Appendix 3

BANDED MATRICES

BABS	-	Back Solve
BACE	-	Condition Estimation
BADC	-	DeComposition
BAFS	-	Forward Solve
BALE	-	Linear Equation solution
BALU	-	LU decomposition
BAML	-	MuLtiplication
BANM	-	NorM
BASS	-	System Solution

Purpose: BABS (BAnded matrix Back Solution) solves AX = B where A is a banded upper triangular matrix. It can be used for the back solution phase of a banded linear system solution. (It is used in this way by the routines BASS and BALE.)

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

N $\longrightarrow$ the number of equations

G $\longrightarrow$ a matrix (which may have been created by the routines BACE, BADC, or BALU) into which A has been packed as follows:

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

i.e. the diagonal is 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

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

MU $\longrightarrow$ the number of nonzero bands in A

Notes: BAFS and BABS can be used directly on the output matrix produced by BADC, BALU, or BACE to solve a general 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	IG < MU
3	IB < N
4	NB < 1
5	MU < 1
10 + k*	singular matrix with 0.0 in the kth position on the diagonal

Double-precision version: DBABS with G and B declared double precision.

Complex version: CBABS with G and B declared complex

Storage: None

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

See also: BAFS, BADC, BALU, BACE, BASS, BALE

Author: Linda Kaufman

Reference: Martin, R. S., and Wilkinson, J. H., Solution of Symmetric and Unsymmetric Band Equations and the Calculation of Eigenvectors of Band Matrices, *Numer. Math. 9* (1967) 279-301.

Example: In this example we present a subroutine which implements the inverse iteration for finding the eigenvector x corresponding to a specified real eigenvalue λ of a banded matrix A. The algorithm is essentially

Determine x_0, an initial approximation to the eigenvector
Until convergence
Solve $(A-\lambda I)x_{k+1} = \alpha_k \; x_k$

where $1/\alpha_k$ is the element of x_k of maximum modulus.

Given the LU decomposition of $A-\lambda I$, the starting vector x_0 is usually set to U^{-1}e, where e is the vector whose elements are all unity. As the above reference indicates, a suitable stopping criteria is $||A|| \; \varepsilon \geq ||(\alpha_{k-1}x_{k-1} - \beta_k x_k)\alpha_k||_\infty$

where ε is the machine precision given by R1MACH(4) and $1/\beta_k$ is the element of x_k in the same position as the unit element of $\alpha_{k-1}x_{k-1}$.

In the subroutine EIGVEC below, BABS is referenced twice, once to obtain the initial approximation and once within the iterative loop. If λ is an eigenvalue of A, $A-\lambda I$ is theoretically a singular matrix and hence when BALU is invoked, one would expect the subroutine to terminate with a recoverable error. Thus after calling BALU the error flag is turned off if necessary. If BALU determines that a diagonal element of the U matrix is not greater than EPS in magnitude, that diagonal element is set to EPS and the computation continues. Although the computed vectors x_k will not be necessarily close to the true solutions of the linear systems, after these vectors are scaled, they will approach an eigenvector of the matrix.

```
        SUBROUTINE EIGVEC(N,M,ML,G,IG,EVAL,EVEC,LIMIT)
C
C GIVEN A BANDED MATRIX PACKED INTO G WITH
C N ROWS, M NONZERO DIAGONALS AND ML NONZERO DIAGONALS
C ON AND BELOW THE DIAGONAL AND GIVEN AN EIGENVALUE OF THE
C MATRIX IN EVAL, THIS SUBROUTINE USES INVERSE ITERATION TO
C DETERMINE THE CORRESPONDING EIGENVECTOR AND RETURNS IT
C IN EVEC.
C LIMIT IS A BOUND ON THE NUMBER OF ITERATIONS
C
       INTEGER N, M, ML, IG, LIMIT
       INTEGER I, JAL, ISTKGT, JINTER, JX, MU, IERR, NERROR
       INTEGER LIM, JJ, ISAMAX, JXI, IST(1000)
       REAL G(IG, N), EVEC(N), EVAL
       REAL BANM, SIZE, R1MACH, EPS, SC, BET, D1, SC2, ABS
       REAL R(1000)
       DOUBLE PRECISION D(500)
       COMMON /CSTAK/ D
       EQUIVALENCE (D(1),IST(1)),(R(1),D(1))
       CALL ENTER(1)
C DETERMINE ITERATION TOLERANCE
       CALL BANM(N,ML,M,G,IG,SIZE)
       EPS=SIZE*R1MACH(4)
C SUBTRACT EIGENVALUE FROM DIAGONAL OF G
       DO 10 I=1,N
          G(ML,I)=G(ML,I) - EVAL
  10   CONTINUE
C GET SPACE FROM STACK FOR AL,INTER, AND SCRATCH VECTOR
       JAL =ISTKGT(N*(ML-1),3)
       JINTER=ISTKGT(N,2)
       JX=ISTKGT(N,3)
C GET LU DECOMPOSITION OF MATRIX
       CALL BALU(N,ML,M,G,IG,R(JAL),ML-1,IST(JINTER),MU,EPS)
C OBTAIN INITIAL RIGHT HAND SIDE
       IF (NERROR(IERR).NE.0) CALL ERROFF
       DO 20 I=1,N
          EVEC(I)=1.0
  20   CONTINUE
       CALL BABS(N,G,IG,EVEC,N,1,MU)
       LIM=0
       JJ=ISAMAX(N,EVEC,1)
       SC=1.0/EVEC(JJ)
C SCALE FIRST RHS TO HAVE INFINITY NORM OF 1
       CALL SSCAL(N,SC,EVEC,1)
C ITERATIVE PHASE BEGINS HERE
  30   LIM=LIM+1
C MAKE A COPY OF OLD APPROXIMATION
       CALL MOVEFR(N,EVEC,R(JX))
C GET NEW APPROXIMATION OF EIGNVECTOR
       CALL BAFS(N,ML,R(JAL),ML-1,IST(JINTER),EVEC,N,1)
       CALL BABS(N,G,IG,EVEC,N,1,MU)
       BET=1.0/EVEC(JJ)
```

```
      JJ=ISAMAX(N,EVEC,1)
      SC2=1.0/EVEC(JJ)
C COMPUTE CONVERGENCE CRITERIA
      D1=0.0
      DO 40 I=1,N
         JXI=JX-1+I
         D1=AMAX1(D1,ABS((R(JXI)-BET*EVEC(I))*SC2))
  40  CONTINUE
      SC=SC2
      CALL SSCAL(N,SC,EVEC,1)
C TEST FOR CONVERGENCE AND IF ITERATION LIMIT EXCEEDED
      IF (D1.GT.EPS.AND.LIM.LT.LIMIT) GO TO 30
      CALL LEAVE
      RETURN
      END
```

