NAG Toolbox

nag_sparse_sym_rcm (f11ye)

1 Purpose

nag_sparse_sym_rcm (f11ye) reduces the bandwidth of a sparse symmetric matrix stored in compressed column storage format using the Reverse Cuthill–McKee algorithm.

2 Syntax

```
[perm, info, ifail] = nag_sparse_sym_rcm(icolzp, irowix, lopts, mask, 'n', n, 'nnz', nnz)
[perm, info, ifail] = f11ye(icolzp, irowix, lopts, mask, 'n', n, 'nnz', nnz)
```

3 Description

nag_sparse_sym_rcm (f11ye) 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).

nag_sparse_sym_rcm (f11ye) 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 nag_sparse_sym_rcm (f11ye)) may require less storage space and/or less computational steps when solving (see Wai-Hung and Sherman (1976)).

nag_sparse_sym_rcm (f11ye) may be used prior to nag_sparse_real_symm_precon_ichol (f11ja) and nag_sparse_real_symm_precon_ichol_solve (f11jb) (see Section 10 in nag_sparse_real_symm_precon_ichol_solve (f11jb)).

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 Parameters

5.1 Compulsory Input Parameters

1: **icolzp**$(\mathbf{n}+\mathbf{1})$ – INTEGER array

icolzp records the index into **irowix** which starts each new column.

Constraints:

$1 \leq$ **icolzp**$(i) \leq$ **nnz** $+ 1$, for $i = 2, 3, \ldots,$ **n**;
icolzp$(1) = 1$;
icolzp$(\mathbf{n}+1) =$ **nnz** $+ 1$, where **icolzp**(i) holds the position integer for the starts of the columns in **irowix**.

2: **irowix**(**nnz**) – INTEGER array

The row indices corresponding to the nonzero elements in the matrix A.

Constraint: $1 \leq$ **irowix**$(i) \leq$ **n**, for $i = 1, 2, \ldots,$ **nnz**.

3: **lopts**(**5**) – LOGICAL array

The options to be used by nag_sparse_sym_rcm (f11ye).

lopts$(1) = true$
Row/column i of the matrix A will only be referenced if **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.

4: **mask**(:) – INTEGER array

The dimension of the array **mask** must be at least **n** if **lopts**$(1) = true$, and at least 0 otherwise

mask is only referenced if **lopts**(1) is *true*. A value of **mask**$(i) = 0$ indicates that the node corresponding to row or column i is not to be referenced. A value of **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.

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the first dimension of **icolzp** $- 1$

n, the order of the matrix A.

Constraint: **n** ≥ 1.

2: **nz** – INTEGER

Default: the dimension of the array **irowix**.

The number of nonzero elements in the matrix A.

Constraint: **nnz** ≥ 0.

5.3 Output Parameters

1: **perm**(**n**) – INTEGER array

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 **perm**(i), $i = 1, 2, \ldots n$.

2: **info**(**4**) – INTEGER array

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) $= true$.

3: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

Constraint: **n** ≥ 1.

ifail $= 2$

Constraint: **nnz** ≥ 1.

ifail $= 3$

Constraint: $1 \leq$ **irowix**$(i) \leq$ **n** for all i.

ifail $= 4$

Constraint: $1 \leq$ **icolzp**$(i) \leq$ **nnz** for all i.

ifail $= 5$

Constraint: **icolzp**$(1) = 1$.

Constraint: **icolzp**$(\mathbf{n}+1) =$ **nnz** $+ 1$.

ifail $= 6$

On entry, the matrix A is not symmetric.

ifail $= -99$

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

ifail $= -399$

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

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

Not applicable.

8 Further Comments

The bandwidth for a matrix $A = \left(a_{ij}\right)$ 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

