NAG Library Routine Document

F11DFF

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

F11DFF computes a block diagonal incomplete LU factorization of a real sparse nonsymmetric matrix, represented in coordinate storage format. The diagonal blocks may be composed of arbitrary rows and the corresponding columns, and may overlap. This factorization can be used to provide a block Jacobi or additive Schwarz preconditioner, for use in combination with F11BEF or F11DGF.

2 Specification

```
SUBROUTINE F11DFF (N, NNZ, A, LA, IROW, ICOL, NB, ISTB, INDB, LINDB,        &
                   LFILL, DTOL, PSTRAT, MILU, IPIVP, IPIVQ, ISTR, IDIAG,    &
                   NNZC, NPIVM, IWORK, LIWORK, IFAIL)

INTEGER            N, NNZ, LA, IROW(LA), ICOL(LA), NB, ISTB(NB+1),          &
                   INDB(LINDB), LINDB, LFILL(NB), IPIVP(LINDB),             &
                   IPIVQ(LINDB), ISTR(LINDB+1), IDIAG(LINDB), NNZC,         &
                   NPIVM(NB), IWORK(LIWORK), LIWORK, IFAIL
REAL (KIND=nag_wp) A(LA), DTOL(NB)
CHARACTER(1)       PSTRAT(NB), MILU(NB)
```

3 Description

F11DFF computes an incomplete LU factorization (see Meijerink and Van der Vorst (1977) and Meijerink and Van der Vorst (1981)) of the (possibly overlapping) diagonal blocks A_b, for $b = 1, 2, \ldots, \mathrm{NB}$, of a real sparse nonsymmetric n by n matrix A. The factorization is intended primarily for use as a block Jacobi or additive Schwarz preconditioner (see Saad (1996)), with one of the iterative solvers F11BEF and F11DGF.

The NB diagonal blocks need not consist of consecutive rows and columns of A, but may be composed of arbitrarily indexed rows, and the corresponding columns, as defined in the arguments INDB and ISTB. Any given row or column index may appear in more than one diagonal block, resulting in overlap. Each diagonal block A_b, for $b = 1, 2, \ldots, \mathrm{NB}$, is factorized as:

$$A_b = M_b + R_b$$

where

$$M_b = P_b L_b D_b U_b Q_b$$

and L_b is lower triangular with unit diagonal elements, D_b is diagonal, U_b is upper triangular with unit diagonals, P_b and Q_b are permutation matrices, and R_b is a remainder matrix.

The amount of fill-in occurring in the factorization of block b can vary from zero to complete fill, and can be controlled by specifying either the maximum level of fill $\mathrm{LFILL}(b)$, or the drop tolerance $\mathrm{DTOL}(b)$.

The parameter $\mathrm{PSTRAT}(b)$ defines the pivoting strategy to be used in block b. The options currently available are no pivoting, user-defined pivoting, partial pivoting by columns for stability, and complete pivoting by rows for sparsity and by columns for stability. The factorization may optionally be modified to preserve the row-sums of the original block matrix.

The sparse matrix A is represented in coordinate storage (CS) format (see Section 2.1.1 in the F11 Chapter Introduction). The array A stores all the nonzero elements of the matrix A, while arrays IROW and ICOL store the corresponding row and column indices respectively. Multiple nonzero elements may not be specified for the same row and column index.

The preconditioning matrices M_b, for $b = 1, 2, \ldots, \text{NB}$, are returned in terms of the CS representations of the matrices

$$C_b = L_b + {D^{-1}}_b + U_b - 2I.$$

4 References

Meijerink J and Van der Vorst H (1977) An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix *Math. Comput.* **31** 148–162

Meijerink J and Van der Vorst H (1981) Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems *J. Comput. Phys.* **44** 134–155

Saad Y (1996) *Iterative Methods for Sparse Linear Systems* PWS Publishing Company, Boston, MA

5 Parameters

1: N – INTEGER *Input*

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

Constraint: $\text{N} \geq 1$.

2: NNZ – INTEGER *Input*

On entry: the number of nonzero elements in the matrix A.

Constraint: $1 \leq \text{NNZ} \leq \text{N}^2$.

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

On entry: the nonzero elements in the matrix A, ordered by increasing row index, and by increasing column index within each row. Multiple entries for the same row and column indices are not permitted. The routine F11ZAF may be used to order the elements in this way.

