UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

FORTRAN '77 PROGRAMS FOR COMPUTING DATA FITTING FUNCTIONS BASED ON A PRINCIPLE OF MINIMUM INTEGRATED SQUARED CURVATURE

by

Raymond D. Watts

Open File Report 82-831

1982

This report consists of listings of FORTRAN '77 computer programs that implement a new algorithm for fitting a continuous, analytical function to a set of data points. The analytical function is defined in a one- or two-dimensional domain that contains the known data points. The fit satisfies the following criteria: (1) it passes through the known data points, and (2) the square of its curvature, integrated over the entire domain of the fit, is minimized.

The major programs, FOURGRID and FOURFIT, are intended to be self-documenting. These programs and the subroutines that they call have been extensively tested using data sets with up to 50 data points, with satisfactory results. The author is presently preparing a report on the mathematical algorithm that these programs implement.

```
      SUBROUTINE FOURGRID(X,Y,F,N,DX,DY,XO,YO,MODE,WORK,NWRK,
     .  LX,LY)
c
c
c     1.0  PURPOSE
c
c     Grids tabulated data in 2 dimensions using FOURFIT.
c
c
c
c     2.0  ARGUMENTS
c
c
c          1.   X - (input) real*4 array of x coordinates of  known
c               points, dimensioned (N).
c
c          2.   Y - (input) real*4 array of y coordinates of  known
c               points, dimensioned (N).
c
c          3.   F - (input) real*4 array of function values at  the
c               known points, dimensioned (N).
c
c          4.   N - (input) integer number of known points.
c
c          5.   DX - (input) grid interval in the x direction.
c
c          6.   DY - (input) grid interval in the y direction.
c
c          7.   XO - (input/output)  x  coordinate  of  lower  left
c               corner   of   grid   (input  or  output  parameter,
c               depending on value of MODE).
c
c          8.   YO - (input/output)  y  coordinate  of  lower  left
c               corner   of   grid   (input  or  output  parameter,
c               depending on value of MODE).
c
c          9.   MODE - (input)
c
c                 -  if MODE.eq.1, then XO and YO are considered  as
c                    input  specifications  of  the  minimum x and y
c                    values of the output grid.
c
c                 -  if MODE.eq.2, then XO and YO are constructed by
c                    FOURGRID  so  that  the grid is centered around
c                    the tabulated input data.  XO  and  YO,  which
c                    still represent the minimum x and y coordinates
c                    of  the  grid,  are  returned  to  the  calling
c                    program.
c
c                 -  if MODE has any other value, an  error  message
c                    is written and the program is STOPped.
c
c
c         10.   WORK - (output) work and  result  array  that  must
c               contain enough storage for all the work arrays used
```

```
c                 by FOURFIT.  Real*4 array dimensioned (NWRK).  The
c                 resulting grid is returned in WORK as a real*4
c                 array dimensioned (0:LX,0:LY).
c
c            11.  NWRK - (input) size of WORK, integer input
c                 parameter.  Minimum NWRK is max(N*(LX+1)*(LY+1)+
c                 2*N**2+5*N,8*LX*LY).  LX and LY are computed by
c                 FOURGRID as described below.
c
c            12.  LX - (output) x size of the output grid.
c
c            13.  LY - (output) y size of the output grid.  LX and LY
c                 are powers of 2 (because of the use of a radix-2
c                 FFT), which are adjusted to the minimum size that
c                 will span the input data range using either the
c                 input values of X0 and Y0 or the values computed by
c                 FOURGRID, depending on the value of MODE.
c
c
c
c
c        3.0  AUTHOR
c
c
c                 R. Watts
c                 U.S. Geological Survey
c                 P.O. Box 25046, Mail Stop 964
c                 Denver, CO 80225
c
c
c
c
c        4.0  TESTING SUMMARY
c
c        FOURGRID was written in December, 1981, and tested on a
c        Digital Equipment Corporation VAX 11/780 computer, using
c        DEC's VAX FORTRAN compiler.  The program is intended to
c        conform to FORTRAN '77 standards, and contains no known
c        non-FORTRAN '77 constructs.
c
c
c
c        5.0  CALLS SUBROUTINES:
c
c
c              -  FOURFIT
c
c              -  FFT2D
c
c              -  TOCOMPLEX
c
c              -  TOREAL
c
c
c
```

```
c
c       6.0  USAGE NOTES
c
c       If all values of X or Y are the same, and if FFT2D is  coded
c       so  that  it  will work with NX or NY = 1, then this program
c       can be used to fit one-dimensional data.
c
c
c
c
c       ------------------------------------------------------------
c
c       declarations:
        REAL*4 X(N),Y(N),F(N),WORK(NWRK)
        DATA PI/3.14159265/
c
c       determine grid parameters.
c       start by scanning for max and min values.
        XMIN=X(1)
        YMIN=Y(1)
        XMAX=XMIN
        YMAX=YMIN
        DO I=2,N
          XMAX=MAX(XMAX,X(I))
          YMAX=MAX(YMAX,Y(I))
          XMIN=MIN(XMIN,X(I))
          YMIN=MIN(YMIN,Y(I))
        END DO
c
c       if MODE is 1, then use X0 and Y0 as minima.
        IF (MODE.EQ.1) THEN
          IF (XMIN.LT.X0) THEN
            WRITE(*,'(1X,A)') 'Minimum value in input X array is'//
     .        ' smaller than specified X0, MODE=1, in FOURGRID.'
            STOP
          ELSE
            XMIN=X0
          END IF
          IF (YMIN.LT.Y0) THEN
            WRITE(*,'(1X,A)') 'Minimum value in input Y array is'//
     .        ' smaller than specified Y0, MODE=1, in FOURGRID.'
            STOP
          ELSE
            YMIN=Y0
          END IF
c
c       check for invalid value for MODE.
        ELSE
          IF (MODE.NE.2) THEN
            WRITE(*,'(1X,A)') 'Illegal value of MODE passed'//
     .        ' to FOURGRID.'
            STOP
          END IF
        END IF
c
```

```
c     determine grid size.
      XSPAN=XMAX-XMIN
      YSPAN=YMAX-YMIN
      LX=1
      DO WHILE(LX*DX.LT.XSPAN)
        LX=2*LX
      END DO
      LY=1
      DO WHILE(LY*DY.LT.YSPAN)
        LY=2*LY
      END DO
c
c     if MODE is 2, center the grid around the data.
      IF (MODE.EQ.2) THEN
        X0=(XMIN+XMAX)/2.-DX*(LX/2)
        Y0=(YMIN+YMAX)/2.-DY*(LY/2)
      END IF
c
c     check size of work array.
      MINSIZE=MAX(N*(LX+1)*(LY+1)+2*N**2+5*N,8*LX*LY)
      IF (NWRK.LT.MINSIZE) THEN
        WRITE(*,'(1X,A)') 'Error in FOURGRID:',
     .    'Size of work array is not sufficient.',
     .    'Grid parameters:'
        WRITE(*,'(4X,A,I6)') 'LX =',lx,'LY =',ly,
     .    'Minimum NWRK =',minsize
        STOP
