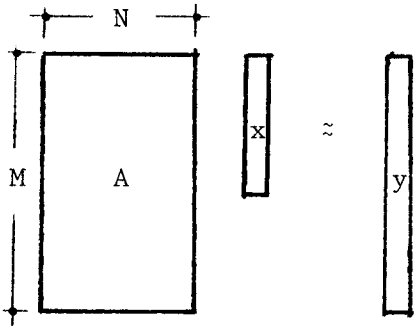
Mixed L_1 and L_2 Norm Problems

by Luis Canales

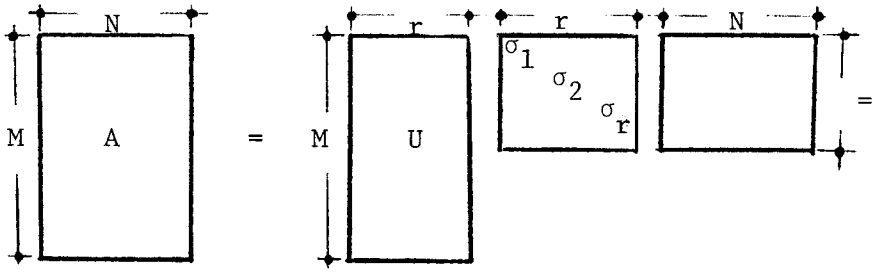
The use of the L_1-norm in overdetermined systems has been suggested by Claerbout and Muir (1973) for the treatment of erratic data. Claerbout (SEP-7) has also suggested its use for underdetermined systems. In the latter case, the L_1-norm gives a solution with only r non-zero values, where r is the rank of the system. If our model is homogeneous in this sense, the resolution obtained by means of the L_1-norm will be much better than that given by the use of L_2-norm. There are however some situations in which you might want to use a mixed norm solution, that is you would like to minimize the norm of the error and the norm of the solution using different norms on each case. You could for instance be solving a non-linear model fitting problem with erratic data so that you want to minimize the L_1-norm of the error, but to assure stability you minimize the L_2 norm of the change from the previous approximation.

To define the mixed problem, let us consider the following linear system.



(1)

The singular value decomposition of the matrix A gives us the matrices U , V and Σ such that:



(2)

where r is the rank of the matrix A , and

$$U^T U = V V^T = I_r$$

and $\sigma_1)\sigma_2)\ldots\sigma_r$ are the singular values (i.e., σ_i^2 are the eigenvalues of $(A^T A)$.

The mixed-norm problem can now be defined as follows:

Step 1. Solve $Uz \simeq y$ (overdetermined) (3)

using either the L_1 or L_2 norm, depending on the data's quality.

Step 2. Solve $Vx = \Sigma^{-1} z$ (underdetermined) (4)

using L_1 or L_2 norms, depending on the model's homogeneity.

Solving Steps 1 and 2 for L_2-norm is easily done by using Golub's method and its dual. Step 1 for L_1 is solved by using the algorithms of Claerbout and Muir (1973) or Barrowdale (1974).

The less standard problem of solving

$$Vx = \omega$$

such that $\|x\|_{L_1}$ is minimized can be solved by means of the following linear programming problem.

$$\text{Minimize} \quad \{ \sum_j x_{1j} + \sum_j x_{2j} \}$$

subject to

$$\begin{bmatrix} V & -V \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \omega \end{bmatrix} \qquad (5)$$

and

$$x_{1j} \geq 0 \qquad x_{2j} \geq 0$$

The singular values in Σ are usually ordered such that σ_1 is the larger, so that you can compare σ_1/σ_r and if this ratio is very small, we neglect it and then consider a more stable linear system. In terms of the L_2 norm it means that we are sacrificing resolution for the sake of small variances in the solution. In terms of the L_1-norm. fixing the rank is deciding how many equations you want to satisfy and how many non-zero components you want in the solution.

There are times in which it makes sense to maximize the objective function in problem (5), instead of minimizing it. As, for instance, instead of the maximum entropy spectrum you could ask for the one that maximizes $\int |s(\omega)| d\omega$. This will give you a spectrum which is zero at all but a few frequencies, and is consistent with the auto-

correlation values. This method will hopefully give you a better resolution in estimation of spectral lines than maximum entropy will.

Note that in this problem you can use the standard linear program instead of (5), because the spectrum has to be positive.

On the following pages is a program (SVD) for doing singular value decomposition. It uses Golub's method, which avoids forming the normal equations.

References:

Claerbout, J. F. And F. Muir (1973), Robust modeling with erratic data, Geophysics, Vol. 38, No. 5, pp. 826-844.