On exit: the first NNZ entries of A contain the nonzero elements of A and the next NNZC entries contain the elements of the matrices C_b, for $b = 1, 2, \ldots, \text{NB}$ stored consecutively. Within each block the matrix elements are ordered by increasing row index, and by increasing column index within each row.

4: LA – INTEGER *Input*

On entry: the dimension of the arrays A, IROW and ICOL as declared in the (sub)program from which F11DFF is called. These arrays must be of sufficient size to store both A (NNZ elements) and C (NNZC elements).

Note: the minimum value for LA is only appropriate if LFILL and DTOL are set such that minimal fill-in occurs. If this is not the case then we recommend that LA is set much larger than the minimum value indicated in the constraint.

Constraint: $\text{LA} \geq 2 \times \text{NNZ}$.

5: IROW(LA) – INTEGER array *Input/Output*

6: ICOL(LA) – INTEGER array *Input/Output*

On entry: the row and column indices of the nonzero elements supplied in A.

Constraints:

IROW and ICOL must satisfy these constraints (which may be imposed by a call to F11ZAF):

$1 \leq \text{IROW}(i) \leq \text{N}$ and $1 \leq \text{ICOL}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZ}$;
either $\text{IROW}(i-1) < \text{IROW}(i)$ or both $\text{IROW}(i-1) = \text{IROW}(i)$ and $\text{ICOL}(i-1) < \text{ICOL}(i)$, for $i = 2, 3, \ldots, \text{NNZ}$.

On exit: the row and column indices of the nonzero elements returned in A.

7: NB – INTEGER *Input*

On entry: the number of diagonal blocks to factorize.

Constraint: $1 \leq \text{NB} \leq \text{N}$.

8: ISTB(NB + 1) – INTEGER array *Input*

On entry: ISTB(b), for $b = 1, 2, \ldots, \text{NB}$, holds the indices in arrays INDB, IPIVP, IPIVQ and IDIAG that, on successful exit from this function, define block b. ISTB(NB + 1) holds the sum of the number of rows in all blocks plus ISTB(1).

Constraint: $\text{ISTB}(1) \geq 1, \text{ISTB}(b) < \text{ISTB}(b+1)$, for $b = 1, 2, \ldots, \text{NB}$.

9: INDB(LINDB) – INTEGER array *Input*

On entry: INDB must hold the row indices appearing in each diagonal block, stored consecutively. Thus the elements INDB(ISTB(b)) to INDB(ISTB($b+1$) − 1) are the row indices in the bth block, for $b = 1, 2, \ldots, \text{NB}$.

Constraint: $1 \leq \text{INDB}(m) \leq \text{N}$, for $m = 1, 2, \ldots, \text{ISTB}(\text{NB}+1) - 1$.

10: LINDB – INTEGER *Input*

On entry: the dimension of the arrays INDB, IPIVP, IPIVQ and IDIAG as declared in the (sub)program from which F11DFF is called.

Constraint: $\text{LINDB} \geq \text{ISTB}(\text{NB}+1) - 1$.

11: LFILL(NB) – INTEGER array *Input*

On entry: if LFILL(b) ≥ 0 its value is the maximum level of fill allowed in the decomposition of the block (see Section 9.2 in F11DAF). A negative value of LFILL(b) indicates that DTOL(b) will be used to control the fill in the block instead.

12: DTOL(NB) – REAL (KIND=nag_wp) array *Input*

On entry: if LFILL(b) < 0 then DTOL(b) is used as a drop tolerance in the block to control the fill-in (see Section 9.2 in F11DAF); otherwise DTOL(b) is not referenced.

Constraint: if LFILL(b) < 0, DTOL(b) ≥ 0.0, for $b = 1, 2, \ldots, \text{NB}$.

13: PSTRAT(NB) – CHARACTER(1) array *Input*

On entry: PSTRAT(b), for $b = 1, 2, \ldots, \text{NB}$, specifies the pivoting strategy to be adopted in the block as follows:

PSTRAT(b) = 'N'
No pivoting is carried out.

PSTRAT(b) = 'U'
Pivoting is carried out according to the user-defined input values of IPIVP and IPIVQ.

PSTRAT(b) = 'P'
Partial pivoting by columns for stability is carried out.

PSTRAT(b) = 'C'
Complete pivoting by rows for sparsity, and by columns for stability, is carried out.

Suggested value: PSTRAT(b) = 'C', for $b = 1, 2, \ldots, \text{NB}$.

