F01BSF – NAG Fortran Library Routine Document

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

F01BSF factorizes a real sparse matrix using the pivotal sequence previously obtained by F01BRF when a matrix of the same sparsity pattern was factorized.

2 Specification

```
 SUBROUTINE F01BSF(N, NZ, A, LICN, IVECT, JVECT, ICN, IKEEP, IW, W,
1                  GROW, ETA, RPMIN, ABORT, IDISP, IFAIL)
 INTEGER           N, NZ, LICN, IVECT(NZ), JVECT(NZ), ICN(LICN),
1                  IKEEP(5*N), IW(8*N), IDISP(2), IFAIL
 real              A(LICN), W(N), ETA, RPMIN
 LOGICAL           GROW, ABORT
```

3 Description

This routine accepts as input a real sparse matrix of the same sparsity pattern as a matrix previously factorized by a call of F01BRF. It first applies to the matrix the same permutations as were used by F01BRF, both for permutation to block triangular form and for pivoting, and then performs Gaussian elimination to obtain the LU factorization of the diagonal blocks.

Extensive data checks are made; duplicated non-zeros can be accumulated.

The factorization is intended to be used by F04AXF to solve sparse systems of linear equations $Ax = b$ or $A^T x = b$.

F01BSF is much faster than F01BRF and in some applications it is expected that there will be many calls of F01BSF for each call of F01BRF.

The method is fully described in Duff [1].

4 References

[1] Duff I S (1977) MA28 – a set of Fortran subroutines for sparse unsymmetric linear equations *AERE Report R8730* HMSO

5 Parameters

1: N — INTEGER *Input*

On entry: n, the order of the matrix A.

Constraint: N > 0.

2: NZ — INTEGER *Input*

On entry: the number of non-zeros in the matrix A.

Constraint: NZ > 0.

3: A(LICN) — ***real*** array *Input/Output*

On entry: A(i), for $i = 1, 2, \ldots$,NZ must contain the non-zero elements of the sparse matrix A. They can be in any order since the routine will reorder them.

On exit: the non-zero elements in the factorization. The array must not be changed by the user between a call of this routine and a call of F04AXF.

4: LICN — INTEGER *Input*

On entry: the dimension of the arrays A and ICN as declared in the (sub)program from which F01BSF is called. It should have the same value as it had for F01BRF.

Constraint: LICN $\geq$ NZ.

5: IVECT(NZ) — INTEGER array *Input*

6: JVECT(NZ) — INTEGER array *Input*

On entry: IVECT(i) and JVECT(i), for $i = 1, 2, \ldots,$NZ must contain the row index and the column index respectively of the non-zero element stored in A(i).

7: ICN(LICN) — INTEGER array *Input*

On entry: the same information as output by F01BRF. It must not be changed by the user between a call of this routine and a call of F04AXF.

8: IKEEP(5∗N) — INTEGER array *Input*

On entry: the same indexing information about the factorization as output from F01BRF. It must not be changed between a call of this routine and a call of F04AXF.

9: IW(8∗N) — INTEGER array *Workspace*

10: W(N) — ***real*** array *Output*

On exit: if GROW = .TRUE., W(1) contains an estimate (an upper bound) of the increase in size of elements encountered during the factorization (see GROW); the rest of the array is used as workspace.

If GROW = .FALSE., the array is not used.

11: GROW — LOGICAL *Input*

On entry: if GROW = .TRUE., then on exit W(1) contains an estimate (an upper bound) of the increase in size of elements encountered during the factorization. If the matrix is well-scaled (see Section 8), then a high value for W(1) indicates that the *LU* factorization may be inaccurate and the user should be wary of the results and perhaps increase the parameter PIVOT for subsequent runs (see Section 7).

12: ETA — ***real*** *Input*

On entry: the relative pivot threshold below which an error diagnostic is provoked and IFAIL is set to 7. If ETA is greater than 1.0, then no check on pivot size is made.

Suggested value: ETA = 10^{-4}.

13: RPMIN — ***real*** *Output*

On exit: if ETA is less than 1.0, then RPMIN gives the smallest ratio of the pivot to the largest element in the row of the corresponding upper triangular factor thus monitoring the stability of the factorization. If RPMIN is very small it may be advisable to perform a new factorization using F01BRF.

