

```

C***** C EASY
TO USE, NO BOUNDS
C*****
C MAIN PROGRAM TO MINIMIZE A FUNCTION (REPRESENTED BY THE ROUTINE SFUN)
C OF N VARIABLES X
C
      DOUBLE PRECISION X(50), F, G(50), W(700)
      EXTERNAL          SFUN
C
C DEFINE SUBROUTINE PARAMETERS
C N - NUMBER OF VARIABLES
C X - INITIAL ESTIMATE OF THE SOLUTION
C F - ROUGH ESTIMATE OF FUNCTION VALUE AT SOLUTION
C LW - DECLARED LENGTH OF THE ARRAY W
C
      OPEN(UNIT=8,FILE='TEST1.OUT',STATUS='NEW')

      N = 4

c      X(I) = I / FLOAT(N+1)
c10     CONTINUE
      x(1)=3.D0
      x(2)=1.D0
      x(3)=3.D0
      x(4)=1.D0
      F = 1.D0
      LW = 700
      CALL TN (IERROR, N, X, F, G, W, LW, SFUN)
      STOP
      END
C
C
      SUBROUTINE SFUN (N, X, F, G)
      DOUBLE PRECISION X(N), G(N), F, T
C
C ROUTINE TO EVALUATE FUNCTION (F) AND GRADIENT (G) OF THE OBJECTIVE
C FUNCTION AT THE POINT X
C
      A=X(2)+X(1)*X(1)
      B=X(4)+X(3)*X(3)
      F=100.*A*A+(1.-X(1))**2+90.*B*B+(1.-X(3))**2
1  +10.1*((X(2)+1.)**2+(X(4)+1.)**2)+19.8*(X(2)+1.)*(X(4)+1.)
      G(1)=2.*(200.*X(1)*A-1.+X(1))
      G(2)=2.*(100.*A+10.1*(X(2)+1.))+9.9*(X(4)+1.)
      G(3)=2.*(180.*X(3)*B-1.0+X(3))
      G(4)=2.*(90.*B+10.1*(X(4)+1.))+9.9*(X(2)+1.)
      RETURN
      END

C      F = 0.D0
c      DO 10 I = 1,N
c          T = X(I) - I
c          F = F + T*T
c          G(I) = 2.D0 * T
c10     CONTINUE

```

```

C      RETURN
C      END
C%% TRUNCATED-NEWTON METHOD:  SUBROUTINES
C   FOR OTHER MACHINES, MODIFY ROUTINE MCHPR1 (MACHINE EPSILON)
C   WRITTEN BY:  STEPHEN G. NASH
C               OPERATIONS RESEARCH AND APPLIED STATISTICS DEPT.
C               GEORGE MASON UNIVERSITY
C               FAIRFAX, VA 22030
C*****
C      SUBROUTINE TN (IERROR, N, X, F, G, W, LW, SFUN)
C      IMPLICIT      DOUBLE PRECISION (A-H,O-Z)
C      INTEGER       IERROR, N, LW
C      DOUBLE PRECISION  X(N), G(N), F, W(LW)
C
C THIS ROUTINE SOLVES THE OPTIMIZATION PROBLEM
C
C      MINIMIZE F(X)
C      X
C
C WHERE X IS A VECTOR OF N REAL VARIABLES.  THE METHOD USED IS
C A TRUNCATED-NEWTON ALGORITHM (SEE "NEWTON-TYPE MINIMIZATION VIA
C THE LANCZOS METHOD" BY S.G. NASH (SIAM J. NUMER. ANAL. 21 (1984),
C PP. 770-778).  THIS ALGORITHM FINDS A LOCAL MINIMUM OF F(X).  IT DOES
C NOT ASSUME THAT THE FUNCTION F IS CONVEX (AND SO CANNOT GUARANTEE A
C GLOBAL SOLUTION), BUT DOES ASSUME THAT THE FUNCTION IS BOUNDED BELOW.
C IT CAN SOLVE PROBLEMS HAVING ANY NUMBER OF VARIABLES, BUT IT IS
C ESPECIALLY USEFUL WHEN THE NUMBER OF VARIABLES (N) IS LARGE.
C
C SUBROUTINE PARAMETERS:
C
C IERROR - (INTEGER) ERROR CODE
C         ( 0 => NORMAL RETURN)
C         ( 2 => MORE THAN MAXFUN EVALUATIONS)
C         ( 3 => LINE SEARCH FAILED TO FIND
C           ( LOWER POINT (MAY NOT BE SERIOUS)
C         (-1 => ERROR IN INPUT PARAMETERS)
C
C N      - (INTEGER) NUMBER OF VARIABLES
C X      - (REAL*8) VECTOR OF LENGTH AT LEAST N; ON INPUT, AN INITIAL
C         ESTIMATE OF THE SOLUTION; ON OUTPUT, THE COMPUTED SOLUTION.
C G      - (REAL*8) VECTOR OF LENGTH AT LEAST N; ON OUTPUT, THE FINAL
C         VALUE OF THE GRADIENT
C F      - (REAL*8) ON INPUT, A ROUGH ESTIMATE OF THE VALUE OF THE
C         OBJECTIVE FUNCTION AT THE SOLUTION; ON OUTPUT, THE VALUE
C         OF THE OBJECTIVE FUNCTION AT THE SOLUTION
C W      - (REAL*8) WORK VECTOR OF LENGTH AT LEAST 14*N
C LW     - (INTEGER) THE DECLARED DIMENSION OF W
C SFUN   - A USER-SPECIFIED SUBROUTINE THAT COMPUTES THE FUNCTION
C         AND GRADIENT OF THE OBJECTIVE FUNCTION.  IT MUST HAVE
C         THE CALLING SEQUENCE
C         SUBROUTINE SFUN (N, X, F, G)
C         INTEGER         N
C         DOUBLE PRECISION X(N), G(N), F
C
C THIS IS AN EASY-TO-USE DRIVER FOR THE MAIN OPTIMIZATION ROUTINE
C LMQN.  MORE EXPERIENCED USERS WHO WISH TO CUSTOMIZE PERFORMANCE
C OF THIS ALGORITHM SHOULD CALL LMQN DIRECTLY.
C

```

```

C-----
C THIS ROUTINE SETS UP ALL THE PARAMETERS FOR THE TRUNCATED-NEWTON
C ALGORITHM.  THE PARAMETERS ARE:
C
C ETA      - SEVERITY OF THE LINESEARCH
C MAXFUN   - MAXIMUM ALLOWABLE NUMBER OF FUNCTION EVALUATIONS
C XTOL     - DESIRED ACCURACY FOR THE SOLUTION X*
C STEPMX   - MAXIMUM ALLOWABLE STEP IN THE LINESEARCH
C ACCRCY   - ACCURACY OF COMPUTED FUNCTION VALUES
C MSGMLVL  - DETERMINES QUANTITY OF PRINTED OUTPUT
C          0 = NONE, 1 = ONE LINE PER MAJOR ITERATION.
C MAXIT    - MAXIMUM NUMBER OF INNER ITERATIONS PER STEP
C
      DOUBLE PRECISION ETA, ACCRCY, XTOL, STEPMX, DSQRT, MCHPR1
      EXTERNAL          SFUN
C
C SET UP PARAMETERS FOR THE OPTIMIZATION ROUTINE
C
      MAXIT = N/2
      IF (MAXIT .GT. 50) MAXIT = 50
      IF (MAXIT .LE. 0) MAXIT = 1
      MSGMLVL = 1
      MAXFUN = 150*N
      ETA = .25D0
      STEPMX = 1.D1
      ACCRCY = 1.D2*MCHPR1()
      XTOL = DSQRT(ACCRCY)
C
C MINIMIZE THE FUNCTION
C
      CALL LMQN (IERROR, N, X, F, G, W, LW, SFUN,
*             MSGMLVL, MAXIT, MAXFUN, ETA, STEPMX, ACCRCY, XTOL)
C
C PRINT THE RESULTS
C
      IF (IERROR .NE. 0) WRITE(*,800) IERROR
      WRITE(*,810) F
      IF (MSGMLVL .LT. 1) RETURN
      WRITE(*,820)
      NMAX = 10
      IF (N .LT. NMAX) NMAX = N
      WRITE(*,830) (I,X(I),I=1,NMAX)
      RETURN
800  FORMAT(//,' ERROR CODE =', I3)
810  FORMAT(//,' OPTIMAL FUNCTION VALUE = ', 1PD22.15)
820  FORMAT(10X, 'CURRENT SOLUTION IS (AT MOST 10 COMPONENTS)', /,
*       14X, 'I', 11X, 'X(I)')
830  FORMAT(10X, I5, 2X, 1PD22.15)
      END
C
C
      SUBROUTINE TNBC (IERROR, N, X, F, G, W, LW, SFUN, LOW, UP, IPIVOT)
      IMPLICIT          DOUBLE PRECISION (A-H,O-Z)
      INTEGER          IERROR, N, LW, IPIVOT(N)
      DOUBLE PRECISION X(N), G(N), F, W(LW), LOW(N), UP(N)
C
C THIS ROUTINE SOLVES THE OPTIMIZATION PROBLEM

```

```

C
C   MINIMIZE      F(X)
C     X
C   SUBJECT TO   LOW <= X <= UP
C
C WHERE X IS A VECTOR OF N REAL VARIABLES.  THE METHOD USED IS
C A TRUNCATED-NEWTON ALGORITHM (SEE "NEWTON-TYPE MINIMIZATION VIA
C THE LANCZOS ALGORITHM" BY S.G. NASH (TECHNICAL REPORT 378, MATH.
C THE LANCZOS METHOD" BY S.G. NASH (SIAM J. NUMER. ANAL. 21 (1984),
C PP. 770-778).  THIS ALGORITHM FINDS A LOCAL MINIMUM OF F(X).  IT DOES
C NOT ASSUME THAT THE FUNCTION F IS CONVEX (AND SO CANNOT GUARANTEE A
C GLOBAL SOLUTION), BUT DOES ASSUME THAT THE FUNCTION IS BOUNDED BELOW.
C IT CAN SOLVE PROBLEMS HAVING ANY NUMBER OF VARIABLES, BUT IT IS
C ESPECIALLY USEFUL WHEN THE NUMBER OF VARIABLES (N) IS LARGE.
C
C SUBROUTINE PARAMETERS:
C
C IERROR  - (INTEGER) ERROR CODE
C           ( 0 => NORMAL RETURN
C           ( 2 => MORE THAN MAXFUN EVALUATIONS
C           ( 3 => LINE SEARCH FAILED TO FIND LOWER
C               POINT (MAY NOT BE SERIOUS)
C           (-1 => ERROR IN INPUT PARAMETERS
C
C N        - (INTEGER) NUMBER OF VARIABLES
C X        - (REAL*8) VECTOR OF LENGTH AT LEAST N; ON INPUT, AN INITIAL
C           ESTIMATE OF THE SOLUTION; ON OUTPUT, THE COMPUTED SOLUTION.
C G        - (REAL*8) VECTOR OF LENGTH AT LEAST N; ON OUTPUT, THE FINAL
C           VALUE OF THE GRADIENT
C F        - (REAL*8) ON INPUT, A ROUGH ESTIMATE OF THE VALUE OF THE
C           OBJECTIVE FUNCTION AT THE SOLUTION; ON OUTPUT, THE VALUE
C           OF THE OBJECTIVE FUNCTION AT THE SOLUTION
C W        - (REAL*8) WORK VECTOR OF LENGTH AT LEAST 14*N
C LW       - (INTEGER) THE DECLARED DIMENSION OF W
C SFUN     - A USER-SPECIFIED SUBROUTINE THAT COMPUTES THE FUNCTION
C           AND GRADIENT OF THE OBJECTIVE FUNCTION.  IT MUST HAVE
C           THE CALLING SEQUENCE
C             SUBROUTINE SFUN (N, X, F, G)
C             INTEGER          N
C             DOUBLE PRECISION X(N), G(N), F
C LOW, UP  - (REAL*8) VECTORS OF LENGTH AT LEAST N CONTAINING
C           THE LOWER AND UPPER BOUNDS ON THE VARIABLES.  IF
C           THERE ARE NO BOUNDS ON A PARTICULAR VARIABLE, SET
C           THE BOUNDS TO -1.D38 AND 1.D38, RESPECTIVELY.
C IPIVOT   - (INTEGER) WORK VECTOR OF LENGTH AT LEAST N, USED
C           TO RECORD WHICH VARIABLES ARE AT THEIR BOUNDS.
C
C THIS IS AN EASY-TO-USE DRIVER FOR THE MAIN OPTIMIZATION ROUTINE
C LMQNBC.  MORE EXPERIENCED USERS WHO WISH TO CUSTOMIZE PERFORMANCE
C OF THIS ALGORITHM SHOULD CALL LMQBC DIRECTLY.
C
C-----
C THIS ROUTINE SETS UP ALL THE PARAMETERS FOR THE TRUNCATED-NEWTON
C ALGORITHM.  THE PARAMETERS ARE:
C
C ETA      - SEVERITY OF THE LINESEARCH
C MAXFUN   - MAXIMUM ALLOWABLE NUMBER OF FUNCTION EVALUATIONS
C XTOL     - DESIRED ACCURACY FOR THE SOLUTION X*