To show that the above program works, a mainline program was written that packed into G the matrix

$$
\begin{array}{cccccccc}
-.75 & -.5 & & & & & & \\
-.5 & 1.0 & -1.0 & & & & & \\
 & -1.0 & 1.0 & -1.0 & & & & \\
 & & -1.0 & 1.0 & -1. & & & \\
 & & & \cdots & & & & \\
 & & & & -1.0 & 1.0 & -1.0 & \\
 & & & & & -1.0 & 1.0 & -.5 \\
 & & & & & & -.5 & -.75
\end{array}
$$

and invoked EIGVEC with the eigenvalue at -1.0.

```
      INTEGER N, I IWRITE, I1MACH
      REAL G(3, 200), EVEC(100)
      N=10
      DO 10 I=1,N
         G(1,I)=-1.0
         G(2,I)=1.0
         G(3,I)=-1.0
  10  CONTINUE
      G(2,1)=-.75
      G(2,N)=-.75
      G(3,1)=-.5
      G(1,2)=-.5
      G(1,N)=-.5
      G(3,N-1)=-.5
      IWRITE=I1MACH(2)
      CALL EIGVEC(N,3,2,G,3,-1.0,EVEC,2)
      DO 20 I=1,N
         WRITE(IWRITE,21)EVEC(I)
20    CONTINUE
21    FORMAT(12H EIGENVECTOR,F16.8)
      STOP
```

```
      END
```

When the above program was executed on the Honeywell 6000 machine at Bell Laboratories, which has about 8 decimal digits of precision, the following was printed:

```
EIGENVECTOR     1.00000000
EIGENVECTOR     0.49999998
EIGENVECTOR     0.49999996
EIGENVECTOR     0.49999994
EIGENVECTOR     0.49999991
EIGENVECTOR     0.49999989
EIGENVECTOR     0.49999986
EIGENVECTOR     0.49999982
EIGENVECTOR     0.49999976
EIGENVECTOR     0.99999938
```

Since the true eigenvector is $(1.0, 0.5, 0.5, ..., 0.5, 1.0)^T$, the fact that the eigenvector was computed by solving an ill-conditioned linear system did not affect our answer.

BACE — LU decomposition of a banded unsymmetric matrix with condition estimation

Purpose: BACE (BAnded matrix Condition Estimation) gives a lower bound for the condition number of a general banded matrix A. It also returns the LU decomposition of the matrix and may be used in place of BADC in a linear equation package.

Usage: CALL BACE (N, ML, M, G, IG, AL, IAL, INTER, MU, COND)

N	$\longrightarrow$	the order of the matrix A
ML	$\longrightarrow$	the number of nonzero bands on and below the diagonal of A
M	$\longrightarrow$	the number of nonzero bands in A
G	$\longrightarrow$	a matrix into which the matrix A has been packed as follows: $$G(ML + j-i, i) = a_{ij}$$ i.e. the leftmost diagonal of A is in the first row of G (See the introduction to this chapter.) G should be dimensional (IG,KG) in the calling program, where IG≥M and KG≥N.
	$\longleftarrow$	the upper triangular factor of A (see **Note 2**)
IG	$\longrightarrow$	the row (leading) dimension of G, as dimensioned in the calling program
AL	$\longleftarrow$	the lower triangular factor of A (see **Note 2**)
IAL	$\longrightarrow$	the row (leading) dimension of AL, as dimensioned in the calling program
INTER	$\longleftarrow$	an integer vector of length N recording the row interchanges performed during the decomposition (see **Note 2**)
MU	$\longleftarrow$	the number of nonzero bands in the upper triangular factor
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 $\mathbf{d} - \log_{10}(\mathrm{COND})$ correct decimal digits. Thus if COND is greater than $10^{\mathrm{Bd}\,P}$, there may be no correct digits.

Note 2: After execution of BACE, the arrays INTER and AL are suitable for input into the forward solve subroutine BAFS, and G is suitable for input into the back solver BABS. The LU decomposition of A satisfies the equation PA=LU where P is a permutation matrix, L is a unit lower triangular matrix and U is an upper triangular matrix. On return from BACE the element u_{ij} is contained in G(j–i+1,i), so that the main diagonal occupies the first row of the G matrix, the first super diagonal occupies the second row, etc. The matrix P can be obtained from INTER (see the introduction to this chapter), and the i^{th} column of the L matrix appears permuted in the i^{th} column of the AL array. Since the diagonal elements of L are all 1, they are not stored.

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	M < ML
4	IG < M
5	IAL < ML – 1
10 + k*	singular matrix whose rank is at least k

Double-precision version: DBACE with G, AL and EPS declared double precision.

