A Highly Stable Computer Algorithm for Solving Discrete Linearly Constrained Maximum Entropy Problems

G. Larry Bretthorst

Department of Physics
Washington University
St. Louis, MO. 63130

ABSTRACT

Linearly constrained maximum entropy has proven useful in a wide variety of applications. Unfortunately this problem is very difficult to solve using standard Newton-Raphson iteration techniques. Here we present a novel transformation of the problem which permits Newton-Raphson to solve it with a high degree of stability. The author has used the routine to solve problems with over one hundred Lagrange multipliers without difficulty.

1. Introduction

The linearly constrained maximum entropy has proven useful in a wide variety of applications ranging from the assignment of prior probabilities[1] to estimating unknown functions[2] to summing series[3] and a wide variety of others.[4,5] Unfortunately this problem is very difficult to solve using standard Newton-Raphson iteration techniques.[2] Here we present a novel transformation of the problem that permits the Newton-Raphson to solve it with a high degree of stability.

Very briefly the problem is to find the maximum of

$$-\sum_{i=1}^{N} P_i \log P_i + \sum_{j=1}^{m} \beta_j \left[u_j - \sum_{i=1}^{N} A_{ij} P_i \right] \tag{1}$$

subject to variations in the probabilities P_i and the Lagrange multipliers β_j. The N by m matrix A_{ij} specifies the "rule" for calculation the constraint value u_j from the probabilities P_i (we have assumed the normalization constraint is a member of the A_{ij} matrix). Variations with respect to the Lagrange multipliers just returns the constraint equations

$$u_j = \sum_{i=1}^{N} A_{ij} P_i \tag{2}$$

and variations with respect to the P_i returns

$$P_i = e^{-\sum_{k=1}^{m} \beta_k A_{ik}} . \tag{3}$$

Combining equations (2) and (3) gives

$$u_j = \sum_{i=1}^{N} A_{ij} e^{-\sum_{k=1}^{m} \beta_k A_{ik}} . \tag{4}$$

Equation (4) is a set of m coupled nonlinear equations for the Lagrange multiplers β_j. This system of equations can be solved exactly only for a few particularly simple problems. The standard way to proceed is to Taylor expand (4) about some initial estimate of the multipliers and then assume the high order expansion terms are small. This allows one to solve for the corrections to the multiplers. The procedure is iterated until convergence is obtained. This procedure (Newton-Raphson) typically works well for a few constraints and becomes more and more unstable as the number of constraints increases.

II. The Algorithm

This instability in the solution algorithm can be removed by reformulation of the problem. The problem can be restated in terms of a different, but equivalent, set of constraints. If one has two maximum entropy problems the first constrained by u_j calculated from A_{ij} a second constrained by v_j and calculated from B_{ij}. Then if u_j, v_j, A_{ij}, and B_{ij} are related by

$$v_j = \sum_{k=1}^{m} e_{kj} u_k$$

$$B_{ij} = \sum_{k=1}^{m} e_{kj} A_{ik}$$

where the matrix e_{jk} is not singular, then the two problems have the same P_i. If one thinks about this for a second it is obviously true. If one has two sets of constraints which are linearly related the "information" content must be the same; maximum entropy must assign them the same prior probabilities. Of course the Lagrange multipliers will be very different for the two problems.

We can use this fact to transform problem (1) into a second problem of the same form that is "easier" for the Newton-Raphson algorithm. The transformation is straight forward: first, the matrix A_{ij} may be considered as a collection of m vectors each of N components; these are normalized

$$A'_{ij} = \frac{A_{ij}}{N_j} \tag{5}$$

where

$$N_j^2 = \sum_{i=1}^{N} A_{ij}^2 \ . \tag{6}$$

This has the effect of scaling the Lagrange multipliers so they are all the same order of magnitude. The new multipliers and constraints are given by

$$\beta'_j = N_j \beta_j \tag{7}$$

$$u'_j = \frac{u_j}{N_j} \ . \tag{8}$$

Next, the constraint "rule" A'_{ij}, constraint values u'_j, and the Lagrange multipliers β'_j are transformed by taking a linear combination of them given by

$$A''_{ik} = \sum_{j=1}^{m} e_{jk} A'_{ij} \tag{9}$$

$$u''_k = \sum_{j=1}^{m} e_{jk} u'_j \tag{10}$$

$$\beta''_k = \sum_{j=1}^{m} e_{jk} \beta'_j \tag{11}$$

where e_{jk} is the j'th component of the k'th eigenvector of the matrix

$$e_{kj} = \sum_{i=1}^{N} A'_{ik} A'_{ij} \ . \tag{12}$$

The maximum entropy problem to solve is

$$-\sum_{i=1}^{N} P_i \log P_i + \sum_{j=1}^{m} \beta''_j \left[u''_j - \sum_{i=1}^{N} A''_{ij} P_i \right]. \tag{13}$$