      END IF
c
c     pointers for work array.
      I1=N*(LX+1)*(LY+1)+1
      I2=I1+N**2
      I3=I2+N**2
      I4=I3+N
      I5=I4+N
      I6=I5+N
      I7=I6+N
c
c     rescale X and Y to principal interval for FOURFIT.
      DO I=1,N
        WORK(I6+I-1)=(X(I)-X0)/DX
        WORK(I7+I-1)=(Y(I)-Y0)/DY
      END DO
c
c     get the Fourier cosine coefficients of the fit.
      CALL FOURFIT(WORK(I6),WORK(I7),F,N,
     .  FLOAT(LX),FLOAT(LY),WORK,LX,LY,WORK(I1),
     .  WORK(I2),WORK(I3),WORK(I4),WORK(I5))
c
c     arrange them for Fourier transformation using FFT.
      CALL TOCOMPLEX(WORK,WORK,LX,LY,2*LX,2*LY)
c
c     do the FFT.
      CALL FFT2D(WORK,2*LX,2*LY,-1.)
c
```

```
c     squeeze out the imaginary part.
      CALL TOREAL(WORK,WORK,LX,LY)
c
c     done
      END
```

```
      SUBROUTINE FOURFIT(X,Y,F,N,SPANX,SPANY,D,NX,NY,M,MWORK,
     .   V,W,LAMBDA)
c
c
c     1.0  PURPOSE
c
c     Computes the 2-D cosine transform of a fit  to  a  function,
c     using the principle of minimum integrated squared curvature.
c     At  the  present  time,  the   mathematical   algorithm   is
c     undocumented,  but  it  can be described briefly as follows:
c     (1) we describe a function (the fit) in terms of  a  Fourier
c     cosine series in D dimensions (the present programs work for
c     D=1 or 2);  (2) we use calculus of  variations  to  minimize
c     the  square  of  the  second  derivative  (or  the square of
c     del-squared) integrated over the domain of  the  fit,  while
c     the  fit  is  simultaneously constrained to pass through the
c     known data points.
c
c
c
c     2.0  ARGUMENTS
c
c
c          1.  X - (input) array of X coordinates of known points,
c              real*4  array dimensioned (N).  The calling program
c              should ensure that all X values fall in  the  range
c              0.  .le.  X(i) .le.  SPANX.
c
c          2.  Y - (input) array of Y coordinates of known points,
c              real*4  array dimensioned (N).  The calling program
c              should ensure that all Y values fall in  the  range
c              0.  .le.  Y(i) .le.  SPANY.
c
c          3.  F - (input) array of values at known points, real*4
c              array dimensioned (N).
c
c          4.  N - (input) number of known points.
c
c          5.  SPANX - (input) half the period of the function  in
c              the  x  dimension.  Since the function must be even
c              as well as periodic, it need be specified only over
c              half a period in each dimension.
c
c          6.  SPANY - (input) half the period of the function  in
c              the y dimension.
c
c          7.  D  -  (output)  work  and  result   array,   real*4
c              dimensioned   (0:NX,0:NY,N).   The  Fourier  cosine
c              coefficients are returned in the first panel  of  D
c              [i.e.  in D(0:NX,0:NY,1)].
c
c          8.  NX - (input) the number of x-dimension  frequencies
c              included  in  the series.  The highest frequency in
c              the X dimension is NX*pi/SPANX.
c
```

```
c            9.  NY - (input) the number of y-dimension  frequencies
c                included  in  the series.  The highest frequency in
c                the Y dimension is NY*pi/SPANY.
c
c           10.  M - work array, real*4 dimensioned (N,N).
c
c           11.  MWORK - work array, real*4 dimensioned (N,N).
c
c           12.  V - work array, real*4 dimensioned (N).
c
c           13.  W - work array, real*4 dimensioned (N).
c
c           14.  LAMBDA - (output) work  array,  real*4  dimensioned
c                (N).   On  exit,  this  array contains the Lagrange
c                multipliers from the minimization.
c
c
c
c
c       3.0  AUTHOR
c
c
c                R.  Watts
c                U.S.  Geological Survey
c                P.O.  Box 25046, Mail Stop 964
c                Denver, CO 80225
c
c
c
c
c       4.0  TESTING SUMMARY
c
c       FOURFIT was written in  December,  1981,  and  tested  on  a
c       Digital  Equipment  Corporation  VAX  11/780 computer, using
c       DEC's VAX FORTRAN compiler.  The  program  is  intended  to
c       conform  to  FORTRAN  '77  standards,  and contains no known
c       non-FORTRAN '77 constructs.
c
c
c
c       5.0  CALLS SUBROUTINES:
c
c
c             -  DOT
c
c             -  MATSOL
c
c             -  DCT
c
c             -  DICT
c
c             -  SUM
c
c
c
```

```
c
c       6.0  USAGE NOTES
c
c       On input, the user provides the known data points in
c       tabulated form.  The coordinates of the i'th known point are
c       (X(i),Y(i)), and the function value at that point is F(i),
c       with i ranging between 1 and N.
c
c       On output, the array D contains the Fourier cosine
c       coefficients.  They are stored as if D were dimensioned
c       (0:NX,0:NY), with the (i,j)th component representing
c       frequencies kx=i*DKX and ky=j*DKY, where DKX=pi/SPANX and
c       DKY=pi/SPANY.
c
c       To use the results of this program with a Fast Fourier
c       Transform for conversion into the space domain, perform the
c       following steps:
c
c            1.  Take the real values output by this program, and
c                make the numbers complex with zero imaginary part.
c
c            2.  Place the numbers into the FFT array or file, using
c                only positive frequencies in both dimensions.
c
c            3.  Apply the FFT, using a form that only applies the
c                necessary exponential factors (or cosine factors)
c                with no normalization by N, 2.*PI, etc.
c
c            4.  Use only the real part of the result.
c
c
c       The solution has two components: (1) a part that depends
c       only on the geometry of the known data points, and (2) a
c       part that is dependent on the data values at those points.
c       Both parts of the problem require solution of an NxN matrix,
c       where N is the number of data points.  If several
c       measurements are made at the same set of stations, then this
c       program might profitably be broken into the data-independent
c       and data-dependent parts, which could be called separately.
c
c       -------------------------------------------------------------
c
c
c       Declarations:
        REAL*4 F(N),M(N,N),MWORK(N,N),V(N),LAMBDA(N),
     .    D(0:NX,0:NY,N),W(N),X(N),Y(N)
        DATA PI/3.14159265/
c
c       The following weights are the number of image points
c       for each data point.
        DO I=1,N
          XNORM=X(I)/SPANX
          IF(ABS(XNORM-NINT(XNORM)).LT.1.E-5) THEN
            WTX=1.
          ELSE
```

```
            WTX=2.
          END IF
          YNORM=Y(I)/SPANY
          IF(ABS(YNORM-NINT(YNORM)).LT.1.E-5) THEN
            WTY=1.
          ELSE
            WTY=2.
          END IF