Complex version: CBACE with G and AL declared complex

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

Time: at most N×(M×ML+6×M+ML) additions
at most N×(ML×M+3×M+ML–1) multiplications
N×(ML+1) divisions

Method: Gaussian elimination with partial pivoting
See the reference below for the method used to estimate the condition number.
BACE calls BALU with EPS=0.0

See also: BADC, BAFS, BABS, BALU, BALE, BASS

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 obtain estimates of the condition numbers of five matrices whose nonzero elements are given by $a_{ij} = i + j$ For each matrix, the 2×ML–1 main diagonals are nonzero. The program fragment packing the matrix A into the array G uses the fact that traversing a column of G is equivalent to traversing a row of A. The extra values that get put into G by the program below are made zero inside the subroutine BACE.

In this example we compare the condition number $K = ||A||\ \ ||A^{-1}||$ with the estimate obtained by BACE. In the program 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 our example $||A||$ is M×(N–ML+1)×2, which is obtained by summing the M elements in column N–ML+1.

```
      INTEGER IG, IGL, N, ML, M, I, J, MU, IWRITE, I1MACH
      INTEGER INTER(80)
      REAL G(13, 80), B(80), X(80), GL(6, 80)
      REAL START, FLOAT, AINNO, COND, CONDNO, ABS, AINNOI
      IG=13
      IGL=6
      N=80
      IWRITE=I1MACH(2)
      DO 60 ML=2,6
C
C CONSTRUCT THE MATRIX A(I,J)=I+J AND PACK IT INTO G
         M=2*ML - 1
         START=-FLOAT(M-ML)
         DO 20 I=1,N
            G(1,I)=START+FLOAT(2*I)
            DO 10 J=2,M
               G(J,I)=G(J-1,I)+1.
```

```
  10           CONTINUE
  20        CONTINUE
C
C DETERMINE AN ESTIMATE OF THE CONDITION NUMBER
C AND COMPUTE THE LU DECOMPOSITION
C
            CALL BACE(N,ML,M,G,IG,GL,IGL,INTER,MU,COND)
C
C DETERMINE THE NORM OF THE INVERSE MATRIX BY
C SOLVING FOR ONE COLUMN OF THE INVERSE MATRIX
C AT A TIME
C
            AINNO=0.0
            DO 50 I=1,N
C
C FIND THE ITH COLUMN OF THE INVERSE MATRIX BY
C SETTING THE RIGHT HAND SIDE TO THE ITH COLUMN
C OF THE IDENTITY MATRIX
C
                DO 30 J=1,N
                   B(J)=0.0
  30            CONTINUE
                B(I)=1.0
                CALL BAFS(N,ML,GL,IGL,INTER,B,80,1)
                CALL BABS(N,G,IG,B,80,1,MU)
C FIND THE NORM OF THE ITH COLUMN
                AINNOI=0.0
                DO 40 J=1,N
                   AINNOI=AINNOI+ABS(B(J))
  40            CONTINUE
                IF(AINNOI.GT.AINNO)AINNO=AINNOI
  50         CONTINUE
             WRITE(IWRITE,51)ML
  51         FORMAT(/6H ML IS ,I4)
             WRITE(IWRITE,52)COND
  52         FORMAT(22H CONDITION ESTIMATE IS,1PE15.7)
             CONDNO=AINNO*FLOAT(M*(N-ML+1)*2)
             WRITE(IWRITE,53)CONDNO
  53         FORMAT(22H TRUE CONDITION NO. IS,1PE15.7)
  60      CONTINUE
          STOP
          END
```

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

```
ML IS   2
CONDITION ESTIMATE IS  6.1040422E 03
TRUE CONDITION NO. IS  1.8941539E 04

ML IS   3
CONDITION ESTIMATE IS  5.9552785E 02
TRUE CONDITION NO. IS  2.1467948E 03

ML IS   4
CONDITION ESTIMATE IS  1.0581919E 07
TRUE CONDITION NO. IS  2.9246300E 07

ML IS   5
CONDITION ESTIMATE IS  3.2465961E 04
TRUE CONDITION NO. IS  6.7086635E 04

ML IS   6
CONDITION ESTIMATE IS  3.5264744E 07
TRUE CONDITION NO. IS  8.6640243E 07
```

In the comparison above of the condition number estimated by BACE and the true condition number, the order of magnitude of the estimate is correct, which is all one is usually interested in. Note that the inverse of a band matrix is usually a full $n \times n$ matrix, and should rarely be calculated.

BADC — decomposition of a banded unsymmetric matrix

Purpose: BADC (BAnded matrix DeComposition) finds the LU decompostion of a general banded matrix A using partial pivoting.

Usage: CALL BADC (N, ML, M, G, IG, AL, IAL, INTER, MU)

N $\longrightarrow$ the order of the matrix A

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

M $\longrightarrow$ the number of nonzero bands in A

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

$$G\,(ML + j-i,\, i) = a_{ij}$$

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

$\longleftarrow$ the upper triangular factor U of A (see **Note 1**)

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

AL $\longleftarrow$ the lower triangular factor of A (see **Note 1**)

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

INTER $\longrightarrow$ an integer vector of length N recording the row interchanges performed during the decomposition (see **Note 1**)

MU $\longleftarrow$ the number of nonzero bands in the upper triangular factor (MU≤M)

Note 1: After execution of BADC, the arrays INTER and AL are suitable for input into the forward solve subroutine BAFS and G is suitable for input into the back solve subroutine BABS. The LU decomposition of A satisfies the equation PA=LU where P is a permutation matrix, L is a unit lower triangular matrix and U is an upper triangular matrix. On return from BADC the element u_{ij} is contained in G(j–i+1,i), so that the main diagonal occupies the first row of the G matrix, the first super diagonal occupies the second row, etc. The matrix P can be obtained from INTER(see the introduction to this chapter), and the i^{th} column of the L matrix appears permuted in the i^{th} column of the AL array. Since the diagonal elements of L are all 1, they are not stored.

Note 2: MU≤M

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	M < ML
4	IG < M
5	IAL < ML − 1
10 + k*	singular matrix whose rank is at least k

Double-precision version: DBADC with G, AL and EPS declared double precision.