This is the same problem as (1) with one major difference; the new constraint matrix A''_{ij} has the properity

$$\sum_{i=1}^{N} A''_{ij} A''_{ik} = \lambda_j \delta_{jk}$$

where λ_j is the j'th eigenvalue of the matrix e_{jk} and δ_{jk} is the Kronecker delta function: the constraints are orthogonal. This problem can now be solved using the same Newton-Raphson procedure however, it is much more stable. The author has used this procedure to solve maximum entropy problems with more than one hundred constraints.

This procedure is very straight forward to implement: one simply sets up the appropriate maximum entropy problem, transforms to the new constraints, uses Newton-Raphson or some other (canned) procedure to solve for the Lagrange multipliers, and transforms back. A FORTRAN implementation of this algorithm is given in Appendix A. This implementation is written as a general subroutine capable of solving any discrete maximum entropy problem without modifications. The computer code will perform all the steps necessary to solve any discrete linearly constrained maximum entropy problem using the transformation (5-13). In Appendix B a small test program is given along with the results from the program.

Should one decide to implement this routine one will have to supply two additional pieces of code, an eigenvector routine and a matrix inversion routine. The modifications are detailed in the Appendix A.

APPENDIX A.
A Highly Stable Computer Algorithm to Solve Linearly Constrained Discrete Maximum Entropy Problems

This is a general implementation of the algorithm (5-13) described in this paper. The routine is double precision and will work unmodified for any number of constraints and any number of probabilities. Like all general routines this requires the user to pass a number of parameters including a work area. The call one must issue is

CALL MAXE(N,M,MU,CU,L,AIJ,PROB,W,ERRM,ERR).

A description of the parameter list follows:

Parm Name	Paper Name	Input/ Output	Description/ function
N	N	input	The number N of discrete probabilities $P \equiv \{P_1, \cdots, P_N\}$.
M	m	input	The number m of constraints $u \equiv \{u_1, \cdots, u_m\}$, including the normalization constraint.
MU	u_j	input	These are the m constraints u_j [Equation (2)].† This vector is defined as DOUBLE PRECISION MU(M).

CU		output	These are the values of the constraints calculated from the output P_j. This is essentially a work area used by the subroutine to determine when it has converged. This area is a double precision real vector of DIMENSION CU(M).
L	β_j	i/o	On input this field contains the initial estimates of the Lagrange multipliers β_j (zero works very well in this algorithm). On output these are the calculated values of the multipliers. NOTE: This field is a double precision floating point array defined as DOUBLE PRECISION L(M).
AIJ	A_{ij}	input	These are the "rules" by which the constraints (2) are calculated. This matrix is a double precision floating point array of DIMENSION AIJ(N,M). The calling program must dimension it as AIJ(N,IX) where N is the number of discrete probabilities N and IX is greater than or equal to the number of constraints M.† The ordering of the subscripts causes FORTRAN to keep each constraint "rule" in continuous storage locations. This way one can vary the number of constraints without changing the dimensions in the calling program. NOTE: the normalization constraint must be included in the A_{ij} matrix.
PROB	P_j	output	These are the calculated discrete probabilities (3). This field must be of DIMENSION PROB(N).
W		work	This is a work area used by the subroutine; it must be of DIMENSION W(3*M*M + 3*M).
EMAX		input	This is the maximum variance one will accept in the calculated moments. This field should never be smaller than 10^{-10} and a more realistic value is 10^{-5}.
ERR		output	This is the calculated variance in the moments and is given by $\underset{I=1 \text{ to } m}{\text{MAX}}\left[\frac{CU(I) - MU(I)}{MU(I)}\right]$ if $MU(I)\neq 0$. If $MU(I)=0$ the denominator is set equal to 1.

