Appendix 4

BANDED< SYMMETRIC< POSITIVE-DEFINITE MATRICES

BPBS - Back Solve
BPCE - Condition Estimation
BPDC - DeComposition
BPFS - Forward Solve
BPLE - Linear Equation solution
BPLD - LDL^T decomposition
BPML - MuLtiplication
BPNM - NorM
BPSS - System Solution

Purpose: BPBS (Banded symmetric Positive definite matrix Back Solve) solves DRX = B where D is diagonal and R is banded unit upper triangular (1's on the diagonal and 0's below the diagonal). It can be used for the back solution phase of a banded linear system solution. (It is used in this way by the routines BPSS and BPLE.)

Usage: CALL BPBS (N, ML, G, IG, B, IB, NB)

N	$\longrightarrow$	the number of equations
ML	$\longrightarrow$	the number of nonzero diagonals of R (including the unit diagonal)
G	$\longrightarrow$	a matrix (which may contain results obtained by the routines BPLD, BPCE, or BPDC) into which D and R are packed as follows: G(1, i) = d_i G(j–i + 1, i) = r_{ij} for j > i (See the introduction to this chapter.) G should be dimensioned (IG,KG) in the calling program, where IG≥ML and KG≥N.
IG	$\longrightarrow$	the row (leading) dimension of G, as dimensioned in the calling program
B	$\longrightarrow$	the matrix of right-hand sides, dimensioned (IB, KB) in the calling program, where IB≥N and KB≥NB.
	$\longleftarrow$	the solution X
IB	$\longrightarrow$	the row (leading) dimension of B, as dimensioned in the calling program
NB	$\longrightarrow$	the number of right-hand sides

Note: BPFS and BPBS can be used directly on the output matrix produced by BPDC, BPLD, or BPCE to solve a banded symmetric positive definite linear system.

Error situations: *(The user can elect to ‘recover’ from those errors marked with an asterisk — see *Error Handling*, Framework Chapter)

Number	Error
1	N < 1
2	ML < 1
3	IG < ML
4	IB < N
5	NB < 1
10 + k*	singular D with kth diagonal element 0.0

Double-precision version: DBPBS, with G and B declared double precision

Complex Hermitian version: CBPBS with G and B declared complex

Storage: None

Time: NB×(ML−1)×N additions
NB×(ML−1)×N multiplications
NB×N divisions

See also: BPCE, BPDC, BPFS, BPLD, BPLE, BPSS

Author: Linda Kaufman

Example: The program fragment below solves a linear system AX = B, where A is a symmetric positive definite band matrix. It is assumed that the A matrix has been packed into G according to the scheme G(j–i+1, i)=a_{ij}. The subroutine BPCE factors A into LDL^T, where L is unit lower triangular and D is diagonal. The factors are returned in G so that BPFS can forward solve (solve LY=B) and BPBS can back solve (solve $DL^TX=Y$).

The subroutine BPCE also provides an estimate of the condition number of A. In the code below if the condition number is larger than the reciprocal of the machine precision (given by R1MACH(4)), the matrix is considered too ill-conditioned and the system is not solved.

```
          IWRITE=I1MACH(2)
          CALL BPCE(N,ML,G,IG,COND)
          IF (COND .GT. 1.0/R1MACH(4)) GO TO 10
          CALL BPFS(N,ML,G,IG,B,IB,NB)
          CALL BPBS(N,ML,G,IG,B,IB,NB)
          GO TO 20
10        WRITE(IWRITE,11)
11        FORMAT(26H MATRIX TOO ILL-CONDITIONED)
20        CONTINUE
```

BPCE — LDL^T decomposition with condition estimation

Purpose: BPCE (Banded symmetric Positive definite matrix Condition Estimation) gives a lower bound for the condition number of a banded symmetric positive definite matrix A. It also returns the LDL^T decomposition of A and may be used in a linear equation package.

Usage: CALL BPCE (N, MU, G, IG, COND)

N	$\longrightarrow$	the order of the matrix A
MU	$\longrightarrow$	the number of nonzero bands on and above the diagonal of A
G	$\longrightarrow$	a matrix into which the upper triangular portion of the matrix A has been packed as follows: $G(j-i+1, i) = a_{ij}$ for $j \geq i$ (See the introduction to this chapter.) G should be dimensioned (IG,KG) in the calling program, where IG≥MU and KG≥N.
	$\longleftarrow$	LDL^T decomposition suitable for input into BPFS and BPBS (see **Note 2**)
IG	$\longrightarrow$	the row (leading) dimension of G, as dimensioned in the calling program
COND	$\longleftarrow$	an estimate of the condition number of A (see **Note 1**)

Note 1: The condition number measures the sensitivity of the solution of a linear system to errors in the matrix and in the right-hand side. If the elements of the matrix and the right-hand side(s) of your linear system have **d** decimal digits of precision, the solution might have as few as $\mathbf{d} - \log_{10}$ (COND) correct decimal digits. Thus if COND is greater than $10^{\mathrm{Bd}P}$, there may be no correct digits.

Note 2: The LDL^T decomposition of A satisfies the equation $A = LDL^T$ where L is lower unit triangular (1's on the diagonal, 0's above the diagonal) and D is diagonal. On return from BPCE, the diagonal of D occupies the first row of G and
$G(i-j+1,i) = l_{ij}$ for i>j.

Note 3: For complex Hermitian matrices ($A = A^*$, where A^* represents the conjugate transpose of A) , the complex version of this subroutine computes the LDL^* decomposition and returns the conjugate of L rather than L in G.

Error situations: *(The user can elect to ‘recover’ from those errors marked with an asterisk — see *Error Handling*, Framework Chapter)

Number	Error
1	N < 1
2	MU < 1
3	IG < MU
10 + k*	singular matrix whose rank is at least k
10 + N + k*	the k^{th} principal minor is not positive definite

Double-precision version: DBPCE with G and COND declared double precision

Complex Hermitian version: CBPCE with G declared complex (see **Note 3** above).

Storage: N real (double precision for DBPCE, complex for CBPCE) locations of scratch storage in the dynamic storage stack

Time: N × ((MU−1) × (10+MU/2)+8) additions
N × ((MU−1) × (6+MU/2)+4) multiplications
N × (MU+1) divisions

Method: Gaussian elimination without pivoting
See the reference below for the method used to estimate the condition number.
BPCE calls BPLD with EPS=0.0

See also: BPBS, BPDC, BPFS, BPLD, BPLE, BPSS

Authors: Doris Ryan and Linda Kaufman

Reference: Cline, A. K., Moler, C. B., Stewart, G. W., and Wilkinson, J. H., An estimate for the condition number, *SIAM J. Numer. Anal. 16* (1979), 368-375.