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

9 Example

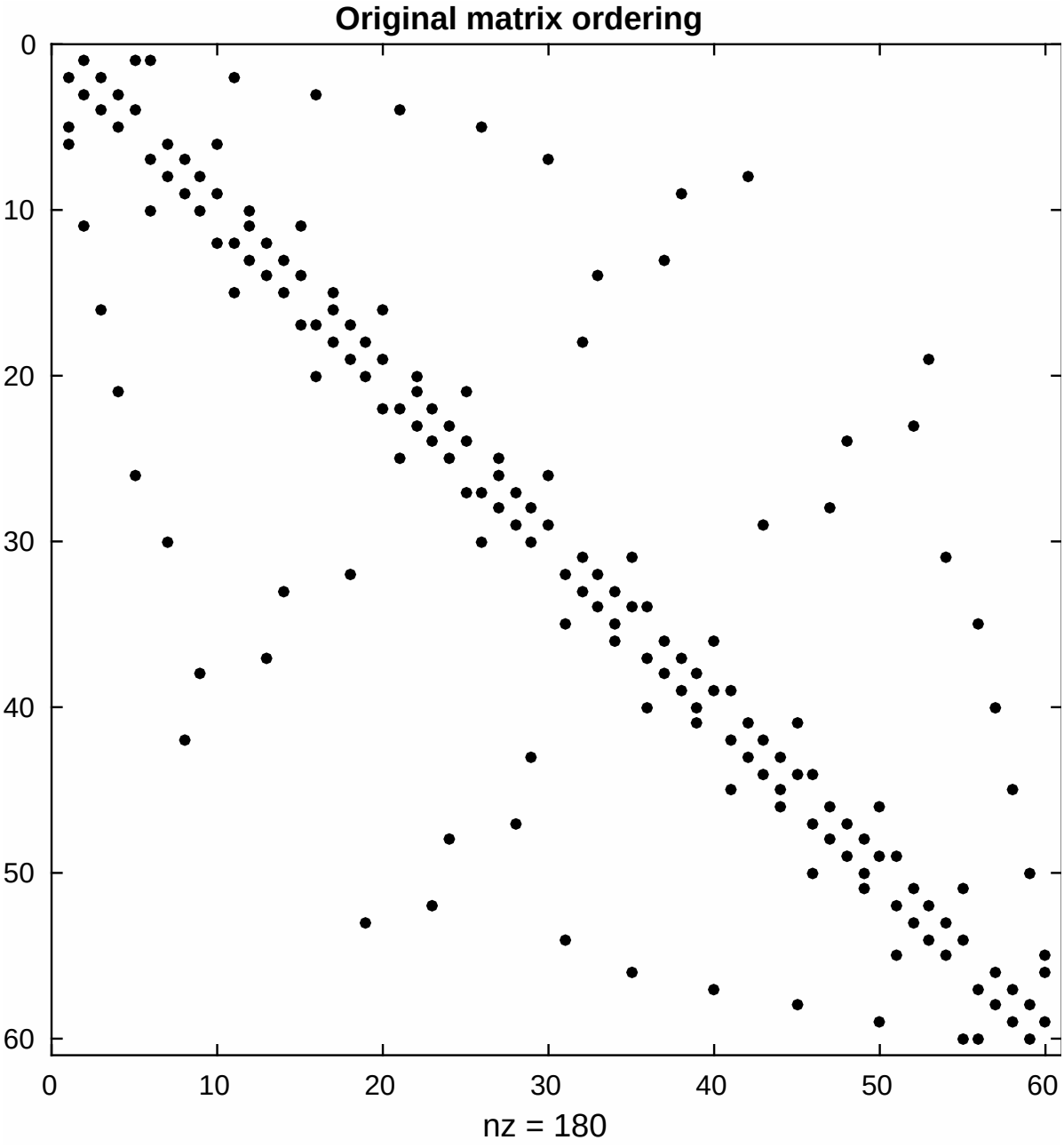
This example reads the CCS representation of a real sparse matrix A and calls nag_sparse_sym_rcm (f11ye) to reorder the rows and columns and displays the results.