14: ABORT — LOGICAL *Input*

On entry: if ABORT = .TRUE., the routine exits immediately (with IFAIL = 8) if it finds duplicate elements in the input matrix. If ABORT = .FALSE., the routine proceeds using a value equal to the sum of the duplicate elements. In either case details of each duplicate element are output on the current advisory message unit (see X04ABF), unless suppressed by the value of IFAIL on entry.

Suggested value: ABORT = .TRUE..

15: IDISP(2) — INTEGER array *Input*

On entry: IDISP(1) and IDISP(2) must be unchanged since the previous call of F01BRF.

16: IFAIL — INTEGER *Input/Output*

For this routine, the normal use of IFAIL is extended to control the printing of error and warning messages as well as specifying hard or soft failure (see Chapter P01).

Before entry, IFAIL must be set to a value with the decimal expansion *cba*, where each of the decimal digits c, b and a must have a value of 0 or 1.

$a = 0$ specifies hard failure, otherwise soft failure;

$b = 0$ suppresses error messages, otherwise error messages will be printed (see Section 6);

$c = 0$ suppresses warning messages, otherwise warning messages will be printed (see Section 6).

The recommended value for inexperienced users is 110 (i.e., hard failure with all messages printed).

Unless the routine detects an error (see Section 6), IFAIL contains 0 on exit.

6 Error Indicators and Warnings

For each error, an explanatory error message is output on the current error message unit (as defined by X04AAF), unless suppressed by the value of IFAIL on entry.

Errors detected by the routine:

IFAIL = 1

On entry, $\mathrm{N} \leq 0$.

IFAIL = 2

On entry, $\mathrm{NZ} \leq 0$.

IFAIL = 3

On entry, $\mathrm{LICN} < \mathrm{NZ}$.

IFAIL = 4

On entry, an element of the input matrix has a row or column index (i.e., an element of IVECT or JVECT) outside the range 1 to N.

IFAIL = 5

The input matrix is incompatible with the matrix factorized by the previous call of F01BRF (see Section 8).

IFAIL = 6

The input matrix is numerically singular.

IFAIL = 7

A very small pivot has been detected (see Section 5, ETA). The factorization has been completed but is potentially unstable.

IFAIL = 8

Duplicate elements have been found in the input matrix and the factorization has been abandoned (ABORT = .TRUE.on entry).

7 Accuracy

The factorization obtained is exact for a perturbed matrix whose (i, j)th element differs from a_{ij} by less than $3\epsilon\rho m_{ij}$ where ϵ is the ***machine precision***, ρ is the growth value returned in W(1) if GROW = .TRUE., and m_{ij} the number of Gaussian elimination operations applied to element (i, j).

If ρ = W(1) is very large or RPMIN is very small, then a fresh call of F01BRF is recommended.

8 Further Comments

If the user has a sequence of problems with the same sparsity pattern then this routine is recommended after F01BRF has been called for one such problem. It is typically 4 to 7 times faster but is potentially unstable since the previous pivotal sequence is used. Further details on timing are given in document F01BRF.

If growth estimation is performed (GROW = .TRUE.), then the time increases by between 5% and 10%. Pivot size monitoring (ETA ≤ 1.0) involves a similar overhead.

We normally expect this routine to be entered with a matrix having the same pattern of non-zeros as was earlier presented to F01BRF. However there is no record of this pattern, but rather a record of the pattern including all fill-ins. Therefore we permit additional non-zeros in positions corresponding to fill-ins.

If singular matrices are being treated then it is also required that the present matrix be sufficiently like the previous one for the same permutations to be suitable for factorization with the same set of zero pivots.