Example: In the following example we obtain estimates of the condition numbers of the matrix

$$\begin{matrix} 1+x & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & -1 & 2 & -1 & & \\ & & & \cdot & & & \\ & & & & \cdot & & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 1+x \end{matrix}$$

with x=1., .01, and .0001. The matrix is singular when x is 0, and becomes more ill-conditioned as x approaches 0.

In this example we compare the condition number $K = ||A||\ ||A^{-1}||$ with the estimate obtained from BPCE. In the code below A^{-1} is computed one column at a time and K is computed using the 1-norm. In the 1-norm, $||A||$ is the maximum column sum in absolute value, which is obviously 4.

```
        INTEGER N, MU, IG, K, I, IWRITE, I1MACH, J
        REAL G(2,100), B(200)
        REAL X, COND, AINVNO, AINORM, ABS
        N=100
        X=1.0
        MU=2
        IG=2
        DO 50 K=1,3
C CONSTRUCT MATRIX
            DO 10 I=1,N
               G(1,I)=2.0
               G(2,I)=-1.0
  10        CONTINUE
            G(1,1)=1.0+X
            G(1,N)=1.0+X
C GET ESTIMATE OF CONDITION NUMBER FROM BPCE
            CALL BPCE(N,MU,G,IG,COND)
            IWRITE=I1MACH(2)
            WRITE(IWRITE,11)X
  11        FORMAT(/10H WHEN X IS,E14.6)
            WRITE(IWRITE,12)COND
  12        FORMAT(25H CONDITION ESTIMATE IS   ,E15.8)
C SINCE CONDITION NUMBER IS NORM(A)*NORM(INVERSE(A)),
C FIND THE NORM OF EACH COLUMN OF INVERSE(A). GENERATE
C THE COLUMNS ONE AT A TIME AND REUSE SPACE
            AINVNO=0.0
            DO 40 I=1,N
C GENERATE ITH COLUMN OF IDENTITY MATRIX AS RIGHT HAND SIDE
                DO 20 J=1,N
                   B(J)=0.0
  20            CONTINUE
                B(I)=1.0
```

```
C SOLVE AX=B TO GET ITH COLUMN OF A(INVERSE)
                CALL BPFS(N,MU,G,IG,B,N,1)
                CALL BPBS(N,MU,G,IG,B,N,1)
C FIND NORM OF COLUMN
                AINORM=0.0
                DO 30 J=1,N
                   AINORM=AINORM+ABS(B(J))
  30            CONTINUE
                IF(AINVNO.LT.AINORM)AINVNO=AINORM
  40        CONTINUE
            COND=4.0*AINVNO
            WRITE(IWRITE,41)COND
  41        FORMAT(25H TRUE CONDITION NUMBER IS,E15.8)
            X=X/100.0
  50     CONTINUE
         STOP
         END
```

When the above code was executed on the Honeywell 6000 machine at Bell Laboratories, the following was printed:

```
WHEN X IS  0.100000E 01
CONDITION ESTIMATE IS    0.40807862E 04
TRUE CONDITION NUMBER IS 0.50999824E 04

WHEN X IS  0.100000E-01
CONDITION ESTIMATE IS    0.23329148E 05
TRUE CONDITION NUMBER IS 0.24899523E 05

WHEN X IS  0.100000E-03
CONDITION ESTIMATE IS    0.19933923E 07
TRUE CONDITION NUMBER IS 0.19950487E 07
```

The comparison above of the condition number estimated by BPCE with the true condition number indicates that the order of magnitude (which is all one usually is interested in) of the estimated condition number is correct. Note that the inverse of a band matrix is usually a full $n \times n$ matrix and should rarely be calculated.

BPDC — LDL^T decomposition of a band symmetric positive definite matrix A

Purpose: BPDC (Banded symmetric Positive definite matrix DeComposition) decomposes a banded symmetric positive definite matrix A into LDL^T where L is lower unit triangular (1's on the diagonal and 0's above the diagonal) and D is diagonal. It is called by BPLE as the first step of the solution of a banded symmetric positive definite linear system.

Usage: CALL BPDC (N, MU, G, IG)

N $\longrightarrow$ the number of equations

MU $\longrightarrow$ the number of nonzero bands on and above the diagonal of A

G $\longrightarrow$ a matrix into which the upper triangular portion of the matrix A has been packed as follows:

$$G(j-i+1, i) = a_{ij} \text{ for } j \geq i$$

(See the introduction to the chapter.)
G should be dimensioned (IG,KG) in the calling program, where IG≥MU and KG≥N.

$\longleftarrow$ LDL^T decomposition suitable for input into BPFS and BPBS (see **Note 1**)

IG $\longrightarrow$ the row (leading) dimension of G, as dimensioned in the calling program

Note 1: The LDL^T decomposition of A satisfies the equation $A = LDL^T$ where L is lower unit triangular (1's on the diagonal, 0's above the diagonal) and D is diagonal. On return from BPDC, the diagonal of D occupies the first row of G and $G(i-j+1, i) = l_{ij}$ for i>j.

Note 2: For complex Hermitian matrices (A = A*, where A* represents the conjugate transpose of A), the complex version of this subroutine computes the LDL* decomposition and returns the conjugate of L, rather than L, in G.

Error situations: *(The user can elect to ‘recover’ from those errors marked with an asterisk — see *Error Handling*, Framework Chapter)

Number	Error
1	N < 1
2	MU < 1
3	IG < MU
10 + k*	singular matrix whose rank is at least k
10 + N + k*	the k^{th} principal minor is not positive definite

Double-precision version: DBPDC, with G declared double precision.

Complex Hermitian version: CBPDC with G declared complex (see **Note 2**).

Storage: None

Time: N×(MU−1)× MU/2 + N×(2MU−1) additions
N×(MU−1)× MU/2 multiplications
N×(MU−1) divisions

Method: BPDC calls BPLD after setting EPS = $||A||\varepsilon$, where ε is machine precision, i.e. the value returned by R1MACH(4) (or, for double precision, by D1MACH(4)).

