NAG Library Routine Document

F11YEF

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

F11YEF reduces the bandwidth of a sparse symmetric matrix stored in compressed column storage format using the Reverse Cuthill–McKee algorithm.

2 Specification

```
SUBROUTINE F11YEF (N, NNZ, ICOLZP, IROWIX, LOPTS, MASK, PERM, INFO,        &
                   IFAIL)
INTEGER N, NNZ, ICOLZP(N+1), IROWIX(NNZ), MASK(*), PERM(N), INFO(4),       &
        IFAIL
LOGICAL LOPTS(5)
```

3 Description

F11YEF takes the compressed column storage (CCS) representation (see Section 2.1.3 in the F11 Chapter Introduction) of an n by n symmetric matrix A and applies the Reverse Cuthill–McKee (RCM) algorithm which aims to minimize the bandwidth of the matrix A by reordering the rows and columns symmetrically. This also results in a lower profile of the matrix (see Section 9).

F11YEF can be useful for solving systems of equations $Ax = b$, as the permuted system $PAP^{\mathrm{T}}(Px) = Pb$ (where P is the permutation matrix described by the vector PERM returned by F11YEF) may require less storage space and/or less computational steps when solving (see Wai-Hung and Sherman (1976)).

F11YEF may be used prior to F11JAF and F11JBF (see Section 10 in F11JBF).

4 References

Pissanetsky S (1984) *Sparse Matrix Technology* Academic Press

Wai-Hung L and Sherman A H (1976) Comparative analysis of the Cuthill–McKee and the reverse Cuthill–McKee ordering algorithms for sparse matrices *SIAM J. Numer. Anal.* **13(2)** 198–213

5 Arguments

1: N – INTEGER *Input*

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

Constraint: N ≥ 1.

2: NNZ – INTEGER *Input*

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

Constraint: NNZ ≥ 0.

3: ICOLZP(N + 1) – INTEGER array *Input*

On entry: ICOLZP records the index into IROWIX which starts each new column.

Constraints:

$1 \leq \text{ICOLZP}(i) \leq \text{NNZ} + 1$, for $i = 2, 3, \ldots, \text{N}$;
$\text{ICOLZP}(1) = 1$;
$\text{ICOLZP}(\text{N} + 1) = \text{NNZ} + 1$, where $\text{ICOLZP}(i)$ holds the position integer for the starts of the columns in IROWIX.

4: IROWIX(NNZ) – INTEGER array *Input*

On entry: the row indices corresponding to the nonzero elements in the matrix A.

Constraint: $1 \leq \text{IROWIX}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZ}$.

5: LOPTS(5) – LOGICAL array *Input*

On entry: the options to be used by F11YEF.

LOPTS(1) = .TRUE.
Row/column i of the matrix A will only be referenced if $\text{MASK}(i) \neq 0$, otherwise MASK will be ignored.

LOPTS(2) = .TRUE.
The final permutation will not be reversed, that is, the Cuthill–McKee ordering will be returned. The bandwidth of the non-reversed matrix will be the same but the profile will be the same or larger (see Wai-Hung and Sherman (1976)).

LOPTS(3) = .TRUE.
The matrix A will be checked for symmetrical sparsity pattern, otherwise not.

LOPTS(4) = .TRUE.
The bandwidth and profile of the unpermuted matrix will be calculated, otherwise not.

LOPTS(5) = .TRUE.
The bandwidth and profile of the permuted matrix will be calculated, otherwise not.

6: MASK($*$) – INTEGER array *Input*

Note: the dimension of the array MASK must be at least N if LOPTS(1) = .TRUE., and at least 0 otherwise.

On entry: MASK is only referenced if LOPTS(1) is .TRUE.. A value of $\text{MASK}(i) = 0$ indicates that the node corresponding to row or column i is not to be referenced. A value of $\text{MASK}(i) \neq 0$ indicates that the node corresponding to row or column i is to be referenced. In particular, rows and columns not referenced will not be permuted.

7: PERM(N) – INTEGER array *Output*

On exit: this will contain the permutation vector that describes the permutation matrix P for the reordering of the matrix A. The elements of the permutation matrix P are zero except for the unit elements in row i and column $\text{PERM}(i)$, $i = 1, 2, \ldots n$.

8: INFO(4) – INTEGER array *Output*

On exit: statistics about the matrix A and the permuted matrix. The quantities below are calculated using any masking in effect otherwise the value zero is returned.

INFO(1)
The bandwidth of the matrix A, if LOPTS(4) = .TRUE..

INFO(2)
The profile of the matrix A, if LOPTS(4) = .TRUE..

INFO(3)
The bandwidth of the permuted matrix PAP^{T}, if LOPTS(5) = .TRUE..

INFO(4)
The profile of the permuted matrix PAP^{T}, if LOPTS(5) = .TRUE..

9: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation 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 argument, 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, N $= \langle value \rangle$.
Constraint: N ≥ 1.

IFAIL $= 2$

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

IFAIL $= 3$

On entry, IROWIX($\langle value \rangle$) $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: $1 \leq$ IROWIX(i) $\leq$ N for all i.