Barrowdale, I. and F. D. K. Roberts (1974), Algorithm 478, Solution of an overdetermined system of equations in the L_1-norm, Comm. of the A.C.M., Vol. 17, No. 6, pp. 319-320.

```
C     ********************  START OF SVD  ****************************
C
      SUBROUTINE  SVD  (A, S, V, MMAX, NMAX, M, N, P, WITHU, WITHV)
C
      INTEGER     MMAX, NMAX, M, N, P
      REAL        A(MMAX,N), S(NMAX), V(NMAX,NMAX)
      LOGICAL     WITHU, WITHV
C
C     ------------------------------------------------------------------
C
C     THIS IS A TRANSLATION OF A CDC 6600 FORTRAN PROGRAM TO IBM 360
C     FORTRAN IV.  THIS SUBROUTINE USES SHORT PRECISION ARITHMETIC.
C     A LONG PRECISION VERSION IS AVAILABLE UNDER THE NAME 'DSVD'.
C
C     THIS SUBROUTINE REPLACES EARLIER SUBROUTINES WITH THE SAME NAME,
C     WHICH WERE TRANSLATIONS OF A COMPLEX ARITHMETIC PROGRAM, PUBLISHED
C     AS ALGORITHM 358.  THIS CURRENT PROGRAM IS FASTER, MORE ACCURATE
C     AND LESS OBSCURE IN DESCRIBING ITS CAPABILITIES.
C
C     ORIGINAL PROGRAMMER:  R. C. SINGLETON
C     360 VERSION BY:       J. G. LEWIS
C     LAST REVISION OF THIS SUBROUTINE:  4 DECEMBER 1973
C
C     ------------------------------------------------------------------
C
C     ADDITIONAL SUBROUTINE NEEDED:  ROTATE
C
C     ------------------------------------------------------------------
C
C     THIS SUBROUTINE COMPUTES THE SINGULAR VALUE DECOMPOSITION
C     OF A REAL M*N MATRIX A, I.E. IT COMPUTES MATRICES U, S, AND V
C     SUCH THAT
C
C                  A = U * S * VT ,
C     WHERE
C               U IS AN M*N MATRIX AND UT*U = I, (UT=TRANSPOSE
C                                                    OF U),
C               V IS AN N*N MATRIX AND VT*V = I, (VT=TRANSPOSE
C                                                    OF V),
C        AND    S IS AN N*N DIAGONAL MATRIX.
C
C     DESCRIPTION OF PARAMETERS:
C
C     A = REAL ARRAY. A CONTAINS THE MATRIX TO BE DECOMPOSED.
C         THE ORIGINAL DATA ARE LOST.  IF WITHV=.TRUE., THEN
C         THE MATRIX U IS COMPUTED AND STORED IN THE ARRAY A.
C
C     MMAX = INTEGER VARIABLE.  THE NUMBER OF ROWS IN THE
C            ARRAY A.
C
C     NMAX = INTEGER VARIABLE.  THE NUMBER OF ROWS IN THE
C            ARRAY V.
C
C     M,N = INTEGER VARIABLES.  THE NUMBER OF ROWS AND COLUMNS
?
```

```
C            IN THE MATRIX STORED IN A.  (N<=M<=100.  IF IT IS
C            NECESSARY TO SOLVE A LARGER PROBLEM, THEN THE
C            AMOUNT OF STORAGE ALLOCATED TO THE ARRAY T MUST
C            BE INCREASED ACCORDINGLY.)  IF M<N, THEN EITHER
C            TRANSPOSE THE MATRIX A OR ADD ROWS OF ZEROS TO
C            INCREASE M TO N.
C
C      P = INTEGER VARIABLE.  IF P~0, THEN COLUMNS N+1, . . . ,
C          N+P OF A ARE ASSUMED TO CONTAIN THE COLUMNS OF AN M*P
C          MATRIX B.  THIS MATRIX IS MULTIPLIED BY UT, AND UPON
C          EXIT, A CONTAINS IN THESE SAME COLUMNS THE N*P MATRIX
C          UT*B.  (P~=0)
C
C      WITHU, WITHV = LOGICAL VARIABLES.  IF WITHU=.TRUE., THEN
C          THE MATRIX U IS COMPUTED AND STORED IN THE ARRAY A.
C          IF WITHV=.TRUE., THEN THE MATRIX V IS COMPUTED AND
C          STORED IN THE ARRAY V.
C
C      S = REAL ARRAY.  S(1), . . . , S(N) CONTAIN THE DIAGONAL
C          ELEMENTS OF THE MATRIX S ORDERED SO THAN S(I)>=S(I+1),
C          I=1, . . . , N-1.
C
C      V = REAL ARRAY.  V CONTAINS THE MATRIX V.  IF WITHU
C          AND WITHV ARE NOT BOTH =.TRUE., THEN THE ACTUAL