See also: BPBS, BPCE, BPFS, BPLD, BPLE, BPSS

Author: Linda Kaufman

Example: This example is designed to indicate the relative efficiency of BPDC, BPLD, and BPCE as a function of the width of the band. All three subroutines compute the same factorization, but the criterion for singularity is treated differently in each of the three. In all the subroutines the matrix is considered singular if for some i, $d_i \leq$ EPS. In BPLD the user provides EPS; in BPDC the subroutine computes EPS (see **Method**); in BPCE, 0.0 is used as EPS. (However, BPCE also provides an estimate of the condition number of the matrix.)

The matrix in the example is derived from the traditional 5-point approximation to the Laplace operator on the unit square. The N×N matrix A has the form

$$\begin{array}{cccccc}
C & -I & & & & \\
-I & C & -I & & & \\
 & -I & C & -I & & \\
 & & \cdots\cdots & & & \\
 & & & -I & C & -I \\
 & & & & -I & C
\end{array}$$

where I is the identity matrix and C is the matrix

$$\begin{array}{cccccc}
4 & -1 & & & & \\
-1 & 4 & -1 & & & \\
 & -1 & 4 & -1 & & \\
 & & & \cdots & & \\
 & & & -1 & 4 & -1 \\
 & & & & -1 & 4
\end{array}$$

In this problem we vary the sizes of the blocks C and I which changes the bandwidth. For a given blocksize, the number of blocks is varied which changes N.

The subroutine ILAPSZ is a timer on the Honeywell 6000 with about 1% accuracy. It counts in 1/64 milliseconds.

```
      INTEGER IG, MLM1, IWRITE, I1MACH, K, N, MU
      INTEGER NBLOK, KBLOK, KK, I, J, IT, ILAPSZ, IT2
      REAL G(17, 100), G2(17, 100), G3(17, 100)
      REAL COND
      IG=17
      MLM1=4
      IWRITE=I1MACH(2)
      DO 70  K=1,3
         DO 60 N=48,96,48
            MU=MLM1+1
            I=0
            NBLOK=N/MLM1
C
C SET UP THREE MATRICES FOR ELLIPTIC PDE IN 2 DIMENSION
C
            DO 30 KBLOK=1,NBLOK
               DO 20 KK=1,MLM1
                  I=I+1
                  G(1,I)=4.0
                  G(2,I)=-1.0
                  G(MU,I)=-1.0
                  DO 10 J=3,MLM1
                     G(J,I)=0.0
 10               CONTINUE
 20            CONTINUE
               G(2,I)=0.0
```

```
 30         CONTINUE
            DO 50 I=1,N
               DO 40 J=1,MU
                  G2(J,I)=G(J,I)
                  G3(J,I)=G(J,I)
 40            CONTINUE
 50         CONTINUE
            WRITE(IWRITE,51)N,MU
 51         FORMAT(/6H N IS ,I4,30H ,NUMBER OF UPPER DIAGONALS IS,I3)
C TIME DECOMPOSITION BY BPLD
            IT=ILAPSZ(0)
            CALL BPLD(N,MU,G,IG,0.0)
            IT=ILAPSZ(0)-IT
            WRITE(IWRITE,52)IT
 52         FORMAT(14H TIME FOR BPLD,I7)
C TIME DECOMPOSITION BY BPDC
            IT2=ILAPSZ(0)
            CALL BPDC(N,MU,G2,IG)
            IT2=ILAPSZ(0)-IT2
            WRITE(IWRITE,53)IT2
 53         FORMAT(14H TIME FOR BPDC,I7)
C TIME DECOMPOSITION BY BPCE
            IT3=ILAPSZ(0)
            CALL BPCE(N,MU,G3,IG,COND)
            IT3=ILAPSZ(0)-IT3
            WRITE(IWRITE,54)IT3
 54         FORMAT(14H TIME FOR BPCE,I7)
 60     CONTINUE
        MLM1=MLM1*2
 70  CONTINUE
     STOP
     END
```

When the above code was run on the Honeywell 6000 at Bell Laboratories with an optimizing compiler the following was printed

```
N IS   48 ,NUMBER OF UPPER DIAGONALS IS  5
TIME FOR BPLD    959
TIME FOR BPDC   1413
TIME FOR BPCE   3670

N IS   96 ,NUMBER OF UPPER DIAGONALS IS  5
TIME FOR BPLD   1868
TIME FOR BPDC   2673
TIME FOR BPCE   7965

N IS   48 ,NUMBER OF UPPER DIAGONALS IS  9
TIME FOR BPLD   2395
TIME FOR BPDC   3181
TIME FOR BPCE   6377

N IS   96 ,NUMBER OF UPPER DIAGONALS IS  9
TIME FOR BPLD   4854
```

```
TIME FOR BPDC   6450
TIME FOR BPCE  13170

N IS   48 ,NUMBER OF UPPER DIAGONALS IS 17
TIME FOR BPLD   7143
TIME FOR BPDC   8227
TIME FOR BPCE  12722

N IS   96 ,NUMBER OF UPPER DIAGONALS IS 17
TIME FOR BPLD  15001
TIME FOR BPDC  17160
TIME FOR BPCE  49809
```

As the example indicates, the cost of computing the condition estimate can be substantially greater than the cost of factoring the matrix when the width of the band is small. The ratio of BPCE to BPDC does not always decrease as N increases, as it does in the case of general linear systems, but depends rather on the scaling that is sometimes done by BPCE to prevent overflow during the calculation of the condition estimate. If no scaling is done, the ratio of the times for BPCE to BPDC remains constant, but for some examples most of the time is spent scaling and the time ratio increases as N increases.

BPFS — band symmetric lower (unit) triangular linear system solution

Purpose: BPFS (Banded symmetric Positive definite matrix Forward Solution) solves LX = B where L is a banded unit lower triangular matrix, (i.e. 1's on the diagonal, 0's above the diagonal). BPFS can be used for the forward solution phase of a band symmetric positive definite linear system. (It is used in this way by the routines BPSS and BPLE.)

Usage: CALL BPFS (N, ML, G, IG, B, IB, NB)

N $\longrightarrow$ the number of equations

ML $\longrightarrow$ the number of nonzero diagonals on and below the diagonal of L

G $\longrightarrow$ a matrix (which may contain the results obtained by the routines BPCE, BPDC, or BPLD) into which L is packed as follows:

$$G(1+i-j, i) = l_{ij} \text{ for } i>j$$

(See the introduction to this chapter.)
G should be dimensioned (IG,KG) in the calling program, where IG≥ML and KG≥N.

