The integrals over the amplitudes are Gaussian quadrature integrals and the integral over the standard deviation is essentially a gamma function integral. Both of which are straight forward to evaluate and we omit the details, one obtains

$$P(\Theta|DI) \propto |g_{jk}|^{-\frac{1}{2}} \prod_{j=1}^m \sqrt{g_{jj}} \beta \prod_{k=1}^3 P(\lambda_{jk}|I) \left[ \frac{Q}{2} \right]^{-\frac{N}{2}}. \quad (12.45)$$

This probability density function is of the form of Students  $t$ -distribution, and it is this  $t$ -distribution that is targeted by the Markov chain Monte Carlo simulation.

For some examples of diffusion tensors imaging, see [14] and for more on the origins of diffusion tensors imaging, see [38, 44, 62, 63]. For those wishing to know more about how these integrals were done see [2, 61]. For a brief overview of this subject, [Wikipedia](#) has a surprisingly good discussion.

## 12.2 Using The Package

The type of data loaded by the diffusion tensor package is dependent on the type of abscissa one selects, there are three options that may be chosen in the “Abscissa Options” box, by checking one of the three check boxes:

**Use g Vectors** tells the package that the abscissa of the input data will contain gradient or g-vectors. When g-vectors are used, the package converts them into b-vectors using

$$b_j = \Gamma \delta \sqrt{(\Delta - \delta/3)} g_j. \quad (12.46)$$

where  $j$  should be replaced by  $x$ ,  $y$  or  $z$ . This conversion requires three additional parameters to be entered,  $\Gamma$  the gyromagnetic ratio in units of  $1/(\text{Gauss Sec})$ ,  $\delta$  the duration of the gradient pulse in units of seconds and  $\Delta$  the time between the two gradient pulses in units of seconds. These parameters are specified by three prior probabilities of prior type parameter, i.e., constants. So when this option is selected, three additional prior probabilities are included in prior list. Note that gradients are three dimensional vectors, so the input Ascii data file must contain: a data point number, the data, the gradients  $g_r$ ,  $g_p$  and  $g_s$ , where the subscripts stand for the readout, phase encode and the slice select directions. Their appearance in the Ascii data file is a convention and they must appear in the given order, see Chapter A for more on Ascii file formats.

**Use b Vectors** indicates that the input Abscissa are b-vectors and consequently, no conversion to b-vectors is required. In this case, the Diffusion Tensor package sets the conversion factor to one. The input Ascii data file must contain: a data point number, the data,  $b_r$ ,  $b_p$  and  $b_s$  in that order.

**Use b Matrix** indicates that the input data has a b-matrix for the abscissa. A b-matrix is a real symmetric  $3 \times 3$  matrix having six independent elements:

$$B \text{ Matrix} \equiv \begin{pmatrix} b_{rr} & b_{pr} & b_{sr} \\ b_{rp} & b_{pp} & b_{sp} \\ b_{rs} & b_{ps} & b_{ss} \end{pmatrix}. \quad (12.47)$$

Consequently, for b-matrix data, the input file format is: a data point number, the data followed by  $b_{rr}$ ,  $b_{pp}$ ,  $b_{ss}$ ,  $b_{rp}$ ,  $b_{rs}$  and  $b_{ps}$ . Note the order of the elements in the b-matrix data file is a convention and must be in the given order.

After selecting the “Abscissa Options” one can load the diffusion data. As explained above, the type of data you must load is dependent on which Abscissa Options you selected. The model is specified by selecting the number of diffusion tensors, and indicating whether or not a constant offset is present. Setting the number of diffusion tensors is done using the “Tensor number” pull-down menu, see Fig. 12.1. Currently, you can select one, two or three tensors. To include a constant offset in the model, one checks the “Include Constant” check box.

The Text outputs files from the Diffusion Tensor packages consist of: “Bayes.prob.model,” “BayesDiffTensor.mcmc.values,” “Bayes.params,” “Console.log,” “Bayes.accepted” and finally a “Bayes.Condensed.File.” These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the mcmc.values report is discussed in Appendix D and a small sample of this file is shown in Fig. 12.2. The other reports are discussed in Chapter 3. Additionally, the “Plot Results Viewer” can be used to view the output probability density functions. In addition to the standard data, model and residual plots there are probability density functions for the eigenvalues, the three Euler angles, and the standard deviation of the noise.

The mean and standard deviation estimates of the various parameters are written to the “Bayes-DiffusionTensor.mcmc.values” output file, see Fig. 12.2 for a sample of this file. This file may be printed, or viewed using the interface. It consists of four general sections, the first section just identifies the model that was processed and relates some information about the posterior probability, the prior, and the likelihood. The second section contains the parameters that maximized the joint posterior probability. The third section contains mean and standard estimates of the parameters appearing in the Diffusion Tensor package. This third section also contains the logarithm of the posterior probability for the model. This probability is used to update the probabilities file and may be used to do model selection.

The Bayesian calculation for joint posterior probability for all of the parameters is done using Markov chain Monte Carlo with simulated annealing. In the annealing phase of the calculation the posterior probability for the model is computed. An example of this file is shown in Fig. 12.3. The “Bayes.prob.model” file contains the name of the model, the natural logarithm of the posterior probability for the model, the normalized probability for the model and the date and time the model was run.

The example shown in Fig. 12.3 was done using the “bayes/Bayes.test.data/DiffTensorAscii.Data”. This simulated data set contains one diffusion tensor plus a constant. To illustrate how to use the

Figure 12.2: Diffusion Tensor Parameter Estimates

```

----- Eigenvectors -----
Tensor: 1      X              Y              Z
Vec: 1  -4.67473323E-01  -2.86124440E-01  -8.36421842E-01
Vec: 2  -7.92821559E-01  -2.82826221E-01   5.39854892E-01
Vec: 3  -3.91027707E-01   9.15501029E-01  -9.46319111E-02

The expected parameter values (mean value of the probability distributions):

Parameter Description          Mean Value      Std. Dev.      Peak Value
Lambda 1 Tensor 1             4.66911E-04    2.21524E-06    4.66811E-04
Lambda 2 Tensor 1             1.55124E-04    2.58916E-06    1.55834E-04
Lambda 3 Tensor 1             6.50448E-07    6.54884E-07    3.50572E-08
Average Lambda 1              2.07562E-04    1.19007E-06    2.07560E-04
RA Tensor 1                   1.93932E-04    1.03217E-06    1.94039E-04
FA Tensor 1                   9.34352E-01    5.63672E-03    9.34857E-01
Theta Tensor 1                9.53311E+01    8.87027E-01    9.54301E+01
Phi Tensor 1                  1.13092E+02    3.09494E-01    1.13128E+02
Psi Tensor 1                  3.27776E+01    4.09517E-01    3.28396E+01
Dxx Tensor 1                  1.14992E-04    1.70590E-06    1.14795E-04
Dxy Tensor 1                  1.85249E-04    1.16739E-06    1.85606E-04
Dxz Tensor 1                  4.48854E-05    2.11995E-06    4.45133E-05
Dyy Tensor 1                  3.06394E-04    2.01249E-06    3.05897E-04
Dyz Tensor 1                  1.04314E-04    2.36628E-06    1.04367E-04
Dzz Tensor 1                  2.01300E-04    1.74817E-06    2.01988E-04
Amp Tensor 1 Set 1            1.00122E+02    6.10312E-02    1.00124E+02

Const 1                        1.00030E+01    1.43077E-01    1.00079E+01

```

Figure 12.2: The output from the Diffusion Tensor Package consists of four parts. The first part identifies the model that was processed and then relates some information about the priors and the likelihood. This is followed by the parameters that maximized the joint posterior probability for the parameters. The maximum posterior probability parameters are followed by the eigenvectors that maximized the posterior probability. And these are followed by the mean and standard deviation estimates for the parameters show here.

Figure 12.3: Diffusion Tensor Posterior Probability For The Model

Model Name	Log(e) Prob	Probability	Date/Time Run
One D	-1.68351E+02	0.00000	Mon Jun 9 10:24:23 2003
One D + Const	-1.31755E+02	0.99997	Mon Jun 9 10:25:05 2003
Two D	-1.72142E+02	0.00000	Mon Jun 9 10:26:49 2003
Two D + Const	-1.42238E+02	0.00003	Mon Jun 9 10:31:02 2003

Figure 12.3: During the annealing phase the Diffusion Tensor Package computes the posterior probability for the model. This probability is written to the “Bayes.prob.model” file located in the experiment. If you run multiple models, the probabilities for the various models are appended to this file and may be used for model selection.

Diffusion Tensor package to do model selection, the one tensor model, then the one plus constant, etc. up to the two tensors plus a constant were run. Note that the logarithm of the posterior probability reaches a peak at the one tensor plus a constant model, and then decreases when the two tensor models were run.



## Chapter 13

# Big Magnetization Transfer

The Big magnetization transfer package analyzes data where two sites are exchanging magnetization under the assumption that one of the sites is essentially infinite compared to the other. Consequently, the big site is essentially unchanging and the solution for the other site simplifies, Eq. (13.1) below. The interface to this package is shown in Fig. 13.1. To use this package, you must do the following:

**Select** the Big Magnetization Transfer package from the Package menu.

**Load** one or more Ascii data sets using the Files menu. When the data have been successfully loaded, the data are displayed in the Ascii Data viewer.

**Check** the Analysis Options/Find Outliers box if you suspect outliers are present in the data.

**Review** the prior probabilities for the  $K_d$ ,  $R_d$  and  $R_w$  parameters using the Prior Viewer.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

### 13.1 The Bayesian Calculation

The Big magnetization transfer problem is one in which a very large spin reservoir is exchanging magnetization with a much smaller spin reservoir. We will call the larger reservoir the Solvent, and the other the Small reservoir or resonance. When one reservoir is much greater than the other, the

Figure 13.1: The Big Magnetization Package Interface

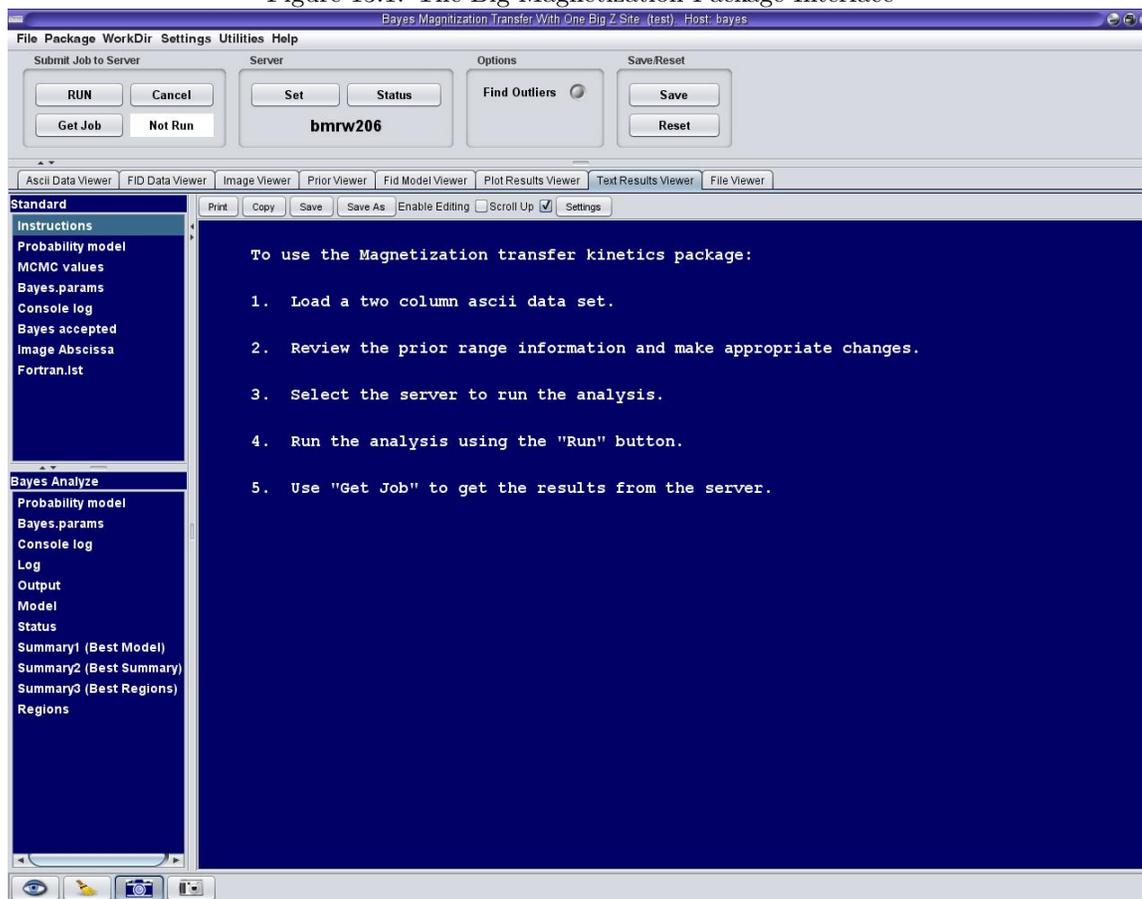


Figure 13.1: The interface to the Big Magnetization Transfer Package is shown here. The Big magnetization transfer interface allows you to analyze magnetization transfer data where one of the two sites may be considered as infinite compared to the other, see text for details.

Solvent is unaffected by the the transfer of magnetization to or from Small reservoir. Consequently, the solution to magnetization exchange equations simplify, one obtains

$$M_d(t) = M_{z0} \left( 1 - \frac{2K_d}{R_w - K_d - R_d} \left[ e^{-(R_d+K_d)t_i} - e^{-R_w t_i} \right] \right) \quad (13.1)$$

where  $M_{z0}$  is the initial Small reservoir magnetization,  $K_d$  is the rate at which the Small reservoir is exchanging magnetization to the Solvent,  $R_w$  is the inverse of the  $T_2$  time for Solvent reservoir,  $R_d$  is the  $R_2$  relaxation rate of the Small reservoir and this equation implicitly assumes that the magnetization is fully inverted.

Markov chain Monte Carlo is used to draw samples from the joint posterior probability for all of the parameters. Form these samples, the marginal posterior probability for each parameter is computed. For example the posterior probability for the exchange rate  $K_d$  is computed as

$$P(K_d|DI) = \int P(K_d M_{z0} R_w R_d \sigma | DI) dM_{z0} dR_w dR_d d\sigma \quad (13.2)$$

where all of the parameters except the parameter of interest have been removed by marginalization. The joint posterior probability for the parameters, the integrand of this equation, is factored using Bayes' theorem and the product rule to become

$$P(K_d M_{z0} R_w R_d \sigma | DI) \propto P(K_d M_{z0} R_w R_d \sigma | I) P(D | K_d M_{z0} R_w R_d \sigma I) \quad (13.3)$$

where  $P(K_d M_{z0} R_w R_d \sigma | I)$  is the joint prior probability for the parameters and  $P(D | K_d M_{z0} R_w R_d \sigma I)$  is the likelihood. The joint prior probability or the parameters is factored into independent prior probabilities for each of the parameters separately

$$P(K_d M_{z0} R_w R_d \sigma | I) = P(K_d | I) P(M_{z0} | I) P(R_w | I) P(R_d | I) P(\sigma | I). \quad (13.4)$$

The prior probability for the standard deviation of the noise prior probability,  $P(\sigma | I)$ , is assigned a Jeffreys' prior,  $P(\sigma | I) \propto 1/\sigma$ . The prior probability for the amplitude  $P(M_{z0} | I)$ , was assigned using a using a very broad unbounded Gaussian of zero mean having a standard deviation of  $3 \times 10^5$ . The remaining three prior probabilities,  $P(K_d | I)$ ,  $P(R_w | I)$  and  $P(R_d | I)$  all default to a prior positive. However, these priors are under user control and they may changed by the user. The likelihood,  $P(D | K_d M_{z0} R_w R_d \sigma I)$  was assigned using a Gaussian prior probability having standard deviation  $\sigma$ . If we now collect all of the priors, assign the likelihood, and evaluate the integrals over both  $M_{z0}$  and  $\sigma$ , one obtains:

$$P(K_d M_{z0} R_w R_d \sigma | DI) \propto P(K_d | I) P(R_w | I) P(R_d | I) \left[ N \overline{d^2} - \overline{h^2} \right]^{-\frac{N}{2}} \quad (13.5)$$

where  $\overline{d^2}$  is the mean-square data value and is given by

$$\overline{d^2} = \frac{1}{N} \sum_{i=1}^N d_i^2. \quad (13.6)$$

The sufficient statistic,  $\overline{h^2}$ , is given by

$$\overline{h^2} = \frac{T^2}{g} \quad (13.7)$$

where  $T$  is the projection of the data onto the model and is given by

$$T = \sum_{i=1}^N d_i \left( 1 - \frac{2K_d}{R_w - K_d - R_d} \left[ e^{-(R_d+K_d)t_i} - e^{-R_w t_i} \right] \right) \quad (13.8)$$

and  $g$  is the squared length of the model:

$$g = \sum_{i=1}^N \left( 1 - \frac{2K_d}{R_w - K_d - R_d} \left[ e^{-(R_d+K_d)t_i} - e^{-R_w t_i} \right] \right)^2. \quad (13.9)$$

It is Eq. (13.5) that is targeted by the Markov chain Monte Carlo simulations using simulated annealing. Note that this posterior probability does not contain the initial magnetization. However, in the process of computing this quantity the maximum posterior probability estimate of this parameter is computed and output for each value of the exchange and relaxation rates. While not strictly the Bayesian estimate of this parameter, the distribution of these estimates provide good mean and standard deviation estimates of the initial magnetization.

## 13.2 Outputs From The Big Magnetization Transfer Package

The Big Magnetization Transfer Package is an example of a preloaded Enter Ascii model. Preloaded means that when the Big Magnetization Transfer package is selected, the interface copies an Ascii model, MtZBig.f, from the system directory into the current experiment and starts up the Enter Ascii package. Consequently, the outputs from this package are all Enter Ascii outputs. The Text outputs files from the Big Magnetization Transfer packages consist of: “Bayes.prob.model,” “BayesEnterAscii.mcmc.values,” “Bayes.params,” “Console.log,” “Bayes.accepted” and a “Bayes.Condensed.File.” These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the mcmc.values report is discussed in Appendix D and the other reports are discussed in Chapter 3. Additionally, the “Plot Results Viewer” can be used to view the output probability density functions. In addition to the standard data, model and residual plots there are probability density functions for the decay rate constants, decay times, the amplitudes for each data set for each exponential

The full spectrum of data typical of this experiment is shown in Fig. 13.2. Note the presence of a water suppression artifact near 5ppm. To acquire this data one typically inverts the water, suppresses it, and finally acquires the Fid. The resonance exchanging magnetization with the water is the small resonances located at 11ppm. If we expand this region and then do a horizontal display, one obtains the spectra shown in Fig. 13.3. The Solvent resonance begins at its equilibrium value. When the Solvent resonances is inverted, negative magnetization flows into the Small reservoir bring its value down. At different delay times, different amounts of negative magnetization exchange with the Solvent resonance affecting its intensity in a predictable manner. The peak heights or amplitudes as determined by Bayes Analyze are used as input to the analysis. An example of data using a peak pick is shown in Fig. 13.3. Note that the Solvent resonance was inverted, as the Solvent exchanges negative magnetization with the Small resonance, the Small resonance is eventually pulled down. As the delay time increases the Small resonance has time to relax back toward equilibrium and eventually recovers.

Figure 13.2: Big Magnetization Transfer Example Fid

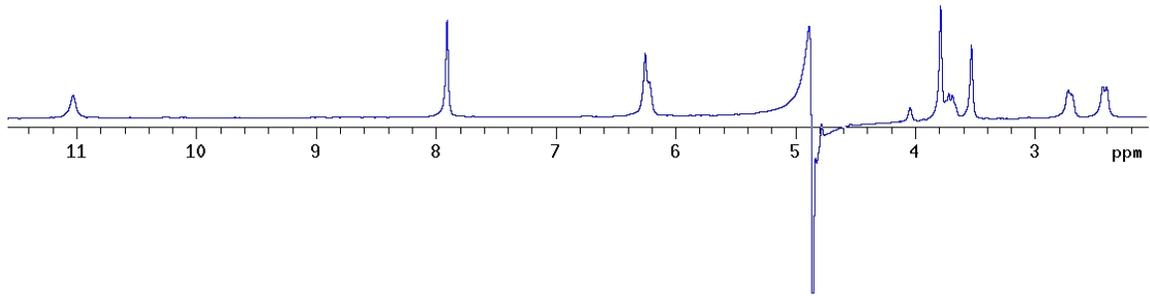


Figure 13.2: The full spectrum of the Fid data used in the big magnetization analysis. Note the water suppression artifact near 5ppm. The peak exchanging magnetization with the water is the small resonance near 11ppm.

Figure 13.3: Big Magnetization Transfer Expansion

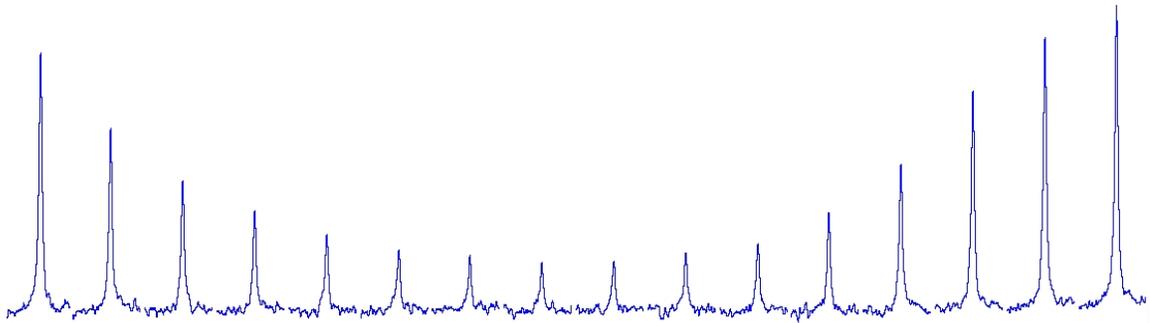


Figure 13.3: The peak exchanging magnetization begins at its equilibrium value. When the Solvent is inverted, negative magnetization flows into the Small reservoir bringing this reservoir down. At different delay times the Solvent is more fully relaxed and the Small resonance begins to recover, until at the longest delay time the Small resonance has returned to equilibrium. It is the amplitudes of the Small resonance that serve as input to this analysis.

Figure 13.4: Big Magnetization Transfer Peak Pick

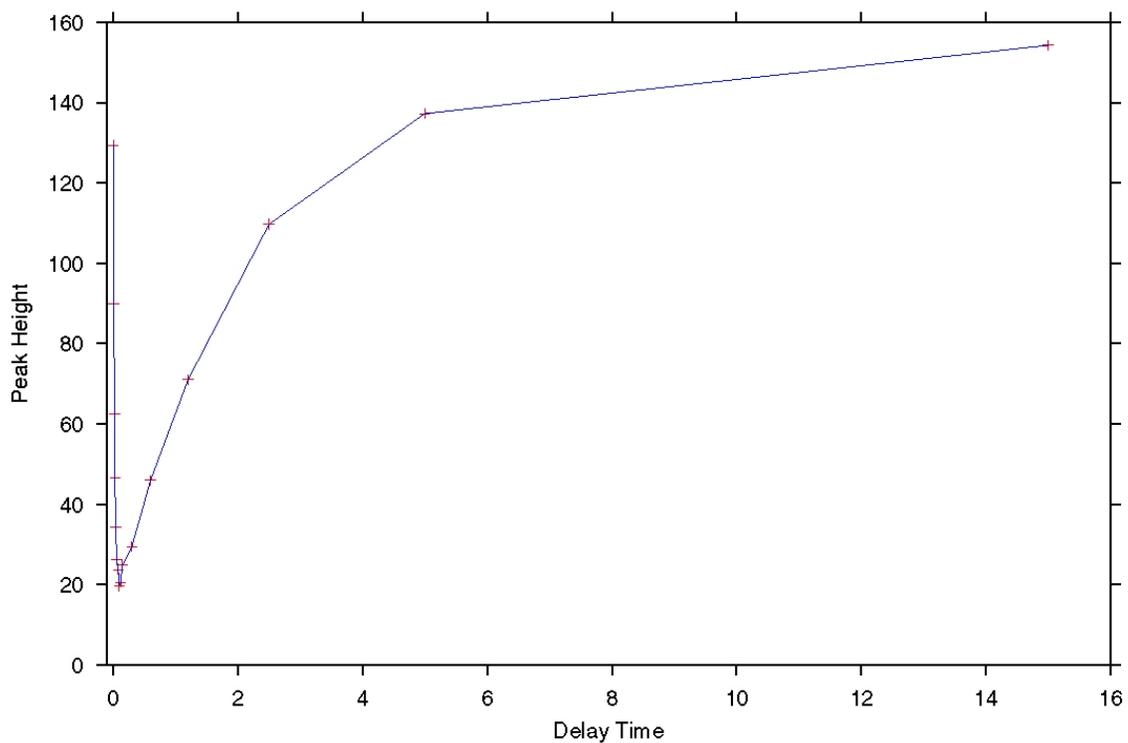


Figure 13.4: The peak heights or amplitudes (as determined by Bayes Analyze) serve as input to the Big magnetization transfer package. Here the peak heights for the spectra shown in Fig. 13.3 were used as input to the analysis. Note the Solvent magnetization starts fully relaxed, then after the Solvent is inverted, the negative Solvent magnetization begins to bring the Small reservoir down. As a function of delay time this continues for a while and then eventually the Small resonance recovers back to equilibrium.

## Chapter 14

# Magnetization Transfer

The Magnetization transfer package analyzes two site magnetization exchange data. This data is obtained from a peak pick, a Bayes Analyze file or it can be manually entered and loaded as an Ascii file. The Ascii data used in this package are generated from an Fid. The Fid data should be an arrayed inversion recovery data set. If we call the two sites that are exchanging magnetization Site A, and Site B, then the data should be in inversion recover data set where Site A was inverted. You should also have a data set where Site B was inverted. Finally, you can also invert both sites and that data may also be used. Preferably, you should array these Fid so as to obtain as many data values as possible on each recovery curve, the more data you have the more precise you parameter estimates will be. Note, that while it is not recommended, a single inversion recover data set can be used and you will be able to get exchange rates. However, because of the limited about of data, they will probably be highly uncertain. The interface to this package is shown in Fig. 14.1. To use this package, you must do the following:

**Select** the Magnetization Transfer package from the Package menu.

**Load** at least one and preferably two three column Ascii data sets using the Files menu. The three columns are the abscissa value, and the amplitude or peak value of the Site A and B magnetization. Additionally, you can load an arrayed Fid and then use a double cursor to mark the center of the two exchanging peaks and use the “Get Peak” button on the bottom right of the Fid viewer. When a data set has been successfully loaded a plot contain the two sites is displayed in the Ascii Data viewer. For Fids, to load a second peak pick or Bayes Analyze file, simply load a second Fid and either load a peak pick or a Bayes Analyze file. Finally, if you have analyzed this Fid using Bayes Analyze you can load the resonance amplitude from the Bayes Analyze files using the “Files/Load Bayes Analyze” menu.

**Check** the Analysis Options/Find Outliers box if you suspect outliers are present in the data.

**Normally** the prior probabilities for the parameters would normally have to be reviewed here. However, the calculations are done using a variable transformation, sum and difference variables and this change of variables makes determining prior ranges so easy that the package does it automatically.

**Select** the server that is to process the analysis.

Figure 14.1: The Magnetization Transfer Package Interface

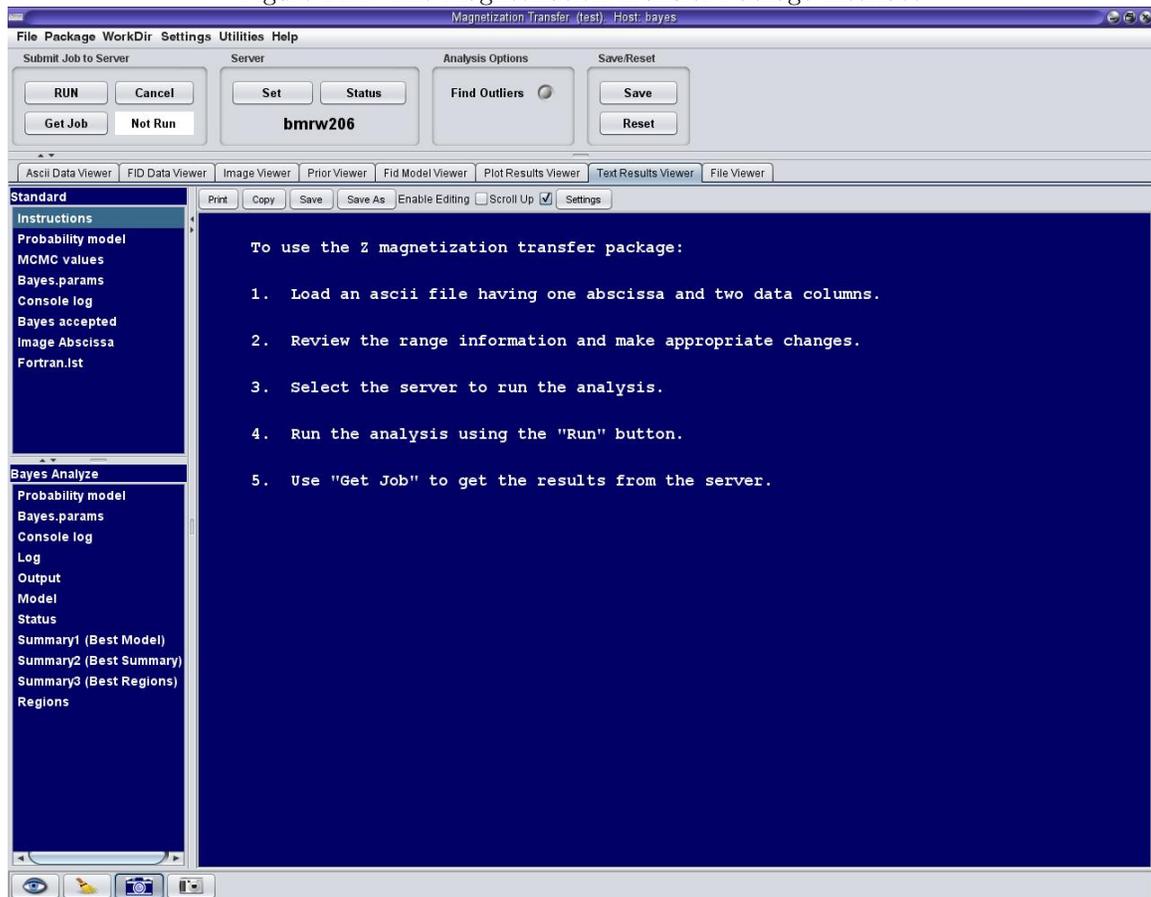


Figure 14.1: The magnetization transfer packages analyzes one or more data set using the equations governing two site magnetization exchange. The inferred parameters are the two exchange rates and the two relaxation rates. For more on the actual calculations and the widgets see the text.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

## 14.1 The Bayesian Calculation

The two site magnetization transfer package solves problems involving magnetization transfer. If we designate the two sites as “*a*” and “*b*” respectively then the magnetization transfer model of the “*a*” site is related to the data by

$$d_a(t_i) = M_a(t_i) + \text{error} \quad (14.1)$$

and similarly for the “*b*” site,

$$d_b(t_i) = M_b(t_i) + \text{error} \quad (14.2)$$

where  $M_a(t)$  and  $M_b(t)$  are the solution to the Bloch-McConnell equations, and “error” represents noise in the data and should not be taken to mean that this noise is the same in both data sets. McConnell’s modification to the Bloch equation is given by

$$\frac{dM_a(t)}{dt} = -R_{1a}[M_a(t) - M_a(\infty)] - K_{ab}M_a(t) + K_{ba}M_b(t) \quad (14.3)$$

$$\frac{dM_b(t)}{dt} = -R_{1b}[M_b(t) - M_b(\infty)] - K_{ba}M_b(t) + K_{ab}M_a(t) \quad (14.4)$$

where  $R_{1a}$  and  $R_{1b}$  are the relaxation rates for the “*a*” and “*b*” sites,  $K_{ba}$  is the rate at which magnetization goes from the “*b*” to the “*a*” site. Similarly  $K_{ab}$  is the rate at which magnetization exchanges from the “*a*” to the “*b*” site.

These equations are coupled linear first order differential equations and their bi-exponential solution is given by

$$M_a(t) = H(t)M_a(0) + G(t)M_b(0) + \left[1 - H(t) - G(t)\frac{K_{ab}}{K_{ba}}\right] M_a(\infty) \quad (14.5)$$

$$M_b(t) = I(t)M_b(0) - J(t)M_a(0) + \left[J(t) + [1 - I(t)]\frac{K_{ba}}{K_{ab}}\right] M_a(\infty) \quad (14.6)$$

where  $M_a(\infty)$  is the equilibrium magnetization for the “*a*” site, and  $M_a(0)$  and  $M_b(0)$  are the initial magnetization for the “*a*” and “*b*” sites. The functions  $G(t)$ ,  $H(t)$ ,  $I(t)$  and  $J(t)$  are defined as

$$G(t) = \frac{\exp\{\alpha_1 t\} - \exp\{\alpha_2 t\}}{U - V}, \quad (14.7)$$

$$H(t) = \frac{U \exp\{\alpha_2 t\} - V \exp\{\alpha_1 t\}}{U - V}, \quad (14.8)$$

$$I(t) = \frac{U \exp\{\alpha_1 t\} - V \exp\{\alpha_2 t\}}{U - V}, \quad (14.9)$$

$$J(t) = UVG(t), \quad (14.10)$$

with

$$U = \frac{\alpha_1 + K_{ab} + R_{1a}}{K_{ba}}, \quad (14.11)$$

$$V = \frac{\alpha_2 + K_{ab} + R_{1a}}{K_{ba}}. \quad (14.12)$$

The observed decay rates  $\alpha_1$  and  $\alpha_2$  are given by

$$\alpha_{1,2} = -\frac{R_{1a} + R_{1b} + K_{ab} + K_{ba}}{2} \pm \frac{1}{2} \sqrt{(R_{1a} - R_{1b} - K_{ab} - K_{ba})^2 + 4K_{ab}K_{ba}} \quad (14.13)$$

where  $\alpha_1$  takes the plus and  $\alpha_2$  the minus.

The four parameters of interest are the two exchange rates  $K_{ab}$  and  $K_{ba}$  and the two relaxation rates  $R_{1a}$  and  $R_{1b}$ . In addition there are three parameters per “a” and “b” site that must be included, two initial conditions magnetizations,  $M_a(0)$  and  $M_b(0)$  and one equilibrium magnetization  $M_a(\infty)$ . The equilibrium condition,

$$M_b(\infty) = M_a(\infty) \frac{K_{ab}}{K_{ba}} \quad (14.14)$$

was used to eliminate  $M_b(\infty)$ .

Before we start the process of computing the posterior probability for the parameters of interest,  $K_{ab}$ ,  $K_{ba}$ ,  $R_{1a}$  and  $R_{1b}$ , we are going to rewrite the model equations and the data into a form that is more convenient for the upcoming analytic calculation. First we are going to define a single data set  $D$  with data item  $d(t_i)$  that is the “a” site data followed by the “b” site data:

$$d(t_i) = \begin{cases} d_a(t_i) & \text{if } 1 \leq i \leq N \\ d_b(t_j) & \text{if } 1 \leq j \leq N \text{ with } j \equiv i - N \end{cases} \quad (14.15)$$

where the index  $i$  ranges from 1 up to  $2N$ . This definition is roughly like defining a complex data set with the “a” site being the real data and the “b” site being the imaginary data. Next we will rewrite Eqs. (14.1 and 14.2) into a single equation:

$$d(t_i) = \sum_{k=1}^3 M_k \mathcal{H}_k(t_i) + \text{error} \quad (14.16)$$

where  $M_1 \equiv M_a(0)$ ,  $M_2 \equiv M_b(0)$  and  $M_3 \equiv M_a(\infty)$ . Finally, the three functions  $\mathcal{H}_1(t_i)$ ,  $\mathcal{H}_2(t_i)$  and  $\mathcal{H}_3(t_i)$  are defined as

$$\mathcal{H}_1(t_i) = \begin{cases} H(t_i) & \text{if } 1 \leq i \leq N \\ -J(t_j) & \text{if } 1 \leq j \leq N \text{ with } j \equiv i - N \end{cases}, \quad (14.17)$$

$$\mathcal{H}_2(t_i) = \begin{cases} G(t_i) & \text{if } 1 \leq i \leq N \\ I(t_j) & \text{if } 1 \leq j \leq N \text{ with } j \equiv i - N \end{cases} \quad (14.18)$$

and

$$\mathcal{H}_3(t_i) = \begin{cases} (1 - H(t_i) - G(t_i)K_{ab}/K_{ba}) & \text{if } 1 \leq i \leq N \\ (J(t_j) + [1 - I(t_j)]K_{ab}/K_{ba}) & \text{if } 1 \leq j \leq N \text{ with } j \equiv i - N \end{cases}. \quad (14.19)$$

In magnetization transfer problems it is often possible to take multiple independent measurements of the exchanging spins. For example one can invert the “*a*” site and then watch the effect of the relaxation on the “*b*” site spins. Similarly, one could invert the “*b*” site and then even invert both sites simultaneously. Consequently, we are going to adopt the notation,  $d_j(t_i)$ , to designate the  $i$ th data item of the  $j$ th inversion recovery. Similarly,  $M_{j1}$  will be the initial “*a*” site magnetization for the  $j$ th inversion recovery. In what follows, it should be understood that different data sets may have different acquisition times, even though we will not adopt a notation to represent this.

The joint posterior probability for the four parameters of interest is represented symbolically by  $P(K_{ab}K_{ba}R_{1a}R_{1b}|DI)$ , where  $D$  are the amplitudes or peak intensities for the “*a*” and “*b*” sites for *all* of the data sets. If we designate the posterior probability for the parameters computed from the  $j$ th data set as  $P(K_{ab}K_{ba}R_{1a}R_{1b}|D_jI)$  then the posterior probability computed from all of the data is given by

$$P(K_{ab}K_{ba}R_{1a}R_{1b}|DI) = \prod_{j=1}^m P(K_{ab}K_{ba}R_{1a}R_{1b}|D_jI). \quad (14.20)$$

The right-hand side of this equation is computed from the joint posterior probability for all of the parameters:

$$P(K_{ab}K_{ba}R_{1a}R_{1b}|D_jI) = \int P(K_{ab}K_{ba}R_{1a}R_{1b}M_{j1}M_{j2}M_{j3}\sigma|D_jI)dM_{j1}dM_{j2}dM_{j3}d\sigma \quad (14.21)$$

where the amplitudes and standard deviation for the noise prior probability have been removed by marginalization. To compute the joint posterior probability for all of the parameters, we first factor the integrand using Bayes’ theorem:

$$\begin{aligned} P(K_{ab}K_{ba}R_{1a}R_{1b}M_{j1}M_{j2}M_{j3}\sigma|D_jI) &\propto P(K_{ab}K_{ba}R_{1a}R_{1b}M_{j1}M_{j2}M_{j3}\sigma_j|I) \\ &\times P(D_j|K_{ab}K_{ba}R_{1a}R_{1b}M_{j1}M_{j2}M_{j3}\sigma_jI). \end{aligned} \quad (14.22)$$

Next the joint prior probability for the parameters,  $P(K_{ab}K_{ba}R_{1a}R_{1b}M_{j1}M_{j2}M_{j3}\sigma_j|I)$ , is factored into a series of independent priors for each of the parameters:

$$\begin{aligned} P(K_{ab}K_{ba}R_{1a}R_{1b}M_{j1}M_{j2}M_{j3}\sigma_j|I) &= P(K_{ab}|I)P(K_{ba}|I)P(R_{1a}|I)P(R_{1b}|I) \\ &\times P(M_{j1}|I)P(M_{j2}|I)P(M_{j3}|I)P(\sigma_j|I). \end{aligned} \quad (14.23)$$

Using Eqs. (14.23,14.22,14.21 and 14.20), one obtains,

$$\begin{aligned} P(K_{ab}K_{ba}R_{1a}R_{1b}|DI) &= P(K_{ab}|I)P(K_{ba}|I)P(R_{1a}|I)P(R_{1b}|I) \\ &\times \prod_{j=1}^m \left[ \int dM_{j1}dM_{j2}dM_{j3}d\sigma_j P(\sigma_j|I) \right. \\ &\quad \times P(M_{j1}|I)P(M_{j2}|I)P(M_{j3}|I) \\ &\quad \left. \times P(D_j|K_{ab}K_{ba}R_{1a}R_{1b}M_{j1}M_{j2}M_{j3}\sigma_jI) \right] \end{aligned} \quad (14.24)$$

as the posterior probability for the parameters of interest.

We have now reached the point in the calculation where we must assign numerical values to represent these probabilities. The prior probabilities for the four parameters of interest are assigned

as Gaussians that are constructed out of the Low-High ranges that are input on the interface. If  $x$  denotes one of the four parameters of interest, then its prior probability is given by

$$P(x|I) \propto \begin{cases} \exp\left\{-\frac{(\text{Mean}-x)^2}{2\text{Sd}^2}\right\} & \text{if Low} \leq x \leq \text{High} \\ 0 & \text{otherwise} \end{cases} \quad (14.25)$$

where ‘‘Low’’ and ‘‘High’’ appropriate inputs from the interface, ‘‘Mean’’ is the average of the low and high, and ‘‘Sd’’ is set so that the High-Low interval represents a 3 standard deviation interval.

The prior probabilities for the standard deviation of the noise was assigned as a Jeffreys’ prior,  $1/\sigma_j$ . The prior probability for the amplitudes was assigned using a uniform prior probability. Finally, the likelihoods were assigned using a Gaussian of standard deviation  $\sigma_j$ , one obtains

$$\begin{aligned} P(K_{ab}K_{ba}R_{1a}R_{1b}|DI) &\propto P(K_{ab}|I)P(K_{ba}|I)P(R_{1a}|I)P(R_{1b}|I) \\ &\times \prod_{j=1}^m \left[ \int dM_{j1}dM_{j2}dM_{j3} \frac{d\sigma_j}{\sigma_j} (2\pi\sigma_j^2)^{-N} \exp\left\{-\frac{Q_j}{2\sigma_j^2}\right\} \right] \end{aligned} \quad (14.26)$$

where we have left the four prior probabilities for the parameters of interest in their symbolic form, and

$$\begin{aligned} Q_j &\equiv \sum_{i=1}^{2N_j} \left[ d_j(t_i) - \sum_{\ell=1}^3 M_{j\ell} \mathcal{H}_j(t_i) \right]^2 \\ &= 2N_j(\overline{d^2})_j - 2 \sum_{\ell=1}^3 M_{j\ell} T_{j\ell} + \sum_{k=1}^3 \sum_{l=1}^3 (g_{kl})_j M_{jk} M_{jl}. \end{aligned} \quad (14.27)$$

The mean-squared data value,  $(\overline{d^2})_j$ , for the  $j$ th inversion is defined as

$$(\overline{d^2})_j = \frac{1}{2N_j} \sum_{i=1}^{2N_j} d_j(t_i)^2 \equiv \frac{1}{2N_j} \sum_{i=1}^{N_j} d_{aj}(t_i)^2 + d_{bj}(t_i)^2 \quad (14.28)$$

where  $N_j$  is the number of complex data values in the  $j$ th inversion. The projection of the  $\ell$ th model function onto the  $j$ th inversion,  $T_{j\ell}$ , is given by

$$T_{j\ell} = \sum_{i=1}^{2N_j} d_j(t_i) \mathcal{H}_{j\ell}(t_i). \quad (14.29)$$

The matrix  $(g_{kl})_j$  is defined by

$$(g_{kl})_j \equiv \sum_{i=1}^{2N_j} \mathcal{H}_{jk}(t_i) \mathcal{H}_{j\ell}(t_i). \quad (14.30)$$

This matrix will depend on the individual inversion if the delay times differ from one inversion to another.

Evaluating the integrals over the amplitudes and standard deviations is straightforward, if not tedious. Evaluating the amplitude integrals one obtains:

$$\begin{aligned} P(K_{ab}K_{ba}R_{1a}R_{1b}|DI) &\propto P(K_{ab}|I)P(K_{ba}|I)P(R_{1a}|I)P(R_{1b}|I) \\ &\times \prod_{j=1}^m \int \frac{d\sigma_j}{\sigma_j} |g_{kl}|_j^{-\frac{1}{2}} (2\pi\sigma_j^2)^{-N+3} \exp\left\{-\frac{2N_j(\overline{d^2})_j - (\overline{h^2})_j}{2\sigma_j^2}\right\} \end{aligned} \quad (14.31)$$

where  $|g_{kl}|_j$  is the determinant of the  $(g_{jk})_j$  matrix. The sufficient statistic,  $(\overline{h^2})_j$ , is the total-squared projection of the model onto the data for the  $j$ th inversion:

$$(\overline{h^2})_j = \sum_{k=1}^3 T_{kj} \hat{M}_{kj}. \quad (14.32)$$

The  $\hat{M}_{kj}$ , are given by the solution to the following linear set of equations:

$$\sum_{k=1}^3 (g_{lk})_j \hat{M}_{kj} = T_{lj} \quad (1 \leq l \leq 3). \quad (14.33)$$

The integrals over the standard deviations of the noise prior probabilities may all be transformed into Gamma functions and evaluating such integrals is straightforward, one obtains

$$\begin{aligned} P(K_{ab}K_{ba}R_{1a}R_{1b}|D_aD_bI) &\propto P(K_{ab}|I)P(K_{ba}|I)P(R_{1a}|I)P(R_{1b}|I) \\ &\times \prod_{j=1}^m |g_{kl}|_j^{-\frac{1}{2}} \left[ 2N_j(\overline{d^2})_j - (\overline{h^2})_j \right]^{\frac{3-N_j}{2}} \end{aligned} \quad (14.34)$$

where we have dropped a number of constants that cancel when this distribution is normalized.

Markov chain Monte Carlo with simulated annealing is used to draw samples from the joint posterior probability for  $K_{ab}$ ,  $K_{ba}$ ,  $R_{1a}$  and  $R_{1b}$ , Eq. (14.34). These samples are used to approximate the marginal posterior probability for each of the parameters separately. In addition, the program also uses these samples to form an approximation to the posterior probability for the volume fractions,

$$p_a = \frac{K_{ba}}{K_{ab} + K_{ba}} \quad \text{and} \quad p_b = \frac{K_{ab}}{K_{ab} + K_{ba}}, \quad (14.35)$$

the exchange times,

$$\tau_{ab} = \frac{1}{K_{ab}} \quad \text{and} \quad \tau_{ba} = \frac{1}{K_{ba}}, \quad (14.36)$$

and the relaxation times,

$$T_{1a} = \frac{1}{R_{1a}} \quad \text{and} \quad T_{1b} = \frac{1}{R_{1b}}. \quad (14.37)$$

Note that in each of these calculations involves a change of variables of the form  $Y = 1/X$ . This change of variables introduces a factor of the form  $dY = -dX X^{-2}$  into the posterior. This factor shift the location of the maximum posterior probability between the paired variables. Consequently, the maximum posterior probability estimate for  $Y$  is not in general equal to  $1/X$ .

## 14.2 Using The Package

The Text outputs files from the Magnetization Transfer packages consist of: “Bayes.prob.model,” “MtZ.mcmc.values,” “Bayes.params,” “Console.log,” “Bayes.accepted” and a “Bayes.Condensed.File.” These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the

Figure 14.2: Magnetization Transfer Package Peak Picking

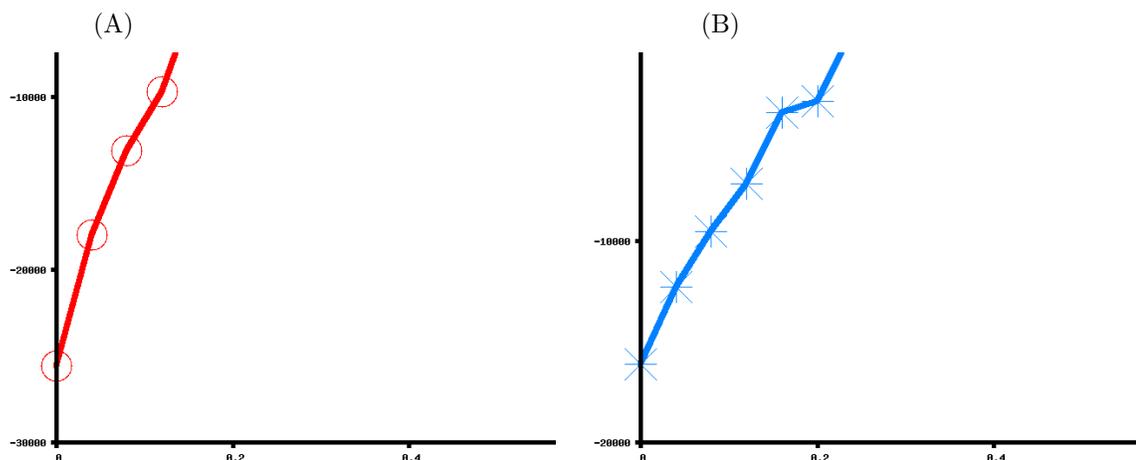


Figure 14.2: The peak intensities of the exchanging sites are analyzed by the magnetization exchange package, consequently, as illustrated here, the data are three column Ascii: a time, and two peak intensities. These peak intensities may be loaded from a peak pick, a Bayes Analyze run, or from an Ascii file. The example shown in this figure is from an peak pick. Panel (A) is the peak intensities when the left-hand peak was inverted and Panel (B) is the peak intensities when the right-hand peak was inverted. The data shown in Panel (A) was generated using a peak pick of the inversion recovery spectrum shown in Figure 14.3.

mcmc.values report is discussed in Appendix D and the other reports are discussed in Chapter 3. Additionally, the “Plot Results Viewer” can be used to view the output probability density functions. In addition to the standard data, model and residual plots there are probability density functions for the decay rate constants, decay times, the amplitudes for each data set for each exponential and finally there are probability density functions for the standard deviation of the noise in each data set.

The data used by this package can be multiple three column Ascii files, see Fig. 14.2 for an example of this data. The data consists of a time axis and the left and right-hand peak intensity from an inversion recover experiment. The data can be loaded in one of three ways: First, a three column Ascii file can be directly loaded using the Files menu; Second, The spectrum of a magnetization transfer inversion recovery Fid can be loaded and the peak amplitudes can be extracted and loaded. Third, the “File/Load Ascii/Bayes Analyze File” button can be used to extract the peak amplitudes from the currently loaded Bayes Analyze files. If Bayes Analyze has not been run on this Fid, you must run it before you can use this option.

Figure 14.3 is the first trace in the spectrum of the inversion recover used to generate the data shown in Fig. 14.2(A). Figure 14.4 is a plot of the spectrum of the two exchanging resonances for each delay time in the inversion recover experiment shown in Fig. 14.3. Note how the left-hand resonances starts inverted and as a function of the delay time it rather quickly recovers. If you examine Fig. 14.4, which is a VnmrJ dssh display of the region around the two exchanging peaks, you can easily see that, initially both peaks have reduced amplitude and as the left-hand peak recovers, both peaks

Figure 14.3: Magnetization Transfer Example Data

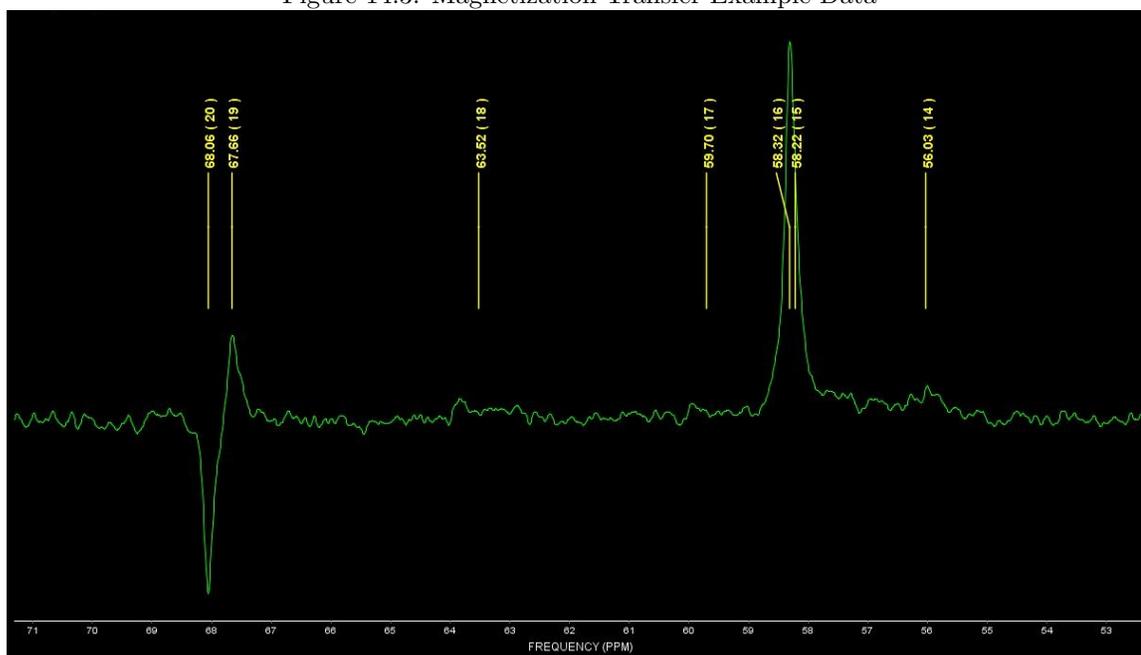


Figure 14.3: The three columns are the abscissa, in this case a delay time, and the amplitudes of the resonances of the A and B Sites. In this spectrum Site A is the resonance near 68PPM, and Site B is near 67.66PPM. To load this data, place a double cursor on each resonance and hit the “Get Peak” button in the lower right. Alternately, If Bayes Analyze has been run on this data, the “File/Load Ascii/Bayes Analyze File” menu can be used to load the resonance amplitudes from the Bayes Analyze files.

Figure 14.4: Magnetization Transfer Example Spectrum

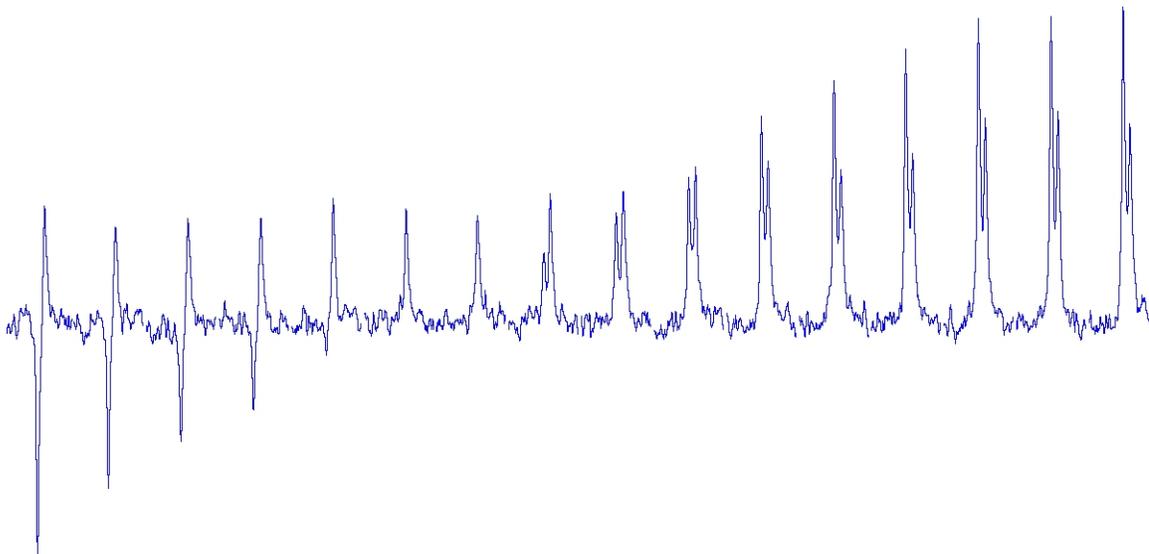


Figure 14.4: These spectra are a typical example of magnetization transfer data. Here the left-hand peak was selectively inverted and allowed to recover. The data analyzed by the magnetization transfer package is the peak intensity or amplitude of the two sites exchanging magnetization. These peak intensities may be loaded using the File menu, extracted using a double cursor and the Get Peak button, or Bayes Analyze can be used to estimate the amplitudes, and these amplitudes can be loaded using the File/Ascii File/Bayes Analyze menu.

increase in intensity. In the three column Ascii data shown in Fig 14.2(A) the left-hand peak shown in Fig 14.3 was selectively inverted and then allowed to relax back to equilibrium. It is the peak intensities of the two resonances exchanging magnetization that are analyzed by the magnetization package.

## Chapter 15

# Magnetization Transfer Kinetics

The Magnetization Transfer Kinetics Package analyzes two site magnetization exchange data as a function of temperature. To accomplish this the equations governing two site exchange are supplemented with the Eyring equation. This equation gives the exchange rate constants as a function of temperature in terms of four more fundamental quantities. These four quantities are the entropies and enthalpies of activation,  $\Delta H_{ab}$ ,  $\Delta H_{ba}$ ,  $\Delta S_{ab}$ ,  $\Delta S_{ba}$  respectively. The primary purpose of this package is to infer these four quantities. For more on these calculations and for a better description of the thermodynamics used in this chapter see [21, 22, 26]. The interface to this package is shown in Fig. 15.1. Note this interface has four package specific widgets: a widget used to set the temperature of each sample, a widget to set the uncertainty in the temperature, a widget to load a viscosity table, and one to display the current viscosity table. To use this package you must:

**Select** the Magnetization Transfer Kinetic Package from the Package menu.

**Load** one or more three column Ascii data sets using either the “Files/Load Ascii/File” menu. Data can also be loaded by loading an Fid and selecting the two exchanging resonances using a double cursor and hitting the “Get Peak” button, or Bayes Analyze amplitude estimates can be loaded using the “Files/Load Ascii/Bayes Analyze” menu. When a data set is successfully loaded the data is plotted in the Ascii Data viewer, see Fig. 14.2 for an example of such data.

**Using** the “Set” entry box, enter the temperature for each set as you load it. Alternatively, you can display each data set in the Ascii File viewer and then enter a temperature for it. Note, this package will not run until a temperatures has been set for each data set.

**Using** the “Uncertainty” widgets set the uncertainty in the temperature measurement. It is assumed that all of the data were acquired on the same sample and that the signal-to-noise ratio is not changing as a function of temperature. Consequently, it is assumed that there is a single uncertainty and it is applied to all temperatures.

**Load** a viscosity table if the solvent is not water. The format of the viscosity table is shown in Fig. 15.3 and is discussed later.

**Review** the prior probabilities for parameters would normally go here, but the Magnetization Transfer Kinetics Package sets its priors internally. So there are no prior probabilities to review.

Figure 15.1: Magnetization Transfer Kinetics Package Interface

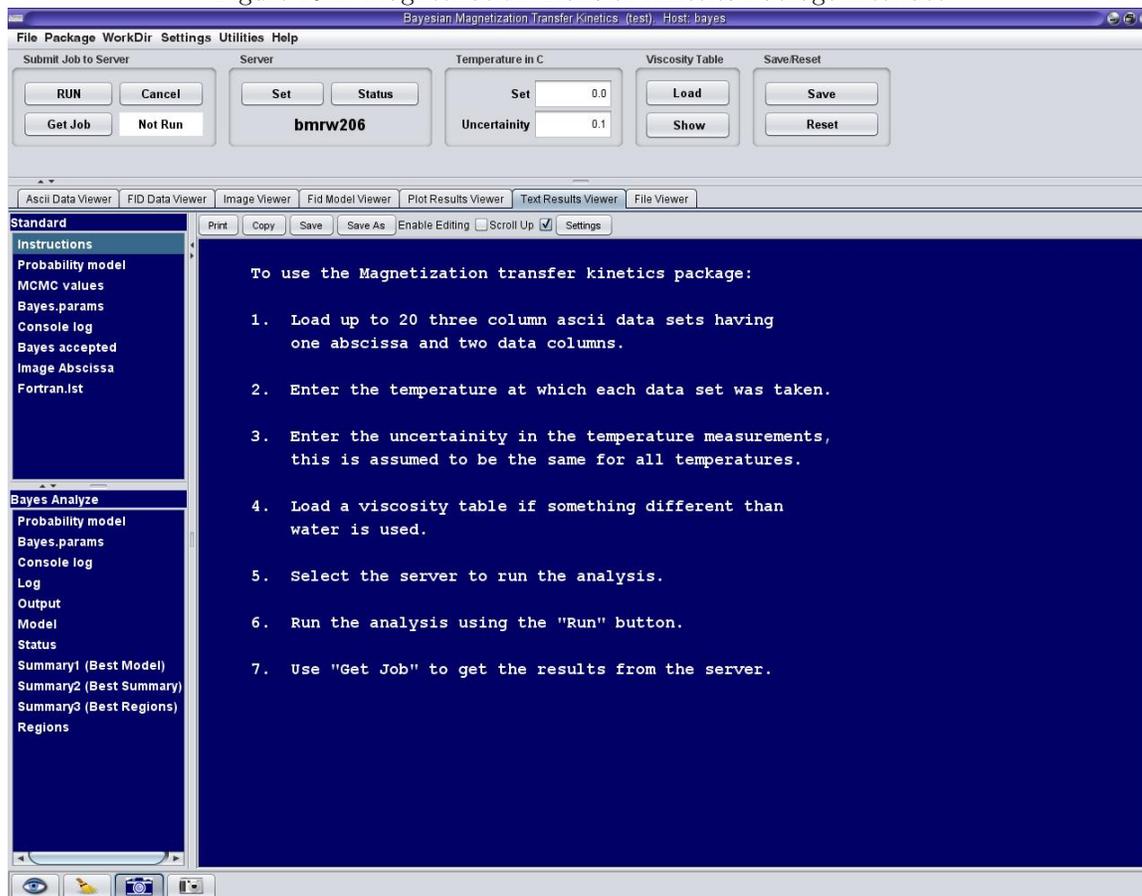


Figure 15.1: The magnetization transfer Kinetics Packages analyzes up to 20 data set having no more than 100 data values using the equations governing two site magnetization exchange with the Eyring equation to predict the exchange rates. The inferred parameters are the Entropy and Enthalpy of activation. For more on the actual calculations and the widgets see the text.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

## 15.1 The Bayesian Calculation

The Magnetization Transfer Kinetics Package analyzes problems involving magnetization transfer. The differential equations governing this exchange are the McConnell modification of the Bloch equations, Eq. (14.3,14.4). For a fixed temperature,  $\tau_j$ , the “a” and “b” site magnetizations must satisfy these equations. These magnetizations, Eq. (14.5 and 14.6), are related to the data by

$$d_a(t_i, \tau_j) = M_a(t_i, \tau_j) + \text{error} \quad (15.1)$$

and

$$d_b(t_i, \tau_j) = M_b(t_i, \tau_j) + \text{error} \quad (15.2)$$

where  $d_a(t_i, \tau_j)$  is the “a” site data gathered at time  $t_i$  at temperature  $\tau_j$ . Similarly,  $d_b(t_i, \tau_j)$  are the “b” site data. The functions,  $M_a(t_i, \tau_j)$  and  $M_b(t_i, \tau_j)$ , are given by Eqs. (14.5 and 14.6) respectively and “error” represents the misfit between the data and the model. The functions  $M_a(t_i, \tau_j)$  and  $M_b(t_i, \tau_j)$  are functions of the exchange rates  $K_{ab}$  and  $K_{ba}$  as well as the relaxation rates  $R_{1a}$  and  $R_{1b}$ . The exchange rates  $K_{ab}(\tau_j)$  and  $K_{ba}(\tau_j)$  are calculated from the Eyring equations,

$$K_{ab}(\tau_j) = \kappa \left( \frac{k\tau_j}{\hbar} \right) \exp \left\{ \frac{\Delta S_{ab}}{R} - \frac{\Delta H_{ab}}{R\tau_j} \right\}, \quad (15.3)$$

and

$$K_{ba}(\tau_j) = \kappa \left( \frac{k\tau_j}{\hbar} \right) \exp \left\{ \frac{\Delta S_{ba}}{R} - \frac{\Delta H_{ba}}{R\tau_j} \right\} \quad (15.4)$$

respectively where  $\kappa$  is a transmission coefficient, here set equal to one,  $k$  is Boltzmann’s constant,  $\hbar$  is Plank’s constant and  $R$  the universal gas constant. Over the temperature range that most experiments can be performed the relaxation times,  $T_{1a}$  and  $T_{1b}$ , can be expanded as a linear function of temperature:

$$\frac{1}{R_{1a}} = T_{1a} = T_{1a\infty} + \Delta T_{1a\infty} \frac{v(\tau_j)}{\tau_j} \quad (15.5)$$

and

$$\frac{1}{R_{1b}} = T_{1b} = T_{1b\infty} + \Delta T_{1b\infty} \frac{v(\tau_j)}{\tau_j} \quad (15.6)$$

where  $T_{1a\infty}$  is the relaxation time at infinite temperature,  $\Delta T_{1a\infty}$  is the slope of this line, and  $v(\tau_j)$  is the viscosity at temperature  $\tau_j$ . Similar definitions hold for the “b” site.

These equations along with the solution to the Bloch-McConnell given enough structure to the problem to be able to solve this problem using Bayesian probability theory and we will do that shortly. However, before doing that we want to note one problem. The prefactor multiplying the exponential in the Eyring equations is on the order of  $6 \times 10^{12} \text{ s}^{-1}$ . So any attempt to modify  $\Delta H_{ab}$  or  $\Delta S_{ab}$  must be done carefully because a tiny change in the exponential will result in a huge change in the exchange rate,  $K_{ab}$ . Consequently, the program that implements this calculation uses a type of sum and difference variables that change both  $H_{ab}$  and  $S_{ab}$  simultaneously in a way that avoids this problem.

The parameters of interest are the entropy, and enthalpy of activation,  $\Delta H_{ab}$ ,  $\Delta H_{ba}$ ,  $\Delta S_{ab}$ ,  $\Delta S_{ba}$ , and to a lesser degree the parameters used in the linear expansion of the relaxation time,  $T_{1a\infty}$ ,  $\Delta T_{1a\infty}$  and  $T_{1b\infty}$ ,  $\Delta T_{1b\infty}$ . In addition, we have designated the true temperature as  $\tau_j$  and it should be clear that these quantities are unknown parameters about which we have to make inferences. We do have a measured value,  $\tau'_j$ , that is related to  $\tau_j$  by

$$\tau'_j = \tau_j + \text{error} \quad (15.7)$$

where “error” represents the error in the temperature measurement. In the course of this calculation, we will eventually have to assign a prior probability for this temperature error and when we do that we will assign a Gaussian prior centered on the measurement and the Gaussian will have a standard deviation  $\sigma_{\tau_j}$  which we will assume is known. So when the program runs it tries to vary these temperatures, consistent with what is known about the temperature uncertainty.

The Bayesian posterior probability for the entropies and enthalpies given the exchange data has much in common with the calculation presented in Chapter 14 and if you have not reviewed that material it would be advisable to do so now. In what follows we will use much of the same notation introduced in that Chapter. For example the three amplitudes of the  $j$ th inversion will be designated as  $M_{j1}$ ,  $M_{j2}$  and  $M_{j3}$ . Similarly, the standard deviation of the noise prior probability used to assign the likelihood of the  $j$ th data set will be represented by  $\sigma_j$ . If we designate the set of interesting parameters as  $\Theta$ ,

$$\Theta \equiv \{\Delta H_{ab}, \Delta H_{ba}, \Delta S_{ab}, \Delta S_{ba}, T_{1a\infty}, \Delta T_{1a\infty}, T_{1b\infty}, \Delta T_{1b\infty}, \tau_1, \dots, \tau_m\}, \quad (15.8)$$

the set of amplitudes as  $M$ , and the set of standard deviations as  $\sigma$  then the posterior probability for the interesting parameters given all the data  $D$  is given by

$$P(\Theta|DI) = \int P(\Theta M \sigma | DI) dM d\sigma \quad (15.9)$$

where the marginalization integrals are over all of the amplitudes and standard deviations. Applying Bayes' theorem to the right-hand side of this equation one obtains

$$P(\Theta|DI) \propto \int P(\Theta M \sigma | I) P(D | \Theta M \sigma I) dM d\sigma. \quad (15.10)$$

If we now factor the prior probability for all of the parameters as

$$P(\Theta M \sigma | I) = P(\Theta | I) \prod_{j=1}^m [P(M_{j1} | I) P(M_{j2} | I) P(M_{j3} | I) P(\sigma_j | I)] \quad (15.11)$$

where for the time being we have left the prior probability for the interesting parameters unfactored. Similarly, the probability for the data,  $P(D|\Theta M\sigma I)$ , may be factored into a product of likelihoods given by

$$P(D|\Theta M\sigma I) = \prod_{j=1}^m P(D_j|\Theta M_{j1}M_{j2}M_{j3}\sigma_j I) \quad (15.12)$$

where  $P(D_j|\Theta M_{j1}M_{j2}M_{j3}\sigma_j I)$  is the likelihood used in Chapter 14. If we now substitute the likelihood Eq. (15.12) and the prior Eq.(15.11) into the joint posterior probability of the interesting parameters, Eq. (15.10), one obtains

$$P(\Theta|DI) \propto P(\Theta|I) \prod_{j=1}^m \left[ \int P(M_{j1}|I)P(M_{j2}|I)P(M_{j3}|I)P(\sigma_j|I) \right. \\ \left. \times P(D_j|\Theta M_{j1}M_{j2}M_{j3}\sigma_j I) dM_{j1}dM_{j2}dM_{j3}d\sigma_j \right] \quad (15.13)$$

as the posterior probability for the interesting parameters. If you examine the quantity in the big square brackets, except for a little notational differences, you will discover that this calculation is exactly the same calculation that occurred when we did the Magnetization Transfer calculation in Chapter 14. The notation difference relates to the fact that in the Magnetization Transfer calculation we were trying to infer the exchange rates, while here its the entropy and enthalpy of activation that is being inferred. If we now substitute the posterior probability computed in Chapter 14 into Eq. (15.13), one obtains

$$P(\Theta|DI) \propto P(\Theta|I) \prod_{j=1}^m |g_{lk}|^{-\frac{1}{2}} \left[ 2N_j(\bar{d}^2)_j - (\bar{h}^2)_j \right]^{\frac{3-N_j}{2}} \quad (15.14)$$

as the posterior probability for the interesting parameters. If we now use the product rule to factor the prior probability for the interesting parameters, Eq. (15.14) becomes

$$P(\Theta|DI) \propto P(\Delta H_{ab}|I)P(\Delta H_{ba}|I)P(\Delta S_{ab}|I)P(\Delta S_{ba}|I) \\ \times P(T_{1a\infty}|I)P(T_{1b\infty}|I)P(\Delta T_{1a\infty}|I)P(\Delta T_{1b\infty}|I) \\ \times \prod_{j=1}^m P(\tau_j|\tau'_j I) |g_{lk}|^{-\frac{1}{2}} \left[ 2N_j(\bar{d}^2)_j - (\bar{h}^2)_j \right]^{\frac{3-N_j}{2}} \quad (15.15)$$

which is the posterior probability for the interesting parameters.

The probability for the temperature,  $\tau_j$ , given the measured value of the temperature,  $\tau'_j$ , will be assigned a Gaussian of the form:

$$P(\tau_j|\tau'_j I) \propto \exp \left\{ -\frac{(\tau'_j - \tau_j)^2}{2\sigma_{\tau_j}^2} \right\} \quad (15.16)$$

where the uncertainty in the temperature,  $\sigma_{\tau_j}$ , is input from the interface.

Finally, if we assign a uniform prior probability for  $P(\Delta H_{ab}|I)$ ,  $P(\Delta H_{ba}|I)$ ,  $P(\Delta S_{ab}|I)$ ,  $P(\Delta S_{ba}|I)$ ,  $P(T_{1a\infty}|I)$ ,  $P(T_{1b\infty}|I)$ ,  $P(\Delta T_{1a\infty}|I)$ , and  $P(\Delta T_{1b\infty}|I)$  the final posterior probability for the interesting parameters becomes

$$P(\Theta|DI) \propto \prod_{j=1}^m \exp \left\{ -\frac{(\tau'_j - \tau_j)^2}{2\sigma_{\tau_j}^2} \right\} |g_{lk}|^{-\frac{1}{2}} \left[ 2N_j(\overline{d^2})_j - (\overline{h^2})_j \right]^{\frac{3-N_j}{2}}. \quad (15.17)$$

It is this posterior probability that is implemented in the Markov chain Monte Carlo simulations.

In addition to the  $\Theta$  parameters the analysis also outputs samples of a number of derived parameters. That is to say the program uses the samples of the interesting parameters to compute some other quantities of interest that can be computed from the interesting parameters, these include:  $K_{ab}$ ,  $K_{ba}$ ,  $\tau$ ,  $T_{1a}$ ,  $T_{1b}$ ,  $R_{1a}$ ,  $R_{1b}$ ,

$$K_{eq} \equiv \frac{K_{ab}}{K_{ba}}, \quad P_a = \frac{K_{ba}}{K_{ab} + K_{ba}}, \quad P_b = \frac{K_{ab}}{K_{ab} + K_{ba}}, \quad \tau_{ab} \equiv \frac{1}{K_{ab}} \quad \text{and} \quad \tau_{ba} \equiv \frac{1}{K_{ab}}. \quad (15.18)$$

Note that each of these quantities is temperature dependent, and the outputs include a posterior probability for these parameters at each temperature.

We mentioned earlier that the large prefactor multiplying the exponential in the Eyring equation introduces substantial correlations in the values of the entropy and enthalpy. This problem is so severe, that the program that does this calculation uses a set of sum and difference variables that automatically change  $\Delta H_{ab}$ ,  $\Delta H_{ba}$ ,  $\Delta S_{ab}$ , and  $\Delta S_{ba}$  in a way that keeps Eqs. (15.3 and 15.4) greater than zero and finite. The real problem is that if you make a change to  $\Delta S$ , either “ $ab$ ” or “ $ba$ ,” and do not make an appropriate change to  $\Delta H$ , the exponential could change by many orders of magnitude. Indeed changing  $\Delta S$  by even one and not changing  $\Delta H$  could change the exchange rates  $10^{10}$ ! The way the program prevents this is to introduce a type of sum and difference variables:

$$U_x = \frac{\Delta S_x}{R} - \frac{\Delta H_x}{RT^*} \quad (15.19)$$

and

$$V_x = \frac{\Delta S_x}{R} + \frac{\Delta H_x}{RT^*} \quad (15.20)$$

where  $x$  means either the “ $ab$ ” or the “ $ba$ ” sites and  $T^*$  is the average input temperature at which the various inversion recovery data sets were acquired. If you look at Eq. (15.3) you will find that  $U_x$  is just the exponent of this equation evaluated at the average temperature. Similarly,  $V_x$  is the exponent but with a plus sign, again evaluated at the average temperature. When running the analysis small changes, on the order of 0.01, can be made to  $U_x$  while changes on the order of one or two can be made to  $V_x$ . These changes translate into substantial changes in the exchange rates. However, the problem remains of how to pick a set of  $U_x$  and  $V_x$  that can be used to initialize the Markov chain Monte Carlo simulations. The program does this by generating set of enthalpies distributed random around zero with a standard deviation of 10,000:

$$\Delta S_x = \pm \sigma_{10,000}. \quad (15.21)$$

Next the program generates an entropy

$$\Delta H_x = \left[ \Delta S_x + R \log \left( \frac{KT^*}{\hbar} \right) \right] T^* \pm \sigma_3 \quad (15.22)$$

where by  $\pm\sigma_3$  we mean that the program add a Gaussian distributed random number of standard deviation 3 to this quantity. If you now look at Eq. (15.3) and plug in these numbers you will find that

$$K_x = \exp\{\pm\sigma_3\}. \quad (15.23)$$

So generating the entropies and enthalpies in this way allows the program to initialize the entropies and enthalpies in such a way that they cover all reasonable values of these quantities. Finally, the program computes these sum and difference variables from the generated entropies and enthalpies, and using these sum and difference variables the program can then thoroughly explore the parameter space.

## 15.2 Using The Package

The Text outputs files from the Magnetization Transfer Kinetics Package consist of: “Bayes.prob.model,” “BayesMtZKinetics.mcmc.values,” “Bayes.params,” “Console.log,” “Bayes.accepted” and a “Bayes.Condensed.File.” These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the mcmc.values report is discussed in Appendix D and the other reports are discussed in Chapter 3. Additionally, the “Plot Results Viewer” can be used to view the output probability density functions. In addition to the standard data, model and residual plots for each data set, there are probability density functions for each parameter appearing in the model, as well as probability density functions for the derived parameters mentioned earlier. Because some of the parameters are temperature dependent, the number of probability density functions can be fairly large, one the order of a hundred.

In addition to the standard plots, the Magnetization Transfer Kinetics Package outputs several additional scatter plots. These plots are thermodynamics plots and are intended to show you how uncertain you are of the thermodynamics. An example of these plots is shown in Fig. 15.2. This figure contains four panels: Panels (A) and (B) contain an Arrhenius and van’t Hoff plot. The Arrhenius plot is for the  $K_{ab}$  exchange rate. The simulations also produce an Arrhenius plot for the  $K_{ba}$  exchange rate, although we did not show it here. Panel (C) contains a plot of the Gibbs free energy for the “a” site going to the transition state, and there to the  $b$  site. Again the simulations also produces a plot of the Gibbs free energy for the  $b$  site going to the transition state and then onto the “a” site. Finally, panel (D) is a plot of the relaxation time,  $T_{1a}$ , for the “a” site. The plot for the  $b$  site is not shown.

The expansion of the relaxation times requires the viscosity of the solvent in the expansion. The program uses the viscosity of water, but allows the user to supply alternative viscosity tables. An example of the viscosity table is shown in Table 15.3. This table contains the first few lines of the viscosity table located in “/vnmr/bayes/Bayes.test.data”. The file name is “WaterViscosityTable”. The table must be uniformly sampled. The first two lines in the table are the low and high temperature covered by the table in Kelvin. The third line is the number of entries in the table, in this case 101. Finally the viscosity entries are in units of “cp”. You can change the viscosity table using the interface by activating the “Load” button under the Viscosity Table label. The maximum number of entries in this table is 501, and the minimum is 2, although using this minimum is probably not very desirable. When the package is running the viscosity is needed as a continuous parameter and to do this from a discrete table the program uses linear interpolation. Linear interpolation requires a reasonably good digitization of the viscosity table to ensure the interpolated values are accurate.