```

```

C STEPMX - MAXIMUM ALLOWABLE STEP IN THE LINESEARCH
C ACCRCY - ACCURACY OF COMPUTED FUNCTION VALUES
C MSGGLVL - CONTROLS QUANTITY OF PRINTED OUTPUT
C          0 = NONE, 1 = ONE LINE PER MAJOR ITERATION.
C MAXIT - MAXIMUM NUMBER OF INNER ITERATIONS PER STEP
C
      DOUBLE PRECISION  ETA, ACCRCY, XTOL, STEPMX, DSQRT, MCHPR1
      EXTERNAL          SFUN
C
C SET PARAMETERS FOR THE OPTIMIZATION ROUTINE
C
      MAXIT = N/2
      IF (MAXIT .GT. 50) MAXIT = 50
      IF (MAXIT .LE. 0) MAXIT = 1
      MSGGLVL = 1
      MAXFUN = 150*N
      ETA = .25D0
      STEPMX = 1.D1
      ACCRCY = 1.D2*MCHPR1()
      XTOL = DSQRT(ACCRCY)
C
C MINIMIZE FUNCTION
C
      CALL LMQNBC (IERROR, N, X, F, G, W, LW, SFUN, LOW, UP, IPIVOT,
*               MSGGLVL, MAXIT, MAXFUN, ETA, STEPMX, ACCRCY, XTOL)
C
C PRINT RESULTS
C
      IF (IERROR .NE. 0) WRITE(*,800) IERROR
      WRITE(*,810) F
      IF (MSGGLVL .LT. 1) RETURN
      WRITE(*,820)
      NMAX = 10
      IF (N .LT. NMAX) NMAX = N
      WRITE(*,830) (I,X(I),I=1,NMAX)
      RETURN
800  FORMAT(//,' ERROR CODE =', I3)
810  FORMAT(//,' OPTIMAL FUNCTION VALUE = ', 1PD22.15)
820  FORMAT(10X, 'CURRENT SOLUTION IS (AT MOST 10 COMPONENTS)', /,
*       14X, 'I', 11X, 'X(I)')
830  FORMAT(10X, I5, 2X, 1PD22.15)
      END
C
C
      SUBROUTINE LMQN (IFAIL, N, X, F, G, W, LW, SFUN,
*                   MSGGLVL, MAXIT, MAXFUN, ETA, STEPMX, ACCRCY, XTOL)
      IMPLICIT      DOUBLE PRECISION (A-H,O-Z)
      INTEGER       MSGGLVL, N, MAXFUN, IFAIL, LW
      DOUBLE PRECISION  X(N), G(N), W(LW), ETA, XTOL, STEPMX, F, ACCRCY
C
C THIS ROUTINE IS A TRUNCATED-NEWTON METHOD.
C THE TRUNCATED-NEWTON METHOD IS PRECONDITIONED BY A LIMITED-MEMORY
C QUASI-NEWTON METHOD (THIS PRECONDITIONING STRATEGY IS DEVELOPED
C IN THIS ROUTINE) WITH A FURTHER DIAGONAL SCALING (SEE ROUTINE NDIA3).
C FOR FURTHER DETAILS ON THE PARAMETERS, SEE ROUTINE TN.
C
      INTEGER I, ICYCLE, IOLDG, IPK, IYK, LOLDG, LPK, LSR,

```

```

*      LWTEST, LYK, LYR, NFTOTL, NITER, NM1, NUMF, NWHY
DOUBLE PRECISION ABSTOL, ALPHA, DIFNEW, DIFOLD, EPSMCH,
*      EPSRED, FKEEP, FM, FNEW, FOLD, FSTOP, FTEST, GNORM, GSK,
*      GTG, GTPNEW, OLDF, OLDGTP, ONE, PE, PEPS, PNORM, RELTOL,
*      RTEPS, RTLEPS, RTOL, RTOLSQ, SMALL, SPE, TINY,
*      TNYTOL, TOLEPS, XNORM, YKSK, YRSR, ZERO
LOGICAL LRESET, UPD1
C
C THE FOLLOWING IMSL AND STANDARD FUNCTIONS ARE USED
C
DOUBLE PRECISION DABS, DDOT, DSQRT, STEP1, DNRM2
EXTERNAL SFUN
COMMON /SUBSCR/ LGV,LZ1,LZK,LV,LSK,LYK,LDIAGB,LSR,LYR,
*      LODG,LHG,LHYK,LPK,LEMAT,LWTEST
C
C INITIALIZE PARAMETERS AND CONSTANTS
C
IF (MSGLVL .GE. -2) WRITE(*,800)
CALL SETPAR(N)
UPD1 = .TRUE.
IRESET = 0
NFEVAL = 0
NMODIF = 0
NLINCG = 0
FSTOP = F
ZERO = 0.D0
ONE = 1.D0
NM1 = N - 1
C
C WITHIN THIS ROUTINE THE ARRAY W(LOLDG) IS SHARED BY W(LHYR)
C
LHYR = LODG
C
C CHECK PARAMETERS AND SET CONSTANTS
C
CALL CHKUCP(LWTEST,MAXFUN,NWHY,N,ALPHA,EPSMCH,
*      ETA,PEPS,RTEPS,RTOL,RTOLSQ,STEPMX,FTEST,
*      XTOL,XNORM,X,LW,SMALL,TINY,ACCRCY)
IF (NWHY .LT. 0) GO TO 120
CALL SETUCR(SMALL,NFTOTL,NITER,N,F,FNEW,
*      FM,GTG,OLDF,SFUN,G,X)
FOLD = FNEW
IF (MSGLVL .GE. 1) WRITE(*,810) NITER,NFTOTL,NLINCG,FNEW,GTG
C
C CHECK FOR SMALL GRADIENT AT THE STARTING POINT.
C
FTEST = ONE + DABS(FNEW)
IF (GTG .LT. 1.D-4*EPSMCH*FTEST*FTEST) GO TO 90
C
C SET INITIAL VALUES TO OTHER PARAMETERS
C
ICYCLE = NM1
TOLEPS = RTOL + RTEPS
RTLEPS = RTOLSQ + EPSMCH
GNORM = DSQRT(GTG)
DIFNEW = ZERO
EPSRED = 5.0D-2

```

```

      FKEEP = FNEW
C
C SET THE DIAGONAL OF THE APPROXIMATE HESSIAN TO UNITY.
C
      IDIAGB = LDIAGB
      DO 10 I = 1,N
          W(IDIAGB) = ONE
          IDIAGB = IDIAGB + 1
10    CONTINUE
C
C .....START OF MAIN ITERATIVE LOOP.....
C
C COMPUTE THE NEW SEARCH DIRECTION
C
      MODET = MSGVLV - 3
      CALL MODLNP(MODET,W(LPK),W(LGV),W(LZ1),W(LV),
*      W(LDIAGB),W(LEMAT),X,G,W(LZK),
*      N,W,LW,NITER,MAXIT,NFEVAL,NMODIF,
*      NLINCG,UPD1,YKSK,GSK,YRSR,LRESET,SFUN,.FALSE.,IPIVOT,
*      ACCRCY,GTPNEW,GNORM,XNORM)
20    CONTINUE
      CALL DCOPY(N,G,1,W(LOLDG),1)
      PNORM = DNRM2(N,W(LPK),1)
      OLDF = FNEW
      OLDGTP = GTPNEW
C
C PREPARE TO COMPUTE THE STEP LENGTH
C
      PE = PNORM + EPSMCH
C
C COMPUTE THE ABSOLUTE AND RELATIVE TOLERANCES FOR THE LINEAR SEARCH
C
      RELTOL = RSTEPS*(XNORM + ONE)/PE
      ABSTOL = - EPSMCH*FTEST/(OLDGTP - EPSMCH)
C
C COMPUTE THE SMALLEST ALLOWABLE SPACING BETWEEN POINTS IN
C THE LINEAR SEARCH
C
      TNYTOL = EPSMCH*(XNORM + ONE)/PE
      SPE = STEPMX/PE
C
C SET THE INITIAL STEP LENGTH.
C
      ALPHA = STEP1(FNEW,FM,OLDGTP,SPE)
C
C PERFORM THE LINEAR SEARCH
C
      CALL LINDER(N,SFUN,SMALL,EPSMCH,RELTOL,ABSTOL,TNYTOL,
*      ETA,ZERO,SPE,W(LPK),OLDGTP,X,FNEW,ALPHA,G,NUMF,
*      NWHY,W,LW)
C
      FOLD = FNEW
      NITER = NITER + 1
      NFTOTL = NFTOTL + NUMF
      GTG = DDOT(N,G,1,G,1)
      IF (MSGVLV .GE. 1) WRITE(*,810) NITER,NFTOTL,NLINCG,FNEW,GTG
      WRITE(8,812) NITER,FNEW,GTG

```

```

        IF (NWHY .LT. 0) GO TO 120
        IF (NWHY .EQ. 0 .OR. NWHY .EQ. 2) GO TO 30
C
C THE LINEAR SEARCH HAS FAILED TO FIND A LOWER POINT
C
        NWHY = 3
        GO TO 100
30      IF (NWHY .LE. 1) GO TO 40
        CALL SFUN(N,X,FNEW,G)
        NFTOTL = NFTOTL + 1
C
C TERMINATE IF MORE THAN MAXFUN EVALUTATIONS HAVE BEEN MADE
C
40      NWHY = 2
        IF (NFTOTL .GT. MAXFUN) GO TO 110
        NWHY = 0
C
C SET UP PARAMETERS USED IN CONVERGENCE AND RESETTING TESTS
C
        DIFOLD = DIFNEW
        DIFNEW = OLDF - FNEW
C
C IF THIS IS THE FIRST ITERATION OF A NEW CYCLE, COMPUTE THE
C PERCENTAGE REDUCTION FACTOR FOR THE RESETTING TEST.
C
        IF (ICYCLE .NE. 1) GO TO 50
        IF (DIFNEW .GT. 2.0D0 *DIFOLD) EPSRED = EPSRED + EPSRED
        IF (DIFNEW .LT. 5.0D-1*DIFOLD) EPSRED = 5.0D-1*EPSRED
50      CONTINUE
        GNORM = DSQRT(GTG)
        FTEST = ONE + DABS(FNEW)
        XNORM = DNRM2(N,X,1)
C
C TEST FOR CONVERGENCE
C
        IF ((ALPHA*PNORM .LT. TOLEPS*(ONE + XNORM)
*       .AND. DABS(DIFNEW) .LT. RTLEPS*FTEST
*       .AND. GTG .LT. PEPS*FTEST*FTEST)
*       .OR. GTG .LT. 1.D-4*ACCRCY*FTEST*FTEST) GO TO 90
C
C COMPUTE THE CHANGE IN THE ITERATES AND THE CORRESPONDING CHANGE
C IN THE GRADIENTS
C
        ISK = LSK
        IPK = LPK
        IYK = LYK
        IOLDG = IOLDG
        DO 60 I = 1,N
            W(IYK) = G(I) - W(IOLDG)
            W(ISK) = ALPHA*W(IPK)
            IPK = IPK + 1
            ISK = ISK + 1
            IYK = IYK + 1
            IOLDG = IOLDG + 1
60      CONTINUE
C
C SET UP PARAMETERS USED IN UPDATING THE DIRECTION OF SEARCH.