IG $\longrightarrow$ the row (leading) dimension of G, as dimensioned in the calling program

B $\longrightarrow$ the matrix of right-hand sides, dimensioned (IB, KB) in the calling program, where IB≥N and KB≥NB.

$\longleftarrow$ the solution X

IB $\longrightarrow$ the row (leading) dimension of B, as dimensioned in the calling program

NB $\longrightarrow$ the number of right-hand sides

Note 1: BPFS and BPBS can be used directly on the output matrix produced by BPDC, BPLD, or BPCE to solve a general linear system.

Note 2: Users who have to solve a sequence of problems with the same coefficient matrix, but different right-hand sides, *not all known in advance*, should not call BPSS or BPLE repeatedly, but should use BPDC to find the LDL^T decompostion of the coefficient matrix, which can then be used repeatedly (and efficiently) for the sequence of forward solutions (using BPFS) and back solutions (using BPBS) for each set of right-hand sides.

Error situations: (All errors in this subprogram are fatal — see *Error Handling*, Framework Chapter)

Number	Error
1	N < 1
2	ML < 1
3	IG < ML
4	IB < N
5	NB < 1

Double-precision version: DBPFS, with G and B declared double precision

Complex Hermitian version: CBPFS with G and B declared complex
G should contain the conjugate of L

Storage: None

Time: $N\times(ML-1)\times NB$ additions
$N\times(ML-1)\times NB$ multiplications

See also: BPBS, BPCE, BPDC, BPLD, BPLE, BPSS

Author: Linda Kaufman

Example: In the following example we solve three problems that might arise from a discretization of a 1-dimensional differential equation. The coefficient matrix in each problem has the form

$$\begin{matrix} 1+x & -1 & & & & & & \\ -1 & 2 & -1 & & & & & \\ & -1 & 2 & -1 & & & & \\ & & -1 & 2 & -1 & & & \\ & & & . & & & & \\ & & & & . & & & \\ & & & & & -1 & 2 & -1 \\ & & & & & & -1 & 1+x \end{matrix}$$

with x=1., .01, and .0001 defining the three problems. To make it easy to detect errors in the solution, the right-hand side has been chosen to make the solution a vector of all 1's.

The program below is an encoding of an iterative refinement algorithm which may be used to obtain a highly accurate solution to a system of linear equations with an ill-conditioned matrix. When the condition number is not excessively high, the iterative refinement algorithm usually returns a solution that is accurate to the working precision of the machine. The iterative refinement algorithm is essentially:

(1) Solve $Ax = b$
(2) Set tol = $\varepsilon \sum |x_i|$
where ε is the precision of the machine
(3) Compute in double precision the residual, $Ax-b$, and put it in the real vector r.
(4) Solve $A\,\delta x = r$
(5) Compute norm = $\sum |\delta x_i|$
(6) Set x to $x + \delta x$
(7) If norm $\le$ tol stop, else return to step 3

In our code, step (1) is accomplished using the the linear equation solver for band positive definite matrices, BPLE. The subroutine BPLE leaves in the G matrix a decomposition which may be used by BPFS and BPBS to solve problems with the same coefficient matrix but different right-hand sides as in step(4). Since it is possible that the matrix will be so ill-conditioned that the iterative refinement algorithm will diverge, steps (3) through (7) in our code are performed only a finite number of times. This number is chosen to be an upper bound on number of of bits in the mantissa of the floating-point number supported by the machine.

This algorithm is not included in PORT because for double-precision matrices part of the computation would have to be done in extended precision.

The following program has been specifically tailored to the three problems.

```
       INTEGER N, ML, IG, NM1, K, I, IWRITE, I1MACH, IT, IEND, ITER
       REAL G(2,100), B(200), R(200)
       REAL X, ERR, AMAX1, RNORM, BNORM, R1MACH, ABS
       DOUBLE PRECISION DBLE
C CONSTRUCT MATRIX AND RIGHT HAND SIDE SO TRUE SOLUTION IS
C COMPOSED ENTIRELY OF 1S
       N=100
       X=1
       ML=2
       IG=2
       NM1=N-1
       DO 90 K=1,3
          DO 10 I=1,N
             G(1,I)=2.0
             G(2,I)=-1.0
             B(I)=0.0
  10      CONTINUE
          G(1,1)=1.0+X
          G(1,N)=1.0+X
          B(1)=X
          B(N)=X
C SOLVE THE SYSTEM
          CALL BPLE(N,ML,G,IG,B,N,1)
          IWRITE=I1MACH(2)
          WRITE(IWRITE,11)X
  11      FORMAT(/5H X IS,F16.8)
C COMPUTE THE ERROR
          ERR=0.0
          DO 20 I=1,N
             ERR=AMAX1(ERR,ABS(B(I)-1.0))
  20      CONTINUE
          WRITE(IWRITE,21)ERR
  21      FORMAT(22H FOR BPLE THE ERROR IS,F16.8)
          IEND=I1MACH(11)*IFIX(R1MACH(5)/ALOG10(2.0)+1.0)
C FIND THE NORM OF THE SOLUTION
          BNORM=0.0
          DO 30 I=1,N
              BNORM=AMAX1(BNORM,ABS(B(I)))
  30      CONTINUE
C REFINE THE SOLUTION
          DO 60 ITER=1,IEND
             IT=ITER
C COMPUTE THE RESIDUAL R=B-AX, IN DOUBLE PRECISION
             DO 40 I=2,NM1
                R(I)=DBLE(B(I-1))+DBLE(B(I+1))-2.D0*DBLE(B(I))
  40         CONTINUE
             R(1)=X+B(2)-DBLE(1.0+X)*DBLE(B(1))
             R(N)=X+B(N-1)-DBLE(1.+X)*DBLE(B(N))
C SOLVE A(DELTAX)=R
             CALL BPFS(N,ML,G,IG,R,N,1)
             CALL BPBS(N,ML,G,IG,R,N,1)
C DETERMINE NORM OF CORRECTION AND ADD IN CORRECTION
             RNORM=0.0
```

```
            DO 50 I=1,N
               B(I)=B(I)+R(I)
               RNORM=RNORM+ABS(R(I))
 50         CONTINUE
            IF(RNORM.LT.R1MACH(4)*BNORM) GO TO 70
 60      CONTINUE
         WRITE(IWRITE,61)
 61      FORMAT(18H REFINEMENT FAILED)
C COMPUTE NEW ERROR
 70      ERR=0.0
         DO 80 I=1,N
            ERR=AMAX1(ERR,ABS(B(I)-1.0))
 80      CONTINUE
         WRITE(IWRITE,81)IT,ERR
 81      FORMAT(24H ERROR AFTER REFINEMENT ,I4,3H IS,E14.7)
         X=X/100.0
 90   CONTINUE
      STOP
      END
```