          W(I)=WTX*WTY
        END DO
c
c       Compute the Fourier cosine transform of each sample function.
c       Since each function is a delta function, the transform is
c       done by a simple DFT rather than by an FFT.
        DKX=PI/SPANX
        DKY=PI/SPANY
        DO I=1,N
          CALL DCT(X(I),Y(I),DKX,DKY,D(0,0,I),NX,NY,W(I))
        END DO
c
c       In the frequency domain, compute the function that when operated
c       on by the square of the Laplacian operator (del**4), yields
c       each sampling function (i.e. a delta function at the location
c       of the corresponding known point, plus delta functions
c       at any symmetry points).  This process must ignore
c       the zero-frequency component, since that is destroyed by the
c       Laplacian operator.
        DO I=1,N
          D(0,0,I)=(0.,0.)
          DO KX=0,NX
            WX=(KX*DKX)**2
            DO KY=MAX(0,1-KX),NY
              WY=(KY*DKY)**2
              WT=(WX+WY)**2
              D(KX,KY,I)=D(KX,KY,I)/WT
            END DO
          END DO
        END DO
c
c       Compute summed inverse transforms at the known sample points.
c       The results are not equally weighted due to the presence of 0,
c       1, or 3 image points resulting from the required even symmetry
c       in 2 dimensions.
c       Since the result is required only at one point, this computation
c       is most efficiently done by discrete cosine transform rather
c       than by FFT.
        DO I=1,N
          DO J=1,I
            TEMPM=DICT(X(I),Y(I),DKX,DKY,D(0,0,J),NX,NY)*W(I)
            M(I,J)=TEMPM
            M(J,I)=TEMPM
          END DO
        END DO
c
c       solve Mx=w.
```

```
      CALL MATSOL(M,MWORK,V,W,N)
c
c     determine <g>.
      GDC=DOT(F,V,W,N)/SUM(V,W,N)
c
c     solve My=f.
      DO I=1,N
        LAMBDA(I)=W(I)*F(I)
      END DO
      CALL MATSOL(M,MWORK,LAMBDA,LAMBDA,N)
c
c     determine Lagrange multipliers.
      DO I=1,N
        LAMBDA(I)=GDC*V(I)-LAMBDA(I)
      END DO
c
c     Determine the positive-frequency part of the transform of
c     the fit function, putting it into the first panel of D.
      DO KX=0,NX
        DO KY=0,NY
          D(KX,KY,1)=-D(KX,KY,1)*LAMBDA(1)
        END DO
      END DO
      DO J=2,N
        DO KX=0,NX
          DO KY=0,NY
            D(KX,KY,1)=D(KX,KY,1)-D(KX,KY,J)*LAMBDA(J)
          END DO
        END DO
      END DO
      D(0,0,1)=GDC
c
c     done
      END
```

```
      SUBROUTINE FFT2D(F,M,N,S)
c
c
c
c     1.0  PURPOSE
c
c     Performs a 2-dimensional,  radix-2  Fast  Fourier  Transform
c     (FFT)  in memory.  This program is efficient only if the 2-D
c     array can be stored in the user's physical memory space.  If
c     the  array  extends  into virtual memory space on disk, then
c     extensive paging will be incurred while doing the  transform
c     in  the  second  dimension,  which  is the part performed by
c     calls to FFT2.
c
c
c
c     2.0  ARGUMENTS
c
c
c          1.  F - (input/output) complex*8 array of input  values
c              that  is  Fourier  transformed  in  place  into the
c              output values.  Dimensioned (M,N).
c
c          2.  M  -  (input)  integer  subscript  range  of  first
c              dimension of F.  Must be an integral power of 2, or
c              FFT will issue an error message and STOP.
c
c          3.  N -  (input)  integer  subscript  range  of  second
c              dimension of F.  Must be an integral power of 2, or
c              FFT2 will issue an error message and STOP.
c
c          4.  S - (input) real*4 sign of transform.   Must  equal
c              +1.   or  -1.,  or FFT will issue an error messages
c              and STOP.
c
c
c
c
c     3.0  AUTHOR
c
c
c              R.  Watts
c              U.S.  Geological Survey
c              P.O.  Box 25046, Mail Stop 964
c              Denver, CO 80225
c
c
c
c
c     4.0  TESTING SUMMARY
c
c     FFT2D is a simple driver to call FFT and FFT2.  FFT does the
c     transforms   in   the  first  dimension  (whose  points  are
c     contiguous).   FFT2  does  the  transforms  in  the   second
c     dimension  (whose  points  are non-contiguous).  The program
```

```
c     was written in 1976 and run on a Honeywell  Multics  system,
c     then  recompiled  and tested in December, 1982, on a Digital
c     Equipment Corporation VAX 11/780.  The program  conforms  to
c     FORTRAN  '77  standards  and  contains no known non-standard
c     constructs.
c
c
c
c     5.0  CALLS SUBROUTINES:
c
c
c             -  FFT
c
c             -  FFT2
c
c
c
c
c     6.0  USAGE
c
c     6.1  Definition Of Result
c
c     The result F in terms of the input  f  (which  occupies  the
c     same storage) is:
c
c                   M-1   N-1
c                   ----  ----                      2
c                   \     \              4iS(pi) klmn/MN
c         F(k,l) =  /     /     f(m,n) e
c                   ----  ----
c                   m=0   n=0
c
c         for k = 0,1,...,M-1 and l = 0,1,...,N-1.
c
c     where i is the square root of -1.
c
c
c
c     6.2  Transform Weighting
c
c     6.2.1  Unweighted Results - If FFT2D is called with S = +1.,
c     then  called again with S = -1., and no weighting is applied
c     to the array, then the result will  be  the  original  array
c     multiplied by M*N.
c
c
c
c     6.2.2  Further Information - See  FFT  for  information   on
c     appropriate  weights  to  apply  to  make a scaled transform
c     pair.
c
c
c
c     6.3  Subscript Range
c
```

```
c     This program only uses the subscript ranges prescribed by  M
c     and  N.   The  calling  program can use a declaration of the
c     form
c
c             COMPLEX F(m1:m2,n1:n2)
c
c     and the only restrictions are:
c
c             m2-m1+1=M
c             n2-n1+1=N
c
c     which is to say, M and N are the number of elements in  each
c     dimension.
c
c
c
c     6.4  Efficiency
c
c     This program is intended to work in physical memory.   If  F
c     is  larger  than  the  user's share of physical memory, then
c     extensive paging will occur during the calls to FFT2.
c
c
c     ------------------------------------------------------------
c
      COMPLEX*8 F(0:M-1,0:N-1)
c
c     perform transforms in 1st dimension.
      DO I=0,N-1
        CALL FFT(F(0,I),M,S)
      END DO
c
c     perform transforms in 2nd dimension.
      DO J=0,M-1
        CALL FFT2(F(J,0),M,N,S)
      END DO
c
c     done
      END
```

```
      SUBROUTINE FFT(F,N,S)
c
c
c
c     1.0  PURPOSE
c
c     Computes  the  radix-2  Fast  Fourier  Transform,  using  an
c     in-place algorithm.  Output F in terms of input f is:
c
c                    N-1
c                    ---
c                    \          iSjk2(pi)/N
c           F(k) =   /    f(j) e
c                    ---
c                    j=0
c