Complex version: CBADC with G and AL declared complex

Storage: None

Time: M×N+(ML−1)×N×(M−ML) ≤ additions ≤ (ML−1)×N×(M−1)+N×M
(ML−1)×N×(M−ML) ≤ multiplications ≤ (ML−1)×N×(M−1)
N×(ML−1) divisions

Method: Gaussian elimination with partial pivoting
BADC calls BALU after setting EPS = $||A||$ ε where ε is the machine precision, i.e. the value returned by R1MACH(4) (or, for double precision, by D1MACH(4)).

See also: BALU, BAFS, BABS, BACE, BALE, BASS

Author: Linda Kaufman

Example: In this example we implement a linearized version of Newton's method for solving $f(x)$=0 where f and x are vectors of length N. Newton's method is normally given as

Set k to 0. Initialize $x^{(0)}$
Until $||f(x^{(k)})|| \leq \varepsilon$ iterate as follows:

Solve $J(x^{(k)})y = f(x^{(k)})$
where $J_{i,l} = \dfrac{\partial f_i}{\partial x_l}$. Set $x^{(k+1)} = x^{(k)} - y$
Set k to k + 1

In some problems, especially those occurring in algorithms for solving time-varying partial differential equations, J is banded and costly to evaluate. Thus to solve f(x)=0, a linearized Newton's method is used in which $x^{(k+1)}$ is updated according to the formula $x^{(k+1)} = x^{(k)} - J(x^{(0)})^{-1} f(x^{(k)})$. In the following subroutine, implementing a linearized Newton method, FUN and JAC are assumed to be user provided functions which evaluate the function and its Jacobian. The function SNRM2 carefully computes the 2-norm of a vector.

```
       SUBROUTINE NEWTON(N,M,ML,X,EPS,FUN,JAC,LIMIT,F)
C
C THIS SUBROUTINE IMPLEMENTS A LINEARIZED FORM OF NEWTONS
C METHOD TO FIND THE ZERO OF A FUNCTION F DEFINED BY
C FUN, WHOSE BAND JACOBIAN (WITH BANDWIDTH M AND ML
C LOWER DIAGONALS) IS EVALUATED IN JAC. LIMIT GIVES
C A BOUND ON THE NUMBER OF ITERATIONS AND IN F THE
C FINAL FUNCTION VALUE IS RETURNED.
C
       INTEGER N, ML, M, LIMIT
       INTEGER JG, JAL, JINTER, ISTKGT, MU, LIM, I
       INTEGER IST(1000)
       REAL EPS, X(N), F(N)
       REAL FU, SNRM2, R(1000)
       DOUBLE PRECISION D(500)
       EXTERNAL FUN,JAC
       COMMON /CSTAK/ D
       EQUIVALENCE (D(1),R(1)),(D(1),IST(1))
C
C GET SPACE FOR G,INTER, AND AL FROM
C THE STORAGE STACK
```

```
C
       JG= ISTKGT(M*N,3)
       JAL = ISTKGT ((ML-1)*N,3)
       JINTER = ISTKGT(N,2)
       CALL JAC(N,M,ML,X,R(JG),M)
       CALL BADC(N,ML,M,R(JH),M,R(JAL),ML-1,IST(JINTER),MU)
       LIM=0
  10   CALL FUN(N,X,F)
       FU=SNRM2(N,F,1)
C
C CHECK FOR CONVERGENCE OR IF ITERATION LIMIT IS REACHED
C
      IF (FU.LE.EPS.OR.LIM.GT.LIMIT) RETURN
      LIM=LIM+1
C SOLVE THE LINEAR SYSTEM
      CALL BAFS(N,ML,R(JAL),ML-1,IST(JINTER),F,N,1)
      CALL BABS(N,R(JG),M,F,N,1,MU)
C CORRECT THE CURRENT ESTIMATE OF THE SOLUTION
      DO 20 I=1,N
         X(I)=X(I)-F(I)
  20  CONTINUE
      GO TO 10
      END
```

BALE — banded linear system solver

Purpose: BALE (BAnded Linear Equation solution) solves AX = B where A is a general banded matrix.

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

N $\longrightarrow$ the number of equations

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

M $\longrightarrow$ the number of nonzero bands of A

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

$$G(ML + j-i,\ i) = a_{ij}$$

i.e. the leftmost band of A is in the first row of G
(See the introduction to this chapter.)
G should be dimensional (IG,KG) in the calling program, where IG≥M and KG≥N.
G is overwritten during the solution.

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 the routine BASS in place of BALE.

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 BALE, but should call subprograms BADC, BAFS and BABS. (See the example in BADC.) BADC is called once to get the LU decomposition (see the introduction to this chapter) and then the pair, BAFS (forward solve) and BABS (back solve), is called for each new right-hand side.

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	M < ML
4	IG < M
5	IB < N
6	NB < 1
10 + k*	singular matrix whose rank is at least k

Double-precision version: DBALE with G and B declared double precision.

Complex version: CBALE with G and B declared complex

Storage: N integer locations of scratch storage in the dynamic storage stack

Time: at most N×(M×(ML+1)+(ML+M−2)×(NB−1)) additions
at most N×(ML×M+(ML+M−2)×(NB−1)) multiplications
N×(NB+ML−1) divisions

Method: Gaussian elimination with partial pivoting.
Transformations to A are not saved.

See also: BABS, BACE, BADC, BAFS, BASS, BALU

Author: Linda Kaufman

Example: In this example the relative efficiencies of BALE and BASS are compared for systems of various bandwidths and dimensions and various numbers of right-hand sides. The subroutine BASS solves a linear system with a banded matrix and also returns an estimate of the condition number of the matrix. The matrix used in this example is given by the formula
$a_{i,j} = |i-j|$
Since each diagonal of the matrix A corresponds to a particular row of the array G, and since all the elements on any diagonal of the matrix in our example are the same, each row of G in the program below was set to a constant. The right-hand sides were chosen randomly.