9 Example

To factorize the real sparse matrices

$$\begin{pmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & -1 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ -2 & 0 & 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & -1 & 2 & -3 \\ -1 & -1 & 0 & 0 & 0 & 6 \end{pmatrix}$$

and

$$\begin{pmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 12 & -3 & -1 & 0 & 0 \\ 0 & 0 & 15 & 0 & 0 & 0 \\ -2 & 0 & 0 & 10 & -1 & 0 \\ -1 & 0 & 0 & -5 & 1 & -1 \\ -1 & -2 & 0 & 0 & 0 & 6 \end{pmatrix}.$$

This example program simply prints the values of W(1) and RPMIN returned by F01BSF . Normally the calls of F01BRF and F01BSF would be followed by calls of F04AXF .

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F01BSF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NMAX, NZMAX, LICN, LIRN
      PARAMETER        (NMAX=20,NZMAX=50,LICN=3*NZMAX,LIRN=3*NZMAX/2)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      real             ETA, RPMIN, U
      INTEGER          I, IFAIL, N, NZ
      LOGICAL          GROW, LBLOCK
*     .. Local Arrays ..
      real             A(LICN), W(NMAX)
      INTEGER          ICN(LICN), IDISP(10), IKEEP(NMAX,5), IRN(LIRN),
     +                 IVECT(NZMAX), IW(NMAX,8), JVECT(NZMAX)
```

```
      LOGICAL          ABORT(4)
*     .. External Subroutines ..
      EXTERNAL         F01BRF, F01BSF, X04ABF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F01BSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NZ
      CALL X04ABF(1,NOUT)
      WRITE (NOUT,*)
      IF (N.GT.0 .AND. N.LE.NMAX .AND. NZ.GT.0 .AND. NZ.LE.NZMAX) THEN
         READ (NIN,*) (A(I),IRN(I),ICN(I),I=1,NZ)
         U = 0.1e0
         LBLOCK = .TRUE.
         GROW = .TRUE.
         ABORT(1) = .TRUE.
         ABORT(2) = .TRUE.
         ABORT(3) = .FALSE.
         ABORT(4) = .TRUE.
         IFAIL = 110
*
         CALL F01BRF(N,NZ,A,LICN,IRN,LIRN,ICN,U,IKEEP,IW,W,LBLOCK,GROW,
     +               ABORT,IDISP,IFAIL)
*
         IF (GROW) THEN
            WRITE (NOUT,*) 'On exit from F01BRF'
            WRITE (NOUT,99998) 'Value of W(1)  = ', W(1)
         END IF
         READ (NIN,*) (A(I),IVECT(I),JVECT(I),I=1,NZ)
         ETA = 0.1e0
         IFAIL = 110
*
         CALL F01BSF(N,NZ,A,LICN,IVECT,JVECT,ICN,IKEEP,IW,W,GROW,ETA,
     +               RPMIN,ABORT(4),IDISP,IFAIL)
*
         IF (GROW) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'On exit from F01BSF'
            WRITE (NOUT,99998) 'Value of W(1)  = ', W(1)
         END IF
         IF (ETA.LT.1.0e0) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,99998) 'Value of RPMIN = ', RPMIN
         END IF
      ELSE
         WRITE (NOUT,*) 'N or NZ is out of range.'
         WRITE (NOUT,99999) 'N = ', N, '  NZ = ', NZ
      END IF
      STOP
*
99999 FORMAT (1X,A,I5,A,I5)
99998 FORMAT (1X,A,F7.4)
      END
```

9.2 Program Data

```
F01BSF Example Program Data
  6 15
   5.0  1  1   2.0  2  2  -1.0  2  3   2.0  2  4   3.0  3  3
  -2.0  4  1   1.0  4  4   1.0  4  5  -1.0  5  1  -1.0  5  4
   2.0  5  5  -3.0  5  6  -1.0  6  1  -1.0  6  2   6.0  6  6
  10.0  1  1  12.0  2  2  -3.0  2  3  -1.0  2  4  15.0  3  3
  -2.0  4  1  10.0  4  4  -1.0  4  5  -1.0  5  1  -5.0  5  4
   1.0  5  5  -1.0  5  6  -1.0  6  1  -2.0  6  2   6.0  6  6
```

9.3 Program Results

```
F01BSF Example Program Results

On exit from F01BRF
Value of W(1)  = 18.0000

On exit from F01BSF
Value of W(1)  = 51.0000

Value of RPMIN =  0.1000
```