c     where F and f are tabulated transform values that occupy the
c     same storage.  i is the square root of -1.  S is the sign of
c     the transform, +/-1.  N is restriced to be an integral power
c     of 2.
c
c
c
c     2.0  ARGUMENTS
c
c
c          1.  F - (input/output) complex input array,  N  complex
c              elements.   The  transform is done in place, so the
c              result is returned to the calling program in F.
c
c          2.  N - (input) the (integer) number of elements in  F.
c              The  subscripts actually used in this routine are 0
c              to N-1.  N is  restricted  to  take  on  values  of
c              integral  powers  of  2.  If N is not so specified,
c              then an error message is issued and the program  is
c              STOPped.
c
c          3.  S  -   (input)   forward/reverse   transform   sign
c              indicator.  Must have a value of +1.  or -1..  If S
c              is specified with any other value,  then  an  error
c              message is issued and the program is STOPped.
c
c
c
c
c     3.0  AUTHOR
c
c
c          R.  Watts
c          U.S.  Geological Survey
c          P.O.  Box 25046, Mail Stop 964
c          Denver, CO 80225
c
c
c
```

```
c
c     4.0  TESTING SUMMARY
c
c     This is a variation of a routine that has been used for many
c     years;  the author obtained its ancestor from Ralph Wiggins
c     at MIT.  This version has been coded to use some nice
c     features of FORTRAN '77, such as subscripts that start at
c     zero and logically controlled DO loops.
c
c
c
c     5.0  CALLS SUBROUTINES:
c
c     None.
c
c
c
c     6.0  USAGE NOTES
c
c     6.1  Domain Spacings
c
c     If the distance between samples in the input  domain  is  D,
c     then  the  distance  between samples in the output domain is
c     D', given by
c
c                  2*pi
c            D' = -------
c                  D*N
c
c
c
c
c     6.2  Weights
c
c     6.2.1  This incarnation of FFT has no weighting applied  for
c     forward/reverse  transformation,  so  the  user  can  define
c     "forward" with whichever sign he prefers.
c
c
c
c     6.2.2  To use FFT to estimate a Fourier integral,  the  user
c     should  multiply the input or the output by the input domain
c     spacing.  To make a transform pair, Fourier theory  requires
c     either  the  "forward"  or  the  "reverse"  integral  to  be
c     multiplied by 1/(2*pi), or both integrals to  be  multiplied
c     by 1/sqrt(2*pi).
c
c
c
c     6.2.3  If FFT is called once with S = +1., then again with S
c     =  -1., or vice-versa, the result will be the original array
c     multiplied by N.  This result is apparent from the foregoing
c     discussion,  which says the round-trip transform pair should
c     be multiplied by (DD')/(2*pi), which is equal to  1/N,  from
c     the formula in paragraph 6.1 .
```

```
c
c
c     ---------------------------------------------------------------
c
      COMPLEX F(0:N),W,WP,X,Y
      INTEGER GROUPSTART,GROUPSIZE,HALFSIZE
c
c     check validity of arguments
      IF (ABS(S) .NE. 1.) THEN
        WRITE(*,*)' Illegal argument S passed to FFT.'
        STOP
      END IF
      NT=1
      DO WHILE (NT .LT. N)
        NT=2*NT
      END DO
      IF (NT .GT. N) THEN
        WRITE(*,*)' FFT requires N to be a power of 2.'
        STOP
      END IF
c
c     Arguments are OK.
c
c     Do subscript bit reversal.
c
c     IREV is bit-reversed counter - set initial value:
      IREV=N/2
c
c     zero and N-1 are their own bit reverses; do the rest
      DO I=1,N-2
c
c       do the reversal only once  (since forward and reverse
c       counters each take on any given value one time):
        IF (IREV .GT. I) THEN
          X=F(IREV)
          F(IREV)=F(I)
          F(I)=X
        END IF
c
c       step the bit-reverse counter, starting at most significant
c       bit:
        J=N/2
        DO WHILE (IREV .GE. J)
          IREV=IREV-J
          J=J/2
        END DO
        IREV=IREV+J
c
c     end of bit reversal.
      END DO
c
c
c     set up for transform.
c
c     W is the twiddle factor base, given by
```

```
c         W = exp (i*2*pi/GROUPSIZE)
      W=(-1.,0.)
c
c     groupsize is the length of the subtransform:
      GROUPSIZE=2
c
c     loop through the groupsizes.
      DO WHILE (GROUPSIZE .LE. N)
c
c       set half-groupsize.
        HALFSIZE=GROUPSIZE/2
c
c       loop for each group.
        DO GROUPSTART=0,N-1,GROUPSIZE
c
c         set the power of W, WP, which is the "twiddle factor".
          WP=(1.,0.)
c
c         loop through the halfgroup.
          DO I=GROUPSTART,GROUPSTART+HALFSIZE-1
c
c           apply the twiddle factor to the second halfgroup,
c           then add and subtract to get the next level output.
            J=I+HALFSIZE
            X=F(I)
            Y=F(J)*WP
            F(I)=X+Y
            F(J)=X-Y
c
c           next twiddle factor:
            WP=WP*W
c
c         end of loop through halfgroup.
          END DO
c
c       end of loop through all groups.
        END DO
c
c       W (twiddle factor base) for next size transform:
        W=SQRT(W)
        IF (AIMAG(W)*S .LT. 0.) W=-W
c
c       next group size.
        GROUPSIZE=2*GROUPSIZE
c
c     end of loop through all groupsizes.
      END DO
c
c     done.
      END
```

```
      SUBROUTINE FFT2(F,M,N,S)
c
c
c
c     1.0  PURPOSE
c
c     Computes  the  radix-2  Fast  Fourier  Transform,  using  an
c     in-place  algorithm.   This  program  is  a duplicate of FFT
c     except that it is coded to access every Mth element  of  the
c     input array F rather than consecutive elements.  Output F in
c     terms of input f is:
c
c                     N-1
c                     ---
c                     \          iSjk2(pi)/N
c            F(k) =   /    f(j) e
c                     ---
c                     j=0
c
c     where F and f are tabulated transform values that occupy the
c     same storage.  i is the square root of -1.  S is the sign of
c     the transform, +/-1.  N is restriced to be an integral power
c     of 2.
c
c
c
c     2.0  ARGUMENTS
c
c
c          1.   F - (input/output) complex input array, M*N complex
c               elements.   The  transform is done in place, so the
c               result is returned to the calling program in F.
c
c          2.   M -  (input)  the  (integer)  skipping  factor  for
c               accessing  the  array  F.  If M = 1, then FFT2 does
c               exactly  the  same  operation  as  FFT,   accessing
c               consecutive  elements of F.  If M = 2, every second
c               element of F is accessed, etc..
c
c          3.   N - (input) the (integer) number of elements in  F.
c               The  subscripts actually used in this routine are 0
c               to N-1.  N is  restricted  to  take  on  values  of