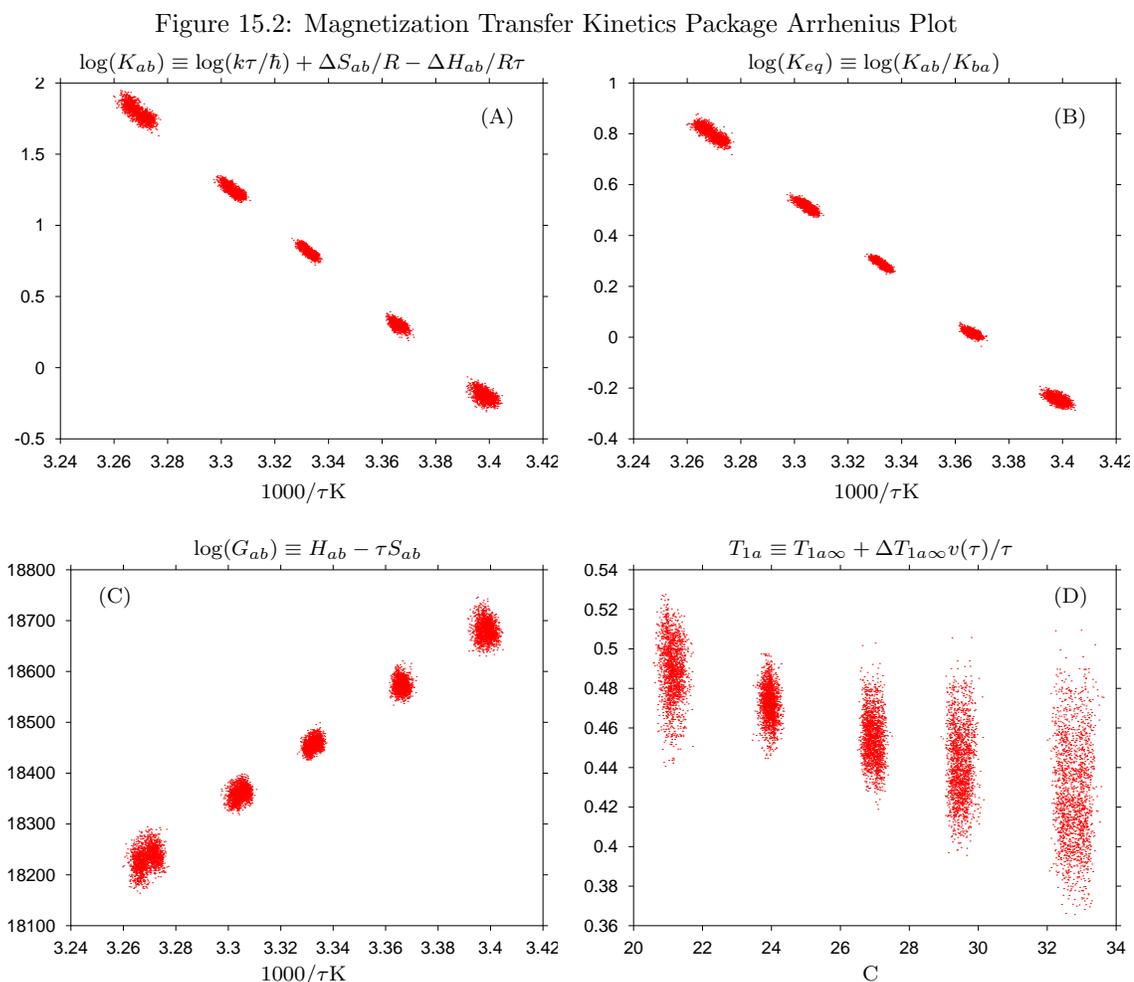


Figure 15.2: Panel (A) Arrhenius plot of the exchange rate. Panel (B) van't Hoff plot of the equilibrium constant. Panel (C) is the natural logarithm of the Gibbs free energy. Panel (D) is a plot of the  $T_{1a}$  relaxation time as a function of temperature in Centigrade. Similar, plots are made for the both sites, although we have shown only one here.

Figure 15.3: Magnetization Transfer Kinetics Water Viscosity Table

273.15	The low temperature in Kelvin
373.15	The high temperature in Kelvin
101	The number of entries in the table
1.787	Water viscosity at 273.15K
1.728	Water viscosity at 274.15K
1.671	Etc.
1.618	
1.567	
1.519	
1.472	
1.428	
1.386	
1.346	

Figure 15.3: This table contains the first few lines out of the water viscosity table used in the simulation. The table is a single column ASCII file. The comments in the above table are for explanation only. The viscosity table is strictly the first column.

In the water viscosity table there are 101 total entries, one at each degree Kelvin, and this is more than enough entries for water. I would recommend you do the same. As far as what temperature range to cover, you must cover all temperatures the program is likely to explore. For example if you data span a temperature range of 30C starting at 10C and ending at 40C and you are uncertain of the temperature to 1 degree, then you will probably need to supply viscosity's from about 5C up to 45C where I have made the assumption that the simulations are very unlikely to explore a five standard deviation temperature variation.



## Chapter 16

# Given Polynomial Order

The Given polynomial Order package fits polynomials to two column Ascii data when the order of the polynomial is known. The interface to the Given Polynomial Order package is shown in Figure 16.1. This interface differs from most others in one respect, there are no parameter ranges to enter, so use of the interface is particularly simple. To use this package, you must do the following:

**Select** the Polynomial Models package from the Package menu. When selected this menu will bring up the “Given” and “Unknown” polynomial model interface.

**Load** one two column Ascii data sets. The data may be loaded using the Files menu. You can also load an arrayed Fid and then use a single cursor to mark the center of a peak and use the “Get Peak” button on the bottom right of the Fid viewer. Finally, the “Files/Load Ascii/Bayes Analyze” button can be used to load an Ascii data set from the amplitudes estimated by Bayes Analyze. When a data set is successfully loaded the data is plotted in the Ascii Data viewer. This package does not allow you to run with multiple data sets. If you attempt to do so, you will be prompted to remove all but a single file.

**Set** the Polynomial order using the “Set Order” selection widget. For the Given Polynomial Order, the order can be from 1 to 55.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

Figure 16.1: Given Polynomial Order Package Interface

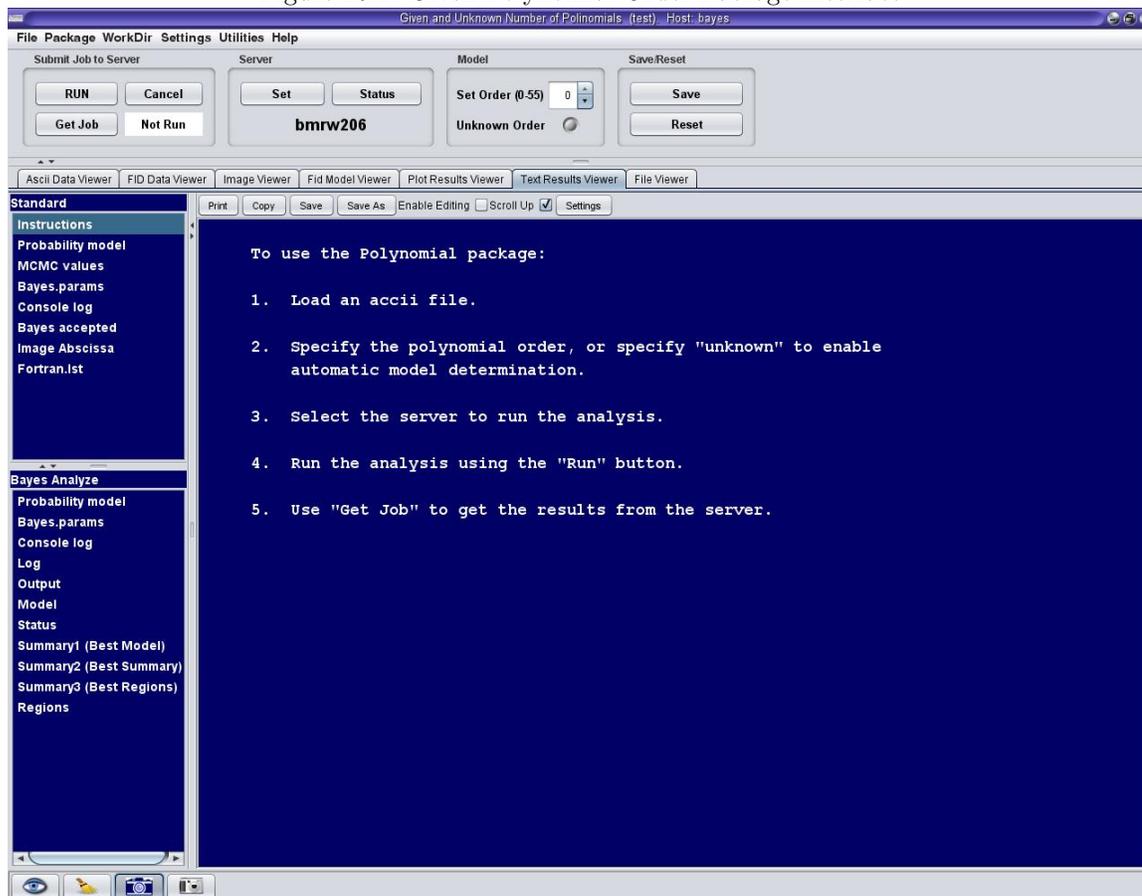


Figure 16.1: This panel is interface to the Given Polynomial Order package. Because of the way this calculation is done very high orders are possible and numerically stable. However, the high orders, above 40, require very high signal-to-noise and even then roundoff errors degrade the accuracy to 4 or 5 decimal places.

## 16.1 The Bayesian Calculation

The polynomial model is just that, its a model in which a polynomial is fit to the data:

$$d_i = \sum_{j=0}^m A_j G_j(t_i) + n_i \quad (16.1)$$

with

$$G_j(t_i) = t_i^j \quad (16.2)$$

where  $A_j$  is the amplitude of the  $j$ th polynomial,  $n_i$  represents noise in the  $i$ th data value and we have written these polynomials as  $G_j(t_i)$  for notational convenience. We will think of these polynomials as functions of time, but in the analysis which follows  $t$  is simply a single column abscissa and may be any quantity. In the problem we are describing here,  $m$  is the given order of the polynomial. Additionally, the assumption that these are polynomials is unimportant in the following discussions, the  $G_j(t_i)$  could be any set of functions.

### 16.1.1 Gram-Schmidt

The Bayesian calculation is implemented using Markov chain Monte Carlo with simulated annealing to draw samples for the joint posterior probability for the parameters. Before we do this calculation we introduce a change of function and a change of variables. The reason for this is simply that computing polynomials of the form  $\sum_{j=0}^m A_j t_i^j$  is computationally vary unstable in the sense that only orders up to about 8 can be computed using single precision arithmetic. You can get to higher orders only by using numerical procedures that avoid differencing large numbers, for example the remainder theorem. However, here we take a different approach by transforming the problem to something that is computationally more stable. Using Gram-Schmidt, the polynomials are transformed to a set of orthogonal polynomials. We then solve the problem using these orthogonal polynomials and finally, we transform the derived amplitudes back to the  $A_j$  given in Eq. (16.1). If we designate the Gram-Schmidt polynomials as  $H_j(t_i)$  and the expansion coefficients as  $B_j$ , Eq. (16.1) becomes:

$$d_i = \sum_{j=0}^m B_j H_j(t_i) + n_i. \quad (16.3)$$

We chose the Gram-Schmidt polynomials because they can be computed trivially from the  $t_i^j$ , they preserve the notation of the order of the polynomial, and each polynomial depends only on the lower orders, not the higher orders. This change of variables and change of function is an identity, i.e., the polynomial expansions in Eq. (16.1) and Eq. (16.3) are exactly equal to each other. Finally, the amplitudes,  $B_j$  and  $A_j$ , are linearly related to each other through an lower triangular matrix, and consequently, the conversion back and forth between these representations is very easy to program.

As a reminder to those unfamiliar with Gram-Schmidt polynomials, the normalized Gram-Schmidt polynomials  $H_j(t_i)$ , are generated recursively from the  $G_j(t_i)$  using:

$$H_j(t_i) = \frac{1}{C_{jj}} \left[ G_j(t_i) - \sum_{\ell=0}^{j-1} C_{j\ell} H_\ell(t_i) \right] \quad (16.4)$$

where the sum is not present for the first polynomial, and

$$C_{j\ell} = \sum_{i=1}^N G_j(t_i) H_\ell(t_i) \quad (0 \leq j \leq \ell). \quad (16.5)$$

Gram-Schmidt polynomials have the property

$$\sum_{i=1}^N H_j(t_i) H_\ell(t_i) = \delta_{j\ell} \quad (16.6)$$

where  $\delta_{j\ell}$  is zero if  $j \neq \ell$  and one if  $j = \ell$ .

To derive the relationship between the  $A_j$  and the  $B_j$ , note that the expansions given by Eq. (16.1) and Eq. (16.3) are identities, so can write

$$\sum_{k=0}^m A_k G_k(t_i) = \sum_{j=0}^m B_j H_j(t_i) \quad (16.7)$$

where we changed the summation index on the left-hand side just to remind people that these summations are independent of each other. Multiplying this equation by  $H_\ell(t_i)$ , and summing over time:

$$\sum_{i=1}^N \sum_{k=0}^m A_k G_k(t_i) H_\ell(t_i) = \sum_{i=1}^N \sum_{j=0}^m B_j H_j(t_i) H_\ell(t_i). \quad (16.8)$$

The right-hand side of this equating is zero unless  $j = \ell$  and then one obtains

$$\sum_{i=1}^N \sum_{k=0}^m A_k G_k(t_i) H_\ell(t_i) = B_\ell \quad (16.9)$$

and the sum over time on the left-hand side of this equation can be written as

$$\sum_{k=0}^m A_k \left[ \sum_{i=1}^N G_k(t_i) H_\ell(t_i) \right] = B_\ell. \quad (16.10)$$

The quantity in big square brackets is just the right-hand side of Eq. (16.5), so this equation becomes

$$\sum_{k=0}^m A_k C_{k\ell} = B_\ell. \quad (16.11)$$

The matrix  $C_{k\ell}$  is a lower triangular matrix, so inverting it is trivial and one can use this equation solve for the nonorthogonal expansion coefficients, the  $A_k$ , from the orthogonal expansion coefficients, the  $B_j$ .

### 16.1.2 The Bayesian Calculation

The Bayesian calculation is for the joint posterior probability for the amplitudes,  $B_j$ , given the data and the prior information. This joint probability, denoted by  $P(B_0 B_1 \dots B_m | DI)$ , is computed by application of Bayes' theorem

$$P(B_0 B_1 \dots B_m | DI) \propto P(B_0 B_1 \dots B_m | I) P(D | B_0 B_1 \dots B_m I) \quad (16.12)$$

where  $P(B_0B_1 \dots B_m|I)$  is the joint prior probability for the amplitudes, and  $P(D|B_0B_1 \dots B_mI)$  is the likelihood. Because each polynomial is orthogonal, we will factor the joint prior probability for the amplitudes,  $P(B_0B_1 \dots B_m|I)$ , into a series of independent prior probabilities:

$$P(B_0B_1 \dots B_m|I) = \prod_{j=0}^m P(B_j|I) \quad (16.13)$$

and each of the  $P(B_j|I)$  will be assigned a unbound Gaussian. The posterior probability for the  $B_j$ , is thus given by:

$$P(B_0B_1 \dots B_m|DI) \propto \left[ \prod_{j=0}^m P(B_j|I) \right] P(D|B_0B_1 \dots B_mI). \quad (16.14)$$

We want the Markov chain to explore the amplitude parameter space, but we don't want it to excessively waist time. All the amplitudes in an orthogonal model are estimated to be  $\pm\sqrt{\langle\sigma^2\rangle}$ , where  $\langle\sigma^2\rangle$  is the mean-square residual given the model. Consequently, if we center the prior probability for an amplitude on the expected amplitude,  $T_j$  Eq. (16.16) below, and make the standard deviation of the prior very wide, then the prior probability for the amplitudes will do little more than keep the Markov chain in the physically meaningful region of the parameter space. Here is the prior actually used for the amplitudes:

$$P(B_j|I) = (2\pi\delta^2)^{-\frac{1}{2}} \exp\left\{-\frac{(T_j - B_j)^2}{2\delta^2}\right\} \quad (16.15)$$

where the expected amplitude,  $T_j$ , is given by

$$T_j \equiv \sum_{i=1}^N d_i H_j(t_i) \quad (16.16)$$

where  $N$  is the total number of data values in the data set. The standard deviation of this prior,  $\delta$ , is 10 times larger than the expected root mean-square residual:

$$\delta = 10\sqrt{\langle\sigma^2\rangle} \quad (16.17)$$

with

$$\sqrt{\langle\sigma^2\rangle} = \sqrt{\frac{d^2 - \bar{h}^2}{N}}. \quad (16.18)$$

The quantity,  $d^2 - \bar{h}^2$ , is the total-squared residual given the polynomial. So the square root is the root mean-square residual given the polynomial order. The sufficient statistic,  $\bar{h}^2$ , is the total-squared projection of the data onto the polynomial and is defined as

$$\bar{h}^2 \equiv \sum_{k=0}^m T_k^2. \quad (16.19)$$

Having assigned the prior probabilities, we must now assign the direct probability. The direct probability,  $P(D|B_0B_1 \dots B_mI)$ , is a marginal probability and is computed from the joint probability for the data and the standard deviation of the noise

$$P(D|B_0B_1 \dots B_mI) = \int P(\sigma D|B_0B_1 \dots B_mI) d\sigma \quad (16.20)$$

which we factor as

$$P(D|B_0B_1 \dots B_m I) = \int P(\sigma|I)P(D|\sigma B_0B_1 \dots B_m I)d\sigma. \quad (16.21)$$

Assigning a Jeffreys' prior to  $P(\sigma|I)$  and a Gaussian likelihood, one obtains

$$P(B_0B_1 \dots B_m|DI) \propto \left[ \prod_{j=0}^m P(B_j|I) \right] \int \frac{1}{\sigma} (2\pi\sigma^2)^{-\frac{N}{2}} \exp \left\{ -\frac{Q}{2\sigma^2} \right\} d\sigma \quad (16.22)$$

where we have left the prior probabilities in their symbolic form. Evaluating the integral over  $\sigma$  and substituting the prior probability for the amplitudes, Eq. (16.15), into Eq. (16.22), one obtains:

$$P(B_0B_1 \dots B_m|DI) \propto \left[ \prod_{j=0}^m \exp \left\{ -\frac{(T_j - B_j)^2}{2\delta^2} \right\} \right] \left[ \frac{Q}{2} \right]^{-\frac{N}{2}} \quad (16.23)$$

where we dropped some constant terms that cancel when this probability is normalized. The function  $Q$  is defined as

$$\begin{aligned} Q &\equiv \sum_{i=1}^N \left( d_i - \sum_{j=0}^m B_j H_j(t_i) \right)^2 \\ &= N\bar{d}^2 - 2 \sum_{j=0}^m B_j T_j + \sum_{j=0}^m B_j^2. \end{aligned} \quad (16.24)$$

One interesting note about the quantity  $Q$ , it does not depend on the individual data values, rather it depends on the total squared data value, the  $N\bar{d}^2$ , and it depends on the projection of the data onto the orthogonal functions, the  $T_j$ . Both of these items can be computed at the beginning of the calculation and used throughout with no further reference to the data. Consequently, the Given Polynomial Order runs very quickly. Also, note that the function  $Q$  could have been written as a single summation rather than two. If the standard deviation for the noise is known, the posterior probability for the  $B_j$  can be factored into a product of probabilities for each amplitude separately, i.e., the amplitudes of the orthogonal polynomials may be estimated separately, they don't have to be estimated jointly. Finally, because each amplitude can be estimated separately, one can simply plot the posterior probability for each amplitude, there is no need to use a Markov chain Monte Carlo simulation to sample the joint posterior. However, joint estimation is required after marginalizing out the standard deviation for the noise. The joint estimation is done using a Markov chain Monte Carlo simulation to sample the joint posterior probability for the amplitudes, Eq. (16.23).

## 16.2 Outputs From the Given Polynomial Order Package

The Text outputs files from the Given Polynomial Order package consist of: "Bayes.prob.model," "BayesPolGiven.mcmc.values," "Bayes.params," "Console.log," "Bayes.accepted" and a "Bayes.Condensed.File." These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the

Figure 16.2: Given Polynomial Order Scatter Plot

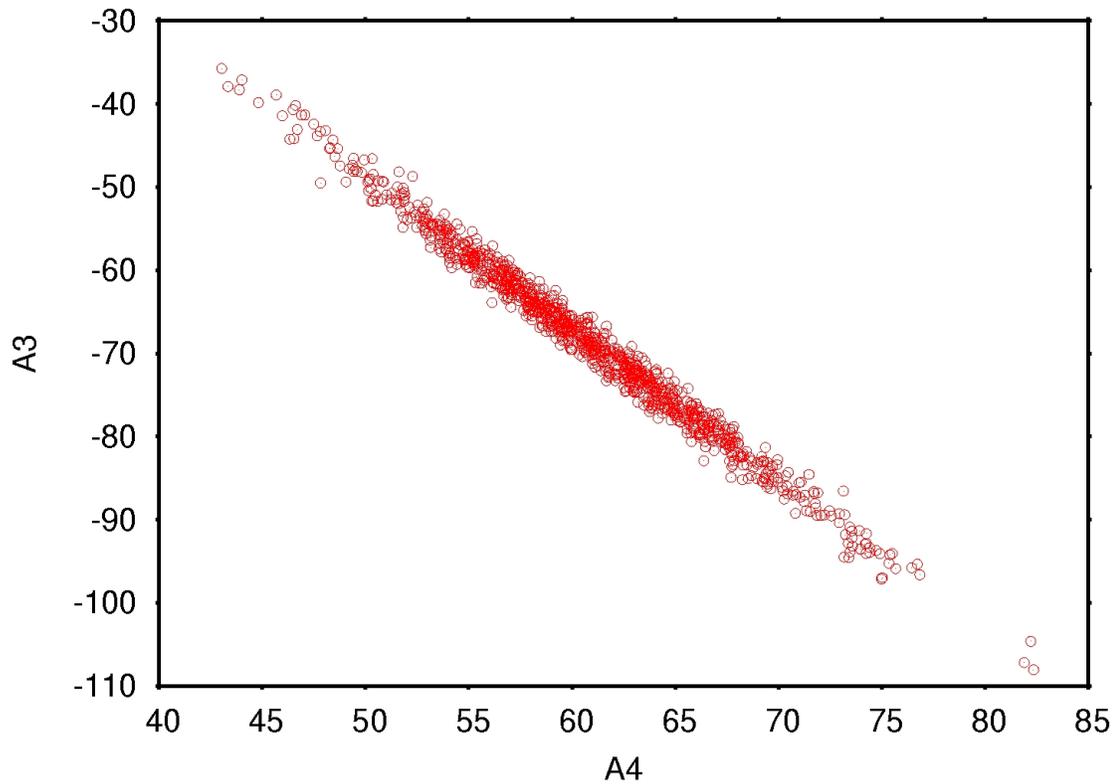


Figure 16.2 The expansion coefficients in a nonorthogonal polynomial expansion tend to be highly correlated. Plotted here is a scatter plot of the 3rd and 4th order expansion coefficients,  $A_3$  and  $A_4$ , as generated by a 6th order expansion of the 6th order polynomial test data. This test data can be downloaded using the “Files” menu.

mcmc.values report is discussed in Appendix D and the other reports are discussed in Chapter 3. Additionally, the “Plot Results Viewer” can be used to view the output probability density functions. In addition to the standard data, model and residual plots there are probability density functions for each  $A_j$  in the given model. And because estimation of the amplitudes in polynomial models tend to be highly correlated, there are covariance plots to help illustrate these correlations. These covariance plots are scatter plots. The scatter plots are generated from the samples drawn from the Markov chain Monte Carlo simulation. In a typical run, there might be 50 simulations and 30 repeats giving a total of 1500 simulations. Each of these simulations contain the estimated parameters from one Markov chain Monte Carlo simulation. A typical scatter plot just put a dot in the plot at the location of parameter 1 versus parameter 2. In this package that would correspond to plotting  $A_j$  versus  $A_k$ . In Fig. 16.2,  $A_3$  versus  $A_4$  is plotted. The data used to generate this figure are the 6th order polynomial expansion data available in our test data kit. This test data can be downloaded using the “Files” menu. The covariance plot shown in Fig. 16.2 is the one generated

form the  $A_3$  and  $A_4$  expansion coefficients in a 6th order expansion of this data. For these two parameters there is a strong correlation, when  $A_3$  increases the  $A_4$  parameter decreases to counter the effect of changing  $A_3$ . Uncorrelated parameters, by contrast, will have elliptical scatter plots with the major and minor axis aligned with coordinate system. The number of possible scatter plots is  $m(m+1)$  where  $m$  is the order of the polynomial. Because the number of scatter plots can become large very quickly, the package only outputs a representative sample of these plots.

## Chapter 17

# Unknown Polynomial Order

The Unknown Polynomial Order package fits polynomials to two column Ascii data when both the order of the polynomial and the polynomial coefficients are unknown. The interface to this package is shown in Figure 17.1. This interface differs from most others in one respect, there are no parameter ranges to enter, so use of the interface is particularly simple. To use this package, you must do the following:

**Select** the Polynomial Models package from the Package menu. When selected this menu will bring up the “Given” and “Unknown” polynomial model interface.

**Check** the “Unknown Order” box to select the Unknown Polynomial Order package. When this check box is activated the “Set Order” widget becomes inactive. This is illustrated in Fig. 17.1 where the “Unknown Order” has been checked, and, consequently, the “Set Order” widget has been grayed out.

**Load** one two column Ascii data sets. The data may be loaded using the Files menu. You can also load an arrayed Fid and then use a single cursor to mark the center of a peak and use the “Get Peak” button on the bottom right of the Fid viewer. Finally, the “Files/Load Ascii/Bayes Analyze” button can be used to load an Ascii data set from the amplitudes estimated by Bayes Analyze. When a data set is successfully loaded the data is plotted in the Ascii Data viewer. This package does not allow you to run with multiple data sets. If you attempt to do so, you will be prompted to remove all but a single file.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

Figure 17.1: Unknown Polynomial Order Package Interface

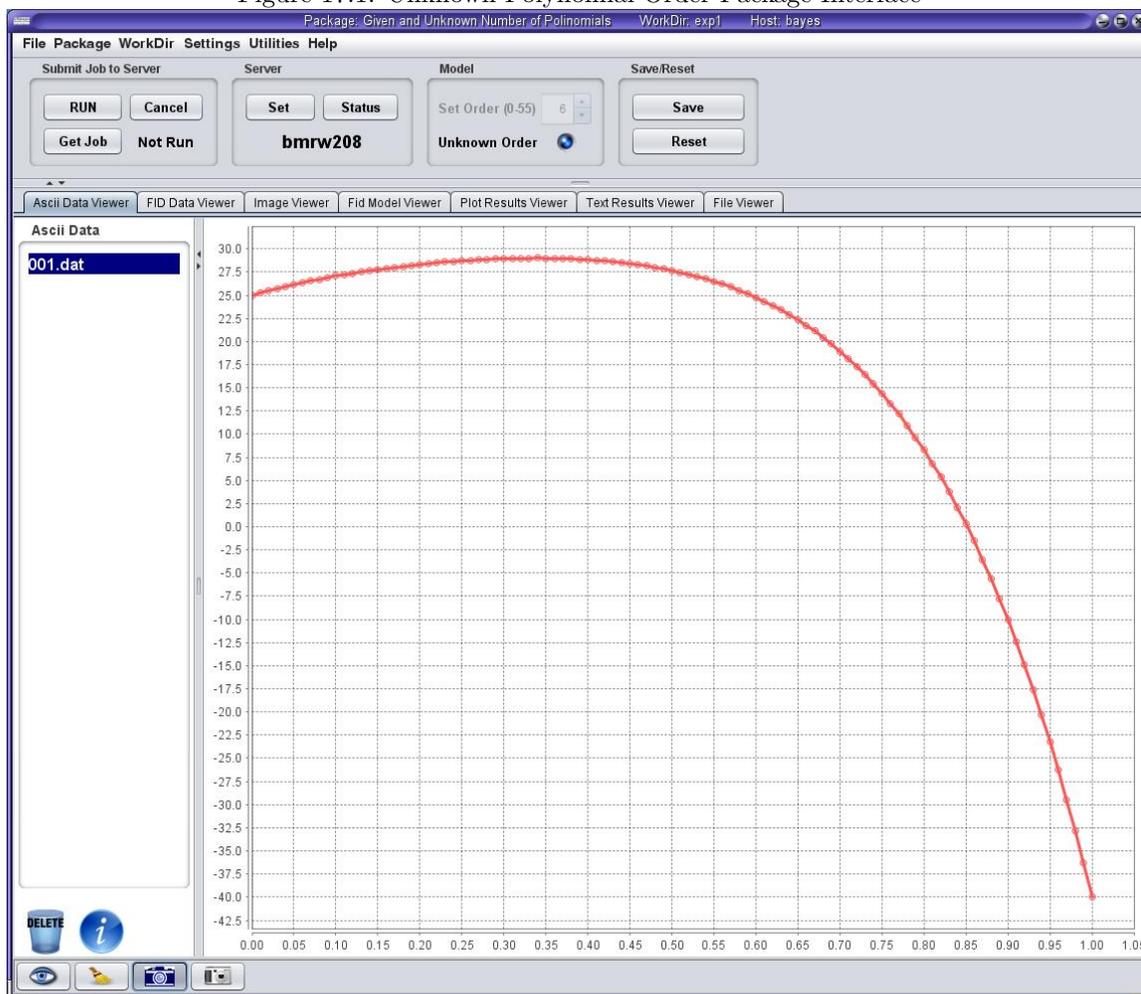


Figure 17.1: This panel is the interface to both the Given and Unknown Polynomial Order packages. Data has already been loaded. Note that in this example, the “Unknown Order” check box has been set. Consequently, the “Set Order” spinner has been deactivated. Because of the way this calculation is done very high orders are possible and numerically stable. However, the high orders, above 40, require very high signal-to-noise and even then roundoff errors degrade the accuracy to 4 or 5 decimal places.

## 17.1 Bayesian Calculations

The Unknown Polynomial Order model is just that, its a model in which a polynomial is fit to the data:

$$d_i = \sum_{j=0}^m A_j t_i^j + n_i \quad (17.1)$$

where  $A_j$  is the amplitude of the  $j$ th polynomial,  $m$  is the unknown order of the polynomial expansion, and  $n_i$  represents noise in the data. As in Chapter 16, we introduce a change of function and a change of variables and we refer the reader to that chapter for a discussion of the change of function and variables. The change of function is to orthonormal polynomials designated by  $G_j(t_i)$ , so the expansion given in Eq. (17.1) becomes:

$$d_i = \sum_{j=0}^m B_j G_j(t_i) + n_i \quad (17.2)$$

where  $B_j$  are the amplitudes in the orthonormal expansion. Note this change of function and variables is an identify, so

$$\sum_{j=0}^m A_j t_i^j = \sum_{j=0}^m B_j G_j(t_i). \quad (17.3)$$

The Bayesian calculation is implemented using Markov chain Monte Carlo with simulated annealing to draw samples from the joint posterior probability for the parameters,  $P(mB_0B_1 \dots B_m|DI)$ . From these samples we then compute the marginal posterior probabilities for the amplitudes and the polynomial order. The joint posterior probability for the parameters is computed by application of Bayes' theorem

$$P(mB_0B_1 \dots B_m|DI) \propto P(mB_0B_1 \dots B_m|I)P(D|mB_0B_1 \dots B_mI) \quad (17.4)$$

where  $P(mB_0B_1 \dots B_m|I)$  is the joint prior probability for the amplitudes and the polynomial order, and  $P(D|mB_0B_1 \dots B_mI)$  is the direct probability for the data given the parameters and the polynomial order. We factor the joint prior probability for the parameters,  $P(mB_0B_1 \dots B_m|I)$ , into a series of independent prior probabilities:

$$P(mB_0B_1 \dots B_m|I) = P(m|I) \prod_{j=0}^m P(B_j|I) \quad (17.5)$$

where  $P(m|I)$  is the prior probability for the polynomial order and  $P(B_j|I)$  is the prior probability for the  $j$ th amplitude. Substituting, Eq. (17.5) into Eq. (17.4) one obtains

$$P(mB_0B_1 \dots B_m|DI) \propto P(m|I) \left[ \prod_{j=0}^m P(B_j|I) \right] P(D|mB_0B_1 \dots B_mI) \quad (17.6)$$

as the joint posterior probability for all of the parameters, including the polynomial order.

### 17.1.1 Assigning Priors

Before we assign the likelihood, we are going to assign the prior probability for the polynomial order,  $P(m|I)$ , and the prior probability for the amplitudes, the  $P(B_j|I)$ . The prior probability for the polynomial order was assigned as a discrete Gaussian with lower bound zero, an upper bound of 50, a mean value of 5, and a standard deviation of 10:

$$P(m|I) = \begin{cases} \frac{1}{C} \exp \left\{ -\frac{(5-m)^2}{2 \times 10^2} \right\} & m \in \{0, 1, \dots, 50\} \\ 0 & \text{otherwise} \end{cases}, \quad (17.7)$$

with the normalization constant  $C$  set so that the sum over the total models is one:

$$C = \sum_{m=0}^{50} \exp \left\{ -\frac{(5-m)^2}{2 \times 10^2} \right\}. \quad (17.8)$$

Which expresses a belief that the polynomial order should be small, we think it very unlikely that the polynomial order would be as large as 50; but we think it reasonably possible for the order to be in the tens or twenties.

The prior probabilities for the amplitudes will be assigned exactly the same way they were when they did the Given Polynomial Order package, Chapter 16. That prior was given by Eq. 16.15 and we simply use that prior here:

$$P(B_j|I) = (2\pi\delta^2)^{-\frac{1}{2}} \exp \left\{ -\frac{(T_j - B_j)^2}{2\delta^2} \right\} \quad (17.9)$$

where  $\delta$  is the standard deviation of this prior probability and indicates how strongly we believe the expected amplitude is  $T_j$ . How we set  $\delta$  is explained shortly. The expected amplitude,  $T_j$ , is given by

$$T_j \equiv \sum_{i=1}^N d_i G_j(t_i). \quad (17.10)$$

In Chapter 16 when this prior was used, we knew the order of the expansion polynomial and thus could determine the mean-square residual. We could use the mean-square residual to set  $\delta$  to a value much wider than any amplitude supported by the data. So this prior probability just acted as a guide to the Markov chain Monte Carlo simulations. Here setting  $\delta$  is harder because we don't know which model to use. However, we still want to set  $\delta$  to a value that will guide the Markov chain Monte Carlo simulations but not make  $\delta$  so large that the simulations never converge. In Chapter 16 we noted that because these amplitudes appear in the model in a linear fashion, we could solve the problem analytically, we don't have to use Markov chain Monte Carlo at all. The only reason for using a Markov chain is for consistency with the other packages in our Bayesian Analysis software. However, there is nothing to stop us from computing  $P(m|DI)$  analytically and using that to set  $\delta$ . Without going into the details of this calculation, the posterior probability for polynomial of order  $m$  is given by:

$$P(m|DI) = P(m|I) \Gamma\left(\frac{m}{2}\right) \Gamma\left(\frac{N-m}{2}\right) \left[\overline{h_m^2}\right]^{-\frac{m}{2}} \left[\frac{\overline{d^2} - \overline{h_m^2}}{2}\right]^{-\frac{N-m}{2}} \quad (17.11)$$

where  $P(m|I)$  is given by Eq. (17.7) and the prior probability for the amplitudes was assigned as a normalized unbounded Gaussian with mean zero and standard deviation  $\gamma$ ; which was marginalized out of the problem using a series of approximations given in [2]. The quantity,  $\overline{d^2} - \overline{h_m^2}$ , is the total-squared residual given a polynomial of order  $m$ . The sufficient statistic,  $\overline{h_m^2}$ , is the total-squared projection of the data onto the given polynomial model and is defined as

$$\overline{h_m^2} \equiv \sum_{k=0}^m T_k^2. \quad (17.12)$$

The expected standard deviation of the noise independent of the model order is given by:

$$\sqrt{\langle \sigma^2 \rangle} = \sum_{m=0}^{Max} P(m|DI) \sqrt{\frac{\overline{d^2} - \overline{h_m^2}}{N}}. \quad (17.13)$$

Finally,  $\delta$  was set to

$$\delta = 10\sqrt{\langle \sigma^2 \rangle}. \quad (17.14)$$

While rather complicated, this calculation was used for two reasons: I needed an estimate of the standard deviation of the noise, which this gives by simple straightforward calculation; and I needed a way to determine where the maximum of the posterior probability for the polynomial order was. I needed this maximum to determine where to center the distribution of simulations that is printed out while this program is running. The problem is illustrated in Fig. 17.2. The maximum order of the polynomial is 50, but there is only room to print out 10 of these probabilities. So the output window must be shifted to cover the maximum posterior probability for the polynomial order. To do that, I needed to know where the maximum was. This calculation solved both of these problems at one time and it did so using Bayesian probability theory. See my book, [2], for more on this calculation and where each of these terms comes from.

### 17.1.2 Assigning The Joint Posterior Probability

Having assigned the prior probabilities, we can now proceed with assigning the joint posterior probability for the parameters, Eq. (17.6). First, however, we assign the direct probability for the data. The direct probability,  $P(D|mB_0B_1 \dots B_mI)$ , is a marginal probability and is computed from the joint probability for the data and the standard deviation of the noise

$$P(D|mB_0B_1 \dots B_mI) = \int P(\sigma D|mB_0B_1 \dots B_mI) d\sigma \quad (17.15)$$

which we factor as

$$P(D|mB_0B_1 \dots B_mI) = \int P(\sigma|I) P(D|\sigma mB_0B_1 \dots B_mI) d\sigma. \quad (17.16)$$

Assigning a Jeffreys' prior to  $P(\sigma|I)$  and a Gaussian likelihood, one obtains

$$P(mB_0B_1 \dots B_m|DI) \propto P(m|I) \left[ \prod_{j=0}^M P(B_j|I) \right] \int \frac{1}{\sigma} (2\pi\sigma^2)^{-\frac{N}{2}} \exp\left\{-\frac{Q}{2\sigma^2}\right\} d\sigma \quad (17.17)$$

Figure 17.2: The Distribution of Models On The Console Log

Phase	Annl Param	<Likelihood>	<StdDevLike>	3	4	5	6	7	8	9	10	11	12
Annealing	0.000	-2.8665E+01	-1.4089E+02	4	5	2	0	4	4	3	4	2	0
	2	0.004	-3.1841E+01	-5.2177E+01	5	5	7	4	3	5	2	4	0
	3	0.007	-3.1615E+01	9.4579E+00	6	5	14	4	5	2	3	1	1
	4	0.012	-3.2178E+01	5.4344E+01	4	7	15	8	4	4	0	1	1
	5	0.017	-3.4670E+01	1.0615E+02	2	6	14	13	7	3	0	1	1
	6	0.025	-3.6671E+01	1.3447E+02	0	4	19	8	7	6	3	1	0
	7	0.039	-3.6100E+01	1.5631E+02	0	0	21	13	7	2	3	2	0
	8	0.065	-3.4743E+01	1.7613E+02	0	0	16	18	7	6	1	0	0
	9	0.088	-3.3993E+01	1.9458E+02	0	0	11	22	12	3	0	0	0
	10	0.109	-3.4516E+01	2.0296E+02	0	0	8	24	10	4	2	0	0
	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.
Phase	Frac	<Likelihood>	<StdDevLike>	3	4	5	6	7	8	9	10	11	12
Sampling	0.500	-3.1813E+01	2.6462E+02	0	0	0	48	0	0	0	0	0	0
	12	0.550	-3.1991E+01	2.6438E+02	0	0	0	46	2	0	0	0	0
	13	0.600	-3.1907E+01	2.6411E+02	0	0	0	47	1	0	0	0	0
	14	0.650	-3.1989E+01	2.6459E+02	0	0	0	46	2	0	0	0	0
	15	0.700	-3.1817E+01	2.6430E+02	0	0	0	48	0	0	0	0	0
	16	0.750	-3.1904E+01	2.6432E+02	0	0	0	47	1	0	0	0	0
	17	0.800	-3.1994E+01	2.6414E+02	0	0	0	46	2	0	0	0	0
	18	0.850	-3.1992E+01	2.6429E+02	0	0	0	46	2	0	0	0	0
	19	0.900	-3.1909E+01	2.6386E+02	0	0	0	47	1	0	0	0	0
	20	0.950	-3.1903E+01	2.6447E+02	0	0	0	47	1	0	0	0	0
Phase	Frac	<Likelihood>	<StdDevLike>	3	4	5	6	7	8	9	10	11	12
Sampling	1.000	-3.1898E+01	2.6484E+02	0	0	0	47	1	0	0	0	0	0

Figure 17.2: While the unknown Polynomial Order package is running, it prints a listing that shows the distribution of the model indicator as a function of the annealing parameter. The 10 columns to the right are the number of simulations in model 0, 1, etc. As the annealing parameter increases these should cluster into one or two columns and the distribution of these simulations is the posterior probability for the polynomial order. In this example the data were a 6th order polynomial. Notice that as soon as the annealing parameter begins to increase the simulations quickly move to high order models and eventually they usually all end up in the 6th order polynomial model.

where we have left the prior probabilities in their symbolic form. Evaluating the integral over  $\sigma$  one obtains

$$P(mB_0B_1 \dots B_m|DI) \propto P(m|I) \left[ \prod_{j=0}^m P(B_j|I) \right] \left[ \frac{Q}{2} \right]^{-\frac{N}{2}} \quad (17.18)$$

as the posterior probability for the parameters including the polynomial order, where

$$\begin{aligned} Q &\equiv \sum_{i=1}^N \left( d_i - \sum_{j=0}^m B_j G_j(t_i) \right)^2 \\ &= N\bar{d}^2 - 2\sum_{j=0}^m B_j T_j + \sum_{j=0}^m B_j^2. \end{aligned} \quad (17.19)$$

In evaluating the integral over  $\sigma$  there were a number of constants that were dropped. In model selection problems that is usually a bad thing to do and will, almost always, cause problem. Here we could to it because each polynomial model contains exactly the same constants and so they always cancel. Finally, substituting the prior probability for the polynomial order, Eq. (17.7), the prior probability for the amplitudes, Eq. (17.9) into Eq. (17.18), the joint posterior probability for the parameters is given by:

$$P(mB_0B_1 \dots B_m|DI) \propto \exp \left\{ -\frac{(5-m)^2}{2 \times 10^2} \right\} \left[ \prod_{j=0}^m (2\pi\delta^2)^{-\frac{1}{2}} \exp \left\{ -\frac{(T_j - B_j)^2}{2\delta^2} \right\} \right] \left[ \frac{Q}{2} \right]^{-\frac{N}{2}}. \quad (17.20)$$

It is this joint probability density function that is targeted by the Markov chain Monte Carlo simulations.

If one were to compute the posterior probability for the polynomial order using Eq. (17.20) and compare it to that given by Eq. (17.11). You would find you get different results. That's because in computing these two sets of equations we used slightly different prior probabilities for the amplitudes. While these prior probabilities were not much different, they are nonetheless different and that difference would manifest itself as a slight difference in the final calculations. As far as which is right, they are both correct given the two sets of prior information. Regardless, they won't differ by much and almost certainly, given the discrete nature of the posterior probability, won't differ at all after you normalize the final posterior probability.

## 17.2 Outputs From the Unknown Polynomial Order Package

The Text outputs from the Unknown Polynomial Order package consist of: “Bayes.prob.model,” “BayesPolUnknown.mcmc.values,” “Bayes.params,” “Console.log” see Fig. 17.2, “Bayes.accepted” and a condensed output file “Bayes.Condensed.File.” These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the mcmc.values report is discussed in Appendix D and the other reports are discussed in Chapter 3.

The main new output from the Unknown Polynomial Order package is a plot of the posterior probability for the polynomial order, Fig. 17.3. The data used to generate this figure are the

Figure 17.3: The Posterior Probability For The Polynomial Order

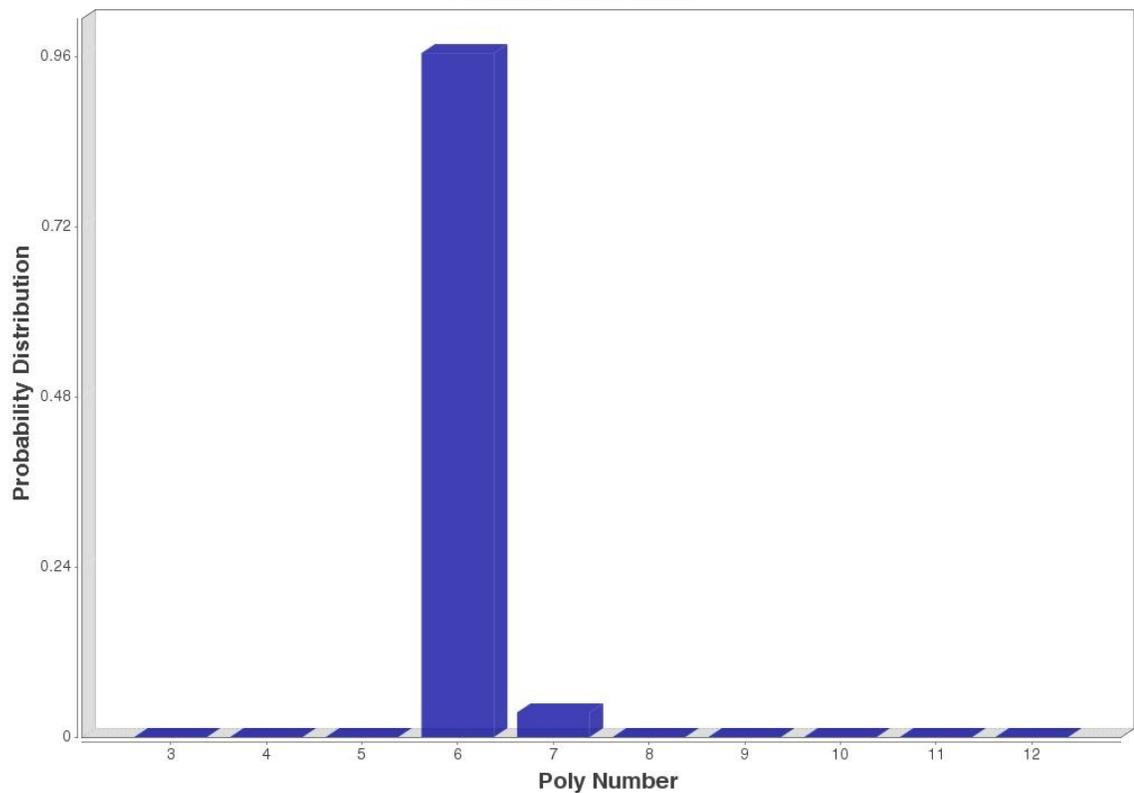


Figure 17.3: The main new output in the Unknown Polynomial Order Package is the posterior probability for the polynomial order, here called polynomial number. This output figure contains 10 probabilities centered around the peak in the posterior probability.

Polynomials.6th.order.dat distributed in the Bayes.test.data. This figure consists of a bar chart of the posterior probability. However, there are only 10 output probabilities, while the posterior probability contains a maximum of 51 probabilities (orders 0 through 50). So these probabilities are centered around the location of the maximum, see Section 17.1.1 for a discussion of how this maximum is located. Usually all of the probability is concentrated in one or two probabilities around the maximum with an abrupt lower bound and a more gentle drop off as you go to higher orders. Consequently, after locating the maximum of the posterior probability for the model order, the lowest output probability is set 3 orders below the maximum and the maximum output probability is 6 orders above the maximum probability order. All this is illustrated in Fig. 17.3, there the maximum is 6, the lowest output order is  $6 - 3 = 3$ , and the highest is  $6 + 6 = 12$ .



## Chapter 18

# Errors In Variables

The “Errors in Variables” package fits polynomials to data when you have errors in both the abscissa  $X$  and the data value  $Y$ . The interface to this package is shown in Fig. 18.1. This interface is used to configure the Errors In Variables Package by setting both the polynomial order and by indicating what errors are known or given. Additionally, depending on the settings of the “Given Errors In” widget, the interface will load two, three or four column Ascii data. To use this package, you must do the following:

**Select** the Errors In Variables Package from the Package menu.

**Using** “Given Errors In” pull down menu select the type of Errors In Variables problem you wish to solve. Your choices are:

1. “Not Given” solves the Errors In Variables problem when the errors in both  $X$  and  $Y$  are not given. This option requires two column Ascii data, see Section 18.2 for a description of these files.
2. “ $X$  and  $Y$ ” solves the Errors In Variables problem when the errors in  $X$  and  $Y$  are given. This option requires four column Ascii data.
3. “ $X$  Only” solves the Errors In Variables problem when the errors in  $X$  are given. This option requires three column Ascii data.
4. “ $Y$  Only” solves the Errors In Variables problem when the errors in  $Y$  are given. This option requires three column Ascii data.

**Load** one data set appropriate to the selected problem. Only “Not Given” uses standard two column Ascii data. Consequently, when this option is selected both Bayes Analyze and a Peak Pick can server as input. All of the other options require input of either three or four column Ascii files which can only be loaded using the “Files/Load Ascii/Files” menu. When you have successfully loaded a data set it will be plotted in the Ascii Data Viewer. However, the plot is a simple XY plot and no attempt is made by the interface to show the error bars in either  $X$  or  $Y$ .

**Using** the “Order” pull down menu, set the order of the polynomial to fit to the data.

Figure 18.1: The Errors In Variables Package Interface

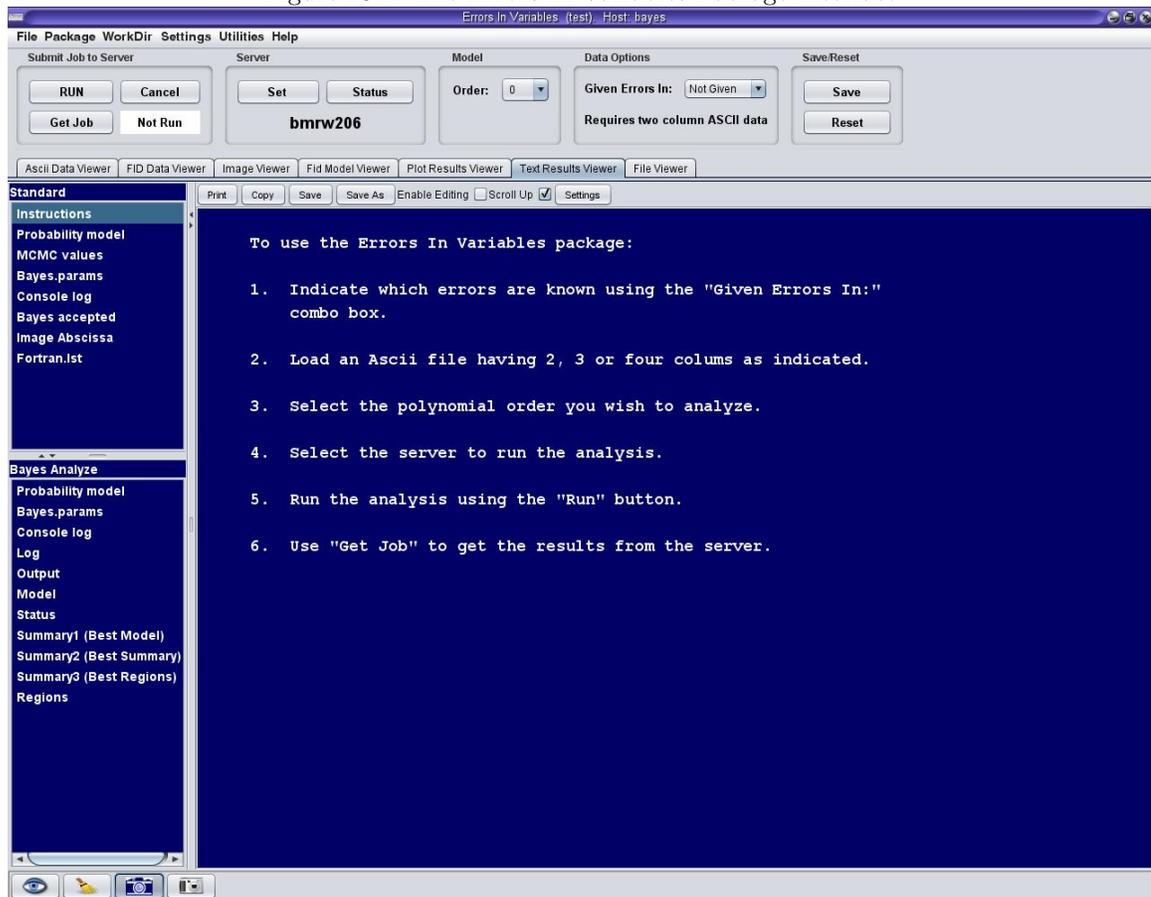


Figure 18.1: The Errors In Variables Package solves the problem of fitting polynomials when there are errors in both  $X$  and  $Y$  where  $X$  is an abscissa and  $Y$  is some function of  $X$ . Here this function is assumed to be a polynomial of a given order.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

## 18.1 The Bayesian Calculation

The problem we are considering is one in which the data consists of measured pairs of numbers,  $(\hat{x}_i, \hat{y}_i)$ , where both the measured abscissa data value,  $\hat{x}_i$ , and the measured ordinate,  $\hat{y}_i$ , contain errors, i.e., noise, and we wish to fit a polynomial to these measured data pairs. If we designate the “true” but unknown values of  $(\hat{x}_i, \hat{y}_i)$  as  $(x_i, y_i)$  then the polynomial one would normally fit is given by

$$y = F(x) = \sum_{j=0}^m B_j H_j(x) \tag{18.1}$$

where the polynomial,  $F(x)$ , is evaluated at the true value of  $x$  giving a true value of  $y$ ,  $B_j$  is the amplitude of the polynomial  $H_j(x)$ . These polynomials are the same polynomials discussed in Chapter 16 and are the Gram-Schmidt polynomials generated from  $x^j$ . Unfortunately, neither the  $y$  values nor the  $x$  values are known. An abscissa data value,  $\hat{x}_i$ , is related to the unknown true abscissa,  $x_i$ , by

$$\hat{x}_i = x_i + e_i \tag{18.2}$$

where  $e_i$  is the error in the measured abscissa. Similarly, a measured ordinate value,  $\hat{y}_i$ , is related to the unknown true ordinate,  $y_i$ , by

$$\hat{y}_i = y_i + n_i \tag{18.3}$$

where  $n_i$  is the error in the measured ordinate.

In the following calculation we will assume that the errors in the abscissa and ordinate are both unknown. We do this for the simple reason that it is the harder, i.e., more interesting calculation. However, in some cases one or both of these errors are actually known and the computer program that implements this package implements four different cases: 1) errors in both abscissa and ordinate known, 2) errors in the abscissa know but the ordinate error is unknown, etc.

If we Taylor expand the polynomial,  $F(x)$ , around  $\hat{x}_i$ , then

$$F(x_i) \approx F(\hat{x}_i) + F'(\hat{x}_i)(\hat{x}_i - x_i) \tag{18.4}$$

where

$$F'(\hat{x}_i) = \frac{dF(\hat{x}_i)}{d\hat{x}_i}. \tag{18.5}$$

This first order Taylor expansion will gives a reasonable approximation to  $F(x)$  if the errors in the unknown abscissa are not large, i.e., if the polynomial is approximately linear over the likely error in

the abscissa. We make the Taylor expansion because, now the unknown true values of  $x_i$  appear in the model in a linear fashion and consequently we will be able to remove them by marginalization.

The probabilities we want to compute are the marginal posterior probabilities for the amplitudes of the polynomials and standard deviations of the noise prior probabilities. To calculate these, the numerical simulations must target the joint posterior probability for the amplitudes,  $\{B_0, \dots, B_m\}$ , and the noise standard deviations,  $\sigma_x$  and  $\sigma_y$  given the abscissa and ordinate data. This joint posterior probability is represented symbolically by  $P(B_0 \cdots B_m \sigma_x \sigma_y | \hat{x} \hat{y} I)$ , where  $\hat{x}$  and  $\hat{y}$  represent the abscissa and ordinate data.

The joint posterior probability for the amplitudes and the noise standard deviations targeted by the Markov chain Monte Carlo simulation,  $P(B_0 \cdots B_m \sigma_x \sigma_y | \hat{x} \hat{y} I)$ , is a marginal probability:

$$P(B_0 \cdots B_m \sigma_x \sigma_y | \hat{x} \hat{y} I) = \int P(B_0 \cdots B_m \sigma_x \sigma_y \{x\} | \hat{x} \hat{y} I) d\{x\} \quad (18.6)$$

where the integrals are over all of the unknown abscissa values. To compute this posterior probability, one factors the right-hand side of this equation using Bayes' theorem and the product rule to obtain

$$\begin{aligned} P(B_0 \cdots B_m \sigma_x \sigma_y \{x\} | \hat{x} \hat{y} I) &\propto P(\sigma_x | I) P(\sigma_y | I) \prod_{j=0}^m P(B_j | I) \\ &\times \prod_{i=1}^N P(x_i | I) \\ &\times \prod_{i=1}^N P(\hat{x}_i | x_i \sigma_x I) \\ &\times \prod_{i=1}^N P(\hat{y}_i | x_i B_0 \cdots B_m \sigma_y I). \end{aligned} \quad (18.7)$$

The prior probability for the standard deviation of the noise for the abscissa data,  $P(\sigma_x | I)$ , will be assigned as a bound Jeffreys' prior

$$P(\sigma_x | I) = \begin{cases} \frac{1}{R_x \sigma_x} & \text{if } \sigma_{xL} \leq \sigma_x \leq \sigma_{xH} \\ 0 & \text{otherwise} \end{cases} \quad (18.8)$$

where  $\sigma_{xL}$  and  $\sigma_{xH}$  are a lower and upper bound on  $\sigma_x$ . The normalization constant  $R_x$  is given by

$$R_x = \int_{\sigma_{xL}}^{\sigma_{xH}} \frac{d\sigma_x}{\sigma_x} = \log(\sigma_{xH}/\sigma_{xL}). \quad (18.9)$$

The bounds,  $\sigma_{xL}$  and  $\sigma_{xH}$ , are set rather pragmatically within the program that implements this package. The program computes the mean-square deviation,  $\langle \bar{x}^2 - \bar{x}^2 \rangle$ , using the  $\hat{x}$  data, and then sets the lower and upper bounds

$$\sigma_{xL} = \frac{\langle \bar{x}^2 - \bar{x}^2 \rangle}{10} \quad \text{and} \quad \sigma_{xH} = 10 \langle \bar{x}^2 - \bar{x}^2 \rangle, \quad (18.10)$$

which restricts  $\sigma_x$  to a two order of magnitude variation. Similarly,  $P(\sigma_y | I)$ , will be assigned

$$P(\sigma_y | I) = \begin{cases} \frac{1}{R_y \sigma_y} & \text{if } \sigma_{yL} \leq \sigma_y \leq \sigma_{yH} \\ 0 & \text{otherwise} \end{cases} \quad (18.11)$$

where the normalization constant,  $R_y$ , and the lower and upper bounds are computed analogously to corresponding abscissa values. The prior probability for the amplitudes,  $P(B_j|I)$ , is assigned a bounded zero mean Gaussian. The bounds are set at plus and minus ten times the largest projection of the model onto the data:

$$P(B_j|I) = \begin{cases} (2\pi\sigma_B^2)^{-1/2} \exp\left\{-\frac{B_j^2}{2\sigma_B^2}\right\} & \text{if } B_L \leq B_j \leq B_H \\ 0 & \text{otherwise} \end{cases} \quad (18.12)$$

with  $B_L = -B_H$  and  $B_H = \max(\text{abs}[\sum_{i=1}^N \hat{y}_i H_j(\hat{x}_i)])$ . The standard deviation of this prior is  $\sigma_B = B_H/3$ . So the interval,  $B_H - B_L$ , represents a 6 standard deviation interval and the prior serves as little more than a way to keep the amplitudes from wandering into an unphysical region of the parameter space.

In this problem the unknown true values, the  $x_i$ , are parameters and one must assign a prior probability to all such parameters. The prior probability for the  $x_i$ , the  $P(x_i|I)$ , are assigned as unbounded Gaussians having mean equal to  $\tilde{x}_i$  and standard deviation,  $\sigma_p$ :

$$P(x_i|I) = (2\pi\sigma_p^2)^{-1/2} \exp\left\{-\frac{(\tilde{x}_i - x_i)^2}{2\sigma_p^2}\right\} \quad (18.13)$$

where  $\tilde{x}_i$  is the sampling point we thought we were measuring, and we set  $\sigma_p$  equal to the average  $x$  interval. We did this because it greatly simplified the formulas that must be programmed and so makes for a faster program, without changing the results to within the error bars.

The likelihood for a measured  $\hat{x}_i$  data value,  $P(\hat{x}_i|x_i\sigma_x I)$ , was assigned a Gaussian

$$P(\hat{x}_i|x_i\sigma_x I) = (2\pi\sigma_x^2)^{-1/2} \exp\left\{-\frac{(\hat{x}_i - x_i)^2}{2\sigma_x^2}\right\} \quad (18.14)$$

and the likelihood for a measured  $\hat{y}_i$  data value,  $P(\hat{y}_i|\{A\}\sigma_y I)$ , was assigned a Gaussian likelihood of the form:

$$P(\hat{y}_i|B_0 \cdots B_m \sigma_y I) = (2\pi\sigma_y^2)^{-1/2} \exp\left\{-\frac{(\hat{y}_i - F(\hat{x}_i) - F'(\hat{x}_i)[\hat{x}_i - x_i])^2}{2\sigma_y^2}\right\}. \quad (18.15)$$

If we now accumulate all of the priors, Eqs. (18.8,18.11,18.13), and likelihoods, Eqs.(18.14,18.15),

and substitute them into Eq. (18.6) one obtains

$$\begin{aligned}
P(B_0 \cdots B_m \sigma_x \sigma_y | \hat{x} \hat{y} I) &\propto \int_{-\infty}^{+\infty} dx_1 \cdots dx_N \frac{1}{R_x \sigma_x} \frac{1}{R_y \sigma_y} \\
&\times (2\pi \sigma_B^2)^{-\frac{m+1}{2}} \exp \left\{ -\sum_{j=0}^m \frac{B_j^2}{2\sigma_B^2} \right\} \\
&\times (2\pi \sigma_p^2)^{-\frac{N}{2}} \exp \left\{ -\sum_{i=1}^N \frac{(\tilde{x}_i - x_i)^2}{2\sigma_p^2} \right\} \\
&\times (2\pi \sigma_x^2)^{-\frac{N}{2}} \exp \left\{ -\sum_{i=1}^N \frac{(\hat{x}_i - x_i)^2}{2\sigma_x^2} \right\} \\
&\times (2\pi \sigma_y^2)^{-\frac{N}{2}} \exp \left\{ -\sum_{i=1}^N \frac{(\hat{y}_i - F(\hat{x}_i) - F'(\hat{x}_i)[\hat{x}_i - x_i])^2}{2\sigma_y^2} \right\}
\end{aligned} \tag{18.16}$$

as the posterior probability for the amplitudes and noise standard deviations. Evaluating the  $N$  integrals over the  $x_i$ , one obtains

$$P(B_0 \cdots B_m \sigma_x \sigma_y | \hat{x} \hat{y}) \propto \frac{1}{\sigma_x \sigma_y} \exp \left\{ -\sum_{j=0}^m \frac{B_j^2}{2\sigma_B^2} \right\} \prod_{i=1}^N \left( \frac{\sigma_x \sigma_y}{\sigma_i} \right) \exp \left\{ -\frac{Q_i}{2\sigma_i^2} \right\} \tag{18.17}$$

where we have dropped some constants that cancel when this probability density function is normalized. The function,  $Q_i$ , is given by

$$Q_i \equiv [\hat{y}_i - F(\hat{x}_i)]^2 [\sigma_p^2 + \sigma_x^2] - 2F'(\hat{x}_i) \sigma_x^2 [\hat{x}_i - \tilde{x}_i] [\hat{y}_i - F(\hat{x}_i)] + [F'(\hat{x}_i)^2 \sigma_x^2 + \sigma_y^2] [\hat{x}_i - \tilde{x}_i]^2 \tag{18.18}$$

where

$$\sigma_i \equiv \sqrt{\sigma_x^2 \sigma_y^2 + \sigma_p^2 \sigma_y^2 + F'(\hat{x}_i) \sigma_p^2 \sigma_x^2}. \tag{18.19}$$

In the special case that the Errors In Variables package implements,  $\tilde{x}_i = \hat{x}_i$ ,  $Q_i$  simplifies and one obtains

$$P(B_0 \cdots B_m \sigma_x \sigma_y | \hat{x} \hat{y}) \propto \frac{1}{\sigma_x \sigma_y} \exp \left\{ -\sum_{j=0}^m \frac{B_j^2}{2\sigma_B^2} \right\} \prod_{i=1}^N \left( \frac{\sigma_x \sigma_y}{\sigma_i} \right) \exp \left\{ -\frac{[\hat{y}_i - F(\hat{x}_i)]^2 [\sigma_p^2 + \sigma_x^2]}{2\sigma_i^2} \right\}. \tag{18.20}$$

Computationally, this special case is simpler to calculate, so the program runs faster without given up the ability to estimate both  $\sigma_x$  and  $\sigma_y$  and it is this probability density function that is targeted by the Markov chain Monte Carlo simulation.

## 18.2 Outputs From The Errors In Variables Package

The Text outputs files from the Errors In Variables packages consist of: “Bayes.prob.model,” “BayesErrInVarsGiven.mcmc.values,” “Bayes.params,” “Console.log,” “Bayes.accepted” and a “Bayes.Condensed.Fil

These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the `mcmc.values` report is discussed in Appendix D and the other reports are discussed in Chapter 3. Additionally, the “Plot Results Viewer” can be used to view the output probability density functions. In addition to the standard data, model and residual plots there are probability density functions for the decay rate constants, decay times, the amplitudes for each data set for each exponential and finally there are probability density functions for the standard deviation of the noise in each data set.

The only thing the least bit unusual about this package is the Ascii data that is required. In most Ascii packages the data are two columns, an abscissa and an ordinate. However, here there are four different file formats:

1. When the errors in both the abscissa and ordinate are unknown, two column Ascii data is required. Column one is the abscissa and column two is the ordinate. These data may be generated and or loaded using the files menu.
2. When the errors are known in the abscissa but not in the ordinate, three column Ascii data is required. Column one is the abscissa, column two is the ordinate, and column three is the error in the abscissa. This input can only be loaded using the “Files/Load Ascii/File” menu.
3. When the errors are known in the ordinate but not in the abscissa, three column Ascii data is required. Column one is the abscissa, column two is the ordinate, and column three is the error in the ordinate. This input can only be loaded using the “Files/Load Ascii/File” menu.
4. When the errors are known in both the abscissa and the ordinate, four column Ascii data is required. Column one is the abscissa, column two is the ordinate, column three is the error in the abscissa and column four is the error in the ordinate. This input can only be loaded using the “Files/Load Ascii/File” menu.

There are four test data sets located in the “`Bayes.test.data/ErrorsInVariables`” directory that may be used for testing. These four data sets are named: `ErrInVar_given_x.dat`, `ErrInVar_given_xy.dat`, `ErrInVar_given_y.dat`, and `ErrInVar_not_given.dat` respectively. An example of the MCMC values report generated by the Errors In Variables package is shown in Fig. 18.2. This report was generated using the “`ErrInVar_given_xy.dat`” data. The top section of the report contains some configuration information followed by information about the priors. This is followed by some averages over the various probabilities including the posterior probability for the model. This is followed by the parameters that maximized the joint marginal posterior probability for the parameters. Finally, the mean and standard deviation estimates of the amplitudes of the polynomials are given. For more on the general layout of the MCMC value file, see Appendix D.

In addition to the MCMC values file there are the standard Data, Model and Residual plots which can be viewed using the Plot Results Viewer. There are posterior probability density functions for the uncertainty in the abscissa data values, this plot is named “Sigma X” and there is a posterior probability density for the uncertainty in the ordinate, named “Sigma Y.” Additionally, there is one output probability density function for each amplitude in the polynomial being analyzed. So if you are analyzing a 6th order polynomial, there are seven output probability density functions. Finally, there are the output plots that come at the end of the plot list. These include the expected logarithm of the likelihood as a function of the annealing parameter, the scatter plots and the logarithm of the posterior probability for each simulation as a function of repeat number.

Figure 18.2: The MCMC Values File Produced By The Errors In Variables Package

The Parameter File Listing for the Errors in Variables Package

```

! BayesErrInVarsGiven Package
! Created 10-Feb-2012 10:11:27 by larry
!
      Output Dir = BayesOtherAnalysis
Number Of Abscissa = 1
Number Of Columns = 1
Number Of Sets = 1
      File Name = BayesOtherAnalysis/ErrInVar_given_xy.dat
MCMC Simulations = 48
MCMC Repeats = 21
Minimum Annealing Steps = 21
Histogram Type = Binned
Outlier Detection = Disabled
Number Of Priors = 0
Package Parameters = 2
Total Mcmc Samples = 1008
Kill Count = 4

```

MCMC Values Report for the Errors In Variables package

Param Desc	Priors Used In This Run			Sigma	Norm
	Low	Mean	High		
Coef 0.1	-1.504E+03	0.000E+00	1.504E+03	4.513E+02	-3.626E+00
Coef 1.1	-1.504E+03	0.000E+00	1.504E+03	4.513E+02	-3.626E+00
X 1.1	-5.445E-02	-4.785E-03	4.488E-02	9.933E-03	-3.222E+00
X 101.1	9.488E-01	9.985E-01	1.048E+00	9.933E-03	-3.222E+00

	Avg.	Sd.
The Average Log Posterior Probability:	194.0262	122.93507
The Average Log Prior Params:	-9.0257	0.00079
The Average Log Likelihood:	4.06109543E+02	1.37923E+00
The Log Posterior Probability For The Model:	3.74864004E+02	

The parameters that maximized the posterior probability are:

Parameter Description	Parameter
Std Dev in X	4.65331682E-04
Std Dev in Y	1.10193510E+00
Coef 0.1	1.01151226E+01
Coef 1.1	9.68326780E+00

The expected parameter values (mean value of the probability distributions):

Parameter Description	Mean Value	Std. Dev.	Peak Value
Sigma X	1.09472E-03	5.34846E-04	4.65332E-04
Sigma Y	1.09118E+00	7.44062E-02	1.10194E+00
Coef 1.0	1.01672E+01	2.01540E-01	1.01151E+01
Coef 1.1	9.59826E+00	3.49706E-01	9.68327E+00

Figure 18.2: This is the Errors In Variables mcmc.values file. It is the primary printed output from the Errors In Variables package. This report was generated using test data found in Bayes.test.data, the ErrorsInVariables subdirectory, the data file was ErrInVar\_given\_xy.dat. The top section of the report contains some configuration information followed by information about the priors. This is followed by some averages over the various probabilities including the posterior probability for the model. This is followed by the parameters that maximized the joint marginal posterior probability. Finally, the mean and standard deviation estimates of the amplitudes of the polynomials are given.

## Chapter 19

# Behrens-Fisher

The Behrens-Fisher problem is the classical medical testing problem. In this problem one has two data sets. Each data set is a repeated measurement of the same quantity, so the signal is presumed constant in both measurements. However, somewhere between measuring the first and second data set, something is changed. That change could be a change in an experimental parameter, a change in the health of the animal, or any thing else that could introduce a change into the measured data. When the package is run, the program computes the posterior probability that something has changed. The interface to the Behrens-Fisher package is shown in Fig. 19. Note that this interface does not have any package specific widgets. However, it does have two prior probabilities that must be reviewed to make sure the interface set them to reasonable values. To use this package, you must do the following:

**Select** the Behrens-Fisher package from the Package menu.

**Load** two Ascii data sets using the Files menu. When a data set is successfully loaded the data is plotted in the Ascii Data viewer.

**Review** the prior probabilities for the mean and standard deviation using the Prior Viewer.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

### 19.1 Bayesian Calculation

In the classical medical testing problem, the question one would like to answer is, are the data sets the same or are they different? If their different, how? Did the means change? Did the standard

Figure 19.1: The Behrens-Fisher Interface

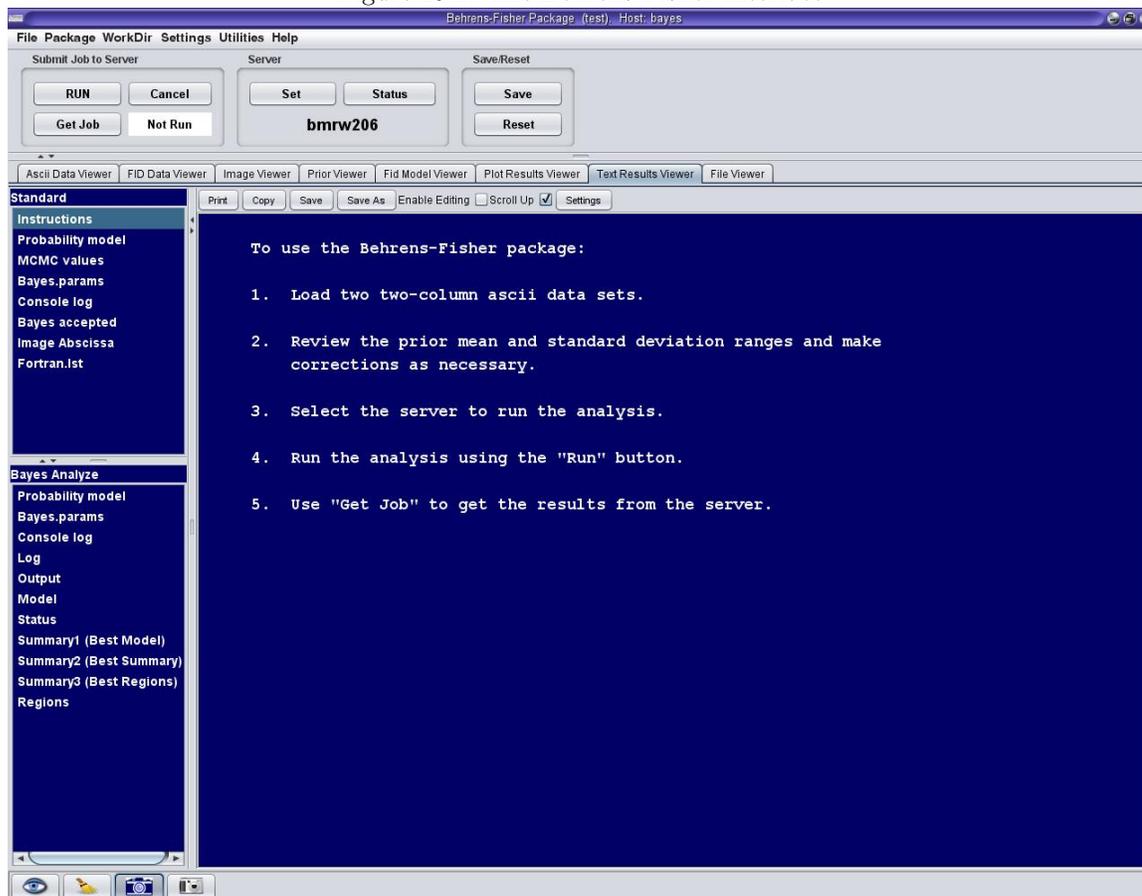


Figure 19.1: When the Behrens-Fisher package is selected, this is the displayed interface. The Behrens-Fisher package does not have any package specific widgets. However, it does have two prior probabilities that should be examined before running the analysis. These priors are for the mean and standard deviation of the data sets. The prior viewer can be used to set these prior probabilities.

Figure 19.2: Behrens-Fisher Hypotheses Tested

Hypotheses	Abbreviation
The means and the variances are the same	$SmSv$
The means are the same and the variances differ	$SmDv$
The means differ and the variances are the same	$DmSv$
The means and variances differ	$DmDv$
The means are the same	$Sm$
The means are not the same	$Dm$
The variances are the same	$Sv$
The variances are not the same	$Dv$
The data sets differ	$Dm + Dv$
The mean in set 1 is equal to $C_1$	$C_1$
The mean in set 2 is equal to $C_2$	$C_2$
The standard deviation in set 1 is equal to $\sigma_1$	$\sigma_1$
The standard deviation in set 2 is equal to $\sigma_2$	$\sigma_2$
The difference, $C_1 - C_2$ , is equal to $\delta$	$\delta$
The sum, $C_1 + C_2$ , is equal to $\gamma$	$\gamma$
The ratio $\sigma_1/\sigma_2$ is equal to $w$	$w$
The ratio $\sigma_2/\sigma_1$ is equal to $x$	$x$

Table 19.2: This table lists the various hypotheses that are of interest in the Behrens-Fisher problem. It has been divided into three sections. The top section is a set of model selection hypotheses and the probabilities for these hypotheses may be used to compute the probabilities in the center section. Finally, the lower section of the table is a series of parameter estimation hypotheses. Each hypotheses in this table requires a separate Bayesian calculation.

deviations change? If something changed by how much. These are only a small sample of all of the hypotheses that are of interest in the Behrens-Fisher problem, see [10] for more on this problem. The entire series of hypotheses about which we wish to make inferences are shown in Fig. 19.2. This table has been divided into three sections. The top section is a series of model selection probabilities, and they are used to determine which of four mutually exclusive and exhaustive hypotheses best describe the data. These probabilities are need to compute the probabilities in both the center and lower sections of this table. The center section of the table are hypotheses who's probabilities may be derived from the four model selection probabilities. The third section of the table is a series of parameter estimation problems. However, as we will see, even these parameter estimation problems depend on first four model selection problems. Because each section of this table produces rather different Bayesian calculations, we will address each section separately.