Constraint: PSTRAT(b) = 'N', 'U', 'P' or 'C', for $b = 1, 2, \ldots, \text{NB}$.

14: MILU(NB) – CHARACTER(1) array *Input*

On entry: MILU(b), for $b = 1, 2, \ldots, \text{NB}$, indicates whether or not the factorization in the block should be modified to preserve row-sums (see Section 9.4 in F11DAF).

MILU(b) = 'M'
The factorization is modified.

MILU(b) = 'N'
The factorization is not modified.

Constraint: MILU(b) = 'M' or 'N', for $b = 1, 2, \ldots, \text{NB}$.

15: IPIVP(LINDB) – INTEGER array *Input/Output*

16: IPIVQ(LINDB) – INTEGER array *Input/Output*

On entry: if PSTRAT(b) = 'U', then IPIVP(ISTB(b) + $k - 1$) and IPIVQ(ISTB(b) + $k - 1$) must specify the row and column indices of the element used as a pivot at elimination stage k of the factorization of the block. Otherwise IPIVP and IPIVQ need not be initialized.

Constraint: if PSTRAT(b) = 'U', the elements ISTB(b) to ISTB($b + 1$) $- 1$ of IPIVP and IPIVQ must both hold valid permutations of the integers on $[1, \text{ISTB}(b + 1) - \text{ISTB}(b)]$.

On exit: the row and column indices of the pivot elements, arranged consecutively for each block, as for INDB. If IPIVP(ISTB(b) + $k - 1$) = i and IPIVQ(ISTB(b) + $k - 1$) = j, then the element in row i and column j of A_b was used as the pivot at elimination stage k.

17: ISTR(LINDB + 1) – INTEGER array *Output*

On exit: ISTR(ISTB(b) + $k - 1$), gives the index in the arrays A, IROW and ICOL of row k of the matrix C_b, for $b = 1, 2, \ldots, \text{NB}$ and $k = 1, 2, \ldots, \text{ISTB}(b + 1) - \text{ISTB}(b)$.

ISTR(ISTB(NB + 1)) contains NNZ + NNZC + 1.

18: IDIAG(LINDB) – INTEGER array *Output*

On exit: IDIAG(ISTB(b) + $k - 1$), gives the index in the arrays A, IROW and ICOL of the diagonal element in row k of the matrix C_b, for $b = 1, 2, \ldots, \text{NB}$ and $k = 1, 2, \ldots, \text{ISTB}(b + 1) - \text{ISTB}(b)$.

19: NNZC – INTEGER *Output*

On exit: the sum total number of nonzero elements in the matrices C_b, for $b = 1, 2, \ldots, \text{NB}$.

20: NPIVM(NB) – INTEGER array *Output*

On exit: if NPIVM(b) > 0 it gives the number of pivots which were modified during the factorization to ensure that M_b exists.

If NPIVM(b) $= -1$ no pivot modifications were required, but a local restart occurred (see Section 9.3 in F11DAF). The quality of the preconditioner will generally depend on the returned values of NPIVM(b), for $b = 1, 2, \ldots, \text{NB}$.

If NPIVM(b) is large, for some block, the preconditioner may not be satisfactory. In this case it may be advantageous to call F11DFF again with an increased value of LFILL(b), a reduced value of DTOL(b), or PSTRAT(b) = 'C'.

21: IWORK(LIWORK) – INTEGER array *Workspace*

22: LIWORK – INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which F11DFF is called.

Constraint: LIWORK $\geq 9 \times \text{N} + 3$.

23: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

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, DTOL($\langle value \rangle$) $= \langle value \rangle$.
Constraint: DTOL(b) ≥ 0.0, for $b = 1, 2, \ldots,$ NB.

On entry, for $b = \langle value \rangle$, ISTB($b+1$) $= \langle value \rangle$ and ISTB(b) $= \langle value \rangle$.
Constraint: ISTB($b+1$) $>$ ISTB(b), for $b = 1, 2, \ldots,$ NB.

On entry, INDB($\langle value \rangle$) $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: $1 \leq$ INDB(m) $\leq$ N, for $m = 1, 2, \ldots,$ ISTB(NB $+ 1) - 1$

On entry, ISTB(1) $= \langle value \rangle$.
Constraint: ISTB(1) ≥ 1.

On entry, LA $= \langle value \rangle$ and NNZ $= \langle value \rangle$.
Constraint: LA $\geq 2 \times$ NNZ.