When the above program was executed on the Honeywell 6000 at Bell Laboratories, the following was printed

```
X IS      1.00000000
FOR BPLE THE ERROR IS      0.00000431
ERROR AFTER REFINEMENT    2 IS 0.

X IS      0.01000000
FOR BPLE THE ERROR IS      0.00002055
ERROR AFTER REFINEMENT    3 IS 0.

X IS      0.00010000
FOR BPLE THE ERROR IS      0.00491761
ERROR AFTER REFINEMENT    4 IS 0.
```

This problem was chosen because as x approaches 0, the matrix becomes more ill-conditioned, the error in the solution grows, and more steps of iterative refinement are required. The reader should be aware that in this example we were able to represent the matrix and the right-hand side precisely, but due to roundoff error this is not always the case. Often the iterative refinement algorithm produces an exact, but useless, solution to a slightly incorrect problem.

BPLD — LDL^T decomposition of a band symmetric positive definite matrix

Purpose: BPLD (Band Positive definite LDL^T decomposition) decomposes a banded symmetric positive definite matrix A into LDL^T where L is lower triangular and D is diagonal. It allows the user to provide a threshold for considering a matrix singular BPLD is called by the decomposition routines BPCE and BPDC.

Usage: CALL BPLD (N, MU, G, IG, EPS)

N → the number of equations

MU → the number of nonzero bands on and above the diagonal of A

G → a matrix into which the upper triangular portion of the matrix A has been packed as follows:

$$G(j-i+1, i) = a_{ij} \text{ for } j \geq i$$

(See the introduction to this chapter.)
G should be dimensioned (IG,KG) in the calling program, where IG≥MU and KG≥N.

← the LDL^T decomposition suitable for input into BPFS and BPBS (see **Note 2**)

IG → the row (leading) dimension of G, as dimensioned in the calling program

EPS → if there exists an index k such that $|d_{kk}| \leq$ EPS then A is considered singular

Note 1: After the execution of BPLD, (if the matrix has not been found singular), the determinant of A is the product of the elements of the first row of G.

Note 2: The LDL^T decomposition of A satisfies the equation A = LDL^T where L is lower unit triangular (1’s on the diagonal, 0’s above the diagonal) and D is diagonal. On return from BPLD, the diagonal of D occupies the first row of G and G(i−j+1,i) = l_{ij} for i>j.

Note 3: For complex Hermitian matrices (A = A*, where A* represents the conjugate transpose of A), the complex version of this subroutine computes the LDL* decomposition and returns the conjugate of L rather than L in G.

Error situations: *(The user can elect to 'recover' from those errors marked with an asterisk — see *Error Handling*, Framework Chapter)

Number	Error
1	N < 1
2	MU < 1
3	IG < MU
10 + k*	singular matrix whose rank is at least k
10 + N + k*	the k^{th} principal minor is not positive definite

Double-precision version: DBPLD with G and EPS declared double precision

Complex Hermitian version: CBPLD with G declared complex (see **Note 3** above)

Storage: None

Timing: N×(MU−1)×MU/2 additions
N×(MU−1)×MU/2 multiplications
(MU−1)×N divisions

Method: Gaussian elimination without pivoting

See also: BPBS, BPCE, BPDC, BPFS, BPLE, BPSS

Author: Linda Kaufman

Example: As noted above, after execution of BPLD, the determinant of the matrix stored in G is the product of the elements stored in the first row of G. The subroutine computes this product taking care to avoid underflow and overflow. The subroutine UMKFL is used to decompose a floating-point number, F, into a mantissa, M, and an exponent E such that $F = Mb^E$ where b is the base of the machine and $1/b \le |M| \le 1$

```
      SUBROUTINE BPDET(N,MU,G,IG,DETMAN,IDETEX)
C
C THIS SUBROUTINE COMPUTES THE DETERMINANT OF A
C BAND SYMMETRIC POSITIVE DEFINITE MATRIX STORED IN G.
C IT IS GIVEN BY DETMAN*BETA**IDETEX
C WHERE BETA IS THE BASE OF THE MACHINE
C AND DETMAN IS BETWEEN 1/BETA AND 1
C
      REAL G(IG,N),DETMAN
      REAL ONOVBE,M
      INTEGER E
      INTEGER IDETEX
      CALL BPLD(N,MU,G,IG,0.0)
C
C THE DETERMINANT IS THE PRODUCT OF THE ELEMENTS OF ROW 1 OF G
C WE TRY TO COMPUTE THIS PRODUCT IN A WAY THAT WILL
C AVOID UNDERFLOW AND OVERFLOW
C
      ONOVBE=1.0/FLOAT(I1MACH(10))
      DETMAN=ONOVBE
      BETA=FLOAT(I1MACH(10))
      IDETEX=1
      DO 10 I=1,N
          CALL UMKFL(G(1,I),E,M)
          DETMAN=DETMAN*M
          IDETEX=IDETEX+E
          IF(DETMAN.GE.ONOVBE) GO TO 10
             IDETEX=IDETEX-1
             DETMAN=DETMAN*BETA
 10   CONTINUE
      RETURN
      END
```

BPLE — band symmetric positive definite linear system solution

Purpose: BPLE (Banded symmetric Positive definite Linear Equation solution) solves the system AX = B where A is a banded symmetric positive definite matrix

Usage: CALL BPLE (N, MU, G, IG, B, IB, NB)

N $\longrightarrow$ the number of equations

MU $\longrightarrow$ the number of nonzero bands on and below the diagonal of A

G $\longrightarrow$ a matrix into which the upper triangular portion of the matrix A has been packed as follows:

$$G(j-i+1,i) = a_{ij} \text{ for } j \geq i$$

(See the introduction to this chapter.)
G should be dimensioned (IG,KG) in the calling program, where IG≥MU and KG≥N.

$\longleftarrow$ L and D from the factorization of A into LDL^T (see **Note 3**)

IG $\longrightarrow$ the row (leading) dimension of G, as dimensioned in the calling program

B $\longrightarrow$ the matrix of right-hand sides, dimensioned (IB, KB) in the calling program, where IB≥N and KB≥NB.