We are going to use the rules of probability theory to compute the probability for each of the hypotheses appearing in Fig. 19.2. Unfortunately, this is a rather tedious calculation simply because the number of hypotheses of interest is large. We will simply take the entries in the table one at a time. Before we start this process we define a little notation. In the following  $D_1$  will designate the first data set. Which data set is first and which is second is an arbitrary designation and the results will not depend on which is which;  $D_1$  simply designate one of the data sets. In the program that implements these calculations,  $D_1$  is literally the first data set loaded by the interface. Similarly,

$D_2$  will designate the other data set. We will use  $D$  to represent all of the data. The means of the first, second and combined data will be written as  $C_1$ ,  $C_2$  and  $C$  respectively. Combining or pooling the data comes about in several of the Bayesian calculations for the probability for the hypotheses shown in Fig. 19.2. For example, in the model the means and variances are the same, probability theory will lead to pooling the data into a single data set. The standard deviations of the noise prior probabilities will be designated as  $\sigma_1$ ,  $\sigma_2$  and  $\sigma$  for first, second and pooled data. Finally,  $N_1$ ,  $N_2$  and  $N = N_1 + N_2$  will represent the total data in the first, second and pooled data.

### 19.1.1 The Four Model Selection Probabilities

With this notation now established we begin by factoring the probability for the hypotheses, “The means and variances are the same.” This hypotheses is abbreviate as “*SmSv*” where we have used the term variance in this hypotheses. The variance would usually mean  $\sigma^2$ , the squared error; while it is always the standard deviation of the noise prior probability,  $\sigma$ , that appears in our probabilities. So the hypotheses really should have been “the means and the standard deviations of the noise are the same.” This was so verbose, that we decided to use the word “variance” in these hypotheses even though it is not quite the correct expression.

We are going to attack the table from the top to the bottom. We will spend a great deal of time on the first four hypotheses as these are the model selection calculations needed to perform the calculations in the reaming sections. We will derive the probabilities in the center section, but it will turn out these are all just linear combinations of the four probabilities computed for the hypotheses in the first section of the table. Finally, to a large degree, we will only sketch how the probabilities in the bottom section of this table are computed.

The set of hypotheses: *SmSv*, *SmDv*, *DmSv*, and *DmDv*, are mutually exclusive and exhaustive given that the signals in the two data sets are constants. Let us define a model indicator  $\ell$ . When every you see  $\ell$  in an equation, replace it by one of the four hypotheses of interest. To perform a model selection calculation, we must compute the posterior probability for the model indication  $\ell$  given all of the data  $D$  and whatever prior information  $I$  we might have. This posterior probability is represented symbolically by  $P(\ell|DI)$ . To compute it, one applies Bayes’ theorem

$$P(\ell|DI) = \frac{P(\ell|I)P(D|\ell I)}{P(D|I)} \quad (19.1)$$

where  $P(\ell|I)$  is the prior probability for the model,  $P(D|\ell I)$  is the probability for the data given  $\ell$  and  $I$ , and  $P(D|I)$  is a normalization constant:

$$P(D|I) = \sum_{\ell} P(\ell|I)P(D|\ell I). \quad (19.2)$$

If we assign a uniform prior probability to the model indicator, then we have

$$P(\ell|DI) = \frac{P(D|\ell I)}{\sum_{\ell} P(D|\ell I)}. \quad (19.3)$$

If we compute all four probabilities represented symbolically by  $P(D|\ell I)$  then we can always compute the normalization constant. So it is sufficient to compute

$$P(\ell|DI) \propto P(D|\ell I). \quad (19.4)$$

It is this discrete probability distribution that is targeted by the simulated annealing portion of the program that implements the Behrens-Fisher calculations. In the following subsections we compute these four model probabilities represented symbolically by  $P(D|\ell I)$ .

#### 19.1.1.1 The Means And Variances Are The Same

Given the hypotheses, the mean and variance are the same, the model equation which relates the parameters of interest to the data is

$$d_i = C + \text{Noise of Standard Deviation } \sigma \quad (19.5)$$

where  $d_i$  is the pooled or combined data,  $C$  is the mean, and  $\sigma$  is the standard deviation of the noise prior probability.

The probability that is needed is the probability for the data given that the means and variances are the same,  $P(D|SmSvI)$ . This is a marginal probability where the mean,  $C$ , and standard deviation of the noise prior probability,  $\sigma$ , have been removed by marginalization:

$$P(D|SmSvI) = \int_{Low}^{High} dC \int_{\sigma_{Low}}^{\sigma_{High}} d\sigma P(DC\sigma|SmSvI) \quad (19.6)$$

where  $Low$  and  $High$  bound the mean, and similarly,  $\sigma_{Low}$  and  $\sigma_{High}$  are the bounds on the standard deviation.

The right-hand side of this equation is factored using Bayes' theorem, to obtain

$$P(D|SmSvI) \propto \int_{Low}^{High} dC \int_{\sigma_{Low}}^{\sigma_{High}} d\sigma P(C\sigma|SmSvI)P(D|C\sigma SmSvI) \quad (19.7)$$

where  $P(C\sigma|SmSvI)$  is the joint prior probability for the mean and the standard deviation given the model, and  $P(D|C\sigma SmSvI)$  is the direct probability for the data given the parameters and the model.

The prior probability for the parameters,  $P(C\sigma|SmSvI)$ , is factored into two independent prior probabilities:

$$P(C\sigma|SmSvI) = P(C|I)P(\sigma|I) \quad (19.8)$$

where  $P(C|I)$  and  $P(\sigma|I)$  are the prior probabilities for the mean and the standard deviation. Note that we have assumed the prior information is independent of the model, i.e., we will assign the same prior probability for a mean or standard deviation regardless of which model we are discussing.

The prior probability for the mean,  $P(C|I)$ , will be assigned as a bound uniform prior probability

$$P(C|I) = \begin{cases} \frac{1}{R_C} & \text{If } Low \leq C \leq High \\ 0 & \text{Otherwise} \end{cases} \quad (19.9)$$

where  $R_C = High - Low$ .

The prior probability for the standard deviation,  $P(\sigma|I)$ , will be assigned as a bounded Jeffreys' prior:

$$P(\sigma|I) = \begin{cases} \frac{1}{\sigma R_\sigma} & \text{If } \sigma_{Low} \leq \sigma \leq \sigma_{High} \\ 0 & \text{Otherwise} \end{cases} \quad (19.10)$$

and

$$1 = \frac{1}{R_\sigma} \int_{\sigma_{Low}}^{\sigma_{High}} d\sigma \frac{1}{\sigma} \quad (19.11)$$

$$R_\sigma = \log(\sigma_{High}/\sigma_{Low}).$$

Whenever we assign a prior probability for a mean or a standard deviation in the following calculations, it will always be of the form of these two priors and these priors will use the exact same prior ranges and normalization constants.

The only term remaining in Eq. (19.7) to be assigned is the likelihood,  $P(D|C\sigma SmSvI)$ , and this will be assigned using a Gaussian as the noise prior probability:

$$P(D|C\sigma SmSvI) \propto (2\pi\sigma^2)^{-\frac{N}{2}} \exp\left\{-\frac{Q_{SmSv}}{2\sigma^2}\right\} \quad (19.12)$$

where

$$Q_{SmSv} \equiv N(\bar{d}^2 - 2C\bar{d} + C^2) \quad (19.13)$$

where  $\bar{d}^2$  is the mean-square of the combined or pooled data and  $\bar{d}$  is the average pooled data:

$$\bar{d}^2 = \frac{1}{N} \sum_{i=1}^N d_i^2 \quad \text{and} \quad \bar{d} = \frac{1}{N} \sum_{i=1}^N d_i \quad (19.14)$$

where these sums are over all of the combined or pooled data. Together  $\bar{d}^2$ ,  $\bar{d}$  and  $N$  are sufficient statistics for computing this direct probability. The presence of these sufficient statistics is the reason that when the Behrens-Fisher program runs, its execution time is almost completely independent of the data sets being analyzed. The only component of the calculation that depends on the data is computing  $\bar{d}^2$  and  $\bar{d}$ . Computing these averages is negligible compared to the other calculations the program must perform.

Combining the prior and the likelihood, one obtains

$$P(D|SmSvI) \propto \int_{Low}^{High} dC \int_{\sigma_{Low}}^{\sigma_{High}} d\sigma \frac{1}{\sigma R_\sigma R_C} (2\pi\sigma^2)^{-\frac{N}{2}} \exp\left\{-\frac{Q_{SmSv}}{2\sigma^2}\right\} \quad (19.15)$$

as the marginal direct probability for the data given the  $SmSv$  hypotheses. We note in passing that the integrand of this equation is proportional to the joint posterior probability for the parameters given the  $SmSv$  model, a quantity we will need in the parameter estimation calculations discussed in a later Section.

Before we used this probability in the model selection calculation, the integral over  $\sigma$  was evaluated. We did this for technical reasons concerning the implementation of the Markov chain. Evaluating this integral changes the functional form of the direct probability from a Gaussian to Students'  $t$ -distribution, and numerically computing the  $t$ -distribution has some advantages, because it effectively introduces a lower bound to the logarithm of the probability. Evaluating the integral over  $\sigma$ , one obtains

$$P(D|SmSvI) \approx \int_{Low}^{High} dC \frac{1}{2R_\sigma R_C} \Gamma\left(\frac{N}{2}\right) \left[\frac{Q_{SmSv}}{2}\right]^{-\frac{N}{2}}. \quad (19.16)$$

It the integrand of Eq. (19.16) that is used in the simulated annealing portion of the calculation to do model selection given the  $SmSv$  model. In deriving this equation we made the approximation

that the lower and upper bound on the standard deviation of the noise prior probability are so wide that we could effectively extend the integral from zero to infinity. It is possible to do this calculation without making this approximation, see [10]. However, experience with the full calculation shows, that under almost all reasonable circumstances the above approximation is good to many decimal places.

### 19.1.1.2 The Mean Are The Same And The Variances Differ

The hypotheses “the means are the same and the variances differ” implies a slightly changed model equation, one has

$$d_{1i} = C + \text{Noise of standard deviation } \sigma_1 \quad (19.17)$$

and

$$d_{2j} = C + \text{Noise of standard deviation } \sigma_2. \quad (19.18)$$

In this model we cannot use the pooled data and must explicitly take into account the difference in the two data sets.

The probability for the data given that the means are the same and the variances are different is represented symbolically by  $P(D|SmDvI)$ . Calculation of this probability is very similar to what was done in the previous section, and we only sketch the details here:

$$P(D|SmDvI) = \int_{Low}^{High} dC \int_{\sigma_{Low}}^{\sigma_{High}} d\sigma_1 \int_{\sigma_{Low}}^{\sigma_{High}} d\sigma_2 P(DC\sigma_1\sigma_2|SmDvI). \quad (19.19)$$

Applying the product rule and Bayes' theorem to the integrand, one obtains

$$P(DC\sigma_1\sigma_2|SmDvDI) \propto P(C|I)P(\sigma_1|I)P(\sigma_2|I)P(D_1|C\sigma_1SmDvI)P(D_2|C\sigma_2SmDvI) \quad (19.20)$$

where  $P(D_1|C\sigma_1SmDvI)$  and  $P(D_2|C\sigma_2SmDvI)$  are the direct probability for the first and second data sets respectively. As noted earlier, the priors will be assigned using Eqs. (19.9,19.10). A Gaussian likelihood will be assigned to  $P(D_1|C\sigma_1SmDvI)$  and  $P(D_2|C\sigma_2SmDvI)$ . After combining all of the terms, one obtains

$$P(D|SmDvI) \propto \int_{Low}^{High} dC \int_{\sigma_{Low}}^{\sigma_{High}} d\sigma_1 d\sigma_2 \frac{(2\pi\sigma_1^2)^{-\frac{N_1}{2}} (2\pi\sigma_2^2)^{-\frac{N_2}{2}}}{\sigma_1\sigma_2 R_\sigma^2 R_C} \exp \left\{ -\frac{Q_{1SmDv}}{2\sigma_1^2} - \frac{Q_{2SmDv}}{2\sigma_2^2} \right\} \quad (19.21)$$

as the direct probability for the data given the  $SmDv$  model. We note again that the integrand is proportional to the joint posterior probability for the parameters given the  $SmDv$  model. This integrand will be used in the Markov chain Monte Carlo simulation to draw samples from the joint posterior probability for the parameters, and these samples used in forming the probabilities for the hypotheses given in the bottom part of Fig. 19.2. Evaluating the two integrals over  $\sigma_1$  and  $\sigma_2$ , one obtains

$$P(DSmDvI) \approx \int_{Low}^{High} dC \frac{1}{4R_\sigma^2 R_C} \Gamma\left(\frac{N_1}{2}\right) \left[\frac{Q_{1SmDv}}{2}\right]^{-\frac{N_1}{2}} \Gamma\left(\frac{N_2}{2}\right) \left[\frac{Q_{2SmDv}}{2}\right]^{-\frac{N_2}{2}} \quad (19.22)$$

as the direct probability for the data given that the means are the same and the variances differ. We again assumed that the limits on the integral were wide compared to the location of the peak of the integrand. The two functions,  $Q_{1SmDv}$  and  $Q_{2SmDv}$  are define analogously to  $Q_{SmSv}$ :

$$Q_{1SmDv} \equiv N_1(\bar{d}_1^2 - 2C\bar{d}_1 + C^2) \quad (19.23)$$

where  $\bar{d}_1$  and  $\bar{d}_1^2$  are the mean and mean-square of the first data set, and

$$Q_{2SmDv} \equiv N_2(\bar{d}_2^2 - 2C\bar{d}_2 + C^2) \quad (19.24)$$

where  $\bar{d}_2$  and  $\bar{d}_2^2$  are the mean and mean-square of the second data set. It is the integrand of Eq. (19.22) that is used in the simulated annealing part of the simulations to perform model selection for the *SmDv* model.

### 19.1.1.3 The Means Differ And The Variances Are The Same

The hypotheses “the means differ and the variances are the same” again implies a slight change to the model equation. The model for the first data set becomes,

$$d_{1i} = C_1 + \text{Noise of standard deviation } \sigma \quad (19.25)$$

and

$$d_{2j} = C_2 + \text{Noise of standard deviation } \sigma \quad (19.26)$$

for the second data sets.

The probability for the data given that the means differ and the variances are the same is represented symbolically by  $P(D|DmSvI)$ . Again this is a marginal probability and is computed by application of the sum rule:

$$P(D|DmSvI) = \int_{Low}^{High} dC_1 \int_{Low}^{High} dC_2 \int_{\sigma_{Low}}^{\sigma_{High}} d\sigma P(DC_1C_2\sigma|DmSvI). \quad (19.27)$$

The right-hand side of this equation is factored using Bayes’ theorem, to obtain

$$P(DC_1C_2\sigma|DmSvI) \propto P(C_1|I)P(C_2|I)P(\sigma|I)P(D_1|C_1\sigma DmSvI)P(D_2|C_2\sigma DmSvI). \quad (19.28)$$

The priors will be assigned using Eqs. (19.9,19.10). A Gaussian likelihood will be assigned to both  $P(D_1|C_1\sigma DmSvI)$  and  $P(D_2|C_2\sigma DmSvI)$ . Collecting these terms, one has

$$P(D|DmSvI) = \int_{Low}^{High} dC_1 \int_{Low}^{High} dC_2 \int_{\sigma_{Low}}^{\sigma_{High}} d\sigma \frac{(2\pi\sigma^2)^{-\frac{N}{2}}}{\sigma R_\sigma R_C^2} \exp \left\{ -\frac{Q_{1DmSv} + Q_{2DmSv}}{2\sigma^2} \right\} \quad (19.29)$$

as the direct probability for the data given the *DmSv* model. Note the integrand is proportional to the joint posterior probability for the parameters appearing in the *DmSv* model. This integrand is used to generate samples from the joint posterior probability for the parameters appearing in the *DmSv* model. These samples are then used to generate samples from posterior probabilities for parameter estimation hypotheses listed in the lower part of Fig. 19.2.

Evaluating the integral over the standard deviation of the noise prior probability, one obtains

$$P(D|DmSvI) \approx \int_{Low}^{High} dC_1 \int_{Low}^{High} dC_2 \frac{1}{2R_\sigma R_C^2} \Gamma \left( \frac{N}{2} \right) \left[ \frac{Q_{1DmSv} + Q_{2DmSv}}{2} \right]^{-\frac{N}{2}} \quad (19.30)$$

where

$$Q_{1DmSv} \equiv N_1(\bar{d}_1^2 - 2C_1\bar{d}_1 + C_1^2) \quad (19.31)$$

and

$$Q_{2DmSv} \equiv N_2(\bar{d}_2^2 - 2C_2\bar{d}_2 + C_2^2). \quad (19.32)$$

In evaluating the integral over  $\sigma$  we again assumed that the peak in the integrand was such that we make only a small error in extending the limits of the integral from zero to infinity. It is this probability that is used in the simulated annealing portion of the calculations to do model selection for the  $DmSv$  model.

#### 19.1.1.4 The Means And Variances Differ

The hypotheses “the means and variances differ” again implies a new model equation,

$$d_{1i} = C_1 + \text{Noise of standard deviation } \sigma_1 \quad (19.33)$$

for the first data set, and

$$d_{2j} = C_2 + \text{Noise of standard deviation } \sigma_2 \quad (19.34)$$

for the second data set.

The probability for the data given that the means and the variances differ is represented symbolically by  $P(D|DmDvI)$ . Again this is a marginal probability and is computed by application of the sum rule:

$$P(D|DmDvI) = \int_{Low}^{High} dC_1 \int_{Low}^{High} dC_2 \int_{\sigma_{Low}}^{\sigma_{High}} d\sigma_1 \int_{\sigma_{Low}}^{\sigma_{High}} d\sigma_2 P(D_1 D_2 C_1 C_2 \sigma_1 \sigma_2 | DmDvI). \quad (19.35)$$

This calculation is just the one done for the same means and variance, but now computed for each data set separately. We do not give the details of the calculation:

$$\begin{aligned} P(D|DmDvI) &\propto \int_{Low}^{High} dC_1 \int_{\sigma_{Low}}^{\sigma_{High}} d\sigma_1 \frac{(2\pi\sigma_1^2)^{-\frac{N_1}{2}}}{\sigma_1 R_\sigma R_C} \exp\left\{-\frac{Q_{1DmDv}}{2\sigma_1^2}\right\} \\ &\times \int_{Low}^{High} dC_2 \int_{\sigma_{Low}}^{\sigma_{High}} d\sigma_2 \frac{(2\pi\sigma_2^2)^{-\frac{N_2}{2}}}{\sigma_2 R_\sigma R_C} \exp\left\{-\frac{Q_{2DmDv}}{2\sigma_2^2}\right\}. \end{aligned} \quad (19.36)$$

Note that the integrand of this equation is proportional to the joint posterior probability for the parameters appearing in the  $DmDv$ . This integrand is used in the program to generate samples for this probability. These samples are in turn used to generate samples from the probabilities for the hypotheses described in the bottom section of Fig. 19.2.

Evaluating the integrals over the standard deviations, one obtains

$$P(D|DmDvI) \approx \int_{Low}^{High} dC_1 \int_{Low}^{High} \frac{dC_2}{4R_\sigma^2 R_C^2} \Gamma\left(\frac{N_1}{2}\right) \left[\frac{Q_{1DmDv}}{2}\right]^{-\frac{N_1}{2}} \Gamma\left(\frac{N_2}{2}\right) \left[\frac{Q_{2DmDv}}{2}\right]^{-\frac{N_2}{2}} \quad (19.37)$$

where

$$Q_{1DmDv} \equiv N_1(\bar{d}_1^2 - 2C_1\bar{d}_1 + C_1^2) \quad (19.38)$$

and

$$Q_{2DmDv} \equiv N_2(\bar{d}_2^2 - 2C_2\bar{d}_2 + C_2^2) \quad (19.39)$$

as the direct probability for data given the  $DmDv$  model. It is this direct probability that is used in the simulated annealing portion of the calculation that performs model selection.

In the Markov chain Monte Carlo simulation that implements the Bayesian calculation it is the probability for the model, Eq. (19.4), that is targeted by the Markov chain. To target this distribution, there are four probabilities that must be computed, Eqs (19.16,19.22,19.30,19.37). The program that performs the Monte Carlo simulation runs multiple independent simulations. The various simulations start out uniformly distributed over the four different models. One of the routines in the program, varies the model indicator. It does this randomly, for example it might arbitrarily try to switch the indicator from the same means and variances model to the different means and variance model. When it does this, it generates a new simulation having new parameter values unrelated to those in the original simulation. It then varies the parameters in this model using a sub-Markov chain to decorrelate this simulation for the others. Finally, the routine compares the probability for original model indication to the probability for this new model indicator. The new model indicator and parameters are accepted or rejected using the acceptance criteria for a Metropolis-Hastings Markov chain. When the program completes the annealing phase, the distribution of simulations is an approximation to the posterior probability for the model indicator.

### 19.1.2 The Derived Probabilities

The center section of the Fig. 19.2 is in some ways simpler than the first section. By simpler we mean that these probabilities may be computed from the probabilities derived in the previous section. So while we have to apply the rules of probability theory to determine how to compute these, after we do that we should find combinations of things we have already computed.

The probability that the means are the same,  $P(Sm|DI)$ , is a marginal probability where we marginalize over all of the different ways that the means could be the same. There are only two ways the means could be the same. So the probability the means are the same is a sum of two terms:

$$P(Sm|DI) = P(SmSv|DI) + P(SmDv|DI) \quad (19.40)$$

where  $P(SmSv|DI)$  is the probability the means and variances are the same, and  $P(SmDv|DI)$  is the probability the means are the same and the variances differ. But these are just two of the probabilities for the model indicator,  $P(\ell|DI)$ , so after normalizing the probability for the model indicator Eq. (19.40) can be trivially computed for it.

Next the probability that the means are not the same is trivially computed from  $P(Sm|DI)$  because the means are the same or differ are mutually exclusive and exhaustive, so

$$P(Dm|DI) = 1 - P(Sm|DI). \quad (19.41)$$

Like the issue of the means being the same, there are only two ways that the variances could be the same. So this is a marginal probability where we marginalize over the different ways the variances could be the same:

$$P(Sv|DI) = P(SmSv|DI) + P(DmSv|DI). \quad (19.42)$$

These again are computed from the probability for the model indicator  $P(\ell|DI)$ . The probability that the variances differ or are the same from a mutually exclusive and exhaustive set. So the probability that the variances differ,  $P(Dv|DI)$ , is given by

$$P(Dv|DI) = 1 - P(Sv|DI). \quad (19.43)$$

The final probability in the center section is the probability that the data sets differ. There are only two ways the data sets could differ, there could be different means or different variances. The probability for different means or different variances is represented symbolically as  $P(Dm + Dv|DI)$ . To compute this probability we apply the sum rule:

$$P(Dm + Dv|DI) = P(Dm|DI) + P(Dv|DI) - P(DmDv|DI) \quad (19.44)$$

the last term in the sum rule, the joint probability for the two hypotheses doesn't show up in most problems because the hypotheses are mutually exclusive, so the term is frequently zero. However, here the probability for different means and variances is not zero and so this term must be subtracted from the sum of the probability for different means and the probability for different variances.

### 19.1.3 Parameter Estimation

The probabilities in the lower section of Fig. 19.2 are all parameter estimation problems. The program that implements these calculations computes these probabilities by a somewhat roundabout way. It does four parameter estimation calculations, one for each of the four basic models. In these parameter estimation calculations a Markov chain Monte Carlo simulation is run using the integrand of Eqs. (19.12,19.21,19.29,19.36) as the target distributions. From these samples and the probabilities computed in the previous sections, it is possible for the program to form samples for each probability associated with the hypotheses listed in Fig. 19.2.

We give only a sketch of how this is done. For example the hypotheses, the difference in means is  $\delta$ . Designating the difference in means by,  $\delta \equiv C_1 - C_2$ , then the probability for  $\delta$  is given by

$$P(\delta|DI) = P(\delta Sv|DI) + P(\delta Dv|DI) \quad (19.45)$$

where we have restricted oneself to the model subspace that permits the means to be different, i.e., the different means and same or different variance. Factoring the right-hand side of this equation we obtain

$$P(\delta|DI) = P(Sv|DI)P(\delta|DmSvDI) + P(Dv|DI)P(\delta|DmDvDI). \quad (19.46)$$

Note that  $P(Sv|DI)$  and  $P(Dv|DI)$  are things that have already been calculated in the previous sections. But the other two probabilities,  $P(\delta|DmSvDI)$  and  $P(\delta|DmDvDI)$ , have not been calculated. We can compute samples from from these probabilities if we have samples from Eqs. (19.29,19.36). Designating  $(C_1 - C_2|DmSv)$  as the difference in means computed from the parameter estimation samples using the  $DmSv$  model, and  $(C_1 - C_2|DmDv)$  as the difference in means computed from the parameter estimation samples drawn from the  $DmDv$  model, then a sample for the difference in means, independent of the model are given by

$$\text{Sample from: } P(\delta|DI) = P(Sv|DI)(C_1 - C_2|DmSv) + P(Dv|DI)(C_1 - C_2|DmDv). \quad (19.47)$$

So computationally the program takes the samples generated from the  $DmSv$  model, computes the difference in means for each sample, multiples this by  $P(Sv|DI)$  and adds this to the difference in mean computed from the samples generated from the  $DmDv$  model multiplied by the probability for different variances,  $P(Dv|DI)$ . These samples are computed in the output portion of the program. In a similar way the probability for the sum of the means is computed by simply computing the sum of the means rather than the difference.

The samples from the probability for the ratio of the standard deviations are computed in an analogous way

$$\text{Sample from: } P(\sigma_1/\sigma_2|DI) = P(Sm|DI)(\sigma_1/\sigma_2|SmDv) + P(Dm|DI)(\sigma_1/\sigma_2|DmDv). \quad (19.48)$$

Again these samples are used to construct mean and standard deviation estimates for the ratio of standard deviations and to construct an histogram that approximates  $P(\sigma_1/\sigma_2|DI)$ . Samples from the probabilities for the remaining hypotheses given in the bottom section of Fig. 19.2 are computed in analogous fashions.

## 19.2 Outputs From Behrens-Fisher Package

The Behrens-Fisher package is sufficiently different from the other package in the Bayesian Analysis software, that we are going to outline how the program works and describe the outputs of the program in a little more detail than is typical of other packages. First, however the package does have the standard reports. The Text outputs files from the Behrens-Fisher Packages consist of: “Bayes.prob.model,” “BayesBF.mcmc.values,” “Bayes.params,” “Console.log,” “Bayes.accepted” and a “Bayes.Condensed.File.” These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the mcmc.values report is discussed in Appendix D and the other reports are discussed in Chapter 3. Additionally, the “Plot Results Viewer” can be used to view the output probability density functions. In addition to the standard data, model and residual plots there are four model independent posterior probabilities, two mean values and two standard deviation. These plots are named C1, C2, Sigma1, and Sigma2 respectively in the Plot Results Viewer. They are model independent in that they are sum over all of the parameter estimates weighted by the posterior probability for the model. Additionally, there are output plots for the sums, differences and ratios of the mean values. Finally, there are output plots for the parameters given each of the four models.

The Behrens-Fisher package starts by running a Markov chain Monte Carlo simulation with simulated annealing to compute the posterior probability for the model. These four probabilities are given by the integrand of Eqs. (19.16,19.22,19.30,19.37) respectively. The simulation varies the model much like any other parameter in a Markov chain Monte Carlo simulation. To do this, the Markov chain Monte Carlo simulation postulates a change in the model by sampling a uniform prior probability for the model. Say, for example, the package postulates a change from the same mean and variance model to a same mean and different variance model. The Markov chain Monte Carlo simulation will then simulate the parameters in this new model until it reaches equilibrium at the current value of the annealing parameter. It will then either accept or reject this change of model using a Metropolis-Hastings acceptance criteria. Multiple simulations are run parallel at a fixed annealing parameter. Between annealing steps the package outputs the number of simulations in each of the four models. An example of this output is shown in Figure 19.3. This output serves as a visual picture of the posterior probability for the model indicator as it is evolving under simulated annealing. When the package first starts this distribution is initialized using a uniform prior probability for the model. As the annealing parameter is increased the distribution of simulations goes into the posterior probability for the model indicator. The posterior probabilities for the four models, shown in Fig. 19.4, are then output, These four probabilities can then be used to compute the probabilities shown in the center section of Fig. 19.4 and these probabilities are also output.

Figure 19.3: Behrens-Fisher Console Log

Phase	%	<Prior>	<Likelihood>	SmSv	SmDv	DmSv	DmDv
Prob Model	80	-5.39	-6.183379E+01	18	12	0	0
	82	-5.66	-6.198974E+01	18	11	0	1
	84	-5.79	-6.160440E+01	15	15	0	0
	86	-5.89	-6.203856E+01	16	14	0	0
	88	-6.11	-6.172997E+01	14	16	0	0
	90	-6.16	-6.177755E+01	16	14	0	0
	92	-6.11	-6.204847E+01	20	10	0	0
	94	-6.29	-6.175797E+01	19	11	0	0
	96	-6.42	-6.205258E+01	19	11	0	0
	98	-6.56	-6.194344E+01	21	8	1	0

Figure 19.3: Between annealing steps, the Behrens-Fisher package counts the number of simulations in each of the four models and outputs this information to the console log. When the annealing parameter is zero these counts should be roughly equal for the four models. As the annealing parameter increases, the distribution of simulations goes into the posterior probability for the model. Here the program is nearing the completion of the annealing phase, and you can see that the same mean, same variance model is preferred in this data.

After outputting the probabilities from the center section of the table, the program proceeds to sample the joint posterior probability for the parameters for each of the four models. Four separate Markov chain Monte Carlo simulations are run without simulated annealing. The target distributions for these simulations are the integrands of Eqs. (19.12,19.21,19.29,19.36). These four integrands are proportional to joint posterior probability for the parameters given the model indicators. These four sets of samples are then used, along with the model probabilities, to generate samples from the probabilities for the hypotheses given in the bottom section of Fig. 19.2. As each of these phases occur, their passage is noted on in the text window, Fig 19.4 shows an example of this output.

Figure 19.5 is an example of the first section of the BayesBF.mcmc.values report. This first section is pretty standard for MCMC values reports, it contains the configuration parameters and the prior probabilities used in the calculations, the names of the input files, the prior probabilities used for the mean and standard deviations. Finally, the last three lines are the names of the original input files. Note that to get this figure to paginate correctly we truncated the priors to three significant digits, the original contained 5.

The next section of the MCMC values report, Fig. 19.6, contains a plot of the probability for the four models. It also contains the bounds on both the amplitudes and standard deviations. It contains the computed mean and standard deviations for the first, second and combined data sets. Finally, it contains the posterior probability for each of the four models:  $P(SmSv|DI)$ ,  $P(SmDv|DI)$ ,  $P(DmSv|DI)$  and  $P(DmDv|DI)$  respectively. These are followed by three sets of three lines. The first set of three lines are the probability the means are the same,  $P(Sm|DI)$ ; the probability the means differ,  $P(Dm|DI)$ ; and the odd ratio either  $P(Sm|DI)/P(Dm|DI)$  or  $P(Dm|DI)/P(Sm|DI)$  depending on which is greater. The second set of three lines is essentially the same as the first but for the standard deviations rather than the means. Finally, the third set of three lines is the same set of calculations but for the data sets themselves. The remaining lines in Fig. 19.6 are the model

Figure 19.4: Behrens-Fisher Status Listing

Sampling	$P(C \ V   S_m S_v, D_1, D_2, I)$	Same mean,	Same variance
Sampling	$P(C \ V_1 \ V_2   S_m D_v, D_1, D_2, I)$	Same mean,	Different variance
Sampling	$P(C_1 \ C_2 \ V   D_m S_v, D_1, D_2, I)$	Different mean,	Same variance
Sampling	$P(C_1 \ C_2 \ V_1 \ V_2   D_m D_v, D_1, D_2, I)$	Different mean,	Different variance
Computing	$P(C_1   D_1, D_2, I)$	The mean in the first set	
Computing	$P(C_2   D_1, D_2, I)$	The mean in the second set	
Computing	$P(\text{Sigma } 1   D_1, D_2, I)$	The Sd in the first set	
Computing	$P(\text{Sigma } 2   D_1, D_2, I)$	The Sd in the second set	
Computing	$P(C_1 - C_2   D_1, D_2, I)$	The Difference in means	
Computing	$P(C_1 + C_2   D_1, D_2, I)$	The Sum of means	
Computing	$P(S_1 / S_2   D_1, D_2, I)$	The Ratio Of Standard Deviations	
Computing	$P(S_2 / S_1   D_1, D_2, I)$		

Figure 19.4: As each of the probability density functions is sampled the program outputs an indication of what it is doing on console log. Here is a sample of this output. Note in these outputs “V” means Variance. So in the first line the program is trying to tell one that it is sampling the mean and variance of the same mean and same variance model. In the first four lines the program is sampling the parameters for the four basic model. Finally the remaining lines indicate that the program is computing the samples for the probability for  $C_1$ ,  $C_2$  etc.

independent parameter estimates expressed as  $\langle x \rangle \pm \sigma_x$ , where  $x$  is any one of the parameters. The third column, are the parameters which maximized the joint posterior probability for the parameters.

The remaining sections of this report are the parameter estimates given a particular model. There are four models, so there are four sections. Each section contains the estimated parameters given one of the four models. These estimates will be different for each of the four models, and they may be very different depending on the data. An example of the output for the same mean and different variances is shown in Fig. 19.7. The first four lines of this section summarizes the state of the probability density functions at the end of the Markov chain Monte Carlo simulations. The parameter estimates are the mean, standard deviation, and maximum posterior probability estimates of the parameters. In this model there is a mean and two standard deviation, so there are three parameter estimates.

In addition to outputting this information to the BayesBF.mcmc.values file, histograms computed from the Markov chain Monte Carlo simulations are also output. These histograms may be accessed using the Plot Results Viewer. The plotted output consists of the probability for the model, the model independent parameter estimates, the parameter estimates given each model, and a plot of the logarithm of the posterior probabilities for each simulations for each of the four models as a function of the repeat number. This last set of plots, are meant as aids in determining if the Markov chain Monte Carlo simulations have converged. When the simulations are mixing correctly, these plots should show no upward trend, and the trajectories of the individual simulations should overlap or mix together.

Figure 19.5: Behrens-Fisher MCMC Values File, The Preamble

Parameter File Listing for the Behrens-Fisher package

```

! BayesBF Package
! Created 13-Feb-2012 14:17:28 by larry
!
      Output Dir = BayesOtherAnalysis
Number Of Abscissa = 1
Number Of Columns = 1
Number Of Sets = 2
      File Name = BayesOtherAnalysis/001.dat
      File Name = BayesOtherAnalysis/002.dat
MCMC Simulations = 48
      MCMC Repeats = 21
Minimum Annealing Steps = 21
      Histogram Type = Binned
Outlier Detection = Disabled
Total Mcmc Samples = 1008
      Kill Count = 4
Number Of Priors = 2

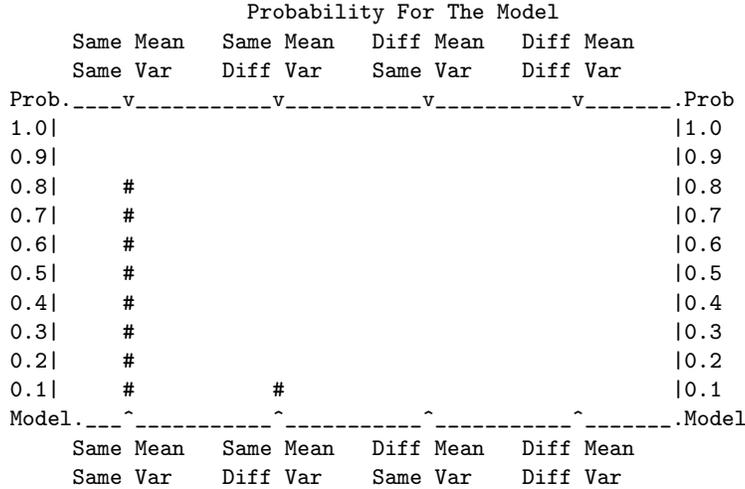
Param Name   Low      Mean     High     Std Dev  Norm     Prior   Ordered  Param Type
Mean         4.69E+01 5.00E+01 5.31E+01 6.25E-01 -3.22E+00 Gaussian NotOrdered NonLinear
StdDev       8.33E-01 1.04E+00 1.25E+00 4.16E-02 -3.22E+00 Gaussian NotOrdered NonLinear

Package Parameters = 2
Input Data Set 1 = BF.smsv.01.dat
Input Data Set 2 = BF.smsv.02.dat

```

Figure 19.5: The preamble of the BayesBF.mcmc.values file is pretty standard for most packages, it contains the configuration parameters used during the run and it contains the prior probabilities used.

Figure 19.6: Behrens-Fisher MCMC Values File, The Middle



The First Input File Is: BF.smsv.01.dat  
 The Second Input File Is: BF.smsv.02.dat

The Amplitude Lower Bound: 46.92  
 The Amplitude Upper Bound: 53.18

The Standard Deviation Lower Bound: 0.8339  
 The Standard Deviation Upper Bound: 1.251

No.	Standard Deviation	Average	Data Set
50	1.1505	49.951	F.smsv.01.dat
50	0.93520	50.146	F.smsv.02.dat
100	1.0477	50.049	Combined

Model	Probability
Same Mean, Same Standard Dev	0.8422619
Different Mean, Same Standard Dev	0.0059524
Same Mean, Different Standard Dev	0.1507937
Different Mean, Different Standard Dev	0.0009921

The probability the means are the same is: 0.9931  
 The probability the means are different is: 0.6944E-02  
 The odds ratio is 143. to 1 in favor of the same means

The probability the standard deviations are the same is: 0.8482  
 The probability the standard deviations are different is: 0.1518  
 The odds ratio is 5.59 to 1 in favor of the same standard deviations

The probability the data sets are the same is: 0.8423  
 The probability the data sets are different is: 0.1577  
 The odds ratio is 5.34 to 1 in favor of the data sets being the same

Figure 19.6: The middle section of this report contains a plot of the normalized posterior probabilities for the four models, It contains the means and standard deviations of the first, second and combined data sets. It contains the posterior probability for the four models. It contains the posterior probability the means are the same. It contains the posterior probability the standard deviations are the same. Finally, it contains the posterior probability the data sets are the same.

Figure 19.7: Behrens-Fisher McMC Values File, The End

```

Model Independent Estimates
Parameter Description      Mean Value      Std. Dev.      Peak Value
C1                          5.00501E+01    9.28245E-02    5.00431E+01
C2                          5.00514E+01    9.27814E-02    5.00445E+01
Sigma 1                     1.06436E+00    6.00040E-02    1.04995E+00
Sigma 2                     1.04186E+00    6.02953E-02    1.02866E+00
Diff In Means              -1.36972E-03    1.25270E-03    -1.36794E-03
Sum Of Means               1.00102E+02    1.85602E-01    1.00088E+02
Ratio Sig1/Sig2           1.02446E+00    1.90638E-02    1.02229E+00
Ratio Sig2/Sig1           9.80543E-01    1.47886E-02    9.70275E-01

Model: SmSv
The Average Log Posterior Probability Was:      Avg.      Sd.
The Average Log Prior Params:                  -6.1835   0.06631
The Average Log Likelihood:                    -147.0154 0.91732
Total Simulations:                              1008
Probability For the Model:                       0.8423

Parameter Description      Mean Value      Std. Dev.      Peak Value
Mean Given Model SmSv     5.00482E+01    1.08521E-01    5.00468E+01
Sd Given Model SmSv       1.05480E+00    7.01560E-02    1.04063E+00

```

Figure 19.7: The final sections of the McMC values file contains a set of model independent parameter estimates. Each estimate consists of the parameter being estimated, the mean value, standard deviation and peak value of the posterior probability. Finally, after these model independent parameter estimates, there are four additional sets of parameter estimates, one estimate for each of the four models. Show here are the estimates from the “SmSv” model: same mean, same variance. These estimates are mean and standard deviation estimates.



## Chapter 20

# Enter Ascii Model

The Enter Ascii Model Package allows you to enter a model of your own and then use Bayesian probability theory to analyze that model.<sup>1</sup> To use this package you do not have to have either Fortran or C installed on your server. However, If you do not have either Fortran or C installed, the only models you will be able to use are the system models. Consequently, installing both Fortran and C is strongly recommended. The interface to this package is shown in Fig. 20.1 To use this package, you must do the following:

**Select** the “Enter Ascii Model” package from the Package menu.

**Load** a Fortran or C model using the “System” or “User” buttons in the “Load And Build Model” widget group.

**Load** one or more Ascii data sets using the Files menu. When a data set is successfully loaded the data is plotted in the Ascii Data viewer. The format of the Ascii data that must be loaded is dependent on the model. Usually the data are two column Ascii, however, in general this package takes multicolumn Ascii data with a multicolumn abscissa. See Appendix A for a detailed description of the Ascii data files used by the Bayesian Analysis software.

**Build** the model using the “Build” button.

**Check** the Analysis Options/Find Outliers box if you suspect outliers are present in the data.

**Review** the prior probabilities for the loaded model using the Prior Viewer.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

---

<sup>1</sup>I would like to build a system library of predefined models. If you have models that you think would be of general use, I would like to hear from you. To have one of your models included, I would need the source code, the parameter file, a brief description of the model equations and data requirements.

Figure 20.1: Enter Ascii Model Package Interface

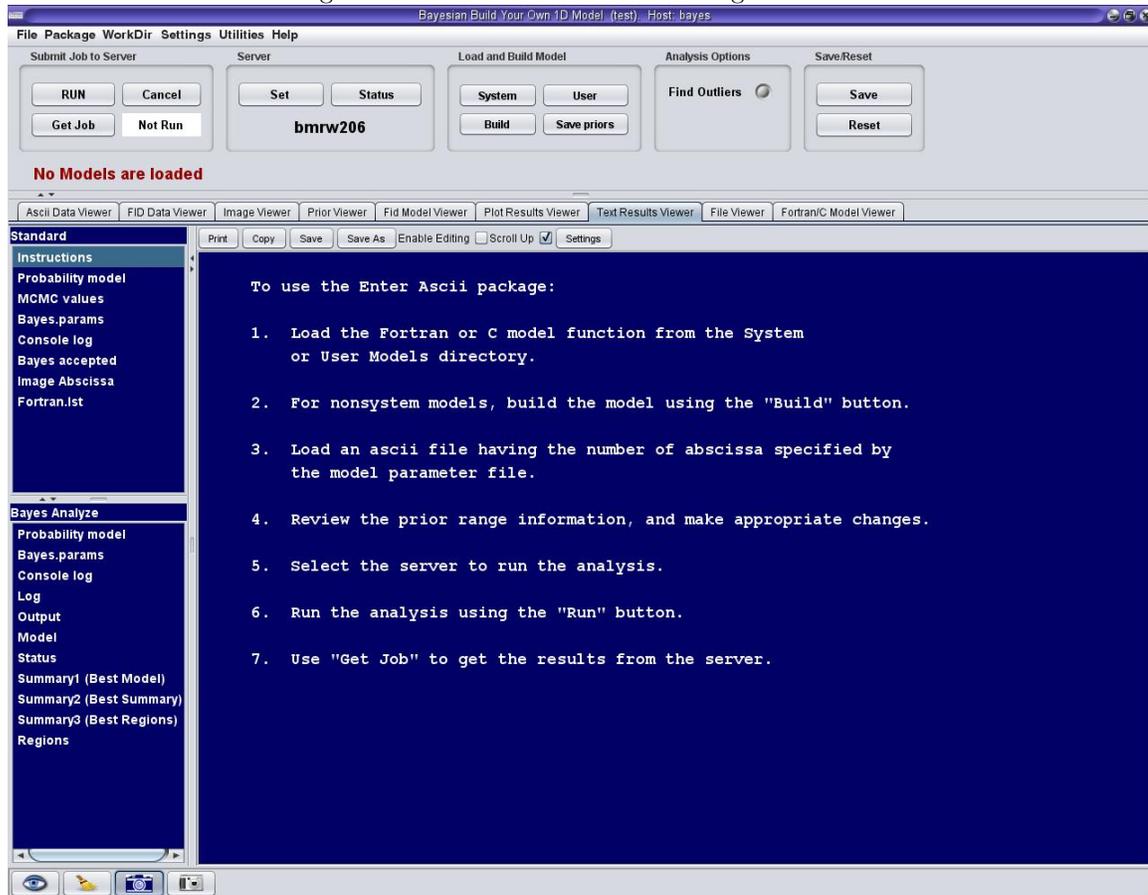


Figure 20.1: All packages that allow the user to load a Fortran or C model have the buttons titled “Load and Build Model.” These buttons allow you to load a model from either the system directory or from you user directory. They allow you to compile a model and save the current prior settings. Additionally, using the “Fortran/C Model Viewer” you can edit, modify and create models, see Appendix E for more on creating Fortran and C models.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

## 20.1 The Bayesian Calculation

The calculation done by Enter Ascii Model Package is a parameter estimation calculation. However, there are two distinct functional forms for the model that are used: one using marginalization over the amplitudes, and one that does not. The model function that does not use marginalization is given by:

$$d_j(t_i) = U_j(t_i, r_1, r_2, \dots) + n_j(t_i) \tag{20.1}$$

where  $d_j(t_i)$  represents a data item in the  $j$ th data set at abscissa value  $t_i$  and  $t_i$  may be vector valued.  $U_j(t_i, r_1, r_2, \dots)$  is the model function.  $r_j$  are the various parameters appearing in the model including any amplitudes that may be present, and  $n_j(t_i)$  represents noise in the  $j$ th data set at abscissa  $t_i$ . Because, this model does not marginalize out the amplitudes, it is possible to restrict the amplitudes ranges using the prior probabilities.

The other model used by this package assumes the amplitudes are to be marginalized from the joint posterior probability for the parameters. The model equation that uses marginalization is similar

$$d_j(t_i) = \sum_{\ell=1}^m A_{j\ell} G_{j\ell}(t_i, r_1, r_2, \dots) + n_j(t_i) \tag{20.2}$$

where the amplitudes are labeled  $A_{jk}$  meaning the  $k$ th amplitude in the  $j$ th data set, the sum is over all of the amplitudes in the model,  $G_{j\ell}(t_i, r_1, r_2, \dots)$  is the  $\ell$ th model function in the  $j$ th data set evaluated at abscissa  $t_i$  and this model equation implicitly assumes that each data set contains same number of amplitudes.

### 20.1.1 The Bayesian Calculations Using Eq. (20.1)

To compute the marginal posterior probability for each parameter using Eq. (20.1), a Markov chain Monte Carlo simulation is run targeting the joint posterior probability for all of the parameters. This joint posterior probability is represented symbolically by  $P(r_1 r_2 \dots | DI)$ . The joint posterior probability for the parameters is factored using Bayes' theorem to obtain

$$P(r_1 r_2 \dots | DI) \propto P(r_1 r_2 \dots | I) P(D | r_1 r_2 \dots I) \tag{20.3}$$

where  $D$  stands for all of the data in all of the data sets,  $P(r_1 r_2 \dots \sigma_1 \dots | I)$ , is factored into independent prior probabilities for each parameter:

$$P(r_1 r_2 \dots | DI) \propto \left[ \prod_{l=1}^m P(r_l | I) \right] P(D | r_1 r_2 \dots I) \tag{20.4}$$

where  $m$  is the total number of parameters in the model, The priors,  $P(r_j | I)$ , are specified in the input parameter file that describes the model. These prior are either the defaults, if you loaded the model from the system directory, or they are the priors set using the interface. Because we don't

know the functional form of these priors, we are going to leave them in symbolic form. Factoring the direct probability for the data into an independent direct probability for each data set, one obtains

$$P(r_1 r_2 \dots | DI) \propto \left[ \prod_{l=1}^m P(r_l | I) \right] \prod_{j=1}^n P(D_j | r_1 r_2 \dots I) \quad (20.5)$$

as the joint posterior probability for the parameters. The direct probability for the data is a marginal likelihood, because the standard deviation of the noise prior probability is not present. Introducing a standard deviation of the noise prior probability,  $\sigma_j$ , for each data set, and using the rules of probability theory to remove these parameters, one obtains:

$$P(r_1 r_2 \dots | DI) \propto \left[ \prod_{l=1}^m P(r_l | I) \right] \prod_{j=1}^n \left[ \int P(\sigma_j | I) P(D_j | \sigma_j r_1 r_2 \dots I) d\sigma_j \right]. \quad (20.6)$$

We have reached the point in this calculation where one has no other choice than to assign probabilities to represent each of these probabilities and then to perform the indicated integrals. Assign a Jeffreys' prior to the prior probability for the noise standard deviation:

$$P(\sigma_j | I) \propto \frac{1}{\sigma_j}, \quad (20.7)$$

and assigning the direct probability for the data using a Gaussian of standard deviation  $\sigma_j$  one obtains

$$P(r_1 r_2 \dots | DI) \propto \left[ \prod_{l=1}^m P(r_l | I) \right] \prod_{j=1}^n \left[ \int \sigma_j^{-(N_j+1)} \exp \left\{ -\frac{Q_j(t_i, r_1, r_2, \dots)}{2\sigma_j^2} \right\} d\sigma_j \right]. \quad (20.8)$$

as the joint posterior probability for the parameters, where  $Q_j(t_i, r_1, r_2, \dots)$  is given by:

$$Q_j(t_i, r_1, r_2, \dots) \equiv \sum_{i=1}^{N_j} [d_j(t_i) - U_j(t_i, r_1, r_2, \dots)]^2, \quad (20.9)$$

and is the total squared residual and is essentially  $\chi^2$ . Evaluating the integral over the standard deviation of the noise, one obtains

$$P(r_1 r_2 \dots | DI) \propto \left[ \prod_{l=1}^m P(r_l | I) \right] \prod_{j=1}^n \left[ \frac{Q_j(t_i, r_1, r_2, \dots)}{2} \right]^{-\frac{N_j}{2}} \quad (20.10)$$

as the joint posterior probability for the parameters, where we have dropped a number of constants that make no difference in this parameter estimation problem.

### 20.1.2 The Bayesian Calculations Using Eq. (20.2)

To compute the marginal posterior probability for each parameter using Eq. (20.2), a Markov chain Monte Carlo simulation is run targeting the joint posterior probability for all of the nonlinear parameters. In this context, nonlinear means all of the parameters appearing in the model in a

nonlinear fashion, i.e., all of the parameters except the amplitudes. This joint posterior probability is represented symbolically by  $P(r_1 r_2 \dots | DI)$ . The joint posterior probability for the nonlinear parameters is factored using Bayes' theorem to obtain

$$P(r_1 r_2 \dots | DI) \propto P(r_1 r_2 \dots | I) P(D | r_1 r_2 \dots I) \tag{20.11}$$

where  $D$  stands for all of the data in all of the data sets, the prior probability for all of the nonlinear parameters is represented by,  $P(r_1 r_2 \dots \sigma_1 \dots | I)$ , and we will factor it into independent prior probabilities for each parameter. Consequently, the joint posterior probability for all of the nonlinear parameters is given by

$$P(r_1 r_2 \dots | DI) \propto \left[ \prod_{j=1}^m P(r_j | I) \right] P(D | r_1 r_2 \dots I) \tag{20.12}$$

where  $m$  is the total number of nonlinear parameters in the model, The priors,  $P(r_j | I)$ , are specified in the input parameter file that describes the model. These prior are either the defaults, if you loaded the model from the system directory, or they are the priors set using the interface. Because we don't know the functional form of these priors, we are going to leave them in symbolic form. Factoring the direct probability for the data into an independent direct probability for each data set, one obtains

$$P(r_1 r_2 \dots | DI) \propto \left[ \prod_{l=1}^m P(r_l | I) \right] \prod_{j=1}^n P(D_j | r_1 r_2 \dots I) \tag{20.13}$$

as the joint posterior probability for the nonlinear parameters.

The direct probability for the data is a marginal likelihood, because neither the standard deviation of the noise prior probability nor the amplitudes are present. To proceed with this calculation, these parameters must be reintroduced into the joint posterior probability for the nonlinear parameters. Representing the standard deviation of the noise prior probability for each data set as  $\sigma_j$  and  $\{A\}_j$  as all of the amplitudes in the  $j$ th data set, one obtains

$$P(r_1 r_2 \dots | DI) \propto \left[ \prod_{l=1}^m P(r_l | I) \right] \prod_{j=1}^n \int P(D_j \sigma_j \{A\}_j | r_1 r_2 \dots I) d\sigma_j d\{A\}_j \tag{20.14}$$

as the joint posterior probability for the parameters. Factoring the right-hand side of this equation, one obtains

$$P(r_1 r_2 \dots | DI) \propto \left[ \prod_{l=1}^m P(r_l | I) \right] \prod_{j=1}^n \int P(\sigma_j | I) P(\{A\}_j | I) P(D_j | \sigma_j \{A\}_j r_1 r_2 \dots I) d\sigma_j d\{A\}_j \tag{20.15}$$

where  $P(\sigma_j | I)$  is the prior probability for the standard deviation of the noise prior probability in the  $j$ th data set. Similarly,  $P(\{A\}_j | I)$  is the joint prior probability for the amplitudes in the  $j$ th data set. If we assume the amplitudes are logically independent, then the joint prior probability for the amplitudes,  $P(\{A\}_j | I)$ , can be factored into a product of prior probabilities for each amplitude:

$$P(r_1 r_2 \dots | DI) \propto \left[ \prod_{l=1}^m P(r_l | I) \right] \prod_{j=1}^n \int P(\sigma_j | I) \left[ \prod_{k=1}^{\nu} P(A_{jk} | I) \right] P(D_j | \sigma_j \{A\}_j r_1 r_2 \dots I) d\sigma_j d\{A\}_j \tag{20.16}$$

where  $P(A_{jk}|I)$  is the prior probability for the  $k$ th amplitude in the  $j$ th data set, and  $\nu$  is the number of data sets. We will assign a zero-mean Gaussian prior probability for each amplitudes. This Gaussian prior probability is given by

$$P(A_{jk}|I) \propto \left( \frac{2\pi\sigma_j^2}{\gamma^2 g_{jkk}} \right)^{-\frac{1}{2}} \exp \left\{ -\frac{A_{jk}^2 \gamma^2 g_{jkk}}{2\sigma_j^2} \right\} \quad (20.17)$$

where

$$g_{jkl} \equiv \sum_{i=1}^{N_j} G_{jk}(t_i) G_{j\ell}(t_i) \quad (20.18)$$

and  $\gamma$  is used to control the width of this prior probability. The reason for this particular functional form is that it allows one to evaluate the integrals over the amplitudes in a concise functional form that aids in doing the numerical calculations. Substituting the prior probability for the amplitudes, Eq. (20.17), into the joint posterior probability for the parameters, Eq. 20.16,

$$\begin{aligned} P(r_1 r_2 \dots | DI) &\propto \left[ \prod_{l=1}^m P(r_l | I) \right] \\ &\times \prod_{j=1}^n \left[ \int \frac{1}{\sigma_j} \left( \frac{2\pi\sigma_j^2}{\gamma^2 g_{j11} \cdots g_{j\nu\nu}} \right)^{-\frac{\nu}{2}} \right. \\ &\times \exp \left\{ -\sum_{k=1}^{\nu} \frac{A_{jk}^2 \gamma^2 g_{jkk}}{2\sigma_j^2} \right\} \\ &\times \left. P(D_j | \sigma_j \{A\}_j r_1 r_2 \dots I) d\sigma_j d\{A\}_j \right] \end{aligned} \quad (20.19)$$

and assigning a Gaussian for the direct probability for the data,  $P(D_j | \sigma_j \{A\}_j r_1 r_2 \dots I)$ , one obtains:

$$\begin{aligned} P(r_1 r_2 \dots | DI) &\propto \left[ \prod_{l=1}^m P(r_l | I) \right] \\ &\times \prod_{j=1}^n \left[ \int \frac{1}{\sigma_j} \left( \frac{2\pi\sigma_j^2}{\gamma^2 g_{j11} \cdots g_{j\nu\nu}} \right)^{-\frac{\nu}{2}} \right. \\ &\times \exp \left\{ -\sum_{k=1}^{\nu} \frac{A_{jk}^2 \gamma^2 g_{jkk}}{2\sigma_j^2} \right\} \\ &\times \left. (2\pi\sigma_j^2)^{-\frac{N_j}{2}} \exp \left\{ -\sum_{i=1}^{N_j} \frac{(d_{ji} - \sum_{k=1}^{\nu} A_{jk} G_{jk}(t_i, r_1 \cdots))^2}{2\sigma_j} \right\} d\sigma_j d\{A\}_j \right]. \end{aligned} \quad (20.20)$$

After evaluating the integrals over the amplitudes, one obtains

$$P(r_1 r_2 \dots | DI) \propto \left[ \prod_{l=1}^m P(r_l | I) \right] \prod_{j=1}^n \left[ \frac{\gamma^2}{g_{j11} \cdots g_{j\nu\nu}} \right] |g_{jkl}|^{-\frac{1}{2}} \left[ \frac{Q_j(r_1 r_2 \dots)}{2} \right]^{-\frac{N_j}{2}} \quad (20.21)$$

with

$$Q_j(r_1 r_2 \dots) \equiv \sum_{i=1}^{N_j} \left[ d_{ji} - \sum_{\ell=1}^{\nu} \hat{A}_{j\ell} G_{j\ell}(t_i r_1 r_2 \dots) \right]^2, \quad (20.22)$$

$|g_{jkl}|$  is the magnitude of the determinate of the  $g_{jkl}$  matrix defined in Eq. (20.18) and the amplitudes  $\hat{a}_{j\ell}$  are given by the solution to

$$\sum_{k=1}^{\nu} g_{jk\ell} \hat{A}_{j\ell} = T_{j\ell} \quad (20.23)$$

with the right-hand side of this equation given by:

$$T_{j\ell} = \sum_{i=1}^{N_j} d_j(t_i) G_{j\ell}(t_i). \quad (20.24)$$

See [2], and [11] for more on how the integrals over the amplitudes are evaluated. Equation 20.21 is the joint posterior probability for the nonlinear parameters that is targeted by the Markov chain Monte Carlo simulations. These simulations only vary the nonlinear parameters, the amplitudes simply do not appear in the posterior probability. However, the amplitudes are output from the simulation. The output amplitudes are given by Eq. (20.23). Because these amplitudes are estimated for each value of the nonlinear parameters, there is as many samples from the distributions of the amplitudes as there is for each of the nonlinear parameters. Consequently, the model that use marginalization do output density functions for the amplitudes.

## 20.2 Outputs Form The Enter Ascii Model Package

The Text outputs files from the Enter Ascii Model packages consist of: “Bayes.prob.model,” “BayesModelAscii.mcmc.values,” “Bayes.params,” “Console.log,” “Bayes.accepted” and a “Bayes.Condensed.File.” These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the mcmc.values report is discussed in Appendix D and the other reports are discussed in Chapter 3. Additionally, the “Plot Results Viewer” can be used to view the output probability density functions. In addition to the standard data, model and residual plots there are probability density functions for each parameter in the currently loaded Fortran/C model. These output probability density functions are named

`ModelFileName.ParamName`

where `ModelFileName` is the name of the currently loaded model. For example, if you have a model named `MyFunnyExp` model, and it has a decay rate named `FunnyRate` the output file containing the posterior probability for `FunnyRate` would be named:

`MyFunnyExp.FunnyRate.`

This naming convention also applies to derived parameters. So, if in addition to generating samples for `FunnyRate`, you also generated samples from a derived inverse decay rate, which was called `FunnyDecayTime` then there would also be an output file named

`MyFunnyExp.FunnyDecayTime`

containing the posterior probability for the decay time. For more on writing Ascii models in either Fortran or C, see Appendix [E](#).

## Chapter 21

# Test Ascii Model

Debugging Ascii models can be very tricky because the user has no way of determining if the model is working correctly. The Test Ascii Model package was designed to help test an Ascii model. This package will let you load your model, data and specify the prior probabilities. When you run the package the program that implements the package will do everything it can to try and break your model. It will evaluate the model at all of the extremes in parameter values and it will generally do everything it can to get your model to throw and interrupt. If it succeeds it will do everything it can to display the parameters and conditions under which the model failed. This has proven very beneficial when trying to debug an Ascii model.

The interface to the Test Ascii Model package is shown in Fig. 21. This interface is nearly identical to the interface to the Enter Ascii Model package, Fig. 20 and using the package is very similar to using the Enter Ascii Model package.

To use this package, you must do the following:

**Select** the “Test Ascii Model” package from the Package menu.

**Load** a Fortran or C model using the “System” or “User” buttons in the “Load And Build Model” widget group.

**Load** one or more Ascii data sets using the Files menu. When a data set is successfully loaded the data is plotted in the Ascii Data viewer. The format of the Ascii data that must be loaded is dependent on the model. Usually the data are two column Ascii, however, in general this package takes multicolumn Ascii data with a multicolumn abscissa. See Appendix A for a detailed description of the Ascii data files used by the Bayesian Analysis software.

**Build** the model using the “Build” button.

**Review** the prior probabilities for the loaded model using the Prior Viewer.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

Figure 21.1: The Test Ascii Model Package Interface

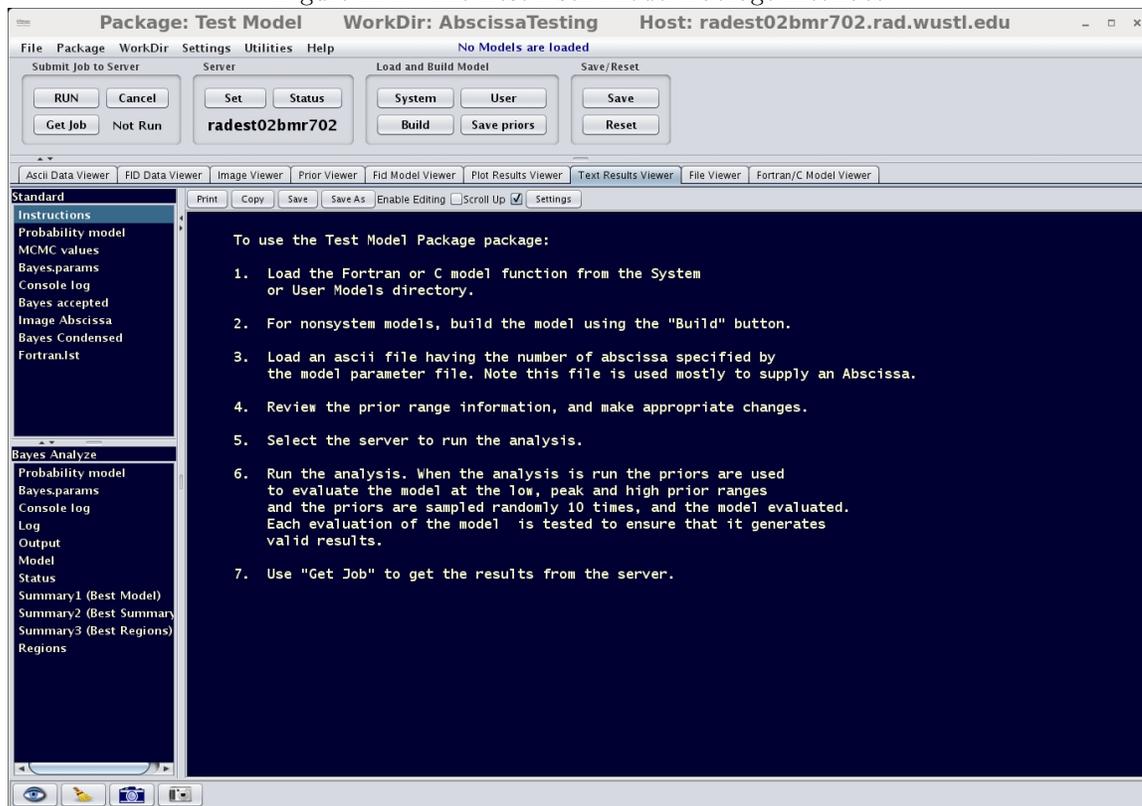


Figure 21.1: This is the interface to a package that lets you test your Ascii models. Debugging Ascii models can be very tricky because the user has no way of determining if the model is working correctly. The Test Ascii Model package was designed to help test an Ascii model. This package will let you load your model, data and specify the prior probabilities. When you run the package the program that implements the package will do everything it can to try and break your model. It will evaluate the model at all of the extremes in parameter values and it will generally do everything it can to get your model to throw and interrupt. If it succeeds it will do everything it can to display the parameters and conditions under which the model failed. This has proven very beneficial when trying to debug an Ascii model.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

The main difference in the Test Ascii Model interface and the Enter Ascii Model interface is that because the Test Ascii Model program is not running an Markov chain Monte Carlo simulation, it has no way to look for outliers. Consequently, the “Find Outliers” widget is not present on the interface.

The output from the Test Ascii Model Package consists of a console log, and a single report Mcmc Values Report., shown in Fig. 21.2. To generate this report, the Inversion Recover model and test data were loaded. The prior range for the decay rate constant was set from zero to one and the program was run. The first line is the results of evaluation the low value for each parameter. Here this generated a technical error, a zero value for a “Positive” prior generates a divide by zero which is undefined. The High and Peak prior values were evaluated successfully. After testing the model using the low and high values from the prior probabilities, a series of 10,000 random parameters samples from the prior were tested. In this case, none of those 10,000 samples caused any problems. Finally, a simple search was run for the value of the parameters that maximized the posterior probability and when these values were evaluated, no errors were found.

Figure 21.2: The Mcmc Values Report For The Test Ascii Model Package

Summary Report for the Test Model package using the InvRec model

```
The "Low Prior" parameter 1, log zero is technically invalid for a "Positive" prior
The "High Prior" parameter check was OK.
The "Peak Prior" parameter check was OK.
The "Random Sample" parameter check for sample 0 was OK.
The "Random Sample" parameter check for sample 1000 was OK.
The "Random Sample" parameter check for sample 2000 was OK.
The "Random Sample" parameter check for sample 3000 was OK.
The "Random Sample" parameter check for sample 4000 was OK.
The "Random Sample" parameter check for sample 5000 was OK.
The "Random Sample" parameter check for sample 6000 was OK.
The "Random Sample" parameter check for sample 7000 was OK.
The "Random Sample" parameter check for sample 8000 was OK.
The "Random Sample" parameter check for sample 9000 was OK.
The "Random Sample" parameter check for sample 10000 was OK.
The "Peak Posterior" parameter check was OK.
```

At least one Status error was found

Figure 21.2: This is the Mcmc Values report out of the Test Ascii Model package. To generate this report, the Inversion Recover model and test data were loaded. I reset the prior range from the decay rate to zero to one and ran the program. The first line is the results of evaluation the low value for each parameter. Here this generated a technical error, a zero value for a "Positive" prior generates  $1/0$  which is undefined. All other values resulted in valid function evaluations. The High and Peak prior values were evaluated successfully. After testing the model using the low and high values from the prior probabilities, a series of 10,000 random samples from the prior were tested. In this case, not of those 10,000 samples caused any problems. Finally, a search was implemented for the value of the parameters that maximized the posterior probability and when these values were evaluated, not errors were found.

## Chapter 22

# Enter Ascii Model Selection

The Enter Ascii Model Selection Package allows you to load one or more Ascii model and then use Bayesian probability theory to compute the posterior probability for the loaded model, thus allowing you to determine which model best accounts for the data.<sup>1</sup> To use this package you do not have to have Fortran or C installed on your server. However, if you do not have Fortran or C installed, you must use the system models. Consequently, installing both Fortran and C is strongly recommended. The interface to this package is shown in Fig. 22. To use this package, you must do the following:

**Select** the “Enter Ascii Model Selection” package from the Package menu.

**Load** one or more Fortran or C model using the “System” or “User” buttons in the “Load And Build Model” widget group.

**Load** one or more Ascii data sets using the Files menu. When a data set is successfully loaded the data is plotted in the Ascii Data viewer. The format of the Ascii data that must be loaded is dependent on the model. Usually the data are two column Ascii, however, in general this package takes multicolumn Ascii data with a multicolumn abscissas. See Appendix A for a detailed description of the Ascii data files used by the Bayesian Analysis software.

**Build** the model using the “Build” button would normally be the next step. However, in this package it is assumed that you have previously compiled and tested each Ascii model, so this package does not have a “Build” button.

**Check** the Find Outliers box if you suspect outliers are present in the data.

**Review** the prior probabilities for the loaded model using the Prior Viewer would normally be the next step. However, because multiple models are loaded and thus multiple parameter sets are loaded, we require the user to test his models prior to running this package. So there is no review of the prior probabilities in this package. You can test your models using either the Enter Ascii Model Package or the Test Ascii Model package and you can review and update the prior information using those packages.

---

<sup>1</sup>I would like to build a system library of predefined models. If you have models that you think would be of general use, I would like to hear from you. To have one of your models included, I would need the source code, the parameter file, a brief description of the model equations and data requirements.

Figure 22.1: The Enter Ascii Model Selection Package Interface

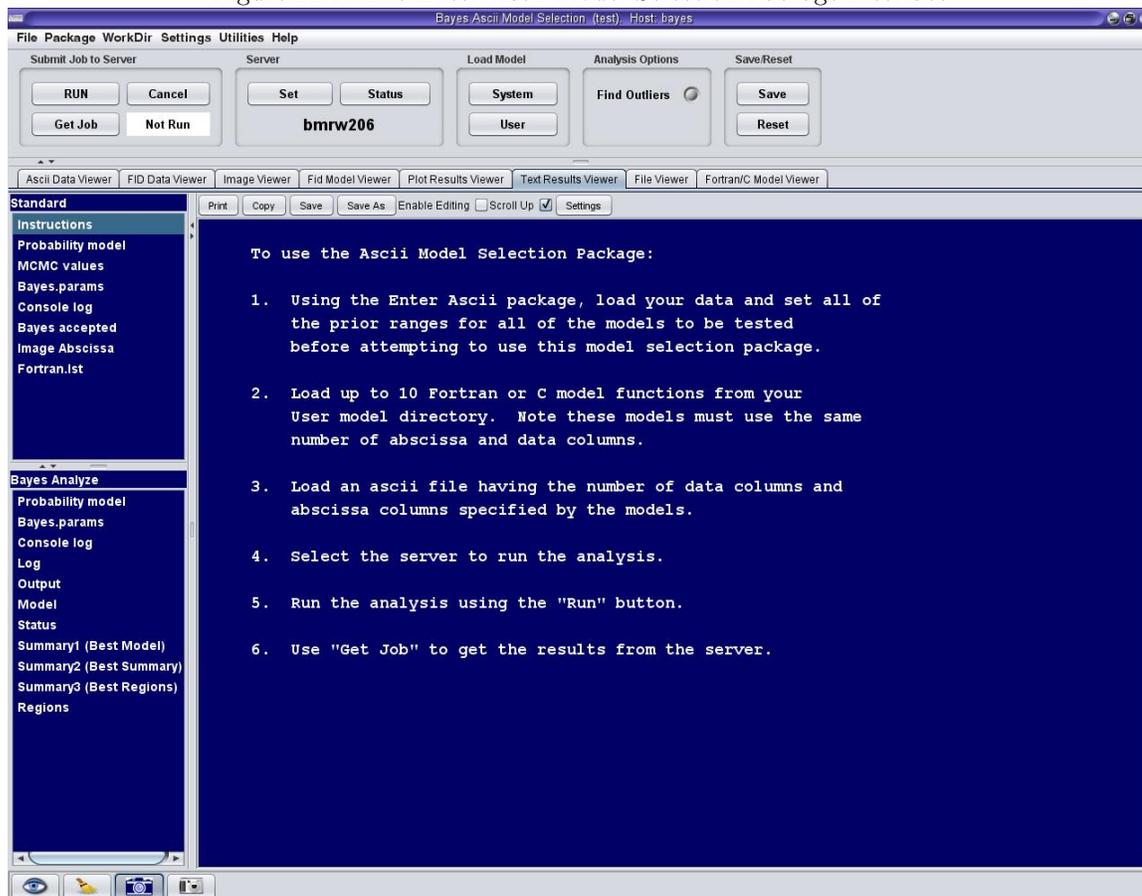


Figure 22.1: This is the interface to the Enter Ascii Model Selection package. This package allows you to load multiple Fortran or C models and then run the the model selection calculation using these models. The program will select the best model from within the set of loaded models. For more on the actual calculations and the widgets see the text.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

## 22.1 The Bayesian Calculations

In the model selection calculation done by the Enter Ascii Model Selection package, it assumes one has a set of models,  $U_j \equiv \{U_1, \dots, U_m\}$ , and one wishes to compute the posterior probability for each of the  $U_j$  models. These models can be loaded from the System directory or they can be loaded from the user directory. They don't need to have common parameters, but they do need to have common data requirements. Each of these models will have a set of parameters associated it and these parameters will be designated as  $\Omega_j$ . The subscript  $j$  indicating that these are the parameters associated with model  $U_j$ . The equation that relates model  $U_j$  to the data is given by

$$d_k(t_i) = U_j(t_i, \Omega_j) + n_k(t_i) \tag{22.1}$$

where  $d_k(t_i)$  represents a data item in the  $k$ th data set, sampled at abscissa value  $t_i$ . The  $t_i$  may be vector valued or it may be a single column of numbers. However, it must have the same number of columns in all loaded models. The noise is represented symbolically by  $n_k(t_i)$  and is the noise value in the  $k$ th data set sampled at abscissa value  $t_i$ .

The posterior probability for each model  $U_j$  is given by Bayes' theorem:

$$P(U_j|DI) = \frac{P(U_j|I)P(D|U_jI)}{P(D|I)} \tag{22.2}$$

where  $P(U_j|DI)$  is the posterior probability for model  $U_j$  given the data and the prior information,  $P(U_j|I)$  is the prior probability for model  $U_j$  given only the prior information,  $P(D|U_jI)$  is the marginal direct probability for all of the data given the model  $U_j$  and the prior information. This term was called a direct probability because it is a probability for the data and it is a marginal probability because none of the parameters from model  $U_j$  appear, so they have been marginalized out.  $P(D|I)$  is a normalization constant that ensures the probabilities sum to one:

$$\sum_{j=1}^m P(U_j|DI) = 1, \tag{22.3}$$

so

$$\sum_{j=1}^m \frac{P(U_j|I)P(D|U_jI)}{P(D|I)} = 1, \tag{22.4}$$

and

$$\sum_{j=1}^m P(U_j|I)P(D|U_jI) = P(D|I). \quad (22.5)$$

Equation (22.2) cannot be computed in this form because the marginal direct probability,  $P(D|U_jI)$ , cannot be assigned. However, the prior probability for the models,  $P(U_j|I)$  can be assigned and in this calculation it is assigned as a uniform prior:

$$P(U_j|I) = \frac{1}{m}. \quad (22.6)$$

The marginal direct probability for the data given the model,  $P(D|U_jI)$ , can be computed if we reintroduce the model parameters  $\Omega_j$  and then use the sum rule of probability theory to marginalize out these parameters

$$P(D|U_jI) = \int P(D\Omega_j|U_jI)d\Omega_j. \quad (22.7)$$

The probability on the right-hand side of this equation is the joint prior probability for the data and the  $\Omega_j$  parameters given the model  $U_j$  and the prior information  $I$ . Applying the product rule to factor the right-hand side of this equation, one obtains

$$P(D|U_jI) = \int P(\Omega_j|I)P(D|\Omega_jU_jI)d\Omega_j \quad (22.8)$$

where  $P(D|\Omega_jU_jI)$  is the direct probability for the data given the parameters and the model, this direct probability is also called a likelihood and  $P(\Omega_j|I)$  is the joint prior probability for the  $\Omega_j$  parameters.

In the Bayesian Analysis software that implements this calculation, there are two types of Ascii models: those that do not use marginalization to remove the amplitudes and those that do. The functional form of the Ascii model is very different between these two types of Fortran/C codes. Consequently, the calculation for the marginal direct probability for the data given the model,  $P(D|U_jI)$ , must be split into two separate calculations: one that uses amplitude marginalization and one that does not.

### 22.1.1 The Direct Probability With No Amplitude Marginalization

In this subsection it will be assumed that the input model is one that has not defined any amplitudes and consequently there is no amplitude marginalization in the posterior probability for the parameters. So in this calculation the  $\Omega_j$  parameters are given by  $\Omega_j \in \{\omega_{j1}, \dots, \omega_{j\nu_j}\}$  where  $\nu_j$  is the number of parameters. The individual Ascii model parameters are designated by  $\omega_{jk}$  and these parameters may include amplitudes, but if it does, the program that implements the calculation has no knowledge of them. Assuming logical independence of the parameters, i.e., knowing the value of one parameter would not help us in making inferences about the other parameters, then the joint prior probability for the  $\Omega_j$  parameters can be factored into an independents prior probability for each parameter,

$$P(\Omega_j|I) = \prod_{k=1}^{\nu_j} P(\omega_{jk}|\Omega_jI) \quad (22.9)$$

where  $P(\omega_{jk}|\Omega_j I)$  is the prior probability for the  $k$  parameter in the  $j$ th model. The model  $\Omega_j$  was include in the prior information because what prior we assign to the parameters will definitely be dependent on the model.

The likelihood,  $P(D|\Omega_j U_j I)$ , is the likelihood function for all of the data  $D$ . However, the data  $D$  is made up of multiple data sets,  $D \equiv \{D_1, \dots, D_n\}$ , where  $n$  is the input number of data sets, and each data set consists of  $N_j$  data values, so  $D_j \equiv \{d_j(t_1), \dots, d_j(t_{N_j})\}$ , where  $d_j(t_i)$  is a data item in the  $j$ th data set sampled at abscissa value  $t_i$ . The number of data values in a given data set,  $N_j$ , need not be the same from one data set to another. Assuming logical independence of the various data sets, the joint direct probability for the data,  $P(D|\Omega U_j I)$ , can be written as

$$P(D|\Omega_j U_j I) = \prod_{k=1}^n P(D_k|\Omega_j U_j I). \tag{22.10}$$

Each of the probabilities,  $P(D_k|\Omega_j U_j I)$ , is a direct probability or likelihood for the data in the  $k$ th data set given the model, the parameters, and the prior information. Such direct probabilities are usually assigned using a Gaussian prior probability for the noise, then the likelihood for a single data set becomes

$$P(D_k|\sigma_k \Omega_j U_j I) = (2\pi\sigma_k^2)^{-\frac{N_k}{2}} \exp\left\{-\frac{Q_{jk}^2}{2\sigma_k^2}\right\} \tag{22.11}$$

where the standard deviations have been added to  $P(D_k|\Omega_j U_j I)$  because the right-hand side of this equation cannot be computed unless the standard deviations are known. The total squared residual,  $Q_{jk}^2$ , is defined as:

$$Q_{jk}^2 \equiv \sum_{i=1}^{N_k} [d_k(t_i) - U_j(t_i, \Omega_j)]^2 \tag{22.12}$$

in the  $k$ th data set given the  $j$ th model and its parameters  $\Omega_j$ . Substituting Eq. (22.11) into Eq. (22.10) one obtains

$$P(D|\sigma_1 \dots \sigma_n \Omega_j U_j I) = \prod_{k=1}^n (2\pi\sigma_k^2)^{-\frac{N_k}{2}} \exp\left\{-\frac{Q_{jk}^2}{2\sigma_k^2}\right\} \tag{22.13}$$

as the direct probability for all of the data, where we have modified the notation to indicate that all of the  $\sigma_j$  must be given. Finally, substituting Eq. (22.13), Eq. (22.9) and Eq. (22.6) into Eq. (22.2) one obtains

$$P(U_j|\sigma_1 \dots \sigma_n D I) \propto \int \frac{1}{m} \left[ \prod_{k=1}^{\nu_j} P(\omega_{jk}|I) \right] \left[ \prod_{k=1}^n (2\pi\sigma_k^2)^{-\frac{N_k}{2}} \exp\left\{-\frac{Q_{jk}^2}{2\sigma_k^2}\right\} \right] d\Omega_j \tag{22.14}$$

as the marginal posterior probability for model  $U_j$ . In the package that processes this analysis, the prior probabilities are specified by the user. And because the prior probability for the model was assigned as uniform, it will cancel when this probability density function is normalized. A Markov chain Monte Carlo simulation is used to numerically integrate this equation and thus obtain the posterior probability for the models.