c               integral  powers  of  2.  If N is not so specified,
c               then an error message is issued and the program  is
c               STOPped.
c
c          4.   S  -   (input)   forward/reverse   transform   sign
c               indicator.  Must have a value of +1.  or -1..  If S
c               is specified with any other value,  then  an  error
c               message is issued and the program is STOPped.
c
c
c
c
c     3.0  AUTHOR
```

```
c
c
c           R.  Watts
c           U.S.  Geological Survey
c           P.O.  Box 25046, Mail Stop 964
c           Denver, CO 80225
c
c
c
c
c     4.0  TESTING SUMMARY
c
c     This is a variation of a routine that has been used for many
c     years;  the author obtained its ancestor from Ralph Wiggins
c     at MIT.  This version  has  been  coded  to  use  some  nice
c     features  of  FORTRAN  '77, such as subscripts that start at
c     zero and logically controlled DO loops.
c
c
c
c     5.0  CALLS SUBROUTINES:
c
c     None.
c
c
c
c     6.0  USAGE NOTES
c
c     6.1  Domain Spacings
c
c     If the distance between samples in the input  domain  is  D,
c     then  the  distance  between samples in the output domain is
c     D', given by
c
c                    2*pi
c             D' = -------
c                    D*N
c
c
c
c
c     6.2  Weights
c
c     6.2.1  This incarnation of FFT2 has no weighting applied for
c     forward/reverse  transformation,  so  the  user  can  define
c     "forward" with whichever sign he prefers.
c
c
c
c     6.2.2  To use FFT2 to estimate a Fourier integral, the  user
c     should  multiply the input or the output by the input domain
c     spacing.  To make a transform pair, Fourier theory  requires
c     either  the  "forward"  or  the  "reverse"  integral  to  be
c     multiplied by 1/(2*pi), or both integrals to  be  multiplied
c     by 1/sqrt(2*pi).
```

```
c
c
c
c     6.2.3  If FFT2 is called once with S = +1., then again  with
c     S  =  -1.,  or  vice-versa,  the result will be the original
c     array multiplied by N.  This result  is  apparent  from  the
c     foregoing  discussion,  which  says the round-trip transform
c     pair should be multiplied by (DD')/(2*pi), which is equal to
c     1/N, from the formula in paragraph 6.1 .
c
c
c     -----------------------------------------------------------
c
c     declarations:
      COMPLEX F(M,0:N-1),W,WP,X,Y
      INTEGER GROUPSTART,GROUPSIZE,HALFSIZE
c
c     check validity of arguments
      IF (ABS(S) .NE. 1.) THEN
        WRITE(*,*)' Illegal argument S passed to FFT2.'
        STOP
      END IF
      NT=1
      DO WHILE (NT .LT. N)
        NT=2*NT
      END DO
      IF (NT .GT. N) THEN
        WRITE(*,*)' FFT2 requires N to be a power of 2.'
        STOP
      END IF
c
c     Arguments are OK.
c
c     Do subscript bit reversal.
c
c     irev is bit-reversed counter - set initial value:
      IREV=N/2
c
c     zero and n-1 are their own bit reverses; do the rest
      DO I=1,N-2
c
c       do the reversal only once  (since forward and reverse
c       counters each take on any given value one time):
        IF (IREV .GT. I) THEN
          X=F(1,IREV)
          F(1,IREV)=F(1,I)
          F(1,I)=X
        END IF
c
c       step the bit-reverse counter, starting at most significant
c       bit:
        J=N/2
        DO WHILE (IREV .GE. J)
          IREV=IREV-J
          J=J/2
```

```
          END DO
          IREV=IREV+J
c
c       end of bit reversal.
        END DO
c
c
c       set up for transform.
c
c       w is the twiddle factor base, given by
c         w = exp (i*2*pi/groupsize)
        W=(-1.,0.)
c
c       groupsize is the length of the subtransform:
        GROUPSIZE=2
c
c       loop through the groupsizes.
        DO WHILE (GROUPSIZE .LE. N)
c
c         set half-groupsize.
          HALFSIZE=GROUPSIZE/2
c
c         loop for each group.
          DO GROUPSTART=0,N-1,GROUPSIZE
c
c           set the power of w, wp, which is the "twiddle factor".
            WP=(1.,0.)
c
c           loop through the halfgroup.
            DO I=GROUPSTART,GROUPSTART+HALFSIZE-1
c
c             apply the twiddle factor to the second halfgroup,
c             then add and subtract to get the next level output.
              J=I+HALFSIZE
              X=F(1,I)
              Y=F(1,J)*WP
              F(1,I)=X+Y
              F(1,J)=X-Y
c
c             next twiddle factor:
              WP=WP*W
c
c           end of loop through halfgroup.
            END DO
c
c         end of loop through all groups.
          END DO
c
c         w (twiddle factor base) for next size transform:
          W=SQRT(W)
          IF (AIMAG(W)*S .LT. 0.) W=-W
c
c         next group size.
          GROUPSIZE=2*GROUPSIZE
c
```

```
c     end of loop through all groupsizes.
      END DO
c
c     done.
      END
```

```
      SUBROUTINE DCT(X,Y,DKX,DKY,D,NX,NY,WIM)
c
c
c
c     1.0  PURPOSE
c
c     Computes a Discrete Cosine Transform as described below:
c
c     If we have a two-dimensional  sequence  of  delta  functions
c     that satisfy the conditions
c
c          1.  Even symmetry about zero in both x and y.
c
c          2.  Periodicity of 2*pi/DKX and 2*pi/DKY, respectively,
c              in the x and y dimensions.
c
c     then that sequence can be  represented  as  an  infinite  2-
c     dimensional  Fourier  cosine series.  DCT computes the first
c     NX columns and NY  rows  (i.e.,  a  rectangle  in  the  low-
c     frequency part) of the 2-dimensional cosine series.
c
c
c
c     2.0  ARGUMENTS
c
c
c          1.  X - (input) x coordinate of one of  the  series  of
c              delta functions.
c
c          2.  Y - (input) y coordinate of one of  the  series  of
c              delta functions.
c
c          3.  DKX - (input) frequency interval of  cosine  series
c              in x dimension.
c
c          4.  DKY - (input) frequency interval of  cosine  series
c              in y dimension.
c
c          5.  D - output array to hold 2-D cosine series.  Real*4
c              array dimensioned (0:NX,0:NY).
c
c          6.  NX - (input) integer number of frequency components
c              to  determine  in x direction.  Maximum x frequency
c              in output 2-D series is NX*DKX.
c
c          7.  NY - (input) integer number of frequency components
c              to  determine  in y direction.  Maximum y frequency
c              in output 2-D series is NY*DKY.
c
c          8.  WIM - (input) real*4 weight due to  images  of  the
c              point  at  (x,y).   WIM  is a multiplicative factor
c              applied to the output.  WIM will ordinarily have  a
c              value  of  4 (for the delta function plus its three
c              images), but is reduced by a factor of 2  for  each
c              symmetry-line  that  (X,Y)  occupies.  For example,
```

