Bayesian Analysis Users Guide
Release 4.00, Manual Version 1

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: larry@bayes.wustl.edu

October 21, 2016

Contents

List of Figures

List of Tables

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

$$d_i = S(t_i) + \sigma_i \tag{E.1}$$

$$S(t_i) = M_\infty + (M_0 - M_\infty) * \exp\{-\alpha t_i\} \tag{E.2}$$

where this model is written as an inversion recovery model: M_0 is the amplitude at time $t = 0$, M_∞ is the amplitude at time $t = \infty$ and α 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_∞ and α 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*Abscissa(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
Mz0       -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 coarse 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 then 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 lease 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 Derived(1) = 1/Params(1) as a derived parameter and the package will output both Params(1) and Derived(1).

Abscissa 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 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 decals 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 decals 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:

$$d_k(t_i) = \sum_{j=1}^{m} A_{jk} G_j(\Omega, t_i) + \sigma_{ki} \tag{E.3}$$

where there are multiple data sets, $k \in \{1, 2, \ldots, n\}$, 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 Ω 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:

$$d(t_i) = M_\infty + (M_0 - M_\infty) \exp\{-\alpha t_i\}. \tag{E.4}$$

This equation is not in the form given by Eq. E.3. However, it is simple to rearrange it into this form:

$$d(t_i) = M_0 \exp\{-\alpha t_i\} + M_\infty (1 - \exp\{-\alpha t_i\}) \tag{E.5}$$

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

$$G_1(\Omega, t_i) = \exp\{-\alpha t_i\} \tag{E.6}$$

and

$$G_2(\Omega, t_i) = 1 - \exp\{-\alpha t_i\}. \tag{E.7}$$

Formally, the model now reduces to

$$d(t_i) = A_1 G_1(\Omega, t_i) + A_2 G_2(\Omega, t_i) \tag{E.8}$$

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 an 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 its 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 $d_i = A_1 G_1(\Omega, t_i) + A_2 G_2(\Omega, t_i) + \cdots$. 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 very 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.

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, Marlyn 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, Marlyn 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 f¨r 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] Marlyn 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 13C 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}H and ^{13}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 13C 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}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.