† The computer code must translate the matrix AIJ and the vector MU into the new internal values. Before it returns it translates these values back to their original form. However, there is always round off errors associated with this process. Therefore, if one intends to call this routine many times using AIJ or MU repeatedly one will need to refresh these matrixes every hundred or so calls.

```
      SUBROUTINE MAXE(N,M,MU,CU,L,AIJ,PROB,W,EMAX,ERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      DOUBLE PRECISION L(M),MU(M),CU(M),PROB(N),AIJ(N,M),W(3*M*(M+1))
C
      IW=1
      IM=IW + M*M
      IT=IM + M*M
      IE=IT + M*M
      IN=IE + M
      I2=IN + M
C
      CALL TRANS(N,M,MU,AIJ,L,PROB,W(IM),W(IT),W(IN),W(IE),W(IW),W(I2))
C
      RELAX=1D-02
      CALL STEP
     C  (N,M,MU,CU,L,AIJ,PROB,W(IW),W(IM),W(IE),W(I2),EMAX,ERR,RELAX)
C
      CALL ITRANS(N,M,AIJ,L,MU,CU,W(IT),W(IN),W(IW))
C
      RETURN
      END

      SUBROUTINE TRANS
     C (N,M,MU,AIJ,L,PROB,METRIC,TRANSM,NORMS,EIGV,WORK1,WORK2)
      IMPLICIT REAL*8 (A-H,O-Z)
      DOUBLE PRECISION MU(M),AIJ(N,M),NORMS(M),L(M),PROB(N),WORK2(M)
      DOUBLE PRECISION METRIC(M,M),EIGV(M),TRANSM(M,M),WORK1(M)
C
C     TRANSFORM TO THE ORTHOGONAL CONSTRAINTS
C
C     NORMALIZE THE VECTORS AIJ, MU AND L
      DO 0200 I=1,M
      TOTAL=0D0
      DO 0100 J=1,N
 0100 TOTAL=TOTAL + AIJ(J,I)**2
      TOTAL=DSQRT(TOTAL)
      MU(I)=MU(I) / TOTAL
      NORMS(I)=TOTAL
      L(I)=L(I)*TOTAL
      DO 0150 J=1,N
 0150 AIJ(J,I)=AIJ(J,I) / TOTAL
 0200 CONTINUE
C
C     COMPUTE METRIC
      DO 2000 I=1,M
      DO 2000 J=1,M
      TOTAL=0D0
      DO 1000 K=1,N
 1000 TOTAL=TOTAL + AIJ(K,I)*AIJ(K,J)
      METRIC(I,J)=TOTAL
 2000 CONTINUE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     CALL EIGERS(M,M,METRIC,EIGV,1,TRANSM,WORK1,WORK2,IERR)       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SET UP THE ORTHOGONAL VECTORS
      DO 3000 K=1,N
      DO 3100 J=1,M
 3100 WORK1(J)=AIJ(K,J)
      DO 3000 I=1,M
      TOTAL=0D0
      DO 3300 J=1,M
 3300 TOTAL=TOTAL + TRANSM(J,I)*WORK1(J)
 3000 AIJ(K,I)=TOTAL
C
C     TRANSFORM THE DATA VALUES
      DO 5500 I=1,M
      TOTAL=0D0
      DO 5000 J=1,M
 5000 TOTAL=TOTAL + MU(J)*TRANSM(J,I)
 5500 WORK1(I)=TOTAL
      DO 5600 I=1,M
 5600 MU(I)=WORK1(I)
C
C     TRANSFORM THE L VALUES
      DO 6500 I=1,M
      TOTAL=0D0
      DO 6000 J=1,M
```

```
 6000   TOTAL=TOTAL + L(J)*TRANSM(J,I)
 6500   WORK1(I)=TOTAL
        DO 6600 I=1,M
 6600   L(I)=WORK1(I)
        RETURN
        END

        SUBROUTINE STEP
     C   (N,M,MU,CU,L,AIJ,PROB,WK1,WK2,WK3,WK4,EMAX,ERR,RELAX)
        IMPLICIT REAL*8 (A-H,O-Z)
        DOUBLE PRECISION L(M),MU(M),CU(M),AIJ(N,M),PROB(N)
        DOUBLE PRECISION WK1(M,M),WK2(M,M),WK3(M),WK4(M)
C
C       CONTROL THE ITERATIONS AND CHECK FOR CONVERGENCE
C
 9999   ERR=0D0
C
        CALL CALP(N,M,L,AIJ,PROB)
C
        DO 2000 I=1,M
        TOTAL=0D0