On entry, LINDB $= \langle value \rangle$, ISTB(NB $+ 1) - 1 = \langle value \rangle$ and NB $= \langle value \rangle$.
Constraint: LINDB $\geq$ ISTB(NB $+ 1) - 1$.

On entry, LIWORK $= \langle value \rangle$.
Constraint: LIWORK $\geq \langle value \rangle$.

On entry, MILU($\langle value \rangle$) $= \langle value \rangle$.
Constraint: MILU(b) $=$ 'M' or 'N' for all b.

On entry, N $= \langle value \rangle$.
Constraint: N ≥ 1.

On entry, NB $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: $1 \leq$ NB $\leq$ N.

On entry, NNZ $= \langle value \rangle$.
Constraint: NNZ ≥ 1.

On entry, NNZ $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: NNZ $\leq$ N^2.

On entry, PSTRAT($\langle value \rangle$) $= \langle value \rangle$.
Constraint: PSTRAT(b) $=$ 'N', 'U', 'P' or 'C' for all b.

IFAIL $= 2$

On entry, element $\langle value \rangle$ of A was out of order.

On entry, ICOL($\langle value \rangle$) $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: $1 \leq$ ICOL(j) $\leq$ N, for $j = 1, 2, \ldots,$ NNZ.

On entry, IROW($\langle value \rangle$) = $\langle value \rangle$ and N = $\langle value \rangle$.
Constraint: $1 \leq \text{IROW}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZ}$.

On entry, location $\langle value \rangle$ of (IROW, ICOL) was a duplicate.

IFAIL = 3

On entry, the user-supplied value of IPIVP for block $\langle value \rangle$ lies outside its range.

On entry, the user-supplied value of IPIVP for block $\langle value \rangle$ was repeated.

On entry, the user-supplied value of IPIVQ for block $\langle value \rangle$ lies outside its range.

On entry, the user-supplied value of IPIVQ for block $\langle value \rangle$ was repeated.

IFAIL = 4

The number of nonzero entries in the decomposition is too large.
The decomposition has been terminated before completion.
Either increase LA, or reduce the fill by reducing LFILL, or increasing DTOL.

IFAIL = −99

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

See Section 3.8 in the Essential Introduction for further information.

IFAIL = −399

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

See Section 3.7 in the Essential Introduction for further information.

IFAIL = −999

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

The accuracy of the factorization of each block A_b will be determined by the size of the elements that are dropped and the size of any modifications made to the pivot elements. If these sizes are small then the computed factors will correspond to a matrix close to A_b. The factorization can generally be made more accurate by increasing the level of fill LFILL(b), or by reducing the drop tolerance DTOL(b) with LFILL(b) < 0.

If F11DFF is used in combination with F11BEF or F11DGF, the more accurate the factorization the fewer iterations will be required. However, the cost of the decomposition will also generally increase.

8 Parallelism and Performance

Not applicable.

9 Further Comments

F11DFF calls F11DAF internally for each block A_b. The comments and advice provided in Section 9 in F11DAF on timing, control of fill, algorithmic details, and choice of parameters, are all therefore relevant to F11DFF, if interpreted blockwise.

10 Example

This example program reads in a sparse matrix A and then defines a block partitioning of the row indices with a user-supplied overlap and computes an overlapping incomplete LU factorization suitable for use as an additive Schwarz preconditioner. Such a factorization is used for this purpose in the example program of F11DGF.