Often the standard deviation of the noise prior probability are not known, so Eq. (22.14) cannot be used. When the standard deviation are not known, one can remove the standard deviations using

the rules of probability theory:

$$P(U_j|DI) = \int P(\sigma_1 \cdots \sigma_n U_j|DI) d\sigma_1 \cdots d\sigma_n d\Omega_j. \quad (22.15)$$

Factoring the right-hand side using the product rule of probability theory, one obtains

$$P(U_j|DI) = \int P(\sigma_1 \cdots \sigma_n|I) P(U_j|\sigma_1 \cdots \sigma_n DI) d\sigma_1 \cdots d\sigma_n d\Omega_j. \quad (22.16)$$

Assuming logical independence, the joint prior probability for the standard deviations can be factored to obtain:

$$P(U_j|DI) = \int \left[ \prod_{k=1}^n P(\sigma_k|I) \right] P(U_j|\sigma_1 \cdots \sigma_n DI) d\sigma_1 \cdots d\sigma_n d\Omega_j. \quad (22.17)$$

Substituting Eq. (22.14) into Eq. (22.17) one obtains

$$P(U_j|DI) \propto \frac{1}{m} \int \left[ \prod_{k=1}^{\nu_j} P(\omega_{jk}|I) \right] \left[ \prod_{k=1}^n (2\pi\sigma_k)^{-\frac{N_k+1}{2}} \exp \left\{ -\frac{Q_{jk}^2}{2\sigma_k^2} \right\} \right] d\sigma_1 \cdots d\sigma_n d\Omega_j \quad (22.18)$$

as the posterior probability for model  $U_j$  given the data and the prior information. The integrals of the standard deviations of the noise prior probability are gamma function integrals. We address those integrals in the next section when we consider the same calculation but with marginalization over the amplitudes. Here we are just going to give the result of marginalizing over the  $\sigma_k$ :

$$P(U_j|DI) \propto \frac{1}{m} \int \left[ \prod_{k=1}^{\nu_j} P(\omega_{jk}|I) \right] \prod_{k=1}^n \left[ \frac{1}{2} \Gamma \left( \frac{N_k}{2} \right) Q_{jk}^{-\frac{N_k}{2}} \right] d\Omega_j. \quad (22.19)$$

For the details of how this integration is performed see subsection 22.1.2.2. When the standard deviation for the noise prior probability is known Eq. (22.18) is used in the numerical simulation that implements this calculation. Knowledge of the standard deviation of the noise prior probability usually comes about in image processing applications because in those applications it is possible to directly sample the noise. However, more often than not, no information is available about the standard deviation of the noise prior probability, then the posterior probability for the model independent of the standard deviations of the noise prior probability, Eq. (22.19), is used. The functional form of this equation is that of a Student's  $t$ -distribution. The Markov chain Monte Carlo simulation that is used to evaluate the remaining integrals targets either Eq. (22.18) or Eq. (22.19) using a Monte Carlo simulation to draw samples from this posterior probability and then uses Monte Carlo integration to evaluate the posterior probability for the models.

### 22.1.2 The Direct Probability With Amplitude Marginalization

The above calculation assumes the amplitudes are not analytically marginalized out. However, the Bayesian Analysis software uses two types of models. One that does not explicitly marginalize out the amplitudes and one that does. The functional form of these models that marginalize out the amplitudes is a special subset of the form assumed above. The models that do not marginalize out the amplitudes may contain amplitudes, but they are not identified as such in the parameter

file associated with the Fortran/C model; rather the amplitudes are just lumped in with the other nonlinear parameters. However, when the amplitudes are marginalized out, the amplitudes are made explicit in the Fortran/C model parameter file and the functional form of the Fortran/C model is different. Consequently, the calculation for the marginal posterior probability for the model is a bit different for these models.

The signal model used in the calculation when the amplitudes are marginalized out, starts out similar to the calculation where no marginalization occurs. The data are related to the  $j$ th model  $U_j$  by

$$d_k(t_i) = U_j(t_i, \mathcal{B}_j, \Omega_j) + n_k(t_i) \quad (22.20)$$

where  $d_k(t_i)$  represents a data item in the  $k$ th data set that was sampled at abscissa value  $t_i$ ,  $n_k(t_i)$  represents the noise in the  $k$ th data set sampled at abscissa value  $t_i$ ,  $\mathcal{B}_j$  is the set of all amplitudes appearing in the  $j$ th model, and  $\Omega_j$  is the set of all nonamplitude parameters,  $\Omega_j \in \{\omega_{j1}, \dots, \omega_{j\nu_j}\}$  and  $\nu_j$  is the number of nonamplitude parameters in the  $j$ th data set. These parameters will be referred to as the nonlinear parameters because they almost always appear in the model in a nonlinear fashion. The model,  $U_j \in \{U_1, \dots, U_m\}$ , is one from the set of models that are being analyzed. When the amplitudes are marginalized out, it is assumed that the  $U_j(t_i, \mathcal{B}_j, \Omega_j)$  have the following functional form:

$$U_j(t_i, \mathcal{B}_j, \Omega_j) = \sum_{\ell=1}^{u_j} B_{jk\ell} G_{j\ell}(t_i, \Omega_j) \quad (22.21)$$

where  $B_{jk\ell}$  is the  $\ell$ th amplitude of the  $j$ th model in the  $k$ th data set, and  $G_{j\ell}(t_i, \Omega_j)$  is the  $\ell$ th subcomponent of the  $j$ th model. Note that the  $G_{j\ell}(t_i, \Omega_j)$  do not depend on the amplitudes in any way; and in this model the only dependence on the amplitudes is a linear dependence. To give an example of this, suppose we are estimating a sum of  $u_j$  sinusoids, the model  $U_j(t_i, \mathcal{B}_j, \Omega_j)$  would be given by

$$U_j(t_i, \mathcal{B}_j, \Omega_j) = \sum_{\ell=1}^{u_j} B_{jk\ell} \cos(2\pi\omega_{j\ell}t_i + \theta_{j\ell}) \quad (22.22)$$

and each submodel component, the  $G_{j\ell}(t_i, \Omega_j)$  are given by

$$G_{j\ell}(t_i, \Omega_j) = \cos(2\pi\omega_{j\ell}t_i + \theta_{j\ell}). \quad (22.23)$$

In this sinusoidal model, the sinusoids depend on the data sets through the amplitude and phase while the frequencies are common to all data sets.

### 22.1.2.1 Marginalizing the Amplitudes

To compute the posterior probability for the model when the amplitudes are marginalized out, we note that that the calculation begins where the previous calculation ends with Eq. (22.14). Rewriting Eq. (22.14) so as to separate out the amplitudes integrals from the integrals over the nonlinear parameters, one obtains

$$P(U_j|DI) \propto \frac{1}{m} \int \left[ \prod_{\ell=1}^{\nu_j} P(\omega_{j\ell}|I) \right] \left[ \prod_{k=1}^n \left( \prod_{z=1}^{u_j} P(B_{jkz}|I) \right) (2\pi\sigma_k)^{-\frac{N_k}{2}} \exp \left\{ -\frac{Q_{jk}^2}{2\sigma_k^2} \right\} \right] dB_j d\Omega_j \quad (22.24)$$

where  $u_j$  is the number of amplitudes in the  $j$ th model. The prior probability for the  $l$ th nonlinear parameter in the  $j$ th model is designated by  $P(\omega_{jl}|I)$ . Finally,  $P(B_{jkz}|I)$  is the prior probability for the  $z$ th amplitude in the  $j$ th model of the  $k$ th data set. Substituting the model, Eq. (22.21), into the definition of  $Q_{jk}^2$ , Eq. (22.12), we can obtain

$$\begin{aligned} Q_{jk}^2 &\equiv \sum_{i=1}^{N_k} [d_k(t_i) - U_k(t_i, \mathcal{B}_j, \Omega_j)]^2 \\ &= \sum_{i=1}^{N_k} \left[ d_k(t_i) - \sum_{\ell=1}^{u_j} B_{jk\ell} G_{j\ell}(t_i, \Omega_j) \right]^2. \end{aligned} \quad (22.25)$$

Multiplying out the brackets one obtains

$$Q_{jk}^2 = d_k(t_i) \cdot d_k(t_i) - 2 \sum_{\ell=1}^{u_j} B_{jk\ell} d_k(t_i) \cdot G_{j\ell}(t_i) + \sum_{\ell=1}^{u_j} \sum_{v=1}^{u_j} B_{jk\ell} B_{jkv} G_{j\ell}(t_i) \cdot G_{jv}(t_i) \quad (22.26)$$

where we are using the “ $\cdot$ ” notation to mean sum over time, so

$$d_k(t_i) \cdot G_{j\ell} = \sum_{i=1}^{N_k} d_k(t_i) G_{j\ell}(t_i). \quad (22.27)$$

And to compress the notation still further, we define

$$d_k^2 = d_k(t_i) \cdot d_k(t_i), \quad (22.28)$$

$$T_{jk\ell} = d_k(t_i) \cdot G_{j\ell}(t_i) \quad (22.29)$$

and

$$g_{j\ell v} = G_{j\ell}(t_i) \cdot G_{jv}(t_i) \quad (22.30)$$

so Eq. (22.26) becomes

$$Q_{jk}^2 = d_k^2 - 2 \sum_{\ell=1}^{u_j} B_{jk\ell} T_{jk\ell} + \sum_{\ell=1}^{u_j} \sum_{v=1}^{u_j} B_{jk\ell} B_{jkv} g_{j\ell v}. \quad (22.31)$$

To evaluate the integrals in Eq. (22.24), we must assign a prior probability for the amplitudes. A zero-mean Gaussian prior probability will be assigned for each amplitudes. This Gaussian prior probability is given by

$$P(B_{jku}|I) \propto \left( \frac{2\pi\sigma_j^2}{\gamma^2 g_{juu}} \right)^{-\frac{1}{2}} \exp \left\{ -\frac{B_{jku}^2 \gamma^2 g_{juu}}{2\sigma_j^2} \right\} \quad (22.32)$$

where  $\gamma$  is used to control the width of this prior probability,  $\sigma_j$  is the standard deviation of the noise prior probability in the  $j$ th data set, and  $g_{jku}$  was defined in Eq. (22.30). The reason for this particular form is that it allows one to evaluate the integrals over the amplitudes in a concise functional form that aids in doing the numerical calculations.

Substituting Eq. (22.32) and Eq. (22.31) into the posterior probability for the model, Eq. (22.24), one obtains

$$\begin{aligned}
 P(U_j|DI) &= \frac{1}{m} \int \left[ \prod_{l=1}^{v_j} P(\omega_{jl}|I) \right] \\
 &\times \prod_{k=1}^n \left[ \frac{1}{\sigma_k} \prod_{\beta=1}^{u_{jk}} \left( \frac{2\pi\sigma_k^2}{\gamma^2 g_{j\beta\beta}} \right)^{-\frac{1}{2}} \exp \left\{ -\frac{B_{jk\beta}^2 \gamma^2 g_{j\beta\beta}}{2\sigma_k^2} \right\} \right] \\
 &\times \left( 2\pi\sigma_k^2 \right)^{-\frac{N_k}{2}} \exp \left\{ -\frac{1}{2\sigma_k^2} \left[ d_k(t_i)^2 - 2 \sum_{\ell=1}^{u_j} B_{jk\ell} T_{jk\ell} + \sum_{\ell=1}^{u_j} \sum_{v=1}^{u_j} B_{jk\ell} B_{jkv} g_{j\ell v} \right] \right\} \\
 &\times dB_{jk1} \cdots dB_{jk u_j} d\sigma_k d\Omega_j
 \end{aligned} \tag{22.33}$$

as the posterior probability for model  $U_j$ . Expanding the product over  $\beta$ , and simplifying this equation somewhat, one obtains

$$\begin{aligned}
 P(U_j|DI) &\propto \int \left[ \prod_{l=1}^{v_j} P(\omega_{jl}|I) \right] \prod_{k=1}^n \left[ \frac{1}{\sigma_k} (2\pi\sigma_k^2)^{-\frac{N_k+u_j}{2}} \gamma^{u_j} g_{j11} \cdots g_{j u_j u_j} \right. \\
 &\times \left. \exp \left\{ -\frac{1}{2\sigma_k^2} \left[ d_k(t_i)^2 - 2 \sum_{\ell=1}^{u_j} B_{jk\ell} T_{jk\ell} + \sum_{\ell=1}^{u_j} \sum_{v=1}^{u_j} B_{jk\ell} B_{jkv} g'_{j\ell v} \right] \right\} \right] \\
 &\times dB_{jk1} \cdots dB_{jk u_j} d\sigma_1 \cdots d\sigma_n d\Omega_j
 \end{aligned} \tag{22.34}$$

where

$$g'_{j\ell v} = g_{j\ell v} (1 + \delta_{\ell v} \gamma^2) \tag{22.35}$$

and  $\delta_{\ell v}$  is the Kronecker delta function. The Kronecker delta function is zero if  $\ell \neq v$  and its one if  $\ell = v$ . In order to evaluate the amplitude integrals a change of variables and functions will be made. The  $g'_{j\ell v}$  matrix is positive definite and of rank  $u_j$ . Let  $e_{\ell v}$  represent the  $v$ th component of the  $\ell$ th normalized eigenvector of  $g'_{j\ell v}$ , then

$$\sum_{v=1}^{u_j} g'_{j\beta v} e_{\ell v} = \lambda_{\ell} e_{\ell\beta} \tag{22.36}$$

where  $\lambda_{\ell}$  is the  $\ell$ th eigenvalue of  $g_{j\ell v}$ . The functions  $H_{j\ell}(t_i)$ , defined as

$$H_{j\ell}(t_i) = \frac{1}{\sqrt{\lambda_{\ell}}} \sum_{v=1}^{u_j} e_{\ell v} G_{jv}(t_i), \tag{22.37}$$

are orthonormal, i.e.,

$$\sum_{i=1}^{N_j} H_{j\ell}(t_i) H_{jv}(t_i) = \delta_{\ell v} \tag{22.38}$$

where  $\delta_{\ell v}$  is the Kronecker delta function and it is these  $H_j$  functions that will be used in the calculation. The model Eq. (22.21) can be rewritten in terms of these orthonormal functions as

$$U_j(t_i, \mathcal{A}_j, \Omega_j) = \sum_{\ell=1}^{u_j} A_{jk\ell} H_{j\ell}(t_i) \tag{22.39}$$

and it is these  $A_{jk\ell}$  that will be used in a change of variables. The amplitudes  $B_{jk\ell}$  are linearly related to the  $A_{jk\ell}$  by

$$B_{jk\ell} = \sum_{\beta=1}^{u_j} \frac{A_{jk\beta} e_{\beta\ell}}{\sqrt{\lambda_\beta}} \quad \text{and} \quad A_{jk\ell} = \sqrt{\lambda_\ell} \sum_{\beta=1}^{u_\beta} B_{jk\beta} e_{\ell\beta}. \quad (22.40)$$

The volume elements in the  $k$ th data set is given by

$$\begin{aligned} dB_{jk_1} \cdots dB_{jk_{u_j}} &= \left| \frac{e_{\ell v}}{\sqrt{\lambda_v}} \right| dA_{jk_1} \cdots dA_{jk_{u_j}} \\ &= \lambda_1^{-\frac{1}{2}} \cdots \lambda_{u_j}^{-\frac{1}{2}} dA_{jk_1} \cdots dA_{jk_{u_j}}. \end{aligned} \quad (22.41)$$

The Jacobin is a function of the  $\Omega_j$  parameters but it is a constant as long as we are not integrating over the  $\Omega_j$  parameters. At the end of the calculation the linear relations between the  $A$ 's and  $B$ 's can be used to calculate the expected values of the  $B$ 's from the expected value of the  $A$ 's,

$$E(B_{jk\ell} | \Omega_j DI) = \langle B_{jk\ell} \rangle = \sum_{\beta=1}^{u_j} \frac{\langle A_{jk\beta} \rangle e_{\beta\ell}}{\sqrt{\lambda_\beta}} \quad (22.42)$$

and the same is true of the second posterior moments:

$$E(B_{jk\ell} B_{jkl} | \Omega_j DI) = \langle B_{jk\ell} B_{jkl} \rangle = \sum_{\beta=1}^{u_j} \sum_{v=1}^{u_j} \frac{e_{\beta\ell} e_{v\ell} \langle A_{jk\beta} A_{jkv} \rangle}{\sqrt{\lambda_\beta \lambda_v}} \quad (22.43)$$

where  $E(B_{jk\ell} | \Omega_j DI)$  means the expected value of the  $\ell$ th amplitude  $B_{jk\ell}$  in the  $j$ th model of the  $k$ th data set given the nonlinear  $\Omega_j$  parameters, the data  $D$  and the prior information  $I$ . Substituting, Eq. (22.40) into Eq. (22.34) and using the definition of the  $H_{jl}$  functions, Eq. (22.37) one can rewrite the posterior probability for the model as:

$$\begin{aligned} P(U_j | DI) &\propto \int \left[ \prod_{l=1}^{v_{jk}} P(\omega_{jkl} | I) \right] \prod_{k=1}^n \left[ \frac{1}{\sigma_k} (2\pi\sigma_k^2)^{-\frac{N_k+u_j}{2}} \gamma^{u_j} \frac{g_{j11} \cdots g_{ju_j u_j}}{\lambda_1^{-\frac{1}{2}} \cdots \lambda_{u_j}^{-\frac{1}{2}}} \right. \\ &\times \exp \left\{ -\frac{1}{2\sigma_k^2} \left( d_k^2 - 2 \sum_{\ell=1}^{u_j} A_{jk\ell} h_{jkl} + \sum_{v=1}^{u_j} A_{jv\ell}^2 \right) \right\} \Big] \\ &\times dA_{jk_1} \cdots dA_{jk_{u_j}} d\sigma_1 \cdots d\sigma_n d\Omega_j \end{aligned} \quad (22.44)$$

where

$$h_{jkl} = \sum_{i=1}^{N_j} d_{jk}(t_i) H_{j\ell}(t_i). \quad (22.45)$$

Completing the square in Eq. (22.44), one obtains:

$$\begin{aligned} P(U_j | DI) &\propto \int \left[ \prod_{l=1}^{v_{jk}} P(\omega_{jkl} | I) \right] \prod_{k=1}^n \left[ \frac{1}{\sigma_k} (2\pi\sigma_k^2)^{-\frac{N_k+u_j}{2}} \gamma^{u_j} \frac{g_{j11} \cdots g_{ju_j u_j}}{\lambda_1^{-\frac{1}{2}} \cdots \lambda_{u_j}^{-\frac{1}{2}}} \right. \\ &\times \exp \left\{ -\frac{1}{2\sigma_k^2} \left[ d_k^2 - h_{jkl}^2 + \sum_{\ell=1}^{u_j} (A_{jk\ell} - h_{jkl})^2 \right] \right\} \Big] \\ &\times dA_{jk_1} \cdots dA_{jk_{u_j}} d\sigma_1 \cdots d\sigma_n d\Omega_j. \end{aligned} \quad (22.46)$$

Designating the integral over the amplitudes as  $I_j$  and isolating it, Eq. (22.46) can be written as:

$$\begin{aligned}
 P(U_j|DI) &\propto \int \prod_{k=1}^n \left[ \frac{1}{\sigma_k} (2\pi\sigma_k^2)^{-\frac{N_k+u_j}{2}} \gamma^{u_j} \frac{g_{j11} \cdots g_{ju_j u_j}}{\lambda_1^{-\frac{1}{2}} \cdots \lambda_{u_j}^{-\frac{1}{2}}} \left[ \prod_{l=1}^{v_j} P(\omega_{jl}|I) \right] \right. \\
 &\quad \left. \times \exp \left\{ \frac{1}{2\sigma_k^2} [d_k^2 - h_{jk\ell}^2] \right\} \times I_j d\sigma_1 \cdots d\sigma_n d\Omega_j \right]
 \end{aligned}
 \tag{22.47}$$

where the amplitude integral is given by:

$$I_j \equiv \prod_{k=1}^n \int \exp \left\{ -\frac{1}{2\sigma_k^2} \sum_{\ell=1}^{u_j} (A_{jk\ell} - h_{jk\ell})^2 \right\} dA_{jk1} \cdots dA_{jk u_j}.
 \tag{22.48}$$

Introducing a change of variables

$$z_{jk\ell} = \frac{1}{\sqrt{2}\sigma_k} (A_{jk\ell} - h_{jk\ell})
 \tag{22.49}$$

with volume element given by

$$\sqrt{2}\sigma_k dz_{jk\ell} = dA_{jk\ell}
 \tag{22.50}$$

the integral  $I_j$  becomes

$$\begin{aligned}
 I_j &\equiv \int_{-\infty}^{\infty} \exp \left\{ -\sum_{\ell=1}^{u_j} z_{jk\ell}^2 \right\} \sqrt{2}\sigma_k dz_{jk1} \cdots \sqrt{2}\sigma_k dz_{jk u_j} \\
 &= (\sqrt{2}\sigma_k)^{u_j} \int_{-\infty}^{\infty} \exp \left\{ -\sum_{\ell=1}^{u_j} z_{jk\ell}^2 \right\} dz_{jk1} \cdots dz_{jk u_j} \\
 &= (\sqrt{2}\sigma_k)^{u_j} (\sqrt{\pi})^{u_j} \\
 &= (\sqrt{2}\sigma_k \sqrt{\pi})^{u_j} \\
 &= (2\pi\sigma_k^2)^{\frac{u_j}{2}}.
 \end{aligned}
 \tag{22.51}$$

Inserting  $I_j$  back into Eq. (22.47), the posterior probability for model  $U_j$  is given by

$$\begin{aligned}
 P(U_j|DI) &\propto \int \prod_{k=1}^n \left[ \frac{1}{\sigma_k} (2\pi\sigma_k^2)^{-\frac{N_k}{2}} \gamma^{u_j} \frac{g_{j11} \cdots g_{ju_j u_j}}{\lambda_1^{-\frac{1}{2}} \cdots \lambda_{u_j}^{-\frac{1}{2}}} \left[ \prod_{l=1}^{v_j} P(\omega_{jl}|I) \right] \right. \\
 &\quad \left. \times \exp \left\{ -\frac{1}{2\sigma_k^2} [d_k^2 - h_{jk\ell}^2] \right\} d\sigma_1 \cdots d\sigma_n d\Omega_j \right].
 \end{aligned}
 \tag{22.52}$$

The reason for the unusual functional form for the prior probability for the amplitudes, Eq. (22.32), is that all of the pesky little values of  $2\pi$  drop out and leave one with a much cleaner functional form for the posterior probability for the model.

### 22.1.2.2 Marginalizing The Noise Standard Deviation

To evaluate the integrals over the  $\sigma_k$ , the integrand must be rearranged to isolate the integral:

$$P(U_j|DI) \propto \int \prod_{k=1}^n \left[ \gamma^{u_j} \frac{g_{j11} \cdots g_{ju_j u_j}}{\lambda_1^{-\frac{1}{2}} \cdots \lambda_{u_j}^{-\frac{1}{2}}} \left[ \prod_{l=1}^{v_j} P(\omega_{jl}|I) \right] \right] \times \int_0^\infty \frac{1}{\sigma_k} (2\pi\sigma_k^2)^{-\frac{N_k}{2}} \exp \left\{ -\frac{1}{2\sigma_k^2} [d_k^2 - h_{jk\ell}^2] \right\} d\sigma_1 \cdots d\sigma_n d\Omega_j. \quad (22.53)$$

All of the integrals over the  $\sigma_k$  have the same functional form, so we can evaluate one of these integrals and then use the results to evaluate the remaining integrals. Evaluating the integral over  $\sigma_k$  one has

$$I_{\sigma_k} \equiv \int_0^\infty \frac{1}{\sigma_k} (2\pi\sigma_k^2)^{-\frac{N_k}{2}} \exp \left\{ -\frac{1}{2\sigma_k^2} [d_k^2 - h_{jk\ell}^2] \right\} d\sigma_k \quad (22.54)$$

and introducing the notation

$$Q_{jk} \equiv \frac{1}{2} [d_k^2 - h_{jk\ell}^2] \quad (22.55)$$

and  $I_{\sigma_k}$  becomes

$$I_{\sigma_k} \equiv (2\pi)^{-\frac{N_k}{2}} \int_0^\infty \sigma_k^{-(N_k+1)} \exp \left\{ -\frac{Q_{jk}}{\sigma_k^2} \right\} d\sigma_k \quad (22.56)$$

This integral can be transformed into a Gamma function. A gamma function is defined as:

$$\Gamma(z) = \int_0^\infty t^{z-1} \exp\{-t\} dz \quad (22.57)$$

so a simple change of variables and a little algebra and the integral  $I_{\sigma_k}$  is then given by

$$I_{\sigma_k} \equiv \left[ \frac{1}{2} \Gamma \left( \frac{N_k}{2} \right) (2\pi Q_{jk})^{-\frac{N_k}{2}} \right] \quad (22.58)$$

Inserting Eq. (22.58) into Eq. (22.53) one obtains

$$P(U_j|DI) \propto \int \left[ \prod_{l=1}^{v_j} P(\omega_{jl}|I) \right] \prod_{k=1}^n \left\{ \left[ \gamma^{u_j} \frac{g_{j11} \cdots g_{ju_j u_j}}{\lambda_1^{-\frac{1}{2}} \cdots \lambda_{u_j}^{-\frac{1}{2}}} \right] \left[ \Gamma \left( \frac{N_k}{2} \right) (2\pi Q_{jk})^{-\frac{N_k}{2}} \right] \right\} d\Omega_j. \quad (22.59)$$

as the posterior probability for the model where we dropped the factor of one half, because it is exactly the same for all of the  $P(U_j|DI)$  and so cancels when this distribution is normalized.

This equation is correct, but not very computationally convenient because it requires an eigenvalue decomposition every time one evaluates the posterior probability. Fortunately, there are a few modifications that can be done that improve computational efficiency significantly without making approximations. First, note the presence of the eigenvalues, they are simply the determinant of the  $g_{jlm}$  matrix and can be computed with a matrix inverse. Making this substitution one obtains

$$P(U_j|DI) \propto \int \left[ \prod_{l=1}^{v_j} P(\omega_{jl}|I) \right] \prod_{k=1}^n \left[ \gamma^{u_j} g_{j11} \cdots g_{ju_j u_j} |g_{jlm}|^{-\frac{1}{2}} \right] Q_{jk}^{-\frac{N_k}{2}} d\Omega_j. \quad (22.60)$$

Next the function  $Q_{jk}$  can be computed using a matrix inverse instead of an eigenvalue decomposition, thus supplying the determinate mentioned earlier as well as allowing evaluation of the  $Q_{jk}$  function. The integral over an amplitude in a Gaussian quadrature integral, just constrains the amplitudes to their maximum posterior probability values. Consequently, we can write

$$Q_{jk} \equiv \sum_{i=1}^{N_j} \left[ d_k(t_i) - \sum_{\ell=1}^{u_j} \hat{B}_{j\ell} G_{j\ell}(t_i, \Omega_j) \right]^2 \quad (22.61)$$

and its mathematically the same as doing the eigenvalue decomposition. The amplitudes in this equation, the  $\hat{B}_{j\ell}$ , are given by the solution to

$$\sum_{k=1}^{\nu} g_{jk\ell} \hat{B}_{j\ell} = T_{j\ell}. \quad (22.62)$$

The right-hand side of this equation is the projection of the data onto the model components and is given by

$$T_{j\ell} = \sum_{i=1}^{N_j} d_j(t_i) G_{j\ell}(t_i). \quad (22.63)$$

See [2], and [11] for more on how the integrals over the amplitudes are evaluated.

Equation 22.60 is the joint posterior probability for the nonlinear parameters and is targeted by the Markov chain Monte Carlo simulations used to evaluate the remaining integrals. These simulations only vary the nonlinear parameters, the amplitudes simply do not appear in the posterior probability. However, the amplitudes are output from the simulation. The output amplitudes are given by Eq. (22.62). Because these amplitudes are estimated for each value of the nonlinear parameters, there is as many samples from the distributions of the amplitudes as there is for each of the nonlinear parameters. Consequently, the model that use marginalization do output density functions for the amplitudes.

## 22.2 Outputs Form The Enter Ascii Model Package

The Text outputs files from the Enter Ascii Model Selection packages consist of: “Bayes.prob.model,” “BayesModelAscii.mcmc.values,” “Bayes.params,” “Console.log,” “Bayes.accepted” and a condensed file “Bayes.Condensed.File.” These output files can be viewed using the Text Viewer or they can be viewed using File Viewer by navigating to the current working directory and then selecting the files. The format of the mcmc.values report is discussed in Appendix D and the other reports are discussed in Chapter 3. Additionally, the “Plot Results Viewer” can be used to view the output probability density functions. In addition to the standard data, model and residual plots there are probability density functions for each parameter in the currently loaded Fortran/C model. These output probability density functions are named

`ModelFileName.ParamName`

where `ModelFileName` is the name of the currently loaded model. For example, if you have a model named `MyFunnyExp` model, and it has a decay rate named `FunnyRate` the output file containing the posterior probability for `FunnyRate` would be named:

`MyFunnyExp.FunnyRate.`

This naming convention also applies to derived parameters. So, if in addition to generating samples for `FunnyRate`, you also generated samples from a derived inverse decay rate, which was called `FunnyDecayTime` then there would also be an output file named

`MyFunnyExp.FunnyDecayTime`

containing the posterior probability for the decay time. For more on writing Ascii models in either Fortran or C, see Appendix [E](#).

## Chapter 23

# Binned Density Function Estimation

The Binned density function estimation package, takes an input data and produces a binned density function with error bars. A binned density function is often called a histogram.

### 23.1 Using The Package

The input data for this package are the samples drawn from an unknown density function. For example, the samples generated from the run of a Markov chain Monte Carlo simulation. For plotting purposes these samples must be numbered, so the input data are a two column Ascii file. This was done so the interface could use the existing plot routines to display the data. To use this package, you must do the following:

**Select** the “Binned Histogram” package from the Package menu.

**Load** the data to be histogrammed from an Ascii input file.

**Set the Histogram Size** box if you wish to force the histogram to be of a certain size. Alternately, use the automatic feature to allow the Binned Histogram to compute the posterior probability for the number of bins.

**Set the Number Of Smoothing Steps** to use. As with the histogram size, you have the option of allowing the Binned Histogram Package to pick the number of smoothing steps. However, as a warning to you, I have never seen the Binned Histogram package select any size other than zero. Consequently, if you want a smoothed binned histogram you will have to manually set the number of smoothing steps.

**Review** the prior probabilities for Binned histogram package. There are two of these, a prior probability for the histogram size and a prior probability for the number of smoothing steps.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server.

Figure 23.1: Binned Histograms Density Estimation Package Interface

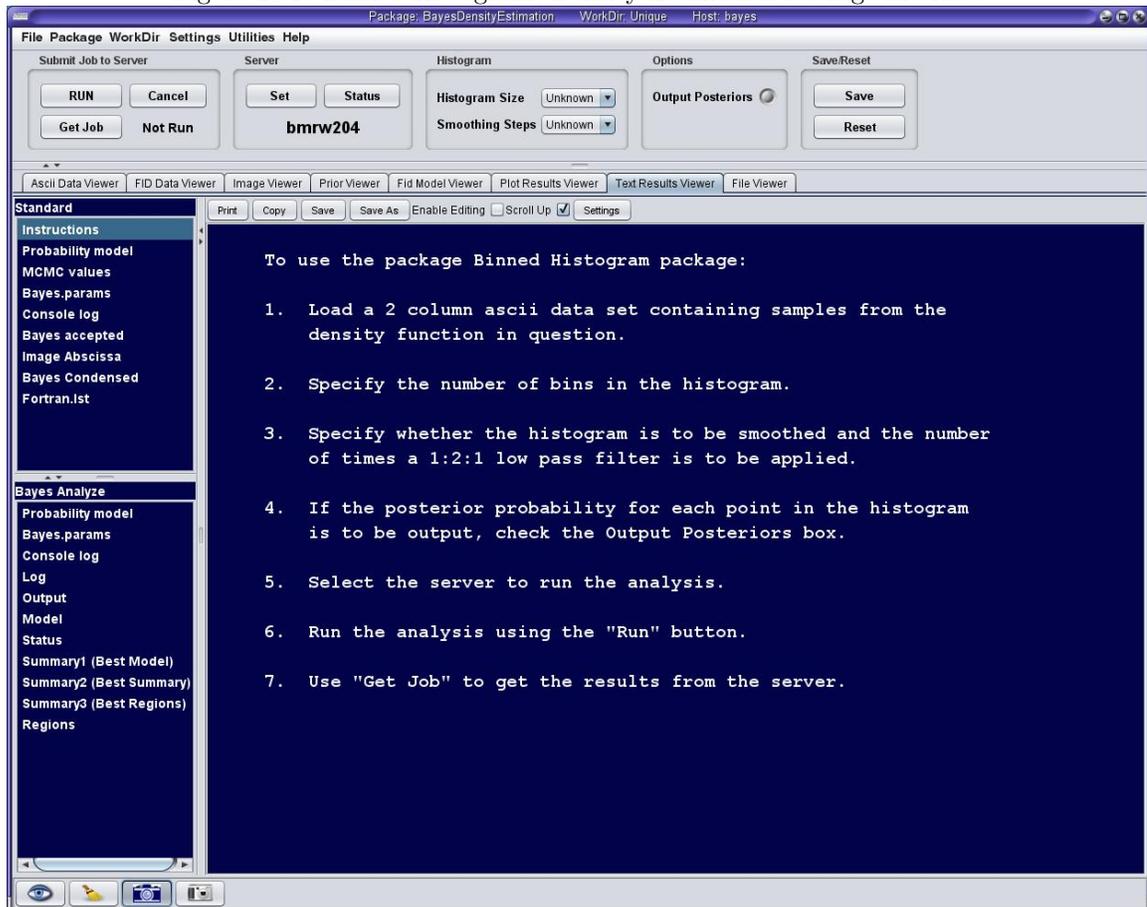


Figure 23.1: This is the interface to the Binned Histogram package. This package will compute the posterior probability for the histogram size and the number of smoothing steps needed to optionally bin the histogram. For more on the actual calculations and the widgets see the text.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

## 23.2 The Bayesian Calculations

The problem of Density Function estimation is an unusual problem because the likelihood of the data is the unknown function that one is trying to determine. In general terms the posterior probability will be related to the data by a prior probability times a direct probability:

$$P(\Omega|DI) = \frac{P(\Omega|I)P(D|\Omega I)}{P(D|I)} \quad (23.1)$$

which is Bayes' theorem [1]. Here  $\Omega$  are a set of parameters that characterize the probability density function,  $P(\Omega|DI)$  is the posterior probability for the  $\Omega$  parameters,  $P(\Omega|I)$  is the prior probability for these parameters,  $P(D|\Omega I)$  is the direct probability for the data and is sometimes called a likelihood, and  $P(D|I)$  is a normalization constant. In the binned histogram problem,  $P(D|\Omega I)$  is given by

$$P(D|\Omega I) = \prod_{i=1}^N P(d_i|\Omega I) \quad (23.2)$$

where this factorization assumes that the probability assigned to any given data values is of the same functional form. The function  $P(x|\Omega I)$  assigns a probability to a value of the data, represented symbolically by  $x$ , given the parameters  $\Omega$ . In the Binned Density estimation problem, the  $\Omega$  parameters are the number of bins and the value of the density function in each bin. If the bins are uniformly spaced then

$$P(d|\Omega I) = \frac{1}{N} P_i \quad d \in x_i \quad (23.3)$$

where  $d \in x_i$  means that the value of  $d$  is in bin  $x_i$  and  $P_i$  is the probability for the data being in the  $i$ th bin. The Binned Histogram package calculates the posterior probability for the number of bins  $m$  and the  $P_i$  for each bin.

To use this package, simply load a two column Ascii data set. An example of such data is shown in Fig. 23.2. This is data set `Bayes.test.data/Histograms/GaussianLikeSamples` and is the samples generated from a Markov chain Monte Carlo simulation. We are going to use this data as an example of how to generate histogram using the Binned Histogram Package. After loading a two column Ascii data set, select the histogram sizes and the number of smoothing steps. In this example we are first going to force the Binned histogram package to use 51 bins and zero smoothing steps. So after setting the number of bins and the number of smoothing steps, we ran the analysis.

The output from this analysis is shown in Fig. 23.3. The Binned Histogram Package uses Markov chain Monte Carlo to estimate the histogram. Typically a number of simulations are run in parallel. These output histograms can be used to compute a mean and standard deviation which we have displayed in Fig. 23.3. The mean value of the estimated histogram is the red line in figure. A one standard deviation error bar is shown as the shaded region around the mean.

Figure 23.2: Binned Histograms Package Example Data

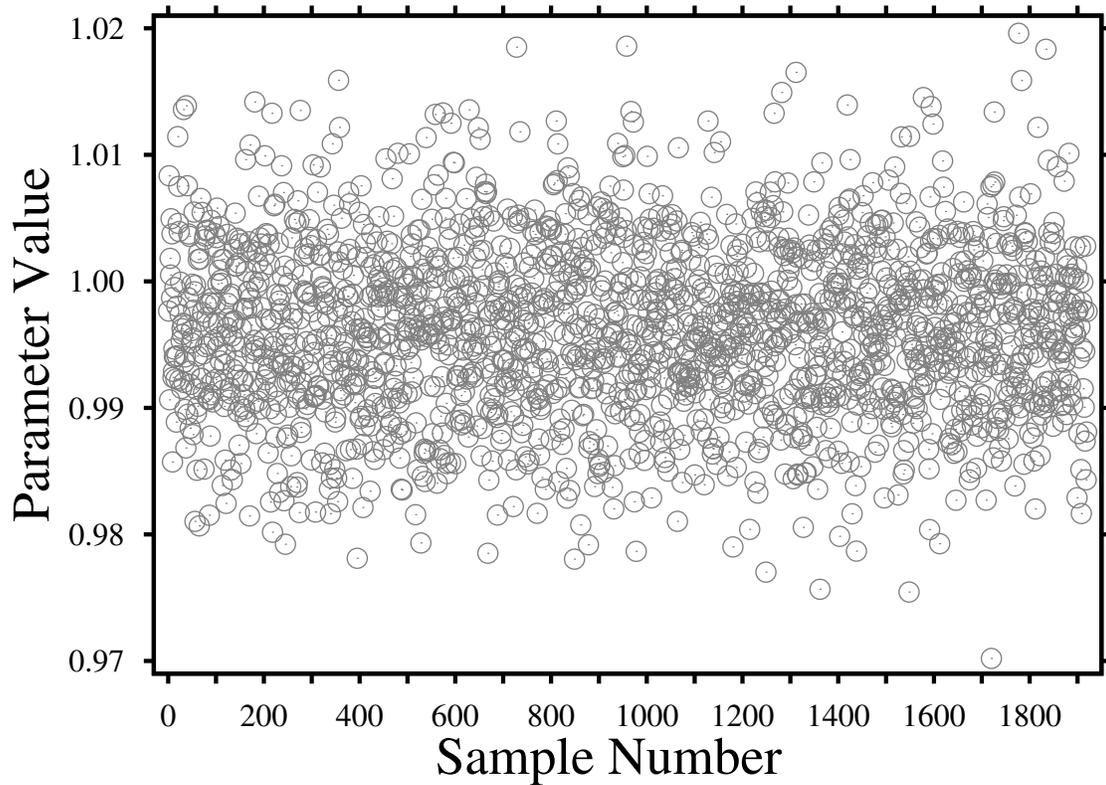


Figure 23.2: This is an example of the Binned Histogram Data. The data are two column, the Abscissa being a simple count of the data. The only function of this Abscissa is when plotting the data. The second column of the data is the one sample drawn from the unknown density function.

Figure 23.3: Binned Histograms Given Number Of Bins

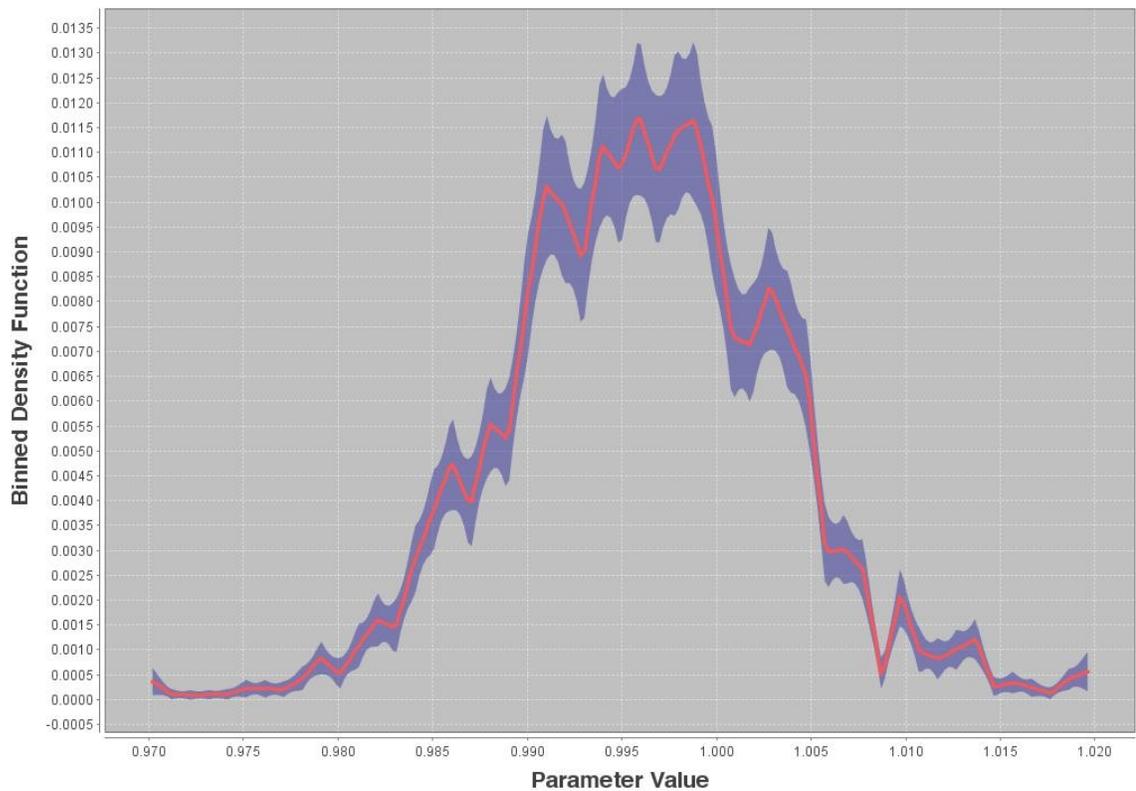


Figure 23.3: This is an example of the Binned Histogram when the number of bins was set to 51. Notice the Binned histogram is very rough due to gaps in the data. The dark shaded region is the region occupied by a one standard deviation in the estimated value of each bin.

Figure 23.4: Binned Histograms With Automatic Determination Of The Number Of Bins

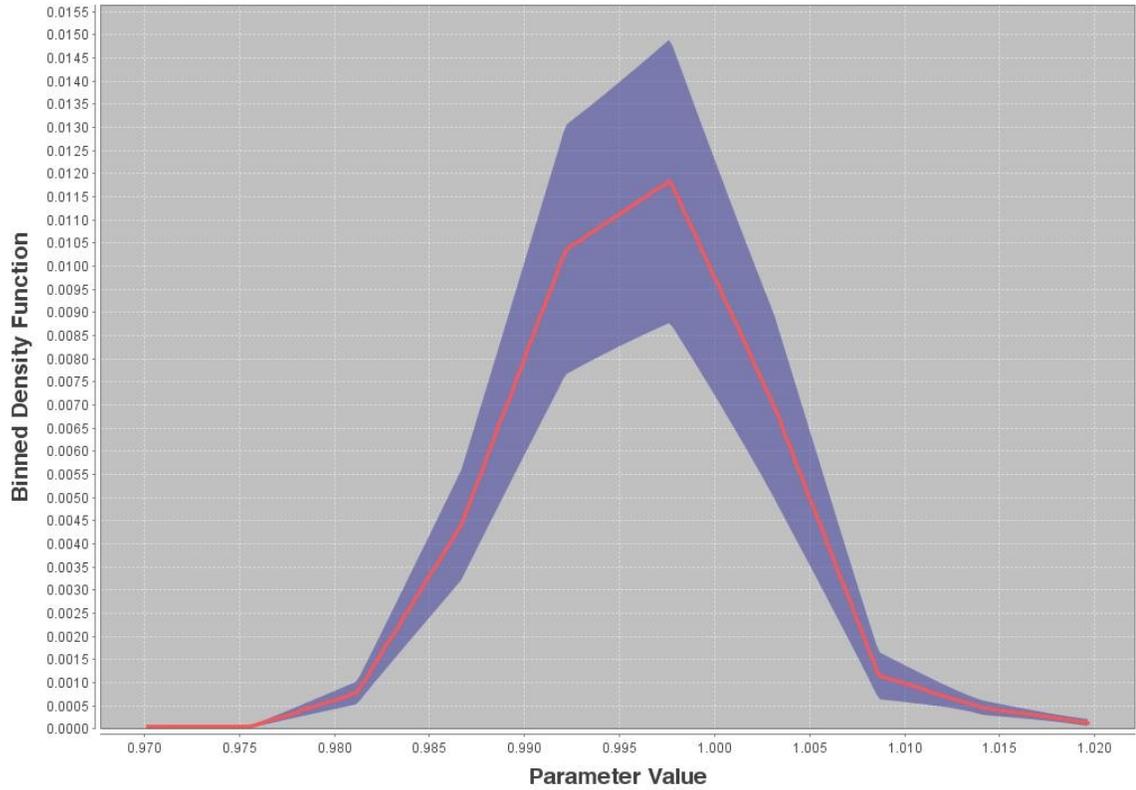


Figure 23.4. This is an example of the Binned Histogram output when the number of bins are determined automatically. Note that the number of bins in this data is fairly small, only 10, but the histogram is much smoother.

Note that, as is normal with a binned histogram, the data has gaps and these gaps result in histogram that are not very smooth.

If you allow the Binned Histogram Package to select the number of bins needed to represent this data, the posterior probability has a strong peak at 10 bins. If we then output the mean and standard deviation of the Binned Histogram, Fig. 23.4

## Chapter 24

# Kernel Density Function Estimation

The Kernel density function estimation package uses Bayesian Probability theory to estimate a kernel density function from a set of input data samples. A kernel is a nonnegative real-valued integrable function. A kernel density function is a superposition of a number of “kernels” where the position and width or smoothness of the kernel are part of the estimation problem. The type of kernels used can be either selected manually or automatically. The program that implements this package has nine different common kernels programmed into it. These nine kernels can be seen [here on Wikipedia](#) and we have included a similar figure plot in Fig. 24.1. The input data for this package are two column Ascii data and are the samples drawn from the unknown density function. The first column of this data is just the sample number or any other convenient “X” axis plotting variable. The second column is the data sample. Even though this package does not use the “X” axis, it is required by the interface so that the data can be plotted using the Ascii plotting tools. These data samples are often the samples generated from the run of a Markov chain Monte Carlo simulation. The problem addressed by this package is selecting the types, number and placement of kernels so as to best represent the unknown sampling distribution function of the data.

### 24.1 Using The Package

The interface to the Kernel Density Function package is shown in Fig. 24.2. In this figure, we have displayed the interface on the Ascii data viewer so that an example data set could be displayed. As already stressed the data are samples drawn from an unknown sampling distribution. These data are be loaded using the Files menu.

To use this package, you must do the following:

**Select** the “Density Estimation Kernel” package from the Package menu.

**Load** the two column Ascii data who’s density function you wish to infer. The interface displayed in Fig. 24.2 has the Ascii data viewer selected with a set of samples loaded. If multiple Ascii data sets are loaded, it is assumed that *all* of the data are samples from a single underlying

Figure 24.1: The Nine Common Kernels Used In This Package

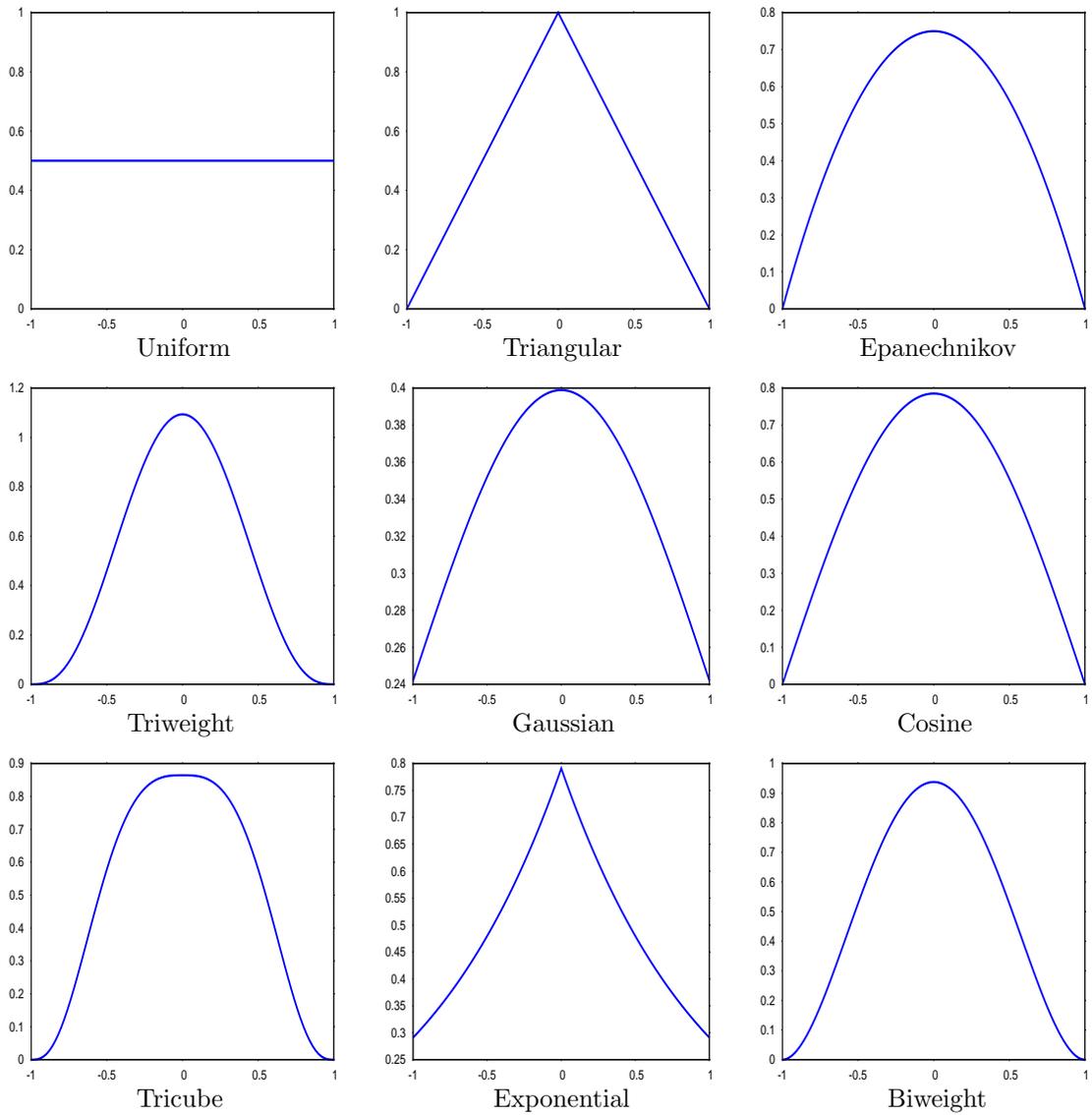


Figure: 24.1: These are a total of nine kernels that have been programmed into the kernel density function estimation package.

Figure 24.2: The interface to the Kernel Density Function Package

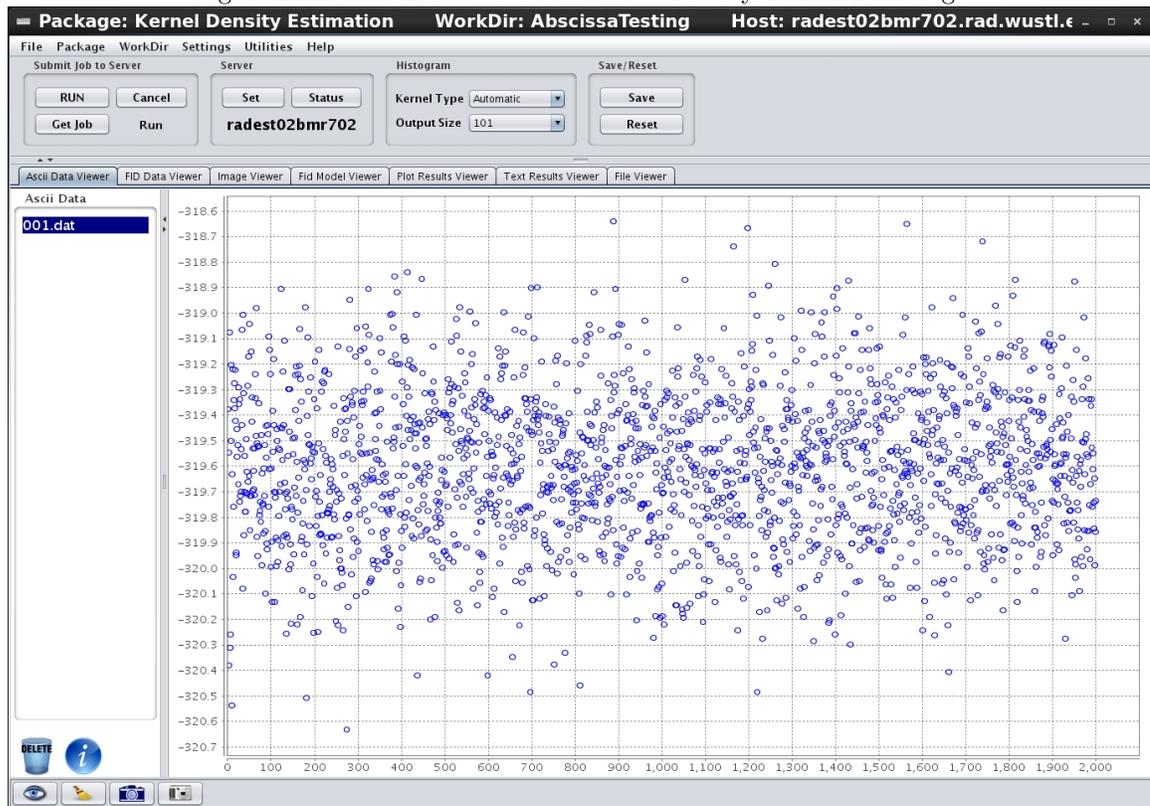


Figure 24.2: This is the interface to the Kernel Density function package. This package will compute the posterior probability for the number and types of kernels needed to represent the unknown density function. The interface displayed here is showing the Ascii data viewer with a set of samples already loaded. For more on the actual calculations and the widgets see the text.

density function. Consequently, all of the data samples are treated as if they were in a single larger data set.

**Set** the “Kernel Type” if you wish to force the package to use a given type of kernel. Alternately, use the “Automatic” feature to allow the Kernel Density function package to compute the posterior probability for the types of kernels needed to best represent the data.

**Set** the number of output bins in the inferred kernel density function. This is done using the pull down menu labeled “Output Size.”

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

## 24.2 The Output Plots

When the analysis has completed the “Get Job” button will fetch the analysis from the server and unpack the results. Like most of the other packages, there are several unique plots and reports and we will give a brief discussion of them here. The first plot we will discuss, Fig. 24.3, is the posterior probability for the kernel type. The kernel type is a discrete variable and is essentially just the name of the kernel. Along the bottom of this plot is the type of kernel and the vertical axis is the probability for this kernel. So the Markov chain Monte Carlo simulation has a variable that takes on the name of the kernel type being used in a given simulation to estimate the density function. When the name is “Uniform” the density function is computed using a uniform kernel, when it’s “Exponential” the kernel density function is computed using an exponential kernel, etc. This plot is of the number of simulations having uniform kernels, exponential kernels, etc. It is normalized so that the sum over all the kernel types is one. In the example shown the “Triweight” kernel was by far the most probable with a small amount of probability in the “Biweight” kernel.

The Kernel Density Function Estimation package has two different discrete model selection calculations going on in the Markov chain Monte Carlo simulations simultaneously. The first, described above, is for the type of kernel that best describes the unknown density function. The second discrete model selection calculation is for the number of kernels needed to represent the density function. Figure 24.4 is an example of this discrete probability distribution. Along the bottom of this plot is the number of kernels needed to represent the unknown density function, in the example shown that number is two. This plot is normally centered on the location of the peak in the posterior probability distribution or situated an the minimum or maximum number of kernels. These minimum and maximum are 1 and 101 respectively. The vertical axis is the posterior probability distribution for the number of kernels. In this example, that posterior probably for two kernels was one, i.e., there were no Markov chain Monte Carlo simulations have 1, 3, 4, etc. kernels; all of the simulations had two kernels.

The third, and perhaps, most important plot is of the model averaged density function. The third

Figure 24.3: Posterior Probability For The Kernel Type

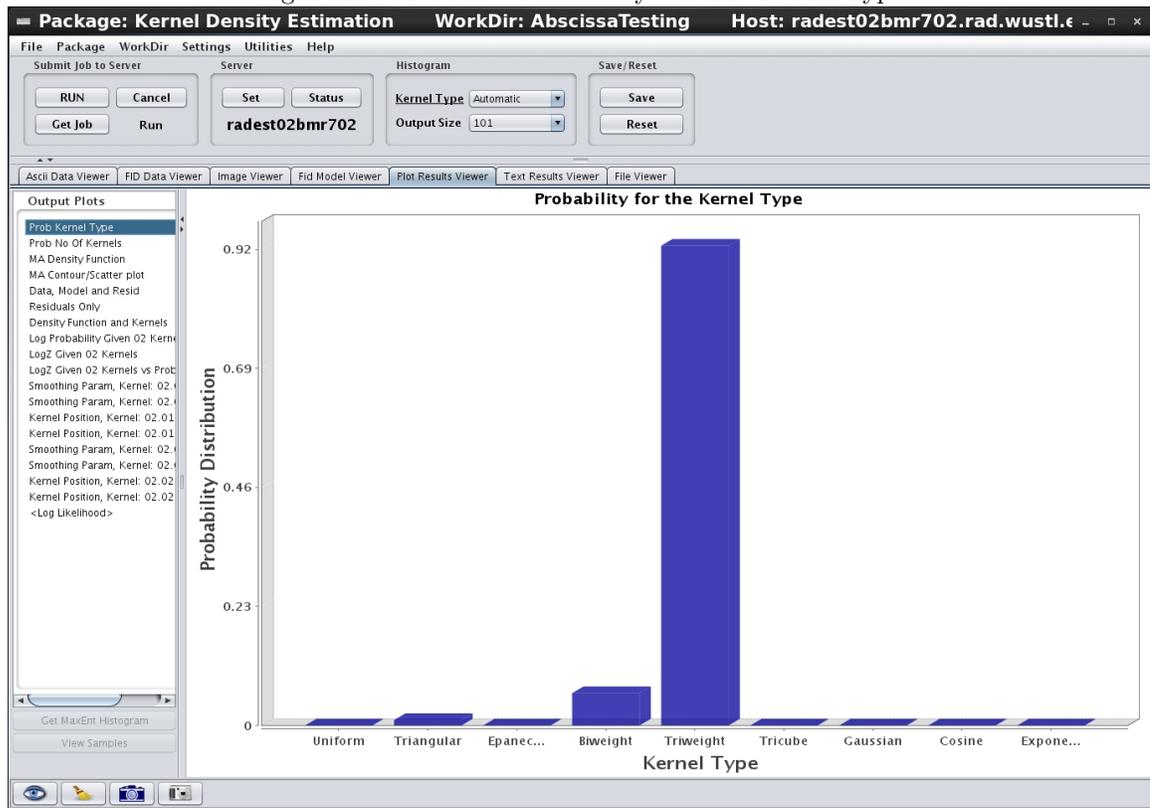


Figure 24.3: The first plot returned by the kernel density function estimation package is the posterior probability for the kernel type given the samples and the prior information. Along the bottom of this plot is the name of the kernel and the vertical axis is the probability for this kernel. In the example shown the “Triweight” kernel was by far the most probable.

Figure 24.4: Posterior Probability For The Number Of Kernels

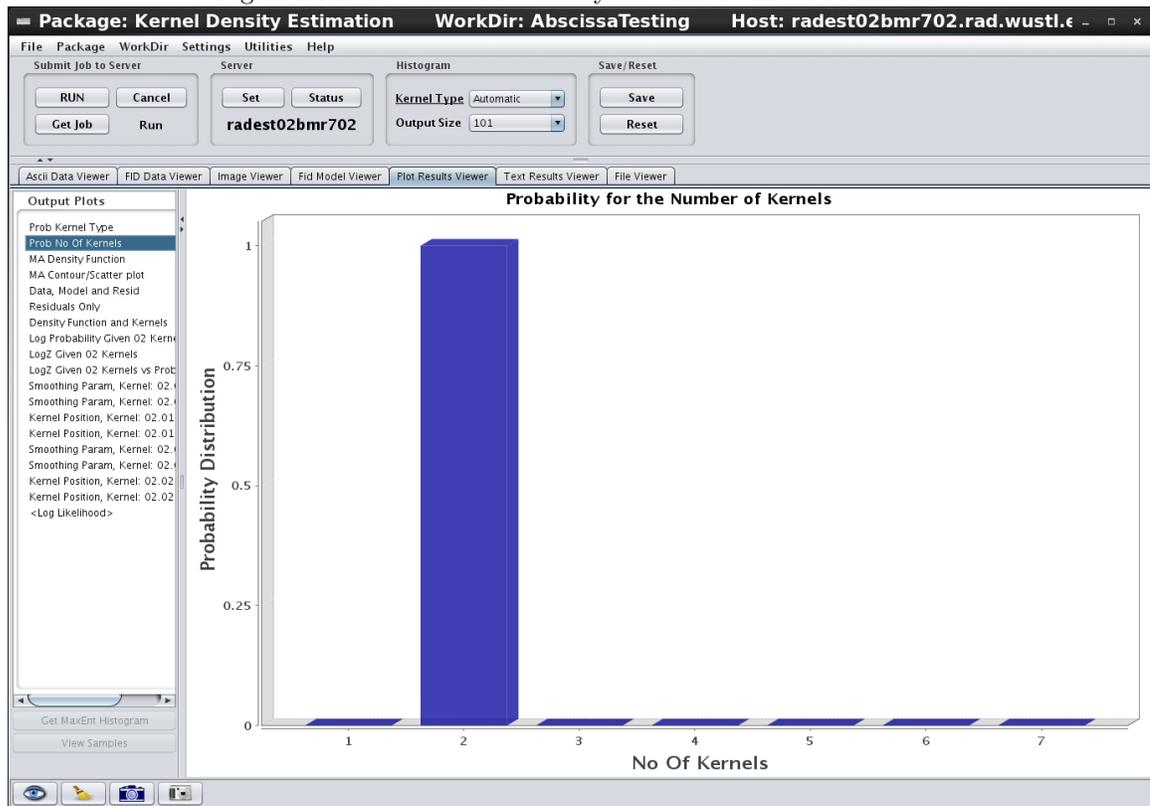


Figure 24.4: The second output plot is for the posterior probability for the number of kernels needed to represent the density function given the data and the prior information. The plot shown here is centered on the location of the peak, the maximum kernels is not seven. There is a hard-coded maximum in the program that implements this calculation of 101 total kernels.

Figure 24.5: The Model Averaged Kernel Density Function

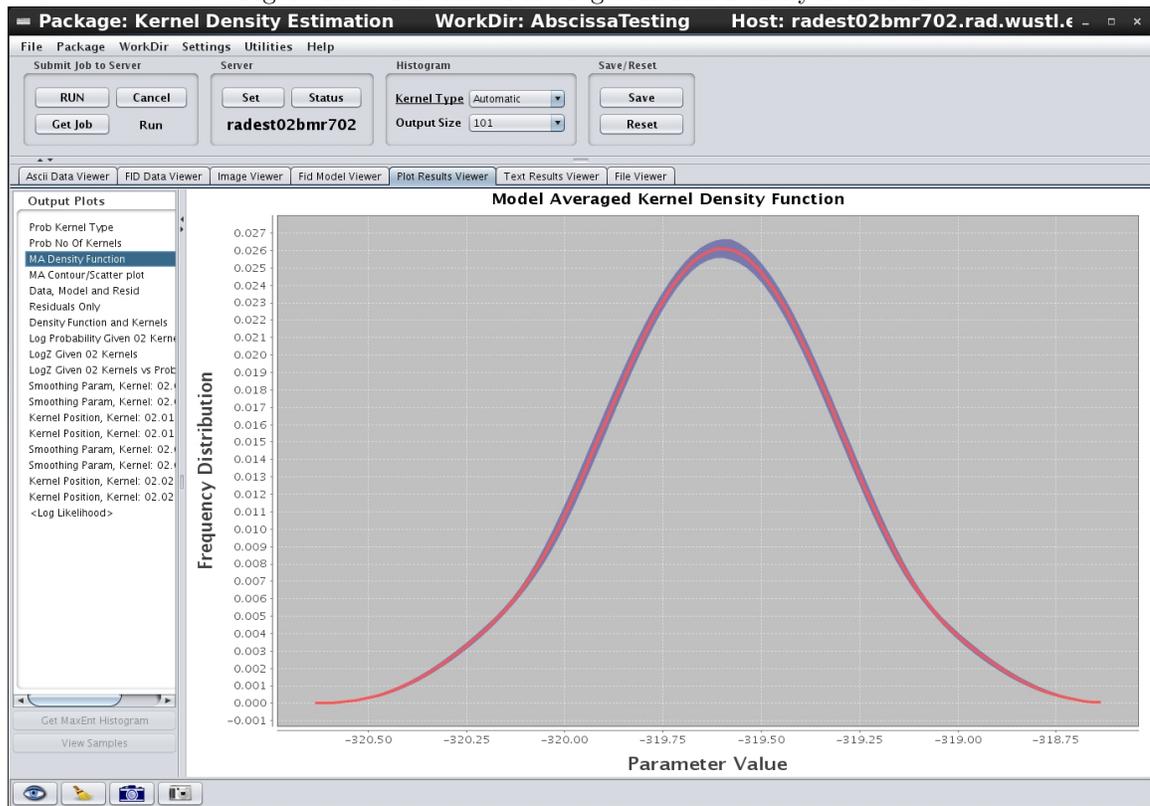


Figure 24.5: The third plot is of the model averaged or mean density function. Each Markov chain Monte Carlo simulation contains a density function generated from the kernels present in that simulation. This mean density function is the average of the density functions from all of the simulations. The shaded region is a one standard deviation computed from mean and standard deviation of these density functions.

Figure 24.6: Kernel Density Function And Samples

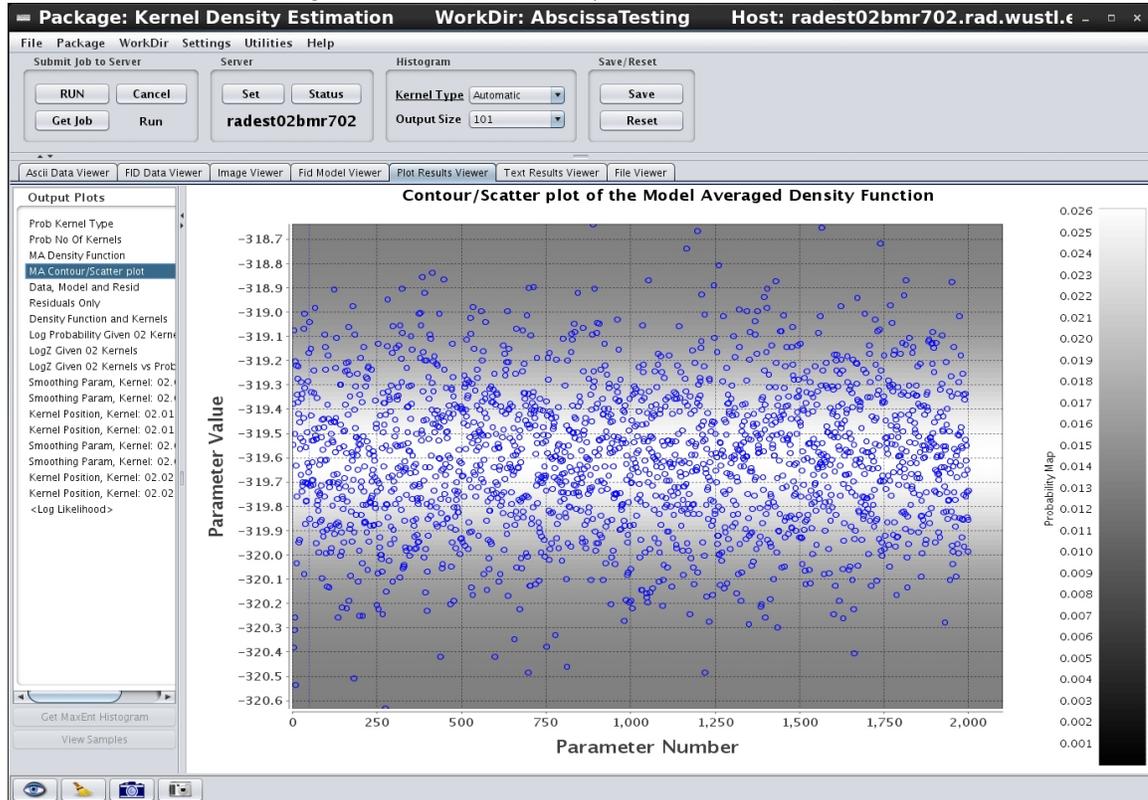


Figure 24.6: The contour/scatter plot of the estimated density function with the individual samples plotted on top of the density function. The gray scale background is the estimated density function. The bar-chart on the right indicates the values of the density function. The open circles are the individual data samples.

plot is of the model averaged or mean density function. Each Markov chain Monte Carlo simulation contains a density function generated from the kernels present in that simulation. Figure 24.5 is the mean density function and is the average of the density functions from all of the simulations. The shaded region is a one standard deviation computed from mean and standard deviation of these density functions. So the shaded region shows how much variability is allowed in the density function given the data and the prior information.

The next plot is a contour plot of the expected density function with the individual samples plotted as points on the density function, Fig. 24.6. The gray scale background is the estimated density function. The bar-chart on the right indicates the values of the density function. The open circles are the individual data samples.

Unlike most other packages, the density estimation packages have no good way to show how well the estimated model fits the data. That's because the density estimation packages don't actually fit the data, the characterize the density of the samples. In a effort to show how well the density estimates fit the data, we first take the data and generate a binned histogram. This histogram is

Figure 24.7: A Binned Histogram, The Model Averaged Density Function And The Difference

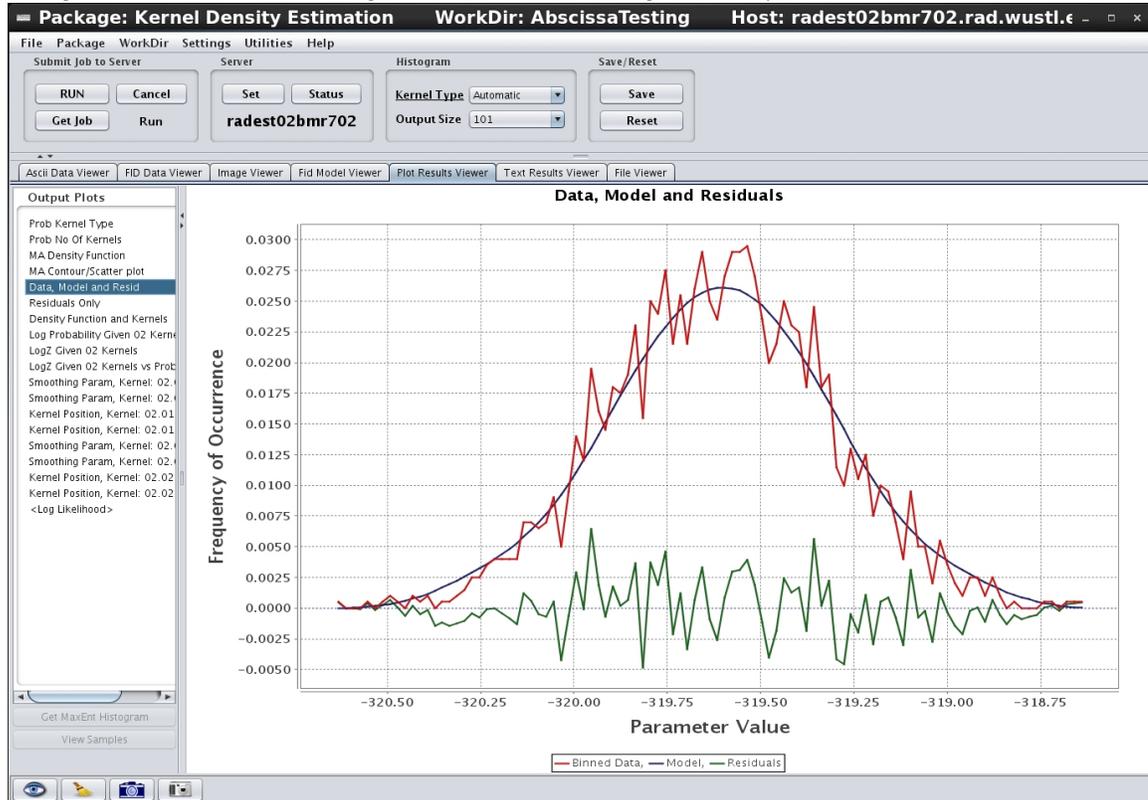


Figure 24.7: Unlike most other packages, in the density estimation packages, there is no model that can be plotted on the data to show how well the model fits the data. This plot is a crude attempt to solve this problem. The red line is a binned histogram. The blue line is the model averaged kernel density function, and the green line is the difference between the binned histogram and the model averaged kernel density function.

the red line in Fig. 24.7. The model averaged density function is the blue line in this figure and the green line is the difference between the binned histogram and the model averaged histogram.