        DO 1000 J=1,N
 1000   TOTAL=TOTAL + PROB(J)*AIJ(J,I)
        CU(I)=TOTAL
        RATIO=DABS( (MU(I) - CU(I)) )
        IF(DABS(MU(I)).GT.1D-50)RATIO=DABS(RATIO / MU(I))
        IF(RATIO.GT.ERR)ERR=RATIO
 2000   CONTINUE
C
        IF(ERR.LT.EMAX)RETURN
C
        CALL NEWT(N,M,MU,CU,L,AIJ,PROB,WK1,WK2,WK3,WK4,RELAX)
        GO TO 9999
        END

        SUBROUTINE NEWT(N,M,MU,CU,L,AIJ,PROB,H,WORK,U,D1,RELAX)
        IMPLICIT REAL*8 (A-H,O-Z)
        DOUBLE PRECISION L(M),MU(M),CU(M),AIJ(N,M),PROB(N),H(M,M),U(M)
        DOUBLE PRECISION WORK(M,M),D1(M)
C
C       PERFORM ONE NEWTON-RAPHSON STEP
C
        DO 2000 I=1,M
        DO 2000 J=1,I
        TOTAL=0D0
        DO 1000 K=1,N
 1000   TOTAL=TOTAL + AIJ(K,J)*AIJ(K,I)*PROB(K)
        H(I,J)=TOTAL
 2000   H(J,I)=TOTAL
C
        DO 3000 I=1,M
        U(I)=MU(I)-CU(I)
 3000   CONTINUE
C
        CALL SOLVE(H,U,M,D1,WORK(1,1),WORK(1,M))
C
        RELAX=DSQRT(RELAX)
        DO 4000 I=1,M
        L(I)=L(I)-U(I)*RELAX
 4000   CONTINUE
        RETURN
        END

        SUBROUTINE SOLVE(H,U,M,DELTA,AK,AJ)
        IMPLICIT REAL*8(A-H,O-Z)
        DIMENSION DELTA(M),H(M,M),U(M),AK(M),AJ(M)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       CALL MATINV(H,M,DET,AK,AJ)                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        DO 1000 I=1,M
        WORK=0D0
        DO 2000 J=1,M
 2000   WORK=WORK + H(I,J) * U(J)
 1000   DELTA(I)=WORK
C
        DO 3000 I=1,M
 3000   U(I)=DELTA(I)
        RETURN
        END
```

```
      SUBROUTINE CALP(N,M,L,AIJ,PROB)
      IMPLICIT REAL*8 (A-H,O-Z)
      DOUBLE PRECISION L(M),AIJ(N,M),PROB(N)
C
C     CALCULATE THE DISCRETE P(I) FROM THE CURRENT L(J)
C
      DO 2000 I=1,N
      Q=1D0
      DO 1000 J=1,M
 1000 Q=Q+L(J)*AIJ(I,J)
      PROB(I)= DEXP(-Q)
 2000 CONTINUE
C
      RETURN
      END

      SUBROUTINE ITRANS(N,M,AIJ,L,MU,CU,TRANSM,NORMS,WORK)
      IMPLICIT REAL*8 (A-H,O-Z)
      DOUBLE PRECISION AIJ(N,M),L(M),MU(M),CU(M),TRANSM(M,M)
      DOUBLE PRECISION NORMS(M),WORK(M)
C
C     PERFORM THE INVERSE TRANSFORMATION
C
      DO 1000 I=1,M
      TOTAL=0D0
      DO 1500 J=1,M
 1500 TOTAL=TOTAL + CU(J)*TRANSM(I,J)
 1000 WORK(I)=TOTAL
      DO 1300 I=1,M
 1300 CU(I)=WORK(I)
C
      DO 2000 I=1,M
      TOTAL=0D0
      DO 2500 J=1,M
 2500 TOTAL=TOTAL + MU(J)*TRANSM(I,J)
 2000 WORK(I)=TOTAL
      DO 2300 I=1,M
 2300 MU(I)=WORK(I)
C
      DO 3000 I=1,M
      TOTAL=0D0
      DO 3500 J=1,M
 3500 TOTAL=TOTAL + L(J)*TRANSM(I,J)
 3000 WORK(I)=TOTAL
      DO 3300 I=1,M
 3300 L(I)=WORK(I)
C
      DO 4000 K=1,N
      DO 4100 J=1,M
 4100 WORK(J)=AIJ(K,J)
      DO 4000 I=1,M
      TOTAL=0D0
      DO 4300 J=1,M
 4300 TOTAL=TOTAL + TRANSM(I,J)*WORK(J)
 4000 AIJ(K,I)=TOTAL
C
      DO 5000 I=1,M
      MU(I) =MU(I)   * NORMS(I)
      L(I)  =L(I)    / NORMS(I)
      CU(I) =CU(I)   * NORMS(I)
      DO 5000 J=1,N
 5000 AIJ(J,I)=AIJ(J,I) * NORMS(I)
C
      RETURN
      END
```