The function ILAPSZ is a timer on the Honeywell 6000 machine

```
with about a 1% accuracy.
It counts in 1/64 milliseconds.
```

```
      INTEGER IG, IWRITE, I1MACH, N, ML, II, MP1, I, K
      INTEGER IB, NB, IT, ILAPSZ
      REAL G(19, 100), B(100, 10), BB(100, 10), GG(19, 100)
      REAL COND, TIME1, TIME2, UNI
C
C THIS PROGRAM SOLVES BANDED SYSTEMS USING BALE AND
C BASS AND COMPARES THE TIME FOR EACH OF THEM. THE
C SYSTEMS HAVE VARIOUS BANDWIDTHS,DIMENSIONS, AND
C NUMBERS OF RIGHT-HAND SIDES
      DOUBLE PRECISION D(600)
      COMMON /CSTAK/ D
C MAKE SURE THE STACK MECHANISM HAS SUFFICIENT SPACE
C FOR BASS
      CALL ISTKIN(1200,3)
      IG=19
      IWRITE=I1MACH(2)
      IB=100
      DO 70 N=50,100,50
         DO 60 ML=2,10,8
            M=2*ML - 1
            MP1=M+1
            DO 50 NB=1,10,9
               WRITE(IWRITE,1)N,M,NB
 1             FORMAT(/5H N IS,I4,6H M IS ,I3,7H NB IS ,I3)
C
C CONSTRUCT THE MATRIX A(I,J)=ABS(I-J) AND PACK IT INTO G
C AND MAKE A COPY OF THE MATRIX SO THE SYSTEM CAN BE
C SOLVED WITH BOTH BALE AND BASS
C
               K=ML - 1
               DO 20 I=1,ML
                  II=MP1 - I
                  DO 10 J=1,N
                     G(I,J)=K
                     GG(I,J)=K
                     G(II,J)=K
                     GG(II,J)=K
 10               CONTINUE
                  K=K - 1
 20            CONTINUE
C
C CONSTRUCT RANDOM RIGHT-HAND SIDES
C AND MAKE A COPY
C
               DO 40 I=1,NB
                  DO 30 II=1,N
                     B(II,I)=UNI(0)
                     BB(II,I)=B(II,I)
 30               CONTINUE
 40            CONTINUE
C
C SOLVE THE SYSTEM USING BOTH BASS AND BALE
C
               IT=ILAPSZ(0)
               CALL BASS(N,ML,M,G,IG,B,IB,NB,COND)
               TIME1=(ILAPSZ(0)-IT)/64.0
               WRITE(IWRITE,41)TIME1
 41            FORMAT(34H TIME FOR BASS IN MILLISECONDS IS ,F10.1)
               IT=ILAPSZ(0)
               CALL BALE(N,ML,M,GG,IG,BB,IB,NB)
               TIME2=(ILAPSZ(0)-IT)/64.0
               WRITE(IWRITE,42)TIME2
 42            FORMAT(34H TIME FOR BALE IN MIILISECONDS IS ,F10.1)
 50         CONTINUE
 60      CONTINUE
 70   CONTINUE
```

```
STOP
END
```

When the above program was run on the Honeywell 6000 machine at Bell Labs with an optimizing compiler, the following was printed:

```
N IS  50 M IS   3 NB IS   1
TIME FOR BASS IN MILLISECONDS IS       50.0
TIME FOR BALE IN MIILISECONDS IS       20.0

N IS  50 M IS   3 NB IS  10
TIME FOR BASS IN MILLISECONDS IS       99.0
TIME FOR BALE IN MIILISECONDS IS       58.2

N IS  50 M IS  19 NB IS   1
TIME FOR BASS IN MILLISECONDS IS      200.8
TIME FOR BALE IN MIILISECONDS IS      147.8

N IS  50 M IS  19 NB IS  10
TIME FOR BASS IN MILLISECONDS IS      397.5
TIME FOR BALE IN MIILISECONDS IS      315.8

N IS 100 M IS   3 NB IS   1
TIME FOR BASS IN MILLISECONDS IS      102.8
TIME FOR BALE IN MIILISECONDS IS       36.4

N IS 100 M IS   3 NB IS  10
TIME FOR BASS IN MILLISECONDS IS      204.0
TIME FOR BALE IN MIILISECONDS IS      112.9

N IS 100 M IS  19 NB IS   1
TIME FOR BASS IN MILLISECONDS IS      416.6
TIME FOR BALE IN MIILISECONDS IS      302.4

N IS 100 M IS  19 NB IS  10
TIME FOR BASS IN MILLISECONDS IS      859.0
TIME FOR BALE IN MIILISECONDS IS      680.3
```

The above example indicates that the overhead for computing the condition estimate in BASS can be quite substantial for narrow banded systems with one right-hand side, but inconsequential if the bandwidth is large or if the system has many right-hand sides. The example also indicates that the execution time is linear in the number of equations, but certainly not linear in the number of right-hand sides. Users with many right-hand sides, which are known in advance and which all correspond to the same coefficient matrix, should obviously not invoke BALE for each new right-hand side, but call BALE once with NB set appropriately.

BALU — LU decomposition of a banded unsymmetric matrix

Purpose: BALU (BAnded matrix LU decomposition) finds the LU decompostion of a general banded matrix A using partial pivoting. It allows the user to specify a threshold for considering a matrix singular. BALU is called by the LU decomposition routines BACE and BADC.

Usage: CALL BALU (N, ML, M, G, IG, AL, IAL, INTER, MU, EPS)

N $\longrightarrow$ the order of the matrix A

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

M $\longrightarrow$ the number of nonzero bands in A

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

G (ML + j−i, i) = a_{ij}

i.e. the leftmost 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≥M and KG≥N.