(0,0) lies on a symmetry line in the x dimension (reducing WIM to 2) and on a symmetry line in the y dimension (further reducing WIM to 1).

3.0 AUTHOR

R. Watts
U.S. Geological Survey
P.O. Box 25046, Mail Stop 964
Denver, CO 80225

4.0 TESTING SUMMARY

DCT was written in December, 1981, and tested on a Digital Equipment Corporation VAX 11/780 computer, using DEC's VAX FORTRAN compiler. The program is intended to conform to FORTRAN '77 standards, and contains no known non-FORTRAN '77 constructs.

5.0 CALLS SUBROUTINES:

None.

6.0 USAGE

6.1 Notes

This program is more efficient than the FFT for computing cosines at regular intervals. Note that DCT does not compute a complete transform, since the input function is restricted to be a single delta function plus its images. The true transform of the delta functions is not band-limited, so the use of DCT must be followed by some band-limiting procedure. When DCT is called by FOURFIT (for which it was originally coded), a factor of 1/(kx**2 + ky**2)**2, which is a strong band-limiting factor, is applied.

6.2 Computed Output

The Fourier cosine series computed by DCT is

```
c
c                        WIM*DKX*DKY
c         D(i,j) = ----------------- cos(X*DKX*i) cos(Y*DKY*j)
c                        (pi**2)*w(i)*w(j)
c
c         i = (0,1,...,NX);  j = (0,1,...,NY).
c
c       where w(k)=2 for k=0 and w(k)=1 otherwise.
c
c
c
c       6.3  Inverse Series
c
c       The output series is a partial representation of  the  input
c       delta-function  and  its symmetric images, if there are any.
c       The full representation is given by:
c
c                     +inf   +inf
c                     ----   ----
c                     \      \
c            d(x,y) = /      /     D(i,j) cos(x*DKX*i) cos(y*DKY*j)
c                     ----   ----
c                     i=0    j=0
c
c       ------------------------------------------------------------
c
c       declarations:
        REAL*4 D(0:NX,0:NY)
        COMPLEX*8 COSX,COSY,COSGENX,COSGENY
        DATA PI/3.14159265/
c
c       weight factors
        WT=WIM*DKX*DKY/PI**2
c
c       complex exponentials for cosine generation
        COSGENX=EXP(CMPLX(0.,DKX*X))
        COSGENY=EXP(CMPLX(0.,DKY*Y))
c
c       do the transform.
        D(0,0)=WT/4.
        COSX=WT/2.
        DO KX=1,NX
                COSX=COSX*COSGENX
                D(KX,0)=REAL(COSX)
        END DO
        COSY=(1.,0.)
        DO KY=1,NY
                COSY=COSY*COSGENY
                D(0,KY)=WT/2.*REAL(COSY)
                COSX=WT
                DO KX=1,NX
                        COSX=COSX*COSGENX
                        D(KX,KY)=REAL(COSX)*REAL(COSY)
                END DO
        END DO
```

```
c
c        done
         END
```

```
      REAL*4 FUNCTION DICT(X,Y,DKX,DKY,D,NX,NY)
c
c
c
c     1.0  PURPOSE
c
c     DICT  (Discrete  Inverse  Cosine  Transform)  is  a  real*4
c     function  that  evaluates  a  2-dimensional  Fourier  cosine
c     series at one output point.  Since the output is required at
c     one point rather than over the entire transform domain, DICT
c     is more efficient than a 2-dimensional FFT.
c
c
c
c     2.0  ARGUMENTS
c
c
c          1.  X  -  (input)  real*4  x  coordinate  of  point  of
c              evaluation.
c
c          2.  Y  -  (input)  real*4  y  coordinate  of  point  of
c              evaluation.
c
c          3.  DKX - (input) real*4 frequency interval  in  the  x
c              dimension.
c
c          4.  DKY - (input) real*4 frequency interval  in  the  y
c              dimension.
c
c          5.  D - (input) coefficients of the 2-D cosine  series,
c              real*4 array dimensioned (0:NX,0:NY).
c
c          6.  NX - (input) integer number  of  non-zero-frequency
c              terms  to  compute  in  the  x  dimension.  Maximum
c              frequency in the x dimension is NX*DKX.
c
c          7.  NY - (input) integer number  of  non-zero-frequency
c              terms  to  compute  in  the  y  dimension.  Maximum
c              frequency in the y dimension is NY*DKY.
c
c
c
c
c     3.0  AUTHOR
c
c
c              R.  Watts
c              U.S.  Geological Survey
c              P.O.  Box 25046, Mail Stop 964
c              Denver, CO 80225
c
c
c
c
c     4.0  TESTING SUMMARY
```

```
c
c       DICT was written in December, 1981, and tested on a  Digital
c       Equipment  Corporation  VAX 11/780 computer, using DEC's VAX
c       FORTRAN compiler.  The program is  intended  to  conform  to
c       FORTRAN  '77  standards  and  contains  no known non-fortran
c       constructs.
c
c
c
c       5.0  CALLS SUBROUTINES:
c
c       None
c
c
c
c       6.0  USAGE NOTES
c
c       The function value computed for DICT is given by:
c
c                      NX      NY
c                     ----    ----
c                     \       \
c            DICT  =  /       /      D(i,j) cos(X*DKX*i) cos(Y*DKY*j)
c                     ----    ----
c                     i=0     j=0
c
c
c       -----------------------------------------------------------
c
c       declarations:
        REAL*4 D(0:NX,0:NY)
        COMPLEX*8 COSX,COSY,COSGENX,COSGENY
c
c       complex exponentials for cosine generation
        COSGENX=EXP(CMPLX(0.,X*DKX))
        COSGENY=EXP(CMPLX(0.,Y*DKY))
c
c       addemup
        SUM=0.
        COSX=(1.,0.)
        DO KX=0,NX
          COSY=(1.,0.)
          DO KY=0,NY
            SUM=SUM+REAL(COSX)*REAL(COSY)*D(KX,KY)
            COSY=COSY*COSGENY
          END DO
          COSX=COSX*COSGENX
        END DO
c
c       done.
        DICT=SUM
        END
```

```
      SUBROUTINE MATSOL(A,AT,B,C,N)
c
c
c
c     1.0  PURPOSE
c
c     Solves the matrix equation Ab=c,  where  A  is  a  real  NxN
c     matrix,  c  is  a  given vector, and b is an unknown vector.
c     Solution is by Gaussian elimination with pivoting.
c
c
c
c     2.0  ARGUMENTS
c
c
c          1.  A - (input) real*4 array dimensioned (N,N).  Matrix
c              for  solution,  remains  unchanged  by operation of
c              subroutine MATSOL.
c
c          2.  AT - (work) real*4 array dimensioned  (N,N).   Work
c              array  used  to  hold  A and its modifications that
c              occur during Gaussian elimination.
c
c          3.  B - (output) real*4 vector dimensioned (N).  Result
c              of solution of the matrix equation.
c
c          4.  C - (input) real*4 vector dimensioned (N).   Right-
c              side vector in matrix equation.
c
c          5.  N - (input) integer size of matrix-vector problem.
c
c
c
c
c     3.0  AUTHOR
c
c
c              R.  Watts
c              U.S.  Geological Survey
c              P.O.  Box 25046, Mail Stop 964
c              Denver, CO 80225
c
c
c
c
c     4.0  TESTING SUMMARY
c
c     MATSOL was written in December, 1981.  It was compiled on  a
c     Digital  Equipment  Corporation  VAX  11/780 computer, using
c     DEC's VAX FORTRAN '77 compiler.  MATSOL  contains  no  known
c     non-FORTRAN '77 constructs.
c
c     MATSOL has been tested with adequate results with N as great
c     as 50.
c
```

```
c
c
c       5.0  CALLS SUBROUTINES:
c
c       None.
c
c
c
c       6.0  USAGE NOTES
c
c       Gaussian elimination  is  used,  with  pivoting  around  the
c       largest   element   in   the   elimination  column.   Output
c       accuracy's dependence on N has not  been  established.   All
c       operations  are  done  in single precision.  No provision is
c       made for detection of singular matrices;   a  divide-by-zero
c       exception occurs if A is singular.
c
c
c       -------------------------------------------------------------