```

```

C
  YKSK = DDOT(N,W(LYK),1,W(LSK),1)
  LRESET = .FALSE.
  IF (ICYCLE .EQ. NM1 .OR. DIFNEW .LT.
*     EPSRED*(FKEEP-FNEW)) LRESET = .TRUE.
  IF (LRESET) GO TO 70
  YRSR = DDOT(N,W(LYR),1,W(LSR),1)
  IF (YRSR .LE. ZERO) LRESET = .TRUE.
70  CONTINUE
  UPD1 = .FALSE.

C
C   COMPUTE THE NEW SEARCH DIRECTION
C
  MODET = MSGLVL - 3
  CALL MODLNP(MODET,W(LPK),W(LGV),W(LZ1),W(LV),
*   W(LDIAGB),W(LEMAT),X,G,W(LZK),
*   N,W,LW,NITER,MAXIT,NFEVAL,NMODIF,
*   NLINCG,UPD1,YKSK,GSK,YRSR,LRESET,SFUN,.FALSE.,IPIVOT,
*   ACCRCY,GTPNEW,GNORM,XNORM)
  IF (LRESET) GO TO 80

C
C   STORE THE ACCUMULATED CHANGE IN THE POINT AND GRADIENT AS AN
C   "AVERAGE" DIRECTION FOR PRECONDITIONING.
C
  CALL DXPY(N,W(LSK),1,W(LSR),1)
  CALL DXPY(N,W(LYK),1,W(LYR),1)
  ICYCLE = ICYCLE + 1
  GOTO 20

C
C RESET
C
80  IRESET = IRESET + 1
C
C INITIALIZE THE SUM OF ALL THE CHANGES IN X.
C
  CALL DCOPY(N,W(LSK),1,W(LSR),1)
  CALL DCOPY(N,W(LYK),1,W(LYR),1)
  FKEEP = FNEW
  ICYCLE = 1
  GO TO 20

C
C .....END OF MAIN ITERATION.....
C
90  IFAIL = 0
  F = FNEW
  RETURN
100 OLDF = FNEW
C
C LOCAL SEARCH HERE COULD BE INSTALLED HERE
C
110  F = OLDF
C
C SET IFAIL
C
120  IFAIL = NWHY
  RETURN
800  FORMAT(// ' NIT   NF   CG', 9X, 'F', 21X, 'GTG',//)

```

```

810  FORMAT(' ',I3,1X,I4,1X,I4,1X,1PD22.15,2X,1PD15.8)
812  FORMAT(3X,I4,2X,1PD22.15,2X,1PD15.8)
      END
C
C
      SUBROUTINE LMQNBC (IFAIL, N, X, F, G, W, LW, SFUN, LOW, UP,
*   IPIVOT, MSGLVL, MAXIT, MAXFUN, ETA, STEPMX, ACCRCY, XTOL)
      IMPLICIT      DOUBLE PRECISION (A-H,O-Z)
      INTEGER      MSGLVL,N,MAXFUN,IFAIL,LW
      INTEGER      IPIVOT(N)
      DOUBLE PRECISION  ETA,XTOL,STPEMX,F,ACCRCY
      DOUBLE PRECISION  X(N),G(N),W(LW),LOW(N),UP(N)
C
C THIS ROUTINE IS A BOUNDS-CONSTRAINED TRUNCATED-NEWTON METHOD.
C THE TRUNCATED-NEWTON METHOD IS PRECONDITIONED BY A LIMITED-MEMORY
C QUASI-NEWTON METHOD (THIS PRECONDITIONING STRATEGY IS DEVELOPED
C IN THIS ROUTINE) WITH A FURTHER DIAGONAL SCALING (SEE ROUTINE NDIA3).
C FOR FURTHER DETAILS ON THE PARAMETERS, SEE ROUTINE TNBC.
C
      INTEGER I, ICYCLE, IOLDG, IPK, IYK, LOLDG, LPK, LSR,
*   LWTEST, LYK, LYR, NFTOTL, NITER, NM1, NUMF, NWHY
      DOUBLE PRECISION ABSTOL, ALPHA, DIFNEW, DIFOLD, EPSMCH, EPSRED,
*   FKEEP, FLAST, FM, FNEW, FOLD, FSTOP, FTEST, GNORM, GSK,
*   GTG, GTPNEW, OLDF, OLDGTP, ONE, PE, PEPS, PNORM, RELTOL,
*   RTEPS, RTLEPS, RTOL, RTOLSQ, SMALL, SPE, TINY,
*   TNYTOL, TOLEPS, XNORM, YKSK, YRSR, ZERO
      LOGICAL CONV, LRESET, UPD1, NEWCON
C
C THE FOLLOWING STANDARD FUNCTIONS AND SYSTEM FUNCTIONS ARE USED
C
      DOUBLE PRECISION DABS, DDOT, DNRM2, DSQRT, STEP1
      EXTERNAL SFUN
      COMMON/SUBSCR/ LGV, LZ1, LZK, LV, LSK, LYK, LDIAGB, LSR, LYR,
*   LOLDG, LHG, LHYK, LPK, LEMAT, LWTEST
C
C CHECK THAT INITIAL X IS FEASIBLE AND THAT THE BOUNDS ARE CONSISTENT
C
      CALL CRASH(N,X,IPIVOT,LOW,UP,IER)
      IF (IER .NE. 0) WRITE(*,800)
      IF (IER .NE. 0) RETURN
      IF (MSGLVL .GE. 1) WRITE(*,810)
C
C INITIALIZE VARIABLES
C
      CALL SETPAR(N)
      UPD1 = .TRUE.
      IRESET = 0
      NFEVAL = 0
      NMODIF = 0
      NLINCG = 0
      FSTOP = F
      CONV = .FALSE.
      ZERO = 0.D0
      ONE = 1.D0
      NM1 = N - 1
C
C WITHIN THIS ROUTINE THE ARRAY W(LOLDG) IS SHARED BY W(LHYR)

```

```

C
    LHYR = LOLDG
C
C CHECK PARAMETERS AND SET CONSTANTS
C
    CALL CHKUCP(LWTEST,MAXFUN,NWHY,N,ALPHA,EPSMCH,
*           ETA,PEPS,RTEPS,RTOL,RTOLSQ,STEPMX,FTEST,
*           XTOL,XNORM,X,LW,SMALL,TINY,ACCRCY)
    IF (NWHY .LT. 0) GO TO 160
    CALL SETUCR(SMALL,NFTOTL,NITER,N,F,FNEW,
*           FM,GTG,OLDF,SFUN,G,X)
    FOLD = FNEW
    FLAST = FNEW
C
C TEST THE LAGRANGE MULTIPLIERS TO SEE IF THEY ARE NON-NEGATIVE.
C BECAUSE THE CONSTRAINTS ARE ONLY LOWER BOUNDS, THE COMPONENTS
C OF THE GRADIENT CORRESPONDING TO THE ACTIVE CONSTRAINTS ARE THE
C LAGRANGE MULTIPLIERS. AFTERWORDS, THE PROJECTED GRADIENT IS FORMED.
C
    DO 10 I = 1,N
        IF (IPIVOT(I) .EQ. 2) GO TO 10
        IF (-IPIVOT(I)*G(I) .GE. 0.D0) GO TO 10
        IPIVOT(I) = 0
10    CONTINUE
        CALL ZTIME(N,G,IPIVOT)
        GTG = DDOT(N,G,1,G,1)
        IF (MSGLVL .GE. 1)
*           CALL MONIT(N,X,FNEW,G,NITER,NFTOTL,NFEVAL,LRESET,IPIVOT)
C
C CHECK IF THE INITIAL POINT IS A LOCAL MINIMUM.
C
    FTEST = ONE + DABS(FNEW)
    IF (GTG .LT. 1.D-4*EPSMCH*FTEST*FTEST) GO TO 130
C
C SET INITIAL VALUES TO OTHER PARAMETERS
C
    ICYCLE = NM1
    TOLEPS = RTOL + RTEPS
    RTLEPS = RTOLSQ + EPSMCH
    GNORM = DSQRT(GTG)
    DIFNEW = ZERO
    EPSRED = 5.0D-2
    FKEEP = FNEW
C
C SET THE DIAGONAL OF THE APPROXIMATE HESSIAN TO UNITY.
C
    IDIAGB = LDIAGB
    DO 15 I = 1,N
        W(IDIAGB) = ONE
        IDIAGB = IDIAGB + 1
15    CONTINUE
C
C .....START OF MAIN ITERATIVE LOOP.....
C
C COMPUTE THE NEW SEARCH DIRECTION
C
    MODET = MSGLVL - 3

```

```

        CALL MODLNP(MODET,W(LPK),W(LGV),W(LZ1),W(LV),
*         W(LDIAGB),W(LEMAT),X,G,W(LZK),
*         N,W,LW,NITER,MAXIT,NFEVAL,NMODIF,
*         NLINCG,UPD1,YKSK,GSK,YRSR,LRESET,SFUN,.TRUE.,IPIVOT,
*         ACCRCY,GTPNEW,GNORM,XNORM)
20    CONTINUE
        CALL DCOPY(N,G,1,W(LOLDG),1)
        PNORM = DNRM2(N,W(LPK),1)
        OLDF = FNEW
        OLDGTP = GTPNEW
C
C PREPARE TO COMPUTE THE STEP LENGTH
C
        PE = PNORM + EPSMCH
C
C COMPUTE THE ABSOLUTE AND RELATIVE TOLERANCES FOR THE LINEAR SEARCH
C
        RELTOL = RSTEPS*(XNORM + ONE)/PE
        ABSTOL = - EPSMCH*FTEST/(OLDGTP - EPSMCH)
C
C COMPUTE THE SMALLEST ALLOWABLE SPACING BETWEEN POINTS IN
C THE LINEAR SEARCH
C
        TNYTOL = EPSMCH*(XNORM + ONE)/PE
        CALL STPMAX(STEPMX,PE,SPE,N,X,W(LPK),IPIVOT,LOW,UP)
C
C SET THE INITIAL STEP LENGTH.
C
        ALPHA = STEP1(FNEW,FM,OLDGTP,SPE)
C
C PERFORM THE LINEAR SEARCH
C
        CALL LINDER(N,SFUN,SMALL,EPSMCH,RELTOL,ABSTOL,TNYTOL,
*         ETA,ZERO,SPE,W(LPK),OLDGTP,X,FNEW,ALPHA,G,NUMF,
*         NWHY,W,LW)
        NEWCON = .FALSE.
        IF (DABS(ALPHA-SPE) .GT. 1.D1*EPSMCH) GO TO 30
        NEWCON = .TRUE.
        NWHY = 0
        CALL MODZ(N,X,W(LPK),IPIVOT,EPSMCH,LOW,UP,FLAST,FNEW)
        FLAST = FNEW
C
30    IF (MSGVLV .GE. 3) WRITE(*,820) ALPHA,PNORM
        FOLD = FNEW
        NITER = NITER + 1
        NFTOTL = NFTOTL + NUMF
C
C IF REQUIRED, PRINT THE DETAILS OF THIS ITERATION
C
        IF (MSGVLV .GE. 1)
*         CALL MONIT(N,X,FNEW,G,NITER,NFTOTL,NFEVAL,LRESET,IPIVOT)
        IF (NWHY .LT. 0) GO TO 160
        IF (NWHY .EQ. 0 .OR. NWHY .EQ. 2) GO TO 40
C
C THE LINEAR SEARCH HAS FAILED TO FIND A LOWER POINT
C
        NWHY = 3

```

```

        GO TO 140
40      IF (NWHY .LE. 1) GO TO 50
        CALL SFUN(N,X,FNEW,G)
        NFTOTL = NFTOTL + 1
C
C TERMINATE IF MORE THAN MAXFUN EVALUATIONS HAVE BEEN MADE
C
50      NWHY = 2
        IF (NFTOTL .GT. MAXFUN) GO TO 150
        NWHY = 0
C
C SET UP PARAMETERS USED IN CONVERGENCE AND RESETTING TESTS
C
        DIFOLD = DIFNEW
        DIFNEW = OLDF - FNEW
C
C IF THIS IS THE FIRST ITERATION OF A NEW CYCLE, COMPUTE THE
C PERCENTAGE REDUCTION FACTOR FOR THE RESETTING TEST.
C
        IF (ICYCLE .NE. 1) GO TO 60
        IF (DIFNEW .GT. 2.D0*DIFOLD) EPSRED = EPSRED + EPSRED
        IF (DIFNEW .LT. 5.0D-1*DIFOLD) EPSRED = 5.0D-1*EPSRED
60      CALL DCOPY(N,G,1,W(LGV),1)
        CALL ZTIME(N,W(LGV),IPIVOT)
        GTG = DDOT(N,W(LGV),1,W(LGV),1)
        GNORM = DSQRT(GTG)
        FTEST = ONE + DABS(FNEW)
        XNORM = DNRM2(N,X,1)
C
C TEST FOR CONVERGENCE
C
        CALL CNVTST(CONV,ALPHA,PNORM,TOLEPS,XNORM,DIFNEW,RTLEPS,
*          FTEST,GTG,PEPS,EPSMCH,GTPNEW,FNEW,FLAST,G,IPIVOT,N,ACCRCY)
        IF (CONV) GO TO 130
        CALL ZTIME(N,G,IPIVOT)
C
C COMPUTE THE CHANGE IN THE ITERATES AND THE CORRESPONDING CHANGE
C IN THE GRADIENTS
C
        IF (NEWCON) GO TO 90
        ISK = LSK
        IPK = LPK
        IYK = LYK
        IOLDG = IOLDG
        DO 70 I = 1,N
            W(IYK) = G(I) - W(IOLDG)
            W(ISK) = ALPHA*W(IPK)
            IPK = IPK + 1
            ISK = ISK + 1
            IYK = IYK + 1
            IOLDG = IOLDG + 1
70      CONTINUE
C
C SET UP PARAMETERS USED IN UPDATING THE PRECONDITIONING STRATEGY.
C
        YKSK = DDOT(N,W(LYK),1,W(LSK),1)
        LRESET = .FALSE.