IFAIL $= 4$

On entry, ICOLZP($\langle value \rangle$) $= \langle value \rangle$ and NNZ $= \langle value \rangle$.
Constraint: $1 \leq$ ICOLZP(i) $\leq$ NNZ for all i.

IFAIL $= 5$

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

On entry, ICOLZP(N $+ 1$) $= \langle value \rangle$ and NNZ $= \langle value \rangle$.
Constraint: ICOLZP(N $+ 1$) $=$ NNZ $+ 1$.

IFAIL $= 6$

On entry, the matrix A is not symmetric.
Element $(\langle value \rangle, \langle value \rangle)$ has no symmetric element.

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

Not applicable.

8 Parallelism and Performance

F11YEF is not threaded in any implementation.

9 Further Comments

The bandwidth for a matrix $A = (a_{ij})$ is defined as

$$b = \max_{ij} |i - j|, \quad i, j = 1, 2, \ldots, n \text{ s.t. } a_{ij} \neq 0.$$

The profile is defined as

$$p = \sum_{j=1}^{n} b_j, \quad \text{where } b_j = \max_{i} |i - j|, \quad i = 1, 2, \ldots n \text{ s.t. } a_{ij} \neq 0.$$

10 Example

This example reads the CCS representation of a real sparse matrix A and calls F11YEF to reorder the rows and columns and displays the results.

10.1 Program Text

```
    Program f11yefe

!     F11YEF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f11yef, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
      Logical, Parameter               :: plotdata = .False.
!     .. Local Scalars ..
      Integer                          :: ifail, n, nnz
      Logical                          :: bw_after, bw_before, check_sym,      &
                                          do_cm, use_mask
!     .. Local Arrays ..
      Integer, Allocatable             :: icolzp(:), irowix(:), mask(:),       &
                                          perm(:)
      Integer                          :: info(4)
      Logical                          :: lopts(5)
!     .. Executable Statements ..
      Write (nout,*) 'F11YEF Example Program Results'

!     Skip heading in data file

      Read (nin,*)

!     Size of the matrix
      Read (nin,*) n
```

```
!     Number of nonzero elements
      Read (nin,*) nnz

      Allocate (icolzp(n+1),irowix(nnz),mask(n),perm(n))

!     Read in data
      Read (nin,*) irowix(1:nnz)
      Read (nin,*) icolzp(1:n+1)

!     Set options
      use_mask = .False.
      do_cm = .False.
      check_sym = .True.
      bw_before = .True.
      bw_after = .True.
      lopts(1:5) = (/use_mask,do_cm,check_sym,bw_before,bw_after/)

!     ifail: behaviour on error exit
!     =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call f11yef(n,nnz,icolzp,irowix,lopts,mask,perm,info,ifail)

!     Print results
      Write (nout,'(A)') 'Permutation (perm):'
      Write (nout,99999) perm(1:n)

      Write (nout,*)
      Write (nout,'(A)') 'Statistics:'
      Write (nout,'(A,I6)') '  Before:  Bandwidth = ', info(1)
      Write (nout,'(A,I6)') '  Before:  Profile   = ', info(2)
      Write (nout,'(A,I6)') '  After :  Bandwidth = ', info(3)
      Write (nout,'(A,I6)') '  After :  Profile   = ', info(4)

!     Print matrix entries and permuted entries in form suitable for printing
      If (plotdata) Then
        Call plot(n,nnz)
      End If

99999 Format (6(4X,I3))

    Contains
      Subroutine uncompress(n,nnz,icolzp,icol)

!       .. Implicit None Statement ..
        Implicit None
!       .. Scalar Arguments ..
        Integer, Intent (In)           :: n, nnz
!       .. Array Arguments ..
        Integer, Intent (Out)          :: icol(nnz)
        Integer, Intent (In)           :: icolzp(n+1)
!       .. Local Scalars ..
        Integer                        :: col_beg, col_end, i
!       .. Executable Statements ..
        Do i = 1, n
          col_end = icolzp(i+1) - 1
          col_beg = icolzp(i)
          If (col_end>=col_beg) Then
            icol(col_beg:col_end) = i
          End If
        End Do
        Return
      End Subroutine uncompress
      Subroutine plot(n,nnz)
!       Put data, suitable for plotting matrix structure, in data file

!       .. Use Statements ..
        Use nag_library, Only: f11zaf
!       .. Implicit None Statement ..
        Implicit None
!       .. Scalar Arguments ..
        Integer, Intent (In)           :: n, nnz
```

```
!       .. Local Scalars ..
        Integer                       :: i, ifail, nnz2
!       .. Local Arrays ..
        Real (Kind=nag_wp), Allocatable :: a(:)
        Integer, Allocatable          :: icolix(:), ipcolix(:), iperm(:),     &
                                         iprowix(:), istr(:), iwork(:)
!       .. Executable Statements ..

        Allocate (icolix(nnz),ipcolix(nnz),iprowix(nnz))
        Allocate (iperm(n),a(nnz),istr(n+1),iwork(n))

!       Decompress icolzp to full set of column indices and invert perm
        Call uncompress(n,nnz,icolzp,icolix)
        Do i = 1, n
          iperm(perm(i)) = i
        End Do

!       Give some nice values, encoding original position
        Do i = 1, nnz
          a(i) = icolix(i)*.01_nag_wp + 1.0_nag_wp*irowix(i)
        End Do

!       Original matrix structure
        Write (*,'(I8,4X,I8,4X,F8.2)')(irowix(i),icolix(i),a(i),i=1,nnz)
        Write (*,'(/)')

!       Permute
        Do i = 1, nnz
          a(i) = icolix(i)*.01_nag_wp + 1.0_nag_wp*irowix(i)
          ipcolix(i) = iperm(icolix(i))
          iprowix(i) = iperm(irowix(i))
        End Do

!       Reorder (in exit: istr contains new CCS icolzp)
        nnz2 = nnz
        ifail = 0
        Call f11zaf(n,nnz2,a,ipcolix,iprowix,'F','K',istr,iwork,ifail)

!       Permuted matrix structure
        Write (*,'(I8,4X,I8,4X,F8.2)')(iprowix(i),ipcolix(i),a(i),i=1,nnz2)
        Deallocate (icolix,ipcolix,iprowix,iperm,a,istr,iwork)

        Return

      End Subroutine plot
    End Program f11yefe
```