### 24.3 The Bayesian Calculations

The Kernel Density Function Package models an unknown density function  $\rho(x)$  as a superposition of kernels, where  $x$  is the parameter or sample value. For example, if the data are some type of exponentially decaying data, then the  $x$  might be a decay rate constants. It would take on positive values up to some maximum. The program that implements the kernel density function estimation package has nine different kernels programmed into it. They are shown in Fig. 24.1. If we represent a kernel as  $K_i(j_i, x, c_i, h_i)$ , where  $i$  is the kernel number ( $i = 1, 2, \dots, n$ ),  $n$  is the unknown number of kernels needed to represent the density function,  $j_i$  represents one of the nine different kernel

types shown in Fig. 24.1 and discussed in [Wikipedia](#). The parameter  $j$  takes on values 1 through 9, and the subscript is a remainder that each  $K_i$  can be a different kernel type. The position or center of the  $i$ th kernel is designated by  $c_i$ , and  $h_i$  is a smoothness parameter associated with the  $i$ th kernel. To give one a feeling for these parameters, if the kernel type is a Gaussian, then the smoothness parameter would be the standard deviation of the Gaussian and the center would be the mean value of the Gaussian. Given this representation of a kernel, the unknown density function,  $\rho(x)$ , is given by

$$\rho(x) = \frac{1}{Z} \sum_{i=1}^n K_i(j_i, x, c_i, h_i), \quad (24.1)$$

where  $Z$  is a normalization constant. In the program that implements this calculation  $Z$  is calculated discretely. That is to say, the density function is discretized into “Output Size” bins, where “Output Size” is the value of interface “Output Size” widget. The Normalization constant,  $Z$ , is set so that the sum of these discrete samples is one. In the kernel density estimation problem as outlined here, the unknowns are the number, types, and smoothness of kernels used. As indicated, the kernel type is represented by  $j_i$  and is a discrete parameter. The number of kernels  $n$  and is also a discrete parameter, however unlike  $j_i$ , there is only a single  $n$  in Eq. (24.1). Finally, each kernel has two additional parameters associated with it, the center  $c_i$ , and the smoothing parameter  $h_i$ . Defining the list of parameters as  $\Omega \equiv \{n, j_1, \dots, j_n, c_1, \dots, c_n, h_1, \dots, h_n\}$ , the the posterior probability for the  $\Omega$  parameters is represented symbolically by  $P(\Omega|DI)$ , where  $D$  represents all of the sample data. The posterior probability for the  $\Omega$  parameters is given by an application of Bayes’ theorem,

$$P(\Omega|DI) = \frac{P(\Omega|I)P(D|\Omega I)}{P(D|I)} \quad (24.2)$$

where  $P(\Omega|I)$  is the joint prior probability for all of the parameters,  $P(D|\Omega I)$  is the probability for the data given the parameters and the prior information, and the denominator is a normalization constant. The joint prior probability for a parameter will be factored into independent prior probabilities for each parameter

$$P(\Omega|I) = P(n|I) \times P(j_1|I) \cdots P(j_n|I) \times P(c_1|I) \cdots P(c_n|I) \times P(h_1|I) \cdots P(h_n|I). \quad (24.3)$$

The prior probability for the number of kernels,  $P(n|I)$ , was assigned using an Gaussian prior probability given by

$$P(n|I) = \frac{1}{Z_n} \exp \left\{ -\frac{(n-1)^2}{2\sigma_n^2} \right\} \quad n \in (1, 2, \dots, M) \quad (24.4)$$

where  $Z_n$  is the normalization constant from this prior probability,  $M$  is the maximum number of kernels and is hard-coded as  $M = 101$ . The standard deviation,  $\sigma_n$  was set to  $M/10$ . The prior probability for the kernel type,  $P(j_k|I)$ , were all assigned a uniform prior probability, so

$$\prod_{k=1}^n P(j_k|I) = \prod_{k=1}^n \frac{1}{9} = \frac{1}{9^n}. \quad (24.5)$$

Under most conditions such constant priors can be discarded, but that is not the case here because of the  $n$  appearing in this prior probability. The number of kernels needed to represent the density

function,  $n$ , is one of the parameters being inferred, and this prior takes on different values depending on  $n$ . The prior probability for the center of the kernel,  $P(c_k|I)$ , was assigned bounded Gaussian. To do this we peaked at the data and found the  $L$  lowest and  $H$  highest data value, and then assigned the prior probability for the center of the kernels as

$$\prod_{k=1}^n P(c_k|I) = \frac{1}{Z_c^n} \prod_{k=1}^n \exp \left\{ -\frac{(M - c_k)^2}{2\sigma_c^2} \right\} \quad (L \leq c_k \leq H) \quad (24.6)$$

where  $M = (L + H)/2$ , and the standard deviation of this Gaussian was set to one fourth  $\sigma_c = (H - L)/4$ . So this prior suggest, not very strongly, that the kernels are centered in the middle of the data range. Because of the way the program generates samples of  $c_k$ , it is not possible for  $c_k$  to be less then  $L$  or greater then  $H$ . Finally, the prior probability for the smoothness parameters,  $P(h_k|I)$ , was assigned as a prior positive given by

$$\prod_{k=1}^n P(h_k|I) = \frac{1}{Z_h^n} \prod_{k=1}^n \frac{h_k}{\delta^2 + h_k^2} \quad (24.7)$$

where  $U$  is the value in the ‘‘Output Size’’ widget and  $\delta$

$$\delta = \frac{(H - L)}{U} \quad (24.8)$$

is a guess at the size of the smoothing parameter, here is set to the size of the bin necessary to step across the data range in  $U$  steps.  $Z_h$  is the normalization constant for this prior probability. Finally, the probability for the data given the parameters is assigned using the unknown density function  $\rho(x)$ , Eq. (24.1). So

$$P(D|\Omega I) = \prod_{k=1}^N \rho(d_k) = \frac{1}{Z^N} \prod_{k=1}^N \left[ \sum_{i=1}^n K_i(j_i, d_k, c_i, h_i) \right] \quad (24.9)$$

where  $N$  is the total number of data samples  $d_k$ . Substituting Eqs. (24.4,24.5,24.6,24.7,24.9) into Eq. (24.2) one obtains

$$\begin{aligned} P(\Omega|DI) &= \frac{1}{9^n} \\ &\times \frac{1}{Z_n} \exp \left\{ -\frac{(n - 1)^2}{2\sigma_n^2} \right\} \\ &\times \frac{1}{Z_c^n} \prod_{i=1}^n \exp \left\{ -\frac{(M - c_i)^2}{2\sigma_c^2} \right\} \\ &\times \frac{1}{Z_h^n} \prod_{i=1}^n \frac{h_i}{\delta^2 + h_i^2} \\ &\times \frac{1}{Z^N} \prod_{\ell=1}^N \left[ \sum_{i=1}^n K_i(j_i, d_\ell, c_i, h_i) \right] \end{aligned} \quad (24.10)$$

as the posterior probability for the  $\Omega$  parameters. It is this posterior probability that is sampled by the Markov chain Monte Carlo simulation that implements this kernel density estimation calculation.

## 24.4 The Output Reports

The Output reports include the console log, a Condensed file, the probability model file and McMC values report which contains the posterior probability for the kernel types printed out in a graph and tabular form. There is also the listing of the simulation that had maximum posterior probability and finally a small table containing the parameter mean and standard deviation estimates,

The model averaged expected parameter values

Parameter Description	Mean Value	Std. Dev.	Peak Value
LogZ Given 02 Kernels	4.42922E+00	2.12301E-02	4.43049E+00
Smoothing Param, Kernel: 02.01	1.05311E+00	1.16737E-02	1.05245E+00
Kernel Position, Kernel: 02.01	-3.19610E+02	9.05720E-03	-3.19608E+02
Smoothing Param, Kernel: 02.02	6.17517E-01	3.38448E-02	6.19907E-01
Kernel Position, Kernel: 02.02	-3.19599E+02	1.45643E-02	-3.19596E+02

where the notation 02.01 means kernel 01 of 02 kernels. So the first number is the number of kernels, while the second is the kernel currently being described.

## Chapter 25

# MaxEnt Density Function Estimation

The Maximum Entropy Method of Moments package, which we will refer to as the MaxEnt package, uses Bayesian Probability theory to compute the posterior probability for the number of Lagrange multipliers need to represent a density function given a set of samples. The input data to this package are the samples drawn from an unknown density function. For example, the samples generated from the run of a Markov chain Monte Carlo simulation might serve as input. Indeed this is exactly what happens when the “Get MaxEnt Histogram” button is activated. For plotting purposes the samples must be numbered, so the input data are a two column Ascii file.

### 25.1 Using The Package

To use this package, you must do the following:

**Select** the “MaxEnt Histogram” package from the Package menu.

**Load** two column Ascii data using the “Files” menu.

**Set** the number of Lagrange Multipliers to use in the analysis, this can be set to a number or to “Automatic.” The automatic feature will cause the package to compute the posterior probability for the number of Lagrange multipliers.

**Set** the number of bins in the histogram.

**Review** the prior probabilities for package. The MaxEnt Histogram package does not allow the user to set prior probabilities.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server.

**Run** the the analysis on the selected server by activating the Run button.

Figure 25.1: MaxEnt Density Function Estimation Package Interface

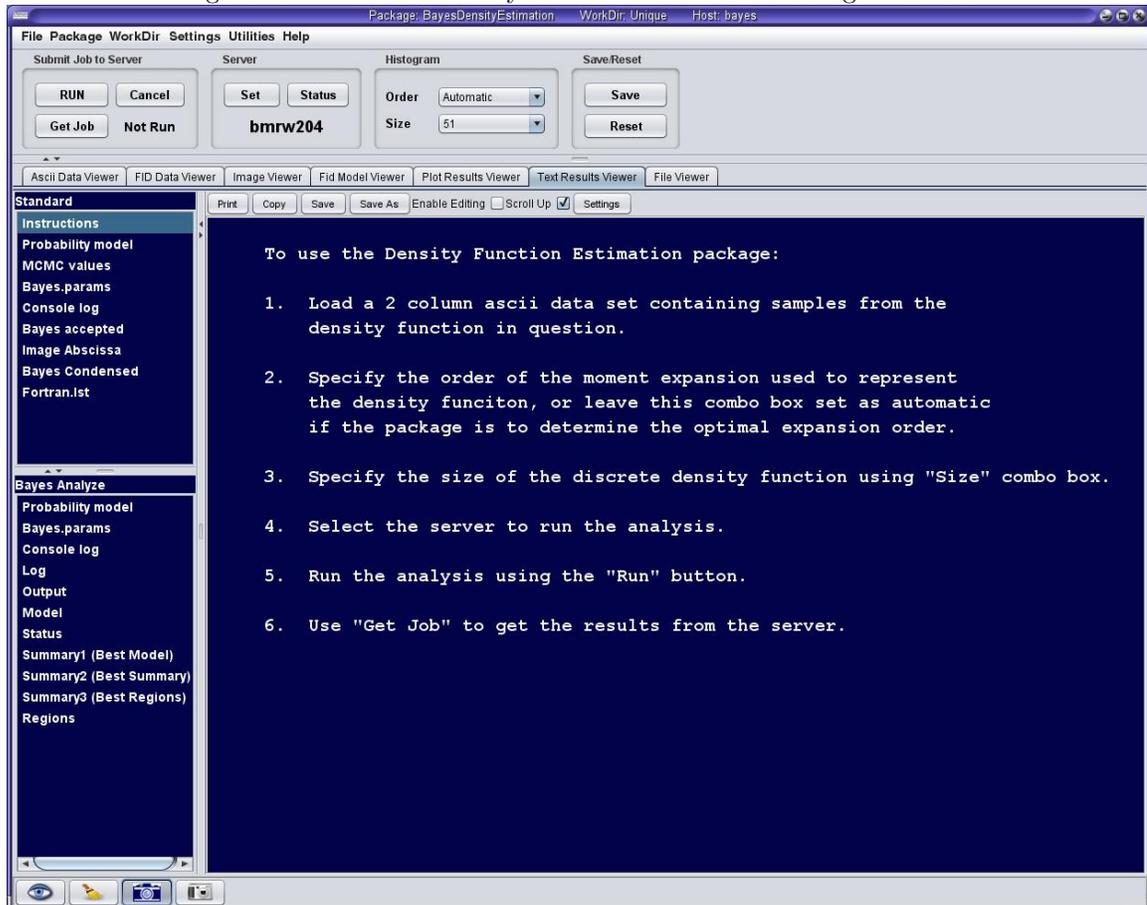


Figure 25.1: This is the interface to the MaxEnt Density Function Estimation package. This package will compute the posterior probability for the Lagrange Multipliers using a Maximum Entropy method of moments calculation with a given or unknown number of multipliers. For more on the actual calculations and the widgets see the text.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

Output from the package consists of an McMC Values report, See Fig. 25.2. The First part of the McMC Values report, not shown, details the parameter settings that were input to the analysis. After the parameter file values, the McMC Values Report has the first few moments of the data. These moments show up in the Bayesian Calculation as sufficient statistics, i.e., the only functions of the data needed to perform the inference calculations. Next there is a bar chart of the posterior probability for the number of multipliers needed to represent the data. This is followed by the Lagrange multipliers that had maximum posterior probability. Finally, the Lagrange multipliers computed as the mean and standard deviations of the Markov chain Monte Carlo

There is also a console log, Figure 25.3 which contains counts of the number of simulations having one, two, etc Lagrange multipliers. This report shows the convergence of the simulations on the number of multipliers. The MaxEnt Density Function Estimation package outputs a running count of the number of simulations having one, two, etc. Lagrange multipliers to the console. This count is the unnormalized posterior probability for the number of Lagrange multipliers as a function of the annealing parameters. When the annealing parameter is small, the simulations distribute themselves according the the prior probability for the number of multipliers, Eq.(25.19) below, which is an  $\exp(-m)$ . So if there are 33 simulations having 0 multipliers, just on the prior one would expect 12 having 1 and about 3 having 2. As one can see, at small values of the annealing parameter, the simulations are distributed according the prior probability for the model. As the annealing parameter increases the data become increasingly important and the by the time the annealing parameters has reached 0.015 the simulations are definitely heading for a model containing 2 multipliers.

In addition to the McMC Values report and the console log, The interface also makes considerable use of Ascii Plot Viewer. The package outputs a number of plots unique to this package. The first of these is the posterior probability for the number of multipliers, Fig. 25.4. The horizontal axis is the number of Lagrange Multipliers and the vertical axis is the probability for indicated number of multipliers. In this problem, this probability was zero everywhere except when the number of multipliers was two and then the probability was one.

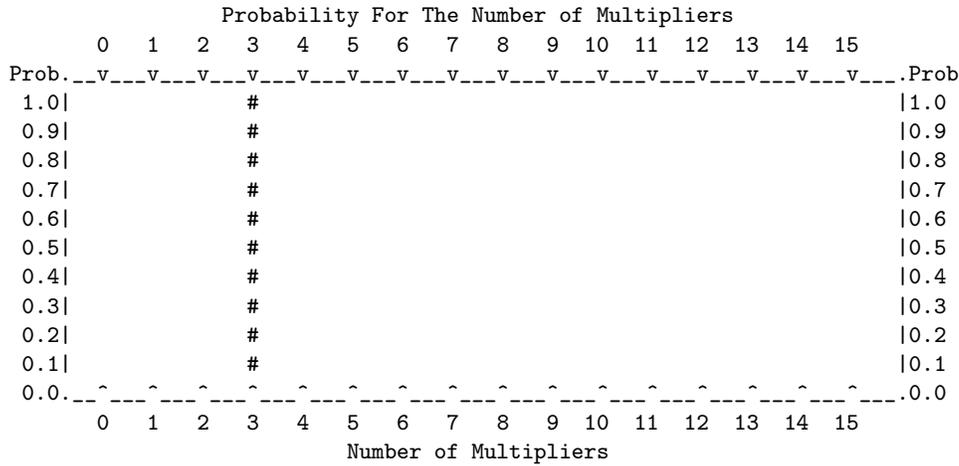
There are three other plots unique to this package, the estimated density function with error bars, Fig 25.12, is discussed in the following sections. In this package the “Data, Model and Residuals” plot is a unique and we will discuss that plot shortly. Finally, a gray scale plot of the density function with the samples also plotted, Fig. 25.5. This scatter plot is meant to illustrate how well the actual samples conform to the inferred Maximum Entropy method of moments density function. The data used in this plot are from the Bayesian Analysis test data, the Histogram subdirectory and the HistFreqW data set. These data samples are from an analysis run using the Bayesian Analysis software and then viewing one of the output histograms and finally activating the “View Samples” button and saving the samples. This particular set of samples is vary Gaussian like. Consequently, there is a bright area in the center of Fig. 25.5 which tapers off symmetrically as you move away from the mean. The horizontal axis is just the repeat simulation number from the Markov chain Monte Carlo simulations. The vertical axis is the parameter value from that simulation.

The “Data, Model and Residual” plot in the Maximum Entropy method of moments package, Fig. 25.6, is unusual for the following reasons: in most packages the data are a continuous function of time and don’t actually change all that much from one point to the next. However, in this package the data are a random sample out of an unknown underlying density function and smooth continuous

Figure 25.2: The MaxEnt Method Of Moments McMC Values Report

## McMC Values Report for the Density Function Estimation Package

Number	Moment	Cumulant
1	-3.19602056E+02	-3.19602056E+02
2	1.02145560E+05	8.56739829E-02
3	-3.26459857E+07	-1.43628567E-04
4	1.04337504E+10	-1.29381267E-04
5	-3.33465926E+12	-3.27148438E-02
6	1.06576843E+15	-2.33284416E+01



Probability for the number of Lagrange Multipliers:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Given 2 Multipliers

The parameters that Maximized the posterior probability are:

Description	Parameter
LogZ given 2	4.83560181E+00
Lagrange Multiplier 1	3.95924945E-01
Lagrange Multiplier 2	-3.82316203E+00

The expected parameter values given a 2 Multiplier Model:

Parameter Description	Mean Value	Std. Dev.	Peak Value
LogZ given 2	4.83756E+00	4.62284E-02	4.83560E+00
Lagrange Multiplier 1	3.96897E-01	7.68460E-02	3.95925E-01
Lagrange Multiplier 2	-3.82703E+00	1.24154E-01	-3.82316E+00

Figure 25.2: The MaxEnt Density Function Estimation Package outputs a standard McMC Values report. The First part of this report, not shown, details the parameter settings that were input to the analysis. After the parameter file values, the McMC Values Report has the first few moments of the data. These moments show up in the Bayesian Calculation as sufficient statistics, i.e., the only functions of the data needed to perform the inference calculations. Next there is a bar chart of the posterior probability for the number of multipliers needed to represent the data. This is followed by the Lagrange multipliers that had maximum posterior probability. Finally, the Lagrange multipliers computed as the mean and standard deviations of the Markov chain Monte Carlo samples.

Figure 25.3: The MaxEnt Density Function Estimation Package Console Log

Phase	Ann1 Parm	<Likelihood>	<StdDevLike>	0	1	2	3	4	5	6	7	8	...
Annealing	45E-06	-1.2847E+04	1.2319E+04	33	11	2	2	0	0	0	0	0	...
2	12E-05	-8.7576E+03	2.9464E+03	40	5	3	0	0	0	0	0	0	...
3	46E-05	-8.0843E+03	1.1540E+03	40	4	2	2	0	0	0	0	0	...
4	0.001	-7.8142E+03	1.3536E+02	45	2	1	0	0	0	0	0	0	...
5	0.008	-7.6652E+03	3.2444E+02	37	1	5	3	2	0	0	0	0	...
6	0.011	-7.4675E+03	4.3092E+02	27	1	13	5	2	0	0	0	0	...
7	0.013	-7.2545E+03	4.1095E+02	14	1	22	4	6	0	0	1	0	...
8	0.015	-7.1117E+03	3.7322E+02	9	0	27	5	5	0	0	1	1	...
9	0.018	-6.9616E+03	2.3343E+02	2	0	35	5	4	0	0	1	1	...
10	0.022	-6.8851E+03	6.3603E+01	0	0	40	5	3	0	0	0	0	...

Figure 25.3: The MaxEnt Density Function Estimation Package outputs a running count to the console. This count is the unnormalized posterior probability for the number of Lagrange multipliers as a function of the annealing parameters. When the annealing parameter is small, the simulations distribute themselves according to the prior probability for the number of multipliers, Eq.(25.19) below, which is an  $\exp(-m)$ . So if there are 33 simulations having 0 multipliers, just on the prior one would expect 12 having 1 and about 3 having 2. As one can see, at small values of the annealing parameter, the simulations are distributed according to the prior probability for the model. As the annealing parameter increases the data become increasingly important and by the time the annealing parameters has reached 0.015 the simulations are definitely heading for a model containing 2 multipliers.

line cannot be drawn through the samples. Consequently, in order to generate a “Data, Model and Residual” plot the data samples were histogrammed using a 51 bin histogram. No smoothing was done on this histogram. It’s just counts divided by the total number of data values. This Histogram is shown in Fig. 25.6 as the red, rather jagged, line. The blue line shown in this plot is the inferred density function generated from the Lagrange multipliers that had maximum posterior probability. The green line is the difference between the binned histogram, red line, and the inferred histogram, blue line. Note this difference is always small at the edges of this plot and they increase as you go to the peak of the inferred histogram.

There is one other unique plot, the inferred density function with error bars and we discuss this plot in the following Section 25.2. Additionally, in that section we will discuss how Bayesian Probability Theory is used to address all of the shortcomings in the Maximum Entropy Method of Moments.

## 25.2 Introduction

The problem of density estimation occurs in many disciplines. For example, in MRI it is often necessary to classify the types of tissues in an image. To perform this classification one must first identify the characteristics of the tissues to be classified. These characteristics might be the intensity of a T1 weighted image and in MRI many other types of characteristic weightings (classifiers) may be generated. In a given tissue type there is no single intensity that characterizes the tissue, rather there is a distribution of intensities. Often this distributions can be characterized by a Gaussian,

Figure 25.4: The Posterior Probability For The Number Of Multipliers

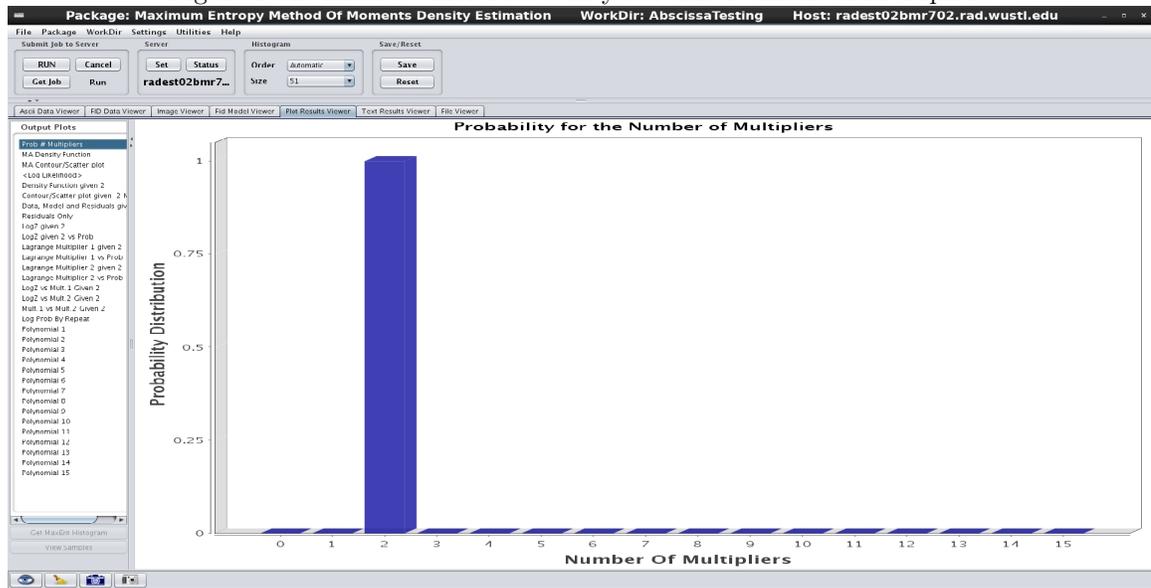


Figure 25.4: The first output generated by the MaxEnt Density Function Estimation package is a plot of the posterior probability for the number of Lagrange Multiplier needed to represent the Maximum Entropy density function. Here that probability indicates that the data are Gaussian, i.e., all probabilities were zero except the 2 Lagrange multiplier model and its probability was one.

Figure 25.5: The Model Averaged Density Function And Samples

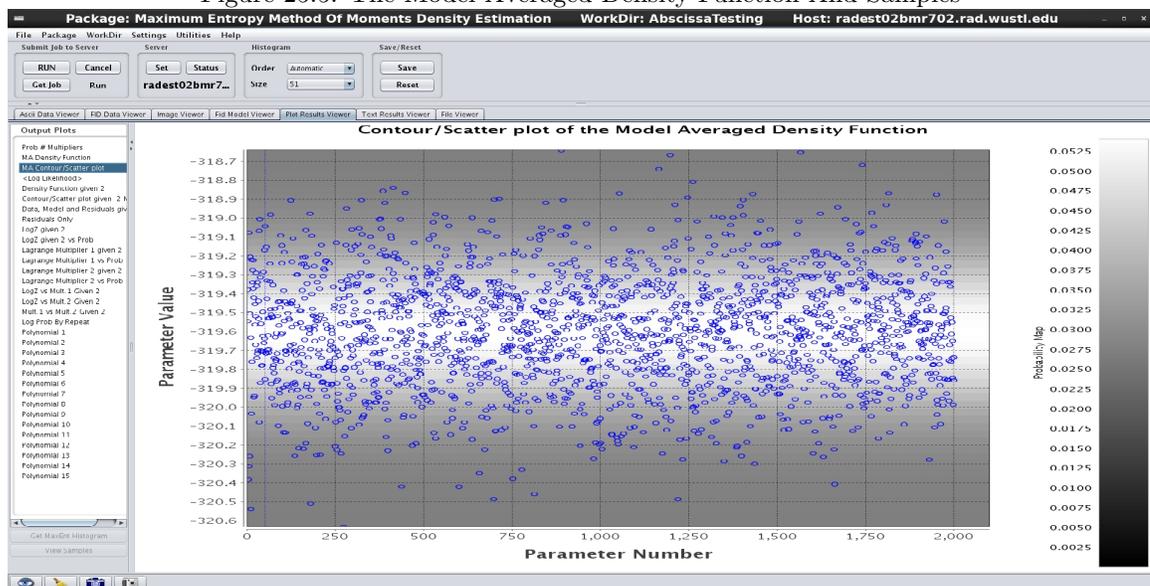


Figure 25.5: After the model averaged density function, the package outputs a gray scale plot of the posterior probability and the samples from the Markov chain Monte Carlo simulations. The horizontal axis is just the repeat simulation number from the Markov chain Monte Carlo simulations. The vertical axis is that simulations corresponding parameter value. This plot is meant as a visual aid in seeing how well the estimated density function and the samples agree.

Figure 25.6: The Data, Model And Residuals

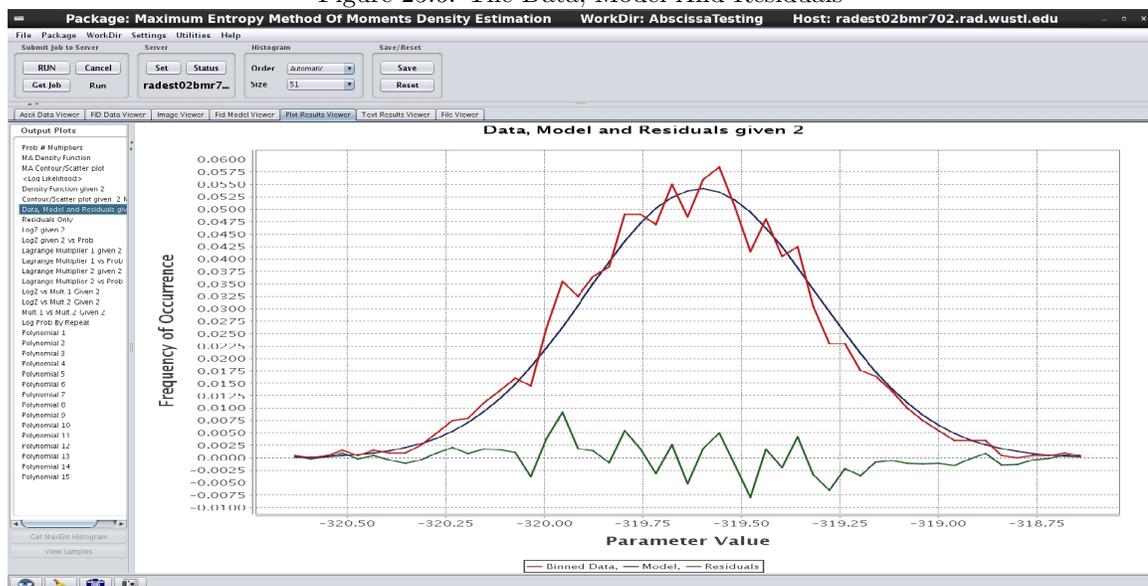


Figure 25.6: In addition to the contour/scatter plot, the Maximum Entropy Method Of Moments package outputs a plot of the data, the model and the residuals. Because the data are a random sample drawn from an unknown underlying density function, we have plotted a binned histogram of the data samples, red line. The blue line is the estimated density function inferred by this package. Finally, the difference between the binned histogram and the estimated density function are shown in green. Note the fit in the wings is always much smaller in the wings and increases until you get to the center of the estimated histogram.

but just as often it is much more complicated. Either way, estimating the distribution of intensities is an inference problem. In the case of a Gaussian distribution, one must estimate the mean and standard deviation. However, in the Non-Gaussian case the shape of the density function itself must be inferred. Three common techniques for estimating density functions are binned histograms [52, 23], kernel density estimation [53, 51], and the maximum entropy method of moments [58, 43]. In the following section, the maximum entropy method of moments will be reviewed. Some of its problems and conditions under which it fails will be discussed. Then in later sections, the functional form of the maximum entropy method of moments probability distribution will be incorporated into Bayesian probability theory. It will be shown that Bayesian probability theory solves all of the problems with the maximum entropy method of moments. One gets posterior probabilities for the Lagrange multipliers, and, finally, one can put error bars on the resulting estimated density function.

In the problem being formulated, one has a data set consisting of samples drawn from an unknown density function. Figure 25.7 displays an illustrative set of such data samples, these data samples (gray circles) were generated in a Markov chain Monte Carlo simulation; although the source of the data samples is unimportant for the problem considered here. The horizontal axis is sample number and the vertical axis is the sample value. There are 2500 samples shown in this figure. The problem is to estimate both the density function and the uncertainty in the estimated density function. Often such data samples can be characterized by a Gaussian density function, but just as often the density function is more complicated. Either way, estimating the distribution of intensities is an inference problem. In the case of the Gaussian, one must estimate both the mean and standard deviation. However, in the Non-Gaussian case, the shape of the density function itself must be inferred. Three common techniques for estimating density functions are binned histograms [52, 23], kernel density estimation [53, 51], and the maximum entropy method of moments [58, 43]. In the following section, the maximum entropy method of moments will be reviewed and some of its problems and conditions under which it fails will be discussed. Then in the following sections, the functional form of the maximum entropy method of moments probability distribution will be incorporated into Bayesian probability theory. Because the resulting Bayesian calculations never solve for the Lagrange multipliers, probability theory never encounters the difficulties involved in solving the maximum entropy method of moments functional equations. As a Bayesian calculation, one gets posterior probabilities for both the number and values of the Lagrange multipliers as well as error bars on the resulting density function. The solid line in Fig. 25.7 is an example of the estimated density function with error bars generated using the techniques and procedures described in this paper.

### 25.3 Review of The Maximum Entropy Method Of Moments

Claude Shannon [58] derived the Shannon entropy as a measure of the information content of a discrete probability distribution. If this discrete probability distribution is represented by  $f_j$ , then the Shannon entropy,  $S$ , is given by

$$S = - \sum_{j=1}^n f_j \log(f_j) \quad (0 \leq f_j \leq 1) \tag{25.1}$$

where  $n$  is the number of discrete probabilities in the distribution. The entropy  $S$  is a measure of the information content of a probability distribution. It reaches its maximum value when all  $f_j = 1/n$

Figure 25.7: Density Function Sample Data

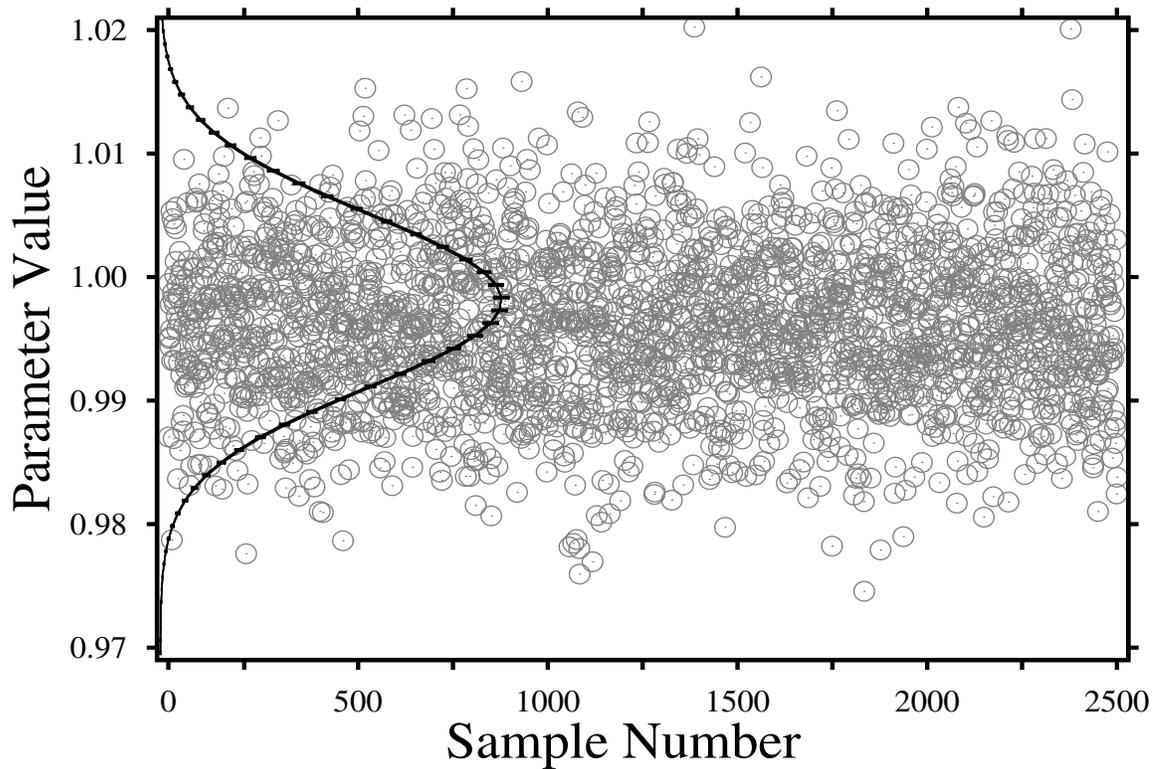


Figure 25.7: In the density estimation problem addressed here, one as a set of samples (open circles) drawn from some unknown density function and one wishes to infer the distribution of the samples (solid line with error bars). This density function was estimated using Bayesian probability theory to determine what probabilities must be assigned. The maximum entropy method of moments was then used to assign the indicated probabilities. Finally, a Markov chain Monte Carlo simulation was used to draw samples from the posterior probability for the density function, see the text for the details.

and  $S = \log(n)$ , and it reaches its minimum value when one of the  $f_j = 1$ , and then  $S = 0$ . Thus the Shannon entropy maps discrete probability distributions onto the interval  $0 \leq S \leq \log(n)$ , with  $S = \log(n)$  the completely uninformative state, and  $S = 0$  the state of certainty. Everything in between represents increasing knowledge for decreasing entropy.

After deriving the entropy function, Shannon proceeded to use the entropy function as a way of assigning maximally uninformative probability distributions that are consistent with some given prior information. In the maximum entropy method of moments, the Shannon entropy is constrained by the power moments. Suppose the probabilities  $f_j$  are defined on a set of discrete points  $x_j$ . In this case, the expected value of the power moments is given by

$$\langle x^k \rangle = \sum_{j=1}^n x_j^k f_j \quad (k = 0, 1, \dots, m) \quad (25.2)$$

where  $k = 0$  is the normalization constraint. Because this is an equality, one can move the sum to the left-hand side of the equation, and because this equation is equal to zero one can multiply through by a constant, called a Lagrange multiplier, and the equation will still be zero:

$$\lambda_k \left[ \langle x^k \rangle - \sum_{j=1}^n x_j^k f_j \right] = 0. \quad (25.3)$$

Additionally, if one has more than one constraint, one can sum over the constraints and the sum is still zero:

$$\sum_{k=0}^m \lambda_k \left[ \langle x^k \rangle - \sum_{j=1}^n x_j^k f_j \right] = 0. \quad (25.4)$$

Because this equation is zero, it can be added to the Shannon entropy without changing its value:

$$S = - \sum_{j=1}^n f_j \log(f_j) + \sum_{k=0}^m \lambda_k \left[ \langle x^k \rangle - \sum_{j=1}^n x_j^k f_j \right]. \quad (25.5)$$

To assign numerical values to the  $f_j$ , Eq. (25.5) is maximized with respect to variations in the  $f_i$ . The resulting equations can be solved for the functional form of the probability. Taking the derivative with respect to  $f_i$  and solving, one obtains:

$$f_i = Z(m, \lambda)^{-1} \exp \left\{ \sum_{k=1}^m \lambda_k x_i^k \right\} \quad (25.6)$$

where  $Z(m, \lambda)$  is a normalization constant that is a function of both the number of Lagrange multipliers and their values. This equations gives the functional form of the maximum entropy method of moments probability distribution in terms of the Lagrange multipliers  $\lambda_j$ , but one must also satisfy the constraints, namely:

$$\langle x^k \rangle = \sum_{i=1}^n x_i^k f_i \quad (k = 1, \dots, m). \quad (25.7)$$

Equations (25.6) and (25.7) are a system of coupled nonlinear equations for the Lagrange multipliers. To solve for the values of the Lagrange multipliers that maximize the entropy, one typically uses

Figure 25.8: The First 10 Power And Central Moments

Moment	Power	Central
1	9.96127964E-01	0.00000000E+00
2	9.92317063E-01	4.61424576E-05
3	9.88566722E-01	1.95337184E-08
4	9.84876378E-01	6.64319953E-09
5	9.81245478E-01	1.06175263E-11
6	9.77673479E-01	1.64171891E-12
7	9.74159848E-01	4.09097233E-15
8	9.70704063E-01	5.63247606E-16
9	9.67305611E-01	1.15656817E-18
10	9.63963991E-01	2.38918537E-19

Figure 25.8: The first 10 power and central moments computed from the samples shown in Fig. 25.7.

a Newton-Raphson [50] searching algorithm. This searching algorithm Taylor expands Eq. (25.5) about the current estimated values of the Lagrange multipliers to second order, and then solves for the values of the change in Lagrange multipliers that makes the derivatives go to zero. The procedure must be iterated a few times and, when it converges, it typically converges quadratically.

To make this more concrete, suppose one computes the first 10 moments, of the samples shown in Fig. 25.7 and uses them in a maximum entropy method of moments calculation. What would happen to the maximum entropy distribution as more and more moments are incorporated into the calculation? The first 10 moments are shown in Fig. 25.8. Two sets of moments are shown, the power moments and the central moments. The power moments are given by

$$\langle \text{Power Moment } k \rangle = \frac{1}{N} \sum_{i=1}^N d_i^k \quad (25.8)$$

and the central moments are given by

$$\langle \text{Central Moment } k \rangle = \frac{1}{N} \sum_{i=1}^N (d_i - \bar{d})^k \quad (25.9)$$

where  $N$  is the total number of data values and  $\bar{d}$  is the mean data value.

If one incorporates the power moments one at a time into a maximum entropy method of moments calculation, the distributions shown in Fig. 25.9 result. The flat line marked with inverted triangles is the zeroth moment, i.e., a uniform distribution. The tilted line marked with closed triangles is an exponential distribution, and because the samples are far from exponentially distributed, this distribution is almost a uniform distribution. Finally, the remaining curves (solid unmarked lines) are the maximum entropy method of moments distributions corresponding to power moments two through seven. Note that these distributions are nearly identical, differing only near the peak values.

In Fig. 25.8, a total of 10 moments were given, however, in Fig. 25.9 only eight maximum entropy method of moment distributions are shown, corresponding to  $k = 0, 1, \dots, 7$ . The reason for this is that the numerical calculation failed to converge when more than the 7 nontrivial moments

Figure 25.9: Maximum Entropy Distributions As A Function Of The Number of Multipliers

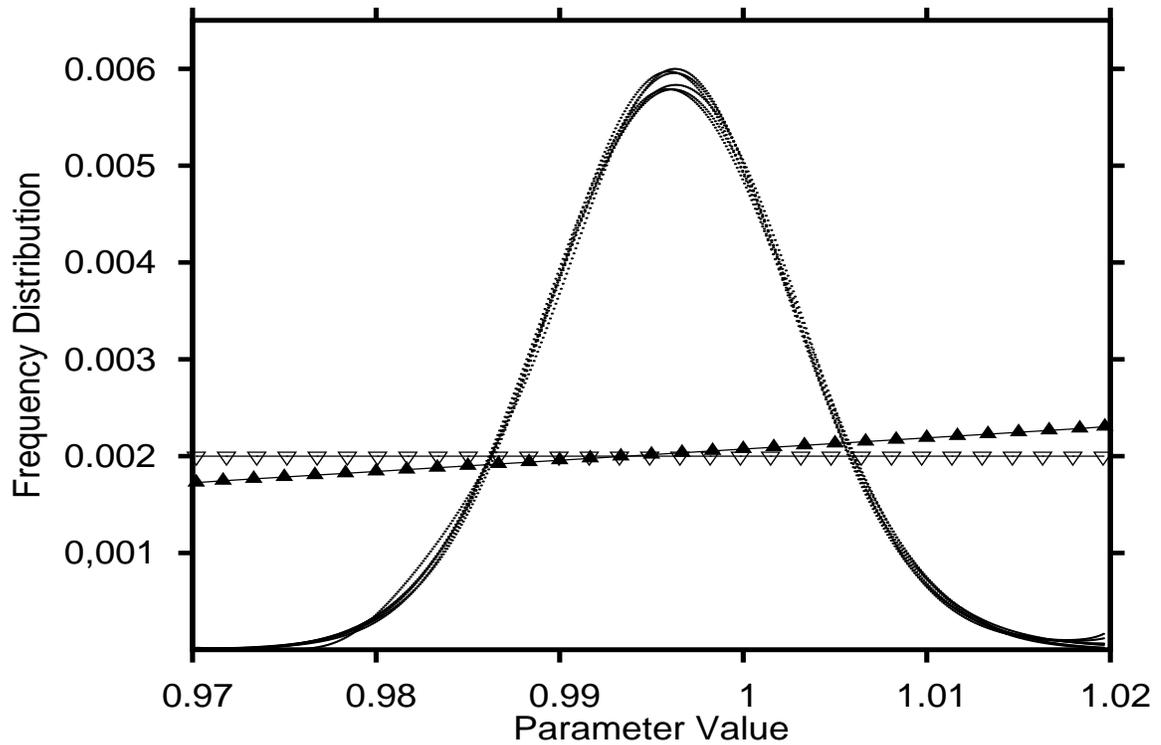


Figure 25.9: The maximum entropy moment distributions as a function of increasing numbers of moments. The flat line with open triangles is a normalized uniform density resulting from using only the zeroth moment. When the first moment is incorporated (line with closed triangles), not much changes because an exponential distribution cannot represent the distribution of samples shown in Fig. 25.7. However, for second and higher moments (dotted unmarked lines) all of the maximum entropy method of moments distributions closely resemble each other with only minor variations. Only density functions corresponding to moments zero through 7 are shown, the numerical algorithm failed to converge for power moments greater than 7.

were incorporated into the calculation. This is typical of the numerical calculations used in solving maximum entropy method of moments problems. Above some number of moments, the searching algorithm fails to converge or the numerical values of the moments were incompatible and no maximum entropy solution exists, see Meed and Papanicolaou [43] for the conditions under which the maximum entropy method of moments can fail.

This completes this review of the maximum entropy method of moments. Here is a short list of some of the problems with this technique:

1. The maximum entropy method of moments did not use the data samples shown in Fig. 25.7; rather one must compute a number of moments from the samples and use these moments in the calculations. From a Bayesian standpoint this is a rather ad hoc thing to do and has no justification whatsoever. By its very nature, Bayesian probability theory uses the raw data; not the moments, and if the moments are needed, they will show up automatically, they won't have to be artificially forced into the problem.
2. There is no way to determine how many moments, Lagrange multipliers, are needed. That is to say, the maximum entropy method of moments has an arbitrary component to it: one must guess the number of moments, or simply continue adding moments until the procedure fails.
3. There is no way to consistently find the maximum entropy method of moments solution. From a finite data sample, the moments can be mutually incompatible and, consequently, no solution may exist [43]. And even if the maximum entropy solution exists, searching algorithms such as Newton-Raphson, which is commonly used on this problem [43, 50], may not be able to find it.
4. There is no way to put error bars on the Lagrange multipliers. Because the maximum entropy method of moments picks out an extremum, the question of putting error bars on the Lagrange multipliers almost does not make sense. After all, maximum entropy picks out a single point. Nonetheless, from a Bayesian perspective, for a finite amount of data one should be able to put error bars on the multipliers, or, better yet, compute the posterior probability for the Lagrange multipliers given the data and the prior information.
5. The same comments apply to the assigned density function. Because the maximum entropy method of moments picks out a single value, there is no way to determine how uncertain one is of the estimated density function. As far as maximum entropy is concerned, there is only a single density function. But from a Bayesian standpoint, this is simply false. From a finite sample, probability theory would never pick out a single density function; rather, probability theory will indicate a range of values that the density function could take on that are consistent with the available data and prior information.

When the maximum entropy method of moments works, it gives a good representation of the underlying density function that quickly converges as a function of the number of constraints (moments). However, the maximum entropy method of moments is not a Bayesian techniques: it does not use the raw data, there is no way to determine how uncertain one is of the resulting density function, and it is not uncommon for the maximum entropy method of moments to fail because the set of moments are incompatible. A true Bayesian calculation does none of these things. It would always give a results in terms of the calculated posterior probability distributions for the number and value of the Lagrange multipliers, and it would do this even if the calculated moments are incompatible.

## 25.4 The Bayesian Calculations

To resolve these difficulties, Bayesian probability theory will be applied to compute the posterior probability for the number of Lagrange multipliers. The posterior probability for the number of multipliers  $m$  given all of the data  $D$  is computed using Bayes' theorem [1]:

$$P(m|DI) = \frac{P(m|I)P(D|mI)}{P(D|I)} \quad (25.10)$$

where  $P(m|I)$  is the prior probability for the number of Lagrange multiples,  $P(D|mI)$  is a marginal direct probability for the data given the number of multipliers. Finally,  $P(D|I)$  is a normalization constant and is computed using the sum and product rules of probability theory:

$$P(D|I) = \sum_{m=1}^{\nu} P(Dm|I) = \sum_{m=1}^{\nu} P(m|I)P(D|mI) \quad (25.11)$$

where  $\nu$  is some given upper limit on the number of Lagrange multipliers.

In Eq. (25.10), the Lagrange multipliers do not appear. Consequently, Eq. (25.10) is a marginal posterior probability where the Lagrange multipliers have been removed from the right-hand side using the sum rule of probability theory:

$$P(m|DI) \propto P(m|I) \int P(D\lambda_1 \cdots \lambda_m|mI)d\lambda_1 \cdots d\lambda_m \quad (25.12)$$

where  $P(D\lambda_1 \cdots \lambda_m|mI)$  is the joint probability for all of the data  $D \equiv \{d_1, \dots, d_N\}$  and the Lagrange multipliers given the number of multipliers  $m$  and the prior information  $I$ . Note that the normalization constant has been dropped, the equal sign has been replaced by a proportionality sign, and this probability distribution must be normalized at the end of the calculation. Applying the product rule to the right-hand side of this equation results in:

$$P(m|DI) \propto P(m|I) \int P(\lambda_1 \cdots \lambda_m|mI)P(D|m\lambda_1 \cdots \lambda_m I)d\lambda_1 \cdots d\lambda_m. \quad (25.13)$$

Assuming logical independence of the data samples, the right-hand side of this equation can be factored:

$$P(m|DI) \propto P(m|I) \int P(\lambda_1 \cdots \lambda_m|mI) \prod_{i=1}^N P(d_i|m\lambda_1 \cdots \lambda_m I)d\lambda_1 \cdots d\lambda_m. \quad (25.14)$$

Finally, assuming logical independence of the Lagrange multipliers,  $P(\lambda_1 \cdots \lambda_m|mI)$  may also be factored to obtain

$$P(m|DI) \propto P(m|I) \int \left[ \prod_{j=1}^m P(\lambda_j|mI) \right] \left[ \prod_{i=1}^N P(d_i|m\lambda_1 \cdots \lambda_m I) \right] d\lambda_1 \cdots d\lambda_m. \quad (25.15)$$

The direct probability for the data given the number of Lagrange multipliers and their values,  $P(d_i|m\lambda_1 \cdots \lambda_m I)$ , is the maximum entropy method of moments probability given in Eq. (25.6).

Substituting Eq. (25.6) into Eq. (25.15) one obtains

$$P(m|DI) \propto P(m|I) \int \left[ \prod_{j=1}^m P(\lambda_j|mI) \right] \left[ \prod_{i=1}^N \frac{1}{Z(m, \lambda)} \exp \left\{ \sum_{k=1}^m \lambda_k d_i^k \right\} \right] d\lambda_1 \cdots d\lambda_m \quad (25.16)$$

as the posterior probability for the number of Lagrange multipliers given the data and the prior information. Expanding the products gives

$$P(m|DI) \propto P(m|I) \int \frac{P(\lambda_1|I) \cdots P(\lambda_m|I)}{Z(m, \lambda)^N} \exp \left\{ \sum_{i=1}^N \sum_{k=1}^m \lambda_k d_i^k \right\} d\lambda_1 \cdots d\lambda_m, \quad (25.17)$$

and evaluating the sum over the data values results in

$$P(m|DI) \propto P(m|I) \int \frac{P(\lambda_1|I) \cdots P(\lambda_m|I)}{Z(m, \lambda)^N} \exp \left\{ \sum_{k=1}^m \lambda_k N \bar{d}^k \right\} d\lambda_1 \cdots d\lambda_m \quad (25.18)$$

where the  $\bar{d}^k$  are the power moments of the samples defined in Eq. (25.7).

The functional form of Eq. (25.18) is interesting in several ways. First, the data do not appear in this equation, rather there are  $m$  power moments of the data. These power moments are called sufficient statistics. They are sufficient in that they are the only quantities needed for the inference; the data itself are irrelevant. Only maximum entropy distributions have sufficient statistics. In this case, the constraint functions are simple polynomials,  $x^k$ , so the sufficient statistics are the power moments calculated using the data samples. Second, every term in the sum in Eq. (25.18) is of the form  $\lambda_k N \bar{d}^k$ , which can always be driven to infinity by choosing  $\lambda_k$  suitably. So one might think that this could not possibly be a well-behaved probability density function. However, this is not the case, because this is a fully normalized probability density function and the normalization constant is a function of both the number of Lagrange multipliers and their values. Any attempt to drive the exponent to infinity simply results in a larger normalization constant that keeps everything finite.

The only remaining steps in the calculation are to assign the prior probabilities appearing in Eq. (25.18) and to perform the indicated calculations. In the numerical calculations that are done, all probability assignments are discretely normalized to ensure that one has probability distributions, not density functions. Probability density functions can be larger than one, and because of the functional form of the posterior probability, Eq. (25.18), this is not allowed. The prior probabilities were assigned as follows. The prior probability for the number of multipliers,  $P(m|I)$ , was assigned using an exponential prior probability:

$$P(m|I) \propto \frac{1}{Z_m(\nu)} \exp \{-m\} \quad (1 \leq m \leq \nu) \quad (25.19)$$

where the  $\nu$  is the upper limit on the number of moments and expresses a belief that the number of multipliers should be small, rather than large. The normalization constant  $Z_m(\nu)$  was computed as

$$Z_m(\nu) = \sum_{m=1}^{\nu} \exp \{-m\} = \frac{1 - e^{-\nu}}{e - 1} \quad (25.20)$$

and ensures that the prior probability for the number of Lagrange multipliers is normalized and always less than one.

The prior probability for each Lagrange multiplier was assigned using a Gaussian of the form:

$$P(\lambda_j|I) \propto \frac{1}{Z_{\lambda_j}} \exp \left\{ -\frac{\lambda_j^2}{2\sigma_\lambda^2} \right\} \quad (\lambda_{\text{Min}} \leq \lambda_j \leq \lambda_{\text{Max}}) \quad (25.21)$$

where  $\sigma_\lambda$  is the standard deviation of this Gaussian,  $\lambda_{\text{Min}}$  is the smallest value the Lagrange multipliers can take on,  $\lambda_{\text{Max}}$  is the largest, and  $Z_{\lambda_j}$  is the normalization constant for the prior probability for the  $j$ th Lagrange multiplier. The standard deviation,  $\sigma_\lambda$ , was set so that the prior decayed to 7  $e$ -foldings at  $\lambda_{\text{Min}}$  and  $\lambda_{\text{Max}}$ . This prior probability distribution was normalized discretely. To compute the normalization constant, the prior range was divided into 500 intervals and then summed. In this sum, the  $k$ th discrete value of the  $j$ th Lagrange multiplier is given by

$$\lambda_{jk} = \lambda_{\text{Min}} + d\lambda(k-1) \quad (1 \leq k \leq 501) \quad (25.22)$$

with

$$d\lambda = \frac{(\lambda_{\text{Max}} - \lambda_{\text{Min}})}{500}. \quad (25.23)$$

The normalization constant was computed as

$$Z_{\lambda_j} = \sum_{k=1}^{501} \exp \left\{ -\frac{\lambda_{jk}^2}{2\sigma_\lambda^2} \right\}. \quad (25.24)$$

It is this normalization constant that is used in Eq. (25.21).

The last normalization constant that must be set is  $Z(m, \lambda)$ , the normalization constant associated with the maximum entropy method of moments probability density function. Again, this probability density function was discretely normalized so that a probability distribution was actually used in the numerical calculations. Thus, all values computed using Eq. (25.6) will strictly be probabilities, not probability densities. The normalization constant is computed using the range of the data samples. If the minimum and maximum data value are represented by  $d_{\text{Min}}$  and  $d_{\text{Max}}$  respectively, then

$$x_i = d_{\text{Min}} + dx(k-1) \quad (1 \leq k \leq 501) \quad (25.25)$$

with

$$dx = \frac{(d_{\text{Max}} - d_{\text{Min}})}{500} \quad (25.26)$$

and

$$Z(m, \lambda) = \sum_{i=1}^{501} \exp \left\{ \sum_{k=1}^m \lambda_k x_i^k \right\}. \quad (25.27)$$

Again, this normalization constant ensures that Eq. (25.6) is a probability distribution and sums to one on the  $x_i$ . The posterior probability for  $m$  is obtained by substituting Eqs. (25.19, 25.21 and 25.27) into Eq (25.18) to obtain:

$$P(m|DI) \propto \int \frac{\exp\{-m\}}{Z_m Z_\lambda^m Z(m, \lambda)^N} \exp \left\{ -\sum_{j=1}^m \frac{\lambda_j^2}{2\sigma_\lambda^2} \right\} \exp \left\{ \sum_{k=1}^m \lambda_k N \bar{d}^k \right\} d\lambda \quad (25.28)$$

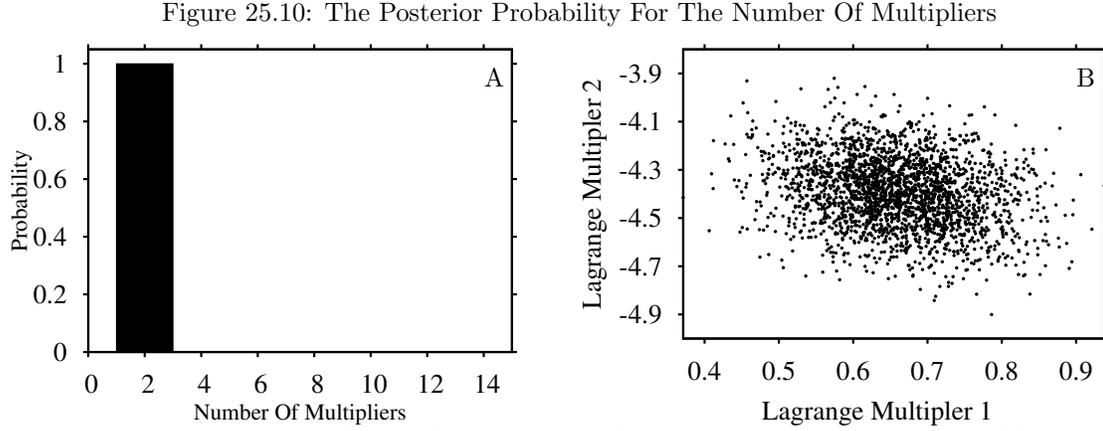


Figure 25.10: The posterior probability for the number of multipliers, Panel A, was computed from Markov chain Monte Carlo samples using the data set shown in Fig. 25.7. This discrete probability distribution is zero everywhere except when the number of Lagrange multipliers is two, and then the probability is one, indicating that these samples are Gaussian. After determining the number of Lagrange multipliers, the posterior probability for the two multipliers was sampled, Panel B. From these samples one can obtain Markov chain Monte Carlo samples for each Lagrange multiplier.

where  $d\lambda$  means the integral over all  $m$  Lagrange multipliers. Equation (25.28) is the posterior probability for the number of Lagrange multipliers.

In addition to computing the posterior probability for the number of Lagrange multipliers, the posterior probability for  $\lambda_j$  given the number of Lagrange multipliers and the data is also needed. However, this calculation is so similar to the one just given that it will not be repeated. Rather, note that the integrand of Eq. (25.28) is the joint posterior probability for all of the parameters,  $P(m\lambda_1 \cdots \lambda_m | DI)$ , and can be used to generate the posterior probability for any one of the Lagrange multipliers by applying the sum rule of probability theory:

$$\begin{aligned}
 P(\lambda_j | mDI) &\propto \int \frac{1}{Z_\lambda^m Z(m, \lambda)^N} \exp \left\{ -\sum_{j=1}^m \frac{\lambda_j^2}{2\sigma_\lambda^2} \right\} \\
 &\times \exp \left\{ \sum_{k=1}^m \lambda_k N \bar{d}^k \right\} d\lambda_1 \cdots d\lambda_{j-1} d\lambda_{j+1} \cdots d\lambda_m.
 \end{aligned} \tag{25.29}$$

To arrive at this result, eliminated the prior probability for the number of Lagrange multipliers, since this probability is a constant when  $m$  is given. Additionally, all the Lagrange multipliers, except  $\lambda_j$ , were removed using marginalization. This results in the posterior probability for the single remaining Lagrange multiplier,  $\lambda_j$ .

A Markov chain Monte Carlo simulation with simulated annealing was used to draw samples from the integrand of Eq. (25.28) using the data shown in Fig. 25.7. In a typical run, 50 simulations are run simultaneously and in parallel, and 50 samples from each simulation are gathered, so there are 2500 total Markov chain Monte Carlo samples for the number of multipliers and their values. Monte Carlo integration was then used to compute the posterior probability for the number of Lagrange

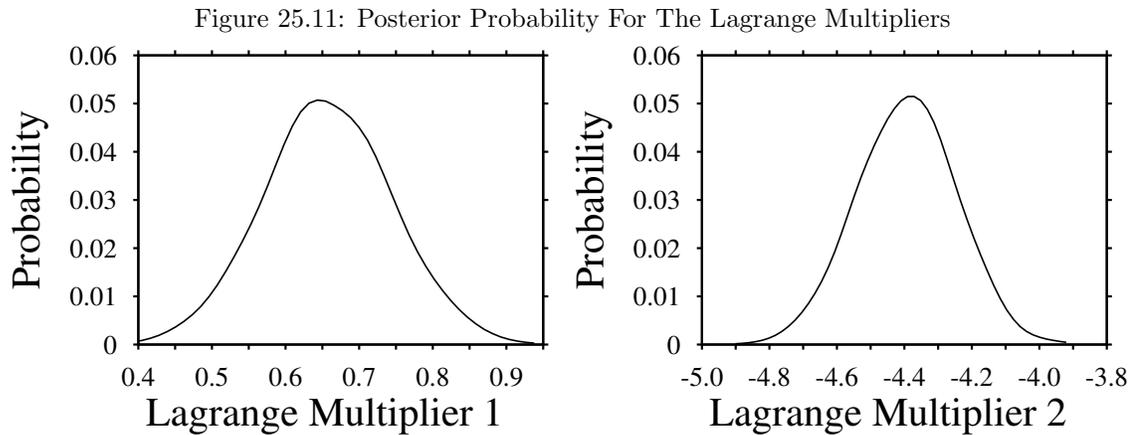


Figure 25.11: The posterior probabilities for the first two Lagrange multipliers. Lagrange multiplier 1 is estimated to be about  $0.65 \pm 0.08$  and multiplier number 2 is approximately  $-4.4 \pm 0.14$ .

multipliers given the data and the prior information. The posterior probability for the number of multipliers is shown in Fig. 25.10(A). Note that this posterior probability indicates that only two Lagrange multipliers are needed to represent the density distribution of the data. Consequently, Bayesian probability theory strongly indicates that the data shown in Fig. 25.7 are Gaussianly distributed.

After determining that the number of Lagrange multipliers was two, the joint posterior probability for the two Lagrange multipliers was sampled. These Markov chain Monte Carlo samples are shown in Fig. 25.10(B). Each dot in this figure is one sample from one of the 2500 Markov chain Monte Carlo simulations. By using Monte Carlo integration, one can obtain samples from the posterior probability for each Lagrange multiplier, Fig. 25.11. These one dimensional samples can be used to compute mean and standard deviation estimates of the Lagrange multipliers. However, a means of visually displaying the samples is also desirable. A binned histogram could be used, but even with 2500 samples such histograms are often very rough. Consequently, the program that implements this calculation uses a Gaussian kernel density estimation procedure to generate its histograms.

The 51 bin histograms shown in Fig. 25.11 were generated using a Gaussian kernel that decays to 3 e-foldings over 6 bins. This kernel was centered on each Markov chain Monte Carlo sample and then added to the histogram by evaluating the kernel at each value of the histogram's x-axis. As a consequence, each of the 2500 samples was smeared out over a 6 bin interval using the Gaussian kernel. Finally, the normalization is set so that the sum over the 51 bins was one. As can be seen from this figure, Lagrange multiplier 1 is estimated to be approximately  $0.65 \pm 0.08$  and multiplier number 2 is approximately  $-4.4 \pm 0.14$ . Note that these probability density functions for the Lagrange multipliers are not very compact, and the standard deviation of these probability density functions are 0.08 and 0.14, respectively. Given that there are about 2000 data values and the data are noiseless, this is not a very good determination. Thus, while maximum entropy gives one Lagrange multiplier for each moment, it does not indicate how uncertain one is of these values and the uncertainty in the value of the Lagrange multipliers can be large. In the example given here, they have a relative uncertainty of about 20% for multiplier 1 and about 7% for multiplier 2.

Each of the 2500 Markov chain Monte Carlo samples of the Lagrange multipliers shown in

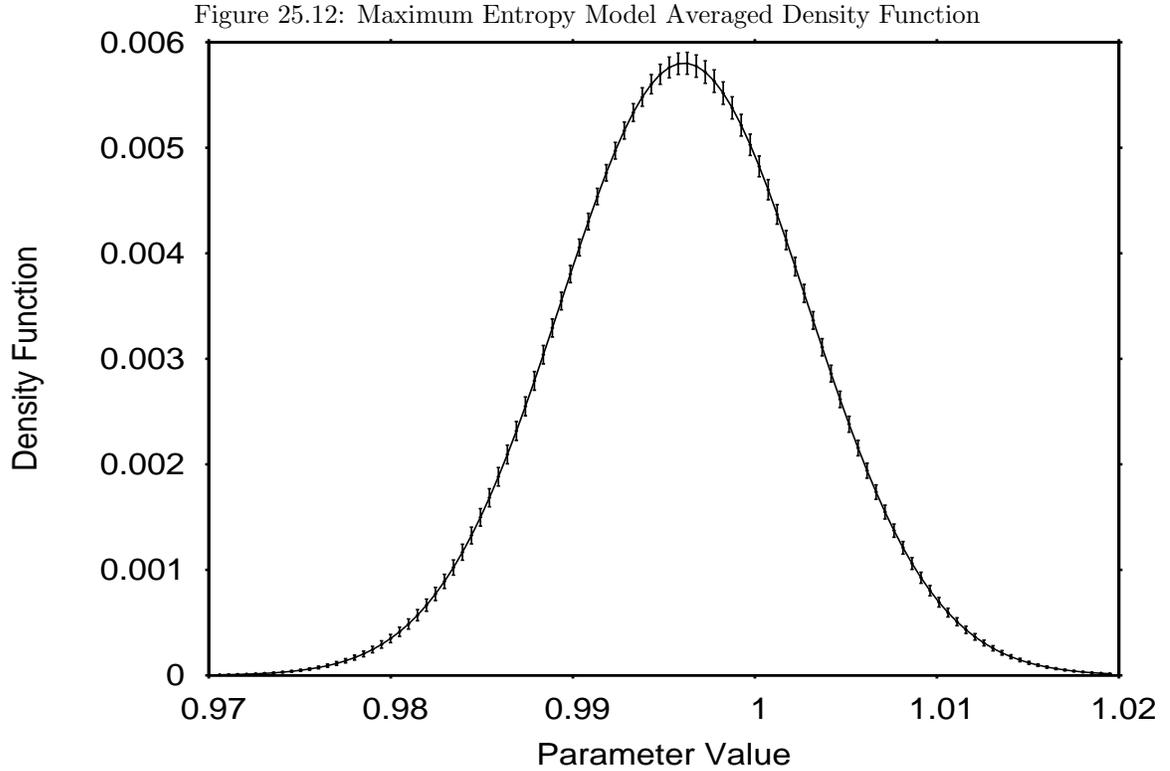


Figure 25.12: The model averaged density function with error bars. A Markov chain Monte Carlo simulation was used to draw samples from the joint posterior probability for the number of multipliers and their values. A total of 2500 samples were drawn. Each sample corresponds to a density function that is consistent with the data and the prior information. The solid line in this plot is the mean value of the 2500 density function estimates and the error bars are the standard deviation of the estimates.

Fig. 25.10(B) corresponds to a density function estimate that is consistent with the given data and prior information. One can use the Lagrange multiplier samples to compute the unknown density function. For example, one could compute the density function at the values specified by Eq. (25.25). For each  $x_i$ , there are 2500 samples of the density function. For a given  $x_i$ , one can compute the mean and standard deviation. This mean and standard deviation are shown in Fig. 25.12. At each point in this plot, the mean is the solid line and the standard deviation is shown as the error bar. These error bars are a direct measure of the amount of uncertainty in the Lagrange multipliers and thus directly reflect the uncertainty in the underlying density function.

## 25.5 Summary and Conclusions

The maximum entropy method of moments is fraught with difficulties. It is computationally unstable. One cannot use the raw data; rather one must compute an unknown number of moments using the data and then use those moments in the maximum entropy method of moments. There is

no way to determine how many moments are needed and, finally, there is no way to determine how uncertain one is of the estimated density function.

However, if one uses the maximum entropy formalism to assign the probability for the data given both the number of moments and the value of the Lagrange multipliers, then maximum entropy will assign Eq. (25.6) as the functional form of the probability distribution. One can then use the rules of Bayesian probability theory to compute the posterior probability for the parameters, including the number of Lagrange multipliers. Because the Bayesian calculations are all computed using a forward calculation, i.e., given the values of the parameters, compute a probability, and never attempt to solve for the values of the multipliers that satisfy the constraints, Eq. (25.7), one never runs into computational difficulties. Additionally, the final results are all expressed as probability distributions, so one always knows how uncertain one is of all of the parameters. Finally, because the calculations are implemented using a Markov chain Monte Carlo simulation, one has samples from the joint posterior probability for all of the parameters appearing in Eq. (25.28). These samples can be used to form a mean and standard deviation estimate of each point in the unknown density function, thus putting error bars on the unknown density functions value.



## Chapter 26

# Phasing An Image

MRI Images present special problems for most data processing algorithms because of the use of the absolute value after the k-space data are Fourier transformed. When the absolute value is taken, the noise and the signal get multiplied. Unless the signal-to-noise in the data is very high, this cross-term can cause big problems in processing the data. However, this problem is eliminated if an absorption mode image is used because then the Fourier transform is a linear operator and if the noise was Gaussian in k-space, it remains Gaussian in the image domain. An example of an absorption mode image is shown in Fig. 26.1 as the left panel. The right panel is the same image in absolute value mode. Note that in the absorption model image, outside the brain the noise oscillates around zero and close inspection of the images will reveal that the absorption mode image is sharper than the absolute value image. The effect is not as pronounced in images as it is in spectroscopic applications because the echos do not decay appreciably in the time needed to acquire them. Nonetheless sharper images and eliminating the noise offsets are two very strong reasons to use absorption mode images. In this Chapter we describe the Bayesian calculations needed to create an absorption mode images.

The image phasing package, Bayes Phase, estimates three phase parameters. These phase parameters may be estimated from one image and then applied to all images in an array, they may be determined for each image separately, or they may be determined for one image. In the all cases the output from the package is a series of FDF and Ascii files that may then be displayed in Vnmr, VnmrJ or they may be viewed using image browser. Additionally, these output phased images may be used to generate input data for other packages, and they may be analyzed in total to generate images of various parameter maps that are output from these other packages. For example, after a set of images have been phased individual voxel intensities may be imported into the diffusion tensor analysis and then analyzed. Or the images could be input to the Image Pixel package and then an image of the diffusion constants could be generated.

The calculations presented in this Chapter describe the imaging model and then present three separate Bayesian calculations: one for the constant phase, and then two identical calculations for the positionally dependent phase shift in each of the two spatial domains. The program that implements this calculation can processes spin echo, gradient echo and EPI images. In the case of EPI images four phase parameters are needed to phase an image, in the directly detected domain an even and odd time delay are needed, additionally, one time delay is needed in the indirectly detected domain and finally one constant phase is needed. The calculations for these four phase parameters are exactly identical to the calculations for the spin echo and gradient echo phase parameters and

Figure 26.1: Absorption Model Images

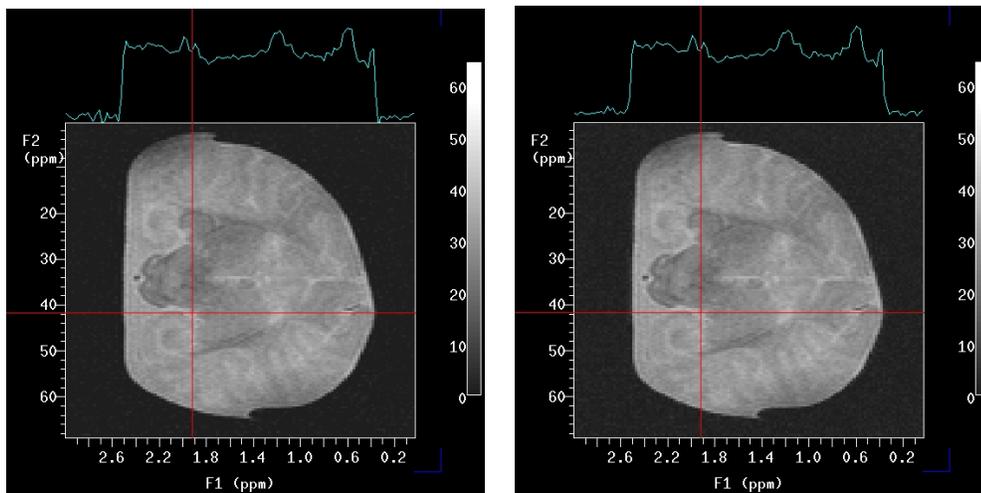


Figure 26.1: The left panel is an example of an absorption mode image while the right panel is the same image in absolute value. A single trace has been displayed on both images. Note that in the absorption image outside the brain the noise oscillates around zero and the image comes down faster on the brain boundary.

we will not have much more to say about EPI images except to note how to analyze them.

The Bayes Phase package is accessed by selecting the “Phase An Image” button on the dispatching menu. When this button is activated the interface window shown in Fig. 26.2 is displayed. The upper panel in this figure is the heart of the VnmrJ interface while the lower panel is the Vnmr interface. Both interfaces set a number of control parameters and then allow one to run the phasing algorithm.

## 26.1 The Bayesian Calculation

There are three phase parameters that must be determined to produce an absorption mode image: a constant phase  $\theta$ , and two time delays which we will designate as  $\tau_x$  and  $\tau_y$ . These time delays may also be thought of as the center of the echo in the k-space, and they are analogous to the frequency dependent phase in spectroscopic measurements. In spectroscopic application, frequency dependent phase shifts are typically small. Indeed it is rare to find spectroscopic data that have frequency dependent shifts that cause more than one or two phase wraps. However, in imaging  $\tau_x$  and  $\tau_y$  are huge and typically cause  $180^\circ$  phase wraps every few points in the Fourier transform. Indeed these phase wraps are so big, that in the image domain the data look as if it has a periodic signal in it, as indeed it does.

To estimate these three phase parameters, one must relate these parameters to the data through a model. If we expand the spin density in sinc functions in the spatial domain, then in the k-space

Figure 26.2: The Interface To The Image Phasing Package

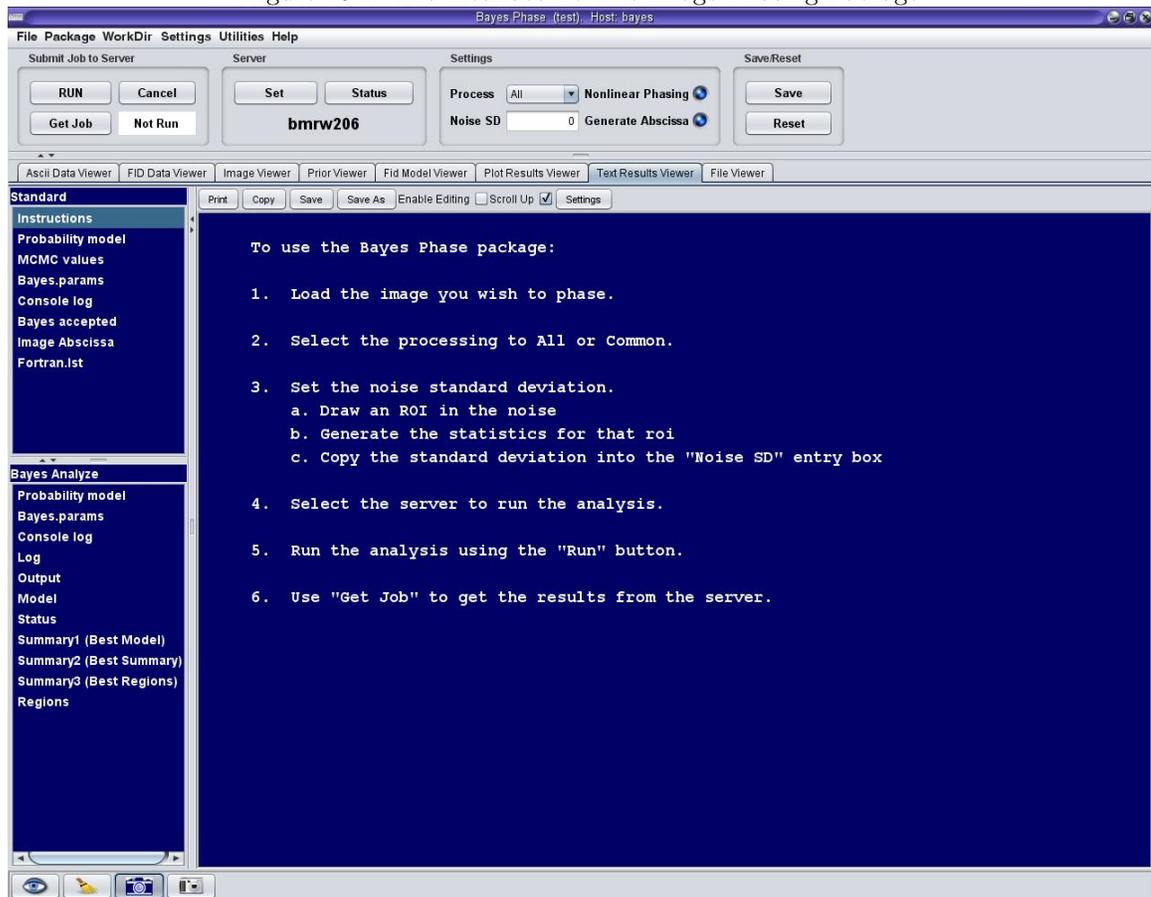


Figure 26.2: The interface to the Linear Phasing package is shown here. The Linear Phasing package outputs a series of FDF files that contain the real and imaginary parts of the phased images. These images may then be used as input to other packages. For example the are often used by the Analyze Image Pixel package.

domain the expansion is a Fourier series:

$$d_{ij} = \exp\{-i\theta\} \sum_{k=1}^{N_x} \sum_{l=1}^{N_y} A_{kl} \exp\{-2\pi i(x_k(t_{xi} + \tau_x) - 2\pi i(y_l(t_{yj} + \tau_y)))\} + \text{noise} \quad (26.1)$$

where we have designated the complex data as  $d_{ij}$ ,  $N_x$  and  $N_y$  are the number of complex data values in the  $x$  and  $y$  domains, and the intensity of the spin density function at position  $x_k$  and  $y_l$  has been designated as  $A_{kl}$ . We have intentionally written this model in a way that explicitly shows that the two sums over the  $k$ -space data are time shifted inverse Fourier transforms. Note that as written the phase parameters  $\tau_x$  and  $\tau_y$  do participate in the sums. However the Bayesian calculations are time domain calculations, and in these calculations the sums will be over  $i$  and  $j$  and we will find that  $\tau_x$  and  $\tau_y$  do not participate in the sums. As a result, the Bayesian calculations may be done using fast discrete Fourier transform when  $N_x$  and  $N_y$  are powers of 2.

We will start this process by estimating  $\tau_x$ . If we are only interested in  $\tau_x$ , the value of both  $\tau_y$  and  $\theta$  are irrelevant to us. For the purposes of estimating  $\tau_x$  we note that the imaging experiment just increments the value of  $y$  by a constant for each  $k$ -space acquisition. Effectively this just changes the constant phase of each new  $k$ -space acquisition. Consequently, for the purposes of estimating  $\tau_x$  we will rewrite Eq. (26.1) as

$$d_{ij} = \exp\{-i\theta_j\} \sum_{k=1}^{N_x} B_{kj} \exp\{-2\pi i(x_k(t_{xi} + \tau_x))\} + \text{noise} \quad (26.2)$$

where  $B_{kj}$  are the amplitudes of the image in the  $j$ th  $k$ -space acquisition. Similarly, the phase  $\theta_j$  is the constant phase in the  $j$ th  $k$ -space acquisition. Both of these quantities are related to the  $A_{kl}$  and  $\theta$  through a complicated sum. Fortunately, we don't care about these expressions for estimating  $\tau_x$ . Separating this model into its real and imaginary parts one has

$$d_{Rij} = \sum_{k=1}^{N_x} B_{kj} M_{Rki} + \text{noise} \quad (26.3)$$

for the real data, and

$$d_{Iij} = \sum_{k=1}^{N_x} -B_{kj} M_{Iki} + \text{noise} \quad (26.4)$$

for the quadrature data where

$$M_{Rki} \equiv \cos(\theta) \cos(2\pi x_k[t_{xi} + \tau_x]) + \sin(\theta) \sin(2\pi x_k[t_{xi} + \tau_x]) \quad (26.5)$$

and

$$M_{Iki} \equiv \cos(\theta) \sin(2\pi x_k[t_{xi} + \tau_x]) - \sin(\theta) \cos(2\pi x_k[t_{xi} + \tau_x]). \quad (26.6)$$

In this model the data may be thought of as  $N_y$  different data sets each of them bearing on the value of  $\tau_x$ . If each data set contributes independent information about  $\tau_x$ , then the posterior probability will just be the product of the probabilities for  $\tau_x$  in each data set separately. Consequently, the marginal posterior probability for  $\tau_x$  can be factored to obtain

$$P(\tau_x|DI) = \int d\sigma \prod_{j=1}^{N_y} \int dB_{1j} \dots dB_{N_x j} d\theta_j P(B_{1j} \dots B_{N_x j} \theta_j \sigma | D_j I) \quad (26.7)$$

where  $D_j$  is just the data for the  $j$ th k-space acquisition.

The right-hand side of this equation is factored using Bayes' theorem to obtain:

$$P(\tau_x|DI) \propto \int d\sigma \prod_{j=1}^{N_y} \int dB_{1j} \dots dB_{N_xj} d\theta_j P(B_{1j} \dots B_{N_xj} \theta_j \sigma | I) P(D_j | B_{1j} \dots B_{N_xj} \theta_j \sigma I). \quad (26.8)$$

Finally, the joint prior probability for the parameters is factored using the product rule to obtain

$$P(\tau_x|DI) \propto \int d\sigma P(\sigma|I) \prod_{j=1}^{N_y} \int dB_{1j} \dots dB_{N_xj} d\theta_j \times P(\theta_j|I) P(B_{1j} \dots B_{N_xj}|I) P(D_j|B_{1j} \dots B_{N_xj} \theta_j \sigma I) \quad (26.9)$$

where we have not factored the prior probability for the amplitudes,  $P(B_{1j} \dots B_{N_xj}|I)$ , into independent prior probabilities because we are going to assign a correlated prior to the amplitudes. That is to say we are going to take into account the fact that images tend to be smoothly varying and that adjacent voxels tend to be very nearly equal.