10.1 Program Text

```
!   F11DFF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

    Program f11dffe

!     .. Use Statements ..
      Use nag_library, Only: f11dff, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter              :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)              :: dtolg
      Integer                         :: i, ifail, k, la, lfillg, lindb,      &
                                         liwork, mb, n, nb, nnz, nnzc, nover
      Character (1)                   :: milug, pstrag
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:), b(:), dtol(:)
      Integer, Allocatable            :: icol(:), idiag(:), indb(:),          &
                                         ipivp(:), ipivq(:), irow(:),         &
                                         istb(:), istr(:), iwork(:),          &
                                         lfill(:), npivm(:)
      Character (1), Allocatable      :: milu(:), pstrat(:)
!     .. Intrinsic Procedures ..
      Intrinsic                       :: maxval, minval
!     .. Executable Statements ..
      Continue

!     Print example header
      Write (nout,*) 'F11DFF Example Program Results'
      Write (nout,*)

!     Skip heading in data file
      Read (nin,*)

!     Get the matrix order and number of non-zero entries.
      Read (nin,*) n
      Read (nin,*) nnz

!     Allocate arrays with lengths based on n and nnz..
      liwork = 9*n + 3
      Allocate (b(n),iwork(liwork))

      la = 20*nnz
      Allocate (a(la),irow(la),icol(la))

      lindb = 3*n
      Allocate (idiag(lindb),indb(lindb),ipivp(lindb),ipivq(lindb), &
        istr(lindb+1))

!     Read the matrix A
      Read (nin,*)(a(i),irow(i),icol(i),i=1,nnz)

!     Read algorithmic parameters
      Read (nin,*) lfillg, dtolg
      Read (nin,*) pstrag
      Read (nin,*) milug
      Read (nin,*) nb, nover

!     Allocate arrays with length based on number of blocks.
```

```
      Allocate (dtol(nb),istb(nb+1),lfill(nb),npivm(nb),milu(nb),pstrat(nb))

!     Define diagonal block indices.
!     In this example use blocks of MB consecutive rows and initialise
!     assuming no overlap.
      mb = (n+nb-1)/nb
      Do k = 1, nb
        istb(k) = (k-1)*mb + 1
      End Do
      istb(nb+1) = n + 1
      Do i = 1, n
        indb(i) = i
      End Do

!     Modify INDB and ISTB to account for overlap.
      Call f11dffe_overlap(n,nnz,la,irow,icol,nb,istb,indb,lindb,nover,iwork)
      If (iwork(1)==-999) Then
        Write (nout,*) '** LINDB too small, LINDB = ', lindb, '.'
        Go To 100
      End If

!     Output matrix and blocking details
      Write (nout,*) ' Original Matrix'
      Write (nout,99997) ' N     =', n
      Write (nout,99997) ' NNZ   =', nnz
      Write (nout,99997) ' NB    =', nb
      Do k = 1, nb
        Write (nout,99993) ' Block ', k, ': order = ', istb(k+1) - istb(k), &
          ', start row = ', minval(indb(istb(k):istb(k+1)-1))
      End Do

!     Set algorithmic parameters for each block from global values
      lfill(1:nb) = lfillg
      dtol(1:nb) = dtolg
      pstrat(1:nb) = pstrag
      milu(1:nb) = milug

!     Calculate factorization
      ifail = 1
      Call f11dff(n,nnz,a,la,irow,icol,nb,istb,indb,lindb,lfill,dtol,pstrat, &
        milu,ipivp,ipivq,istr,idiag,nnzc,npivm,iwork,liwork,ifail)
      If (ifail/=0) Then
        Write (nout,99995) ifail
        Go To 100
      End If

!     Output details of the factorization
      Write (nout,99996) ' Factorization'
      Write (nout,99997) '  NNZC  =', nnzc

      Write (nout,99996) ' Elements of factorization'
      Write (nout,99996) '        I   J        C(I,J)     Index'
      Do k = 1, nb
        Write (nout,99994) ' C_', k, ' -------------------------------'
!       Elements of the k-th block
        Do i = istr(istb(k)), istr(istb(k+1)) - 1
          Write (nout,99992) irow(i), icol(i), a(i), i
        End Do
      End Do

      Write (nout,99996) ' Details of factorized blocks'
      If (maxval(npivm(1:nb))>0) Then
!       Including pivoting details.
        Write (nout,99996) &
          '  K   I      ISTR(I)  IDIAG(I)   INDB(I)  IPIVP(I)  IPIVQ(I)'
        Do k = 1, nb
          i = istb(k)
          Write (nout,99999) k, i, istr(i), idiag(i), indb(i), ipivp(i), &
            ipivq(i)
          Do i = istb(k) + 1, istb(k+1) - 1