$\longleftarrow$ the solution X

IB $\longrightarrow$ the row (leading) dimension of B, as dimensioned in the calling program

NB $\longrightarrow$ the number of right-hand sides

Note 1: Unless the given matrix A is known in advance to be well-conditioned, the user should use BPSS instead of BPLE.

Note 2: Users who wish to solve a sequence of problems with the same coefficient matrix, but different right-hand sides *not all known in advance*, should call subprograms BPLE, BPFS and BPBS. (See the example in BPFS.) BPLE is called once to get the LDL^T decomposition (see the introduction to this chapter) and to solve for the first solution and then the pair, BPFS (forward solve) and BPBS (back solve), is called for each additional right-hand side.

Note 3: The LDL^T decomposition of A satisfies the equation $A = LDL^T$ where L is lower unit triangular (1's on the diagonal, 0's above the diagonal) and D is diagonal. On return from BPLE, the diagonal of D occupies the first row of G and $G(i-j+1,i) = l_{ij}$ for i>j.

Note 4: For complex Hermitian matrices (A = A*, where A* represents the conjugate transpose of A), the complex version of this subroutine computes the LDL* decomposition and returns the conjugate of L rather than L in G.

Error situations: *(The user can elect to 'recover' from those errors marked with an asterisk — see *Error Handling*, Framework Chapter)

Number	Error
1	N < 1
2	MU < 1
3	IG < MU
4	IB < N
5	NB < 1
10 + k*	singular matrix whose rank is at least k
10 + N + k*	k^{th} principal minor is not positive definite

Double-precision version: DBPLE with G and B declared double precision

Complex Hermitian version: CBPLE with G and B declared complex (see **Note 4**).

Storage: None

Time: N×((MU−1)×(MU/2+2×(NB+1))+1) additions
N×(MU−1)×(MU/2+2×NB) multiplications
N×(MU−1+NB) divisions

Method: BPLE calls BPDC to form the LDL^T decomposition, and then calls BPFS for the forward solution, and BPBS for the back solution.

See also: BPBS, BPCE, BPDC, BPFS, BPLD, BPSS

Author: Linda Kaufman

Example: The matrix in this example is derived from the usual five-point approximation to the Laplace operator on the unit square with an 11×11 mesh. The 100×100 matrix A has the form

$$\begin{matrix} C & -I & & & & \\ -I & C & -I & & & \\ & -I & C & -I & & \\ & & \cdots & & & \\ & & & -I & C & -I \\ & & & & -I & C \end{matrix}$$

where I is the identity matrix of order 10 and C is the matrix

$$\begin{matrix} 4 & -1 & & & & \\ -1 & 4 & -1 & & & \\ & -1 & 4 & -1 & & \\ & & & \cdots & & \\ & & & -1 & 4 & -1 \\ & & & & -1 & 4 \end{matrix}$$

To make it easy to detect errors in the example, the right-hand side has been chosen to make the solution a vector of all 1's. To construct the right-hand side the subroutine BPML is used which produces b=Ax where A is a band symmetric positive definite matrix packed into the matrix G.

```
      INTEGER IG, N, MU, MLM1, I, KBLOK, KK, J
      INTEGER IWRITE, I1MACH
      REAL G(11,100), B(100), X(100)
      REAL ERR, AMAX1
      IG=11
      N=100
      MU=11
C
C SET UP MATRIX FOR ELLIPTIC PDE IN 2 DIMENSIONS
C
      MLM1=MU-1
      I=0
      DO 30 KBLOK=1,MLM1
         DO 20 KK=1,MLM1
            I=I+1
            G(1,I)=4.0
            G(2,I)=-1.0
            DO 10 J=3,MLM1
               G(J,I)=0.0
   10       CONTINUE
```

```
                 G(MU,I)=-1.0
   20         CONTINUE
              G(2,I)=0.0
   30      CONTINUE
C
C SET UP RIGHT HAND SIDE SO SOLUTION IS ALL 1'S
C
           DO 40 I=1,N
              X(I)=1.0
  40       CONTINUE
           CALL BPML(N,MU,G,IG,X,B)
C
C SOLVE THE SYSTEM
C
           CALL BPLE(N,MU,G,IG,B,100,1)
C
C COMPUTE THE ERROR
C
           ERR=0.0
           DO 50 I=1,N
              ERR=AMAX1(ERR,ABS(B(I)-1.0))
  50       CONTINUE
           IWRITE=I1MACH(2)
           WRITE(IWRITE,51)ERR
  51       FORMAT(31H ERROR IN SOLUTION FROM BPLE IS,F15.8)
           STOP
           END
```

When the above code was run on the Honeywell 6000 machine at Bell Laboratories, the following was printed:

```
 ERROR IN SOLUTION FROM BPLE IS     0.00000012
```

BPML — banded positive definite matrix - vector multiplication

Purpose: BPML (Banded Positive definite matrix MuLtiplication) forms the product *Ax* where *A* is a symmetric banded positive matrix stored in packed form.

Usage: CALL BPML (N, MU, G, IG, X, B)

N $\longrightarrow$ the length of *x*

MU $\longrightarrow$ the number of nonzero bands on and above the diagonal of A

G $\longrightarrow$ a matrix into which the upper triangular portion of the matrix A has been packed as follows:

$$G(1 + j-i,i) = a_{ij} \text{ for } j \geq i.$$

(See the introduction to this chapter.) G should be dimensioned (IG, KG) in the calling program, where IG≥MU and KG≥N.

IG $\longrightarrow$ the row (leading) dimension of G, as dimensioned in the calling program

X $\longrightarrow$ the vector *x* to be multiplied

B $\longleftarrow$ the vector A*x*

Error situations: (All errors in this subprogram are fatal — see *Error Handling*, Framework Chapter)

Number	Error
1	N < 1
2	MU < 1
3	IG < MU

Double-precision version: DBPML with G, X, and B declared double precision.