c
        REAL*4 A(N,N),AT(N,N),B(N),C(N)
c
c       copy A into AT, C into B.
        DO I=1,N
          B(I)=C(I)
          DO J=1,N
            AT(I,J)=A(I,J)
          END DO
        END DO
c
c       scan through elimination pivot points.
        DO IPIVOT=1,N-1
c
c         search for largest number in pivot column.
          CMAX=ABS(AT(IPIVOT,IPIVOT))
          IMAX=IPIVOT
          DO I=IPIVOT+1,N
            CTEMP=ABS(AT(I,IPIVOT))
            IF(CTEMP.GT.CMAX) THEN
              CMAX=CTEMP
              IMAX=I
            END IF
          END DO
c
c         swap it into pivot position.
          IF(IMAX.NE.IPIVOT) THEN
            DO J=IPIVOT,N
              T=AT(IPIVOT,J)
              AT(IPIVOT,J)=AT(IMAX,J)
              AT(IMAX,J)=T
            END DO
            T=B(IPIVOT)
            B(IPIVOT)=B(IMAX)
            B(IMAX)=T
          END IF
```

```
c
c         do the elimination.
          PVAL=AT(IPIVOT,IPIVOT)
          BVAL=B(IPIVOT)
c
c         scan down the rows.
          DO IROW=IPIVOT+1,N
            RATIO=AT(IROW,IPIVOT)/PVAL
c
c           scan across row.
            DO J=IPIVOT+1,N
c
c             eliminate.
              AT(IROW,J)=AT(IROW,J)-AT(IPIVOT,J)*RATIO
c
            END DO
c
c           and adjust B in the same way.
            B(IROW)=B(IROW)-BVAL*RATIO
c
          END DO
c
        END DO
c
c       back substitution.
        B(N)=B(N)/AT(N,N)
c
c       scan up the rows.
        DO IPIVOT=N-1,1,-1
c
c         add up the known parts.
          SUM=B(IPIVOT)
          DO J=IPIVOT+1,N
            SUM=SUM-AT(IPIVOT,J)*B(J)
          END DO
c
c         determine the unknown.
          B(IPIVOT)=SUM/AT(IPIVOT,IPIVOT)
c
        END DO
c
        END
```

```
      SUBROUTINE TOCOMPLEX(DR,DC,LX,LY,MX,MY)
c
c
c
c     1.0  PURPOSE
c
c     Takes the real, 2-dimensional array DR and puts it into  the
c     bottom  corner  of  the  complex,  2-dimensional  array  DC,
c     converting the real numbers into complex numbers  with  zero
c     imaginary part.
c
c
c
c     2.0  ARGUMENTS
c
c
c          1.  DR - (input) array of real numbers that are  to  be
c              made   complex  and  placed  in  the  corresponding
c              locations of the complex array DC (which must  have
c              the  same or larger dimension).  Dimensioned (0:LX,
c              0:LY).
c
c          2.  DC - (output) complex array,  dimensioned  (0:MX-1,
c              0:MY-1).
c
c          3.  LX - (input) integer dimension for DR.
c
c          4.  LY - (input) integer dimension for DR.
c
c          5.  MX - (input) integer dimension  for  DC.   Must  be
c              greater than LX.
c
c          6.  MY - (input) integer dimension  for  DC.   Must  be
c              greater than LY.
c
c
c
c
c     3.0  AUTHOR
c
c
c              R.  Watts
c              U.S.  Geological Survey
c              P.O.  Box 25046, Mail Stop 964
c              Denver, CO 80225
c
c
c
c
c     4.0  TESTING SUMMARY
c
c     TOCOMPLEX was compiled and tested in December,  1981,  on  a
c     Digital  Equipment  Corporation  VAX  11/780 computer, using
c     DEC's VAX  FORTRAN  '77 compiler.  It  contains  no  known
c     non-FORTRAN '77 constructs.
```

```
c
c
c
c       5.0  CALLS SUBROUTINES:
c
c       None.
c
c
c
c       6.0  USAGE NOTES
c
c       TOCOMPLEX is called by FOURGRID  in  preparation  for
c       generating the grid using a 2-dimensional FFT.
c
c       TOCOMPLEX works if DC(0,0) and DR(0,0) are the same  storage
c       location    (i.e.,    an   in-place   real-to-complex   array
c       conversion).  This is the reason the DO loops are  run  from
c       high to low subscript.
c
c       ------------------------------------------------------------
c
c       declarations:
        REAL*4 DR(0:LX,0:LY)
        COMPLEX*8 DC(0:MX-1,0:MY-1)
c
c       check the array sizes.
        IF(MX.LE.LX .OR. MY.LT.LY) THEN
          WRITE(*,'(1XA)') 'Output array dimensions are smaller than '//
     .      'input array dimensions in TOCOMPLEX.'
          STOP
        END IF
c
c       copy the non-zero part, filling line ends with zeroes.
        DO IY=LY,0,-1
          DO IX=LX,0,-1
            DC(IX,IY)=CMPLX(DR(IX,IY),0.)
          END DO
          DO IX=LX+1,MX-1
            DC(IX,IY)=(0.,0.)
          END DO
        END DO
c
c       fill remainder of DC with zeroes.
        DO IY=LY+1,MY-1
          DO IX=0,MX-1
            DC(IX,IY)=(0.,0.)
          END DO
        END DO
c
c       done.
        END
```

```
      SUBROUTINE TOREAL(DC,DR,LX,LY)
c
c
c
c     1.0  PURPOSE
c
c     Performs the following operations on the complex  output  of
c     an FFT:
c
c          1.  Keeps only the real part of the  2-dimensional  FFT
c              output, and only the part with even symmetry in the
c              x dimension.  This  corresponds  to  terms  of  the
c              form:
c
c                            i(k x + k y)     i(-k x + k y)
c                 1             x     y           x     y
c                ---  Real [ e             + e               ]
c                 2
c
c              This reduces to terms of the form:
c
c                cos(k x) cos(k y)
c                     x        y
c
c
c          2.  Moves the result into the low-subscript  corner  of
c              the array DR.
c
c     By  keeping  the  specified  terms,  the  operation   of   a
c     2-dimensional FFT followed by a call to TOREAL is equivalent
c     to a 2-dimensional cosine transform.
c
c
c
c     2.0  ARGUMENTS
c
c
c          1.  DC - (input) array of complex numbers that  are  to
c              be  made  real by discarding the imaginary part and
c              keeping the part that is even in x, and placing the
c              results  in the corresponding locations of the real
c              array DR.  Dimensioned (0:2*LX-1,0:2*LY-1).  DC  is
c              assumed to be arranged in the usual fashion for FFT
c              arrays, with  positive  frequencies  in  the  range
c              (0:LX,0:LY)  and  negative frequencies in the range
c              (LX+1:2*LX-1,LY+1:2*LY-1).
c
c          2.  DR - (output)  complex  array,  dimensioned  (0:LX,
c              0:LY).   Normally  DR  occupies the same storage as
c              DC, since compression-in-place is possible.
c
c          3.  LX - (input) integer dimension for DC and DR.
c
c          4.  LY - (input) integer dimension for DC and DR.
c
```

```
c
c
c
c     3.0  AUTHOR
c
c
c             R.  Watts
c             U.S.  Geological Survey
c             P.O.  Box 25046, Mail Stop 964
c             Denver, CO 80225
c
c
c
c
c     4.0  TESTING SUMMARY
c
c     TOREAL was compiled and  tested  in  December,  1981,  on  a
c     Digital  Equipment  Corporation  VAX  11/780 computer, using
c     DEC's VAX  FORTRAN  '77  compiler.   It  contains  no  known
c     non-FORTRAN '77 constructs.
c
c
c
c     5.0  CALLS SUBROUTINES:
c
c     None.
c
c
c
c     6.0  USAGE NOTES
c
c     TOREAL is called by FOURGRID to keep the  desired  parts  of
c     the output of a 2-dimensional FFT.
c
c     TOREAL works if DC(0,0) and DR(0,0)  are  the  same  storage
c     location    (i.e.,    an    in-place    complex-to-real    array
c     conversion).
c
c
c     -------------------------------------------------------------
c
c     declarations:
      REAL*4 DR(0:LX,0:LY)
      COMPLEX*8 DC(0:2*LX-1,0:2*LY-1)
c
c     do the work.
      NX=2*LX
      DO IY=0,LY
        DR(0,IY)=REAL(DC(0,IY))
        DO IX=1,LX
          DR(IX,IY)=(REAL(DC(IX,IY))+REAL(DC(NX-IX,IY)))/2.
        END DO
      END DO
c
c     done.
```