There are two simple modifications one must make to this code: an eigenvector routine and a matrix inversion routine must be supplied. The first modification is in subroutine TRANS. The CALL to replace is boxed (commented) and looks like

CALL EIGERS(M,M,METRIC,EIGV,1,TRANSM,WORK1,WORK2,IERR)

the replacement routine must calculate the eigenvalues and eigenvectors of the matrix METRIC. This real symmetric matrix is of DIMENSION METRIC(M,M). The replacement routine can use METRIC as a work area if need be. The eigenvalues must be placed in the vector EIGV and the eigenvectors must be in TRANSM. The square matrix TRANSM is of

DIMENSION TRANSM(M,M). The eigenvectors must be stored as follows: TRANSM(J,K) is the J'th component of the K'th eigenvector. The eigenvalues and eigenvectors must be in one-to-one correspondence. The two work areas WORK1 and WORK2 are both vectors of DIMENSION WORK1(M),WORK2(M). The replacement routine can use both of these as scratch areas.

The second modification is in subroutine SOLVE, a matrix inversion routine must be supplied. The CALL to replace is boxed (comment) and looks like

CALL MATINV(H,M,DET,AI,AJ).

The inverse of the real symmetric matrix H of DIMENSION H(M,M) must be calculated and must replace H. The field DET is not needed by the code it was required by MATINV. The two vectors AI and AJ of DIMENSION AI(M),AJ(M) are work areas used by MATINV; feel free to use these if need be.

APPENDIX B
A Simple Test Problem

This second routine is meant to demonstrate how to call MAXE and to supply a test to verify the code is working correctly. This test will calculate the maximum entropy solution for a linearly constrained problem using five moments as constraints. The routine works as follows: the function $1/x^2$ is generated on the integers ($1 \le x \le 501$); this function is then normalized; the zero through fourth moments are calculated (remember the normalization constraint must be included in A_{ij}); the subroutine MAXE is called and, after returning, prints the multipliers, the true moments, the calculated moments, the error in each of these moments, and last the routine prints the discrete probabilities.

This test illustrates one of the points mentioned earlier: this program was set up to do up to 10 constraints; even though only 5 are actually done in this test. If one wishes to try it on more moments simply set the value of M to the number of constraints to pass (of course M must be less than or equal to 10).

One last note the matrix inversion routine will often have elements with large dynamic ranges; underflows sometimes occur. It is often worth while to call the ERRSET routine on IBM machines to suppress these messages.