C          PARAMETER CORRESPONDING TO A AND V MAY BE THE SAME.
C
C      THIS SUBROUTINE IS A REAL VERSION OF A FORTRAN SUBROUTINE
C      BY BUSINGER AND GOLUB, ALGORITHM 358:  SINGULAR VALUE
C      DECOMPOSITION OF A COMPLEX MATRIX, COMM. ACM, V. 12,
C      NO. 10, PP. 564-565 (OCT. 1969).
C      WITH REVISIONS BY RC SINGLETON, MAY 1972.
C      ------------------------------------------------------------
C
       REAL        R, M, CS, SN, TOL, F, X, EPS, G, T, Y
       REAL        ETA, H, Q, Z
       INTEGER     I, J, K, L, L1, M1, NP
       DIMENSION   T(100)
C
       DATA ETA /Z3B100000/
       DATA TOL /Z06100000/
C
C      ETA (16**-6) AND TOL (16**-59) ARE MACHINE DEPENDENT CONSTANTS
C      FOR IBM 360/370 COMPUTERS (SHORT FORM ARITHMETIC).
C      ETA IS THE MACHINE EPSILON (RELATIVE ACCURACY);
C      TOL IS THE SMALLEST REPRESENTABLE REAL DIVIDED BY ETA.
C
       NP = N + P
       M1 = N + 1
C
C      HOUSEHOLDER REDUCTION TO BIDIAGONAL FORM
       G = 0.0
       EPS = 0.0
       L = 1
    10 T(L) = G
       K = L
?
```

```
      L = L + 1
C
C     ELIMINATION OF A(I,K), I=K+1, . . . . , M
      S(K) = 0.0
      Z = 0.0
      DO 20 I = K,M
   20    Z = Z + A(I,K)**2
      IF (Z.LT.TOL) GOTO 50
      G = SQRT(Z)
      F = A(K,K)
      IF (F.GE.0.0) G = - G
      S(K) = G
      H = G * (F - G)
      A(K,K) = F - G
      IF (K.EQ.NP) GOTO 50
      DO 40 J = L,NP
         F = 0
         DO 30 I = K,M
   30       F = F + A(I,K)*A(I,J)
         F = F/H
         DO 40 I = K,M
   40       A(I,J) = A(I,J) + F*A(I,K)
C
C     ELIMINATION OF A(K,J), J=K+2, . . . . , N
   50 EPS = AMAX1(EPS,ABS(S(K)) + ABS(T(K)))
      IF (K.EQ.N) GOTO 100
      G = 0.0
      Z = 0.0
      DO 60 J = L,N
   60    Z = Z + A(K,J)**2
      IF (Z.LT.TOL) GOTO 10
      G = SQRT(Z)
      F = A(K,L)
      IF (F.GE.0.0) G = - G
      H = G * (F - G)
      A(K,L) = F - G
      DO 70 J = L,N
   70    T(J) = A(K,J)/H
      DO 90 I = L,M
         F = 0
         DO 80 J = L,N
   80       F = F + A(K,J)*A(I,J)
         DO 90 J = L,N
   90       A(I,J) = A(I,J) + F*T(J)
C
      GOTO 10
C
C     TOLERANCE FOR NEGLIGIBLE ELEMENTS
  100 EPS = EPS*ETA
C
C     ACCUMULATION OF TRANSFORMATIONS
      IF (.NOT.WITHV) GOTO 160
      K = N
      GOTO 140
  110    IF (T(L).EQ.0.0) GOTO 140
?
```

```
            H = A(K,L)*T(L)
            DO 130 J = L,N
               Q = 0
               DO 120 I = L,N
  120             Q = Q + A(K,I)*V(I,J)
               Q = Q/H
               DO 130 I = L,N
  130             V(I,J) = V(I,J) + Q*A(K,I)
  140       DO 150 J = 1,N
  150          V(K,J) = 0
            V(K,K) = 1.0
            L = K
            K = K - 1
            IF (K.NE.0) GOTO 110
C
  160    K = N
         IF (.NOT.WITHU) GOTO 230
         G = S(N)
         IF (G.NE.0.0) G = 1.0/G
         GO TO 210
  170       DO 180 J = L,N
  180          A(K,J) = 0
            G = S(K)
            IF (G.EQ.0.0) GOTO 210
            H = A(K,K)*G
            DO 200 J = L,N
               Q = 0
               DO 190 I = L,N
  190             Q = Q + A(I,K)*A(I,J)
               Q = Q/H
               DO 200 I = K,N
  200             A(I,J) = A(I,J) + Q*A(I,K)
            G = 1.0/G
  210       DO 220 J = K,N
  220          A(J,K) = A(J,K)*G
            A(K,K) = A(K,K) + 1.0
            L = K
            K = K - 1
            IF (K.NE.0) GOTO 170
C
C        OR DIAGONALIZATION
         K = N
C
C        TEST FOR SPLIT
  230       L = K