$\longleftarrow$ the upper triangular factor of A (see **Note 2**)

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

AL $\longleftarrow$ the lower triangular factor of A (see **Note 2**)

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

INTER $\longleftarrow$ an integer vector of length N recording the row interchanges performed during the decomposition (see **Note 2**)

MU $\longleftarrow$ the number of nonzero bands in the upper triangular factor

EPS $\longrightarrow$ if A = LU and there exists an index k such that $|u_{kk}| \le$ EPS then A is considered singular

Note 1: After execution of BALU, (if the matrix is not found to be singular), the value of the determinant is INTER(N) × G(1,1) × G(1,2) × . . . × G(1,N). INTER(N) contains the sign of the permutation.

Note 2: After execution of BALU, the arrays INTER and AL are suitable for input into the forward solve subroutine BAFS and G is suitable for input into the back solve subroutine BABS. The LU decomposition of A satisfies the equation PA=LU where P is a permutation matrix, L is a unit lower triangular matrix and U is an upper triangular matrix. On return from BALU the element u_{ij} is contained in G(j−i+1,i), so that the main diagonal occupies the first row of the G matrix, the first super diagonal occupies the second row, etc. The matrix P can be obtained from INTER (see the introduction to this chapter), and the i^{th} column of the L matrix appears permuted in the i^{th} column of the AL array. Since the diagonal elements of L are all 1, they are not stored.

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	M < ML
4	IG < M
5	IAL < ML − 1
10 + k*	singular matrix whose rank is at least k

Double-precision version: DBALU with G, AL and EPS declared double precision.

Complex version: CBALU with G and AL declared complex

Storage: None

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

Method: Gaussian elimination with partial pivoting

See also: BADC, BAFS, BABS, BACE, BALE, BASS

Author: Linda Kaufman

Example: The program below computes the determinant of a band matrix stored in G in packed form. After the call to BALU the determinant is just INT(N) $\times$ the product of the elements in the first row of G. Since the subroutine BADET requires the user to provide only the space needed to hold the original matrix, it uses the stack mechanism provided in PORT to get the extra space needed by BALU. The subroutine tries to avoid underflow and overflow during the calculation. The subroutine UMKFL is used to decompose a floating-point number, F, into a mantissa, S, and an exponent E such that $F = Sb^E$ where b is the base of the machine and $1/b \le |S| < 1$

```
       SUBROUTINE BADET(N,ML,M,A,IA,DETMAN,IDETEX)
C
C THIS SUBROUTINE COMPUTES THE DETERMINANT OF A
C BANDED MATRIX STORED IN PACKED FORM IN A
C THE DETERMINANT IS COMPUTED AS DETMAN*BETA**IDETEX,
C WHERE BETA IS THE BASE OF THE MACHINE AND
C DETMAN IS BETWEEN 1/BETA AND 1 IN ABSOLUTE VALUE
C
       INTEGER ML, M, N, IA, IDETEX
       INTEGER E, ISPAC, IALOW, ISTKGT, ISIGN, INTER, I, MU
       INTEGER IN(1000)
       REAL A(IA,1), DETMAN, BETA, ONOVBE, S
       REAL R(1000)
       DOUBLE PRECISION D(500)
       COMMON /CSTAK/D
       EQUIVALENCE(D(1),R(1)),(D(1),IN(1))
C
C ALLOCATE SPACE FROM THE STACK FOR THE PIVOT ARRAY
C AND THE EXTRA SPACE TO HOLD THE LOWER TRIANGLE
C
       ISPAC=(ML-1)*N
       IALOW=ISTKGT(ISPAC,3)
       INTER=ISTKGT(N,2)
       CALL BALU(N,ML,M,A,IA,R(IALOW),ML-1,IN(INTER),MU,0.0)
C
C THE DETERMINANT IS THE PRODUCT OF THE ELEMENTS OF
C ROW 1 OF A TIMES THE LAST ELEMENT IN THE ARRAY INTER.
C WE TRY TO COMPUTE THIS PRODUCT IN A WAY THAT WILL
C AVOID UNDERFLOW AND OVERFLOW.
C
       BETA=FLOAT(I1MACH(10))
       ONOVBE=1.0/BETA
       ISIGN=INTER+N-1
       DETMAN=IN(ISIGN)*ONOVBE
       IDETEX=1
       DO 10 I=1,N
          CALL UMKFL(A(1,I),E,S)
          DETMAN=DETMAN*S
          IDETEX=IDETEX+E
          IF (ABS(DETMAN).GE.ONOVBE) GO TO 10
             IDETEX=IDETEX-1
             DETMAN=DETMAN*BETA
10      CONTINUE
        CALL ISTKRL(2)
        RETURN
        END
```

BAML — banded matrix - vector multiplication

Purpose: BAML matrix (BAnded matrix MuLtiplication) forms the product Ax where A is a general banded matrix stored in packed form.

Usage: CALL BAML (N, ML, M, G, IG, X, B)

N → the order of the matrix A

ML → the number of nonzero bands on and below the diagonal of A

M → the number of nonzero bands of A

G → a matrix into which the matrix A has been packed as follows:

$$G(ML + j-i, i) = a_{ij}$$

i.e. the leftmost band 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≥M and KG≥N.