Complex Hermitian version: CBPML with G, X, and B declared complex

Storage: None

Time: (2MU–1)×N additions
(2MU–1)×N multiplications

See also: BPBS, BPCE, BPDC, BPLU, BPLE, BPSS

Author: Linda Kaufman

Example: This example checks the consistency of BPML and BPSS the banded system solver. First the example uses BPML to compute for a given vector x and a given matrix A the vector $b = Ax$. Then the problem is inverted, i.e., BPSS is used to find the vector x which satisfies $Ax = b$. This x is then compared with the original vector. The vector x is generated randomly and the 10×10 matrix A is given by

$$\begin{array}{cccccccc}
4 & -1 & -1 & & & & & \\
-1 & 4 & -1 & -1 & & & & \\
-1 & -1 & 4 & -1 & -1 & & & \\
 & -1 & -1 & 4 & -1 & -1 & & \\
 & & . & . & . & & & \\
 & & & -1 & -1 & 4 & -1 & -1 \\
 & & & & -1 & -1 & 4 & -1 \\
 & & & & & -1 & -1 & 4
\end{array}$$

```
      INTEGER IG, N, MU, I, IWRITE, I1MACH
      REAL G(3,20), X(20), B(20)
      REAL UNI, ERR, COND, SASUM, ABS
      IG=3
      N=10
      MU=3
C
C CONSTRUCT MATRIX A AND PACK IT INTO G
C
       DO 10 I=1,N
          G(1,I)=4.0
          G(2,I)=-1.0
          G(3,I)=-1.0
  10   CONTINUE
C
C CONSTRUCT A RANDOM VECTOR
C
       DO 20 I=1,N
          X(I)=UNI(0)
  20   CONTINUE
C
C CONSTRUCT B=AX
```

```
C
          CALL BPML(N,MU,G,IG,X,B)
C
C SOLVE THE SYSTEM AX=B
C
          CALL BPSS(N,MU,G,IG,B,N,1,COND)
C
C PRINT OUT THE TRUE SOLUTION AND THE COMPUTED SOLUTION
C
          IWRITE=I1MACH(2)
          WRITE(IWRITE,21)
  21      FORMAT(34H TRUE SOLUTION   COMPUTED SOLUTION)
          WRITE(IWRITE,22)(X(I),B(I),I=1,N)
  22      FORMAT(1H ,2E16.8)
          ERR=0.0
          DO 30 I=1,N
              ERR=ERR+ABS(B(I)-X(I))
  30      CONTINUE
          ERR=ERR/SASUM(N,X,1)
          WRITE(IWRITE,31)ERR
  31      FORMAT(19H RELATIVE ERROR IS ,1PE15.7)
          WRITE(IWRITE,32)COND
  32      FORMAT(20H CONDITION NUMBER IS,1PE15.7)
          STOP
          END
```

When the above program was executed on the Honeywell 6000 machine at Bell Laboratories, which has a machine precision of $1.\times 10^{-8}$, the following was printed:

```
TRUE SOLUTION   COMPUTED SOLUTION
  0.22925607E 00  0.22925608E 00
  0.76687502E 00  0.76687504E 00
  0.68317685E 00  0.68317687E 00
  0.50919111E 00  0.50919112E 00
  0.87455959E 00  0.87455962E 00
  0.64464101E 00  0.64464103E 00
  0.84746840E 00  0.84746842E 00
  0.35396343E 00  0.35396345E 00
  0.39889160E 00  0.39889160E 00
  0.45709422E 00  0.45709422E 00
RELATIVE ERROR IS   3.1985797E−08
CONDITION NUMBER IS  2.1901962E 01
```

The condition number of the matrix and the precision of the Honeywell suggest that even in the absence of roundoff error in BPML, a relative error of 2.2×10^{-7} would not be surprising. The value computed above is quite reasonable.

BPNM — norm of a banded symmetric positive definite matrix

Purpose: BPNM (Banded Positive definite matrix NorM) computes the norm of a banded symmetric positive definite matrix A stored in packed form. The infinity norm is defined as $\max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$

Type: Real function

Usage: <answer> = BPNM (N, MU, G, IG)

N $\longrightarrow$ the number of rows in A

MU $\longrightarrow$ the number of nonzero bands in A on and above the diagonal

G $\longrightarrow$ a matrix into which the upper triangular portion of the matrix A has been packed as follows:

$$G(1 + j-i, i) = a_{ij} \text{ for } j \ge i$$

i.e. the main diagonal of A is in the first row of G
(See the introduction to this chapter.)
G should be dimensioned (IG,KG) in the calling program, where IG≥MU and KG≥N.

IG $\longrightarrow$ the row (leading) dimension of G, as dimensioned in the calling program

<answer> $\longleftarrow$ $\max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$

Error situations: (All errors in this subprogram are fatal — see *Error Handling*, Framework Chapter)

Number	Error
1	N < 1
2	MU < 1
3	IG < MU

Double-precision version: DBPNM with G and DBPNM declared double precision

Complex version: CBPNM with G declared complex

Storage: None

Time: $N \times (2 \times MU - 1)$ additions

See also: BPDC, BPLD, BPLE, BPSS, BPCE

Author: Linda Kaufman

Example: Because of roundoff error it is often very difficult to decide whether a matrix is singular. One criteria often used for symmetric positive definite matrices is to compute the LDL^T decomposition of the matrix and declare the matrix singular if any element of the diagonal matrix D is less than $\varepsilon \| A \|$ where ε is the machine precision and is computed by R1MACH(4).

The following program fragment might be used to indicate whether the symmetric positive definite banded matrix packed into G is singular or nearly singular. It uses the fact that the subroutine BPLD, which computes the LDL^T decomposition of a banded symmetric matrix, issues a recoverable error when it detects an element of D less than EPS, an input parameter to the subroutine.

```
         IWRITE=I1MACH(2)
         CALL ENTSRC(IROLD,1)
         EPS=BPNM(N,MU,G,IG)*R1MACH(4)
         CALL BPLD(N,MU,G,IG,EPS)
         IF (NERROR(IERR).EQ.0) GO TO 10
            CALL ERROFF
            WRITE(IWRITE,1)
1           FORMAT(16H SINGULAR MATRIX)
10       CONTINUE
```

BPSS — band positive definite linear system solution with condition estimation

Purpose: BPSS (Band Positive definite System Solution) solves the system AX = B where A is a band symmetric positive definite matrix. An estimate of the condition number of A is provided.

Usage: CALL BPSS (N, MU, G, IG, B, IB, NB, COND)

N $\longrightarrow$ the number of equations

MU $\longrightarrow$ the number of nonzero bands on and above the diagonal of A

G $\longrightarrow$ a matrix into which the upper triangular portion of the matrix A has been placed as follows:

$$G(j-i+1, i) = a_{ij} \text{ for } j \geq i$$

(See the introduction to this chapter.)
G should be dimensioned (IG,KG) in the calling program, where IG≥MU and KG≥N.