            Write (nout,99998) i, istr(i), idiag(i), indb(i), ipivp(i), &
```

```
              ipivq(i)
            End Do
            Write (nout,*) &
              ' ------------------------------------------------'
          End Do
        Else
!         No pivoting on any block.
          Write (nout,99996) '  K   I      ISTR(I)  IDIAG(I)   INDB(I)'
          Do k = 1, nb
            i = istb(k)
            Write (nout,99999) k, i, istr(i), idiag(i), indb(i)
            Do i = istb(k) + 1, istb(k+1) - 1
              Write (nout,99998) i, istr(i), idiag(i), indb(i)
            End Do
            Write (nout,*) ' ----------------------------------'
          End Do
        End If

100   Continue

99999 Format (1X,I3,1X,I3,5(I10))
99998 Format (1X,I7,5(I10))
99997 Format (1X,A,I4)
99996 Format (1X/1X,A)
99995 Format (1X/1X,' ** F11DFF returned with IFAIL = ',I5)
99994 Format (1X,A3,I1,A)
99993 Format (1X,A,I3,A,I3,A,I3)
99992 Format (6X,2I4,E16.5,I8)

    Contains

      Subroutine f11dffe_overlap(n,nnz,la,irow,icol,nb,istb,indb,lindb,nover, &
        iwork)

!       Purpose
!       =======
!       This routine takes a set of row indices INDB defining the diagonal
!       blocks to be used in F11DFF to define a block Jacobi or additive Schwarz
!       preconditioner, and expands them to allow for NOVER levels of overlap.
!       The pointer array ISTB is also updated accordingly, so that the returned
!       values of ISTB and INDB can be passed to F11DFF to define overlapping
!       diagonal blocks.
!       ------------------------------------------------------------------------

!       .. Implicit None Statement ..
        Implicit None
!       .. Scalar Arguments ..
        Integer, Intent (In)             :: la, lindb, n, nb, nnz, nover
!       .. Array Arguments ..
        Integer, Intent (In)             :: icol(la), irow(la)
        Integer, Intent (Inout)          :: indb(lindb), istb(nb+1)
        Integer, Intent (Out)            :: iwork(3*n+1)
!       .. Local Scalars ..
        Integer                          :: i, ik, ind, iover, k, l, n21,      &
                                            nadd, row
!       .. Executable Statements ..
        Continue

!       Find the number of non-zero elements in each row of the matrix A, and
!       and start address of each row. Store the start addresses in
!       IWORK(N+1,...,2*N+1).

        iwork(1:n) = 0
        Do k = 1, nnz
          iwork(irow(k)) = iwork(irow(k)) + 1
        End Do
        iwork(n+1) = 1
        Do i = 1, n
          iwork(n+i+1) = iwork(n+i) + iwork(i)
        End Do
```

```
!       Loop over blocks.
blocks: Do k = 1, nb

!         Initialize marker array.
          iwork(1:n) = 0

!         Mark the rows already in block K in the workspace array.
          Do l = istb(k), istb(k+1) - 1
            iwork(indb(l)) = 1
          End Do

!         Loop over levels of overlap.
          Do iover = 1, nover

!           Initialize counter of new row indices to be added.
            ind = 0

!           Loop over the rows currently in the diagonal block.
            Do l = istb(k), istb(k+1) - 1
              row = indb(l)

!             Loop over non-zero elements in row ROW.
              Do i = iwork(n+row), iwork(n+row+1) - 1

!               If the column index of the non-zero element is not in the
!               existing set for this block, store it to be added later, and
!               mark it in the marker array.
                If (iwork(icol(i))==0) Then
                  iwork(icol(i)) = 1
                  ind = ind + 1
                  iwork(2*n+1+ind) = icol(i)
                End If
              End Do
            End Do

!           Shift the indices in INDB and add the new entries for block K.
!           Change ISTB accordingly.
            nadd = ind
            If (istb(nb+1)+nadd-1>lindb) Then
              iwork(1) = -999
              Exit blocks
            End If

            Do i = istb(nb+1) - 1, istb(k+1), -1
              indb(i+nadd) = indb(i)
            End Do
            n21 = 2*n + 1
            ik = istb(k+1) - 1
            indb(ik+1:ik+nadd) = iwork(n21+1:n21+nadd)
            istb(k+1:nb+1) = istb(k+1:nb+1) + nadd
          End Do
        End Do blocks

        Return

      End Subroutine f11dffe_overlap
    End Program f11dffe
```