We have now reached the point in the Bayesian calculation where one has no choice but to assign a numerical value to represent each of these probabilities. The prior probability for the noise standard deviation,  $P(\sigma|I)$ , will be assigned a Jeffreys' prior

$$P(\sigma|I) \propto \frac{1}{\sigma}. \quad (26.10)$$

The prior probability for the phase,  $P(\theta_j|I)$ , will be assigned a uniform prior probability and this prior will restrict the integration over the phase to zero to  $2\pi$ .

In assigning the prior probability for the amplitudes we wish to take into account the fact that adjacent amplitudes tend to be nearly equal. Of course, there are always exceptions to this, but nonetheless, in this analysis we are going to put in a prior that will try and make adjacent voxels equal. Here is how this is done. If  $B_{kj} \approx B_{k+1j}$  then

$$B_{kj} \approx B_{k+1j} \Rightarrow B_{kj} - B_{k+1j} \approx 0 \Rightarrow \sum_{k=1}^{N_x-1} (B_{kj} - B_{k+1j})^2 \text{ is small.} \quad (26.11)$$

If the principle of Maximum Entropy is used to assign a prior probability that imposes this condition, Maximum Entropy will lead to a Gaussian assignment for the prior. This Gaussian will be written as

$$P(B_{1j} \dots B_{N_xj}|I) \propto \left(\frac{\sigma}{\beta}\right)^{-N_x} |U_{kl}|^{-\frac{1}{2}} \exp \left\{ - \sum_{k=1}^{N_x} \sum_{l=1}^{N_x} \frac{B_{lj} \beta^2 U_{kl} B_{kj}}{2\sigma^2} \right\} \quad (26.12)$$

where the matrix  $U_{kl}$  is a tri-diagonal matrix having [-1, 2,-1] as its three non-zero diagonals and  $\beta$  expresses how strongly we believe adjacent voxels should be equal. In the program that implements this calculation  $\beta = 0.1$ , so the prior says that we think small oscillations, on the order of 0.01 of the maximum signal value are probably noise. Note we are using this condition only in the calculation of  $\tau_x$ , we do not use this condition in generating the final images. This condition is equivalent to imposing a smoothness constraint on the first derivative of the image and because the Fourier transform is symmetric this prior imposes what is often referred to as a circular boundary condition.

If we assign the likelihood using a Gaussian, the joint posterior probability for  $\tau_x$ , Eq. (26.9), is given by:

$$P(\tau_x|DI) \propto \int \frac{d\sigma}{\sigma} \prod_{j=1}^{N_y} \int d\theta_j dB_{1j} \dots dB_{N_x j} \sigma^{-3N_x} \exp \left\{ -\frac{Q_j}{2\sigma^2} \right\} \quad (26.13)$$

where we have dropped some constants that cancel when this distribution is normalized. The quantity  $Q_j$  is given by

$$Q_j \equiv \sum_{k=1}^{N_x} \sum_{l=1}^{N_x} B_{lj} \beta^2 U_{kl} B_{kj} + \sum_{i=1}^{N_x} \left( d_{Rij} - \sum_{k=1}^{N_x} B_{kj} M_{Rki} \right)^2 + \left( d_{Iij} + \sum_{k=1}^{N_x} B_{kj} M_{Iki} \right)^2 \quad (26.14)$$

and, up to the term from the prior probability for the amplitudes, is Chi-squared evaluated for each of the k-space data sets. If we substitute the definitions of  $M_{Rki}$  and  $M_{Iki}$ , Eqs. (26.5 and 26.6) respectively then we obtain:

$$Q_j \equiv N_x \overline{d_{xj}^2} - 2 \sum_{i=1}^{N_x} B_{ij} [\cos \theta F_{Rij} + \sin \theta F_{Iij}] + \sum_{k=1}^{N_x} \sum_{l=1}^{N_x} B_{kj} B_{lj} V_{klj} \quad (26.15)$$

with

$$V_{klj} \equiv N_x \delta_{kl} + \beta^2 U_{kl}, \quad (26.16)$$

where  $\delta_{kl}$  is a delta function,

$$\overline{d_{xj}^2} \equiv \frac{1}{N_x} \sum_{i=1}^{N_x} d_{ij}^2 \quad (26.17)$$

is the mean-square data value in the  $j$ th k-space acquisition. The projections of the data onto the model,

$$F_{Rij} \equiv \sum_{i=1}^{N_x} d_{Rij} \cos(2\pi x_k [t_{xi} + \tau_x]) - d_{Iij} \sin(2\pi x_k [t_{xi} + \tau_x]) \quad (26.18)$$

and

$$F_{Iij} \equiv \sum_{i=1}^{N_x} d_{Rij} \sin(2\pi x_k [t_{xi} + \tau_x]) + d_{Iij} \cos(2\pi x_k [t_{xi} + \tau_x]), \quad (26.19)$$

are essentially the real and imaginary parts of a time shifted discrete Fourier transform. While we have not separated the time delays from the other parts of the Fourier transform, a simple trigonometric identity will reduce these quantities to linear combinations of the real and imaginary parts of the discrete Fourier transform.

The functional form of  $Q_j$  is a quadratic in the  $B_{kj}$ , so the integrals over the  $B_{kj}$  are Gaussian quadrature integrals. Such integrals are easily evaluated and we only give the results here, one obtains

$$P(\tau_x|DI) \propto \int \frac{d\sigma}{\sigma} \prod_{j=1}^{N_y} |V_{klj}|^{-\frac{1}{2}} \int d\theta_j \sigma^{-2N_x} \exp \left\{ -\frac{N_x \overline{d_{xj}^2} - \sum_{i=1}^{N_x} \hat{B}_{ij} T_{ij}}{2\sigma^2} \right\} \quad (26.20)$$

where

$$T_{ij} \equiv \cos \theta F_{Rij} + \sin \theta F_{Iij} \quad (26.21)$$

and

$$\hat{B}_{ij} = \cos \theta \hat{a}_{ij} + \sin \theta \hat{b}_{ij} \quad (26.22)$$

with

$$\hat{a}_{ij} = V_{ikj}^{-1} F_{Rkj} \quad \text{and} \quad \hat{b}_{ij} = V_{ikj}^{-1} F_{Ikj}. \quad (26.23)$$

The quantities  $\hat{a}_{ij}$  and  $\hat{b}_{ij}$  are essentially the real and imaginary parts of the discrete Fourier transform, while  $\hat{B}_{ij}$  is the expected amplitude of the signal in the phased image.

The integral over the phase is tedious and not very illuminating, and we only sketch how this integral is evaluated. One begins by taking the sufficient statistic, the sum in Eq. (26.20), and substitutes the definitions of  $T_{ij}$  and  $\hat{B}_{ij}$ . This results in a quadratic expression in  $\cos \theta$  and  $\sin \theta$ . These quadratics are then reduced to  $\sin(2\theta)$  and  $\cos(2\theta)$  using trigonometric identities. The resulting expression may then be rewritten in terms of  $\cos(2\theta + \psi)$ , where  $\psi$  is a phase. In this form the integral is of the form  $\exp\{\cos(\phi)\}$  which is the integral representation of the  $I_0$  Bessel function, one obtains

$$P(\tau_x|DI) \propto \int \frac{d\sigma}{\sigma} \prod_{j=1}^{N_y} |V_{klj}|^{-\frac{1}{2}} \sigma^{-2N_x} \exp \left\{ -\frac{N_x \overline{d_{xj}^2} - \frac{1}{2} \sum_{i=1}^{N_x} (\hat{a}_{ij} F_{Rij} + \hat{b}_{ij} F_{Iij})}{2\sigma^2} \right\} I_0 \left( \frac{\sqrt{W_j^2 + X_j^2}}{2\sigma^2} \right) \quad (26.24)$$

with

$$W_j = \sum_{i=1}^{N_x} \frac{\hat{a}_{ij} F_{Rij} - \hat{b}_{ij} F_{Iij}}{2} \quad (26.25)$$

and

$$X_j = \sum_{i=1}^{N_x} \frac{\hat{a}_{ij} F_{Iij} + \hat{b}_{ij} F_{Rij}}{2}. \quad (26.26)$$

We note in passing that the quantity

$$\psi_j = -\frac{1}{2} \tan^{-1} \left( \frac{X_j}{W_j} \right) \quad (26.27)$$

is the estimated constant part of the phase for each of the k-space acquisitions. We mention this because in the full calculation, a quantity almost identical to this will appear as the estimated constant phase for the entire data set.

In this form the integral over the standard deviation of the noise prior probability,  $\sigma$ , is not easily represented in closed form. Fortunately, there is a simple easy approximation that is good to many decimal places around the maximum in Eq.(26.24). For large argument the  $I_0$  Bessel function is nearly exponential, then Eq.(26.24) is very nearly equal to

$$P(\tau_x|DI) \approx \int \frac{d\sigma}{\sigma} \prod_{j=1}^{N_y} |V_{klj}|^{-\frac{1}{2}} \sigma^{-2N_x} \exp \left\{ -\frac{N_x \overline{d_{xj}^2} - \frac{1}{2} \sum_{i=1}^{N_x} (\hat{a}_{ij} F_{Rij} + \hat{b}_{ij} F_{Iij}) - \sqrt{W_j^2 + X_j^2}}{2\sigma^2} \right\} \quad (26.28)$$

and the integral over the standard deviation may be transformed into a gamma function and we omit the details of evaluating this integral, one obtains

$$P(\tau_x|DI) \propto \prod_{j=1}^{N_y} |V_{klj}|^{-\frac{1}{2}} \left[ N_x \overline{d_{xj}^2} - \frac{1}{2} \sum_{i=1}^{N_x} (\hat{a}_{ij} F_{Rij} + \hat{b}_{ij} F_{Iij}) - \sqrt{W_j^2 + X_j^2} \right]^{-N_x}. \quad (26.29)$$

This probability density function is of the form of Student's  $t$ -distribution, and it is this  $t$ -distribution that is computed in the phasing algorithm.

In addition to estimating  $\tau_x$  one also needs to compute the posterior probability for  $\tau_y$ . However, all one needs to do is to exchange the role of  $x$  and  $y$  and in the above equations to obtain  $P(\tau_y|DI)$ . Consequently, we do not give this calculation. Finally, one needs to compute the posterior probability for  $P(\theta|DI)$ , but we already noted that the calculation is essentially identical to Eq. (26.27). Indeed all that needs to be done is to replace the sums over  $x$  by a sum over  $x$  and  $y$  and then Eq. (26.27) will give the expected value of the phase.

So here is how the calculation is actually implemented. One first computes the fast discrete Fourier transform and uses these projections to compute posterior probability for  $\tau_x$  on a coarse grid. In dimensionless units  $\tau_x$  varies from  $N_x/4 \leq \tau_x \leq 3N_x/4$ . Outside this range the posterior probability is aliased and no additional information is available. After finding the location of the peak on this coarse grid, the algorithm does a binary search for the maximum posterior probability estimate of  $\tau_x$ . Then using the estimated value of  $\tau_x$  the positionally dependent phase is unwrapped in the  $x$  domain. This calculation is then repeated in the  $y$  domain and the phase is again unwrapped. The constant phase is then computed. However, there is an ambiguity in the constant phase. If the calculated value of the constant phase is  $\Theta$ , then the phase that gives positive amplitudes could be  $\Theta$  or  $\Theta + 180^\circ$ . Before setting the constant phase the program does a quick calculation to determine which phase is appropriate and finally the constant part of the phase is unwrapped. After all of the phases have been set, the program outputs the phased images as PDF files. These PDF files are what are displayed in VNMR.

## 26.2 Using The Package

To use the phasing package begin by loading an image. This may be done using the Vnmr files menu or you may use the **Load An Image** on the window, see Fig. 26.2. In VnmrJ the corresponding function is done on the housekeeping folder using the CWD file menu. When an image is loaded under Vnmr, the macros test to see if the image has been previously analyzed. If it has and the current setting of the parameters are the same as when the images were phase the run indicator is turned on and the package is set run. You may rerun the images at any time but assuming the previous settings of the variables are OK, there is no need to do this.

After loading an image, specify whether the image is a spin echo or EPI image. This is done using the **Image Type** menu. Here the term spin echo means only that the image may be phased using three phase parameters, so gradient echo images should be selected as spin echo. While EPI means that 4 phase parameters are needed to phase an image: the constant phase,  $\tau_y$  and an even and odd delay,  $\tau_{ex}$  and  $\tau_{ox}$ , in  $x$ .

Next indicate how the images are to be processed using the **process** menu. The choices are All, Common or One, where "All" means that each image is to be phased using parameters specific to that image. "Common" means that phase parameters are to be computed from the currently

Figure 26.3: Linear Phasing Package The Console Log

Array	Slice	Delay X	Delay Y	Phase
33	1	66.37122551	47.85130018	285.95947785
1	1	66.18384758	47.96787733	308.70159650
34	1	66.20887199	47.97886366	304.08678779
2	1	66.20948234	47.96055311	306.64428485
35	1	66.35840813	47.91508192	292.95579385
3	1	66.21192375	47.97764295	306.10371565
36	1	66.26502434	47.90165418	294.32695098
4	1	66.23938957	47.95567030	305.71194166
37	1	66.24671379	48.01442248	302.01416187
5	1	66.29920402	47.96909803	302.93402804
38	1	66.24671379	47.94590467	300.11338126
6	1	66.26075188	47.96177381	303.08474711
39	1	66.34314934	48.01747424	299.16341694
7	1	66.20398918	47.96299451	303.84929962
40	1	66.30774895	47.87754530	292.07343083
8	1	66.20765129	47.97398084	304.35076867
41	1	66.43714348	47.64805311	266.41805910
9	1	66.20368400	47.96421522	305.24927395
42	1	66.44019523	47.96787733	292.28161251

Figure 26.3: The Phasing routine does write the value of the phase parameter to the mcmc.values file. The exact format of this file varies somewhat between spin echo images and EPI, EPI images have a fourth column: the even and odd  $\tau_x$  value.

displayed image and then those phase parameters are to be applied to every image. Finally, “One” means to compute the phase parameters for the currently displayed image.

There are a number of widgets on the interface that are used to control the display and used to set the image sizes. The entry boxes labeled “fn” and “fn1” are used to enter the sizes of the Fourier transforms. If these sizes differ from the “np” and 2 times “nv” the program does the calculations using “np” and “nv” sizes and then computes and phases the final images at the “fn” and “fn1” values.

Finally, the entry boxes **cf**, **Display Array Element** and **Display** many be used to control which image is being displayed. In all cases if the phasing algorithm has been run, then the phased image is displayed, otherwise the image is displayed in absolute value mode. Changing “cf” will cause different slices to be displayed. Similarly, changing “Display Array Element” will display an image from the new array element. Finally, changing “Display” from Real to Imaginary will cause the imaginary part of the image to be displayed. Note this last widget does nothing if the phase algorithm has not be run.

The phasing routine does write the phases to the “mcmc.values” file located in the BayesOther-Analysis directory in the current experiment. An example of this file is shown in Fig. 26.3 The phasing algorithm does use multiple threads, but not to the extent that most other algorithms do. In the case of the phasing algorithm if multiple images are to be phased each image is dispatched

to a separate thread to run. This is indicated in the output list because the order of the “array” index is mixed up. This output is simply written as each thread completes phasing an image, so the order can get mixed up. Note that in the case of the image processed to produce this figure the delay in both  $x$  and  $y$  was very stable, while the constant phase did vary a little. However, even with this variation it would have been possible to phase this image using a single common set of phase parameters.

## Chapter 27

# Phasing An Image Using Non-Linear Phases

The phasing algorithm presented in Chapter 26 can phase any NMR image in which the phases vary linearly in both domains. Consequently that phasing algorithm works well for spin echo and EPI images, but it does not work for gradient echos because in gradient echos the phase varies non-linearly. This effect is illustrated in Fig. 27.1 panels (A) and (B). The linear phasing algorithm has removed most of the oscillations, but has left behind a slowly varying phase that causes the signal to oscillate between the real and imaginary channels. In Panel (C) and (D) we have displayed the real and imaginary images after the nonlinear phasing routine has been run. The nonlinear phasing routine has moved all of the image from the imaginary changes into the real channel and left behind what appears to be white noise. Indeed if you compute the standard deviation of the noise from a region that contains no signal in the real and imaginary images, you will find they are nearly identical. In this chapter we describe the interface to the nonlinear phasing package, BayesPhase2, and give the calculations needed to phase a gradient echo.

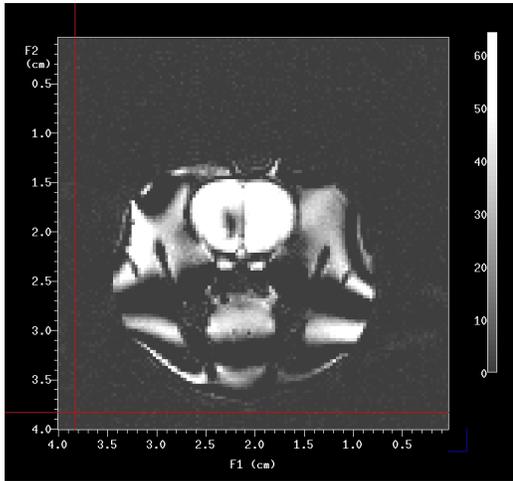
### 27.1 The Model Equation

As in all Bayesian calculations, the calculations begin by relating the parameters of interest to the data, i.e., by stating the model. The model we are going to use is a pixel model. That is to say, we will use exactly the same model on every pixel and each pixel will be treated independently of every other pixel. Because each pixel is being treated separately, if the image is 128 by 256 then there are a total of 32K different calculations that must be performed. However, each calculation involves only one complex data and two parameters so the calculations are very fast.

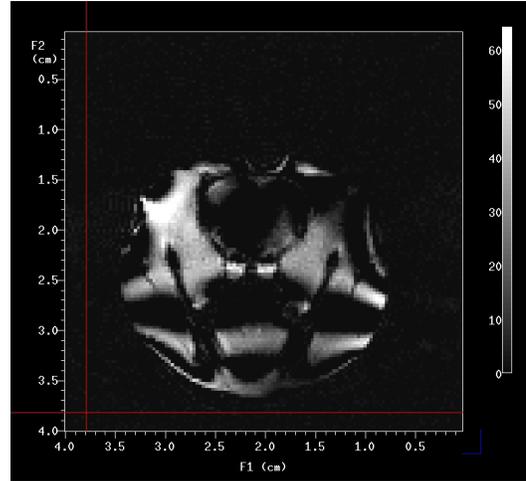
Because the discrete Fourier transform is an information preserving transformation, it does not matter if we do the calculation in the time or image domain; both calculations are equivalent. For convenience we will present this calculation in the image domain. What we wish to do is to determine how uncertain we are of both the phase and the amplitude in each pixel and then use the phase to generate a phased image. In a real sense we are not interested in either the amplitude and phase, rather we are interested in how uncertain we are of these quantities, because the resulting phased image must reflect these uncertainties.

Figure 27.1: Nonlinear Phasing Example

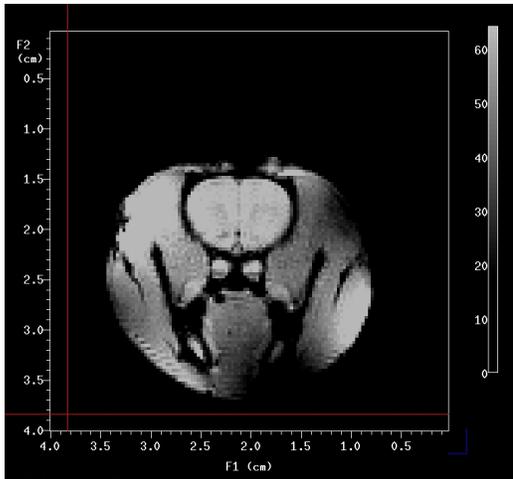
(A) Linear Phases, the Real image



(B) Linear Phases, the Imaginary image



(C) Non-Linear Phases, the Real image



(D) Non-Linear Phases, the Imaginary image

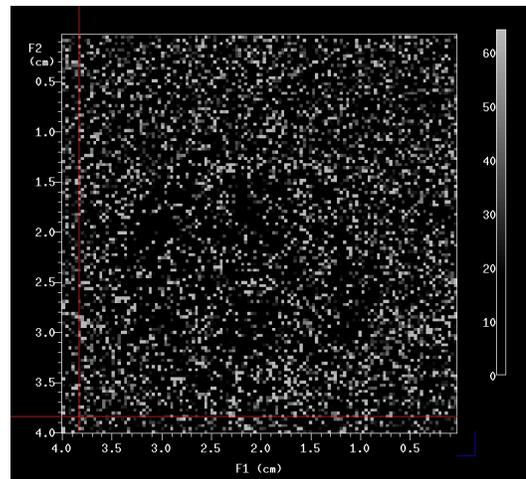


Figure 27.1: Panels (A) and (B) are the real and imaginary images generated from a gradient echo when the linear phasing algorithm is used. Note that the imaginary image, panel (B), still has a strong signal. This signal oscillates positive and negative in both the real and imaginary images. Panel (C) is the real image generated by the nonlinear phasing algorithm. The imaginary image (D) is essentially noise. If you compute the standard deviation of the noise from a section of the real and imaginary images you find they are essentially identical; indicating that the image is almost perfectly phased.

If  $d$  represents a complex image pixel value, then the model for any given pixel is

$$d = A \exp \{-i\theta\} + n \quad (27.1)$$

where  $A$  is the amplitude of the image and  $\theta$  is the phase. The quantity  $n$  represents the noise, and in this calculation we will assume the standard deviation of the noise,  $\sigma$ , is known. Separating the real and imaginary parts of this signal one has

$$d_R = A \cos(\theta) + n_R \quad (27.2)$$

for the real channel and

$$d_I = -A \sin(\theta) + n_I \quad (27.3)$$

for the imaginary channel, where  $d_R$  and  $d_I$  represent the real and imaginary pixel values and  $n_R$  and  $n_I$  represent the real and imaginary noise values. In the calculations which follow it will be assumed that the standard deviation of the noise is the same in both the real and imaginary channels and that the noise values are the same over the entire image.

## 27.2 The Bayesian Calculations

The Bayesian calculation consists of applying Bayes' theorem

$$P(A\theta|\sigma d_R d_I I) = \frac{P(A\theta|\sigma I)P(d_R d_I|\sigma A\theta I)}{P(d_R d_I|\sigma I)} \quad (27.4)$$

where  $P(A\theta|\sigma d I)$  is the joint posterior probability for the parameters given the noise standard deviation, the data and the prior information  $I$ . The joint prior probability for the parameters,  $P(A\theta|\sigma I)$ , represents what was known about these parameters before the data were acquired. The direct probability for the data,  $P(d_R d_I|\sigma A\theta I)$ , is essentially the likelihood function and  $P(d_R d_I|\sigma I)$  is a normalization constant. If we normalize this probability density function at the end of the calculation and factor the joint prior probability for the parameters, one obtains

$$P(A\theta|\sigma d I) \propto P(A|I)P(\theta|I)P(d|A\theta\sigma I) \quad (27.5)$$

as the joint posterior probability for the parameters.

In this calculation we will assign independent prior probabilities for the amplitude and Phase. And the likelihood will be assigned using a Gaussian prior probability. By definition the amplitude should be positive, but we do not wish to insert a hard lower bound on the amplitude. The reason for this is that in this analysis we are not doing a model selection calculation; rather we are doing a parameter estimation calculation. If we were doing a model selection calculation, then we could select between Eq. (27.1) and the "no signal" model. If we had done that then a hard cutoff would have worked fine as the prior probability. In regions where there is no signal, we would just get noise, and in regions where the signal is large we would get a properly phased signal. However, if we are doing parameter estimation, then a hard lower bound will necessarily force the real image to be positive, and so put a constant offset into the image: exactly the same thing that happens when one uses an absolute image. Consequently, we will use a prior that strongly suggests the amplitude

should be positive:

$$P(A|\sigma I) \propto \begin{cases} \exp\left[-\frac{A^2}{2\sigma^2}\right] & \text{if } A < -3\sigma \\ \exp\left[-\frac{A^2}{2\delta^2}\right] & \text{otherwise} \end{cases} \quad (27.6)$$

where  $\delta$  is one half the maximum intensity in the image. So this prior expresses the belief that the amplitude should be small rather than large. Around zero this prior has almost no effect on negative amplitudes until the amplitude become more negative than three noise standard deviations, then suddenly this prior expresses a rather strong belief that the amplitude should be closer to zero. So small negative values are allowed, provided they are small enough not to protrude above the noise floor.

The prior probability for the phase is also assigned using a Gaussian prior probability of the form:

$$P(\theta|I) \propto \exp\left\{-\frac{\theta^2}{2 \times 2^2}\right\}. \quad (27.7)$$

This prior expresses a slight belief that the phase should be zero. The reason for this is simply that noise does not really have a phase, and so the phase should be zero. Any data item having even a small significant amplitude will quickly override this prior.

The likelihood,  $P(d_R d_I | A\theta\sigma I)$  was assigned using a Gaussian given by

$$P(d_R d_I | A\theta\sigma I) \propto \exp\left\{-\frac{Q}{2\sigma^2}\right\} \quad (27.8)$$

where

$$Q \equiv [d_R - A \cos(\theta)]^2 + [d_I + A \sin(\theta)]^2. \quad (27.9)$$

In the program that implements the calculation, the noise standard deviation  $\sigma$  must be determined. There are many ways this might be done, but whatever is done, it must be general enough to work on any image. If you examine the output from the linear phasing routine, Fig. 27.1 panels (A) and (B), you will note that in small patches of the image, say  $3 \times 3$  pixels, the signal in the real and imaginary channels are roughly constant; different constants in the real and imaginary, but constant nonetheless. If you postulate a model that is a constant in this small region and then compare the constant model to the “no signal” model, one can compute the posterior probability for these two models. If there is no signal present, then that region can be used to compute the noise standard deviation. By going over the entire image using a  $3 \times 3$  set on pixels one can quickly get a very accurate estimate of the noise standard deviation by using only pixels for which the probability for no signal is much greater than the probability for the constant model.

The Bayesian calculations are implemented using Markov chain Monte Carlo, without simulated annealing. The Markov chain has Eq. (27.5) as its target distribution, using Eqs. (27.6,27.7) as the prior probabilities and Eq.(27.8) as the direct probability. The calculations are implemented in parallel, with parallelization occurring at the pixel level. Consequently, it 32 processors are available then 32 pixels are processed at one time. The simulations are initialized using the maximum likelihood estimated of both the amplitude and phase. Consequently, these simulations start very near the maximum of the joint posterior probability and all that is necessary is to run the simulations long enough for them to reach equilibrium.

In most Markov chain Monte Carlo simulations, it is the means and standard deviations of the parameter samples that are output. If we had used the mean phases to generate the output image, we would almost certainly get the maximum of the direct probability in regions where there is a signal, and, consequently, the output would essentially be an absolute value spectrum. However, we are not trying to estimate the amplitudes and phases, we are trying to phase an image. In the Markov chain Monte Carlo simulations each of the samples from the simulations are characteristic of the phase and amplitude supported by the data. Consequently, we randomly take one phase from one of the Markov chain Monte Carlo simulations and use that phase to produce an absorption mode pixel. This process is repeated for each pixel, giving an absorption model image. Figure 27.1 panels (C) and (D) are the real and imaginary parts of a gradient echo image that was phased using this procedure. Note that the real image (C) contains the fully phased image, plus noise; while the imaginary image (D) contains only noise. If one computes the noise standard deviation for these two images one finds essentially the same value in both real and imaginary images, and this value is the same as what is found in both panels (A) and (B), the real and imaginary parts of the linearly phased image; so this procedure has moved the positive intensity to the real channel and left the noise behind.

### 27.3 The Interfaces To The Nonlinear Phasing Routine

The interfaces to the Nonlinear Phasing package are shown in Fig. 27.2. To use this package you must:

- Run the linear phasing algorithm and have that package write both the imaginary and real FDF images.
- Select the type of processing:
  - **All**: all of images are to be phased separately.
  - **One**: only the currently displayed image is to be phased.
  - **Common**: the phase from the currently displayed image are to be computed, and those phases are then used on all of the images.
- Indicate if the imaginary images are to be written, the default is not to write these images.
- Indicate if the Ascii image are to be written, the default is not to write these images.

The remaining widgets on this interface are used to view the outputs and to indicate where the analysis is to be run. These widgets are pretty standard and we do not discuss them further. If you need to know more about the function of these widgets activate the help button and then activate the widget in question.

Figure 27.2: The Interface To The Nonlinear Phasing Package

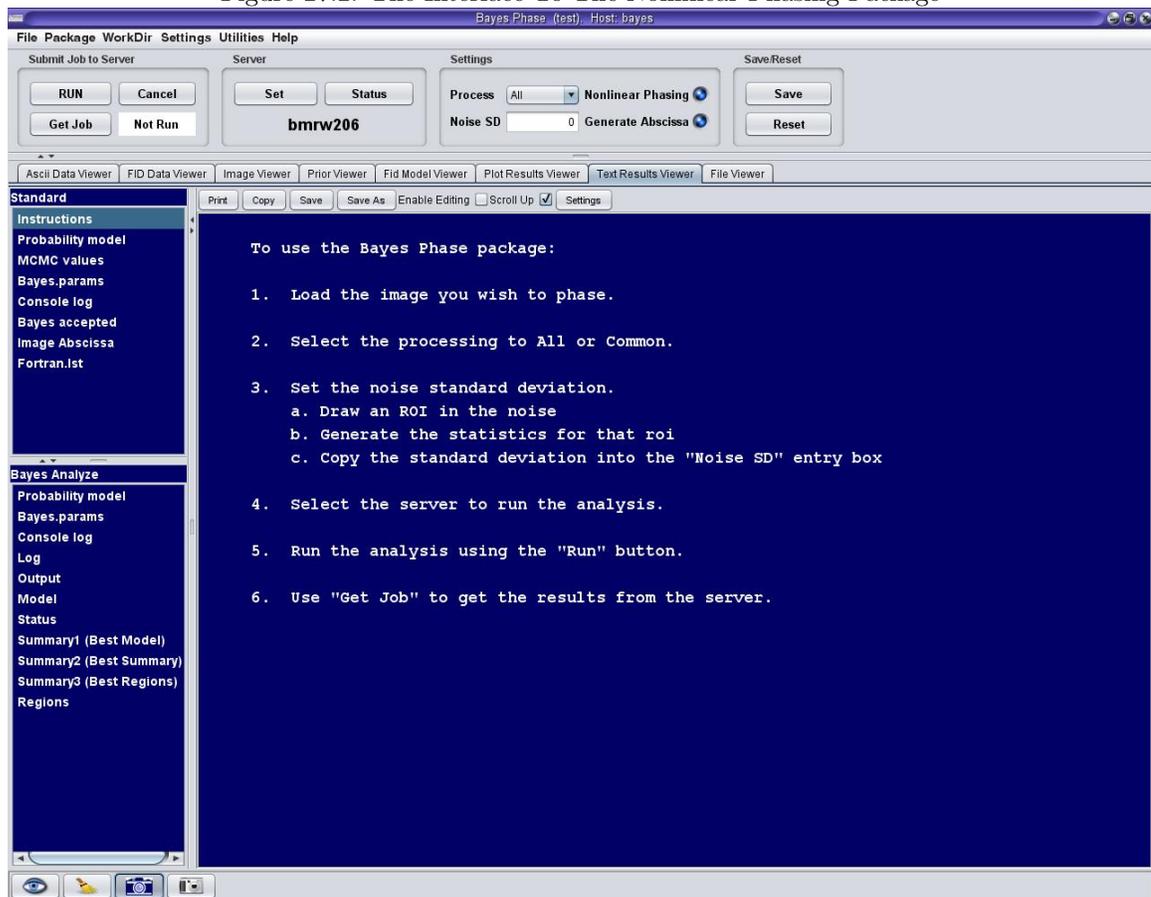


Figure 27.2: The interface to the Nonlinear phasing package is shown here. The output from this package is the phased real and imaginary parts of the input image. These images, usually just the real image, can be used in other analysis such as inversion recovery and diffusion tensor packages.

## Chapter 28

# Analyze Image Pixel

The Analyze Image Pixel package allows you to enter a model of your own and then use Bayesian probability theory to analyze that model.<sup>1</sup> The Java interface to the Image Pixel package is shown in Fig. 28.1. The Bayesian calculations performed by this package are identical to those in Bayes Enter Ascii, Chapter 20, and consequently we are not going to repeat those calculation here; rather in this Chapter we will concentrate our attention on the problem of how to use this package. To use this package, you must do the following:

**Select** the “Analyze Image Pixel” package from the Package menu.

**Load** the image data that is to be processed by the package. The analyze image Pixel package analyzes arrayed images on a pixel by pixel basis.

**Load** a Fortran or C model using the “System” or “User” buttons in the “Load And Build Model” widget group.

**Load** an abscissa file. A typical arrayed image data set is a stack of images each gathered at some parameter settings. For example, if the data are diffusion tensor data, then the abscissa would be a vector specifying the  $b$  values for each element in the array. These  $b$  values would be  $b_x$ ,  $b_y$  and  $b_z$ . So the abscissa file would be a three column Ascii file containing the  $b$  values each element in the array. In general the user specifies the number of abscissa columns, See Chapter A.3 for more on the abscissa file. Note the Analyze Image Pixel package uses the same abscissa for every pixel in the data.

**Build** the model using the “Build” button.

**Check** the Analysis Options boxes as you see fit.

**Find Outliers** tells the package to use an outlier model to select and eliminate pixels with odd characteristics.

---

<sup>1</sup>I would like to build a system library of predefined models. If you have models that you think would be of general use, I would like to hear from you. To have one of your models included, I would need the source code, the parameter file, a brief description of the model equations and data requirements

Figure 28.1: The Interface To The Analyze Image Pixels Package

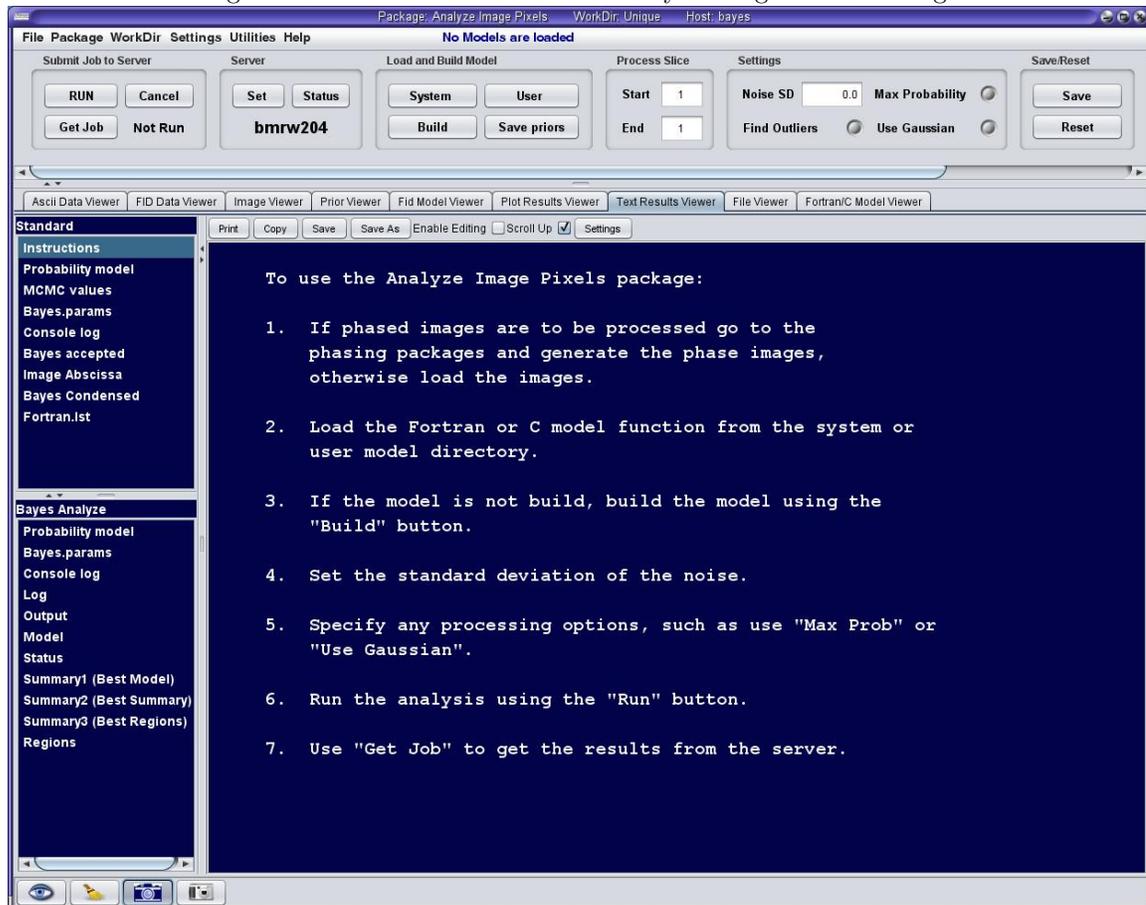


Figure 28.1: The Analyze Image Pixel package allows the user to load an arrayed image and then load a model, either system defined or user defined, and analyze the input image on a pixel by pixel basis using the loaded model. Prior to running an analysis, you must also load an appropriate Abscissa file. Unlike Ascii input data, which requires an Abscissa to load, any image can be loaded. So to process the data the Abscissa values must be loaded.

**Max Probability** tells the package to locate the maximum of the posterior probability. Locating the maximum is done using a searching algorithm and is much faster than using a Markov chain Monte Carlo algorithm. When Max Probability is selected the output images are generated using the parameters that maximized the posterior probability.

**Use Gaussian** tells the package to switch from using a  $t$ -distribution to using a Gaussian. In this case the input standard deviation of the noise is used as the standard deviation of the Gaussian likelihood. If Use Gaussian is not selected, a Student's  $t$ -distribution is used for the likelihood. The input noise standard deviation is used to threshold pixels. Arrays of pixels having a root mean-square smaller than the input standard deviation of the noise are not processed.

**Set the Noise SD** to a value equal to your best estimate of the standard deviation of the noise in the image stack that is to be processed. To get an estimate of the noise standard deviation, draw an ROI in a region where there is no signal and hit the "Get Statistics" button. The output on the right-hand side labeled "RMS" contains the root mean-square image pixel value and is a good estimate of the noise standard deviation.

**Review** the prior probabilities for the loaded model using the Prior Viewer.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

The actual processing done by this package are essentially the same as that done in the Bayes Enter Ascii package and we refer you to that package, Chapter 20, for more on the processing being done. However, there is one difference: the Analyze Image Pixel package does a model selection calculation. In this calculation there are two models: a No Signal model, and a signal model. The No Signal model assumes the signal consists of only noise while the signal model assumes the signal is given by your model plus noise. When the no signal model is selected, there are no output parameters. Output parameter maps contain zero in these no signal regions. However, there is an exception to this, on the output map containing the standard deviation of the noise, both signal and no signal regions have an estimate of the noise standard deviation. In regions with no signal, the standard deviation is computed as the root mean-square of the total data value. In regions containing signals, the output standard deviation is computed as the root means-square residual.

## 28.1 Modification History

In release 4.10, the Markov chain Monte Carlo routine in Bayes Image Pixels was replaced by one as similar to the one in the Enter Ascii Model package, Chapter 20, as possible. This was done because in some cases the Enter Ascii package's Markov chain was able to solve problems that Bayes Image Pixel package could not. Given that the data and the model were identical in the cases at hand, I

concluded the Markov chain in Bayes Image Pixels package needed replacing. The new routine is almost line by line identical to the routine in the Enter Ascii package.

In addition to replacing the Markov chain, I also modified the output routines so that it tests each output image to see if it is null, i.e., all zero. If the image is zero everywhere nothing is written. This was done to eliminate a perceived problem. In the process of testing a new model, a user generated an image containing no signal and then ran the Bayes Image Pixel using the no signal image and his new model. The old Bayes Image Pixel package correctly identified the pixels as no signal and so set the output images to zero. When processing was complete, the output routines wrote these zero images, and the users, myself included, interpreted these zero output images as an error on the part of Bayes Image Pixels package. That interpretation was incorrect and as soon as I realized what was happening, I modified the output routines so that zero images are not written.

## Chapter 29

# The Image Model Selection Package

The Image Model Selection Package allows you to load one or more Ascii model and then use Bayesian probability theory to compute the posterior probability for the loaded models, thus allowing you to determine which model best accounts for the data.<sup>1</sup> To use this package you do not have to have Fortran or C installed on your server. However, if you do not have Fortran or C installed, you must use the system models. Consequently, installing both Fortran and C is strongly recommended. The interface to this package is shown in Fig. 29.1. To use this package, you must do the following:

**Select** the “Analyze Image Pixel” package from the Package menu.

**Load** one or more Fortran or C model using the “System” or “User” buttons in the “Load And Build Model” widget group.

**Load** an Arrayed image using Files/Load Image menu. viewer. used by the Bayesian Analysis software.

**Load** an Abscissa using Files/Load Abscissa menu.

**Enter** the Standard Deviation of the noise.

**Check** the Use Gaussian widget, if the number of array elements is small. Here small means if the number of parameters being estimated is within a factor of 2 of the number of array elements.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

---

<sup>1</sup>I would like to build a system library of predefined models. If you have models that you think would be of general use, I would like to hear from you. To have one of your models included, I would need the source code, the parameter file, a brief description of the model equations and data requirements.

Figure 29.1: The Interface To The Image Model Selection Package

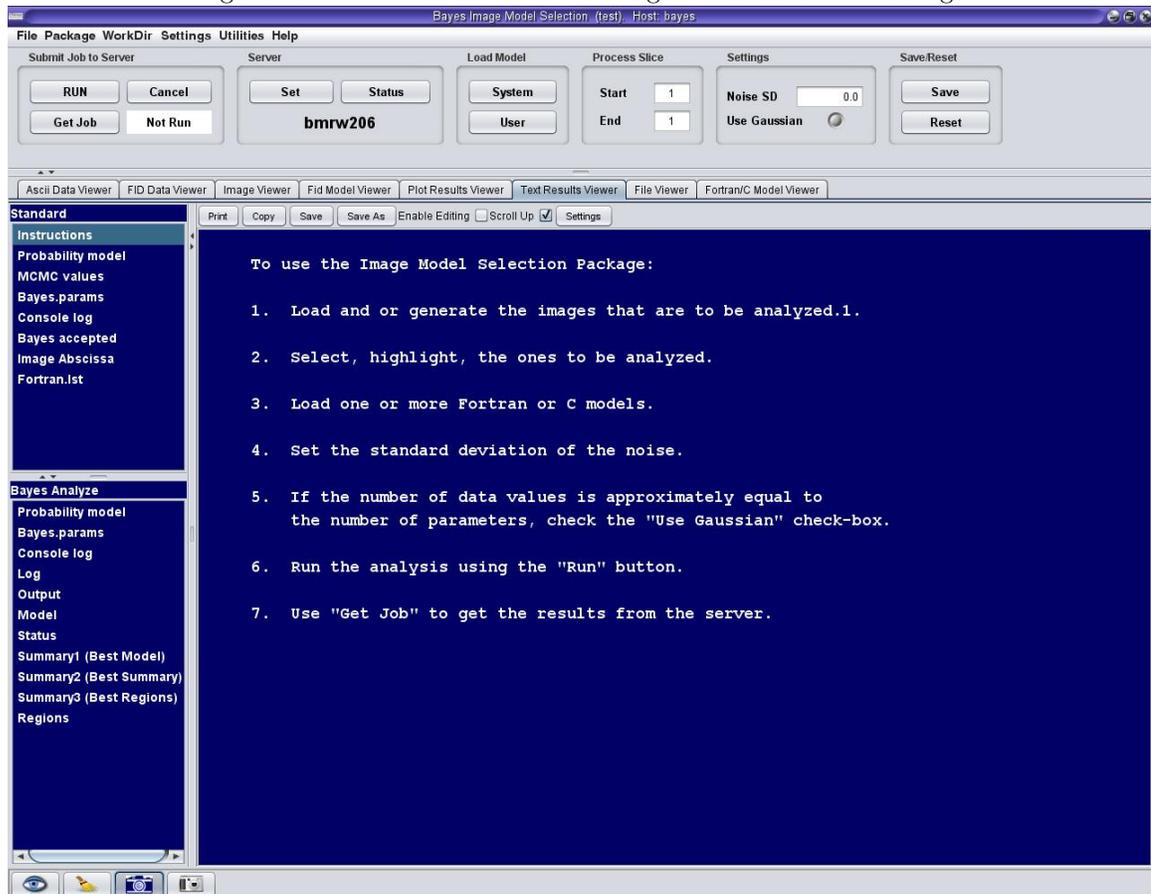


Figure 29.1: The Image Model Selection package allows the user to load an arrayed image and then load several models, either system defined or user defined, and analyze the input image on a pixel by pixel basis using the loaded models. The program that implements this package computes the posterior probability for the modes. Prior to running an analysis, you must also load an appropriate Abscissa file.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

## 29.1 The Bayesian Calculations

The Bayesian calculation done in this package is identical to that done in Chapter 22 and we will not repeat that calculation here. However, we will describe how the Image Model Selection package differs from the Enter Ascii Model Selection package. First, the Image Model Selection package process each pixel in an image. That is to say the same calculation is performed on each pixel in the image. It is assumed that the image is arrayed and the models describe the signal in the arrayed pixels. The program does not process all pixels at one time using a common model; rather it process each pixel in parallel.

In the model selection calculation done by the Image Model Selection package, it assumes one has a set of models,  $U_j \equiv \{U_1, \dots, U_m\}$ , and one wishes to compute the posterior probability for each of the  $U_j$  models in every pixel in the image. These models can be loaded from the System directory or they can be loaded from the user directory. They don't need to have common parameters, but they do need to have common data requirements because they all use the same data. Each of these models will have a set of parameters associated with it. Unlike the Analyze Image Pixel routine which output's parameter maps, the Image Model Selection routine cannot output a parameter map because the different models will in general have different parameterizations and consequently no map can be output.

To circumvent this problem, we require the Fortran/C models to have the same derived parameters so that an output parameter map can be generated by the user. How the user constructs these maps, is up to the individual who is writing the Fortran/C codes. For example the exponential models supplied in the system directories, define three output decay rate constants: a low, middle and high decay rate constant. The one exponential model sets the low, middle, and high derived decay rate constants to be equal. The two exponential model, sets the low decay rate to the lowest estimated rate constant, and the middle and high derived rate constant are set to the highest estimated rate constant. Finally, the three exponential model sets the low, middle and high as the three estimated low, middle and high rate constants. By setting the output derived images in this fashion, the low rate constant will always be the lowest decay rate. The middle decay rate will always be second largest decay rate and the high decay rate constant will always be the largest decay rate constant found.

The equation that relates model  $U_j$  to the data is given by:

$$d_{kl}(t_i) = U_j(t_i, \Omega_j) + n_{kl}(t_i) \quad (29.1)$$

where  $d_{kl}(t_i)$  is the data at  $l$ th readout of the  $k$ th phase encode pixel sampled at abscissa values  $t_i$ . The  $t_i$  may be vector valued or it may be a single column of numbers. However, it must have the same number of columns in all loaded models. And it must be the same abscissa for all pixels. The noise is represented symbolically by  $n_{kl}(t_i)$ . The models  $U_j$  can be any system or user defined model and they can use marginalization or not depending on the model.

The program that implements this calculation loops over all of the pixels in the image and performs the model selection calculation described in Enter Ascii Model Selection, Chapter 22.

Equation 22.60 is the joint posterior probability for the nonlinear parameters when the amplitudes are marginalized out of the posterior probability, and Eq. (22.18) is used when there are no parameters marginalized out of the posterior probability. It is these two equations that are targeted by the Markov chain Monte Carlo simulations used to implement the Image Pixel Model Selection Calculation.

## 29.2 Outputs Form The Image Model Selection Package

The test data set contained in the `bayes.test.data/ImagePixelsAll` directory is shown in Fig. 29.2. The test image shown in Fig. 29.2 is a  $5 \times 5 \times 51$ , having a dwell time of 0.1 seconds. So the first data value is a  $t_1 = 0$  and the last data value is at  $t = 5$  seconds. The decay rate constants are 1, 2, 3, 4, and 5 in each row. And the amplitudes are 0, 20, 30, 40 and 50 for each row. The noise has standard deviation of one. When this test data was run, 6 models were loaded: `ExpOneNoConst_Marg.f`, `ExpOneConst_Marg.f`, `ExpTwoNoConst_Marg.f`, `ExpTwoConst_Marg`, `ExpThreeNoConst_Marg.f` and `ExpThreeConst_Marg.f`. The no signal model, model zero, is always loaded and is used to designate which pixels do not have a signal. The posterior probability for the `ExpOneNoConst.f` model is shown Fig. 29.4. Note the black band on the left-hand side of this image. These are the no-signal pixels and the program correctly identified them as containing no signal and did not process them. The nonzero area contains the expected model number in the data. In this case the models are numbers are 0, 1, 2, 3, 4, and 5. Model number zero is the no-signal model, model number 1 is the `ExpOneNoConst_Marg.f` model, etc. I computed the mean and standard deviation in this region and its  $0.986 \pm 0.003$  meaning that single exponential with no constant model was identified with a probability averaging 0.986.

The image model selection package outputs parameter maps of all of the derived parameters. So for example, when testing one, two and three exponential models you get a low, middle and high decay rate constant. Each parameter map comes as a set of three images: the mean parameter, the peak parameter, and the standard deviation of the parameter. The mean parameter map, is the mean or average value of the parameter in all of the Markov chain Monte Carlo simulations. The “peak” value of the parameter is the value of the parameter in the simulation that had maximum posterior probability. Finally, the standard deviation of the parameter is the standard deviation of the parameters computed from all of the samples gathered in the Markov chain Monte Carlo simulation. Additionally, you get maps of the amplitudes that occurs in the model. For example in the exponential model selection example we have been discussing you get both the amplitude and constant offset maps.

There are three additional outputs, one is the average logarithm of the likelihood. And can be used to spot outliers. The posterior probability for the model is output. In this case the expected mean value of the model number is output. For example, in the one, two and three exponential model example. I loaded the one, two and three exponential models in that order. So model number one was the one exponential model, two was the two exponential model, etc. The output model number map, is the average value of the model number for each pixel in the image. Model number zero, is the no signal model and implies that none of the models fit the data than the no signal model.

You also get parameter maps for each of the high probability models. For example if you load the “`ExpOneNoConst.img`” file and run the one, two and three exponential model selection, you will get parameter maps for the one exponential model, but not the two and three. That’s because in this

Figure 29.2: Single Exponential Example Image

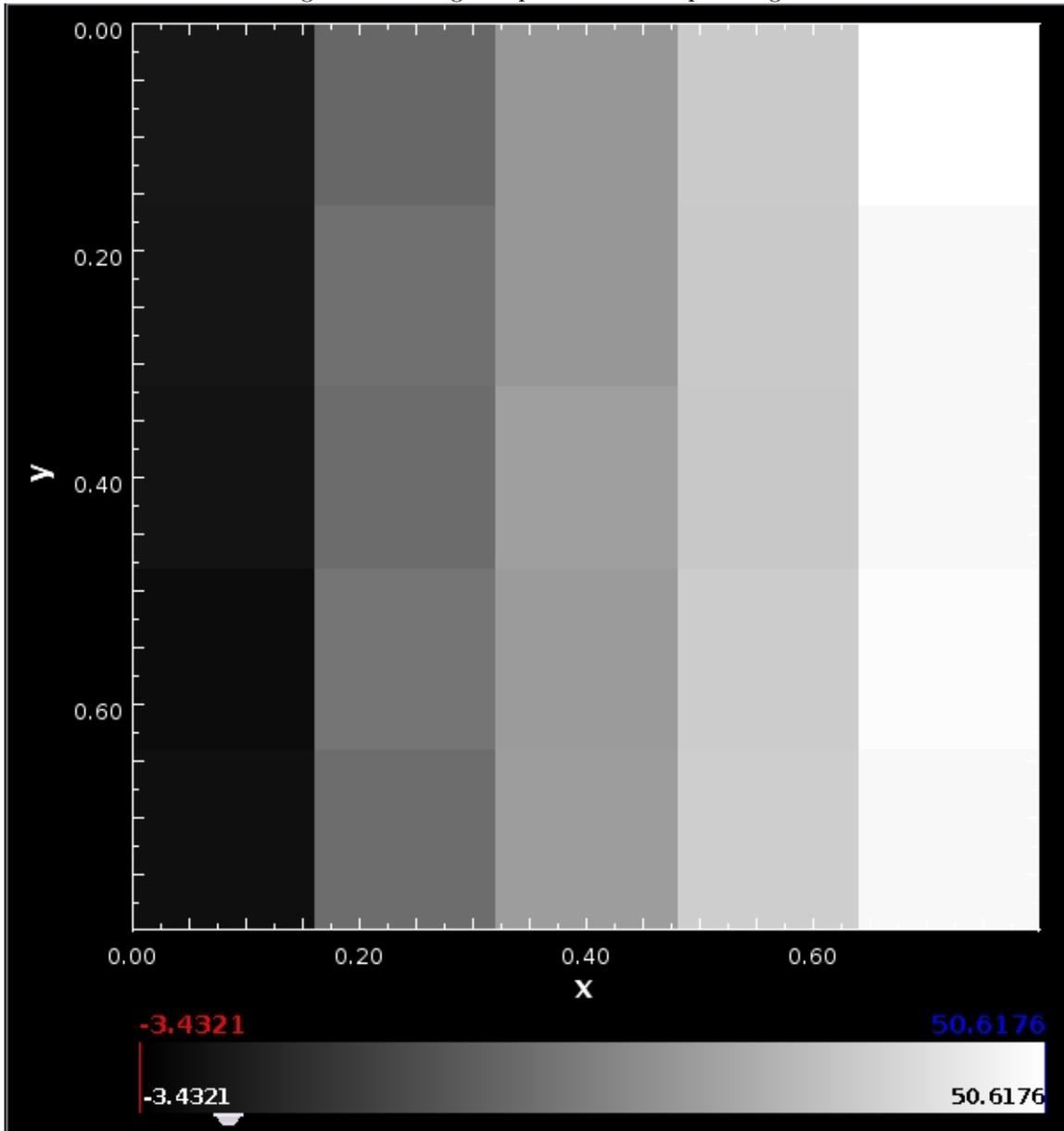


Figure 29.2: This is an test arrayed image that contains exponentially decaying data without a constant. The image was generated so that the decay rate constant in the horizontal rows have decay rate constants of 1, 2, 3, ..., 5. The amplitude of each row is 0, 20, 30, 40, and 50.

Figure 29.3: Single Exponential Example Data

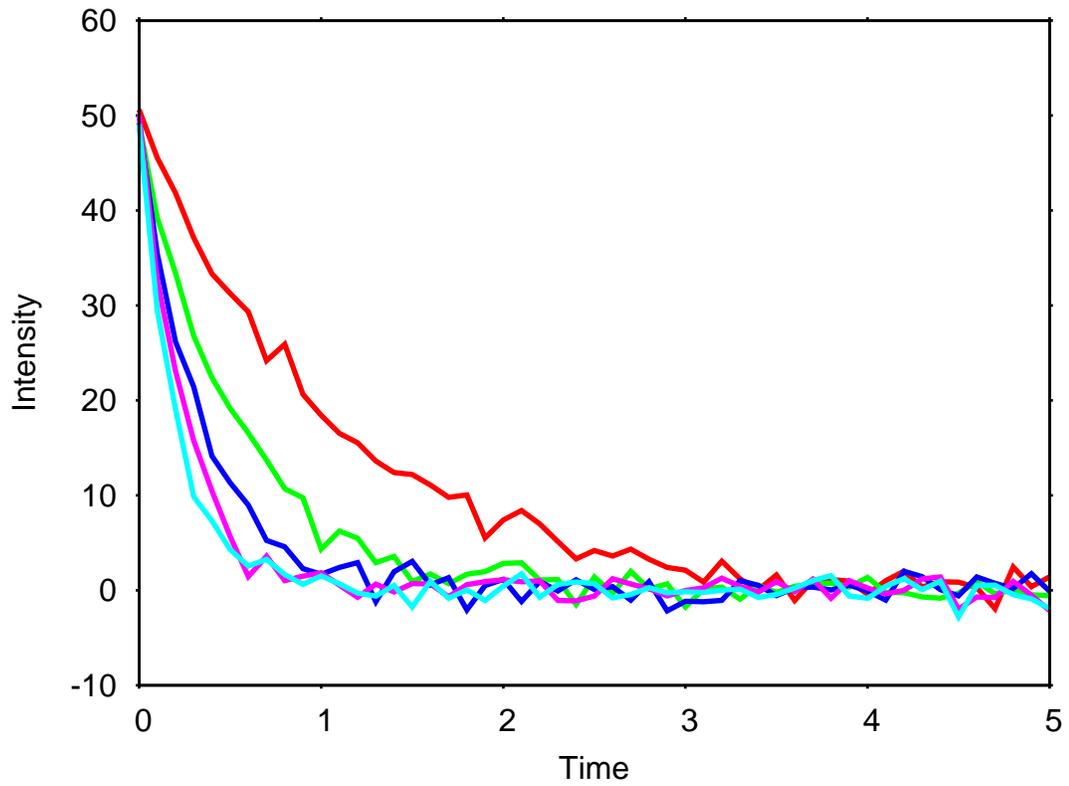


Figure 29.3: If one extracts the arrayed data in the 5th column, you will obtain the 5 exponentially decaying Ascii data sets shown here. Note that the intensity is the same in each data set, but the decay rate constant are rapidly increasing.

Figure 29.4: Posterior Probability For The ExpOneNoConst Model

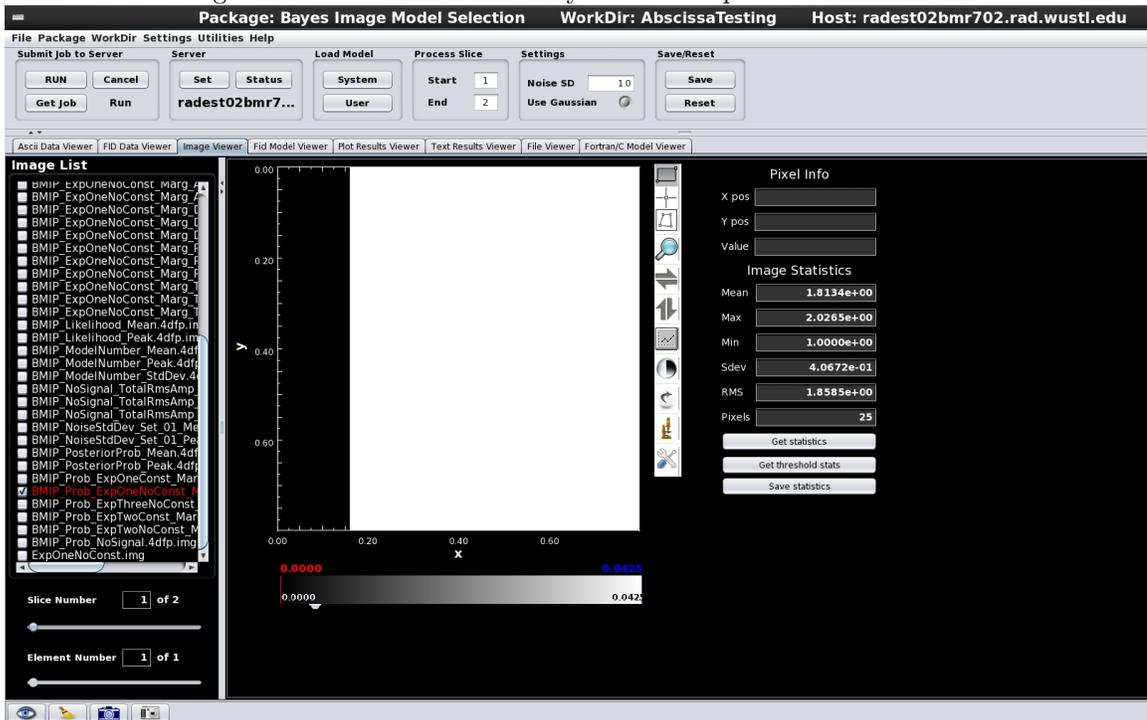


Figure 29.4: The posterior probability distributes itself across all size models being tested. When the posterior probability is near one, the model is almost certain. In this figure, we show the posterior probability for the ExpOneNoConst.f model. The posterior probability averages about  $0.983 \pm 0.003$ , so probability theory is almost positive all of the pixels are single exponential without a constant. And given that this is simulated test data, that is indeed the correct answer.

data set the two and three exponential models are never selected, and so there are no parameters to output.

## Chapter 30

# Analyze Image Pixel Unique

The Analyze Image Pixel Unique abscissa package is a copy of the Analyze Image Pixel package and as such has all of the functionality as its parent. It differs from the Bayes Image Pixel package in that the Unique package takes two input images one containing the data and an image of exactly the same size containing an abscissa file: one abscissa value for each data value. So this is a generalization of the Analyze Image Pixels package so that more than one abscissa can be processed. The interface to the Analyze Image Pixel Package with a unique abscissa is shown in Fig. 30.1. To use the Analyze Image Pixel Unique abscissa package, you must do the following:

**Select** the “Analyze Image Pixel Unique” abscissa package from the Package menu.

**Load** the data; i.e., load an arrayed image that is to be processed by this package. Note the data file name must alphabetically come before the abscissa file. That is to say the package assumes the data is the first image and the abscissa is the second.

**Load** an abscissa image file. The abscissa image contains the abscissa for each pixel in the image to be used as the data. For example, a typical inversion recovery image data set is a stack of images each gathered at some sampling times  $t_i$ . The abscissa image file must contain the sampling times  $t_i$  for each pixel in the loaded data file. For this package only, the abscissa file is required to be a single column of numbers.

**Load** a Fortran or C model using the “System” or “User” buttons in the “Load And Build Model” widget group. Because this package assumes a single column abscissa, you will not be able to run models that use more than a single column abscissa. For example, because of this restriction, you will not be able to run a diffusion tensor models.

**Build** the model using the “Build” button.

**Check** the Analysis Options boxes as you see fit.

**Find Outliers** tells the package to use an outlier model to select and eliminate pixels with odd characteristics.

**Use Gaussian** tells the package to switch from using a  $t$ -distribution to using a Gaussian. In this case the input standard deviation of the noise is used as the standard deviation

Figure 30.1: The Analyze Image Pixel Unique Abscissa Package Interface

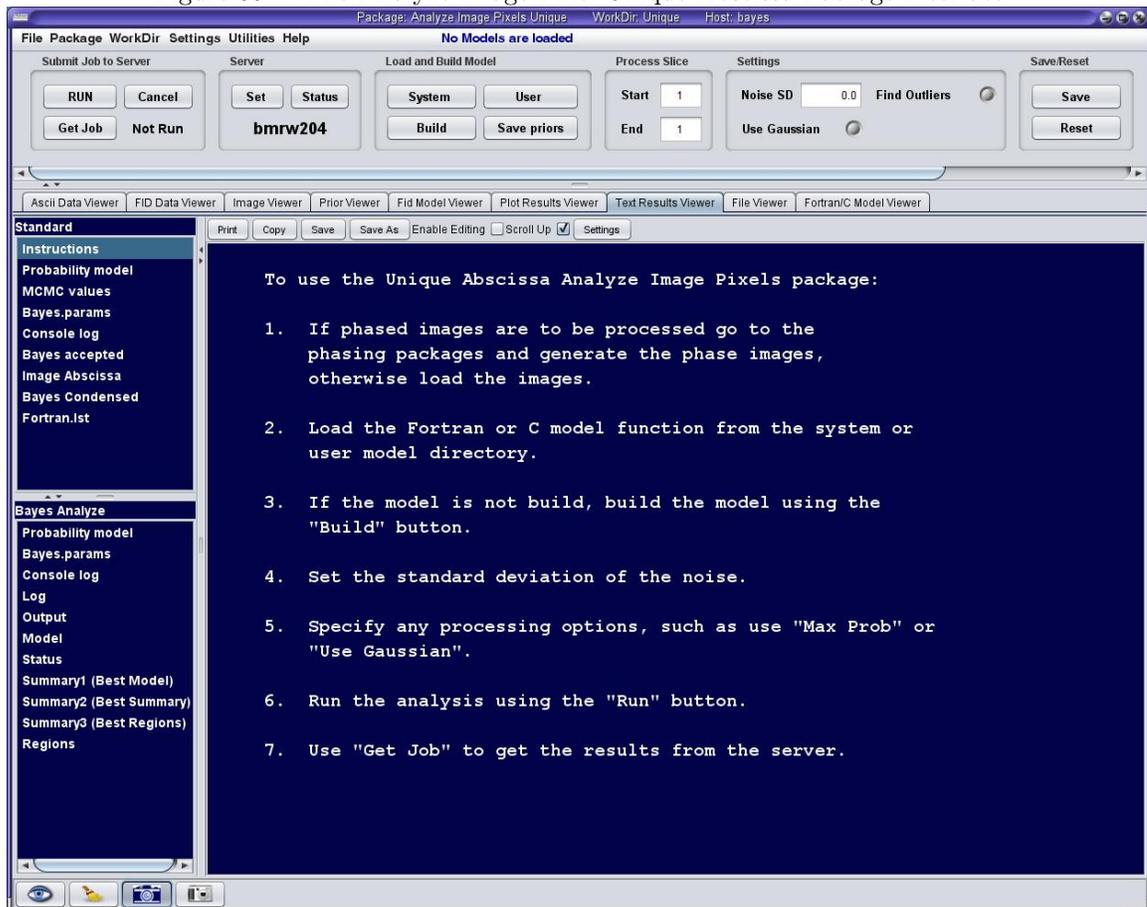


Figure 30.1: The Analyze Image Pixel Unique Abscissa package is a copy of the Analyze Image Pixel Package modified to take a unique abscissa for each pixel value in the input images.

of the Gaussian likelihood. If Use Gaussian is not selected, a Students'  $t$ -distribution is used for the likelihood and the input noise standard deviation is used to threshold which pixels. Array's of pixels having total root mean-square smaller then the input standard deviation of the noise are not processed.

**Set the Noise SD** to a value equal to your best estimate of the standard deviation of the noise in the image stack that is to be processed. To get an estimate of the noise standard deviation, draw an ROI in a region where there is no signal and hit the "Get Statistics" button. The output on the right-hand side labeled "RMS" contains the root mean-square image pixel value and is a good estimate of the noise standard deviation.

**Review** the prior probabilities for the loaded model using the Prior Viewer.

**Highlight** the data and the abscissa images.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis.

As noted the Analyze Image Pixel Unique abscissa package takes two input images and an input model and then processes each pixel in the images on a pixel by pixel basis. The first image in the selection list is treated as the input data while the second input image is treated as the abscissa. The program process each pixel one at a time. It takes one arrayed pixel from the input data image and the same arrayed pixel from the abscissa image and it passes these vectors the the Markov chain Monte Carlo routine. The Markov chain Monte Carlo routine then computes the posterior probability for the parameters given the data, the abscissa and the model. It outputs the parameter estimates for each parameter in the model and it outputs derived parameters if so requested. The outputs are in the form of unarraied parameter maps having the number of slices specified by the from and two slice numbers. The only text reports are an accepted report, an MCMC Values report and the console log. The Bayes Accepted report is used to log progress of the analysis, The MCMC Values report contains the configuration parameters and the prior probabilities needed to reproduce the run if need be. The console log contains some information about the posterior probabilities and mostly is used to monitor progress when the analysis is running.



## Chapter 31

# Bayes Test Data

The Bayes Test Data package allows the user to load either a user or system model, and then generate test data using that model. The interface to the Bayes Test Data package is shown in Fig. 31. Using the package is very similar to using the Enter Ascii Model package, Chapter 20.

### 31.1 Running the Package

To use this package, you must do the following:

**Select** the “Bayes Test Data” package from the Package menu.

**Load** a Fortran or C model using the “System” or “User” buttons in the “Load And Build Model” widget group.

**Build** the model using the “Build” button.

**Select** the abscissa type using the “Abscissa” widget. Valid values are NonUniform, Uniform or Read to load you own abscissa. When the “Read” option is selected an abscissa loading widget will popup. Use that popup to navigate to your abscissa file and then load it.

**Set** the maximum abscissa value. When multicolumn abscissa data are generated, each abscissa value may be thought of as a vector. The value in the Max Value widget is the maximum magnitude of the generated abscissa vector.

**Set** the size of the output image by setting the number of readout pixels (Ro), the number of phase encode pixels (Pe), the number of slices (# Slices), and the array dimension (ArrayDim).

**Set** the number of output images using the (# images) entry box. If this entry box is set to 2, for example, then there will be 2 output arrayed images generated.

**Set** the standard deviation of the Gaussian noise added to the output images. For example if the noise standard deviation is 1, and the generated signal has an initial amplitude of 10, then the peak signal-to-noise in the output image is 10 to 1.

**Review** the prior probabilities for the loaded model using the Prior Viewer.

Figure 31.1: The Bayes Test Data Package Interface

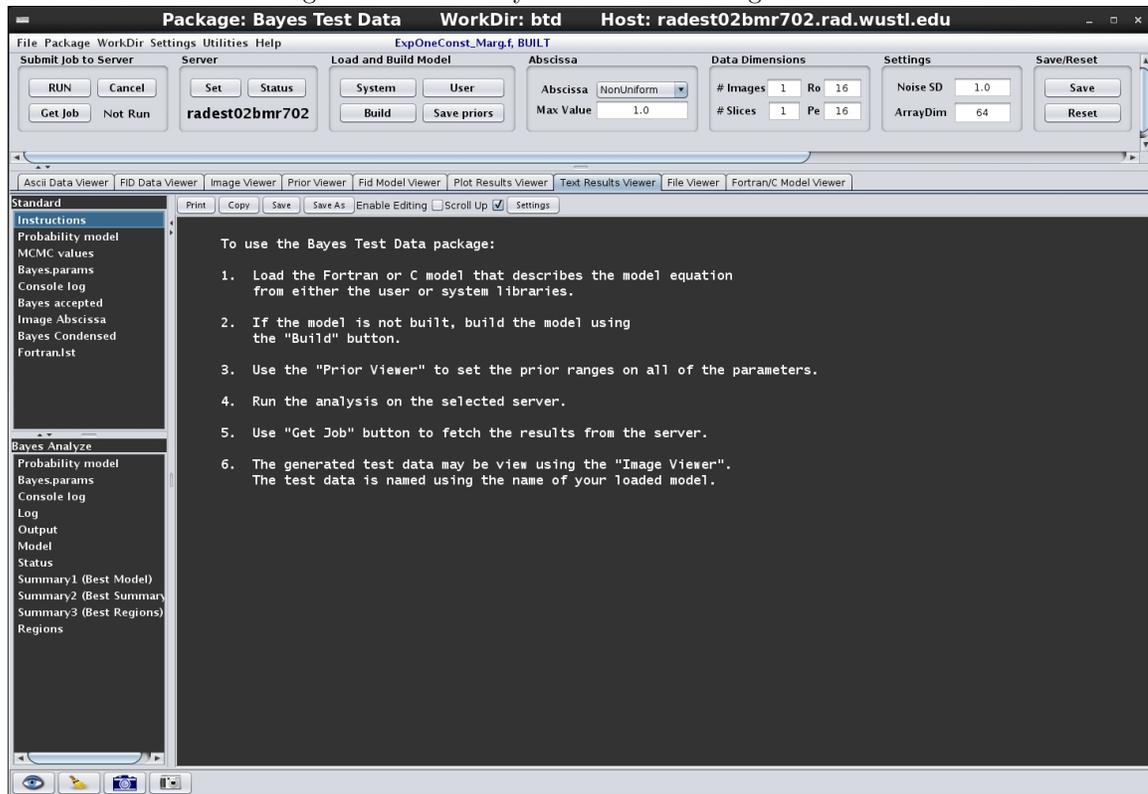


Figure 31.1. This is the interface to the Bayes Test Data package. To use the package, load and build a model, set the prior probabilities, set the dimensions of the image and run the analysis. The resulting 4dfp image file contains pixels that are samples of the model signal generated using random samples drawn from the prior probabilities for the model parameters, including the amplitudes.

**Select** the server that is to process the analysis.

**Check** the status of the selected server to determine if the server is busy, change to another server if the selected server is busy.

**Run** the the analysis on the selected server by activating the Run button.

**Get** the the results of the analysis by activating the Get Job button. If the analysis is running, this button will return the Accepted report containing the status of the current run. Otherwise, it will fetch and display the results from the current analysis and the generated images will be displayed in the Image Viewer.

As with the interface to all Bayes Packages, holding the cursor over a given widget will result in a tool tip being displayed. These tool tips are hints on what type of input is expected by a given box. For example, holding the cursor over the “Ro” widget will display:

Figure 31.2: The Output Image Test Data

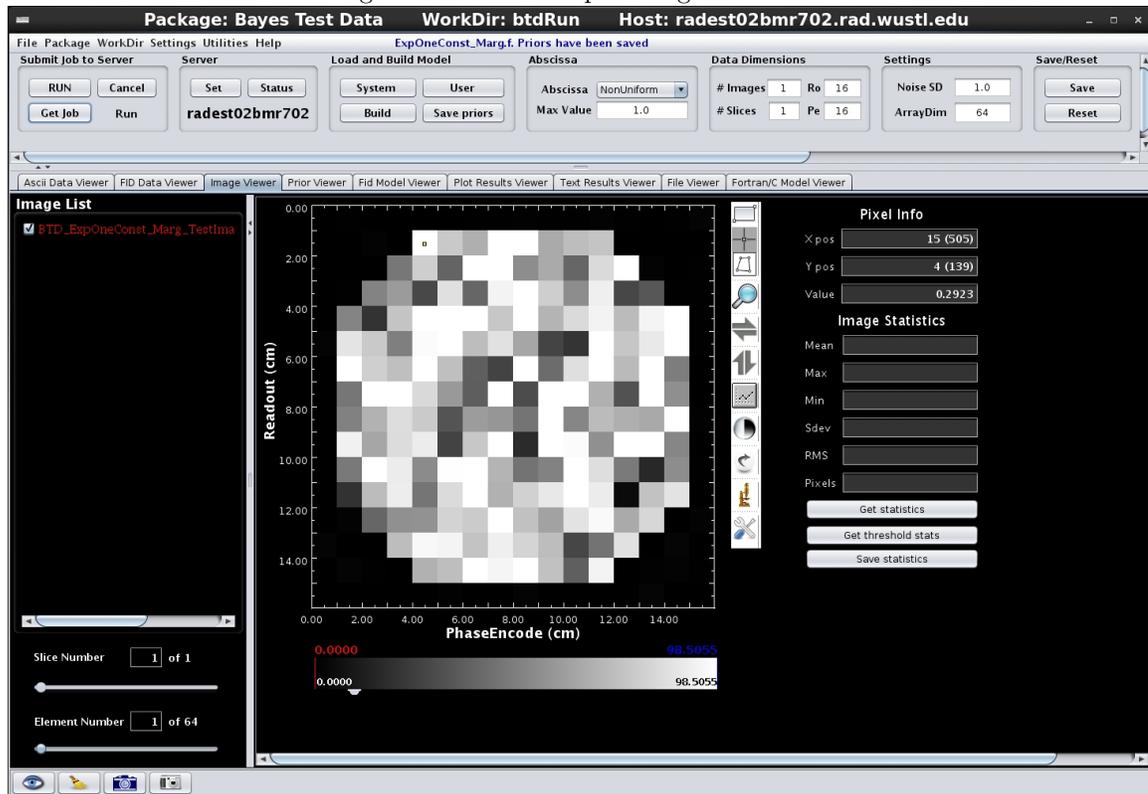


Figure 31.2. This is a typical output image generated using the single exponential with a constant model (ExpOneConst.NoMarg.f). To generate this data, the model was loaded, the parameters adjusted and then the package was run. Each pixel in the interior contains one exponentially decaying signal plus a constant. The output model data contained in the top left most pixel is displayed in Fig. 31.3.

The number of pixels in the read out (vertical) direction.  
Valid values are from 5 to 25.

The allowed values for the number of pixels in the read out and phase encode directions, 5 to 25, are pretty arbitrary. The only justification for them is that it was thought, by the author, that images smaller than  $5 \times 5$  do not really supply enough data to display and properly test a package; while images greater than  $25 \times 25$  provide more than is typically needed. After all a  $25 \times 25$  image contains 625 test data sets (pixels), enough for almost any testing purposes.

## 31.2 Outputs From the Bayes Test Data Package

The output from the Bayes Test Data Package consists of a console log, an MCMC Values Report, a Condensed Report, and an image. An example output image is shown in Fig. 31.2. This output

Figure 31.3: Example Exponential Test Data



Figure 31.3. Each output image data set is arrayed based on the value of the ArrayDim parameter. If the ArrayDim parameter is 64, then each output image consists of 64 arrayed images. Each arrayed pixel is a test data set. This is a plot of the arrayed values taken from the top left most signal pixel in Fig. 31.2. If you examine Fig. 31.2 carefully you will find the cursor in this pixel. The extracted signal is an exponential decay to a constant with SNR of about 110:1.

image is a standard “4dfl” image and you can find out more about this file format in Appendix G. The image is an ellipsoid having at least one “no signal” pixel around all of the edges. If the image is square the ellipsoid will be a circle. Deviations from these shapes are due to the small number of pixels. This  $16 \times 16$  pixel image was generated by loading and running the ExpOneConst\_Marg.f model. The prior probability for the decay rate constant was a positive prior with a low of zero, a high of 20, and a peak at 3. The prior probability for the amplitude was a Gaussian, with a low of zero, a high of 100, a mean of 50, and a standard deviation of 30. The prior probability for the constant was also a Gaussian having a low and mean value of zero, a high of 30, and a standard deviation of 10.

In the image shown in figure 31.2, pixels that show up as black on the first element array are inverted, i.e., negative, while bright pixels are positive. Because the initial amplitude of the exponential had a lower bound of zero, most of the output pixels are white, i.e., positive. Some of these pixels are black because of the Gaussian noise added to the array elements. The uniform gray area around the image is the region of the image that does not contain a signal. In the interface, if you place the cursor on the top-left signal pixel and hit the extract image pixel button, the figure shown in 31.3 will be displayed. This particular pixel has a signal-to-noise ratio of about 110:1.