```

```

      IF (ICYCLE .EQ. NM1 .OR. DIFNEW .LT.
*      EPSRED*(FKEEP-FNEW)) LRESET = .TRUE.
      IF (LRESET) GO TO 80
      YRSR = DDOT(N,W(LYR),1,W(LSR),1)
      IF (YRSR .LE. ZERO) LRESET = .TRUE.
80    CONTINUE
      UPD1 = .FALSE.

C
C      COMPUTE THE NEW SEARCH DIRECTION
C
90    IF (UPD1 .AND. MSGGLVL .GE. 3) WRITE(*,830)
      IF (NEWCON .AND. MSGGLVL .GE. 3) WRITE(*,840)
      MODET = MSGGLVL - 3
      CALL MODLNP(MODET,W(LPK),W(LGV),W(LZ1),W(LV),
*      W(LDIAGB),W(LEMAT),X,G,W(LZK),
*      N,W,LW,NITER,MAXIT,NFEVAL,NMODIF,
*      NLINCG,UPD1,YKSK,GSK,YRSR,LRESET,SFUN,.TRUE.,IPIVOT,
*      ACCRCY,GTPNEW,GNORM,XNORM)
      IF (NEWCON) GO TO 20
      IF (LRESET) GO TO 110

C
C COMPUTE THE ACCUMULATED STEP AND ITS CORRESPONDING
C GRADIENT DIFFERENCE.
C
      CALL DXPY(N,W(LSK),1,W(LSR),1)
      CALL DXPY(N,W(LYK),1,W(LYR),1)
      ICYCLE = ICYCLE + 1
      GOTO 20

C
C RESET
C
110   IRESET = IRESET + 1
C
C INITIALIZE THE SUM OF ALL THE CHANGES IN X.
C
      CALL DCOPY(N,W(LSK),1,W(LSR),1)
      CALL DCOPY(N,W(LYK),1,W(LYR),1)
      FKEEP = FNEW
      ICYCLE = 1
      GO TO 20

C
C .....END OF MAIN ITERATION.....
C
130   IFAIL = 0
      F = FNEW
      RETURN

140   OLDF = FNEW
C
C LOCAL SEARCH COULD BE INSTALLED HERE
C
150   F = OLDF
      IF (MSGGLVL .GE. 1) CALL MONIT(N,X,
*      F,G,NITER,NFTOTL,NFEVAL,IRESET,IPIVOT)

C
C SET IFAIL
C
160   IFAIL = NWHY

```

```

      RETURN
800  FORMAT(' THERE IS NO FEASIBLE POINT; TERMINATING ALGORITHM')
810  FORMAT(// ' NIT  NF  CG', 9X, 'F', 21X, 'GTG', //)
820  FORMAT('          LINESEARCH RESULTS:  ALPHA,PNORM', 2(1PD12.4))
830  FORMAT(' UPD1 IS TRUE - TRIVIAL PRECONDITIONING')
840  FORMAT(' NEWCON IS TRUE - CONSTRAINT ADDED IN LINESEARCH')
      END
C
C
      SUBROUTINE MONIT(N,X,F,G,NITER,NFTOTL,NFEVAL,IRESET,IPIVOT)
C
C PRINT RESULTS OF CURRENT ITERATION
C
      IMPLICIT          DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION X(N),F,G(N),GTG
      INTEGER           IPIVOT(N)
C
      OPEN(UNIT=8,FILE='TEST1.OUT',STATUS='NEW')
      GTG = 0.D0
      DO 10 I = 1,N
          IF (IPIVOT(I) .NE. 0) GO TO 10
          GTG = GTG + G(I)*G(I)
10     CONTINUE
      WRITE(*,800) NITER,NFTOTL,NFEVAL,F,GTG
      WRITE(8,812) NITER,F,GTG
      RETURN
812  FORMAT(3X,I4,2X,1PD22.15,2X,1PD15.8)
800  FORMAT(' ',I4,1X,I4,1X,I4,1X,1PD22.15,2X,1PD15.8)
      END
C
C
      SUBROUTINE ZTIME(N,X,IPIVOT)
      IMPLICIT          DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION X(N)
      INTEGER           IPIVOT(N)
C
C THIS ROUTINE MULTIPLIES THE VECTOR X BY THE CONSTRAINT MATRIX Z
C
      DO 10 I = 1,N
          IF (IPIVOT(I) .NE. 0) X(I) = 0.D0
10     CONTINUE
      RETURN
      END
C
C
      SUBROUTINE STPMAX(STEPMX,PE,SPE,N,X,P,IPIVOT,LOW,UP)
      IMPLICIT          DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION LOW(N),UP(N),X(N),P(N),STEPMX,PE,SPE,T
      INTEGER           IPIVOT(N)
C
C COMPUTE THE MAXIMUM ALLOWABLE STEP LENGTH
C
      SPE = STEPMX / PE
C SPE IS THE STANDARD (UNCONSTRAINED) MAX STEP
      DO 10 I = 1,N
          IF (IPIVOT(I) .NE. 0) GO TO 10
          IF (P(I) .EQ. 0.D0) GO TO 10

```

```

        IF (P(I) .GT. 0.D0) GO TO 5
        T = LOW(I) - X(I)
        IF (T .GT. SPE*P(I)) SPE = T / P(I)
        GO TO 10
5       T = UP(I) - X(I)
        IF (T .LT. SPE*P(I)) SPE = T / P(I)
10      CONTINUE
        RETURN
        END

C
C
SUBROUTINE MODZ(N,X,P,IPIVOT,EPSMCH,LOW,UP,FLAST,FNEW)
IMPLICIT      DOUBLE PRECISION (A-H,O-Z)
DOUBLE PRECISION X(N), P(N), EPSMCH, DABS, TOL, LOW(N), UP(N),
*             FLAST, FNEW
INTEGER      IPIVOT(N)

C
C UPDATE THE CONSTRAINT MATRIX IF A NEW CONSTRAINT IS ENCOUNTERED
C
DO 10 I = 1,N
    IF (IPIVOT(I) .NE. 0) GO TO 10
    IF (P(I) .EQ. 0.D0) GO TO 10
    IF (P(I) .GT. 0.D0) GO TO 5
    TOL = 1.D1 * EPSMCH * (DABS(LOW(I)) + 1.D0)
    IF (X(I)-LOW(I) .GT. TOL) GO TO 10
    FLAST = FNEW
    IPIVOT(I) = -1
    X(I) = LOW(I)
    GO TO 10
5     TOL = 1.D1 * EPSMCH * (DABS(UP(I)) + 1.D0)
    IF (UP(I)-X(I) .GT. TOL) GO TO 10
    FLAST = FNEW
    IPIVOT(I) = 1
    X(I) = UP(I)
10    CONTINUE
    RETURN
    END

C
C
SUBROUTINE CNVTST(CONV,ALPHA,PNORM,TOLEPS,XNORM,DIFNEW,RTLEPS,
*             FTEST,GTG,PEPS,EPSMCH,GTPNEW,FNEW,FLAST,G,IPIVOT,N,ACCRCY)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
LOGICAL CONV,LTEST
INTEGER IPIVOT(N)
DOUBLE PRECISION G(N), ALPHA, PNORM, TOLEPS, XNORM, DIFNEW,
*             RTLEPS, FTEST, GTG, PEPS, EPSMCH, GTPNEW, FNEW, FLAST, ONE,
*             CMAX, T, ACCRCY

C
C TEST FOR CONVERGENCE
C
IMAX = 0
CMAX = 0.D0
LTEST = FLAST - FNEW .LE. -5.D-1*GTPNEW
DO 10 I = 1,N
    IF (IPIVOT(I) .EQ. 0 .OR. IPIVOT(I) .EQ. 2) GO TO 10
    T = -IPIVOT(I)*G(I)
    IF (T .GE. 0.D0) GO TO 10

```

```

        CONV = .FALSE.
        IF (LTEST) GO TO 10
        IF (CMAX .LE. T) GO TO 10
        CMAX = T
        IMAX = I
10     CONTINUE
        IF (IMAX .EQ. 0) GO TO 15
        IPIVOT(IMAX) = 0
        FLAST = FNEW
        RETURN
15     CONTINUE
        CONV = .FALSE.
        ONE = 1.D0
        IF ((ALPHA*PNORM .GE. TOLEPS*(ONE + XNORM)
*         .OR. DABS(DIFNEW) .GE. RTLEPS*FTEST
*         .OR. GTG .GE. PEPS*FTEST*FTEST)
*         .AND. GTG .GE. 1.D-4*ACCRCY*FTEST*FTEST) RETURN
        CONV = .TRUE.
C
C FOR DETAILS, SEE GILL, MURRAY, AND WRIGHT (1981, P. 308) AND
C FLETCHER (1981, P. 116). THE MULTIPLIER TESTS (HERE, TESTING
C THE SIGN OF THE COMPONENTS OF THE GRADIENT) MAY STILL NEED TO
C MODIFIED TO INCORPORATE TOLERANCES FOR ZERO.
C
        RETURN
        END
C
C
        SUBROUTINE CRASH(N,X,IPIVOT,LOW,UP,IER)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        DOUBLE PRECISION X(N),LOW(N),UP(N)
        INTEGER IPIVOT(N)
C
C THIS INITIALIZES THE CONSTRAINT INFORMATION, AND ENSURES THAT THE
C INITIAL POINT SATISFIES LOW <= X <= UP.
C THE CONSTRAINTS ARE CHECKED FOR CONSISTENCY.
C
        IER = 0
        DO 30 I = 1,N
            IF (X(I) .LT. LOW(I)) X(I) = LOW(I)
            IF (X(I) .GT. UP(I)) X(I) = UP(I)
            IPIVOT(I) = 0
            IF (X(I) .EQ. LOW(I)) IPIVOT(I) = -1
            IF (X(I) .EQ. UP(I)) IPIVOT(I) = 1
            IF (UP(I) .EQ. LOW(I)) IPIVOT(I) = 2
            IF (LOW(I) .GT. UP(I)) IER = -I
30     CONTINUE
        RETURN
        END
C
C THE VECTORS SK AND YK, ALTHOUGH NOT IN THE CALL,
C ARE USED (VIA THEIR POSITION IN W) BY THE ROUTINE MSOLVE.
C
        SUBROUTINE MODLNP(MODET,ZSOL,GV,R,V,DIAGB,EMAT,
*         X,G,ZK,N,W,LW,NITER,MAXIT,NFEVAL,NMODIF,NLINCG,
*         UPD1,YKSK,GSK,YRSR,LRESET,SFUN,BOUNDS,IPIVOT,ACCRCY,
*         GTP,GNORM,XNORM)

```

```

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER MODET,N,NITER,IPIVOT(1)
      DOUBLE PRECISION ZSOL(N),G(N),GV(N),R(N),V(N),DIAGB(N),W(LW)
      DOUBLE PRECISION EMAT(N),ZK(N),X(N),ACCRCY
      DOUBLE PRECISION ALPHA,BETA,DELTA,GSK,GTP,PR,
*      QOLD,QNEW,QTEST,RHSNRM,RNORM,RZ,RZOLD,TOL,VGW,YKSK,YRSR
      DOUBLE PRECISION GNORM,XNORM
      DOUBLE PRECISION DDOT,DNRM2
      LOGICAL FIRST,UPD1,LRESET,BOUNDS
      EXTERNAL SFUN
C
C THIS ROUTINE PERFORMS A PRECONDITIONED CONJUGATE-GRADIENT
C ITERATION IN ORDER TO SOLVE THE NEWTON EQUATIONS FOR A SEARCH
C DIRECTION FOR A TRUNCATED-NEWTON ALGORITHM.  WHEN THE VALUE OF THE
C QUADRATIC MODEL IS SUFFICIENTLY REDUCED,
C THE ITERATION IS TERMINATED.
C
C PARAMETERS
C
C MODET      - INTEGER WHICH CONTROLS AMOUNT OF OUTPUT
C ZSOL      - COMPUTED SEARCH DIRECTION
C G         - CURRENT GRADIENT
C GV,GZ1,V  - SCRATCH VECTORS
C R         - RESIDUAL
C DIAGB,EMAT - DIAGONAL PRECONDITONING MATRIX
C NITER     - NONLINEAR ITERATION #
C FEVAL     - VALUE OF QUADRATIC FUNCTION
C
C *****
C INITIALIZATION
C *****
C
C GENERAL INITIALIZATION
C
      IF (MODET .GT. 0) WRITE(*,800)
      IF (MAXIT .EQ. 0) RETURN
      FIRST = .TRUE.
      RHSNRM = GNORM
      TOL = 1.D-12
      QOLD = 0.D0
C
C INITIALIZATION FOR PRECONDITIONED CONJUGATE-GRADIENT ALGORITHM
C
      CALL INITPC(DIAGB,EMAT,N,W,LW,MODET,
*              UPD1,YKSK,GSK,YRSR,LRESET)
      DO 10 I = 1,N
         R(I) = -G(I)
         V(I) = 0.D0
         ZSOL(I) = 0.D0
10    CONTINUE
C
C *****
C MAIN ITERATION
C *****
C
      DO 30 K = 1,MAXIT
         NLINCG = NLINCG + 1