END

```
      REAL*4 FUNCTION DOT(F,G,W,N)
c
c
c
c     1.0  PURPOSE
c
c     Computes the dot product of F and G,  weighted  by  W.   The
c     function value is:
c
c              N
c             ----
c             \
c       DOT = /      F  G  W    .
c             ----    i  i  i
c             i=1
c
c
c
c
c     2.0  ARGUMENTS
c
c
c          1.  F - (input) real*4 vector dimensioned  (N).   First
c              factor of dot product.
c
c          2.  G - (input) real*4 vector dimensioned (N).   Second
c              factor of dot product.
c
c          3.  W  -  (input)  real*4   vector  dimensioned   (N).
c              Weighting factor of dot product.
c
c          4.  N -  (input)  integer  length  of  vectors  in  dot
c              product.
c
c
c
c
c     3.0  AUTHOR
c
c
c              R.  Watts
c              U.S.  Geological Survey
c              P.O.  Box 25046, Mail Stop 964
c              Denver, CO 80225
c
c
c
c
c     4.0  TESTING SUMMARY
c
c     DOT  was  compiled  and  tested  on  a  Digital  Equipment
c     Corporation VAX 11/780 computer, using DEC's VAX FORTRAN '77
c     compiler.  It contains no known non-FORTRAN '77 constructs.
c
c
```

```
c
c     5.0  CALLS SUBROUTINES:
c
c     None.
c
c
c
c     6.0  USAGE NOTES
c
c     The weighting factor is included as a facility for  FOURFIT,
c     which  applies  weights  to  dot  products  according to the
c     number of images possessed by a data point  in  a  symmetric
c     fitting domain.  See FOURFIT documentation.
c
c     ------------------------------------------------------------
c
c     declarations:
      REAL*4 F(N),G(N),W(N)
c
c     clear the sum.
      SUM=0.
c
c     addemup.
      DO I=1,N
        SUM=SUM+F(I)*G(I)*W(I)
      END DO
c
c     transfer the answer to the function value.
      DOT=SUM
c
c     done.
      END
```

```
      FUNCTION SUM(F,W,N)
c
c
c
c     1.0  PURPOSE
c
c     Computes the sum of F, weighted by W.   The  function  value
c     is:
c
c                 N
c                ----
c                \
c       SUM =    /      F  W  .
c                ----    i  i
c                i=1
c
c     Obviously, this result can also be  considered  as  the  dot
c     product of F and W.
c
c
c
c     2.0  ARGUMENTS
c
c
c          1.  F - (input) real*4 vector dimensioned (N).   Vector
c              to be summed.
c
c          2.  W - (input) real*4 vector dimensioned (N).  Weights
c              to be applied to summation vector.
c
c          3.  N  -  (input)  integer  length  of  summation   and
c              weighting vectors.
c
c
c
c
c     3.0  AUTHOR
c
c
c              R.  Watts
c              U.S.  Geological Survey
c              P.O.  Box 25046, Mail Stop 964
c              Denver, CO 80225
c
c
c
c     4.0  TESTING SUMMARY
c
c     DOT  was  compiled  and  tested  on  a  Digital   Equipment
c     Corporation VAX 11/780 computer, using DEC's VAX FORTRAN '77
c     compiler.  It contains no known non-FORTRAN '77 constructs.
c
c
c
```

```
c     5.0  CALLS SUBROUTINES:
c
c     None.
c
c
c
c     6.0  USAGE NOTES
c
c     This routine is called by FOURFIT to compute summed elements
c     of  vectors,  with  weights  applied  to  compensate for the
c     number of images possessed by each known  data  point.   See
c     FOURFIT documentation.
c
c
c     ------------------------------------------------------------
c
c     declarations:
      REAL*4 F(N),W(N)
c
c     clear the sum.
      SUM=0.
c
c     addemup.
      DO I=1,N
        SUM=SUM+F(I)*W(I)
      END DO
c
c     done.
      END
```