NAG Library Routine Document

F01BSF

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,        &
                   GROW, ETA, RPMIN, ABORT, IDISP, IFAIL)
INTEGER            N, NZ, LICN, IVECT(NZ), JVECT(NZ), ICN(LICN),           &
                   IKEEP(5*N), IW(5*N), IDISP(2), IFAIL
REAL (KIND=nag_wp) A(LICN), W(N), ETA, RPMIN
LOGICAL            GROW, ABORT
```

3 Description

F01BSF 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 nonzeros can be accumulated.

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

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 (1977).

A more recent algorithm for the same calculation is provided by F11MEF.

4 References

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

5 Arguments

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 nonzero elements in the matrix A.

Constraint: NZ > 0.

3: A(LICN) – REAL (KIND=nag_wp) array *Input/Output*

On entry: A(i), for $i = 1, 2, \ldots, \text{NZ}$, must contain the nonzero elements of the sparse matrix A. They can be in any order since F01BSF will reorder them.

On exit: the nonzero elements in the LU factorization. The array must **not** be changed by you between a call of F01BSF 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 $i = 1, 2, \ldots, \text{NZ}$, must contain the row index and the column index respectively of the nonzero element stored in A(i).

7: ICN(LICN) – INTEGER array *Input*

ICN contains, on entry, the same information as output by F01BRF. It must not be changed by you between a call of F01BSF and a call of F04AXF.

ICN is used as internal workspace prior to being restored on exit and hence is unchanged.

8: IKEEP($5 \times$ N) – INTEGER array *Communication Array*

On entry: the same indexing information about the factorization as output in IKEEP by F01BRF.

You must **not** change IKEEP between a call of F01BSF and subsequent calls to F04AXF.

9: IW($5 \times$ N) – INTEGER array *Workspace*

10: W(N) – REAL (KIND=nag_wp) 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 9), then a high value for W(1) indicates that the LU factorization may be inaccurate and you should be wary of the results and perhaps increase the argument PIVOT for subsequent runs (see Section 7).

12: ETA – REAL (KIND=nag_wp) *Input*

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

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

13: RPMIN – REAL (KIND=nag_wp) *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., F01BSF exits immediately (with IFAIL = 8) if it finds duplicate elements in the input matrix.

If ABORT = .FALSE., F01BSF 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 *Communication Array*

On entry: IDISP(1) and IDISP(2) must be as output in IDISP by 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 Section 3.4 in How to Use the NAG Library and its Documentation).

On 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).

On exit: IFAIL = 0 unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL = 1

On entry, N ≤ 0.

IFAIL = 2

On entry, NZ ≤ 0.

IFAIL = 3

On entry, LICN < 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 9).

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).

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

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

8 Parallelism and Performance

F01BSF is not threaded in any implementation.

9 Further Comments

If you have a sequence of problems with the same sparsity pattern then F01BSF 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 the document for 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 nonzeros 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 nonzeros 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.

10 Example

This example factorizes 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.

10.1 Program Text

```
    Program f01bsfe

!     F01BSF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f01brf, f01bsf, nag_wp, x04abf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: iset = 1, nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: eta, rpmin, u
      Integer                          :: i, ifail, licn, lirn, n, nz, outchn
      Logical                          :: grow, lblock
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:), w(:)
      Integer, Allocatable             :: icn(:), ikeep(:,:), irn(:),          &
                                          ivect(:), iw(:,:), jvect(:)
      Integer                          :: idisp(10)
      Logical                          :: abort(4)
!     .. Executable Statements ..
      Write (nout,*) 'F01BSF Example Program Results'
      outchn = nout
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n, nz
      licn = 3*nz
      lirn = 3*nz/2
      Allocate (a(licn),w(n),icn(licn),ikeep(n,5),irn(lirn),ivect(nz),iw(n,8), &
        jvect(nz))
      Call x04abf(iset,outchn)
      Write (nout,*)
      Read (nin,*)(a(i),irn(i),icn(i),i=1,nz)
      u = 0.1E0_nag_wp
      lblock = .True.
      grow = .True.
      abort(1) = .True.
      abort(2) = .True.
      abort(3) = .False.
      abort(4) = .True.
```

```
!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      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,99999) 'Value of W(1)  = ', w(1)
      End If
      Read (nin,*)(a(i),ivect(i),jvect(i),i=1,nz)
      eta = 0.1E0_nag_wp

!     ifail: behaviour on error exit
!            =110 for noisy, hard exit
      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,99999) 'Value of W(1)  = ', w(1)
      End If
      If (eta<1.0E0_nag_wp) Then
        Write (nout,*)
        Write (nout,99999) 'Value of RPMIN = ', rpmin
      End If

99999 Format (1X,A,F7.4)
    End Program f01bsfe
```

10.2 Program Data

```
F01BSF Example Program Data
  6 15                                                          : n, nz
   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    : a
```

10.3 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
```