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

X → the vector x to be multiplied

B ← the vector Ax

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

Number	Error
1	N < 1
2	ML < 1
3	M < ML
4	IG < M

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

Complex version: CBAML with G, X, and B declared complex

Storage: None

Time: M×N additions
M×N multiplications

See also: BABS, BACE, BADC, BALU, BALE, BASS

Author: Linda Kaufman

Example: This example checks the consistency of BAML and BASS the banded system solver. First the example uses BAML to compute for a given vector x and a given matrix A the vector $b = Ax$. Then the problem is inverted, i.e., BASS 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

$$\begin{array}{cccccccc}
0 & 1 & 2 & & & & & \\
1 & 0 & 1 & 2 & & & & \\
2 & 1 & 0 & 1 & 2 & & & \\
 & 2 & 1 & 0 & 1 & 2 & & \\
 & & & . & . & . & & \\
 & & & 2 & 1 & 0 & 1 & 2 \\
 & & & & 2 & 1 & 0 & 1 \\
 & & & & & 2 & 1 & 0
\end{array}$$

```
         INTEGER IG, M, ML, N, I, IWRITE, I1MACH
         REAL G(5,20), X(20), B(20), UNI, ERR, SASUM, ABS, COND
         IG=5
         M=5
         N=10
         ML=3
C
C CONSTRUCT THE A MATRIX AND PACK IT INTO G
C
          DO 10 I=1,N
             G(1,I)=2.0
             G(2,I)=1.0
             G(3,I)=0.0
             G(4,I)=1.0
             G(5,I)=2.0
  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 BAML(N,ML,M,G,IG,X,B)
C
C SOLVE THE SYSTEM AX=B
C
          CALL BASS(N,ML,M,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 ,2E17.8)
C
C COMPUTE THE RELATIVE ERROR
C
          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 $1.\times10^{-8}$, the following was printed:

```
TRUE SOLUTION   COMPUTED SOLUTION
   0.22925607E 00   0.22925605E 00
   0.76687502E 00   0.76687499E 00
   0.68317685E 00   0.68317685E 00
   0.50919111E 00   0.50919110E 00
   0.87455959E 00   0.87455962E 00
   0.64464101E 00   0.64464102E 00
   0.84746840E 00   0.84746839E 00
   0.35396343E 00   0.35396345E 00
   0.39889160E 00   0.39889155E 00
   0.45709422E 00   0.45709425E 00
RELATIVE ERROR IS   3.8447574E-08
CONDITION NUMBER IS  4.3333333E 01
```

The condition number of the matrix and the precision of the Honeywell suggest that even in the absence of roundoff error in BAML, a relative error of 4.3×10^{-7} would not be surprising. The value computed above is quite reasonable.

BANM — norm of a banded unsymmetric matrix

Purpose: BANM (BAnded matrix NorM) computes the infinity norm of a general banded matrix A. The infinity norm is defined as $\max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$

Type: Real function

Usage: <answer> = BANM (N, ML, M, G, IG)

N $\longrightarrow$ the number of rows in A

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

M $\longrightarrow$ the number of nonzero bands in A

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

G (ML + j−i, i) = a_{ij}

i.e. the leftmost 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≥M 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	ML < 1
3	M < ML
4	IG < M

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

Complex version: CBANM with G declared complex

Storage: None

Time: N × M additions
N comparisons

See also: BADC, BALU, BALE, BASS, BACE

Author: Linda Kaufman

Example: In this example we verify the correctness of BANM by obtaining norms of five matrices of various bandwidths whose nonzero elements are given by $a_{ij} = i + j$ For each matrix, the 2×ML−1 main diagonals are nonzero. The program fragment packing the matrix A into the array G uses the fact that traversing a column of G is equivalent to traversing a row of A. The extra values that get put into G by the program are not referenced by BANM.

In our example the norm computed by BANM is compared with the true norm, known to be M×(N−ML+1)×2, which is obtained by summing the M elements in column N−ML+1.

```
      INTEGER IG,  ML,  M,  N,  I,  J, IWRITE, I1MACH
      REAL G(13,  80),  START,  BANM,  TRNORM
      IG=13
      N=80
      DO 30 ML=2,6
C
C CONSTRUCT THE MATRIX A(I,J)=I+J AND PACK IT INTO G
C
         M=2*ML-1
         START=-FLOAT(M-ML)
         DO 20 I=1,N
            G(1,I)=START+FLOAT(2*I)
            DO 10 J=2,M
               G(J,I)=G(J-1,I)+1.0
 10         CONTINUE
 20      CONTINUE
C
C PRINT OUT THE NORM CALCULATED FROM BANM AND THE TRUE NORM
C
         TRNORM=M*(N-ML+1)*2
         IWRITE=I1MACH(2)
         WRITE(IWRITE,21)ML
 21      FORMAT(/6H ML IS ,I4)
         WRITE(IWRITE,22)TRNORM,BANM(N,ML,M,G,IG)
 22      FORMAT(15H THE TRUE NORM=,E15.5,15H COMPUTED NORM=,E15.5)
 30   CONTINUE
      STOP
      END
```

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

```
 ML IS   2
 THE TRUE NORM=    0.47400E 03 COMPUTED NORM=    0.47400E 03
```

```
ML IS   3
THE TRUE NORM=    0.78000E 03 COMPUTED NORM=    0.78000E 03

ML IS   4
THE TRUE NORM=    0.10780E 04 COMPUTED NORM=    0.10780E 04

ML IS   5
THE TRUE NORM=    0.13680E 04 COMPUTED NORM=    0.13680E 04

ML IS   6
THE TRUE NORM=    0.16500E 04 COMPUTED NORM=    0.16500E 04
```

BASS — banded linear system solution with condition estimation

Purpose: BASS (BAnded System Solution) solves AX=B where A is a general banded matrix. An estimate of the condition number of A is provided.

Usage: CALL BASS (N, ML, M, G, IG, B, IB, NB, COND)

N $\longrightarrow$ the number of equations

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

M $\longrightarrow$ the number of nonzero bands of A

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

$$G(ML+j-i,\ i) = a_{ij}$$

i.e. the leftmost band 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≥M and KG≥N.
G is overwritten during the solution

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 BALE, which is a little faster than BASS. Ordinarily, however, the user is strongly urged to choose BASS, 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 BASS, but should call subprograms BACE, BAFS and BABS. (See the example of BADC.) BACE is called once to get the LU decomposition (see the introduction to this chapter) and then the pair, BAFS (forward solve) and BABS (back solve), is called for each new right-hand side.

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	M < ML
4	IG < M
5	IB < N
6	NB < 1
10 + k*	singular matrix whose rank is at least k

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

Complex version: CBASS with G and B declared complex

Storage: N integer locations and N×ML real (double precision for DBASS, complex for CBASS) locations of scratch storage in the dynamic storage stack

Time: at most N×((ML+M−2)×(NB+1)+M×(ML+5)+3) additions
at most N×((ML+M−2)×(NB+1)+M×(ML+2)+2) multiplications
N×(NB+ML+1) divisions

Method: Gaussian elimination with partial pivoting
See the reference below for the method to estimate the condition number.
BASS calls BACE, BAFS, and BABS

See also: BABS, BACE, BADC, BALU, BALE, BAFS

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 this example several banded linear systems are solved with various number of diagonals and two right-hand sides each. Estimates of the condition numbers of the coefficient matrices are found and the relative errors in the solution are calculated. In the example the nonzero elements of the matrix are given by $a_{ij} = i + j$ For each matrix the 2×ML−1 main diagonals are nonzero. The program fragment packing the matrix A into the array G uses the fact that traversing a column of G is equivalent to traversing a row of A. The right-hand sides are determined so that the elements of the first solution are all ones and the i^{th} element of the second solution is i. The subroutine BAML, which multiplies a vector by a banded matrix packed appropriately into G, is invoked to compute the right-hand sides.