  240          IF (ABS(T(L)).LE.EPS) GOTO 290
               L = L - 1
               IF (ABS(S(L)).GT.EPS) GOTO 240
C
C        CANCELLATION
            CS = 0.0
            SN = 1.0
            L1 = L
            L = L + 1
            DO 280 I = L,K
?
```

```
                F = SN*T(I)
                T(I) = CS*T(I)
                IF (ABS(F).LE.EPS) GOTO 290
                H = S(I)
                W = SQRT(F*F + H*H)
                S(I) = W
                CS = H/W
                SN = - F/W
                IF (WITHU) CALL ROTATE(A(1,L1), A(1,I), CS, SN, N)
                IF (NP.EQ.N) GOTO 280
                DO 270 J = N1,NP
                   Q = A(L1,J)
                   R = A(I,J)
                   A(L1,J) = Q*CS + R*SN
  270              A(I,J) = R*CS - Q*SN
  280           CONTINUE
C
C       TEST FOR CONVERGENCE
  290      W = S(K)
           IF (L.EQ.K) GOTO 360
C
C       ORIGIN SHIFT
           X = S(L)
           Y = S(K-1)
           G = T(K-1)
           H = T(K)
           F = ((Y - W)*(Y + W) + (G - H)*(G + H))/(2.0*H*Y)
           G = SQRT(F*F + 1.0)
           IF (F.LT.0.0) G = - G
           F = ((X - W)*(X + W) + (Y/(F + G) - H)*H)/X
C
C       QR STEP
           CS = 1.0
           SN = 1.0
           L1 = L + 1
           DO 350 I = L1,K
              G = T(I)
              Y = S(I)
              H = SN*G
              G = CS*G
              W = SQRT(H*H + F*F)
              T(I-1) = W
              CS = F/W
              SN = H/W
              F = X*CS + G*SN
              G = G*CS - X*SN
              H = Y*SN
              Y = Y*CS
              IF (WITHV) CALL ROTATE(V(1,I-1), V(1,I), CS, SN, N)
              W = SQRT(H*H + F*F)
              S(I-1) = W
              CS = F/W
              SN = H/W
              F = CS*G + SN*Y
              X = CS*Y - SN*G
 ?
```

```
            IF (WITHU) CALL ROTATE(A(1,I-1), A(1,I), CS, SN, M)
            IF (N.EQ.NP) GOTO 350
            DO 340 J = N1,NP
               Q = A(I-1,J)
               R = A(I,J)
               A(I-1,J) = Q*CS + R*SN
  340          A(I,J) = R*CS - Q*SN
  350       CONTINUE
C
         T(L) = 0.0
         T(K) = F
         S(K) = X
         GOTO 230
C
C     CONVERGENCE
  360    IF (W.GE.0.0) GOTO 380
         S(K) = - W
         IF (.NOT.WITHV) GOTO 380
         DO 370 J = 1,N
  370       V(J,K) = - V(J,K)
  380    K = K - 1
         IF (K.NE.0) GO TO 230
C
C     SORT SINGULAR VALUES
      DO 450 K = 1,N
         G = -1.0
         DO 390 I = K,N
            IF (S(I).LT.G) GOTO 390
            G = S(I)
            J = I
  390       CONTINUE
         IF (J.EQ.K) GOTO 450
         S(J) = S(K)
         S(K) = G
         IF (.NOT.WITHV) GOTO 410
         DO 400 I = 1,N
            Q = V(I,J)
            V(I,J) = V(I,K)
  400       V(I,K) = Q
  410    IF (.NOT.WITHU) GOTO 430
         DO 420 I = 1,M
            Q = A(I,J)
            A(I,J) = A(I,K)
  420       A(I,K) = Q
  430    IF (N.EQ.NP) GOTO 450
         DO 440 I = N1,NP
            Q = A(J,I)
            A(J,I) = A(K,I)
  440       A(K,I) = Q
  450    CONTINUE
C
      RETURN
      END
?
```

```
      SUBROUTINE   ROTATE  (X, Y, CS, SN, N)
      INTEGER N
      REAL     X(N), Y(N), CS, SN
C
C
      REAL     XX
      INTEGER J
C
C
      DO 10 J = 1, N
         XX = X(J)
         X(J) = XX*CS + Y(J)*SN
   10    Y(J) = Y(J)*CS - XX*SN
      RETURN
      END
C     ************************  END OF SVD  ********************************
?
```