10.2 Program Data

```
F11DFF Example Program Data
  9  : n
  33                : nnz
  64.0      1       1
 -20.0      1       2
 -20.0      1       4
 -12.0      2       1
  64.0      2       2
 -20.0      2       3
 -20.0      2       5
```

```
-12.0      3       2
 64.0      3       3
-20.0      3       6
-12.0      4       1
 64.0      4       4
-20.0      4       5
-20.0      4       7
-12.0      5       2
-12.0      5       4
 64.0      5       5
-20.0      5       6
-20.0      5       8
-12.0      6       3
-12.0      6       5
 64.0      6       6
-20.0      6       9
-12.0      7       4
 64.0      7       7
-20.0      7       8
-12.0      8       5
-12.0      8       7
 64.0      8       8
-20.0      8       9
-12.0      9       6
-12.0      9       8
 64.0      9       9    : a(i), irow(i), icol(i) for i=1,nnz
 0   0.0               : lfillg, dtolg
'N'                    : pstrag
'N'                    : milug
 3   1                 : nb, nover
```

10.3 Program Results

```
F11DFF Example Program Results

 Original Matrix
 N     =   9
 NNZ   =  33
 NB    =   3
 Block   1: order =   6, start row =   1
 Block   2: order =   9, start row =   1
 Block   3: order =   6, start row =   4

 Factorization
  NNZC  =  73

 Elements of factorization

        I   J        C(I,J)     Index
 C_1 --------------------------------
        1   1     0.15625E-01      34
        1   2    -0.31250E+00      35
        1   4    -0.31250E+00      36
        2   1    -0.18750E+00      37
        2   2     0.16598E-01      38
        2   3    -0.33195E+00      39
        2   5    -0.33195E+00      40
        3   2    -0.19917E+00      41
        3   3     0.16662E-01      42
        3   6    -0.33324E+00      43
        4   1    -0.18750E+00      44
        4   4     0.16598E-01      45
        4   5    -0.33195E+00      46
        5   2    -0.19917E+00      47
        5   4    -0.19917E+00      48
        5   5     0.17847E-01      49
        5   6    -0.35693E+00      50
        6   3    -0.19994E+00      51
        6   5    -0.21416E+00      52
        6   6     0.17948E-01      53
```

```
C_2 --------------------------------
       1   1     0.15625E-01      54
       1   2    -0.31250E+00      55
       1   4    -0.18750E+00      56
       1   5    -0.31250E+00      57
       2   1    -0.18750E+00      58
       2   2     0.16598E-01      59
       2   3    -0.33195E+00      60
       2   6    -0.19917E+00      61
       2   7    -0.33195E+00      62
       3   2    -0.19917E+00      63
       3   3     0.16662E-01      64
       3   8    -0.19994E+00      65
       3   9    -0.33324E+00      66
       4   1    -0.31250E+00      67
       4   4     0.16598E-01      68
       4   6    -0.33195E+00      69
       5   1    -0.18750E+00      70
       5   5     0.16598E-01      71
       5   7    -0.33195E+00      72
       6   2    -0.33195E+00      73
       6   4    -0.19917E+00      74
       6   6     0.17847E-01      75
       6   8    -0.35693E+00      76
       7   2    -0.19917E+00      77
       7   5    -0.19917E+00      78
       7   7     0.17847E-01      79
       7   9    -0.35693E+00      80
       8   3    -0.33324E+00      81
       8   6    -0.21416E+00      82
       8   8     0.17948E-01      83
       9   3    -0.19994E+00      84
       9   7    -0.21416E+00      85
       9   9     0.17948E-01      86
C_3 --------------------------------
       1   1     0.15625E-01      87
       1   2    -0.31250E+00      88
       1   4    -0.18750E+00      89
       2   1    -0.18750E+00      90
       2   2     0.16598E-01      91
       2   3    -0.33195E+00      92
       2   5    -0.19917E+00      93
       3   2    -0.19917E+00      94
       3   3     0.16662E-01      95
       3   6    -0.19994E+00      96
       4   1    -0.31250E+00      97
       4   4     0.16598E-01      98
       4   5    -0.33195E+00      99
       5   2    -0.33195E+00     100
       5   4    -0.19917E+00     101
       5   5     0.17847E-01     102
       5   6    -0.35693E+00     103
       6   3    -0.33324E+00     104
       6   5    -0.21416E+00     105
       6   6     0.17948E-01     106

Details of factorized blocks

 K   I      ISTR(I)  IDIAG(I)   INDB(I)
 1   1        34        34         1
     2        37        38         2
     3        41        42         3
     4        44        45         4
     5        47        49         5
     6        51        53         6
-----------------------------------
 2   7        54        54         4
     8        58        59         5
     9        63        64         6
    10        67        68         1
    11        70        71         7
```

```
      12         73         75          2
      13         77         79          8
      14         81         83          3
      15         84         86          9
  ------------------------------------------
 3    16         87         87          7
      17         90         91          8
      18         94         95          9
      19         97         98          4
      20        100        102          5
      21        104        106          6
  ------------------------------------------
```