```
       INTEGER N, IG, ML, M, I, J, IWRITE, I1MACH
       REAL G(13,80), B(80,2), X(80)
       REAL START, FLOAT, ERR, ERR2, ABS, COND
       IG=13
       N=80
       DO 60 ML=2,6
C
C CONSTRUCT THE MATRIX A(I,J)=I+J AND PACK IT INTO G
C
          M=2*ML-1
          START=-FLOAT(M-ML)
          DO 20 I=1,N
             G(1,I)=START+FLOAT(2*I)
             IF(M.EQ.1) GO TO 20
             DO 10 J=2,M
                G(J,I)=G(J-1,I)+1.
  10         CONTINUE
  20      CONTINUE
C CONSTRUCT FIRST RIGHT-HAND SIDE SO SOLUTION IS ALL 1S
          DO 30 I=1,N
  30         X(I)=1
          CALL BAML(N,ML,M,G,IG,X,B)
C CONSTRUCT THE SECOND COLUMN SO X(I)=I
          DO 40 I=1,N
  40         X(I)=I
          CALL BAML(N,ML,M,G,IG,X,B(1,2))
C SOLVE THE SYSTEM
          CALL BASS(N,ML,M,G,IG,B,80,2,COND)
C COMPUTE THE ERRORS IN THE SOLUTION
          ERR=0.0
          ERR2=0.0
          DO 50 I=1,N
             ERR=ERR+ABS(B(I,1)-1.0)
             ERR2=ERR2+ABS(B(I,2)-FLOAT(I))
  50      CONTINUE
          ERR=ERR/FLOAT(N)
          ERR2=ERR2/FLOAT(N*(N+1))*2.0
          IWRITE=I1MACH(2)
          WRITE(IWRITE,51)ML,COND
  51      FORMAT(/9H WHEN ML=,I4,21H THE CONDITION NO. IS,1PE15.7)
          WRITE(IWRITE,52)ERR
  52      FORMAT(38H REL. ERROR IN THE FIRST SOLUTION IS  ,1PE15.7)
          WRITE(IWRITE,53)ERR2
  53      FORMAT(38H REL. ERROR IN THE SECOND SOLUTION IS ,1PE15.7)
  60   CONTINUE
  70  CONTINUE
      STOP
      END
```

When the above program was executed on on the Honeywell 6000 computer at Bell Labs, the following was printed.

```
WHEN ML=   2 THE CONDITION NO. IS  6.1040422E 03
REL. ERROR IN THE FIRST SOLUTION IS    5.0114468E-07
REL. ERROR IN THE SECOND SOLUTION IS   6.0025235E-07

WHEN ML=   3 THE CONDITION NO. IS  5.9552785E 02
REL. ERROR IN THE FIRST SOLUTION IS    1.2554228E-07
REL. ERROR IN THE SECOND SOLUTION IS   1.1807790E-07

WHEN ML=   4 THE CONDITION NO. IS  1.0581919E 07
REL. ERROR IN THE FIRST SOLUTION IS    5.9645883E-04
REL. ERROR IN THE SECOND SOLUTION IS   2.1994722E-03

WHEN ML=   5 THE CONDITION NO. IS  3.2465961E 04
REL. ERROR IN THE FIRST SOLUTION IS    2.4201348E-06
REL. ERROR IN THE SECOND SOLUTION IS   7.6222429E-07

WHEN ML=   6 THE CONDITION NO. IS  3.5264744E 07
REL. ERROR IN THE FIRST SOLUTION IS    4.2312816E-04
REL. ERROR IN THE SECOND SOLUTION IS   2.4684287E-03
```

The above program certainly indicates that there is a correlation between the relative errors in the solution and the condition number of the coefficient matrix. Moreover it indicates that the relative error depends also on the choice of the right-hand side. Although the relative errors for some of the systems might appear quite large, they are not unreasonably large in light of the following analysis:

Let Δb represent a perturbation in the right-hand side of a linear system. If $Ax = b$ then $A(x+\Delta x) = b+\Delta b$ where $\frac{||\Delta x||}{||x||} \le K(A)\left[\frac{||\Delta b||}{||b||}\right]$ where $K(A)$ is the condition number of A, $K(A) = ||A||\ ||A^{-1}||$ and $||\cdot||$, is some norm, e.g., $||x||_1 = \sum_{i=1}^{n}|x_i|$ if x is a vector.

The methods used in our linear equation package are guaranteed to provide an accurate answer to a slightly perturbed problem. If we assume that our method produces the correct answer to a problem where $||\Delta b|| \le \varepsilon||b||$, where ε is the machine precision, then on the Honeywell 6000 where ε is about 10^{-8}, a relative error for the above problem (when ML=6) of 3.5×10^{-1} is not surprising.