```

```

        IF (MODET .GT. 1) WRITE(*,810) K
C
C CG ITERATION TO SOLVE SYSTEM OF EQUATIONS
C
        IF (BOUNDS) CALL ZTIME(N,R,IPIVOT)
        CALL MSOLVE(R,ZK,N,W,LW,UPD1,YKSK,GSK,
*           YRSR,LRESET,FIRST)
        IF (BOUNDS) CALL ZTIME(N,ZK,IPIVOT)
        RZ = DDOT(N,R,1,ZK,1)
        IF (RZ/RHSNRM .LT. TOL) GO TO 80
        IF (K .EQ. 1) BETA = 0.D0
        IF (K .GT. 1) BETA = RZ/RZOLD
        DO 20 I = 1,N
            V(I) = ZK(I) + BETA*V(I)
20    CONTINUE
        IF (BOUNDS) CALL ZTIME(N,V,IPIVOT)
        CALL GTIMS(V,GV,N,X,G,W,LW,SFUN,FIRST,DELTA,ACCRCY,XNORM)
        IF (BOUNDS) CALL ZTIME(N,GV,IPIVOT)
        NFEVAL = NFEVAL + 1
        VGV = DDOT(N,V,1,GV,1)
        IF (VGV/RHSNRM .LT. TOL) GO TO 50
        CALL NDIA3(N,EMAT,V,GV,R,VGV,MODET)
C
C COMPUTE LINEAR STEP LENGTH
C
        ALPHA = RZ / VGV
        IF (MODET .GE. 1) WRITE(*,820) ALPHA
C
C COMPUTE CURRENT SOLUTION AND RELATED VECTORS
C
        CALL DAXPY(N,ALPHA,V,1,ZSOL,1)
        CALL DAXPY(N,-ALPHA,GV,1,R,1)
C
C TEST FOR CONVERGENCE
C
        GTP = DDOT(N,ZSOL,1,G,1)
        PR = DDOT(N,R,1,ZSOL,1)
        QNEW = 5.D-1 * (GTP + PR)
        QTEST = K * (1.D0 - QOLD/QNEW)
        IF (QTEST .LT. 0.D0) GO TO 70
        QOLD = QNEW
        IF (QTEST .LE. 5.D-1) GO TO 70
C
C PERFORM CAUTIONARY TEST
C
        IF (GTP .GT. 0) GO TO 40
        RZOLD = RZ
30    CONTINUE
C
C TERMINATE ALGORITHM
C
        K = K-1
        GO TO 70
C
C TRUNCATE ALGORITHM IN CASE OF AN EMERGENCY
C
40    IF (MODET .GE. -1) WRITE(*,830) K

```

```

CALL DAXPY(N,-ALPHA,V,1,ZSOL,1)
GTP = DDOT(N,ZSOL,1,G,1)
GO TO 90
50 CONTINUE
IF (MODET .GT. -2) WRITE(*,840)
60 IF (K .GT. 1) GO TO 70
CALL MSOLVE(G,ZSOL,N,W,LW,UPD1,YKSK,GSK,YRSR,LRESET,FIRST)
CALL NEGVEC(N,ZSOL)
IF (BOUNDS) CALL ZTIME(N,ZSOL,IPIVOT)
GTP = DDOT(N,ZSOL,1,G,1)
70 CONTINUE
IF (MODET .GE. -1) WRITE(*,850) K,RNORM
GO TO 90
80 CONTINUE
IF (MODET .GE. -1) WRITE(*,860)
IF (K .GT. 1) GO TO 70
CALL DCOPY(N,G,1,ZSOL,1)
CALL NEGVEC(N,ZSOL)
IF (BOUNDS) CALL ZTIME(N,ZSOL,IPIVOT)
GTP = DDOT(N,ZSOL,1,G,1)
GO TO 70
C
C STORE (OR RESTORE) DIAGONAL PRECONDITIONING
C
90 CONTINUE
CALL DCOPY(N,EMAT,1,DIAGB,1)
RETURN
800 FORMAT(' ',//,' ENTERING MODLNP')
810 FORMAT(' ',//,' ### ITERATION ',I2,' ###')
820 FORMAT(' ALPHA',1PD16.8)
830 FORMAT(' G(T)Z POSITIVE AT ITERATION ',I2,
*      ' - TRUNCATING METHOD',/)
840 FORMAT(' ',10X,'HESSIAN NOT POSITIVE-DEFINITE')
850 FORMAT(' ',/,8X,'MODLAN TRUNCATED AFTER ',I3,' ITERATIONS',
*      ' RNORM = ',1PD14.6)
860 FORMAT(' PRECONDITIONING NOT POSITIVE-DEFINITE')
END
C
C
SUBROUTINE NDIA3(N,E,V,GV,R,VGV,MODET)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DOUBLE PRECISION E(N),V(N),GV(N),R(N),VGV,VR,DDOT
C
C UPDATE THE PRECONDITIOING MATRIX BASED ON A DIAGONAL VERSION
C OF THE BFGS QUASI-NEWTON UPDATE.
C
VR = DDOT(N,V,1,R,1)
DO 10 I = 1,N
E(I) = E(I) - R(I)*R(I)/VR + GV(I)*GV(I)/VGV
IF (E(I) .GT. 1.D-6) GO TO 10
IF (MODET .GT. -2) WRITE(*,800) E(I)
E(I) = 1.D0
10 CONTINUE
RETURN
800 FORMAT(' *** EMAT NEGATIVE: ',1PD16.8)
END
C

```

```

C     SERVICE ROUTINES FOR OPTIMIZATION
C
C     SUBROUTINE NEGVEC(N,V)
C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C     INTEGER N
C     DOUBLE PRECISION V(N)
C
C     NEGATIVE OF THE VECTOR V
C
C     INTEGER I
C     DO 10 I = 1,N
C         V(I) = -V(I)
10    CONTINUE
C     RETURN
C     END
C
C
C     SUBROUTINE LSOUT(ILOC, ITEST, XMIN, FMIN, GMIN, XW, FW, GW, U, A,
*     B, TOL, EPS, SCXBD, XLAMDA)
C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C     DOUBLE PRECISION XMIN, FMIN, GMIN, XW, FW, GW, U, A, B,
*     TOL, EPS, SCXBD, XLAMDA
C
C     ERROR PRINTOUTS FOR GETPTC
C
C     DOUBLE PRECISION YA, YB, YBND, YW, YU
C     YU = XMIN + U
C     YA = A + XMIN
C     YB = B + XMIN
C     YW = XW + XMIN
C     YBND = SCXBD + XMIN
C     WRITE(*,800)
C     WRITE(*,810) TOL, EPS
C     WRITE(*,820) YA, YB
C     WRITE(*,830) YBND
C     WRITE(*,840) YW, FW, GW
C     WRITE(*,850) XMIN, FMIN, GMIN
C     WRITE(*,860) YU
C     WRITE(*,870) ILOC, ITEST
C     RETURN
800    FORMAT('///' OUTPUT FROM LINEAR SEARCH')
810    FORMAT(' TOL AND EPS'/2D25.14)
820    FORMAT(' CURRENT UPPER AND LOWER BOUNDS'/2D25.14)
830    FORMAT(' STRICT UPPER BOUND'/D25.14)
840    FORMAT(' XW, FW, GW'/3D25.14)
850    FORMAT(' XMIN, FMIN, GMIN'/3D25.14)
860    FORMAT(' NEW ESTIMATE'/2D25.14)
870    FORMAT(' ILOC AND ITEST'/2I3)
C     END
C
C
C     DOUBLE PRECISION FUNCTION STEP1(FNEW, FM, GTP, SMAX)
C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C     DOUBLE PRECISION FNEW, FM, GTP, SMAX
C
C     *****
C     STEP1 RETURNS THE LENGTH OF THE INITIAL STEP TO BE TAKEN ALONG THE

```

```

C VECTOR P IN THE NEXT LINEAR SEARCH.
C *****
C
      DOUBLE PRECISION ALPHA,D,EPSMCH
      DOUBLE PRECISION DABS,MCHPR1
      EPSMCH = MCHPR1()
      D = DABS(FNEW-FM)
      ALPHA = 1.D0
      IF (2.D0*D .LE. (-GTP) .AND. D .GE. EPSMCH)
*       ALPHA = -2.D0*D/GTP
      IF (ALPHA .GE. SMAX) ALPHA = SMAX
      STEP1 = ALPHA
      RETURN
      END

C
C
      DOUBLE PRECISION FUNCTION MCHPR1()
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION X

C
C RETURNS THE VALUE OF EPSMCH, WHERE EPSMCH IS THE SMALLEST POSSIBLE
C REAL NUMBER SUCH THAT 1.0 + EPSMCH .GT. 1.0
C
C FOR VAX
C
      MCHPR1 = 1.D-17

C
C FOR SUN
C
      MCHPR1 = 1.0842021724855D-19
      RETURN
      END

C
C
      SUBROUTINE CHKUCP(LWTEST,MAXFUN,NWHY,N,ALPHA,EPSMCH,
*       ETA,PEPS,RTEPS,RTOL,RTOLSQ,STEPMX,TEST,
*       XTOL,XNORM,X,LW,SMALL,TINY,ACCRCY)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER LW,LWTEST,MAXFUN,NWHY,N
      DOUBLE PRECISION ACCRCY,ALPHA,EPSMCH,ETA,PEPS,RTEPS,RTOL,
*       RTOLSQ,STEPMX,TEST,XTOL,XNORM,SMALL,TINY
      DOUBLE PRECISION X(N)

C
C CHECKS PARAMETERS AND SETS CONSTANTS WHICH ARE COMMON TO BOTH
C DERIVATIVE AND NON-DERIVATIVE ALGORITHMS
C
      DOUBLE PRECISION DABS,DSQRT,MCHPR1
      EPSMCH = MCHPR1()
      SMALL = EPSMCH*EPSMCH
      TINY = SMALL
      NWHY = -1
      RTEPS = DSQRT(EPSMCH)
      RTOL = XTOL
      IF (DABS(RTOL) .LT. ACCRCY) RTOL = 1.D1*RTEPS

C
C CHECK FOR ERRORS IN THE INPUT PARAMETERS
C

```

```

      IF (LW .LT. LWTEST
*       .OR. N .LT. 1 .OR. RTOL .LT. 0.D0 .OR. ETA .GE. 1.D0 .OR.
*       ETA .LT. 0.D0 .OR. STEPMX .LT. RTOL .OR.
*       MAXFUN .LT. 1) RETURN
      NWHY = 0
C
C SET CONSTANTS FOR LATER
C
      RTOLSQ = RTOL*RTOL
      PEPS = ACCRCY**0.6666D0
      XNORM = DNRM2(N,X,1)
      ALPHA = 0.D0
      TEST = 0.D0
      RETURN
      END
C
C
      SUBROUTINE SETUCR(SMALL,NFTOTL,NITER,N,F,FNEW,
*       FM,GTG,OLDF,SFUN,G,X)
      IMPLICIT      DOUBLE PRECISION (A-H,O-Z)
      INTEGER      NFTOTL,NITER,N
      DOUBLE PRECISION F,FNEW,FM,GTG,OLDF,SMALL
      DOUBLE PRECISION G(N),X(N)
      EXTERNAL     SFUN
C
C CHECK INPUT PARAMETERS, COMPUTE THE INITIAL FUNCTION VALUE, SET
C CONSTANTS FOR THE SUBSEQUENT MINIMIZATION
C
      FM = F
C
C COMPUTE THE INITIAL FUNCTION VALUE
C
      CALL SFUN(N,X,FNEW,G)
      NFTOTL = 1
C
C SET CONSTANTS FOR LATER
C
      NITER = 0
      OLDF = FNEW
      GTG = DDOT(N,G,1,G,1)
      RETURN
      END
C
C
      SUBROUTINE GTIMS(V,GV,N,X,G,W,LW,SFUN,FIRST,DELTA,ACCRCY,XNORM)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION V(N),GV(N),DINV,DELTA,G(N)
      DOUBLE PRECISION F,X(N),W(LW),ACCRCY,DSQRT,XNORM
      LOGICAL FIRST
      EXTERNAL SFUN
      COMMON/SUBSCR/ LGV,LZ1,LZK,LV,LSK,LYK,LDIAGB,LSR,LYR,
*       LHYR,LHG,LHYK,LPK,LEMAT,LWTEST
C
C THIS ROUTINE COMPUTES THE PRODUCT OF THE MATRIX G TIMES THE VECTOR
C V AND STORES THE RESULT IN THE VECTOR GV (FINITE-DIFFERENCE VERSION)
C
      IF (.NOT. FIRST) GO TO 20