10.2 Program Data

```
F11YEF Example Program Data
60                                  : n
180                                 : nnz

  2   5   6   1   3  11   2   4  16
  3   5  21   1   4  26   1   7  10
  6   8  30   7   9  42   8  10  38
  6   9  12   2  12  15  10  11  13
 12  14  37  13  15  33  11  14  17
  3  17  20  15  16  18  17  19  32
 18  20  53  16  19  22   4  22  25
 20  21  23  22  24  52  23  25  48
 21  24  27   5  27  30  25  26  28
 27  29  47  28  30  43   7  26  29
 32  35  54  18  31  33  14  32  34
 33  35  36  31  34  56  34  37  40
 13  36  38   9  37  39  38  40  41
 36  39  57  39  42  45   8  41  43
 29  42  44  43  45  46  41  44  58
 44  47  50  28  46  48  24  47  49
 48  50  51  46  49  59  49  52  55
```

```
 23  51  53  19  52  54  31  53  55
 51  54  60  35  57  60  40  56  58
 45  57  59  50  58  60  55  56  59  : irowix

  1    4    7   10   13   16
 19   22   25   28   31   34
 37   40   43   46   49   52
 55   58   61   64   67   70
 73   76   79   82   85   88
 91   94   97  100  103  106
109  112  115  118  121  124
127  130  133  136  139  142
145  148  151  154  157  160
163  166  169  172  175  178
181                           : icolzp
```

10.3 Program Results

```
 F11YEF Example Program Results
Permutation (perm):
     1      5      2      6      4      3
    26      7     30     11     12     10
    21     16     27     25      8     29
    17     15     13      9     22     20
    28     24     42     43     18     14
    37     38     23     19     47     48
    41     44     32     33     36     39
    52     53     46     49     45     31
    34     40     51     54     50     58
    35     57     55     59     56     60

Statistics:
  Before:  Bandwidth =     34
  Before:  Profile   =    490
  After :  Bandwidth =     10
  After :  Profile   =    458
```

Example Program
Figure 1 : Original Matrix Ordering

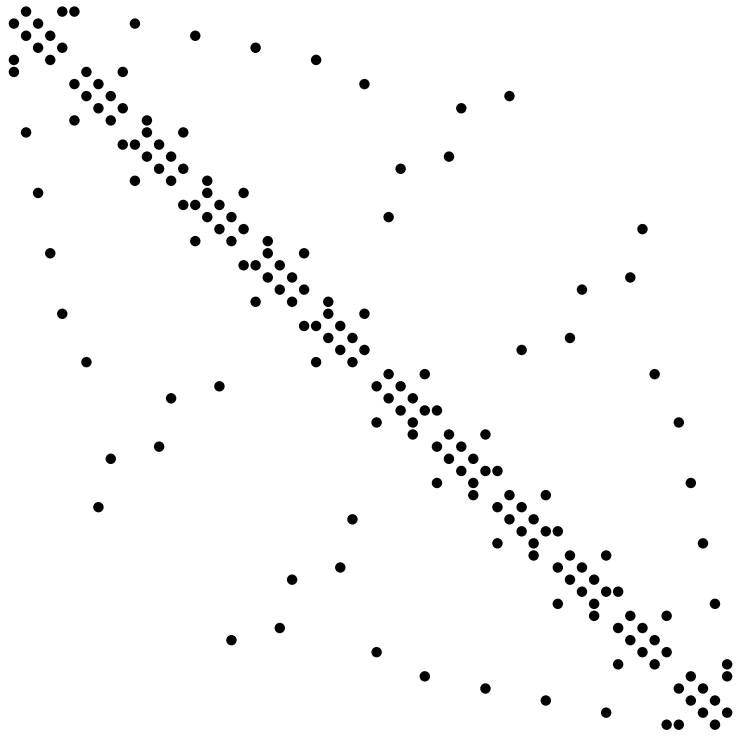

Figure 2 : Reverse Cuthill-McKee Reordering