This is the output from the test program:

```
L -4.92990576d-01 MU 1.00000000d+00 CU 1.00000000d+00 ERR     2.65793d-10
L  4.98452045d-01 MU 4.13576834d+00 CU 4.13576839d+00 ERR     1.31242d-08
L -5.50345996d-03 MU 3.04941133d+02 CU 3.04941144d+02 ERR     3.71516d-08
L  2.08880787d-05 MU 7.65402244d+04 CU 7.65402268d+04 ERR     3.16094d-08
L -2.36030514d-08 MU 2.55899483d+07 CU 2.55899489d+07 ERR     2.09709d-08

   1     3.67887d-01     2     2.27168d-01     3     1.41792d-01     4     8.94484d-02
   5     5.70242d-02     6     3.67331d-02     7     2.39063d-02     8     1.57172d-02
   9     1.04373d-02    10     7.00006d-03    11     4.74094d-03    12     3.24208d-03
  13     2.23835d-03    14     1.56000d-03    15     1.09740d-03    16     7.79103d-04
  17     5.58171d-04    18     4.03487d-04    19     2.94261d-04    20     2.16484d-04
  21     1.60642d-04    22     1.20222d-04    23     9.07306d-05    24     6.90423d-05
  25     5.29690d-05    26     4.09661d-05    27     3.19356d-05    28     2.50915d-05
  29     1.98669d-05    30     1.58504d-05    31     1.27411d-05    32     1.03178d-05
  33     8.41645d-06    34     6.91496d-06    35     5.72167d-06    36     4.76740d-06
  37     3.99964d-06    38     3.37827d-06    39     2.87249d-06    40     2.45848d-06
  41     2.11775d-06    42     1.83585d-06    43     1.60144d-06    44     1.40557d-06
  45     1.24114d-06    46     1.10246d-06    47     9.85024d-07    48     8.85162d-07
```

To conserve space we did not list all 501 of the P_i.

```
      IMPLICIT REAL*8 (A-H,O-Z)
      DOUBLE PRECISION L(10),MU(10),CU(10),PROB(501),AIJ(501,10)
      DOUBLE PRECISION WORK(3*10*11)
C
C     M CONTROLS THE NUMBER OF CONSTRAINTS
      M=05
      N=501
C
C     GENERATE THE FUNCTION 1/X**2 ON THE INTEGERS
      TOTAL=0D0
      DO 1000 I=1,N
      PROB(I)=1D0/(I*I)
 1000 TOTAL=TOTAL + PROB(I)
C
C     NORMALIZE THIS FUNCTION
      DO 2000 I=1,N
 2000 PROB(I)=PROB(I)/TOTAL
C
C     SET UP THE AIJ MATRIX
      DO 3000 I=1,M
      DO 3000 J=1,N
      AA=J
 3000 AIJ(J,I)=AA ** (I-1D0)
C
C     CALCULATE THE MOMENTS OF THE TRUE FUNCTION
      DO 4000 I=1,M
      TOTAL=0D0
      DO 3500 J=1,N
 3500 TOTAL=TOTAL + PROB(J)*AIJ(J,I)
 4000 MU(I)=TOTAL
C
C     SET UP THE INITIAL ESTIMATE OF THE MULTIPLIERS
      DO 5000 I=1,M
 5000 L(I)=0D0
C
      CALL MAXE(N,M,MU,CU,L,AIJ,PROB,WORK,1D-7,ERR)
C
      DO 6100 I=1,M
      ERRS=DABS((MU(I) - CU(I))/MU(I))
 6100 WRITE(6,6000)L(I),MU(I),CU(I),ERRS
 6000 FORMAT(' L ',1PD15.8,' MU ',D15.8,' CU ',D15.8,' ERR',D15.8)
      WRITE(6,6200)(I,PROB(I),I=1,501)
 6200 FORMAT(I5,1PD15.5,I5,D15.5,I5,D15.5,I5,D15.5)
C
      STOP
      END
```

References

1. G. Larry Bretthrost, *Bayesian Spectrum Analysis and Parameter Estimation*, Ph.D. thesis, University Microfilms Inc., Washington University, St. Louis, MO, Aug. 1987.
2. Laurence R. Mead and N. Papanicolaou, "Maximum Entropy in the Problem of Moments," *Journal of Mathematical Physics*, vol. 25(8), Aug. 1984.
3. Carl M. Bender, Laurence R. Mead, and N. Papanicolaou, "Maximum Entropy Summation of Divergent Perturbation Series," *Journal of Mathematical Physics*, (in preparation).
4. P. A. Fedders and A. E. Carlsson, "Information Theoretic Approach to High Temperature Spin Dynamics," *Physics Reviews B*, vol. 32(1), pp. 229-232, July 1985.
5. E. T. Jaynes, "Where Do We Stand On Maximum Entropy?," in *Papers on Probability, Statistics and Statistical Physics*, ed. R. D. Rosenkrantz, p. 210, D. Reidel, Boston, 1983.