The Bayes Test Data package generates test data suitable for analysis by other packages. For

example, the data set shown in Fig. 31.2, could be analyzed by several different packages. It could be analyzed by the Exponential package, the Enter Ascii Model package, the Ascii Model Selection package and by Image Pixel and Image Model Selection packages. If one runs the Unknown Number of Exponentials package on the extracted pixel shown in Fig. 31.3, the Unknown Number of Exponentials package will tell you that this data is single exponential plus a constant data. The decay rate constant of is estimated to be  $7.9 \pm 0.1$ . The actual value generated by the Bayes Test Data package was 7.65.

When the Bayes Test Data package runs, the value of the parameters, here the amplitude, constant and decay rate constants are written to the McMC Values report and to the Condensed file. To show format of the McMC values report it was necessary to make two example figures. The front part of the McMC Values report is shown in Fig. 31.4. This part of the report contains the standard header showing the configuration parameters, i.e., things like the number of abscissa columns, the number of data columns, etc. In addition toward the middle of this figure you will find the prior probabilities used in this run. These priors are written into the McMC values file to make a semi-permanent record of what values the prior probabilities had at the time of this run. Following the prior probabilities are the parameters that specify the image size, the type of abscissa and the name of the model used in this run.

The second figure Fig. 31.5 contains a mask of the output image. The horizontal direction in the mask is the phase encode direction, just as it is in an actual image, while the vertical direction is the readout direction. The pixels in this mask are either zero or one. If a pixel is zero then in the output image this pixel is a no-signal pixel: no-signal means the pixel contains a noise sample but does not contain a signal. Conversely, when the mask contains a one, the output image contains a signal plus noise. The parameters used in generating the signal are obtained by sampling the prior probabilities. Each pixel containing a signal contains a unique set of parameter values. These parameter values are written to the McMC values file along with the slice, row, column and image number. For example the first signal pixel is row 2, starting in column 5 and the signal was generated having a decay rate constant of roughly 7.65, the amplitude was 67.5 and the constant offset was 7.346 and in this test data the noise standard deviation was one, so the signal to noise in this pixel is roughly  $SNR = (67.5 + 7.34)/1 \approx 74.84$ .

Figure 31.4: The Front of the MCMC Values Report For Bayes Test Data Package

## Parameter File Listing for the Bayes Test Data package

```

! BayesTestData Package
! Created 07-Jul-2014 13:17:56 by larry
!
      Output Dir = BayesOtherAnalysis
      Number Of Abscissa = 1
      Number Of Columns = 1
      Number Of Sets = 1
      MCMC Simulations = 48
      MCMC Repeats = 30
      Minimum Annealing Steps = 31
      Histogram Type = Binned
      Outlier Detection = Disabled
      Number Of Priors = 3

Param Name  Low      Mean     High     Std Dev  Norm     Prior   Ordered  Param Type
DecayRate1  0.00E+00 3.00E+00 1.00E+01 1.00E+00 -2.52E+00 Positive NotOrdered NonLinear
Amplitude   -1.00E+02 0.00E+00 1.00E+02 3.00E+01 -3.62E+00 Gaussian NotOrdered Amplitude
Constant    -1.00E+02 0.00E+00 1.00E+02 3.00E+01 -3.62E+00 Gaussian NotOrdered Amplitude
      Package Parameters = 22
      Number Of Output Images = 1
      Output Image Directory = images
          NoRo = 16
          NoPe = 16
          ArrayDim = 64
          No Slices = 1
          Noise Std Dev = 1.0
          Abscissa = NonUniform
      Maximum Abscissa Value = 1.0
          Model Name = ExpOneConst_Marg.f
          Number Of Models = 2
          Number of Derived = 6 (Plus the Noise Standard Deviation, not shown)
          Derived 1 = DecayRateLow
          Derived 2 = DecayRateMiddle
          Derived 3 = DecayRateHigh
          Derived 4 = DecayTimeHigh
          Derived 5 = DecayTimeMiddle
          Derived 6 = DecayTimeLow

```

Figure 31.4: This is the front part of the Mcmc Values report generated by the Bayes Test Data package. This report starts with the the standard header used in the Mcmc Value report, for more on this report see Appendix D. Everything starting from the line labeled “Number of Priors” is unique to this package. The output lines following this label are the prior probabilities used in this model. These priors are used to generate random samples from the various parameters. The information following the priors are the settings on the interface and the information used to describe the model and its parameters.

Figure 31.5: The Bottom Part of the Mcmc Values Report For The Bayes Test Data Package

CurPe	Image Mask					
1	0000000000000000					
2	00001111111110000					
3	0001111111111000					
4	0011111111111100					
5	0111111111111110					
6	0111111111111110					
7	0111111111111110					
8	0111111111111110					
9	0111111111111110					
10	0111111111111110					
11	0111111111111110					
12	0111111111111110					
13	0011111111111100					
14	0001111111111000					
15	00001111111110000					
16	0000000000000000					

Slice	Pe#	Ro#	Image	DecayRate1	Amplitude	Constant
1	5	2	1	7.6506489E+00	6.7506945E+01	7.3460393E+00
1	6	2	1	3.2301096E+00	2.6132491E+01	1.1235835E-01
1	7	2	1	9.5950591E+00	2.5740339E+01	4.8940688E+00
1	8	2	1	4.2623961E+00	1.2127842E+01	4.3179767E+00
1	9	2	1	1.5723227E+01	8.7417089E+01	8.3247151E+00
1	10	2	1	3.9364772E+00	7.5561216E+01	2.0195897E+01
1	11	2	1	1.8492263E+01	5.3259875E+01	1.1037756E+01
1	12	2	1	5.5277300E+00	6.0724474E+01	5.8999031E+00
1	4	3	1	8.5354559E+00	7.0867363E+01	2.2884514E+01
1	5	3	1	5.5954907E+00	5.9384810E+01	5.0304735E+00
1	6	3	1	7.5992709E+00	9.9951402E+01	3.6237320E+00
1	7	3	1	5.4838608E+00	7.2471122E+01	2.3330880E+00
1	8	3	1	1.2847239E+01	8.6102847E+01	8.1473464E+00
1	9	3	1	2.0274396E+00	7.7853581E+01	7.6334825E-01
1	10	3	1	8.6948068E+00	3.4109774E+01	7.0458705E+00
1	11	3	1	2.5478723E+00	2.8521783E+01	1.6737161E+00
1	12	3	1	1.1370564E+01	4.7914121E+01	7.1693099E-01
1	13	3	1	1.8540065E+01	5.7873255E+01	8.2557251E+00

Figure 31.5: This is the bottom or back part of the Mcmc Values report generated by the Bayes Test Data package. This part of the report consists of an image mask and the corresponding parameter values. The mask is one byte per pixel with the horizontal axis being the readout direction and the vertical part being the phase encode. The the ones are the pixels that have a signal, the zero pixels have noise but no signal. So the first nonzero pixel is current Pe of 2 and current Ro of 5. If you look at the list of slices at the bottom the first output is Pe=2 and Ro=5, the numerical values are the actual values used by the program to generate this pixel.



# Appendix A

## Ascii Data File Formats

Ascii data files are used through out the entire Bayesian Analysis software. Often they are used for simple things like input to various packages. Sometimes they are used to loading Abscissa into plotting routines. And sometimes they are used to generate Fid and Image files. In all cases the exact file formats vary depending on the type of data and the package that is loading the Ascii file. This Appendix is a description of these file formats and how they are used. Most of the time the file formats are pretty simple and rather self explanatory and we will list a few of these shortly. However, in a few cases the actual file format of the Ascii data can be complicated. We give these more complicated file formats in this Appendix. For now here is a list of some of the more simple Ascii file formats:

**Images** can be loaded in Ascii. The interface Files/Load Image/Single-Column text file widget expects the data to be single column Ascii with each row in the image stacked one line after another.

**FID** data can be loaded as Ascii data. In this case one complex (real and imaginary) number are expected on each like of the FID. This type of data is used in the Files/Load Spectroscopic Fid/Text File widget.

**Ascii Input Files** for most packages, like an exponential package, allow Ascii data to be loaded. In these simple cases, the files are simply two column Ascii: one abscissa and one data column. However, see below because there are major exceptions to this rule.

### A.1 Ascii Input Data Files

The format of a general input Ascii file used in the Bayesian Analysis software is shown in Fig. A.1. Each line in an input Ascii file consist of three parts. This is illustrated by the double vertical lines in the figure. The first part is a single column plot abscissa, and as its name implies, this is the abscissa used in plotting the Ascii data. All plots of the Ascii data done by the interface are of the data versus the plot abscissa. If there are multiple data columns, then multiple plots are drawn by the interface. However, all of these data plots use the input plot abscissa on the current data set.

The second part of an Ascii input file, is the data and as indicated in Fig. A.1, there can be up to  $N$  data columns, where  $N$  is specified by the model, i.e., it is user defined. The number of these

Figure A.1: Ascii Data File Format

Plot Abscissa	Data Col 1	...	Data Col $N$	Abcissa Col 1	...	Abcissa Col $M$
1	25.1	...	30.2	1	...	3
2	15.1	...	10.2	2	...	-1
$\vdots$	$\vdots$	...	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Figure A.1: This table shows a schematic of a general input Ascii data file. The first column is simply a plot variable and in the case of a single column abscissa, this variable should be the abscissa. The columns labeled “Data Col 1” though “Data Col  $N$ ” are the  $N$  input data columns. The number of these column is implicit in the package model or it is defined explicitly in the model parameter file. Finally, when multiple abscissa’s are present, these abscissa follow the data columns. Here these  $M$  columns are labeled “Abcissa Col 1” through “Abcissa Col  $M$ ”. If there is only a single abscissa, then these abscissa columns are not present.

columns is often implicit in the nature of the package. For example, an inversion recovery data set would require only a single data column. However, a magnetization transfer packages might require real and imaginary data so the data section might have two columns. User defined models can be written that use any number of input data columns. In Fig. A.1 these data columns are labeled as “Data Col 1” through “Data Col  $N$ ” where there are  $N$  data columns.

Finally the computational abscissa section is present only when the package expects multicolumn abscissa. For example, in a diffusion tensor analysis each abscissa consist of a 3 dimensional  $B$  vector, so 3 abscissa are needed. When a model runs that uses a computational abscissa, the abscissa passed to the model program contains as many columns as specified by the model. For example, a diffusion tensor analysis using a  $B$  vector would be a 3 by  $n$  abscissa, where 3 is the number of abscissa columns and  $n$  is the number of data values in the current data set.

In Ascii packages multiple data sets can be loaded and analyzed jointly. Each data set can have a differing number of data values and different abscissa values. However, the number of abscissa columns is fixed for a given analysis. When the posterior probability is computed the model program must generate the model given the current parameters and abscissa. After this information is generated, the posterior probability for the current data set is computed and used in the Bayesian calculations. So while you could have different abscissa values in the different data sets, all data sets must use the same number of abscissa columns.

## A.2 Ascii Image File Formats

The previous section describes the file format for Ascii data files that are used as input to the various packages. However, Ascii images can also be loaded and in this case there are three distinct image formats, one single column, one multicolumn and one  $k$ -space Ascii format.

In the case of single column Ascii images, the images are stacked in the Ascii file one pixel at a time. If the image consists of  $N$  rows by  $M$  columns then the interface expects rows 1 (all  $M$  values) followed by row 2, etc until all  $N$  rows are read. Images can be stacked by slice or element number and you can specify in the popup how they are stacked so the outer loops can be either slice number or array element number.

Multicolumn Ascii files can also be loaded. In this case all elements in a give row are read from record one of the input file. Record 2 corresponds to the second row in the image, etc. The ordering of the images by slice and element is again under user control and can be specified when the images are loaded.

Finally, images can be complex  $k$ -spaced data. In this case, the complex  $k$ -space data is two column Ascii data. Each row of the complex data corresponds to one readout and readouts are stacked one after another. This complex  $k$ -space data is converted into a Varian fid. This fid is then Fourier transformed, phased, and displayed as a complex image. If multiple  $k$ -space images are present, the slice and element orders are again under user control. For  $k$ -space fid data the output from the interface is the real, imaginary and absolute value images.

All images are whether or not they are input as Ascii files,  $k$ -space files, or any other formats are copied into your current working directory. These images are located in BayesHome/WorkDir/images, where “BayesHome” is your current Bayes home directory, “WorkDir” is your current working directory, and “images” is the images subdirectory. Loaded images preserve the name of the name of the input image and duplicates will overwrite the image directory. However, loaded  $k$ -space data are first converted into a Varian fid. This fid is located in the BayesHome/WorkDir/Image.fid subdirectory. After this conversion, the  $k$ -space data is then Fourier transformed and displayed as an image. These images are output with the name “LoadedImage\_Real.4dfp.ima,” “LoadedImage\_Imag.fdfp.ima,” and “LoadedImage\_Abs.4dfp.ima” for the real, imaginary and absolute value images respectively.

In all of these cases, an abscissa file will almost certainly have to be loaded and in the case of  $k$ -space data is required for the conversion. The next section briefly discusses the format of an abscissa file used with images.

### A.3 The Abscissa File Format

If an image is arrayed, the values of the array variable must be specified in an abscissa file before any image processing can be done on the image. Additionally, when  $k$ -space images are loaded, the abscissa must be available to create the Varian fid file. These abscissa values are given in multicolumn Ascii files, one column for each abscissa column. Unusually abscissa are one dimensional like, for example, in an inversion recovery experiment. In which case the abscissa file is a simple single column Ascii file. But sometimes they can be much more complicated, for example, a  $B$  matrix diffusion experiment would have to have a 6 column abscissa, one entry for each of the 6 independent  $B$  values in a diffusion experiment. In all cases the actual meaning of the columns in an abscissa are specified by the model used in processing the image data. For example a diffusion tensor image would have an abscissa consisting of the gradient or  $B$  values used in the experiment. These gradient or  $B$  values could be one, two, etc. up to 6 dimensional depending on the problem. So the abscissa file could be one column, as in diffusion along a single direction, two column, for diffusion in a plane, all the way up to a 6 column abscissa for a full  $B$  matrix diffusion tensor experiment.



## Appendix B

# Markov chain Monte Carlo With Simulated Annealing

Most of the packages in the Bayesian Analysis software use Markov chain Monte Carlo simulations to approximate the Bayesian posterior probability. To understand how a Markov chain can be used to do this, suppose there is some quantity called  $M$ . This quantity could be a set of parameters or it could be a selection of models and the object of the Bayesian calculation is to estimate the parameters or to determine which model best characterizes the data. While these two problems sound very different, they are really one and the same problem. To see this suppose  $M$  is just the decay rate constant in a simple exponential decay, then discrete values of  $M$ , just specify a set of models  $\{M_1, M_2, \dots, M_n\}$  and in parameter estimation we compute the posterior probability for each model, just as we do in model selection. So model selection and parameter estimation are fundamentally the same problem. The major difference is that in parameter estimation problems, the functional form of the models is the same, while it can be different in model selection.

The estimation problems addressed by the various packages are all structurally similar, suppose  $M$  consists of a set of parameters or model indicators,  $M \in \{M_1, M_2, \dots, M_n\}$ , and we wish to compute the posterior probability for an individual parameters  $P(M_j|DI)$  where  $M_j$  is the hypothesis of interest,  $D$  represents all of the data and  $I$  is the prior information. Applying the rules of Bayesian probability theory, the joint posterior probability for all of the parameters is given by

$$P(M_1 \dots M_n|DI) = \frac{P(M_1 \dots M_n|I)P(D|M_1 \dots M_nI)}{P(D|I)} \quad (\text{B.1})$$

which is Bayes' theorem [1]. For those unfamiliar with the rules of Bayesian probability theory, see Chapter 4 for a tutorial on probability theory or consult [31, 3, 11, 33, 61, 32] for more detailed descriptions of probability theory when treated as extended logic.

If we normalize this posterior probability at the end of the calculation, then  $P(D|I)$  can be dropped and one obtains:

$$P(M_1 \dots M_n|DI) \propto P(M_1 \dots M_n|I)P(D|M_1 \dots M_nI). \quad (\text{B.2})$$

Using logical independence and the product rule, the joint prior probability,  $P(M_1 \dots M_n|DI)$ , can

be factored to obtain

$$P(M_1 \dots M_n | DI) \propto P(M_1 | I) \dots P(M_n | I) P(D | M_1 \dots M_n I) \quad (\text{B.3})$$

and it is this joint posterior probability that most of the Markov chain Monte Carlo simulation approximate. To determine exactly how an individual package implements this calculation, see the Chapter describing that package.

If all a Markov chain Monte Carlo simulation did was to approximate this joint posterior probability, it would not be very useful because what is really needed in the Bayesian calculation is not  $P(M_1 \dots M_n | DI)$ , but  $P(M_j | DI)$  where  $P(M_j | DI)$  is computed using the sum rule of probability theory:

$$P(M_j | DI) = \int_{M_1} \dots \int_{M_n} P(M_1 | I) \dots P(M_n | I) P(D | M_1 \dots M_n I) dM_1 \dots dM_n \quad (\text{B.4})$$

where the integrals are over all  $M_i$  except  $j$ . In the Markov chain Monte Carlo simulation this part of the calculation is done using Monte Carlo integration which consists of sampling the joint posterior probability, the integrand, and then using the samples for each parameter as samples from the marginal posterior probability for each parameter separately.

## B.1 Metropolis-Hastings Algorithm

All of the packages that implement their calculations using Markov chain Monte Carlo use the **Metropolis-Hastings** algorithm and you can read more about the Metropolis-Hastings algorithm at the previous Wikipedia link. You can consult the original 1953 paper [45] paper, Gilks et al. have written extensively on Markov chain Monte Carlo as used in Bayesian probability theory [24] and Radford Neal did his dissertation on that subject [46]. Here we will briefly summarize how the Metropolis-Hastings algorithm is used to approximate the Bayesian posterior probabilities.

At their heart all Markov chain simulations are random number generators in which you, the author, can choose the distribution of the random numbers. In the Bayesian calculations done in this software the chosen distribution is the joint posterior probability for all of the parameters and model indicators given the data and the prior information. By running the chain one can sample the joint posterior probability built into the Markov chain simulation. To run a Markov chain Monte Carlo simulation, one must be able to compute Eq. (B.3) for a given set of parameters and model indicators. Here is a very toy version of how one runs a Markov chain Monte Carlo simulation to sample the joint posterior probability:

1. One begins the process of generating a Markov chain by simply sampling the parameter from their valid range. We are going to call these parameters  $M_0$  and the joint posterior probability computed using  $M_0$  will be designated as  $P_0$ .
2. Next propose a new value for one or more of the parameter. Call this new proposed set of parameters  $M_1$ , and compute the joint posterior probability,  $P_1$  using the proposed values.
3. Accept the proposal if  $P_1$  is greater than  $P_0$ . Here accepting the proposal means that you replace  $M_0$  and  $P_0$  by  $M_1$  and  $P_1$  respectively and go back to Step 2.

4. If  $P_1$  is less than or equal to  $P_0$ , then draw a random number,  $r$ , from a uniform  $(0-1)$  random number generator, and if the ratio  $P_1/P_0$  is greater than  $r$ , accept the proposed value of  $P_1$  and  $M_1$ , i.e., replace  $M_0$  and  $P_0$  with  $M_1$  and  $P_1$  and go to Step 2.
5. Otherwise, reject the proposed values. Here rejecting the proposed values simply means going back to Step 2 without replacing  $M_0$  or  $P_0$ .

This simple 5 step procedure is all it takes to generate a Markov chain Monte Carlo simulation. Unfortunately, implementing the calculation in practice, is more of an art than a science and, shortly, we will describe a few of the tricks used in ensuring the calculations work correctly.

There are several major problems with the Markov chain Monte Carlo simulation as described so far: First, it is possible for the simulation to become stuck in local maxima and one would never know it. So one needs a mechanism for detecting simulations that are trapped in local maxima. Second, even if the chain converges correctly its very difficult to tell if the Markov chain has reached a stationary point. One can run a single chain over multiple steps and then look at the path of the simulation, but this is a very unreliable method of testing whether or not a simulation has converged because simulations often deviate from the maximum. And third, with a single chain it is very difficult to adjust the acceptance rate, the number of times a change to a simulation is accepted divided by the total number of times one changed the simulation.

In the following sections we are going to describe how multiple simulations, simulated annealing, killing simulations and adjusting the rate of acceptance in the simulation can be used to generate Markov chains that are highly robust, and almost impossible to get stuck in a local maxima.

## B.2 Multiple Simulations

We do not run a single Markov chain Monte Carlo simulation; rather we run an ensemble of simulations in parallel. Typically, the ensemble is on the order of a few 10's, for example the defaults number of simulations used in the interface is 50, 50 because experience with running multiple simulations indicates that most of the time 50 simulations is enough to explore most parameter spaces; while running fewer increases the risk of nonconvergence and running more usually make things run longer without improving convergence.

For reasons that will become apparent shortly, we initialize the simulations from the prior probability for the parameters. We then run the simulations through a fixed number of steps. Here running a simulation means that we vary the parameters in one simulation, and then either accept or reject the modified simulation based on the prescription given above, Section B.1. This procedure is repeated for each parameter in each simulation and we repeat this procedure at least 25 times for each parameter. So for example if there are 50 simulations, 20 parameters, and each parameter is varied 25 times, the operations count is about  $50 \times 20 \times 25 = 25,000$  operations to bring the ensemble of simulations to equilibrium.

Between annealing steps various statistics are computed from the multiple simulations and these statistics are used to aid in judging convergence. Additionally, the expected value of the logarithm of the likelihood is used in thermodynamic integration, Section C, and the trajectories of each simulation are good visual aids in determining convergence.

### B.3 Simulated Annealing

These simulations are run using simulated annealing. In simulated annealing one introduces an annealing parameter, which we call  $\beta$ , into the calculation of the joint posterior probability, Eq. (B.3). This annealing parameter is introduced by raising the direct probability for the data to the  $\beta$  power:

$$P(M_1 \dots M_n | \beta D I) = P(M_1 | I) \dots P(M_n | I) P(D | M_1 \dots M_n I)^\beta \quad (\text{B.5})$$

where we have modified the notation to indicate that the joint posterior probability is a function of  $\beta$ . It is this modified joint probability density function that is used in the Markov chain Monte Carlo simulation. When  $\beta = 0$ , the likelihood is raised to the power of zero and the data completely disappears from the problem, one is sampling the prior. Consequently, when the Markov chain Monte Carlo simulations are initialized they are initialized from the prior probability for the parameters. Also note, that when  $\beta = 1$  we are sampling the full joint posterior probability for the parameters and model indicators. The annealing parameter,  $\beta$ , is varied from zero to one according to some annealing schedule, discussed shortly.

Typically one starts the simulations with  $\beta = 0$  and runs the simulations until they reach equilibrium. Running the simulations means changing the parameters in a simulation and then accepting or rejecting the change according to the simple perception given earlier, Chapter B.1. For a given value of  $\beta$ , when running the simulations, the posterior probability will increase to an equilibrium point. That is to say, the posterior probability will quit increasing and simply fluctuate about the peak in the posterior probability.

Once the simulations are in equilibrium, we increase the annealing parameter by small amount. This has the effect of knocking the ensemble of simulations out of equilibrium, so we again run the simulations until they reach equilibrium at this new value of  $\beta$ . When the annealing parameter is increased, the likelihood becomes more important and the simulations will begin to cluster around the high likelihood regions. However, because the annealing parameter is still small, the simulations will explore a much larger part of parameter space simply because the likelihood is not let strongly constraining them. As the annealing parameter is increased the likelihood becomes increasingly important and the simulations begin to cluster around increasingly high and higher likelihood regions.

### B.4 The Annealing Schedule

The annealing schedule, the way the annealing parameter  $\beta$  is varied from zero to one, can be something as simple as varying the annealing parameter linearly to something much more elaborate. In earlier versions of the software, a linear annealing schedule was used. In a linear annealing schedule the annealing parameter was given by:

$$\beta = \frac{j}{n} \quad (0 \leq j \leq n) \quad (\text{B.6})$$

where  $n$  is the number of nonzero steps taken in the annealing. This worked well for many problems, but sometimes ran into difficulty when the logarithm of the likelihood is very rapidly changing, because the first tentative steps in simulated annealing can raise the likelihood many hundreds of orders of magnitude and consequently the simulations can fail to local the global maximum of the posterior probability.

In the current version of the software the annealing parameter is adjusted dynamically as follows. The annealing parameter starts at zero, and the simulations are run until they reach equilibrium. Call this step  $n$ . For the next step, the  $n + 1$  step, the annealing parameter is given by

$$\beta_{n+1} = \text{Min}(1, \beta_n + d\beta_n) \quad (\text{B.7})$$

where  $\beta_{n+1}$  is the value of  $\beta$  to be used in the next annealing step. If the minimum number of annealing steps is  $N$ , then  $d\beta$  is given by:

$$d\beta = \text{Min} \left( \frac{1}{\sigma + N}, 1 - \beta \right) \quad (\text{B.8})$$

where  $\sigma$  is the standard deviation of the logarithm of the likelihood computed from the ensemble of simulations. Note that if the standard deviation of the logarithm likelihood is small, then this method of computing  $\beta$  just reduces to Eq. (B.6), i.e., a linear annealing schedule. However, when the simulations are first initialized by sampling the prior probability for the parameters, the standard deviation of the logarithm of the likelihood is usually large, and consequently, the simulations initially move slowly, gaining speed as the simulations converge on the global maxima.

It is this initial slow annealing that allows the multiple Markov chain simulations to explore the parameter space and locate the global maximum of the posterior probability. However, slowing the annealing down at small values of  $\beta$  is not enough to ensure that the simulations reach the global maximum; it is still possible for simulations to become stuck in local maxima.

## B.5 Killing Simulations

As noted, slowing down the annealing for small values of  $\beta$  works very well for giving the simulations time to find the global maxima. However, it is not enough, it is still possible for simulations to become stuck in a local maxima. These trapped simulations must be found and fixed as quickly as possible if the simulations are to reach a stationary point.

Up to now the Markov chain simulations have been described as having multiple chains running in parallel using simulated annealing with an annealing parameter set dynamically based on the standard deviation of the logarithm of the likelihood. Each step in the Markov chain proceeds roughly as follows, the value of the annealing parameter is computed and set for this step in the simulated annealing cycle. Prior to setting to the annealing parameter, simulations should be in an equilibrium. However, incrementing the annealing parameter throws the simulations out of equilibrium and because the simulations are out of equilibrium, we make a number of other modifications to the simulations, we adjust the rate of acceptance, and we kill off a number of simulations.

Between annealing steps, the algorithm doing the Markov chain simulation kills off low probability simulations. To do this the algorithm computes the logarithm of the posterior probability for each simulation. This table of logarithms is then indexed, sorted, and used to replace low probability simulations. In this step typically 10% of the simulations are replaced by higher probability simulations. The program simply takes the lowest probability simulation and then replaces the simulation by one of the simulations having higher probability. The higher probability replacement is chosen by drawing a random number from a Gaussian having a standard deviation that is roughly one third of the total simulations. So when a simulation is replaced, it is replaced by a higher probability simulation, but not necessarily the highest probability simulation.

## B.6 the Proposal

When doing a Metropolis-Hastings Markov chain Monte Carlo simulation one must be careful in proposing new values of a parameter. If the current value of a parameter is  $M_0$ , one proposes a new parameter value  $M_1$  as follows:

$$M_1 = M_0 + \delta M \quad (\text{B.9})$$

where  $\delta M$  is the change that is being made to the parameter. The exact method one obtains this  $\delta M$  doesn't matter except for one propriety that must be enforced. If the probability of moving from  $M_0$  to  $M_1$  is given by  $P(M_1|M_0I)$  then the probability of moving from  $M_1$  to  $M_0$  must be the same:

$$P(M_0|M_1I) = P(M_1|M_0I) \quad (\text{B.10})$$

That is to say jumps in the proposal probability must be symmetric. There are countless modifications and addendum to this rule and you can look at the various references on Markov chain simulations on what these modifications are, but in the calculations implemented in this software package, Eq. (B.10) is the rule implemented using a simple Gaussian proposal. A Gaussian proposal has a number of advantages, for example it is symmetric in its argument and thus automatically satisfies Eq. (B.10). Additionally, a Gaussian has one additional parameter that is important, its standard deviation. By adjusting the size of the standard deviation of the proposal one can control how often the Markov chain transitions from one proposed value to another.

Now one might ask why this is important and the answer is simple, if the proposal is too small the simulation will not explore the parameter space and if the proposal is too large, the change in the posterior probability will be so great that the probability of accepting the change is zero and again the simulations do not explore the parameter space. Consequently, it is important to monitor the size of the proposal and to adjust it between annealing steps to ensure it is neither too small or too large.

The way the programs that implement the Markov chain simulations control the proposal is by keeping track of the acceptance rate for a given parameter. The acceptance rate is simply the ratio of the number of times a proposed parameter was accepted divided by the total number of times one proposed a new parameter value. There is not hard and fast rule on how often one should accept a parameter, but too often or too little are both bad. Additionally, if one is to error, then err on the side of more exploration of the parameter space is probably a good thing. Consequently, the programs that implement these calculations try and keep the acceptance rate between 20 and 30%. If the acceptance rate falls below 20% the proposal is decreased and if the acceptance rate is above 30% the proposal is increased. If it is between 20 and 30% no change is made. There are many addendum that could be added to this description, but it captures what the program actually does. Indeed, there is an output report generated by most of the packages in the Bayesian Analysis Software called an accepted report and that report is available while a package is running and needless to say, its primary output is the current acceptance rate for the various parameters.

## Appendix C

# Thermodynamic Integration

Thermodynamic Integration is a technique used in Bayesian probability theory to compute the posterior probability for a model. As a reminder, if a set of  $m$  models is designated as  $M \in \{1, 2, \dots, m\}$ , then one can compute the posterior probability for the models by an application of Bayes' Theorem [1]

$$P(M|DI) \propto P(M|I)P(D|MI) \tag{C.1}$$

where we have dropped a normalization constant,  $M = 1$  means we are computing the posterior probability for model 1,  $M = 2$  the probability for model 2, etc. The three terms in this equation, going from left to right, are the posterior probability for the model indicator given the data and the prior information,  $P(M|DI)$ , the prior probability for the model given only the prior information,  $P(M|I)$ , and the marginal direct probability for the data given the model and the prior information,  $P(D|MI)$ .

The marginal direct probability for the data given a model can be computed from the joint posterior probability for the data and the model parameters, which we will call  $\Omega$ , given the Model and the prior information

$$P(D|MI) = \int d\Omega P(\Omega|MI)P(D|\Omega MI). \tag{C.2}$$

Unfortunately, the integrals on the right-hand side of this equation can be very high dimensional. Consequently, although we know exactly what calculation must be done to compute the marginal direct probability, in most applications the integrals are not tractable.

The goal is to show how thermodynamic integration can be used to compute the desired posterior probability, taking the logarithm one obtains

$$\log P(M|DI) = \log P(M|I) + \log P(D|MI). \tag{C.3}$$

Note by taking the logarithm, we have essentially switched to a scale in which there is a arbitrary constant, the normalization constant, which we may or may not know. However, by comparing ratio's of of these logarithm of the logarithm of this normalization constant cancels and one can rank models by their logarithm of the odds ration.

Thermodynamic integration is a method of approximating these integrals. One derives this approximation, by introducing an annealing parameter  $\beta$  into the joint posterior probability for the

parameters given the data and the model:

$$P(\Omega|M\beta DI) = \frac{P(\Omega|MI)P(D|\Omega MI)^\beta}{P(D|M\beta I)} \quad (\text{C.4})$$

with

$$P(D|M\beta I) = \int d\Omega P(\Omega|MI)P(D|\Omega MI)^\beta. \quad (\text{C.5})$$

Clearly, this expression is not the calculation we want to do, however it does have two interesting limits. First, when  $\beta = 0$ , the calculation is computing

$$P(D|M, \beta = 0, I) = \int P(\Omega|MI)d\Omega = 1 \quad (\text{C.6})$$

which is just our normalization constant for the prior. Assuming a normalized prior then the above equality holds. Second, when  $\beta = 1$ , then

$$P(D|M, \beta = 1, I) = \int P(\Omega|MI)P(D|\Omega MI)d\Omega \quad (\text{C.7})$$

is the exact calculation we wish to do.

If we take the derivative with respect to  $\beta$  of  $\log P(D|M\beta I)$ , one obtains

$$\frac{d}{d\beta} \log P(D|M\beta I) = \frac{1}{P(D|M\beta I)} \frac{d}{d\beta} P(D|M\beta I). \quad (\text{C.8})$$

Substituting, Eq. (C.5), for  $P(D|M\beta I)$ , and rearranging the integral, one obtains

$$\frac{d}{d\beta} \log P(D|M\beta I) = \int \log P(D|M\Omega I)P(\Omega|MD\beta I)d\Omega \quad (\text{C.9})$$

which is the expected value of the logarithm of the likelihood. Defining

$$\langle \log P(D|MI) \rangle_\beta = \int \log P(D|M\Omega I)P(\Omega|MD\beta I)d\Omega \quad (\text{C.10})$$

for this expectation, one obtains

$$\frac{d}{d\beta} \log P(D|M\beta I) = \langle \log P(D|MI) \rangle_\beta \quad (\text{C.11})$$

and integrating with respect to  $\beta$ , one obtains

$$\log P(D|MI) = \int_0^1 d\beta \langle \log P(D|MI) \rangle_\beta. \quad (\text{C.12})$$

So if we can calculate the integral on the right-side of this equation, it gives us an indirect method of computing the logarithm of the marginal direct probability for data given the model, Eq. (C.2). For more on thermodynamic integration and how to implement these types of calculations see [37, 45, 24, 25] and [46]

In an earlier versions of this Bayesian Analysis software, thermodynamic integration was implemented by varying  $\beta$  between zero and one in uniform steps and a simple sum was used to approximate the integral. As simple as it was, this procedure worked very well. More recently, implementing thermodynamic integration has become a bit more probabilistic because of the use of a nonuniform annealing schedule. Because the values of  $\beta$  are not evenly spaced, approximating the above integral is much harder. However, thermodynamic integration was never used to do model selection in the Bayesian calculations done in this software system. Rather the model selection programs implement model selection by directly sampling the discrete model indicators. Consequently, out model selection does not depend on how one anneals or performs the thermodynamic integration calculation. Thermodynamic integration was so that the user had a simple test to determine which of a selection of models was the most appropriate given the data and the prior information. To facilitate this, whenever an Ascii package runs it computes the expected logarithm of the likelihood and writes it into a file named "Bayes.prob.model" and this file can be viewed in the interface by activating the Text Report named "Probabilities".



## Appendix D

# McMC Values Report

The McMC Values report is the main output report from the various packages that run Markov chain Monte Carlo simulations. The report consists of three parts, the first part is shown in Fig. D.1 This part of the report is essentially a listing of the parameter file that was used when the package was run. These parameters consists of the various parameter settings used to control the Markov chain Monte Carlo simulations, the top part of this listing. The middle part consists of the information about the various prior probabilities and finally the bottom part of the parameter file is any configuration parameters that are set for this particular package. The parameter listings at the top of Fig. D.1 are standardized across all packages. Additionally, when prior probabilities are show, there format is also standardized. For those looking closely at this figure, you may notice that I have reformed this first part of the report to get it to paginate correctly.

The middle part of the Mcmc Values report is shown in Fig. D.2. The middle part of the Mcmc Values report lists the simulation that had maximum posterior probability. Since simulations are essentially defined by their posterior probabilities, the first part of this report shown the posterior probability, the likelihood and the prior probability. The NonLinear parameter estimates are just the parameters that had maximum posterior probability. Note that this is a listing of the parameters which were actually used in the Markov chain Monte Carlo simulation. For packages that use marginalization, beneath the nonlinear parameters are the estimated amplitudes. For packages that do not use marginalization, but treat amplitudes like nonlinear parameters, this section does not occur. When multiple data sets are processed these amplitudes will be repeated one time for each data set. Finally, the last part of this center section is the estimated noise standard deviation for each data set.

The last part of the Mcmc Values report is shown in Fig. D.3. The bottom part of the Mcmc Values report is the heart of this report. Before I discuss the parameter estimates, I want to draw your attention to the top part of this lower section, Fig. D.3, in particular the line containing “Log probability for the Model:”, This line is the expected logarithm of the likelihood computed using thermodynamic integration, see Chapter C for more on how this number is computed. It contains the mean and standard deviation parameter estimates. Each nonlinear parameter is listed at the front. These nonlinear parameters are followed by the amplitudes, and then the estimated noise standard deviation for each data set. The mean values estimates are computed by averaging the value of a parameter from all of the Markov chain Monte Carlo simulations. Similarly, the standard deviation is the standard deviation of the parameter estimates taken from all Markov chain Monte

Figure D.1: The McMC Values Report Header

Parameter File Listing for the Given Exponential package

```

! BayesExpGiven Package
! Created 03-Oct-2011 16:21:18 by larry
!
      Output Dir = BayesOtherAnalysis
    Number Of Abscissa = 1
      Number Of Columns = 1
        Number Of Sets = 1
          File Name = BayesOtherAnalysis/001.dat
            McMC Simulations = 96
              McMC Repeats = 30
                Total Mcmc Samples = 2880
                  Kill Count = 9
                    Minimum Annealing Steps = 101
                      Histogram Type = Binned
                        Outlier Detection = Disabled
                          Number Of Priors = 3

Param Name  Low      Mean      High      Std Dev   Norm      Prior   Ordered  Param Type
Rate        0.000E+00 0.000E+00 2.432E+00 8.109E-01 -3.7417E+00 Gaussian LowHigh   NonLinear
Amplitude  -1.000E+06 0.000E+00 1.000E+06 3.000E+05 -3.6262E+00 Gaussian NotOrdered Amplitude
Constant   -1.000E+06 0.000E+00 1.000E+06 3.000E+05 -3.6262E+00 Gaussian NotOrdered Amplitude

      Package Parameters = 2
        Number of Exp = 2
          Constant = YES

```

Figure D.1: The McMC Value report is the main report from packages that run Markov chain Monte Carlo simulations. The report consists of three parts, the first part is shown here. This part of the report is essentially a listing of the parameter file that was used when the package was run. These parameters consists of the various parameter settings used to control the Markov chain Monte Carlo simulations, the top part of this listing. The middle part consists of the information about the various prior probabilities and finally the bottom part of the parameter file is any configuration parameters that are set for this particular package.

Figure D.2: MCMC Values Report, The Middle

MCMC Values Report for the Given Exponential Package (2 Exp with a Constant)

```

----- Simulation With Maximum Posterior Probability -----
      Probability:  -0.11967275E+03
      Likelihood:   -0.10055787E+03
      Prior:        -0.19114882E+02
Number of Parameters:  2
Number of Derived:    2
Number of Model Vectors: 3
Number of Outliers:   0
Number of Sets:       1

NonLinear Parameter Estimates
  Param#  Name                Value
    1  Rate_1                 1.42948109E-04
    2  Rate_2                 9.95064682E-01

Amplitude Estimate, (Peak Posterior)
  Set#  Amp%  Amplitude_Name  Value
    1    1  Amplitude_1      3.75907093E+01
    1    2  Amplitude_2     -1.25075235E+02
    1    3  Constant          3.76596504E+01

Derived Parameters Estimates
  Derived#  Name                Value
    1  Time_1                 6.99554552E+03
    2  Time_2                 1.00495980E+00

Noise Std Dev Estimates By Set
NoiseStdDev#  Name                Value
    1  NoiseStdDev, Set 1      7.71617382E-01
----- End Simulation With Maximum Posterior Probability -----

```

Figure D.2: This is the back or bottom part of the mcmc values report generated by the Bayes Test Data package. This part of the report consists of an image mask. This mask is one byte per pixel with the horizontal axis being the readout direction and the vertical part being the phase encode. The locations of the ones are the pixels that have a signal, the zero pixels have noise but no signal. So the first nonzero pixel is  $ro=2$  and  $pe=5$ . If you look at the list of slices at the bottom the first output is  $pe=2$  and  $ro=5$ , the numerical values are the actual values used by the program to generate this pixel.

Figure D.3: The MCMC Values Report, The End

	Avg.	Sd.	
The Average Log Posterior Probability Was:	-122.3195	1.04559	
The Average Log Prior Params:	-19.6689	0.46091	
The Average Log Likelihood:	-102.6507	0.95375	
Log Probability for the Model:	-105.1542		
The expected parameter values (mean value of the probability distributions):			
Parameter Description	Mean Value	Std. Dev.	Peak Value
Rate_1	6.70911E-01	2.77481E-01	1.42948E-04
Rate_2	1.08244E+00	1.41600E-01	9.95065E-01
Time_1	6.61852E+00	1.35603E+02	6.99555E+03
Time_2	9.35569E-01	9.17764E-02	1.00496E+00
Amplitude_1, Set 1	-3.53161E+01	3.61905E+01	3.75907E+01
Amplitude_2, Set 1	-8.97797E+01	3.59562E+01	-1.25075E+02
Constant, Set 1	7.52243E+01	2.39747E+00	3.76597E+01
AmpRms.Set.1	1.31971E+02	1.25815E+01	1.35923E+02
NoiseStdDev, Set 1	7.80983E-01	9.35638E-03	7.71617E-01

Figure D.3: The bottom part of the MCMC Values report is the heart of this report. It contains the mean and standard deviation parameter estimates. Each nonlinear parameter is listed at the front. These nonlinear parameters are followed by the amplitudes, and then the estimated noise standard deviation for each data set. The mean values estimates are computed by averaging the value of a parameter from all of the Markov chain Monte Carlo simulations. Similarly, the standard deviation is the standard deviation of the parameter estimates taken from all Markov chain Monte Carlo simulations. The Peak Value is just a repeat of what is shown in the center section of this report and is shown for informational purposes.

Carlo simulations. The Peak Value is just a repeat of what is shown in the center section of this report and is shown for informational purposes.

The McMC values report is meant as a kind of summary of every important aspect of the analysis. As noted the first part of this report contains the setup information. This information includes things like the number of Markov chain simulations run, the number of repeats gathered, the total number of samples gathered and the minimum number of annealing steps. The middle part of the report is a print out of the simulation that had maximum posterior probability. This print out includes probabilities, posterior, prior and likelihood and it includes the parameters that had maximum posterior probability. Note that Markov chain Monte Carlo simulations explore the posterior probability, they do not actually locate the maximum. Consequently, the parameters printed in this section are strictly the parameters that had maximum posterior for all of the simulations gathered in the analysis; they are not strictly the parameters that maximized the posterior. The bottom part of this report contains the mean and standard deviation parameter estimates and it contains an estimate and standard deviation of the various probabilities. Finally, it also includes the expected value of the logarithm of the likelihood. Because each model in each Ascii package is different, this section of the report varies from package to package, and in the case of the Ascii Model packages the report varies with the loaded model. Finally, the bottom part of this report contains the mean and standard deviation estimates.



## Appendix E

# Writing Fortran/C Models

In this Chapter we are going to describe how to write Fortran and C models. We are going to do this primarily for Fortran and we will briefly discuss how to do this in C. Obviously, if you are going to write Fortran or C models to be used by this system, then you must have Fortran and C installed on your system. If this is not the case, there will be nothing of use to you in this Chapter. First, there are two different types of models used in the Ascii packages, one's that marginalize out the amplitudes and those that do not. Conceptually, those that do not marginalize out the amplitudes are easiest to understand. Lets suppose the data are a simple exponential decaying data set that contains a constant offset. The signal equation,  $S(t_i)$  for this data would be simple

$$d_i = S(t_i) + \sigma_i \quad (\text{E.1})$$

$$S(t_i) = M_\infty + (M_0 - M_\infty) * \exp\{-\alpha t_i\} \quad (\text{E.2})$$

where this model is written as an inversion recovery model:  $M_0$  is the amplitude at time  $t = 0$ ,  $M_\infty$  is the amplitude at time  $t = \infty$  and  $\alpha$  is the decay rate constant. For a single exponential model, the data, the  $d_i$ , are a single column of numbers. Similarly, that abscissa, the  $t_i$ , are also a single column of numbers. Finally, there are three parameters  $M_0$ ,  $M_\infty$  and  $\alpha$  that Markov chain Monte Carlo simulations must estimate.

### E.1 Model Subroutines, No Marginalization

The system model, system models are models which ship with the software, which implements a single exponential plus a constant with no marginalization is the ExpOneConst\_NoMarg.f model. Figure E.1 is a copy of ExpOneConst\_NoMarg.f without most of the comments. These were stripped out for no other reason that to make the code fit on a single page. Lines 01 through 12 are the interface to the model subroutines. This interface is exactly the same for every model and we will describe what each of these parameters are shortly. Lines 13 through 25 are the Fortran declarations for the interface, the arguments on the call list, and as such are generally things you should not change. Finally, Lines 27 through 36 are the lines of code written by me to implement the single exponential plus a constant model. Lines 30 to 32 pull out the three parameters of interest and place them in temporary work areas. This is done for readability. The three parameters are the decay

Figure E.1: Writing Models A Fortran Example

```

01  Subroutine Model(CurSet,          ! The current data set number
02 C      NoOfParams,              ! The number of nonlinear parameters
03 C      NoOfDerived,             ! The number of derived parameters
04 C      TotalDataValues,         ! The number of hyper-complex data values
05 C      MaxNoOfDataValues,       ! The largest number of data values in all sets
06 C      NoOfDataCols,           ! The current number of data column
07 C      NoOfAbscissaCols,       ! The current number of abscissa columns
08 C      NoOfModelVectors,       ! The number of model vectors
09 C      Params,                 ! The input model parameters
10 C      Derived,                ! The output derived parameters
11 C      Abscissa,               ! The abscissa values
12 C      Signal)                 ! The output model signal
13  Implicit None
14  Integer,      Intent(In)::  CurSet
15  Integer,      Intent(In)::  NoOfParams
16  Integer,      Intent(In)::  NoOfDerived
17  Integer,      Intent(In)::  TotalDataValues
18  Integer,      Intent(In)::  MaxNoOfDataValues
19  Integer,      Intent(In)::  NoOfDataCols
20  Integer,      Intent(In)::  NoOfAbscissaCols
21  Integer,      Intent(In)::  NoOfModelVectors
22  Real (Kind=8), Intent(In)::  Params(NoOfParams)
23  Real (Kind=8), Intent(Out)::  Derived(NoOfDerived)
24  Real (Kind=8), Intent(In)::  Abscissa(NoOfAbscissaCols,MaxNoOfDataValues)
25  Real (Kind=8), Intent(InOut)::Signal(NoOfDataCols,MaxNoOfDataValues)
26
27  Integer      CurEntry
28  Real (Kind=8) DecayRate1,Amp,Const
29
30  DecayRate1 = Params(1)
31  Amp        = Params(2)
32  Const      = Params(3)
33
34  Do CurEntry = 1, TotalDataValues
35      Signal(1,CurEntry) = Const+Amp*Exp(-DecayRate1*Abcissa(1,CurEntry))
36  EndDo
37
38  Return
39  End

```

Figure E.1: This is an example of a Fortran routine to analyze a single exponential plus a constant without marginalization. The code from line 27 through 37 is what I entered to produce this model. Lines 30 to 32 pull out the three parameters of interest and place them in temporary work areas. This is done for readability. Lines 34 through 36 generate the exponential evaluated at the abscissa values and store the resulting model signal in the output “Signal” vector.

Figure E.2: Writing Models A C Example

```

01 #include <stdio.h>
02 #include <math.h>
03
04 void model_(int *CurSet,
05             int *NoOfParams,
06             int *NoOfDerived,
07             int *TotalDataValues,
08             int *MaxNoOfDataValues,
09             int *NoOfDataCols,
10             int *NoOfAbscissaCols,
11             int *NoOfModelVectors,
12             double Params[],
13             double Derived[],
14             double Abscissa[],
15             double Signal[])
16 {
17
18     int CurEntry;
19     double Rate, AmpZero, AmpInfty;
20
21     Rate    = Params[0];
22     AmpZero = Params[1];
23     AmpInfty= Params[2];
24
25     if (Rate == 0.0)
26         {Derived[0] = 0.0;}
27     else
28         {Derived[0] = 1.0 / Rate;}
29
30     for (CurEntry = 0; CurEntry < *TotalDataValues; CurEntry++)
31         {
32             Signal[CurEntry]=AmpInfty+(AmpZero-AmpInfty)*exp(-Rate*Abscissa[CurEntry]);
33         }
34
35     return;
36 }

```

Figure E.2: This is an example of a CC routine to analyze a single exponential plus a constant without marginalization. However, in this code we have written the model as an inversion recovery model. The code from line 18 through 36 is what I entered to produce this model. Lines 21 to 23 pull out the three parameters of interest and place them in temporary work areas. This is done for readability. Lines 25 through 28 are examples of how one might set a derived parameter, in this case a decay time. Note that care was taken to avoid possible divide by zeros in the event the decay rate constant is allowed to go to zero. Lines 30 through 34 generate the exponential evaluated at the abscissa values and store the resulting model signal in the output “Signal” vector.

rate constant and the two amplitudes. Line 34 is a Fortran loop construct that tells the compiler it is to loop over the code between the “Do” and the ”EndDo”, while doing this the Fortran integer variable ”CurEntry” is varied from 1 to the TotalDataValues in steps of 1. Lines 35 generates the exponential plus the constant at the abscissa value specified by the index “CurEntry”. Finally, as indicated Line 46 terminates the Do loop.

Figure E.2 is CC version of this code. The structure of the code is almost identical to the Fortran. However, note the name of the model is “model\_”. When Fortran compiles a subroutine it automatically appends this underscore to a model name and consequently when writing CC modes, the name must be “model\_”. Lines 21 through 23 pull the parameters out of the input parameter vector and place them in temporary work areas. Lines 26 through 28 are an example of how one might set a derived parameter, in this case the inverse of the decay rate constant, or the decay time. Note that the code is careful to check that the value of the decay rate is not zero, thus avoiding a possible divide by zero. Lines 30 through 33 generate the exponential signal and place them in the signal vector, just as the Fortran does.

## E.2 The Parameter File

Now an interesting question comes up here as to how it is known that parameter 1 is the decay rate constant, and that 2 and 3 are the amplitude and constant? The answer to this is that each model file must be accompanied by a parameter file and that file describes the parameters and their prior probabilities. The order of the parameters in the prior probabilities list defines the order of the parameters in the “Params” vector in the model subroutine. The parameter file associated with this exponential model is shown in Fig. E.3. In general terms the parameter file consists of three parts, the top part defines some structural features of the model, things like how many model vectors, data columns, abscissa columns and number of priors. Normally, these parameters are not used by the model subroutine, but they are used by the package to determine what parameters must be passed to the subroutine. The middle part of the parameter file contains the prior probabilities for each parameter, including prior probabilities for amplitudes that are marginalized. Finally the bottom part of this file contains a list of the derived parameter names. In this example this list has no entries because there are no derived parameters. Here is a brief description of what each line in parameter file does:

**Number of Abscissa** tell the packages how many abscissa columns this model uses. Something as simple as this single exponential plus a constant only uses a single abscissa. However, routines like the diffusion tensor analysis with a “B” matrix take 6 abscissa. As a reminder, when more than one data column or abscissa are present the file format for Ascii data files becomes a bit more complicated, see Chapter A for a description of these files.

**Number of model vectors** tell the packages how many amplitudes are being marginalized from the posterior probability. For models like this one, where no amplitudes are marginalized, the number of model vectors is zero.

**Number of data cols** tell the packages how many data columns must be present in the Ascii data file. data.

**Number of Priors** is the number of input priors on the following lines. In this case 3 priors follow.

Figure E.3: Writing Models, The Parameter File

```

1 Number of Abscissa
0 Number of model vectors
1 Number of data cols
3 Number of Priors
DecayRate 0.001E+00 1.000E+00 1.000E+01 1.000E+00 Positive(e) NotOrdered(ne) NonLinear
MzO      -1.000E+03 0.000E+00 0.000E+00 3.000E+02 Gaussian(e) NotOrdered(ne) NonLinear
MzInfty  0.000E+03 0.000E+00 1.000E+03 3.000E+02 Gaussian(e) NotOrdered(ne) NonLinear
1 Number of Derived parameters
DecayTime

```

Figure E.3: Every model file is accompanied by a parameter file. The parameter file shown here is for the CC model shown in Fig. E.2. Of course, the format of the parameter file is the same for both C and Fortran programs. However, in the exponential decaying signal discussed earlier, the Fortran model did not have any derived parameters while the C version did. The top part of this file contains some configuration parameters. The middle part contains the parameter names and a description of their prior probabilities. These prior probabilities, the three lines starting with “DecayRate” describe the parameter, their ranges and their prior probabilities. The last part is a list of the derived parameters. See the text for a more extensive description of this file.

**DecayRate** is name of the parameter. The name is used to assign output file names to the probabilities. So if you were to run this model, there would be a file with “DecayRate” in the name and that file would contain the posterior probability for the decay rate parameter. The lines starting with the DecayRate are used to define parameters and their prior probabilities. We are going to call these three lines, the prior definitions and we will use this terminology in describing these prior probabilities. The name of this parameter is “DecayRate” and this name is used in the outputs from the packages. However, to define a prior you have to have more than then name, you need the other fields on this line:

**Low** The first number on the prior definitions is the lowest, the smallest, allowed value of the parameter, in this case the decay rate constant. This Low bound is a hard bound and the Markov chain Monte Carlo simulation restricts the decay rate to be greater than or equal to the low value.

**Mean or Peak** is the second number on the prior definitions and it is used as the mean value in a Gaussian prior probability. It is used as the peak value for a positive prior probability and it is used as the parameter value when the prior type is set to parameter. For all other prior types, this parameter is ignored.

**High** is the third number on the prior definition and is the highest value the decay rate parameter is allowed to take on.

**StdDev** is fourth number on the prior definitions is the standard deviation of a Gaussian prior probability and this field, while present on other priors, is not used unless the prior type is Gaussian.

**Positive(e)** this quantity is the type of prior probability that is to be used and may be set by the user. Valid values for the prior type are: Gaussian, Uniform, Parameter, Exponential

and Positive. These prior types are set using a pull down menu, so you cannot set one incorrectly on the interface. The “(e)” tells the interface that the prior type is editable; while “(ne)” indicates the the prior type cannot be changed.

**NotOrdered(ne)** tells the package that this parameter is not an ordered parameter. Valid values are NotOrdered, LowHigh and HighLow. The “ne” tells the interface that this field is Not Editable, i.e., cannot be changed. As with the Positive entry, the “(ne)” can also take on “(e)” meaning the ordering relationship can be changed. When Ordering is used, one must order at least two parameters. Ordering less will result in an error.

**NonLinear** is an indicator that tells the packages that this parameter is to be treated as if it appears in the model in a nonlinear way and as a result the parameter must be varied in the Markov chain Monte Carlo simulation. The other valid values in this field are “Parameter” and “Amplitude”. Parameter tells the packages that this is just a single number given by the mean, and that the Markov chain Monte Carlo simulations do not vary this parameter. Amplitude tells the packages that this parameter is an amplitude and is to be marginalized from the posterior probability.

Note that the second and third prior are clearly amplitude even though they have been declared as NonLinear parameters. When the prior type is declared as “NonLinear” the parameter is simulated in the Markov chain Monte Carlo simulation. However, when a parameter is declared as “Amplitude” the amplitude is marginalized from the posterior probability. Marginal probability density functions are often more sharply peaked than nonmarginal distributions, but when the marginalization was done the amplitudes were integrated from minus to plus infinity, thus the marginal probability can effectively constrain an amplitude to either negative or positive values and for some model this is not appropriate. Consequently, we allow the user to specify the amplitudes when needed.

The Fortran code shown in Fig. E.1 is a Fortran subroutine using a fixed format. The code consists of an interface, Lines 01 through 25, some user parameter declarations, Lines 27 and 28, and the code to generate the exponential decay, Lines 30 through 36. We are going to describe each of these three items separately starting with defining the interface.

### E.3 The Subroutine Interface

In Fortran the interface to a subroutine is often pretty simple, consisting of little more than a list of interface parameters and their definitions. In Fig. E.1 Lines 01 through 12 are the interface. They tell Fortran that this subroutine receives 12 arguments. Here is a description of these arguments and what they are used for:

**CurSet** is a 4 byte signed integer and contains the number of the current data set. The data set number is passed for the simple reason that the model could be data set dependent. If the model is data set dependent, it is up to the user to generate the appropriate signal function for the current data set.

**NoOfParams** is a 4 byte signed integer indicating the number of nonlinear priors specified in the “.params” file. Note that in general this is not the same thing as the number of prior in the .params file. However, for nonmarginalization models, as this one is, the number of parameters is the same as the number of priors. If this number is 5, then there will be 5 values in the Params vector that are used as the NonLinear parameters in the model..

**NoOfDerived** is a 4 byte signed integer containing the number of derived parameters in the model. Note that Derived is specified as an output parameter and dimensioned by NoOfDerived. Because of this dimension, when the model is called, if the number of derived is zero, a one is passed to this routine. This is done simply to avoid run time errors on some system. When a model does not generate any derived parameters, the Derived vector should not be touched or manipulated in any way.

**TotalDataValues** is a 4 byte signed integer containing the number of hyper-complex data values in CurSet. Note that this number is data set dependent and different data sets can have differing number of data values and abscissa values.

**MaxNoOfDataValues** is a 4 byte signed integer containing the maximum number of data values in all data sets. This number is used to dimension the Abscissa and the Signal parameters.

**NoOfDataCols** is the number of data columns in this data set. Usually this is just one. However, for complex data this would be 2 and for more complicated types of data, this could be any number.

**NoOfAbscissaCols** is a 4 byte signed integer similar to the NoOfDataCols except this is the number of abscissa columns. Again this is usually one, but things like diffusion tensors can have 3 or 6 or more depending on the type of data.

**NoOfModelVectors** is a 4 byte signed integer containing the number of declared amplitude parameters in the .params file. For nonmarginalized models, as this one is, the number of model vectors is zero.

**Params** is a real vector containing NoOfParams parameters. Specifically, if the parameter file contains 5 NonLinear parameters, then Params is a vector of 5 parameters, one for each parameter in the parameter file. Note that this comment is true of the packages that use Ascii models, but not necessarily true of all packages that read Ascii files. Note that the code declares this parameter as input, so you must not change the values in this vector. Having said that, you can change this parameter from “Intent(In)” to “Intent(InOut)” and then you can change these parameters. An example of when you might want to do that is when the input parameters are not ordered and the processing requires them to be ordered, so they are sorted in place.

**Derived** is a real output vector of containing space for NoOfDerived parameters. Inside of this subroutine NoOfDerived is always greater than or equal to one and the work area passed to this subroutine contains at least one entry. The reason for this is simply because Derived is declared as an output vector and as a result some Fortran compilers insist that you set its value, even when there are no derived parameters. Consequently, we pass a work area large enough to hold at least one derived parameter and the subroutine can set this parameter to satisfy the Fortran compilers. Derived parameters are a way of outputting functions of the various parameters. For example in this exponential model, one might be interested in the decay time as well as the decay rate. In that case one could compute something like  $\text{Derived}(1) = 1/\text{Params}(1)$  as a derived parameter and the package will output both  $\text{Params}(1)$  and  $\text{Derived}(1)$ .

**Abcissa** is a real input vector containing the abscissa for the current data set. Note that the abscissa is a multicolumn vector the length of the current data set. In the model discussed

here, only a single abscissa is present so the abscissa is referenced as “Abscissa(1,CurEntry)” where the 1 means the first abscissa column and “CurEntry” is the current time or abscissa point. However, for something like a B vector there would be three abscissa, lets call them Bx, By and Bz, then these three abscissa would be referred to by Abscissa(1,CurEntry), Abscissa(2,CurEntry) and Abscissa(3,CurEntry) for each of the three abscissa values and which Abscissa value is Bx, By or Bz is something determined by the user.

**Signal** is an 8 byte vector used to output the model signal. This output signal is a two dimensional vector, the first dimension being the number of data columns, and the second is the maximum number of data values. Please note that the total number of data values and the maximum number of data values are in general different. Consequently, when you generate a signal, you generate a vector containing the number of data values. You do not generate a vector containing the maximum number of data values.

When the packages call a Fortran or C model, they pass the subroutine 12 arguments. Arguments 1 through 11, excluding the derived parameters, are input arguments and are set when the model is executed. And as a general rule, it is unwise to change these values. However, the derived parameters, argument 10, and the signal vector, argument 12, are outputs that must be set by the model subroutine. Note that the signal vector may have multiple columns and if the signal vector is 5 columns, then you must compute all 5 signals for all abscissa vectors.

## E.4 The Subroutine Declarations

Fortran lines 27 and 28 and C lines 18 and 19 are declarations. In Fortran and C, these declarations tell the compiler if a variable is a real, complex or integer number. And how many bytes of storage a variable is to occupy. The various parameters that are passed to the model subroutines are either 4 byte integers are 8 byte real numbers. For those unfamiliar with this terminology a real number is a number in scientific notation.

In Fortran model codes I have written, I use a feature of Fortran called Implicit None. This feature tells Fortran not to assume the type of data based on its naming convention and causes Fortran to look for a declarations of the variable type. In C all variables must be declared so issues of default variable types do not occur. You can see in Fig. E.2 that the declarations are specified at the time the parameters are declared on the argument list. However, in Fortran these declarations are separate and you must declare all variables in any Model routine you write you write. In Fortran, there are three types of declarations that might be used: Integer, Real and Complex. It is my understanding that C does not support a complex definition and you must program you complex arithmetic by hand, this is not true in Fortran. All of your Fortran declarations must go after, line 25, but before the start of any executable code. The order of these declarations can sometimes mater, but usually not. The exception is when you define a parameter used in a dimension, then the parameter must be defined first. Figure E.4 contains some examples of the types of Fortran declarations you might need.

Line 01 declares a 4 byte real number named FourByToNumber and this quantity could be used as a variable in the calculations done in the Model subroutine. Line 02 declares an 8 byte real number named FourByToNumber. Line 03 declares an integer called MyLoopVariable. Line 04 declares a second integer called NoOfParams. Line 05 decalats that NoOfParams is to be assigned the value 4 and this value is a constant that cannot be changed. Line 06 is an example of a single one

Figure E.4: Writing Models Fortran Declarations

```

01   Real (Len=4)   FourByToNumber
02   Real (Len=8)   EightByToNumber
03   Integer       MyLoopVariable
04   Integer       NoOfParams
05   Parameter     (NoOfParams=4)
06   Real (Kind=8) Vector(NoOfParams)
07   Real (Kind=8) Vector2D(2,NoOfParams)
08   Complex(Kind=8)ComplexVector(NoOfParams)

```

Figure E.4: Line 01 declares a 4 byte real number named FourByToNumber. Line 02 declares an 8 byte real number named EightByToNumber. Line 03 declares an integer called MyLoopVariable. Line 04 declares a second integer called NoOfParams. Line 05 decal that NoOfParams is to be assigned the value 4 and this value is a constant that cannot be changed. Line 06 is an example of a single one dimensional vector of containing NoOfParams. Line 07 is an example of a single two dimensional vector of containing NoOfParams entries, but each entry consists of two 8 eight byte variables. Finally, Line 08 is an example of a complex variable. This variable contains NoOfParams complex numbers and because each complex number consists of a real and imaginary part this complex vector takes exactly as much storage as Vector2D and what's more the storage arrangement of these vectors is identical.

dimensional vector of containing NoOfParams. Line 07 is an example of a single two dimensional vector of containing NoOfParams entries, but each entry consists of two 8 eight byte variables. Finally, Line 08 is an example of a complex variable. This variable contains NoOfParams complex numbers and because each complex number consists of a real and imaginary part this complex vector takes exactly as much storage as Vector2D and what's more the storage arrangement of these vectors is identical.

## E.5 The Subroutine Body

The body of the Fortran Model subroutine starts after Line 30 through the end of the file. In the C routine the body starts on line 21 and proceeds to the end of the file. The body of the code is where you put the instructions to compute your model signal. Ignoring the integer variables, you receive as you input the parameters, contained in the "Params" vector and the abscissa and you must compute the model signal for each value of the abscissa and place this value in the signal vector. For example in the single exponential plus a constant model, the output signal vector is a single column vector containing an exponential plus a constant.

Additionally, the body of the code must compute the derived parameters and place them in the derived vector. A derived parameter is some function of the input parameters that the user wishes to output. For example, in this model we process the exponential signal using a decay rate constant. However, we could have used a decay time constant. The two formulations are exactly identical, but the probability density functions for a decay rate are not the same as the probability density functions for a decay time. If we had desired to see the probability density function for the decay time, we could have defined a derived parameter, and by adding a single line of code to the above,

“Derived(1) = 1.0/DecayRate1” we could output both probability density functions.

Any valid executable statement can be in the body of the Fortran or CC code including function, subroutine calls and IO statements. However, with I/O extreme care must be exercised to make sure that you do your IO only a single time and save the results in variables having the save attribute because, your model function will be called millions to times by the Markov chain Monte Carlo simulation.

If there are function or subroutine calls in your model subroutine, these routines must be part of the same physical file. For example, suppose the loaded model is named test.f and it calls a routine named “MyModel,” then test.f must consists of the “test” model subroutine and the “MyModel” subroutine. If your model calls multiple subroutines, then these subroutines must also be a part of the same physical source code. Finally, if you use a model which calls multiple subroutines or functions in a model selection calculation, *then all subroutine and function names called by all of your models must be unique.*

## E.6 Model Subroutines With Marginalization

There are two basic types of model, those which do not marginalize out any amplitudes and those that do. Model routines that marginalize amplitudes are different in their functional requirements. Typically, the relationship between the data and the model is given by:

$$d_k(t_i) = \sum_{j=1}^m A_{jk} G_j(\Omega, t_i) + \sigma_{ki} \quad (\text{E.3})$$

where there are multiple data sets,  $k \in \{1, 2, \dots, n\}$ , and each data set is modeled as the same functional form but having a unique set of amplitudes for each data set. We usually call the functions,  $G_j(\Omega, t_i)$ , model vectors because each  $G_j(\Omega, t_i)$  is an  $N$  dimensional vector in  $t_i$ . Finally, we are using  $\Omega$  to stand for the collection of all of the nonlinear parameters in the model.

When generating a subroutine to process this model using marginalization, it is the  $G_j(\Omega, t_i)$  that must be computed, not the sum, because in a marginal probability distribution there are no amplitudes. Rather one computes the  $G_j(\Omega, t_i)$  and uses these quantities in the calculation for the posterior probability, see Chapter 4 for more on how marginal probabilities are computed. To make this more concrete, suppose we have a single exponential plus a constant model:

$$d(t_i) = M_\infty + (M_0 - M_\infty) \exp\{-\alpha t_i\}. \quad (\text{E.4})$$

This equation is not in the form given by Eq. E.3. However, it is simple to rearrange it into this form:

$$d(t_i) = M_0 \exp\{-\alpha t_i\} + M_\infty (1 - \exp\{-\alpha t_i\}) \quad (\text{E.5})$$

and in this form, the two amplitudes are given by  $A_1 = M_0$  and  $A_2 = M_\infty$  and the two model equations,  $G_1(\Omega, t_i)$  and  $G_2(\Omega, t_i)$  are given by

$$G_1(\Omega, t_i) = \exp\{-\alpha t_i\} \quad (\text{E.6})$$

and

$$G_2(\Omega, t_i) = 1 - \exp\{-\alpha t_i\}. \quad (\text{E.7})$$

Formally, the model now reduces to

$$d(t_i) = A_1 G_1(\Omega, t_i) + A_2 G_2(\Omega, t_i) \quad (\text{E.8})$$

and it these model functions,  $G_1(\Omega, t_i)$  and  $G_2(\Omega, t_i)$ , that must be programmed into a model subroutine using marginalization. As illustrated in Fig. E.5, the two model functions are programmed into the model subroutine. Because it is the model functions, the  $G_j(\Omega, t_i)$ , that must be programmed, the name of the output signal vector has been changed from “Signal” to “Gij”. Note that the dimension of the Gij vector is given by the number of data columns, the maximum number of data values and finally the number of model vectors. This Gij vector must be filled in for each data column, for each data value in the current set and for each model function. Also note that in this model subroutine there is only a single parameter, the decay rate constant; the amplitudes are not present in the parameter vector. Only the parameters that are not marginalized appear in the parameter vector. The order of the parameters in this vector, is the order given in the parameter file. In the parameter file, the amplitude parameters *must* appear after all other parameters. They cannot appear before any nonlinear parameters. If they do appear out of order, the program that implements the calculation will issue an error on the console and in the mcmc.values report and the stop.

In addition to computing the model vectors, the  $G_j(\Omega, t_i)$ , rather than the signal,  $S(t_i)$ , there are also changes in the parameter file, see Fig. E.6. In particular the number of model vectors is now set to 2 because there are two  $G_j(\Omega, t_i)$ . As noted, the nonlinear parameters, in this case the decay rate constant, must come before the amplitudes and again this is illustrated in Fig. E.6. The amplitude parameters are now designated as “Amplitudes” and, finally, the range on the amplitudes is set to a large number covering both negative and positive values. This range is not actually used by the program that implements the calculation, rather the program requires the prior range for an amplitude to be  $(-\infty \leq A_j \leq \infty)$ , because that was the range used when the amplitudes were marginalized from the posterior probability. Consequently, if it is important to impose a prior range on an amplitude, you should use a nonmarginalization routine where you can specify the exact prior probabilities. One last note, in Fig. E.6 the prior type and the ordering relationship for the amplitudes are set to “ne”, not editable, because the prior type and the ordering relationships are built into the calculations and cannot be changed by the users. Even if you tried to set these parameters to “editable”, and even if the interface permits you to do this, the programs that implement the calculation will not use that information.

Figure E.5: Writing Models Fortran Example

```

01      Subroutine Model(CurSet,          ! The current data set number
02      C          NoOfParams,          ! The number of nonlinear parameters
03      C          NoOfDerived,         ! The number of derived parameters
04      C          TotalDataValues,     ! The number of hyper-complex data values
05      C          MaxNoOfDataValues,   ! The largest number of data values in all sets
06      C          NoOfDataCols,       ! The number of data column
07      C          NoOfAbscissaCols,    ! The number of abscissa columns
08      C          NoOfModelVectors,    ! The number of model vectors
09      C          Params,              ! The input parameters
10      C          Derived,             ! The output derived parameters
11      C          Abscissa,            ! The abscissa values
12      C          Gij)                ! The output Gij vector
13      Implicit None
14      Integer,      Intent(In)::  CurSet
15      Integer,      Intent(In)::  NoOfParams
16      Integer,      Intent(In)::  NoOfDerived
17      Integer,      Intent(In)::  TotalDataValues
18      Integer,      Intent(In)::  MaxNoOfDataValues
19      Integer,      Intent(In)::  NoOfDataCols
20      Integer,      Intent(In)::  NoOfAbscissaCols
21      Integer,      Intent(In)::  NoOfModelVectors
22      Real (Kind=8), Intent(In)::  Params(NoOfParams)
23      Real (Kind=8), Intent(Out):: Derived(NoOfDerived)
24      Real (Kind=8), Intent(In)::  Abscissa(NoOfAbscissaCols,MaxNoOfDataValues)
25      Real (Kind=8), Intent(InOut)::Gij(NoOfDataCols,MaxNoOfDataValues,NoOfModelVectors)
26      Integer      CurEntry
27      Real (Kind=8) DecayRate1
28
29      DecayRate1 = Params(1)
30
31      Do CurEntry = 1, TotalDataValues
32          Gij(1,CurEntry,1) = Exp(-DecayRate1*Abscissa(1,CurEntry))
33          Gij(1,CurEntry,2) = 1-Exp(-DecayRate1*Abscissa(1,CurEntry))
34      EndDo
35
36      Return
37      End

```

Figure E.5: This is a single exponential plus a constant model when the amplitudes are marginalization from the posterior probability. As explained in the text, the model equation must be written in the form  $d_i = A_1 G_1(\Omega, t_i) + A_2 G_2(\Omega, t_i) + \dots$ . In a marginalized model, the two model functions,  $G_1(\Omega, t_i)$  and  $G_2(\Omega, t_i)$ , are programmed into the model.

Figure E.6: Writing Models The Parameter File

```

1 Number of Abscissa
2 Number of model vectors
1 Number of data cols
3 Number of Priors
DecayRate1 0.000E+00 1.000E+00 1.000E+01 1.000E+00 Positive(e) NotOrdered(ne) NonLinear
AmpInit    -1.000E+06 0.000E+00 1.000E+06 3.000E+05 Gaussian(ne) NotOrdered(ne) Amplitude
AmpFinal   -1.000E+06 0.000E+00 1.000E+06 3.000E+05 Gaussian(ne) NotOrdered(ne) Amplitude
0 Number of Derived parameters

```

Figure E.6: The parameter file for a model subroutine using marginalization is shown here. The major modification are that the number of model vectors is now 2, one for each of the two model functions shown in Fig. E.5 lines 32 and 33. Additionally, note that the amplitudes vary over a large range, and this range in the programs that implements this calculation is taken to be minus to plus infinity. Also note that the prior type and the ordering relationships are not editable. The marginalization is switched on by specifying the parameter type as a Amplitude and Amplitude parameters must be the last entries in the prior list.



## Appendix F

# the Bayes Directory Organization

When the interface is started for the first time, it will define a “Bayes” directory in the user home directory. This home directory is the default location. This default can be changed on the Settings/Preferences menu. After using the software for a while, the home directory will accumulate various subdirectories and files. Here is a typical example of what is in a typical Bayes Home directory:

<code>Bayes.Predefined.Spec</code>	<code>System.out.txt</code>	<code>exp4</code>
<code>BayesAsciiModels</code>	<code>exp1</code>	<code>exp7</code>
<code>BayesManual.pdf</code>	<code>exp2</code>	<code>plugins</code>
<code>System.err.txt</code>	<code>exp3</code>	<code>resources</code>

Here is a brief description of these files and what is contained in them:

**Bayes.Predefined.Spec** is used by the Metabolite package and contains copies of the system and user modified metabolite files. For a description of the metabolite package and the metabolite file formations, see Chapter 10.

**BayesAsciiModels** is a subdirectory that contains both system and user defined Ascii Models. Each model consists of at least two files, the Fortran or C code defining the model, and a parameter file that specifies among other things the prior probabilities for the parameter in the model, see Chapters 20, 22, 21, 28 and 29. Additionally, if the user happens to use the Magnetization Transfer Kinetics package, then a WaterViscosityTable will also be present in this directory. For details on this file, see Chapter 15.

**BayesManual.pdf** is your default copy of this manual. The default copy of the manual is located in your Bayes directory in home directory. This manual is distributed with the software and the user may download updated copies of this manual. Downloaded copies of the manual are written into the current Bayes Home directory.

**System.err.txt** is a file containing Java error messages and is used by us to assist in diagnosing problems, should they occur.

**System.out.txt** is a file containing Java console messages and is used by us to assist in diagnosing problems, should they occur.

**exp1, exp2, exp4, exp4** and **exp7** are the working directories defined in this Bayes directory.

Working directories are work areas where an analysis is stored while it is being setup, run and analyzed. Working directories can have any names, and are not necessarily prefixed by “exp”.

**plugins** is a directory created and used by Java. Plugins are installed in this directory as needed.

**resources** is a directory containing the Java properties files. This file contains various settings that are remembered by the system. For example, it contains the list of servers, their names, IP address, port numbers, etc.

The Work Directories named `exp1`, `exp2`, etc. are users defined working directories. These directories are used by the interface to contain an analysis. Only a single analysis is contained in a given working directory. However, multiple working directories can be in use at any given time. Just as the Bayes home directory contained a number of files and subdirectories, so too, the individual working directories contain a number of files and subdirectories. However the contents of these subdirectories is very similar. Here is a typical example of what is found inside a working directory:

```
Bayes.model.fid      BayesOtherAnalysis  fid                images
BayesAnalyzeFiles   dir.info            image.fid          model.compile
```

Each of these files and subdirectories are used by the interface for very specific purposes. Here is a very brief discussion of what these files and directories are used for:

**Bayes.model.fid** is written by packages that process fid data. For example, when the frequency finding program is run and a simulated model of the data is being viewed, that fid model is written into the `Bayes.model.fid` directory. This fid directory is in standard Varian fid file format and, consequently, the directory contains a fid, `procpair` and text file.

**BayesAnalyzeFiles** is a subdirectory contains outputs from the frequency finding package named `Bayes analyze`. For more on these files see Chapter 8.1.

**BayesOtherAnalysis** is a subdirectory that contains the inputs and outputs for most Ascii packages, i.e., exponential, enter Ascii, magnetization transfer, miscellaneous and histogram packages all use this directory to store inputs and outputs.

**dir.info** contains the current status of the analysis in this package. When the user leaves a directory for any reason, the interface writes this file and when the user rejoins this working directory the `dir.info` file is used to restore the analysis to its status at the time the user departed.

**fid** contains the current spectroscopic fid loaded into this working directory. This directory is in standard Varian fid file format.

**image.fid** contains the current image fid loaded into this working directory. This directory is in standard Varian image fid file format.

**images** contains the images and Abscissa currently loaded into this working directory. All images contained in this directory are in “4dfp” format, see Chapter G for a description of this file format.

**model.compile** is a subdirectory that is used whenever a model is built by the interface. This directory typically contains the last model built, a compile listing and if the compile was successful the executable.

## Appendix G

# 4dfp Overview

Files stored in the images subdirectory ending in 4dfp.img are 4dfp (4-dimensional floating point) format images. The corresponding file ending in 4dfp.ifh are the image file headers. The 4dfp.img files are purely binary files containing 4 dimensional arrayed images. The images stored in the 4dfp.img file are a single UNIX binary file containing a stack of images. The four dimensions are the array, slice, readout and phase encode. Symbolically, the loops needed to correctly read and store an image on a Big Endian machine are given by:

```
Do CurEle = 1, ArrayDim           ! loop over the array dimension
  Do CurSlice = 1, NoSlices       ! loop over the slice dimension
    Do CurRo = 1, NoRo           ! loop over the readout dimension
      Do CurPe = 1, NoPe         ! loop over the phase encodes

          Read the next 4 byte Big Endian number and place it in "Work"

          Copy "Work" into the image matrix

          Images(CurEle, CurPe, CurRo, CurSlice) = Work

      EndDo
    EndDo
  EndDo
EndDo
```

where ArrayDim is the size of the array, NoSlices is the number of slices, NoRo is the size of the image in the readouts direction and NoPe is the size of the image in the phase encode dimension. The images used in the Bayesian Analysis software are stored in Big Endian format and it does not matter what hardware platform writes the images: images are always written in Big Endian regardless of the hardware. If your reading one of our output images on a Little Endian machine, for example a PC, you must swap the byte order. Failure to do so will give essentially meaningless numbers in your image.