```

```

DELTA = DSQRT(ACCRCY)*(1.D0+XNORM)
FIRST = .FALSE.
20 CONTINUE
DINV = 1.D0/DELTA
IHG = LHG
DO 30 I = 1,N
    W(IHG) = X(I) + DELTA*V(I)
    IHG = IHG + 1
30 CONTINUE
CALL SFUN(N,W(LHG),F,GV)
DO 40 I = 1,N
    GV(I) = (GV(I) - G(I))*DINV
40 CONTINUE
RETURN
END

C
C
SUBROUTINE MSOLVE(G,Y,N,W,LW,UPD1,YKSK,GSK,
*   YRSR,LRESET,FIRST)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DOUBLE PRECISION G(N),Y(N),W(LW),YKSK,GSK,YRSR
LOGICAL UPD1,LRESET,FIRST

C
C THIS ROUTINE SETS UP THE ARRAYS FOR MSLV
C
COMMON/SUBSCR/ LGV,LZ1,LZK,LV,LSK,LYK,LDIAGB,LSR,LYR,
*   LHYR,LHG,LHYK,LPK,LEMAT,LWTEST
CALL MSLV(G,Y,N,W(LSK),W(LYK),W(LDIAGB),W(LSR),W(LYR),W(LHYR),
*   W(LHG),W(LHYK),UPD1,YKSK,GSK,YRSR,LRESET,FIRST)
RETURN
END
SUBROUTINE MSLV(G,Y,N,SK,YK,DIAGB,SR,YR,HYR,HG,HYK,
*   UPD1,YKSK,GSK,YRSR,LRESET,FIRST)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DOUBLE PRECISION G(N),Y(N)

C
C THIS ROUTINE ACTS AS A PRECONDITIONING STEP FOR THE
C LINEAR CONJUGATE-GRADIENT ROUTINE. IT IS ALSO THE
C METHOD OF COMPUTING THE SEARCH DIRECTION FROM THE
C GRADIENT FOR THE NON-LINEAR CONJUGATE-GRADIENT CODE.
C IT REPRESENTS A TWO-STEP SELF-SCALED BFGS FORMULA.
C
DOUBLE PRECISION DDOT,YKSK,GSK,YRSR,RDIAGB,YKHYK,GHYK,
*   YKSR,YKHYP,YRHYR,GSR,GHYR
DOUBLE PRECISION SK(N),YK(N),DIAGB(N),SR(N),YR(N),HYR(N),HG(N),
*   HYK(N),ONE
LOGICAL LRESET,UPD1,FIRST
IF (UPD1) GO TO 100
ONE = 1.D0
GSK = DDOT(N,G,1,SK,1)
IF (LRESET) GO TO 60

C
C COMPUTE HG AND HY WHERE H IS THE INVERSE OF THE DIAGONALS
C
DO 57 I = 1,N
    RDIAGB = 1.0D0/DIAGB(I)
    HG(I) = G(I)*RDIAGB

```

```

        IF (FIRST) HYK(I) = YK(I)*RDIAGB
        IF (FIRST) HYR(I) = YR(I)*RDIAGB
57    CONTINUE
        IF (FIRST) YKSR = DDOT(N,YK,1,SR,1)
        IF (FIRST) YKHYR = DDOT(N,YK,1,HYR,1)
        GSR = DDOT(N,G,1,SR,1)
        GHYR = DDOT(N,G,1,HYR,1)
        IF (FIRST) YRHYR = DDOT(N,YR,1,HYR,1)
        CALL SSBFGS(N,ONE,SR,YR,HG,HYR,YRSR,
*       YRHYR,GSR,GHYR,HG)
        IF (FIRST) CALL SSBFGS(N,ONE,SR,YR,HYK,HYR,YRSR,
*       YRHYR,YKSR,YKHYR,HYK)
        YKHYK = DDOT(N,HYK,1,YK,1)
        GHYK = DDOT(N,HYK,1,G,1)
        CALL SSBFGS(N,ONE,SK,YK,HG,HYK,YKSK,
*       YKHYK,GSK,GHYK,Y)
        RETURN
60    CONTINUE
C
C COMPUTE GH AND HY WHERE H IS THE INVERSE OF THE DIAGONALS
C
        DO 65 I = 1,N
            RDIAGB = 1.D0/DIAGB(I)
            HG(I) = G(I)*RDIAGB
            IF (FIRST) HYK(I) = YK(I)*RDIAGB
65    CONTINUE
            IF (FIRST) YKHYK = DDOT(N,YK,1,HYK,1)
            GHYK = DDOT(N,G,1,HYK,1)
            CALL SSBFGS(N,ONE,SK,YK,HG,HYK,YKSK,
*       YKHYK,GSK,GHYK,Y)
            RETURN
100   CONTINUE
        DO 110 I = 1,N
110   Y(I) = G(I) / DIAGB(I)
        RETURN
        END
C
C
        SUBROUTINE SSBFGS(N,GAMMA,SJ,YJ,HJV,HJYJ,YJSJ,YJHYJ,
*       VSJ,VHYJ,HJP1V)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        INTEGER N
        DOUBLE PRECISION GAMMA,YJSJ,YJHYJ,VSJ,VHYJ
        DOUBLE PRECISION SJ(N),YJ(N),HJV(N),HJYJ(N),HJP1V(N)
C
C SELF-SCALED BFGS
C
        INTEGER I
        DOUBLE PRECISION BETA,DELTA
        DELTA = (1.D0 + GAMMA*YJHYJ/YJSJ)*VSJ/YJSJ
*       - GAMMA*VHYJ/YJSJ
        BETA = -GAMMA*VSJ/YJSJ
        DO 10 I = 1,N
            HJP1V(I) = GAMMA*HJV(I) + DELTA*SJ(I) + BETA*HJYJ(I)
10    CONTINUE
        RETURN
        END

```

C

C ROUTINES TO INITIALIZE PRECONDITIONER

C

```
      SUBROUTINE INITPC(DIAGB,EMAT,N,W,LW,MODET,
*      UPD1,YKSK,GSK,YRSR,LRESET)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION DIAGB(N),EMAT(N),W(LW)
      DOUBLE PRECISION YKSK,GSK,YRSR
      LOGICAL LRESET,UPD1
      COMMON/SUBSCR/ LGV,LZ1,LZK,LV,LSK,LYK,LDIAGB,LSR,LYR,
*      LHYR,LHG,LHYK,LPK,LEMAT,LWTEST
      CALL INITP3(DIAGB,EMAT,N,LRESET,YKSK,YRSR,W(LHYK),
*      W(LSK),W(LYK),W(LSR),W(LYR),MODET,UPD1)
      RETURN
      END
      SUBROUTINE INITP3(DIAGB,EMAT,N,LRESET,YKSK,YRSR,BSK,
*      SK,YK,SR,YR,MODET,UPD1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION DIAGB(N),EMAT(N),YKSK,YRSR,BSK(N),SK(N),
*      YK(N),COND,SR(N),YR(N),DDOT,SDS,SRDS,YRSK,TD,D1,DN
      LOGICAL LRESET,UPD1
      IF (UPD1) GO TO 90
      IF (LRESET) GO TO 60
      DO 10 I = 1,N
         BSK(I) = DIAGB(I)*SR(I)
10      CONTINUE
         SDS = DDOT(N,SR,1,BSK,1)
         SRDS = DDOT(N,SK,1,BSK,1)
         YRSK = DDOT(N,YR,1,SK,1)
         DO 20 I = 1,N
            TD = DIAGB(I)
            BSK(I) = TD*SK(I) - BSK(I)*SRDS/SDS+YR(I)*YRSK/YRSR
            EMAT(I) = TD-TD*TD*SR(I)*SR(I)/SDS+YR(I)*YR(I)/YRSR
20      CONTINUE
         SDS = DDOT(N,SK,1,BSK,1)
         DO 30 I = 1,N
            EMAT(I) = EMAT(I) - BSK(I)*BSK(I)/SDS+YK(I)*YK(I)/YKSK
30      CONTINUE
         GO TO 110
60      CONTINUE
         DO 70 I = 1,N
            BSK(I) = DIAGB(I)*SK(I)
70      CONTINUE
         SDS = DDOT(N,SK,1,BSK,1)
         DO 80 I = 1,N
            TD = DIAGB(I)
            EMAT(I) = TD - TD*TD*SK(I)*SK(I)/SDS + YK(I)*YK(I)/YKSK
80      CONTINUE
         GO TO 110
90      CONTINUE
         CALL DCOPY(N,DIAGB,1,EMAT,1)
110     CONTINUE
         IF (MODET .LT. 1) RETURN
         D1 = EMAT(1)
         DN = EMAT(1)
         DO 120 I = 1,N
            IF (EMAT(I) .LT. D1) D1 = EMAT(I)
```

```

        IF (EMAT(I) .GT. DN) DN = EMAT(I)
120  CONTINUE
      COND = DN/D1
      WRITE(*,800) D1,DN,COND
800  FORMAT(' ',//8X,'DMIN =',1PD12.4,' DMAX =',1PD12.4,
*         ' COND =',1PD12.4,/)
      RETURN
      END

C
C
      SUBROUTINE SETPAR(N)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER LSUB(14)
      COMMON/SUBSCR/ LSUB,LWTEST

C
C SET UP PARAMETERS FOR THE OPTIMIZATION ROUTINE
C
      DO 10 I = 1,14
          LSUB(I) = (I-1)*N + 1
10  CONTINUE
      LWTEST = LSUB(14) + N - 1
      RETURN
      END

C
C   LINE SEARCH ALGORITHMS OF GILL AND MURRAY
C
      SUBROUTINE LINDER(N,SFUN,SMALL,EPSMCH,RELTOL,ABSTOL,
*      TNYTOL,ETA,SFTBND,XBND,P,GTP,X,F,ALPHA,G,NFTOTL,
*      IFLAG,W,LW)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER N,NFTOTL,IFLAG,LW
      DOUBLE PRECISION SMALL,EPSMCH,RELTOL,ABSTOL,TNYTOL,ETA,
*      SFTBND,XBND,GTP,F,ALPHA
      DOUBLE PRECISION P(N),X(N),G(N),W(LW)

C
C
      INTEGER I,IENTRY,ITEST,L,LG,LX,NUMF,ITCNT
      DOUBLE PRECISION A,B,B1,BIG,E,FACTOR,FMIN,FPRESN,FU,
*      FW,GMIN,GTEST1,GTEST2,GU,GW,OLDF,SCXBND,STEP,
*      TOL,U,XMIN,XW,RMU,RTSMML,UALPHA
      LOGICAL BRAKTD

C
C   THE FOLLOWING STANDARD FUNCTIONS AND SYSTEM FUNCTIONS ARE
C   CALLED WITHIN LINDER
C
      DOUBLE PRECISION DDOT,DSQRT
      EXTERNAL SFUN

C
C   ALLOCATE THE ADDRESSES FOR LOCAL WORKSPACE
C
      LX = 1
      LG = LX + N
      LSPRNT = 0
      NPRNT = 10000
      RTSMML = DSQRT(SMALL)
      BIG = 1.D0/SMALL
      ITCNT = 0

```

```

C
C   SET THE ESTIMATED RELATIVE PRECISION IN F(X).
C
FPRESN = 10.D0*EPSMCH
NUMF = 0
U = ALPHA
FU = F
FMIN = F
GU = GTP
RMU = 1.0D-4

C
C   FIRST ENTRY SETS UP THE INITIAL INTERVAL OF UNCERTAINTY.
C
IENTRY = 1
10  CONTINUE
C
C TEST FOR TOO MANY ITERATIONS
C
ITCNT = ITCNT + 1
IFLAG = 1
IF (ITCNT .GT. 20) GO TO 50
IFLAG = 0
CALL GETPTC(BIG,SMALL,RTSMALL,RELTOL,ABSTOL,TNYTOL,
*   FPRESN,ETA,RMU,XBND,U,FU,GU,XMIN,FMIN,GMIN,
*   XW,FW,GW,A,B,OLDF,B1,SCXBND,E,STEP,FACTOR,
*   BRAKTD,GTEST1,GTEST2,TOL,IENTRY,ITEST)
CLSOUT
IF (LSPRNT .GE. NPRNT) CALL LSOUT(IENTRY,ITEST,XMIN,FMIN,GMIN,
*   XW,FW,GW,U,A,B,TOL,RELTOL,SCXBND,XBND)

C
C   IF ITEST=1, THE ALGORITHM REQUIRES THE FUNCTION VALUE TO BE
C   CALCULATED.
C
IF (ITEST .NE. 1) GO TO 30
UALPHA = XMIN + U
L = LX
DO 20 I = 1,N
  W(L) = X(I) + UALPHA*P(I)
  L = L + 1
20  CONTINUE
CALL SFUN(N,W(LX),FU,W(LG))
NUMF = NUMF + 1
GU = DDOT(N,W(LG),1,P,1)

C
C   THE GRADIENT VECTOR CORRESPONDING TO THE BEST POINT IS
C   OVERWRITTEN IF FU IS LESS THAN FMIN AND FU IS SUFFICIENTLY
C   LOWER THAN F AT THE ORIGIN.
C
IF (FU .LE. FMIN .AND. FU .LE. OLDF-UALPHA*GTEST1)
*   CALL DCOPY(N,W(LG),1,G,1)
GOTO 10

C
C   IF ITEST=2 OR 3 A LOWER POINT COULD NOT BE FOUND
C
30  CONTINUE
NFTOTL = NUMF
IFLAG = 1