$\longleftarrow$ L and D from the factorization of A into LDL^T
(see **Note 3**)

IG $\longrightarrow$ the row (leading) dimension of G, as dimensioned in the calling program

B $\longrightarrow$ the matrix of right-hand sides, dimensioned (IB, KB) in the calling program, where IB≥N and KB≥NB.

$\longleftarrow$ the solution X

IB $\longrightarrow$ the row (leading) dimension of B, as dimensioned in the calling program

NB $\longrightarrow$ the number of right-hand sides

COND $\longleftarrow$ an estimate of the condition number of A (See **Note 1**)

Note 1: The condition number measures the sensitivity of the solution of a linear system to errors in the matrix and in the right-hand side. If the elements of the matrix and the right-hand side(s) of your linear system have **d** decimal digits of precision, the solution might have as few as **d** – $\log_{10}$(COND) correct decimal digits. Thus if COND is greater than $10^{\mathrm{Bd}\,P}$, there may be no correct digits.

If the given matrix, A, is known in advance to be well-conditioned, then the user may wish to use the routine BPLE, which is a little faster than BPSS. Ordinarily, however, the user is strongly urged to choose BPSS, and to follow it by a test of the condition estimate.

Note 2: Users who wish to solve a sequence of problems with the same coefficient matrix, but different right-hand sides *not all known in advance*, should not use BPSS, but should call subprograms BPCE, BPFS and BPBS. (See the example of BPFS.) BPCE is called once to get the LDL^T decomposition (see the introduction to this chapter) and then the pair, BPFS (forward solve) and BPBS (back solve), is called for each new right-hand side.

Note 3: The LDL^T decomposition of A satisfies the equation $A = LDL^T$ where L is lower unit triangular (1's on the diagonal, 0's above the diagonal) and D is diagonal. On return from BPSS, the diagonal of D occupies the first row of G and $G(i-j+1,i) = l_{ij}$ for i>j.

Note 4: For complex Hermitian matrices (A = A*, where A* represents the conjugate transpose of A), the complex version of this subroutine computes the LDL* decomposition and returns the conjugate of L rather than L in G.

Error situations: *(The user can elect to 'recover' from those errors marked with an asterisk — see *Error Handling*, Framework Chapter)

Number	Error
1	N < 1
2	MU < 1
3	IG < MU
4	IB < N
5	NB < 1
10 + k*	singular matrix whose rank is at least k
10 + N + k*	k^{th} principal minor is not positive definite

Double-precision version: DBPSS, with G, B, and COND declared double precision.

Complex Hermitian version: CBPSS with G and B declared complex (see **Note 4** above)

Storage: N real (double precision for DBPSS, complex for CBPSS) locations of scratch storage in the dynamic storage stack

Time: N×((MU−1)×(MU/2+2×NB+8)+8) additions
N×((MU−1)×(MU/2+2×NB+6)+4) multiplications
N×(MU+1+NB) divisions

Method: BPSS calls BPCE to form the LDL^T decomposition, and then calls BPFS for the forward solution, and BPBS for the back solution. See the reference below for the method used to estimate the condition number.

See also: BPBS, BPCE, BPDC, BPFS, BPLD, BPLE

Author: Linda Kaufman

Reference: Cline, A. K., Moler, C. B., Stewart, G. W., and Wilkinson, J. H., An estimate for the condition number, *SIAM J. Numer. Anal. 16* (1979), 368-375.

Example: In the following example we solve three problems that might arise from a discretization of a 1-dimensional differential equation. The coefficient matrix in each problem has the form

$$
\begin{array}{ccccccc}
1+x & -1 & & & & & \\
-1 & 2 & -1 & & & & \\
 & -1 & 2 & -1 & & & \\
 & & -1 & 2 & -1 & & \\
 & & & \cdot & & & \\
 & & & & \cdot & & \\
 & & & & -1 & 2 & -1 \\
 & & & & & -1 & 1+x
\end{array}
$$

with x=1.0, .01, and .0001 defining the three problems. The matrix is singular when x is 0, and, as our output suggests, the matrix becomes more ill-conditioned as x approaches 0. To make it easy to detect errors in the solution, the right-hand side has been chosen to make the solution a vector of all 1's. The reader should notice in the output the correlation between the condition number and the error. As with most problems with nearly constant diagonal, it was very easy to write the code to set up the problem for BPSS.

```
      INTEGER N, K, I, IWRITE, I1MACH, MU
      REAL G(2,100), B(200)
      REAL X, COND, ERR, AMAX1
C CONSTRUCT MATRIX AND RIGHT-HAND SIDE SO TRUE SOLUTION IS
```

```
C COMPOSED ENTIRELY OF ONES
       N=100
       X=1
       DO 30  K=1,3
          DO 10 I=1,N
             G(1,I)=2.0
             G(2,I)=-1.0
             B(I)=0.0
  10      CONTINUE
          G(1,1)=1.0+X
          G(1,N)=1.0+X
          B(1)=X
          B(N)=X
C SOLVE THE SYSTEM
          MU=2
          CALL BPSS(N,MU,G,2,B,N,1,COND)
          IWRITE=I1MACH(2)
          WRITE(IWRITE,11)X
  11      FORMAT(/5H X IS,F15.7)
          WRITE(IWRITE,12)COND
  12      FORMAT(20H CONDITION NUMBER IS,1PE15.7)
C COMPUTE THE ERROR
          ERR=0.0
          DO 20 I=1,N
             ERR=AMAX1(ERR,ABS(B(I)-1.0))
  20      CONTINUE
          WRITE(IWRITE,21)ERR
  21      FORMAT(22H FOR BPSS THE ERROR IS,F16.8)
          X=X/100.
  30   CONTINUE
       STOP
       END
```

When the above program was executed on the Honeywell 6000 machine at Bell Laboratories, the following was printed

```
X IS      1.0000000
CONDITION NUMBER IS  4.0807862E 03
FOR BPSS THE ERROR IS      0.00000431

X IS      0.0100000
CONDITION NUMBER IS  2.3329148E 04
FOR BPSS THE ERROR IS      0.00002055

X IS      0.0001000
CONDITION NUMBER IS  1.9933923E 06
FOR BPSS THE ERROR IS      0.00491761
```