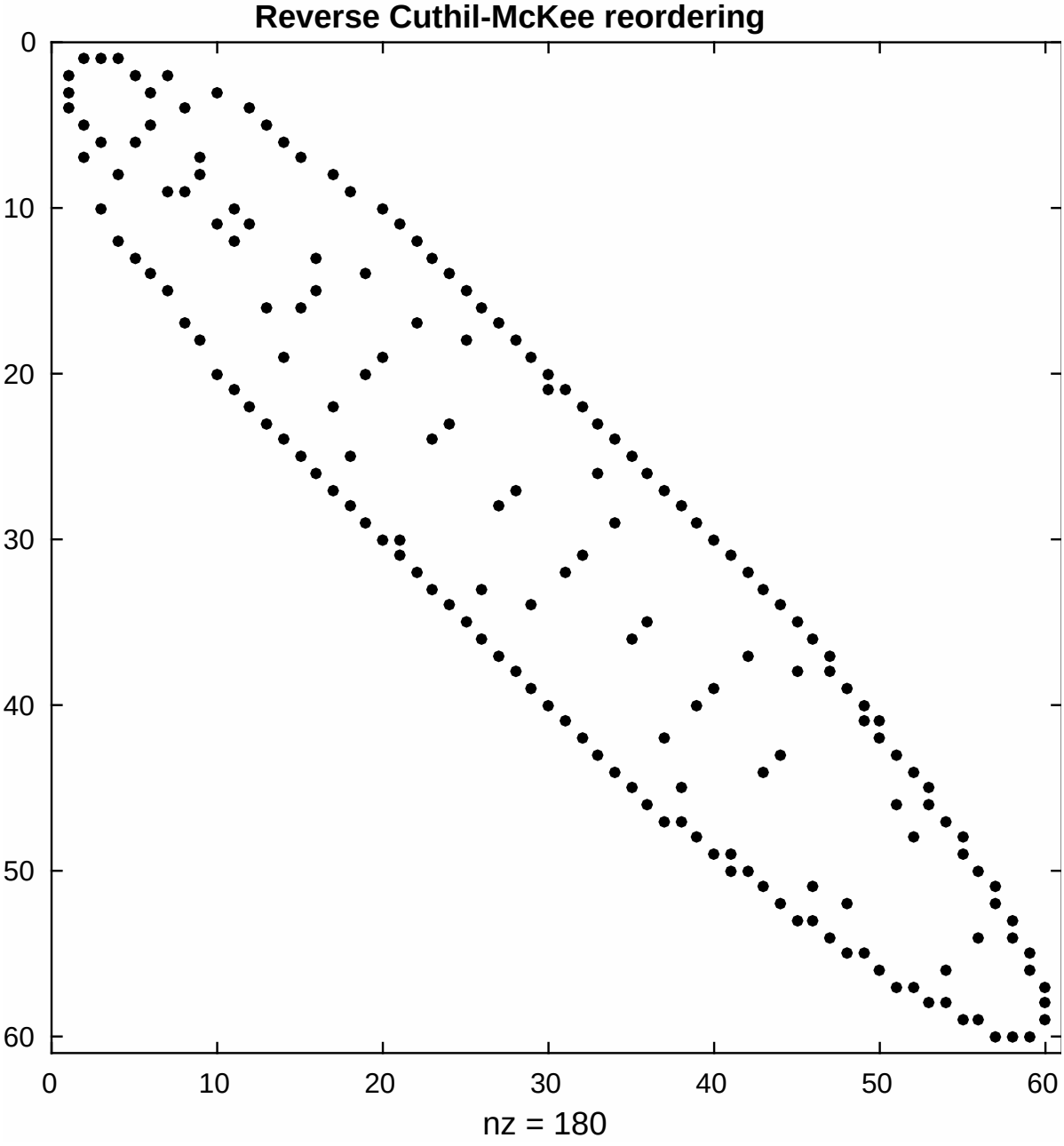
9.1 Program Text

```
function f11ye_example

fprintf('f11ye example results\n\n');

% Reduce the bandwidth of sparse symmetric matrix A

% Define entries of A using Bucky matrix
A = bucky;
[irow,icol,a] = find(A);

n = nag_int(size(A,1));
nz = nag_int(size(irow,1));

% Use f11za to get Compressed column storage array icolzp.
irow = nag_int(irow);
icol = nag_int(icol);
dup = 'S';
zero = 'K';
[nz, a, icol, irow, icolzp, ifail] = ...
  f11za(...
        n, nz, a, icol, irow, dup, zero);

% Perform bandwidth reduction
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];
mask       = [nag_int(0)];

[perm, info, ifail] = f11ye( ...
                           icolzp, irow, lopts, mask);

% Display results
disp('Permutation (perm):');
fprintf(' %4d %4d %4d %4d %4d %4d %4d %4d %4d %4d\n',perm)
fprintf('\nStatistics:\n');
fprintf('  Before:  Bandwidth = %6d\n', info(1));
fprintf('  Before:  Profile   = %6d\n', info(2));
fprintf('  After :  Bandwidth = %6d\n', info(3));
fprintf('  After :  Profile   = %6d\n', info(4));

% Permuted matrix
B = A(perm,perm);

fig1 = figure;
spy(A);
title('Original matrix ordering');
fig2 = figure;
spy(B);
title('Reverse Cuthil-McKee reordering');
```

9.2 Program Results

```
     f11ye example 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
```




Reverse Cuthil-McKee reordering
0
10
20
30
40
50
60
0
10
20
30
40
50
60
nz = 180