```

```

      IF (ITEST .NE. 0) GO TO 50
C
C      IF ITEST=0 A SUCCESSFUL SEARCH HAS BEEN MADE
C
      IFLAG = 0
      F = FMIN
      ALPHA = XMIN
      DO 40 I = 1,N
          X(I) = X(I) + ALPHA*P(I)
40    CONTINUE
50    RETURN
      END
C
C
      SUBROUTINE GETPTC(BIG,SMALL,RTSMLL,RELTOL,ABSTOL,TNYTOL,
*      FPRESN,ETA,RMU,XBND,U,FU,GU,XMIN,FMIN,GMIN,
*      XW,FW,GW,A,B,OLDF,B1,SCXBND,E,STEP,FACTOR,
*      BRAKTD,GTEST1,GTEST2,TOL,IENTRY,ITEST)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL BRAKTD
      INTEGER IENTRY,ITEST
      DOUBLE PRECISION BIG,SMALL,RTSMLL,RELTOL,ABSTOL,TNYTOL,
*      FPRESN,ETA,RMU,XBND,U,FU,GU,XMIN,FMIN,GMIN,
*      XW,FW,GW,A,B,OLDF,B1,SCXBND,E,STEP,FACTOR,
*      GTEST1,GTEST2,TOL,DENOM
C
C *****
C GETPTC, AN ALGORITHM FOR FINDING A STEPLENGTH, CALLED REPEATEDLY BY
C ROUTINES WHICH REQUIRE A STEP LENGTH TO BE COMPUTED USING CUBIC
C INTERPOLATION. THE PARAMETERS CONTAIN INFORMATION ABOUT THE INTERVAL
C IN WHICH A LOWER POINT IS TO BE FOUND AND FROM THIS GETPTC COMPUTES A
C POINT AT WHICH THE FUNCTION CAN BE EVALUATED BY THE CALLING PROGRAM.
C THE VALUE OF THE INTEGER PARAMETERS IENTRY DETERMINES THE PATH TAKEN
C THROUGH THE CODE.
C *****
C
      LOGICAL CONVRG
      DOUBLE PRECISION ABGMIN,ABGW,ABSR,A1,CHORDM,CHORDU,
*      D1,D2,P,Q,R,S,SCALE,SUMSQ,TWOTOL,XMIDPT
      DOUBLE PRECISION ZERO, POINT1,HALF,ONE,THREE,FIVE,ELEVEN
C
C THE FOLLOWING STANDARD FUNCTIONS AND SYSTEM FUNCTIONS ARE CALLED
C WITHIN GETPTC
C
      DOUBLE PRECISION DABS, DSQRT
C
      ZERO = 0.D0
      POINT1 = 1.D-1
      HALF = 5.D-1
      ONE = 1.D0
      THREE = 3.D0
      FIVE = 5.D0
      ELEVEN = 11.D0
C
C      BRANCH TO APPROPRIATE SECTION OF CODE DEPENDING ON THE
C      VALUE OF IENTRY.
C

```

```

GOTO (10,20), IENTRY
C
C   IENTRY=1
C   CHECK INPUT PARAMETERS
C
10  ITEST = 2
    IF (U .LE. ZERO .OR. XBND .LE. TNYTOL .OR. GU .GT. ZERO)
*   RETURN
    ITEST = 1
    IF (XBND .LT. ABSTOL) ABSTOL = XBND
    TOL = ABSTOL
    TWOTOL = TOL + TOL
C
C A AND B DEFINE THE INTERVAL OF UNCERTAINTY, X AND XW ARE POINTS
C WITH LOWEST AND SECOND LOWEST FUNCTION VALUES SO FAR OBTAINED.
C INITIALIZE A, SMIN, XW AT ORIGIN AND CORRESPONDING VALUES OF
C FUNCTION AND PROJECTION OF THE GRADIENT ALONG DIRECTION OF SEARCH
C AT VALUES FOR LATEST ESTIMATE AT MINIMUM.
C
    A = ZERO
    XW = ZERO
    XMIN = ZERO
    OLDF = FU
    FMIN = FU
    FW = FU
    GW = GU
    GMIN = GU
    STEP = U
    FACTOR = FIVE
C
C   THE MINIMUM HAS NOT YET BEEN BRACKETED.
C
    BRAKTD = .FALSE.
C
C SET UP XBND AS A BOUND ON THE STEP TO BE TAKEN. (XBND IS NOT COMPUTED
C EXPLICITLY BUT SCXBND IS ITS SCALED VALUE.) SET THE UPPER BOUND
C ON THE INTERVAL OF UNCERTAINTY INITIALLY TO XBND + TOL(XBND).
C
    SCXBND = XBND
    B = SCXBND + RELTOL*DABS(SCXBND) + ABSTOL
    E = B + B
    B1 = B
C
C COMPUTE THE CONSTANTS REQUIRED FOR THE TWO CONVERGENCE CRITERIA.
C
    GTEST1 = -RMU*GU
    GTEST2 = -ETA*GU
C
C SET IENTRY TO INDICATE THAT THIS IS THE FIRST ITERATION
C
    IENTRY = 2
    GO TO 210
C
C IENTRY = 2
C
C UPDATE A, B, XW, AND XMIN
C

```

```

20     IF (FU .GT. FMIN) GO TO 60
C
C IF FUNCTION VALUE NOT INCREASED, NEW POINT BECOMES NEXT
C ORIGIN AND OTHER POINTS ARE SCALED ACCORDINGLY.
C
      CHORDU = OLDF - (XMIN + U)*GTEST1
      IF (FU .LE. CHORDU) GO TO 30
C
C THE NEW FUNCTION VALUE DOES NOT SATISFY THE SUFFICIENT DECREASE
C CRITERION. PREPARE TO MOVE THE UPPER BOUND TO THIS POINT AND
C FORCE THE INTERPOLATION SCHEME TO EITHER BISECT THE INTERVAL OF
C UNCERTAINTY OR TAKE THE LINEAR INTERPOLATION STEP WHICH ESTIMATES
C THE ROOT OF F(ALPHA)=CHORD(ALPHA).
C
      CHORDM = OLDF - XMIN*GTEST1
      GU = -GMIN
      DENOM = CHORDM-FMIN
      IF (DABS(DENOM) .GE. 1.D-15) GO TO 25
          DENOM = 1.D-15
          IF (CHORDM-FMIN .LT. 0.D0) DENOM = -DENOM
25     CONTINUE
      IF (XMIN .NE. ZERO) GU = GMIN*(CHORDU-FU)/DENOM
      FU = HALF*U*(GMIN+GU) + FMIN
      IF (FU .LT. FMIN) FU = FMIN
      GO TO 60
30     FW = FMIN
      FMIN = FU
      GW = GMIN
      GMIN = GU
      XMIN = XMIN + U
      A = A-U
      B = B-U
      XW = -U
      SCXBND = SCXBND - U
      IF (GU .LE. ZERO) GO TO 40
      B = ZERO
      BRAKTD = .TRUE.
      GO TO 50
40     A = ZERO
50     TOL = DABS(XMIN)*RELTOL + ABSTOL
      GO TO 90
C
C IF FUNCTION VALUE INCREASED, ORIGIN REMAINS UNCHANGED
C BUT NEW POINT MAY NOW QUALIFY AS W.
C
60     IF (U .LT. ZERO) GO TO 70
      B = U
      BRAKTD = .TRUE.
      GO TO 80
70     A = U
80     XW = U
      FW = FU
      GW = GU
90     TWOTOL = TOL + TOL
      XMIDPT = HALF*(A + B)
C
C CHECK TERMINATION CRITERIA

```

```

C
CONVRG = DABS(XMIDPT) .LE. TWOTOL - HALF*(B-A) .OR.
*   DABS(GMIN) .LE. GTEST2 .AND. FMIN .LT. OLDF .AND.
*   (DABS(XMIN - XBND) .GT. TOL .OR. .NOT. BRAKTD)
IF (.NOT. CONVRG) GO TO 100
ITEST = 0
IF (XMIN .NE. ZERO) RETURN

C
C IF THE FUNCTION HAS NOT BEEN REDUCED, CHECK TO SEE THAT THE RELATIVE
C CHANGE IN F(X) IS CONSISTENT WITH THE ESTIMATE OF THE DELTA-
C UNIMODALITY CONSTANT, TOL. IF THE CHANGE IN F(X) IS LARGER THAN
C EXPECTED, REDUCE THE VALUE OF TOL.
C
ITEST = 3
IF (DABS(OLDF-FW) .LE. FPRESN*(ONE + DABS(OLDF))) RETURN
TOL = POINT1*TOL
IF (TOL .LT. TNYTOL) RETURN
RELTOL = POINT1*RELTOL
ABSTOL = POINT1*ABSTOL
TWOTOL = POINT1*TWOTOL

C
C CONTINUE WITH THE COMPUTATION OF A TRIAL STEP LENGTH
C
100 R = ZERO
Q = ZERO
S = ZERO
IF (DABS(E) .LE. TOL) GO TO 150

C
C FIT CUBIC THROUGH XMIN AND XW
C
R = THREE*(FMIN-FW)/XW + GMIN + GW
ABSR = DABS(R)
Q = ABSR
IF (GW .EQ. ZERO .OR. GMIN .EQ. ZERO) GO TO 140

C
C COMPUTE THE SQUARE ROOT OF (R*R - GMIN*GW) IN A WAY
C WHICH AVOIDS UNDERFLOW AND OVERFLOW.
C
ABGW = DABS(GW)
ABGMIN = DABS(GMIN)
S = DSQRT(ABGMIN)*DSQRT(ABGW)
IF ((GW/ABGW)*GMIN .GT. ZERO) GO TO 130

C
C COMPUTE THE SQUARE ROOT OF R*R + S*S.
C
SUMSQ = ONE
P = ZERO
IF (ABSR .GE. S) GO TO 110

C
C THERE IS A POSSIBILITY OF OVERFLOW.
C
IF (S .GT. RTSMLL) P = S*RTSMLL
IF (ABSR .GE. P) SUMSQ = ONE +(ABSR/S)**2
SCALE = S
GO TO 120

C
C THERE IS A POSSIBILITY OF UNDERFLOW.

```

```

C
110  IF (ABSR .GT. RTSMLL) P = ABSR*RTSMLL
      IF (S .GE. P) SUMSQ = ONE + (S/ABSR)**2
      SCALE = ABSR
120  SUMSQ = DSQRT(SUMSQ)
      Q = BIG
      IF (SCALE .LT. BIG/SUMSQ) Q = SCALE*SUMSQ
      GO TO 140
C
C COMPUTE THE SQUARE ROOT OF R*R - S*S
C
130  Q = DSQRT(DABS(R+S))*DSQRT(DABS(R-S))
      IF (R .GE. S .OR. R .LE. (-S)) GO TO 140
      R = ZERO
      Q = ZERO
      GO TO 150
C
C COMPUTE THE MINIMUM OF FITTED CUBIC
C
140  IF (XW .LT. ZERO) Q = -Q
      S = XW*(GMIN - R - Q)
      Q = GW - GMIN + Q + Q
      IF (Q .GT. ZERO) S = -S
      IF (Q .LE. ZERO) Q = -Q
      R = E
      IF (B1 .NE. STEP .OR. BRAKTD) E = STEP
C
C CONSTRUCT AN ARTIFICIAL BOUND ON THE ESTIMATED STEPLENGTH
C
150  A1 = A
      B1 = B
      STEP = XMIDPT
      IF (BRAKTD) GO TO 160
      STEP = -FACTOR*XW
      IF (STEP .GT. SCXBND) STEP = SCXBND
      IF (STEP .NE. SCXBND) FACTOR = FIVE*FACTOR
      GO TO 170
C
C IF THE MINIMUM IS BRACKETED BY 0 AND XW THE STEP MUST LIE
C WITHIN (A,B).
C
160  IF ((A .NE. ZERO .OR. XW .GE. ZERO) .AND. (B .NE. ZERO .OR.
      *   XW .LE. ZERO)) GO TO 180
C
C IF THE MINIMUM IS NOT BRACKETED BY 0 AND XW THE STEP MUST LIE
C WITHIN (A1,B1).
C
      D1 = XW
      D2 = A
      IF (A .EQ. ZERO) D2 = B
C THIS LINE MIGHT BE
C
      IF (A .EQ. ZERO) D2 = E
      U = - D1/D2
      STEP = FIVE*D2*(POINT1 + ONE/U)/ELEVEN
      IF (U .LT. ONE) STEP = HALF*D2*DSQRT(U)
170  IF (STEP .LE. ZERO) A1 = STEP
      IF (STEP .GT. ZERO) B1 = STEP

```

```

C
C REJECT THE STEP OBTAINED BY INTERPOLATION IF IT LIES OUTSIDE THE
C REQUIRED INTERVAL OR IT IS GREATER THAN HALF THE STEP OBTAINED
C DURING THE LAST-BUT-ONE ITERATION.
C
180  IF (DABS(S) .LE. DABS(HALF*Q*R) .OR.
      *      S .LE. Q*A1 .OR. S .GE. Q*B1) GO TO 200
C
C A CUBIC INTERPOLATION STEP
C
      STEP = S/Q
C
C THE FUNCTION MUST NOT BE EVALUTATED TOO CLOSE TO A OR B.
C
      IF (STEP - A .GE. TWOTOL .AND. B - STEP .GE. TWOTOL) GO TO 210
      IF (XMIDPT .GT. ZERO) GO TO 190
      STEP = -TOL
      GO TO 210
190  STEP = TOL
      GO TO 210
200  E = B-A
C
C IF THE STEP IS TOO LARGE, REPLACE BY THE SCALED BOUND (SO AS TO
C COMPUTE THE NEW POINT ON THE BOUNDARY).
C
210  IF (STEP .LT. SCXBND) GO TO 220
      STEP = SCXBND
C
C MOVE SXBD TO THE LEFT SO THAT SBND + TOL(XBND) = XBND.
C
      SCXBND = SCXBND - (RELTOL*DABS(XBND)+ABSTOL)/(ONE + RELTOL)
220  U = STEP
      IF (DABS(STEP) .LT. TOL .AND. STEP .LT. ZERO) U = -TOL
      IF (DABS(STEP) .LT. TOL .AND. STEP .GE. ZERO) U = TOL
      ITEST = 1
      RETURN
      END
C%% TRUNCATED-NEWTON METHOD: BLAS
C  NOTE: ALL ROUTINES HERE ARE FROM LINPACK WITH THE EXCEPTION
C        OF DXPY (A VERSION OF DAXPY WITH A=1.0)
C  WRITTEN BY:  STEPHEN G. NASH
C                OPERATIONS RESEARCH AND APPLIED STATISTICS DEPT.
C                GEORGE MASON UNIVERSITY
C                FAIRFAX, VA 22030
C*****
      DOUBLE PRECISION FUNCTION DDOT(N,DX,INCX,DY,INCY)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C  FORMS THE DOT PRODUCT OF TWO VECTORS.
C  USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO ONE.
C  JACK DONGARRA, LINPACK, 3/11/78.
C
      DOUBLE PRECISION DX(1),DY(1),DTEMP
      INTEGER I,INCX,INCY,IX,IY,M,MP1,N
C
      DDOT = 0.0D0
      DTEMP = 0.0D0