The 4dfp.img file contains the binary image, however that binary cannot be read without first parsing the associated 4dfp.ifh file. The 4dfp.ifh file is a separate image file header and it contains

among other things the dimensions of the images, Fig. G.1 is an example of this file. Most of the items in this header are pretty straight-forward, however we are going to dwell on the matrix size and the scaling factors a bit because unless you understand these you will not be able to correctly display an image. First, the four elements in the matrix size are: [1] the number of pixels in the phase encode (x domain), [2] the number of pixels in the readout (y domain), [3] the number of slices and [4] the size of the array variable. Similarly, the three scaling factors are the image pixels sizes in mm. the three scaling factors are the phase encode, the read out and the slice scaling factors. For our images the byte order will always be “bigendian”. Finally, the two file names are usually fully qualified path names to the appropriate files. Here they have been truncated to get them to paginate correctly.

Figure G.1: Example FDF File Header

```

INTERFILE                :=
version of keys          := 3.3
conversion program       := BayesPhase
program version          := 1.0
name of data file        := Bayes/test/images/LoadedImage_Abs.4dfp.ifh
source data file name    := Bayes.test.data/BayesPhase/image_IR.fid
patient ID               := N/A
date                    := 12-Oct-2011 13:50:29
number format            := float
number of bytes per pixel := 4
orientation              := 2
number of dimensions     := 4
matrix size [1]          := 128
matrix size [2]          := 128
matrix size [3]          := 1
matrix size [4]          := 5
scaling factor (mm/pixel) [1] := 0.15625
scaling factor (mm/pixel) [2] := 0.15625
scaling factor (mm/pixel) [3] := 0.15000000596046448
slice thickness (mm/pixel) := 0.15000000596046448
imagedata byte order     := bigendian
x label                  := Phase Encode (cm)
y label                  := Readout (cm)

```

Figure G.1 This is a example image file header (ifh file type) written by the linear phasing package. The image file header contains the parameters necessary to read a 4dfp.img file. The most critical part of the header are the matrix sizes and the scale factors. The matrix sizes are the phase encode, readout, number of slices and array size respectively. The three scaling factors are the pixel sizes in the images. In the Bayesian Analysis interface the real 4 byte numbers written in the image are always written in Big Endian form. The x and y labels are just text strings used to set the labels on the displayed images. Finally, the name of data file and source data file name are normally fully qualified path names to the appropriate files. Here, these names were truncated to get them to paginate correctly.



## Appendix H

# Outlier Detection

From a Bayesian perspective one must define what is meant by an outlier and then apply the rules of Bayesian probability theory to compute the appropriate posterior probabilities. In the calculations done in the Bayesian Analysis software, all of the Ascii models are of the form

$$d_i = G(\Omega, t_i) + n_i \quad (\text{H.1})$$

where  $d_i$  represents a single data item sampled at time  $t_i$ ,  $G(\Omega, t_i)$  is a model function containing some parameters  $\Omega$  to be estimated and the data are contaminated by additive noise  $n_i$  of some unknown standard deviation  $\sigma$ .

By an outlier we mean that for some times  $t_j$  this model does not apply, rather at these times the data and the model are related by

$$d_j = A_j + n_j \quad (\text{H.2})$$

where  $A_j$  is the value of the outlier at time  $t_j$ . These two models can be combined to obtain a model that includes both the parameters and the outliers:

$$d_i = G(\Omega, t_i)(1 - \delta_i) + \delta_i A_i + n_i \quad (\text{H.3})$$

where  $\delta_i$  is the probability that the  $i$ th data value is an outlier and  $A_i$  is the outlier value.

In practice, we simply assume that  $\delta_j = 0$  for the vast majority of the data, i.e., there are only a few outliers in a given data set. So when the Markov chain Monte Carlo simulations are running they propose a change in the number of outliers. This change could be an increase or a decrease in the number of outliers. If its a decrease, the Markov chain Monte Carlo simulation chooses one of the outliers at random and proposes that it was not an outlier after all. If its an increase, the proposed change is to change the  $\delta_i$  corresponding to the largest residual to one. Here the residual means the difference between the data and the model, and the model is  $G(\Omega, t_i)$  where there is no outlier and its  $A_j$  when there is an outlier. When proposing an outlier, one must propose a value for the  $A_j$  and the obvious choice is to set  $A_j$  to  $d_j$  and then simulate  $A_j$  like any other parameter. Using this proposed model, all of the parameters in the the Markov chain Monte Carlo simulation are varied until they reaches equilibrium. Finally, the proposed updated simulation is accepted or rejected according the the acceptance rules for a Metropolis-Hastings algorithm.

In a typical data set, the number of outliers is zero. Additionally, the computational burden introduced by turning on outlier detection is substantial, often doubling the time required to run an

Figure H.1: The Posterior Probability For The Number of Outliers

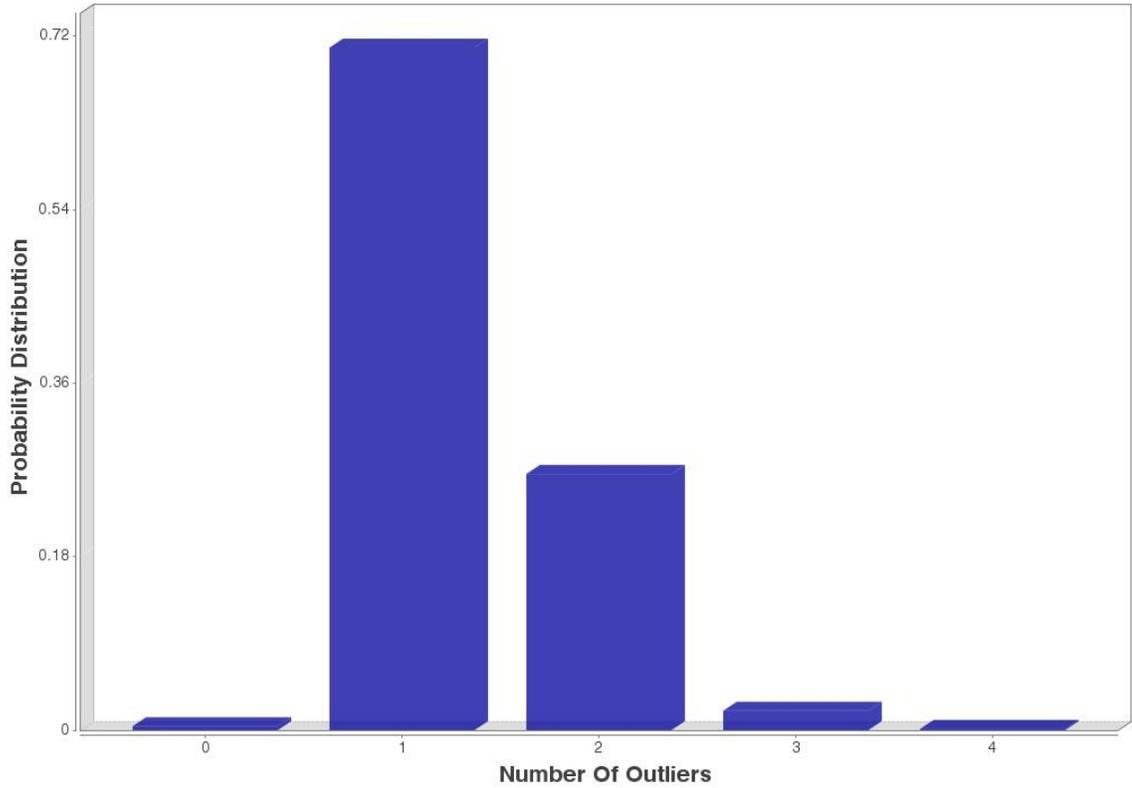


Figure H.1: When outlier detection is enabled in a Markov chain Monte Carlo simulation, a set of extra parameters are added to the model. These parameters include the number, location and value of each outlier. As a result, the posterior probability for the number of outliers is added to the outputs. Additionally, the expected values of these parameters are written into the McMC values report.

analysis. Consequently, we do not look for outliers unless the “Find Outliers” check box is toggled on, i.e., you have pretty good evidence that there is an outlier in the data.

When outlier detection is turned on there is one additional output plot, the posterior probability for the number of outliers. This plot is illustrated in Fig. H.1. The way this posterior probability is generate is to simply count the number of simulations having zero outliers, one, two, etc. and then normalize the posterior by dividing by the total number of simulations. From the plot shown in Fig. H.1, its pretty obvious that about 70% of the simulations had at least one outlier and another 30% had two or three.

When we are computing the means and standard deviations of the output parameters when outlier detection is activated, none of this makes any difference. We simply compute the mean value

of a parameter using all of the McMC simulations and output its value. This is written formally as:

$$\langle \Omega \rangle = \frac{1}{N} \sum_{j=1}^N \Omega_j \tag{H.4}$$

where  $\Omega$  is the parameter being estimates and  $\Omega_j$  is the value of the parameter in the  $j$ th Markov chain Monte Carlo simulation. Suppose that some fraction of the simulations had one outlier, lets call that fraction  $r$ . We are free to split the simulations into two sets, one set of  $(1-r)N$  simulations having zero outliers, and one set of  $rN$  simulations having one outlier. Lets adopt the notation  $\Omega_k$  for the parameter in  $k$ th simulations having no outliers, and  $\hat{\Omega}_j$  for the parameters in  $j$ th simulations having one outlier. Then the expected value of the parameter can be written as:

$$\begin{aligned} \langle \Omega \rangle &= \frac{1}{N} \left[ \sum_{j=1}^{rN} \hat{\Omega}_j + \sum_{k=1}^{(1-r)N} \Omega_j \right] \\ &= \frac{rN}{rN^2} \sum_{j=1}^{rN} \hat{\Omega}_j + \frac{(1-r)N}{(1-r)N^2} \sum_{k=1}^{(1-r)N} \Omega_j \\ &= \frac{rN}{N} \bar{\hat{\Omega}} + \frac{(1-r)N}{N} \bar{\Omega} \\ &= r\bar{\hat{\Omega}} + (1-r)\bar{\Omega} \end{aligned} \tag{H.5}$$

where  $\bar{\hat{\Omega}}$  is the average value of the  $\Omega$  parameter in the simulations having an outlier, and similarly  $\bar{\Omega}$  is the average value of the  $\Omega$  parameter in simulations having no outliers. So the expected value of the parameter is a weighted average of the parameters estimates having zero and one outliers where the weights are just the probability for the number of number of outliers. We mentioned earlier when we computed the average value of a parameter across the simulations that no attempt was made to account for the presence of an outlier, and the reason now becomes obvious. The relative weighting of the various simulations has already been taken care of when the posterior probability for the number of outliers was computed.

In addition to the plot containing the posterior probability for the number of outliers, we also modify the data, model and residual plots. This modification is illustrated in Fig. H.2 where we have placed a large red dot at the location of an outlier. If multiple outliers had been present, the multiple red dot would be present. Note one last thing on this plot, when we compute the model having an outlier, the point at which the outlier is present is the outlier value not the value computed form the model equation. In this example, the model equation is an exponential plus a constant and such a function could never exhibit the dip seen in Fig. H.2.

Figure H.2: The Data, Model and Residual Plot With Outliers

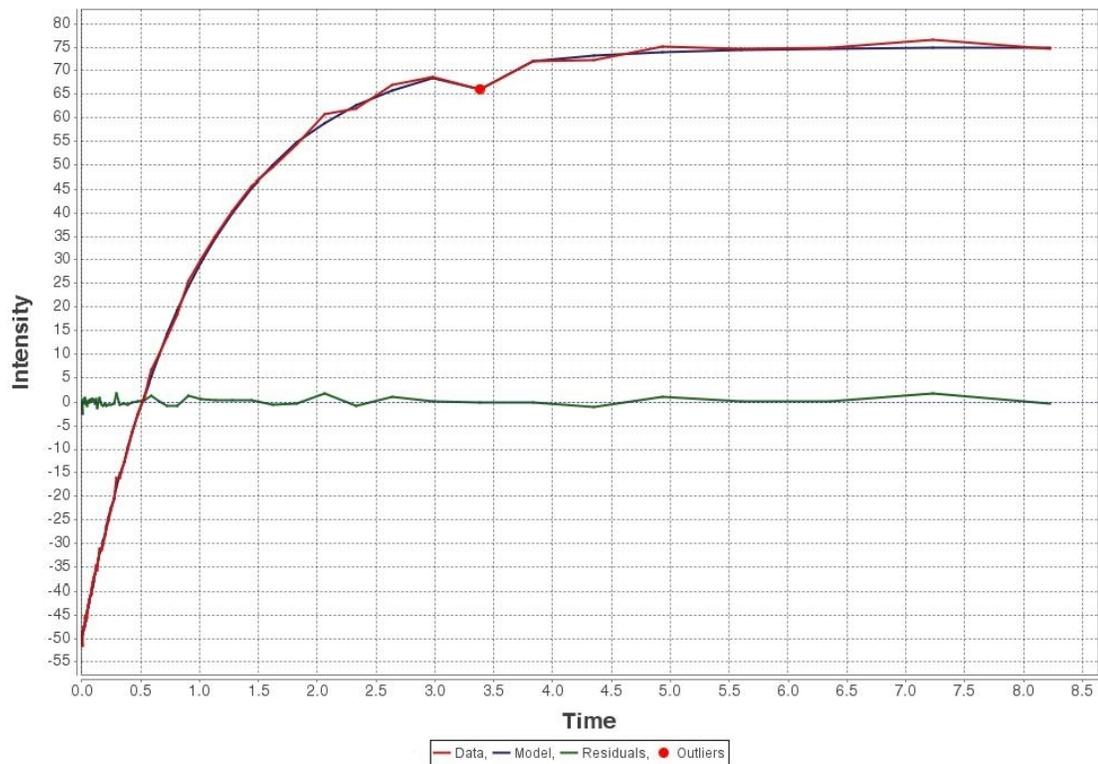


Figure H.2: The Data, Model and Residual plot is also modified when an outlier is present. The modification is pretty simple, at the point where an outlier is present we mark the location with a large red dot. Also note that the outlier value replaces the value computed from the model equation, in this case the exponential model is replaced by the outlier value at the location of the outlier.

# Bibliography

- [1] Rev. Thomas Bayes (1763), “An Essay Toward Solving a Problem in the Doctrine of Chances,” *Philos. Trans. R. Soc. London*, **53**, pp. 370-418; reprinted in *Biometrika*, **45**, pp. 293-315 (1958), and *Facsimiles of Two Papers by Bayes*, with commentary by W. Edwards Deming, New York, Hafner, 1963.
- [2] G. Larry Bretthorst (1988), “Bayesian Spectrum Analysis and Parameter Estimation,” in *Lecture Notes in Statistics*, **48**, J. Berger, S. Fienberg, J. Gani, K. Krickenberg, and B. Singer (eds), Springer-Verlag, New York, New York.
- [3] G. Larry Bretthorst (1990), “An Introduction to Parameter Estimation Using Bayesian Probability Theory,” in *Maximum Entropy and Bayesian Methods*, Dartmouth College 1989, P. Fougère ed., pp. 53-79, Kluwer Academic Publishers, Dordrecht the Netherlands.
- [4] G. Larry Bretthorst (1990), “Bayesian Analysis I. Parameter Estimation Using Quadrature NMR Models” *J. Magn. Reson.*, **88**, pp. 533-551.
- [5] G. Larry Bretthorst (1990), “Bayesian Analysis II. Signal Detection And Model Selection” *J. Magn. Reson.*, **88**, pp. 552-570.
- [6] G. Larry Bretthorst (1990), “Bayesian Analysis III. Examples Relevant to NMR” *J. Magn. Reson.*, **88**, pp. 571-595.
- [7] G. Larry Bretthorst (1991), “Bayesian Analysis. IV. Noise and Computing Time Considerations,” *J. Magn. Reson.*, **93**, pp. 369-394.
- [8] G. Larry Bretthorst (1992), “Bayesian Analysis. V. Amplitude Estimation for Multiple Well-Separated Sinusoids,” *J. Magn. Reson.*, **98**, pp. 501-523.
- [9] G. Larry Bretthorst (1992), “Estimating The Ratio Of Two Amplitudes In Nuclear Magnetic Resonance Data,” in *Maximum Entropy and Bayesian Methods*, C. R. Smith et al. (eds.), pp. 67-77, Kluwer Academic Publishers, the Netherlands.
- [10] G. Larry Bretthorst (1993), “On The Difference In Means,” in *Physics & Probability Essays in honor of Edwin T. Jaynes*, W. T. Grandy and P. W. Milonni (eds.), pp. 177-194, Cambridge University Press, England.
- [11] G. Larry Bretthorst (1996), “An Introduction To Model Selection Using Bayesian Probability Theory,” in *Maximum Entropy and Bayesian Methods*, G. R. Heidbreder, ed., pp. 1-42, Kluwer Academic Publishers, Printed in the Netherlands.

- [12] G. Larry Bretthorst (1999), “The Near-Irrelevance of Sampling Frequency Distributions,” in *Maximum Entropy and Bayesian Methods*, W. von der Linden *et al.* (eds.), pp. 21-46, Kluwer Academic Publishers, the Netherlands.
- [13] G. Larry Bretthorst (2001), “Nonuniform Sampling: Bandwidth and Aliasing,” in *Maximum Entropy and Bayesian Methods in Science and Engineering*, Joshua Rychert, Gary Erickson and C. Ray Smith *eds.*, pp. 1-28, American Institute of Physics, USA.
- [14] G. Larry Bretthorst, Christopher D. Kroenke, and Jeffrey J. Neil (2004), “Characterizing Water Diffusion In Fixed Baboon Brain,” in *Bayesian Inference And Maximum Entropy Methods In Science And Engineering*, Rainer Fischer, Roland Preuss and Udo von Toussaint *eds.*, AIP conference Proceedings, **735**, pp. 3-15.
- [15] G. Larry Bretthorst, William C. Hutton, Joel R. Garbow, and Joseph J.H. Ackerman (2005), “Exponential parameter estimation (in NMR) using Bayesian probability theory,” *Concepts in Magnetic Resonance*, 27A, Issue 2, pp. 55-63.
- [16] G. Larry Bretthorst, William C. Hutton, Joel R. Garbow, and Joseph J. H. Ackerman (2005), “Exponential model selection (in NMR) using Bayesian probability theory,” *Concepts in Magnetic Resonance*, 27A, Issue 2, pp. 64-72.
- [17] G. Larry Bretthorst, William C. Hutton, Joel R. Garbow, and Joseph J.H. Ackerman (2005), “How accurately can parameters from exponential models be estimated? A Bayesian view,” *Concepts in Magnetic Resonance*, 27A, Issue 2, pp. 73-83.
- [18] G. Larry Bretthorst, W. C. Hutton, J. R. Garbow, and Joseph J. H. Ackerman (2008), “High Dynamic Range MRS Time-Domain Signal Analysis,” *Magn. Reson. in Med.*, **62**, pp. 1026-1035.
- [19] V. Chandramouli, K. Ekberg, W. C. Schumann, S. C. Kalhan, J. Wahren, and B. R. Landau (1997), “Quantifying gluconeogenesis during fasting,” *American Journal of Physiology*, **273**, pp. H1209-H1215.
- [20] R. T. Cox (1961), “The Algebra of Probable Inference,” Johns Hopkins Univ. Press, Baltimore.
- [21] André d’Avignon, G. Larry Bretthorst, Marilyn Emerson Holtzer, and Alfred Holtzer (1998), “Site-Specific Thermodynamics and Kinetics of a Coiled-Coil Transition by Spin Inversion Transfer NMR,” *Biophysical Journal*, **74**, pp. 3190-3197.
- [22] André d’Avignon, G. Larry Bretthorst, Marilyn Emerson Holtzer, and Alfred Holtzer (1999), “Thermodynamics and Kinetics of a Folded-Folded Transition at Valine-9 of a GCN4-Like Leucine Zipper,” *Biophysical Journal*, **76**, pp. 2752-2759.
- [23] David Freedman, and Persi Diaconis (1981), “On the histogram as a density estimator:  $L_2$  theory,” *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, **57**, 4, pp. 453-476.
- [24] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter (1996), “Markov Chain Monte Carlo in Practice,” Chapman & Hall, London.

- [25] Paul M. Goggans, and Ying Chi (2004), “Using Thermodynamic Integration to Calculate the Posterior Probability in Bayesian Model Selection Problems,” in *Bayesian Inference and Maximum Entropy Methods in Science and Engineering: 23rd International Workshop*, **707**, pp. 59-66.
- [26] Marilyn Emerson Holtzer, G. Larry Bretthorst, D. André d’Avignon, Ruth Hogue Angelette, Lisa Mints, and Alfred Holtzer (2001), “Temperature Dependence of the Folding and Unfolding Kinetics of the GCN4 Leucine Lipper via  $^{13}\text{C}$  alpha-NMR,” *Biophysical Journal*, **80**, pp. 939-951.
- [27] E. T. Jaynes (1968), “Prior Probabilities,” *IEEE Transactions on Systems Science and Cybernetics*, SSC-4, pp. 227-241; reprinted in [30].
- [28] E. T. Jaynes (1978), “Where Do We Stand On Maximum Entropy?” in *The Maximum Entropy Formalism*, R. D. Levine and M. Tribus *Eds.*, pp. 15-118, Cambridge: MIT Press, Reprinted in [30].
- [29] E. T. Jaynes (1980), “Marginalization and Prior Probabilities,” in *Bayesian Analysis in Econometrics and Statistics*, A. Zellner *ed.*, North-Holland Publishing Company, Amsterdam; reprinted in [30].
- [30] E. T. Jaynes (1983), “Papers on Probability, Statistics and Statistical Physics,” a reprint collection, D. Reidel, Dordrecht the Netherlands; second edition Kluwer Academic Publishers, Dordrecht the Netherlands, 1989.
- [31] E. T. Jaynes (1957), “How Does the Brain do Plausible Reasoning?” unpublished Stanford University Microwave Laboratory Report No. 421; reprinted in *Maximum-Entropy and Bayesian Methods in Science and Engineering* **1**, pp. 1-24, G. J. Erickson and C. R. Smith *Eds.*, 1988.
- [32] E. T. Jaynes (2003), “Probability Theory—The Logic of Science,” edited by G. Larry Bretthorst, Cambridge University Press, Cambridge UK.
- [33] Sir Harold Jeffreys (1939), “Theory of Probability,” Oxford Univ. Press, London; Later editions, 1948, 1961.
- [34] John G. Jones, Michael A. Solomon, Suzanne M. Cole, A. Dean Sherry, and Craig R. Malloy (2001) “An integrated  $^2\text{H}$  and  $^{13}\text{C}$  NMR study of gluconeogenesis and TCA cycle flux in humans,” *American Journal of Physiology, Endocrinology, and Metabolism*, **281**, pp. H848-H856.
- [35] John Kotyk, N. G. Hoffman, W. C. Hutton, G. Larry Bretthorst, and J. J. H. Ackerman (1992), “Comparison of Fourier and Bayesian Analysis of NMR Signals. I. Well-Separated Resonances (The Single-Frequency Case),” *J. Magn. Reson.*, **98**, pp. 483–500.
- [36] Pierre Simon Laplace (1814), “A Philosophical Essay on Probabilities,” John Wiley & Sons, London, Chapman & Hall, Limited 1902. Translated from the 6th edition by F. W. Truscott and F. L. Emory.
- [37] N. Lartillot, and H. Philippe (2006), “Computing Bayes Factors Using Thermodynamic Integration,” *Systematic Biology*, **55** (2), pp. 195-207.

- [38] D. Le Bihan, and E. Breton (1985), “Imagerie de diffusion in-vivo par rsonance,” Comptes rendus de l’Acadmie des Sciences (Paris), **301** (15), pp. 1109-1112.
- [39] N. R. Lomb (1976), “Least-Squares Frequency Analysis of Unevenly Spaced Data,” *Astrophysical and Space Science*, **39**, pp. 447-462.
- [40] T. J. Loredo (1990), “From Laplace To SN 1987A: Bayesian Inference In Astrophysics,” in *Maximum Entropy and Bayesian Methods*, P. F. Fougere (ed), Kluwer Academic Publishers, Dordrecht, The Netherlands.
- [41] Craig R. Malloy, A. Dean Sherry, and Mark Jeffrey (1988), “Evaluation of Carbon Flux and Substrate Selection through Alternate Pathways Involving the Citric Acid Cycle of the Heart by  $^{13}\text{C}$  NMR Spectroscopy,” *Journal of Biological Chemistry*, **263** (15), pp. 6964-6971.
- [42] Craig R. Malloy, Dean Sherry, and Mark Jeffrey (1990), “Analysis of tricarboxylic acid cycle of the heart using  $^{13}\text{C}$  isotope isomers,” *American Journal of Physiology*, **259**, pp. H987-H995.
- [43] Lawrence R. Mead and Nikos Papanicolaou, “Maximum entropy in the problem of moments,” *J. Math. Phys.* **25**, 2404–2417 (1984).
- [44] K. Merboldt, Wolfgang Hanicke, and Jens Frahm (1969), “Self-diffusion NMR imaging using stimulated echoes,” *Journal of Magnetic Resonance*, **64** (3), pp. 479-486.
- [45] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller (1953), “Equation of State Calculations by Fast Computing Machines,” *Journal of Chemical Physics*. The previous link is to the Americain Institute of Physics and if you do not have access to Science Sitations you many not be able to retrieve this paper.
- [46] Radford M. Neal (1993), “Probabilistic Inference Using Markov Chain Monte Carlo Methods,” technical report CRG-TR-93-1, Dept. of Computer Science, University of Toronto.
- [47] Jeffrey J. Neil, and G. Larry Bretthorst (1993), “On the Use of Bayesian Probability Theory for Analysis of Exponential Decay Data: An Example Taken from Intravoxel Incoherent Motion Experiments,” *Magn. Reson. in Med.*, **29**, pp. 642–647.
- [48] H. Nyquist (1924), “Certain Factors Affecting Telegraph Speed,” *Bell System Technical Journal*, **3**, pp. 324-346.
- [49] H. Nyquist (1928), “Certain Topics in Telegraph Transmission Theory,” *Transactions AIEE*, **3**, pp. 617-644.
- [50] William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery (1992), “Numerical Recipes The Art of Scientific Computing Second Edition,” Cambridge University Press, Cambridge UK.
- [51] Emanuel Parzen (1962), “On Estimation of a Probability Density Function and Mode,” *Annals of Mathematical Statistics* **33**, 1065–1076
- [52] Karl Pearson (1895), “Contributions to the Mathematical Theory of Evolution. II. Skew Variation in Homogeneous Material,” *Phil. Trans. R. Soc. A* **186**, 343–326.

- [53] Murray Rosenblatt, "Remarks on Some Nonparametric Estimates of a Density Function," *Annals of Mathematical Statistics* **27**, 832–837 (1956).
- [54] Jeffery D. Scargle (1981), "Studies in Astronomical Time Series Analysis I. Random Process In The Time Domain," *Astrophysical Journal Supplement Series*, **45**, pp. 1-71.
- [55] Jeffery D. Scargle (1982), "Studies in Astronomical Time Series Analysis II. Statistical Aspects of Spectral Analysis of Unevenly Sampled Data," *Astrophysical Journal*, **263**, pp. 835-853.
- [56] Jeffery D. Scargle (1989), "Studies in Astronomical Time Series Analysis. III. Fourier Transforms, Autocorrelation Functions, and Cross-correlation Functions of Unevenly Spaced Data," *Astrophysical Journal*, **343**, pp. 874-887.
- [57] Arthur Schuster (1905), "The Periodogram and its Optical Analogy," *Proceedings of the Royal Society of London*, **77**, p. 136-140.
- [58] Claude E. Shannon (1948), "A Mathematical Theory of Communication," *Bell Syst. Tech. J.*, **27**, pp. 379-423.
- [59] John E. Shore, and Rodney W. Johnson (1981), "Properties of cross-entropy minimization," *IEEE Trans. on Information Theory*, **IT-27**, No. 4, pp. 472-482.
- [60] John E. Shore and Rodney W. Johnson (1980), "Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy," *IEEE Trans. on Information Theory*, **IT-26** (1), pp. 26-37.
- [61] Devinderjit Sivia, and John Skilling (2006), "Data Analysis: A Bayesian Tutorial," Oxford University Press, USA.
- [62] Edward O. Stejskal and Tanner, J. E. (1965), "Spin Diffusion Measurements: Spin Echoes in the Presence of a Time-Dependent Field Gradient." *Journal of Chemical Physics*, **42** (1), pp. 288-292.
- [63] D. G. Taylor and Bushell, M. C. (1985), "The spatial mapping of translational diffusion coefficients by the NMR imaging technique," *Physics in Medicine and Biology*, **30** (4), pp. 345-349.
- [64] Myron Tribus (1969), "Rational Descriptions, Decisions and Designs," Pergamon Press, Oxford.
- [65] P. M. Woodward (1953), "Probability and Information Theory, with Applications to Radar," McGraw-Hill, N. Y. Second edition (1987); R. E. Krieger Pub. Co., Malabar, Florida.
- [66] Arnold Zellner (1971), "An Introduction to Bayesian Inference in Econometrics," John Wiley and Sons, New York.

# Index

- $A_k$  definition, 349
- $H_{j\ell}(t_i)$  definition, 349
- $\lambda_\ell$  definition, 349
- $g_{jk}$  eigenvalue, 349
- Abscissa, **437**
  - Computational, 436
  - Generating, 427
  - Loading, 39
  - Multicolumn, 437
  - Number Of Columns, 458
  - Total Data Values, 456
- Aliases, 113, **126**
- Amplitudes orthonormal definition, 349
- Analyze Image Pixel Package, **411**
  - Modification History, 413
  - Phased Images, 397
  - Reports
    - Bayes Accepted, 413
  - Using, 413
  - Viewers
    - Fortran/C Models, 411
    - Image, 411
    - Prior Probabilities, 413
  - Widgets
    - Abscissa File, 411
    - Build, 411
    - Find Outliers, 411
    - Get Statistics, 413
    - System, 411
    - User, 411
- Analyze Image Pixel Unique Package, **423**
  - Highlight
    - Abscissa, 425
    - Data, 425
  - Input Image
    - Abscissa, 423
    - Data, 423
  - Reports
    - Bayes Accepted, 425
    - Console Log, 425
    - McMC Values, 425
  - Using, 425
  - Viewers
    - Fortran/C Models, 423
    - Image, 423
    - Prior Probabilities, 425
  - Widgets
    - Build, 423
    - Find Outliers, 423
    - Get Statistics, 425
    - System, 423
    - User, 423
- Ascii Data Viewer, **53**
- Assigning Probabilities, **118**
- Bandwidth, **111, 127**
- Bayes Analyze Package, **155**
  - Levenberg-Marquardt , 171
    - Step, 194
  - Algorithm, 175
  - Amplitudes, 197, 198
  - Bayes Model, 159, 161
  - Bayesian Calculations, 167
  - Bruker, 162
  - Build BA Model, 159
  - Covariance, 174
  - Default Parameters Settings, 155
  - Error Messages, 200
  - Fid Model Viewer, 160
  - Interface, 156
  - Likelihood
    - Gaussian, 158
    - Student's  $t$ -distribution, 158

- Log File, [193](#), [195](#)
- Lorentzian lineshape, [161](#)
- Marking Resonances, [157](#)
- Model
  - $J_o$ , [165](#)
  - $J_p$ , [165](#)
  - $J_s$ , [165](#)
  - Amplitude, [163](#), [164](#)
  - Bessel Function, [163](#)
  - Constants Models, [157](#)
  - Correlated, [157](#), [162](#), [164](#)
  - Equation, [161](#), [164](#), [164](#)
  - First Order Phase, [157](#), [162](#), [164](#)
  - First Point, [162](#), [164](#)
  - Gaussian, [163](#)
  - Imaginary Constant, [164](#)
  - Multi-Exponential, [163](#)
  - Multiple Data Sets, [165](#)
  - Multiplet Order, [164](#)
  - Multiplet Orders, [164](#)
  - Multiplets, [162](#)
  - Multiplets of Multiplets, [164](#)
  - Non-Lorentzian, [163](#)
  - Offsets, [162](#)
  - Real Constant, [164](#)
  - Relative Amplitude, [164](#)–[166](#)
  - Resonance Frequency, [165](#)
  - Shim Order, [163](#)
  - Shimming, [166](#)
  - Shimming Order, [164](#)
  - Uncorrelated, [157](#), [162](#), [164](#)
  - Zero Order Phase, [157](#), [162](#), [164](#)
- Model Interface, [160](#)
- Multiplets, [158](#)
- Newton-Raphson, [171](#)
- Noise File, [158](#)
- Noise Standard Deviation, [158](#)
- Outputs
  - Bayes.accepted File, [177](#)
  - bayes.log.nnnn File, [177](#), [193](#), [193](#)
  - bayes.model.nnnn File, [177](#), [185](#), [197](#), [197](#)
  - bayes.noise File, [180](#)
  - bayes.noise.nnnn File, [158](#), [180](#)
  - bayes.output.nnnn File, [176](#), [186](#), [186](#)
  - bayes.params File, [176](#), [177](#)
  - bayes.params.nnnn File, [176](#), [177](#), [177](#)
  - bayes.probabilities.nnnn File, [177](#), [190](#), [190](#)
  - bayes.status.nnnn File, [177](#), [196](#), [200](#)
  - bayes.summary1.nnnn File, [177](#), [198](#), [198](#)
  - bayes.summary2.nnnn File, [177](#), [199](#), [199](#)
  - bayes.summary3.nnnn File, [177](#), [200](#), [200](#)
  - Global Parameters, [182](#), [183](#)
  - Model File, [184](#)
  - Probabilities file, [191](#)
  - Zero Order Phase, [182](#)
- Parameter File
  - Activate Shims, [180](#)
  - Analysis Directory, [178](#)
  - By Fid, [181](#)
  - Data Type, [180](#)
  - Default Model, [181](#)
  - Directory Organization, [180](#)
  - Fid Model Name, [178](#)
  - File Version, [178](#)
  - First Fid, [181](#)
  - First Order Phase, [180](#), [183](#)
  - Imaginary Constant, [184](#)
  - Last Fid, [181](#)
  - lb, [182](#)
  - Maximum Candidates, [182](#)
  - Maximum New Resonances, [182](#)
  - Model Fid Number, [181](#)
  - Model Name, [184](#)
  - Model Names, [181](#)
  - Model Number, [184](#)
  - Model Points, [181](#)
  - Multiplets of Multiplets, [185](#)
  - Noise Start, [181](#)
  - Numerical Parameters, [178](#)
  - Output Format, [180](#)
  - Prior Odds, [182](#)
  - Procpair, [178](#)
  - Real Constant, [184](#)
  - Relative Amplitude, [183](#)
  - Resonance Model, [185](#)
  - Shim Order, [182](#)
  - Spectrometer Frequency, [182](#)
  - Text Parameters, [178](#)
  - Total Complex Data Values, [181](#)
  - Total Data Values, [181](#)
  - Total Sampling Time, [182](#)
  - True Reference, [182](#)

- Units, 180
- Use Noise StdDev, 180
- User Reference, 182
- Prior Probabilities, 167
- Probabilities File, 191
- Product Rule, 168
- Relative Amplitude, 167
- Remove Resonances, 159
- Reports
  - Bayes Status, 155
- Save/Reset, 159
- Search, 166
  - Levenberg-Marquardt , 166
- Short Parameter Description, 195
- Siemens, 162
- Status File, 196
- Steepest Descents, 173
- Sum Rule, 168
- Summary File, 198
- Summary Reports, 176
- Summary2, 199
- Summary3, 201
- Units, 161
- Using, 157
- Varian/Agilent, 162
- Widgets, 155
  - By, 158, 176
  - First Point, 157, 163
  - From, 158, 176
  - Imag Offset, 163
  - Imaginary Offset, 157
  - Mark, 159
  - Max New Res, 157
  - New, 159
  - Noise, 158
  - Phase, 157
  - Primary, 158
  - Real Offset, 157, 163
  - Remove, 159
  - Remove All, 159
  - Reset, 159, 193
  - Restore, 159
  - Save, 159
  - Secondary, 159
  - Shim Order, 157, 163
  - Signal, 158
  - To, 158, 176
- Bayes Find Resonances Package, **239**
  - Bayesian Calculations, 241
  - Current Fid, 239
  - Model Equation, 241
  - Number of data sets, 239
  - Phase Model
    - Automatic, 239, 242
    - Common, 239, 242
    - Independent, 239, 242
  - Prior Probabilities, 243–245
  - Reports
    - Bayes Accepted, 241, 246
    - Condensed, 246
    - Console log, 246
    - MCMC Values, 246
    - Prob Model, 246
  - Using, 239, 241
  - Viewers
    - Fid Data, 240
    - Fid Model, 240, 246
    - File, 246
    - Plot Results, 246
    - Text, 246
  - Widgets
    - Build FID Model, **240, 241, 246**
    - Constant, 239, 242
    - First Trace, 239
    - Last Trace, 239
    - Model Fid Number, 241
    - Phase Model, 239, 242
- Bayes Home Directory, 45, **49**
- Bayes Manual pdf, 469
- Bayes Metabolite Package
  - Widgets
    - Shift Left, 222
    - Shift Right, 222
- Bayes Metabolite Package, **219**
  - Aligning Resonances, 221
  - Bayesian Calculation, 225
  - Metabolite Locations, 221
  - Model Equation, 223
  - Reports
    - Bayes Accepted, 221, 238
    - Condensed, 238
    - Console log, 238

- McMC Values, [238](#)
- Prob Model, [238](#)
- Viewers
  - Fid Data, [219](#)
  - Fid Model, [221](#), [236](#)
  - File, [222](#), [238](#)
  - Metabolite, [221](#)
  - Plot Results, [238](#)
  - Text, [238](#)
- Widgets
  - Fid Model, [221](#)
  - Fid Model Viewer, [221](#)
  - Load System Metabolite File, [219](#)
  - Load System Resonance File, [221](#)
  - Load User Metabolite File, [219](#)
  - Load User Resonance File, [221](#)
  - Shift Left, [221](#)
  - Shift Right, [221](#)
- Bayes Model, [159](#), [159](#)
- Bayes Test Data Package, [427](#)
  - Parameters, [431](#)
  - Reports
    - Bayes Accepted, [428](#)
    - Condensed, [429](#)
    - McMC Values, [429](#), [431–433](#)
  - Viewers
    - Fortran/C Models, [427](#)
    - Image, [428](#)
    - Prior Probabilities, [427](#)
    - Text Data, [430](#)
    - Text Results, [429](#)
  - Widgets
    - # Images, [427](#)
    - # Slices, [427](#)
    - Abscissa, [427](#)
    - ArrayDim, [427](#)
    - Build, [427](#)
    - Get Job, [428](#)
    - Max Value, [427](#)
    - Noise SD, [427](#)
    - Parameter Ranges, [428](#)
    - Pe, [427](#)
    - Ro, [427](#)
    - Run, [428](#)
    - Set (server), [428](#)
    - Status, [428](#)
- Bayes' Theorem, [100](#), [139](#), [145](#), [153](#), [167](#), [211](#), [226](#), [243](#), [252](#), [261](#), [269](#), [278](#), [288](#), [295](#), [306](#), [314](#), [315](#), [317](#), [318](#), [331](#), [333](#), [343](#), [370](#), [399](#), [407](#), [439](#)
- Bayes.accepted
  - Body, [77](#)
  - Header, [76](#)
- Behrens-Fisher Package, [311](#)
  - Bayesian Calculations
    - Derived Probabilities, [320](#)
    - Different Mean And Same Variance, [318](#)
    - Different Mean And Variance, [319](#)
    - Parameter Estimation, [321](#)
    - Same Mean And Different Variance, [317](#)
    - Same Mean And Variance, [315](#)
  - Model Equation
    - Different Mean And Same Variance, [318](#)
    - Different Mean And Variance, [319](#)
    - Same Mean And Different Variance, [317](#)
    - Same Mean And Variance, [315](#)
  - Number of data sets, [311](#)
  - Parameter Listing, [323](#)
  - Prior Probabilities
    - Different Mean And Same Variance, [318](#)
    - Different Mean And Variance, [319](#)
    - Same Mean And Different Variance, [317](#)
    - Same Means And Same Variance, [315](#)
  - Reports
    - Bayes Accepted, [311](#), [322](#)
    - Condensed, [322](#)
    - Console Log, [322](#), [323](#)
    - McMC Values, [322](#), [323](#)
    - Prob Model, [322](#)
  - Using, [311](#)
  - Viewers
    - File, [322](#)
    - Plot Results, [322](#), [324](#)
    - Prior Probabilities, [311](#)
    - Text, [322](#)
  - Widgets
    - None, [311](#)
- Big Endian, [471](#), [473](#)
- Big Magnetization Transfer Package, [259](#)
  - Bayesian Calculations, [259](#)
  - Files
    - Bayes Analyze, [264](#)

- Fid, [263](#)
- Peak Pick, [262](#)
- Model Equation, [261](#)
- Number of data sets, [259](#)
- Prior Probabilities, [261](#)
- Reports
  - Bayes Accepted, [259](#), [262](#)
  - Condensed, [262](#)
  - Console log, [262](#)
  - McMC Values, [262](#)
  - Prob Model, [262](#)
- Using, [259](#)
- Viewers
  - Ascii Data, [259](#)
  - File, [262](#)
  - Prior Probabilities, [259](#)
  - Text, [262](#)
- Widgets
  - Find Outliers, [259](#)
- Big Peak/Little Peak Package, [207](#)
- Bayesian Calculations, [209](#)
- Fid Analyzed, [207](#)
- Model Equation, [210](#)
  - Metabolites, [209](#)
  - Solvent, [210](#)
- Number of data sets, [207](#)
- Prior Probabilities
  - Metabolite, [207](#)
  - Solvent, [207](#)
- Removing Resonances, [207](#)
- Reports
  - Bayes Accepted, [209](#), [216](#)
  - Condensed, [216](#)
  - Console log, [216](#)
  - McMC Values, [216](#)
  - Prob Model, [216](#)
- Using, [207](#)
- Viewers
  - File, [216](#)
  - Model, [209](#)
  - Plot Results, [216](#)
  - Prior Probabilities, [207](#)
  - Text, [216](#)
- Widgets
  - Metabolite, [207](#)
  - Solvent, [207](#)
- Binned Density Function Estimation, [355](#)
- Binned Histogram Package
  - Reports
    - Bayes Accepted, [357](#)
  - Viewers
    - Ascii, [355](#)
- Binned Histograms Package
  - Using, [357](#)
  - Viewers
    - Prior Probabilities, [355](#)
- Bloch-McConnell Equations, [267](#), [277](#)
- Changing the Bayes Home Directory, [469](#)
- Compilers, [29](#)
  - CC, [29](#), [455](#)
  - Fortran, [29](#), [455](#)
- Correlations, [91](#)
- Diffusion Tensor Package, [247](#)
  - Ascii File Formats, [247](#), [254](#), [255](#)
  - Bayesian Calculations, [249](#)
  - Prior Probabilities
    - $\Delta$ , [254](#)
    - $\Gamma$ , [254](#)
    - $\delta$ , [254](#)
    - $\sigma$ , [253](#)
  - Amplitudes, [253](#)
  - Eigenvalues, [253](#)
  - Euler Angles, [253](#)
  - Likelihood, [253](#)
  - Parameter, [254](#)
- Reports
  - Bayes Accepted, [247](#), [255](#)
  - Condensed, [255](#)
  - Console log, [255](#)
  - McMC Values, [255](#)
  - Prob Model, [255](#)
- Symmetries, [253](#)
- Using, [247](#)
- Viewers
  - File, [247](#), [255](#)
  - Plot Results, [255](#)
  - Prior Probabilities, [247](#), [253](#)
  - Text, [255](#)
- Widgets
  - Abscissa Options, [248](#)

- Find Outliers, 247
- Include Constant, 247, 248, 255
- Tensor Number, 247, 248, 255
- Use b Matrix, 255
- Use b Vectors, 255
- Use g Vectors, 254
- Discrete Fourier Transform, 110, 113, 123
- Enter Ascii Model Package, **329**
  - Bayesian Calculations, 332
    - Marginalization, 332
    - No Marginalization, 331
  - Fortran/C Models, 330, 335
  - Model Equation
    - Marginalization, 331
    - No Marginalization, 331
  - Output Names
    - Derived, 335
    - Parameters, 335
  - Reports
    - Bayes Accepted, 331, 335
    - Bayes Params, 335
    - Condensed, 335
    - Console log, 335
    - McMC Values, 335
    - Prob Model, 335
  - Using, 331
  - Viewers
    - Ascii Data, 329
    - File, 335
    - Fortran/C Models, 329
    - Plot Results, 335
    - Prior Probabilities, 329
    - Text, 335
  - Widgets
    - Build, 329
    - Find Outliers, 329
    - System, 329
    - User, 329
- Enter Ascii Model Selection Package, **341**
  - Bayesian Calculations
    - Marginalization, 346
    - No Marginalization, 344
  - Fortran/C Models, 341, 343, 353
  - Model Equation, 343
    - No Marginalization, 343
    - With Marginalization, 347
  - Output Names
    - Derived, 354
    - Parameters, 353
  - Reports
    - Bayes Accepted, 343, 353
    - Condensed, 353
    - Console log, 353
    - McMC Values, 353
    - Params File, 353
    - Prob Model, 353
  - Using, 343
  - Viewers
    - Ascii Data, 341
    - File, 353
    - Fortran/C Models, 341
    - Plot Results, 353
    - Prior Probabilities Not Used, 341
    - Text, 353
  - Widgets
    - Build Not Used, 341
    - Find Outliers, 341
    - System, 341
    - User, 341
- Errors In Variables Package, **303**
  - Ascii File Formats
    - Errors In X and Y Known, 303, 309
    - Errors In X Known, 303, 309
    - Errors In Y Known, 303, 309
    - Errors Unknown, 303, 309
  - Bayesian Calculations, 305
  - Data Error Bars, 303
  - Files
    - Ascii, 303
    - Bayes Analyze, 303
    - Peak Pick, 303
  - Model Equation, 305
  - Number of data sets, 303
  - Reports
    - Bayes Accepted, 305, 309
    - Condensed, 309
    - Console log, 309
    - McMC Values, 309
    - Prob Model, 309
  - Using, 305
  - Viewers

- Ascii Data, [303](#)
  - File, [309](#)
  - Plot Results, [309](#)
  - Text, [309](#)
- Widgets
  - Given Errors In, [303](#)
  - Order, [303](#)
- Exponentials
  - Given Package, [137](#)
  - Inversion Recovery Package, [151](#)
  - Magnetization Transfer Package, [267](#)
  - Unknown Number of Package, [143](#)
- Fid Data Viewer, [53](#)
- Fid Model Viewer, [68](#)
- File Format
  - Ascii, [436](#)
- File Viewer, [80](#)
- Files
  - 4dfp, [59](#), [428](#), [430](#), [470](#), [471](#)
    - Header, [473](#)
    - Reading, [471](#)
  - Abscissa, [39](#), [77](#), [470](#)
  - afh, [53](#)
  - ASCII, [35](#), [36](#)
  - Ascii, [53](#), [54](#), [435](#)
    - k*-space, [437](#)
    - Abscissa, [435](#), [436](#), [437](#)
    - Data, [435](#)
    - Image, [436](#)
  - Bayes Analyze, [36](#)
  - Bayes.accepted, [51](#), [76](#)
  - Bayes.params, [76](#), [79](#)
  - Bayes.prob.model, [447](#)
  - BayesManual.pdf, [469](#)
  - Condensed, [77](#), [78](#)
  - Console.log, [76](#), [79](#), [465](#)
  - dir.info, [470](#)
  - fid, [470](#), [470](#)
    - ASCII, [36](#)
    - ffh, [56](#)
    - Model, [68](#), [70](#)
    - procpa, [470](#)
    - Siemens Raw, [36](#)
    - Siemens Rda, [36](#)
    - Spectroscopic, [53](#)
    - Varian fid, [36](#)
  - Fortran/C Models, [42](#), [455](#), [457](#), [458](#), [465](#)–[467](#)
  - Images
    - 4dfp, [38](#)
    - Binary, [38](#)
    - Bruker 2dseq, [38](#)
    - Bruker stack, [38](#)
    - DICOM, [38](#)
    - FDF, [38](#)
    - Multi-Column Text, [38](#)
    - Siemens IMA, [38](#)
  - k*-space
    - Text, [36](#)
    - Varian fid, [36](#)
  - mcmc.values, [76](#), [449](#)
  - Model Listing, [77](#)
  - prob.model, [76](#)
  - procpa, [470](#)
  - Raw, [36](#)
  - RDA, [36](#)
  - Statistics, [65](#)
  - System.err.txt, [469](#)
  - System.out.txt, [469](#)
  - Varian fid, [36](#)
  - WaterViscosityTable, [469](#)
- Fortran/C Model Viewer, [93](#)
  - Popup Editor, [93](#)
- Fortran/C Models, [42](#), [330](#), [335](#), [353](#), [455](#)
  - Abscissa, [463](#)
  - Body, [463](#)
    - Abscissa, [457](#)
  - Declarations, [462](#)
  - Derived Parameters, [457](#), [459](#), [463](#)
  - Edit/Create New Model, [42](#), [455](#)
  - I/O, [464](#)
  - Marginalization, [464](#)
    - $G_j(\Omega, t_i)$ , [464](#)
    - Amplitude Range, [465](#)
    - Example, [465](#), [466](#)
    - Model Vectors, [465](#)
    - Ordering Amplitudes, [465](#)
    - Parameter File, [465](#), [467](#)
    - Parameter Order, [465](#)
    - Parameters, [465](#)
- Model Files, [455](#)

- Model Selection, 464
- No Marginalization, 457
  - $S(t_i)$ , 455
  - Example, 456
- Parameter File, 458, 459, 465
- Parameters, 463
- Signal, 463
- Subroutine Interface, 460
  - Abscissa, 462
  - Current Set, 460
  - Derived Parameters, 461
  - Maximum No Of Data Values, 461
  - Number Of Abscissa Columns, 461
  - Number Of Data Columns, 461
  - Number Of Derived Parameters, 461
  - Number Of Model Vectors, 461
  - Number Of Parameters, 460
  - Parameters, 461
  - Signal, 462
  - Total Complex Data Values, 461
- Subroutines and Functions, 464
- Frequency Estimation, 114, 132
- Given Exponential Package, 137
  - Bayesian Calculations, 140
  - Files
    - Ascii, 137
    - Bayes Analyze, 137
    - Peak Pick, 137
  - Model Equation, 139
  - Number of data sets, 139
  - Prior Probabilities, 139–141
  - Reports
    - Bayes Accepted, 137, 141
    - Condensed, 141
    - Console log, 141
    - McMC Values, 141
    - Prob Model, 141
  - Symmetries, 141, 148
  - Using, 137
  - Viewers
    - File, 141
    - Plot Results, 141
    - Prior Probabilities, 137, 139
    - Text, 141
  - Widgets
- Constant, 137, 139
- Find Outliers, 137
- Given Order, 27
- Include Constant, 27
- Order, 137, 139
- Given Polynomial Order Package, 285
  - Bayesian Calculations, 288
  - Files
    - Ascii, 285
    - Bayes Analyze, 285
    - Peak Pick, 285
  - Gram-Schmidt, 287
  - Model Equation, 287
  - Number of data sets, 285
  - Prior Probabilities, 289
  - Reports
    - Bayes Accepted, 285, 291
    - Condensed, 291
    - Console log, 291
    - McMC Values, 291
    - Prob Model, 291
  - Scatter Plots, 292
  - Using, 285
  - Viewers
    - File, 290
    - Plot Results, 291
    - Text, 290
  - Widgets
    - Set Order, 285
- Histograms
  - Binned, 381
  - Kernel Density, 381
- Image Model Selection Package, 415
  - Abscissa, 415
  - Fortran/C Models, 415, 417
  - Reports
    - Bayes Accepted, 417
  - Using, 417
  - Viewers
    - Fortran/C Models, 415
    - Image, 415
  - Widgets
    - Noise SD, 415
    - System, 415

- Use Gaussian, 415
- User, 415
- Image Viewer, 59
- Images
  - Flip
    - Horizontal, 63
    - Vertical, 63
  - Grayscale, 63
  - ImageJ, 63
  - Original, 63
- Inversion Recovery Package, 151
  - Bayesian Calculations, 153
  - Model Equation, 153
  - Number of data sets, 153
  - Prior Probabilities, 153
  - Reports
    - Bayes Accepted, 151, 154
    - Condensed, 154
    - Console Log, 154
    - McMC Values, 154
    - Prob Model, 154
  - Using, 151
  - Viewers
    - Plot Results, 154
    - Prior Probability, 151
  - Widgets
    - Find Outliers, 151
- Kernel Density Function Package, 361
  - Ascii File Format, 361
  - Bayesian Calculations, 369
  - Data Requirements, 361
  - Data, Model And Residuals, 369
  - Kernels, 369
    - Biweight, 362
    - Cosine, 362
    - Epanechnikov, 362
    - Exponential, 362
    - Gaussian, 362, 370
    - nonnegative, 361
    - Real Valued, 361
    - Triangular, 362
    - Tricube, 362
    - Triweight, 362
    - Uniform, 362
  - Likelihood, 371
  - Number of data sets, 364
  - Plots
    - Expected Density Function, 367, 368
    - Mean Density Function, 367, 368
    - Posterior Probability for the Kernel Type, 365
    - Posterior Probability for the Number Of Kernels, 366
    - Scatter Plots of Model Averaged Density Function, 368
    - Standard Deviation of the Mean Density Function, 367, 368
  - Prior Probabilities
    - Kernel Center, 371
    - Kernel Smoothing Parameter, 371
    - Kernel Type, 370
    - Number Of Kernels, 370
  - Reports
    - Bayes Accepted, 364
    - Condensed, 372
    - McMC Values, 372
    - Prob Model, 372
  - Using, 364
  - Viewers
    - Ascii, 361
  - Widgets
    - Kernel Type, 364
    - Output Size, 364
- Levenberg-Marquardt, 171
- Linear Phasing Package, 395, 409
  - Interface, 397
  - Model Equation, 398
  - Widgets
    - cf, 403
    - Display, 403
    - Display Array Element, 403
    - fn, 403
    - fn1, 403
    - Image Type, 402
    - Load An Image, 402
    - np, 403
    - nv, 403
    - Process, 403
  - Load Working Directory, 33
  - Logical Independence, 117

- Magnetization Transfer Kinetics Package, **275**
  - Arrhenius Plot, **281**
  - Bayesian Calculation, **278**
  - Boltzmann's Constant, **277**
  - Eyring Equation, **275, 276, 277, 280**
  - Model Equation, **277**
  - Plank's Constant, **277**
  - Prior Probabilities, **279**
  - Reports
    - Bayes Accepted, **277, 281**
    - Condensed, **281**
    - Console log, **281**
    - McMC Values, **281**
    - Prob Model, **281**
  - Sum and Difference Variables, **280**
  - Transmission coefficient, **277**
  - Universal Gas Constant, **277**
  - Using, **277**
  - van't Hoff Plot, **281**
  - Viewers
    - Ascii File, **275**
    - File, **281**
    - Prior Probabilities, **275**
    - Text, **281**
  - Widgets
    - Load, **275, 281**
    - Set, **275**
    - Uncertainty, **275**
- Magnetization Transfer Package, **265**
  - Bayesian Calculations, **267**
  - Files
    - Ascii, **265**
    - Bayes Analyze, **265**
    - Inversion Recovery, **272**
    - Peak Pick, **265**
  - Model Equation, **267**
  - Number of data sets, **265**
  - Prior Probabilities, **265, 270**
  - Reports
    - Bayes Accepted, **267, 272**
    - Condensed, **272**
    - Console log, **272**
    - McMC Values, **272**
    - Prob Model, **272**
  - Three Column Data, **265**
  - Using, **267**
- Viewers
  - Ascii Data, **265**
  - Fid Data, **272**
  - File, **271**
  - Plot Results, **262, 272, 281**
  - Prior Probabilities, **265**
  - Text, **271**
- Widgets
  - Find Outliers, **265**
- Marginalization, **100**
  - Bayes Analyze Package, **174**
  - Behrens-Fisher, **315**
  - Big Magnetization Transfer, **261**
  - Big Peak/Little Peak, **211**
  - Diffusion Tensors, **252**
  - Enter Ascii Model Package, **331**
  - Errors In Variables, **306**
  - Fortran/C Models, **464**
  - Given Exponential, **139**
  - Inversion Recovery, **153**
  - Linear Phasing, **399**
  - Magnetization Transfer, **269**
  - Magnetization Transfer Kinetics, **278**
  - Metabolic Analysis, **225**
  - Nonexhaustive Hypotheses, **101**
  - Nuisance Hypotheses, **100**
  - Nuisance Parameter, **100**
  - Unknown Number of Exponentials, **146**
- Markov chain Monte Carlo, **132, 439**
  - Acceptance Rate, **444**
  - Annealing Schedule, **91, 442**
    - Dynamic, **443**
    - Linear, **442**
  - Killing Simulations, **443**
  - Maximum Posterior Probability, **91**
  - Metropolis-Hastings, **439**
  - Mixing, **91**
  - Monte Carlo Integration, **440**
  - Multiple Simulations, **441**
  - Posterior Probability, **440**
  - Random Number Generators, **440**
  - Repeats, **91**
  - Sampling, **91**
  - Simulated Annealing, **442**
  - the Proposal, **444**

- MaxEnt Density Function Estimation Package, **373**
  - Data Requirements, **381**
  - Plots
    - Contour/Scatter, **375, 379**
    - Number Of Multipliers, **375, 378**
  - Reports
    - Bayes Accepted, **375**
    - Console Log, **375**
  - Using, **375**
  - Viewers
    - Ascii, **373**
    - Plot, **375, 378**
    - Prior Probabilities, **373**
  - Widgets
    - Histogram Size, **373**
    - Order, **373**
- Maximum Entropy Method Of Moments, **102, 377, 381**
  - Advantages, **386**
  - Problems, **386**
  - Review, **381**
- Maximum Entropy Method Of Moments Package
  - Bayesian Calculations, **387**
  - Plots
    - Data, Model and Residuals, **380**
- Menus
  - Files, **24, 35**
    - 4dfp, **37, 38**
    - Abscissa, **35, 39**
    - ASCII, **35, 36**
    - Binary, **38**
    - Bruker, **37**
    - Bruker 2dseq, **38**
    - Bruker Stack, **38**
    - DICOM, **37, 38**
    - FDF, **37, 38**
    - fid, **36, 37**
    - General Binary, **37**
    - Images, **35**
    - Import Working Directories in Batch, **40**
    - Import Working Directory, **40**
    - Load Images, **36, 37, 59**
    - Load Working Directory, **35**
    - Multi-Column Text, **37, 38**
    - Save Working Directory, **35, 39**
  - Siemens IMA, **37, 38**
  - Single-Column Text, **38**
  - Spectroscopic Fid, **35**
  - Test Data, **35, 39**
  - Text k-space, **36**
  - Text k-space fid, **37**
  - User Manual, **35, 39**
- Help, **24**
- Packages, **22, 24, 33, 40**
- Settings, **46**
  - Add Server, **48**
  - Auto Configure Server, **48**
  - MCMC Parameters, **24, 46, 48**
  - Min Annealing Steps, **48, 48**
  - Port number, **48**
  - Preferences, **49, 63**
  - Remove Server, **48, 49**
  - Repetitions, **46, 48**
  - Server Name, **48**
  - Server Setup, **24, 26, 48**
  - Set Window Size, **49**
  - Simulations, **46, 48**
  - View Server Installation Info, **48, 49**
- Spectroscopy fid, **36**
- Utilities, **24, 50**
  - Memory Monitor, **50**
  - Software Updates, **50**
  - System Information, **50**
- WorkDir
  - Creating, **22, 33, 46**
  - Deleting, **22, 33, 46**
  - List, **24, 46**
  - Loading, **46**
  - Name, **46**
  - Popup, **47**
- Model Comparison
  - Big Peak/Little Peak Package, **211**
- model orthonormal definition, **349**
- Mouse
  - Control-left, **59**
  - Fid Data Viewer
    - Left, **56**
    - Right, **56**
  - Shift-left, **59**
- Multiplets
  - J-Coupling

- Center, [159](#)
- Primary, [159](#)
- Secondary, [159](#)
- Newton-Raphson, [171](#)
- Noise Standard Deviation, [64](#)
- Non-Linear Phasing Package, [405](#)
  - Calculations, [407](#)
  - Model Equation, [405](#), [407](#)
  - Widgets
    - Process, [409](#)
    - Write Ascii images, [409](#)
    - Write imaginary images, [409](#)
- Nuisance Parameter, [100](#), [115](#), [135](#)
- Nyquist Critical Frequency, [111](#), [127](#)
- orthonormal, [349](#)
- Outliers, [475](#)
  - Mean Parameter, [477](#)
  - Model, [475](#)
  - Prob Number of, [476](#)
  - Proposal, [475](#)
  - Red dot, [477](#)
  - Weighted Average, [477](#)
- Packages
  - Analyze Image Pixel Unique, [423](#)
  - Bayes Analyze, [20](#), [43](#), [57](#), [155](#), [200](#)
  - Bayes Find Resonances, [21](#), [239](#)
  - Bayes Test Data, [427](#)
  - Behrens-Fisher, [21](#), [44](#), [311](#)
  - Big Magnetization Transfer, [20](#), [43](#), [259](#)
  - Big Peak/Little Peak, [20](#), [43](#), [207](#)
  - Binned Density Function Estimation, [355](#)
  - Binned Histograms, [21](#), [44](#)
  - Diffusion Tensors, [20](#), [40](#), [247](#)
  - Enter ASCII Model, [42](#)
  - Enter Ascii Model, [20](#), [329](#)
  - Enter ASCII Model Selection, [42](#)
  - Enter Ascii Model Selection, [20](#), [341](#)
  - Errors In Variables, [21](#), [44](#), [303](#)
  - Find Resonances, [43](#)
  - Given Exponential, [20](#), [40](#), [137](#)
  - Given Polynomial Order, [285](#)
  - Image Model Selection, [415](#)
  - Image Pixel, [21](#), [45](#), [411](#)
  - Image Pixel Model Selection, [22](#), [45](#)
  - Inversion Recovery, [20](#), [40](#), [151](#)
  - Kernel Density Function, [361](#)
  - Linear Phasing, [21](#), [44](#), [395](#)
  - Magnetization Transfer, [20](#), [42](#), [265](#)
  - Magnetization Transfer Kinetics, [20](#), [43](#), [275](#)
  - Maximum Entropy Method Of Moments, [21](#), [44](#), [373](#)
  - Metabolic Analysis, [21](#), [43](#), [219](#)
  - Non-Linear Image Phasing, [21](#), [45](#), [405](#)
  - Polynomials
    - of Given Order, [21](#), [44](#)
    - of Unknown Order, [21](#), [44](#)
  - Test ASCII Model, [42](#)
  - Test Ascii Model, [20](#), [337](#)
  - Unknown Number of Exponentials, [20](#), [40](#), [143](#)
  - Unknown Polynomial Order, [293](#)
- Parameter File, [42](#)
- Number Of
  - Abscissa, [458](#)
  - Data Columns, [458](#)
  - Model Vectors, [458](#)
  - Priors, [458](#)
- Prior Probability, [459](#)
  - Amplitude, [460](#)
  - High, [459](#)
  - Low, [459](#)
  - Mean, [459](#)
  - NonLinear, [460](#)
  - Ordered, [460](#)
  - Parameter File, [459](#)
  - Peak, [459](#)
  - Prior Type, [460](#)
  - Standard Deviation, [459](#)
- Phase Cycling, [162](#)
- Plot Results Viewer, [71](#)
- Plots
  - Data and Model, [81](#)
  - Data, Model and Residuals, [81](#)
  - Expected Log Likelihood, [88](#)
  - Logarithm of the Posterior Probability, [91](#)
  - Maximum Entropy Histogram, [84](#)
  - Maximum Entropy Histograms, [83](#)
  - McMC Samples, [83](#), [85](#)
  - Parameter Vs Posterior Probability, [86](#), [87](#)

- Posterior Probability, [82](#)
- Posterior Probability Vs Parameter Value, [86](#)
- Residuals, [81](#)
- Scatter, [88](#), [91](#)
- png graphics, [59](#)
- Posterior Probability Vs Parameter Value, [86](#)
- Power Spectrum, [112](#), [123](#), [124](#)
- Prior Probabilities
  - Bayes Phase, [399](#)
  - Big Magnetization Transfer, [261](#)
  - Big Peak/Little Peak, [212](#)
  - Diffusion Tensor, [253](#)
  - Enter Ascii Model, [331](#), [333](#)
  - Errors In Variables, [306](#)
  - Magnetization Transfer, [269](#)
  - Magnetization Transfer Kinetics, [279](#)
  - Non-Linear Phasing Package
    - A, [408](#)
    - $\theta$ , [408](#)
- Prior Probability, [42](#), [65](#), [65](#)
  - Exponential, [67](#), [459](#)
  - Gaussian, [67](#), [104](#), [106](#), [459](#)
  - Jeffreys', [118](#)
  - Normalization Constant, [67](#)
  - Parameter, [68](#), [459](#)
  - Positive, [68](#), [460](#)
  - Uniform, [67](#), [103](#), [118](#), [459](#)
- Prior Viewer, [65](#), [93](#)
- Probabilities
  - Expected Log Likelihood, [453](#)
  - Likelihood, [453](#)
  - Posterior, [453](#)
  - Prior, [453](#)
- Product Rule, [99](#), [119](#), [344](#), [439](#)
- Referencing
  - Setting, [59](#)
- Reports
  - Accepted File, [76](#)
  - McMC Values File
    - General Description, [449](#)
    - Maximum Posterior Probability Simulations, [451](#)
    - Mean Values, [452](#)
    - Prior, [450](#)
  - Standard Deviations, [453](#)
- Restoring An Analysis, [22](#), [35](#), [40](#)
- ROI
  - Expanding, [63](#)
  - Pixels, [63](#)
  - Point, [62](#)
  - Polygon, [62](#)
  - Square, [62](#)
- Saving An Analysis, [35](#), [39](#)
- Schuster Periodogram, [112](#), [123](#)
- Screen Captures, [49](#)
- Settings
  - httpd server, [19](#)
- Software
  - Bayes Account, [29](#)
  - CC, [29](#)
  - Fortran, [29](#)
  - Installation, [29](#)
  - javaws, [29](#)
  - OS requirements, [29](#)
  - root requirements, [30](#)
- Start Up Window, [22](#), [33](#)
- Steepest Descents, [173](#)
- Subdirectories, [469](#)
  - Bayes, [39](#)
  - Bayes.model.fid, [470](#)
  - Bayes.Predefined.Spec, [469](#)
  - Bayes.test.data, [39](#)
  - BayesAnalyzeFiles, [470](#)
  - BayesAsciiModels, [93](#), [469](#)
  - BayesOtherAnalysis, [35](#), [73](#), [470](#)
  - fid, [36](#), [53](#)
  - images, [36](#), [38](#), [39](#), [59](#), [470](#)
  - model.compile, [470](#)
  - plugins, [470](#)
  - Properties, [470](#)
  - Resources, [470](#)
  - Spectroscopic
    - fid, [470](#)
  - Working Directories, [470](#)
- Subroutine Names, [464](#)
- Sufficient Statistics, [122](#)
  - Definition, [105](#)
  - Location Parameter, [108](#)
- Sum Rule, [100](#), [119](#), [344](#), [440](#)

- Test Ascii Model Package, **337**
  - Reports
    - Bayes Accepted, **339**
    - Mcmc Values, **339**
  - Using, **339, 428**
  - Viewers
    - Ascii Data, **337**
    - Fortran/C Models, **337**
    - Prior Probabilities, **337**
  - Widgets
    - Build, **337**
    - Find Outliers, **339**
    - System, **337**
    - User, **337**
- Thermodynamic Integration, **445, 449**
- Uninstall, **49**
- Unknown Number of Exponentials Package, **143**
  - Bayesian Calculations, **145**
  - Model Equation, **145**
  - Reports
    - Bayes Accepted, **143, 148**
    - Condensed, **148**
    - Console Log, **148, 149**
    - McMC Values, **148**
    - Prob Model, **148**
  - Using, **143**
  - Viewers
    - File, **148**
    - Plot Results, **149, 150**
    - Prior, **143**
    - Text, **148**
  - Widgets
    - Constant, **143**
    - Find Outliers, **143**
    - Order, **143**
- Unknown Polynomial Order Package, **293**
  - Bayesian Calculations, **295**
  - Files
    - Ascii, **293**
    - Bayes Analyze, **293**
    - Peak Pick, **293**
  - Model Equation, **295**
  - Number of data sets, **293**
  - Reports
    - Bayes Accepted, **293, 299**
    - Condensed, **299**
    - Console Log, **298, 299**
    - McMC Values, **299**
    - Polynomial Order Plot , **301**
    - Prob Model, **299**
  - Using, **293**
  - Viewers
    - File, **299**
    - Text, **299**
  - Widgets
    - Set Order, **293, 294**
    - Unknown Order, **293, 294**
- Viewers, **27, 52**
  - ASCII Data, **36**
  - Ascii Data, **27, 53, 56, 63, 137, 265, 275, 285, 293, 311, 329, 337, 341**
  - Expanding Plot, **53**
  - Printing, **53**
  - Right click, **53**
  - Bayes Model, **160**
  - Fid Data, **27, 265**
  - fid Data, **53, 53, 285, 293**
    - Auto Range, **59**
    - Autoscale, **56**
    - Clear Cursors, **56**
    - Clear Data, **57**
    - Copy, **59**
    - Cursor, **56**
    - Data Info, **57**
    - Expand, **56**
    - fn, **57**
    - Full, **56**
    - Get Peak, **56**
    - Phase Popup, **57**
    - Print, **59**
    - Properties, **59**
    - Referencing, **59**
    - Save As, **57, 59**
    - Set Preference, **57**
    - Units, **59**
    - Zoom, **59**
  - Fid Model, **27**
  - fid Model, **68, 186**
    - Build BA Model, **70, 159**
    - Data, **71**

- Horizontal, 71
- Model, 71
- Overlay, 71
- Report, 71
- Residual, 71
- Stacked, 71
- Trace, 71
- Vertical, 71
- File, 28, 80
- Fortran/C Models, 93, 330
- Image, 27, 59, 415
  - Autoset Grayscale, 61
  - Copy Selected, 62
  - Delete All, 61
  - Delete Selected, 61
  - Display Full, 61
  - Element Selection, 60
  - Export, 62
  - Get Statistics, 64, 65
  - Get Threshold Statistics, 65
  - Grayscale, 63
  - Image Selection, 60
  - List, 59
  - Load Selected Pixels, 61
  - Max, 64
  - Mean, 64
  - Min, 64
  - Right Click, 61
  - RMS, 64
  - Save Displayed, 62
  - Save Statistics, 65
  - Sdev, 64
  - Set Image Area, 62
  - Show Histogram, 61
  - Show Info, 62
  - Slice, 62
  - Slice Selection, 60
  - Statistics, 60
  - Value, 64
  - View Selected Pixels, 61
  - Viewer Settings, 62
  - Viewing, 62
  - X Pos, 64
  - Y Pos, 64
- Plot Results, 28, 71
- Prior, 27, 65
  - Prior Probabilities, 138, 312
  - Text, 141, 271, 281, 290, 309, 322, 335, 353
  - Text Results, 26, 28, 52, 74
    - Bayes Analyze, 176
- Widgets
  - Analyze Image Pixel Package
    - Build, 411
    - Find Outliers, 411
    - Get Statistics, 413
    - System, 411
    - User, 411
  - Analyze Image Pixel Unique Package
    - Build, 423
    - Find Outliers, 423
    - Get Statistics, 425
    - System, 423
    - User, 423
  - Ascii Data Viewer
    - Delete, 53
    - Left-mouse, 53
    - Right-mouse, 53
  - Bayes Analyze Package
    - By, 158, 176
    - First Point, 163
    - From, 158, 176
    - Imag Offset, 163
    - Mark, 159
    - Max New Res, 157
    - New, 159
    - Noise, 158
    - Phase, 157
    - Primary, 158
    - Real Offset, 163
    - Remove, 159
    - Remove All, 159
    - Reset, 159, 193
    - Restore, 159
    - Save, 159
    - Secondary, 159
    - Shim Order, 157, 163
    - Signal, 158
    - To, 158, 176
  - Bayes Find Resonances Package
    - Build FID Model, 240, 241, 246
    - Constant, 239, 242

- First Trace, 239
- Last Trace, 239
- Model Fid Number, 241
- Phase Model, 239, 242
- Bayes Metabolite Package
  - Fid Model, 221
  - Fid Model Viewer, 221
  - Load System Metabolite File, 219
  - Load System Resonance File, 221
  - Load User Metabolite File, 219
  - Load User Resonance File, 221
  - Shift Left, 221, 222
  - Shift Right, 221, 222
- Bayes Test Data Package
  - # Images, 427
  - # Slices, 427
  - Abscissa, 427
  - ArrayDim, 427
  - Build, 427
  - Get Job, 428
  - Max Value, 427
  - Noise SD, 427
  - Pe, 427
  - Ro, 427
  - Run, 428
  - Set (server), 428
  - Status, 428
  - System, 427
  - User, 427
- Big Magnetization Transfer Package
  - Find Outliers, 259
- Big Peak/Little Peak Package
  - Metabolite, 207
  - Solvent, 207
- Diffusion Tensor Package
  - Abscissa Options, 248
  - Find Outliers, 247
  - Include Constant, 247, 248, 255
  - Tensor Number, 247, 248, 255
  - Use b Matrix, 255
  - Use b Vectors, 254, 255
  - Use g Vectors, 254
- Enter Ascii Model Package
  - Find Outliers, 329
  - System, 329
  - User, 329
- Enter Ascii Model Selection Package
  - Find Outliers, 341
  - System, 341
  - User, 341
- Errors In Variables Package
  - Given Errors In, 303
  - Order, 303
- Fid Data Viewer
  - Autoscale, 56
  - Clear Cursors, 56
  - Cursor A, 56
  - Cursor B, 56
  - Delta, 56
  - Display Type, 56
  - Expand, 56
  - Full, 56
  - Get Peak, 56
  - Left-mouse, 56
  - Options, 57, 59
  - Right-mouse, 56
  - Trace, 70
- Fortran/C Model Viewer
  - Abscissa Spinner, 93
  - Add Prior, 96
  - Allow/Disallow Editing, 97
  - Cancel and Exit, 96
  - Changing Models, 94
  - Code, 93, 94
  - Compile Results, 97
  - Compiling, 96
  - Create/Edit Model, 93
  - Data Columns Spinner, 93
  - Derived, 96
  - Edit/Create New Model, 93, 94
  - High, 97
  - Low, 97
  - Mean, 97
  - Model, 96
  - Model Vectors, 93
  - Name (parameter), 97
  - Order, 97
  - Parameter Type, 97
  - Parameters button, 93, 94, 96
  - Prior Type, 97
  - Priors, 96
  - Remove All (priors), 96

- Remove Prior, 96
- Remove Selected Model, 93
- Save and Load, 96
- Standard Deviation, 97
- Given Exponential Package
  - Constant, 137, 139
  - Find Outliers, 137
  - Order, 137, 139
- Given Polynomial Order Package
  - Set Order, 285
- Global
  - Bayes Find Outliers, 27
  - Cancel, 26, 51
  - Edit Servers, 26
  - Get Job, 26, 51, 137, 143, 151, 155, 209, 221, 241, 247, 259, 267, 277, 285, 293, 305, 311, 331, 339, 343, 357, 364, 375, 413, 417, 425, 428
  - Reset, 27
  - Restore Analysis, 22
  - Run, 26, 51, 137, 143, 151, 155, 207, 221, 241, 247, 248, 259, 267, 277, 285, 293, 305, 311, 329, 337, 343, 357, 364, 373, 413, 415, 425, 428
  - Save, 27
  - Set (server), 26, 52, 137, 143, 151, 155, 207, 221, 239, 247, 259, 265, 277, 285, 293, 305, 311, 329, 337, 343, 355, 364, 373, 413, 415, 425, 428
  - Status, 26, 52, 137, 143, 151, 155, 207, 221, 241, 247, 259, 267, 277, 285, 293, 305, 311, 329, 337, 343, 355, 364, 373, 413, 415, 425, 428
- Image Model Selection Package
  - System, 415
  - User, 415
- Image Viewer
  - Element Number, 62
  - Get Statistics, 64
  - Get Threshold Statistics, 65
  - Grayscale, 63
  - Save Statistics, 65
  - Slice Number, 62
  - Value, 64
  - X Pos, 64
  - Y Pos, 64
- Inversion Recovery Package
  - Find Outliers, 151
- Kernel Density Function Package
  - Kernel Type, 364
  - Output Size, 364
- Linear Phasing Package
  - cf, 403
  - Display, 403
  - Display Array Element, 403
  - fn, 403
  - fn1, 403
  - Image Type, 402
  - Load An Image, 402
  - np, 403
  - nv, 403
  - Process, 403
- Magnetization Transfer Kinetics Package
  - Load, 275, 281
  - Set, 275
  - Uncertainty, 275
- Magnetization Transfer Package
  - Find Outliers, 265
- MaxEnt Density Function Estimation Package
  - Histogram Size, 373
  - Order, 373
- Non-Linear Phasing Package
  - Process, 409
  - Write Ascii images, 409
  - Write imaginary images, 409
- Prior Viewer
  - High, 65
  - Low, 65
  - Mean, 65
  - Prior Type, 67
- Server
  - Edit, 52
  - Name, 26, 52, 52
  - Set (server), 48
  - Setup, 48, 52
- Test Ascii Model Package
  - Find Outliers, 339
  - System, 337
  - User, 337
- Text Results Viewer
  - Copy, 74

- Down arrow, [74](#)
- Enable Editing, [74](#)
- Print, [74](#)
- Save (a copy), [74](#)
- Save As, [74](#)
- Settings, [74](#)
- Up arrow, [74](#)
- Unknown Number of Exponentials Package
  - Constant, [143](#)
  - Find Outliers, [143](#)
  - Order, [143](#)
- Unknown Polynomial Order Package
  - Set Order, [293](#), [294](#)
  - Unknown Order, [293](#), [294](#)
- WorkDir
  - Creating, [22](#), [33](#), [46](#)
  - Deleting, [22](#), [33](#), [46](#)
  - List, [24](#), [46](#)
  - Loading, [46](#)
  - Name, [46](#)
  - Popup, [47](#)