```

```

IF(N.LE.0)RETURN
IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C      CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C      NOT EQUAL TO 1
C
IX = 1
IY = 1
IF(INCX.LT.0)IX = (-N+1)*INCX + 1
IF(INCY.LT.0)IY = (-N+1)*INCY + 1
DO 10 I = 1,N
    DTEMP = DTEMP + DX(IX)*DY(IY)
    IX = IX + INCX
    IY = IY + INCY
10 CONTINUE
DDOT = DTEMP
RETURN
C
C      CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C      CLEAN-UP LOOP
C
20 M = MOD(N,5)
IF( M .EQ. 0 ) GO TO 40
DO 30 I = 1,M
    DTEMP = DTEMP + DX(I)*DY(I)
30 CONTINUE
IF( N .LT. 5 ) GO TO 60
40 MP1 = M + 1
DO 50 I = MP1,N,5
    DTEMP = DTEMP + DX(I)*DY(I) + DX(I + 1)*DY(I + 1) +
*   DX(I + 2)*DY(I + 2) + DX(I + 3)*DY(I + 3) + DX(I + 4)*DY(I + 4)
50 CONTINUE
60 DDOT = DTEMP
RETURN
END
SUBROUTINE DAXPY(N,DA,DX,INCX,DY,INCY)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C      CONSTANT TIMES A VECTOR PLUS A VECTOR.
C      USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO ONE.
C      JACK DONGARRA, LINPACK, 3/11/78.
C
DOUBLE PRECISION DX(1),DY(1),DA
INTEGER I,INCX,INCY,IX,IY,M,MP1,N
C
IF(N.LE.0)RETURN
IF (DA .EQ. 0.0D0) RETURN
IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C      CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C      NOT EQUAL TO 1
C
IX = 1
IY = 1
IF(INCX.LT.0)IX = (-N+1)*INCX + 1

```

```

IF(INCY.LT.0)IY = (-N+1)*INCY + 1
DO 10 I = 1,N
  DY(IY) = DY(IY) + DA*DX(IX)
  IX = IX + INCX
  IY = IY + INCY
10 CONTINUE
RETURN

C
C      CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C      CLEAN-UP LOOP
C
20 M = MOD(N,4)
IF( M .EQ. 0 ) GO TO 40
DO 30 I = 1,M
  DY(I) = DY(I) + DA*DX(I)
30 CONTINUE
IF( N .LT. 4 ) RETURN
40 MP1 = M + 1
DO 50 I = MP1,N,4
  DY(I) = DY(I) + DA*DX(I)
  DY(I + 1) = DY(I + 1) + DA*DX(I + 1)
  DY(I + 2) = DY(I + 2) + DA*DX(I + 2)
  DY(I + 3) = DY(I + 3) + DA*DX(I + 3)
50 CONTINUE
RETURN
END
DOUBLE PRECISION FUNCTION DNRM2 ( N, DX, INCX)
IMPLICIT      DOUBLE PRECISION (A-H,O-Z)
INTEGER      NEXT
DOUBLE PRECISION DX(1),CUTLO,CUTHI,HITEST,SUM,XMAX,ZERO,ONE
DATA  ZERO, ONE /0.0D0, 1.0D0/

C
C      EUCLIDEAN NORM OF THE N-VECTOR STORED IN DX() WITH STORAGE
C      INCREMENT INCX .
C      IF N .LE. 0 RETURN WITH RESULT = 0.
C      IF N .GE. 1 THEN INCX MUST BE .GE. 1
C
C      C.L.LAWSON, 1978 JAN 08
C
C      FOUR PHASE METHOD      USING TWO BUILT-IN CONSTANTS THAT ARE
C      HOPEFULLY APPLICABLE TO ALL MACHINES.
C      CUTLO = MAXIMUM OF DSQRT(U/EPS)  OVER ALL KNOWN MACHINES.
C      CUTHI = MINIMUM OF DSQRT(V)      OVER ALL KNOWN MACHINES.
C      WHERE
C      EPS = SMALLEST NO. SUCH THAT EPS + 1. .GT. 1.
C      U   = SMALLEST POSITIVE NO.      (UNDERFLOW LIMIT)
C      V   = LARGEST NO.                 (OVERFLOW LIMIT)
C
C      BRIEF OUTLINE OF ALGORITHM..
C
C      PHASE 1      SCANS ZERO COMPONENTS.
C      MOVE TO PHASE 2 WHEN A COMPONENT IS NONZERO AND .LE. CUTLO
C      MOVE TO PHASE 3 WHEN A COMPONENT IS .GT. CUTLO
C      MOVE TO PHASE 4 WHEN A COMPONENT IS .GE. CUTHI/M
C      WHERE M = N FOR X() REAL AND M = 2*N FOR COMPLEX.

```

```

C
C   VALUES FOR CUTLO AND CUTHI..
C   FROM THE ENVIRONMENTAL PARAMETERS LISTED IN THE IMSL CONVERTER
C   DOCUMENT THE LIMITING VALUES ARE AS FOLLOWS..
C   CUTLO, S.P.   U/EPS = 2**(-102) FOR HONEYWELL.  CLOSE SECONDS ARE
C                 UNIVAC AND DEC AT 2**(-103)
C                 THUS CUTLO = 2**(-51) = 4.44089E-16
C   CUTHI, S.P.   V = 2**127 FOR UNIVAC, HONEYWELL, AND DEC.
C                 THUS CUTHI = 2**(63.5) = 1.30438E19
C   CUTLO, D.P.   U/EPS = 2**(-67) FOR HONEYWELL AND DEC.
C                 THUS CUTLO = 2**(-33.5) = 8.23181D-11
C   CUTHI, D.P.   SAME AS S.P.  CUTHI = 1.30438D19
C   DATA CUTLO, CUTHI / 8.232D-11, 1.304D19 /
C   DATA CUTLO, CUTHI / 4.441E-16, 1.304E19 /
C   DATA CUTLO, CUTHI / 8.232D-11, 1.304D19 /
C
C   IF(N .GT. 0) GO TO 10
C       DNRM2 = ZERO
C       GO TO 300
C
C   10 ASSIGN 30 TO NEXT
C       SUM = ZERO
C       NN = N * INCX
C
C                                     BEGIN MAIN LOOP
C   I = 1
C   20 GO TO NEXT,(30, 50, 70, 110)
C   30 IF( DABS(DX(I)) .GT. CUTLO) GO TO 85
C       ASSIGN 50 TO NEXT
C       XMAX = ZERO
C
C
C           PHASE 1.  SUM IS ZERO
C
C   50 IF( DX(I) .EQ. ZERO) GO TO 200
C       IF( DABS(DX(I)) .GT. CUTLO) GO TO 85
C
C
C           PREPARE FOR PHASE 2.
C   ASSIGN 70 TO NEXT
C   GO TO 105
C
C
C           PREPARE FOR PHASE 4.
C
C   100 I = J
C       ASSIGN 110 TO NEXT
C       SUM = (SUM / DX(I)) / DX(I)
C   105 XMAX = DABS(DX(I))
C       GO TO 115
C
C
C           PHASE 2.  SUM IS SMALL.
C           SCALE TO AVOID DESTRUCTIVE UNDERFLOW.
C
C   70 IF( DABS(DX(I)) .GT. CUTLO ) GO TO 75
C
C
C           COMMON CODE FOR PHASES 2 AND 4.
C           IN PHASE 4 SUM IS LARGE.  SCALE TO AVOID OVERFLOW.
C
C   110 IF( DABS(DX(I)) .LE. XMAX ) GO TO 115
C       SUM = ONE + SUM * (XMAX / DX(I))**2

```

```

        XMAX = DABS(DX(I))
        GO TO 200
C
115 SUM = SUM + (DX(I)/XMAX)**2
    GO TO 200
C
C
C           PREPARE FOR PHASE 3.
C
75 SUM = (SUM * XMAX) * XMAX
C
C
C   FOR REAL OR D.P. SET HITEST = CUTHI/N
C   FOR COMPLEX      SET HITEST = CUTHI/(2*N)
C
85 HITEST = CUTHI/FLOAT( N )
C
C           PHASE 3.  SUM IS MID-RANGE.  NO SCALING.
C
    DO 95 J =I,NN,INCX
    IF(DABS(DX(J)) .GE. HITEST) GO TO 100
95   SUM = SUM + DX(J)**2
    DNRM2 = DSQRT( SUM )
    GO TO 300
C
200 CONTINUE
    I = I + INCX
    IF ( I .LE. NN ) GO TO 20
C
C           END OF MAIN LOOP.
C
C           COMPUTE SQUARE ROOT AND ADJUST FOR SCALING.
C
    DNRM2 = XMAX * DSQRT(SUM)
300 CONTINUE
    RETURN
    END
    SUBROUTINE DCOPY(N,DX,INCX,DY,INCY)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C   COPIES A VECTOR, X, TO A VECTOR, Y.
C   USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO ONE.
C   JACK DONGARRA, LINPACK, 3/11/78.
C
    DOUBLE PRECISION DX(1),DY(1)
    INTEGER I,INCX,INCY,IX,IY,M,MP1,N
C
    IF(N.LE.0)RETURN
    IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C           CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C           NOT EQUAL TO 1
C
    IX = 1
    IY = 1
    IF(INCX.LT.0)IX = (-N+1)*INCX + 1
    IF(INCY.LT.0)IY = (-N+1)*INCY + 1

```

```

DO 10 I = 1,N
  DY(IY) = DX(IX)
  IX = IX + INCX
  IY = IY + INCY
10 CONTINUE
RETURN

C
C      CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C      CLEAN-UP LOOP
C
20 M = MOD(N,7)
  IF( M .EQ. 0 ) GO TO 40
  DO 30 I = 1,M
    DY(I) = DX(I)
30 CONTINUE
  IF( N .LT. 7 ) RETURN
40 MP1 = M + 1
  DO 50 I = MP1,N,7
    DY(I) = DX(I)
    DY(I + 1) = DX(I + 1)
    DY(I + 2) = DX(I + 2)
    DY(I + 3) = DX(I + 3)
    DY(I + 4) = DX(I + 4)
    DY(I + 5) = DX(I + 5)
    DY(I + 6) = DX(I + 6)
50 CONTINUE
RETURN
END

C*****
C SPECIAL BLAS FOR Y = X+Y
C*****
SUBROUTINE DXPY(N,DX,INCX,DY,INCY)
  IMPLICIT DOUBLE PRECISION (A-H,O-Z)

C
C      VECTOR PLUS A VECTOR.
C      USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO ONE.
C      STEPHEN G. NASH 5/30/89.
C
  DOUBLE PRECISION DX(1),DY(1)
  INTEGER I,INCX,INCY,IX,IY,M,MP1,N

C
  IF(N.LE.0)RETURN
  IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20

C
C      CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C      NOT EQUAL TO 1
C
  IX = 1
  IY = 1
  IF(INCX.LT.0)IX = (-N+1)*INCX + 1
  IF(INCY.LT.0)IY = (-N+1)*INCY + 1
  DO 10 I = 1,N
    DY(IY) = DY(IY) + DX(IX)
    IX = IX + INCX
    IY = IY + INCY

```

```
10 CONTINUE
   RETURN
```

```
C
C
C
C
C
```

```
       CODE FOR BOTH INCREMENTS EQUAL TO 1
```

```
       CLEAN-UP LOOP
```

```
20 M = MOD(N,4)
   IF( M .EQ. 0 ) GO TO 40
   DO 30 I = 1,M
     DY(I) = DY(I) + DX(I)
30 CONTINUE
   IF( N .LT. 4 ) RETURN
40 MP1 = M + 1
   DO 50 I = MP1,N,4
     DY(I) = DY(I) + DX(I)
     DY(I + 1) = DY(I + 1) + DX(I + 1)
     DY(I + 2) = DY(I + 2) + DX(I + 2)
     DY(I + 3) = DY(I + 3) + DX(I + 3)
50 CONTINUE
   RETURN
   END
```