NAG Toolbox

nag_sparse_real_gen_precon_ilu (f11da)

1   Purpose

nag_sparse_real_gen_precon_ilu (f11da) computes an incomplete LU factorization of a real sparse nonsymmetric matrix, represented in coordinate storage format. This factorization may be used as a preconditioner in combination with nag_sparse_real_gen_basic_solver (f11be) or nag_sparse_real_gen_solve_ilu (f11dc).

2   Syntax

```
[a, irow, icol, ipivp, ipivq, istr, idiag, nnzc, npivm, ifail] =
nag_sparse_real_gen_precon_ilu(nnz, a, irow, icol, lfill, dtol, milu, ipivp,
ipivq, 'n', n, 'la', la, 'pstrat', pstrat)

[a, irow, icol, ipivp, ipivq, istr, idiag, nnzc, npivm, ifail] = f11da(nnz, a,
irow, icol, lfill, dtol, milu, ipivp, ipivq, 'n', n, 'la', la, 'pstrat', pstrat)
```

3   Description

nag_sparse_real_gen_precon_ilu (f11da) computes an incomplete LU factorization (see Meijerink and Van der Vorst (1977) and Meijerink and Van der Vorst (1981)) of a real sparse nonsymmetric n by n matrix A. The factorization is intended primarily for use as a preconditioner with one of the iterative solvers nag_sparse_real_gen_basic_solver (f11be) or nag_sparse_real_gen_solve_ilu (f11dc).

The decomposition is written in the form

$$A = M + R$$

where

$$M = PLDUQ$$

and L is lower triangular with unit diagonal elements, D is diagonal, U is upper triangular with unit diagonals, P and Q are permutation matrices, and R is a remainder matrix.

The amount of fill-in occurring in the factorization can vary from zero to complete fill, and can be controlled by specifying either the maximum level of fill **lfill**, or the drop tolerance **dtol**.

The argument **pstrat** defines the pivoting strategy to be used. 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 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 matrix M is returned in terms of the CS representation of the matrix

$$C = L + D^{-1} + U - 2I.$$

Further algorithmic details are given in Section 9.3.

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

Salvini S A and Shaw G J (1996) An evaluation of new NAG Library solvers for large sparse unsymmetric linear systems *NAG Technical Report TR2/96*

5   Parameters

5.1   Compulsory Input Parameters

1:     **nz** – INTEGER

The number of nonzero elements in the matrix A.

Constraint: $1 \leq \mathbf{nnz} \leq \mathbf{n}^2$.

2:     **a**(**la**) – REAL (KIND=nag_wp) array

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 function nag_sparse_real_gen_sort (f11za) may be used to order the elements in this way.

3:     **irow**(**la**) – INTEGER array
4:     **icol**(**la**) – INTEGER array

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 nag_sparse_real_gen_sort (f11za)):

$1 \leq \mathbf{irow}(i) \leq \mathbf{n}$ and $1 \leq \mathbf{icol}(i) \leq \mathbf{n}$, for $i = 1, 2, \ldots, \mathbf{nnz}$;
either $\mathbf{irow}(i-1) < \mathbf{irow}(i)$ or both $\mathbf{irow}(i-1) = \mathbf{irow}(i)$ and $\mathbf{icol}(i-1) < \mathbf{icol}(i)$, for $i = 2, 3, \ldots, \mathbf{nnz}$.

5:     **lfill** – INTEGER

If **lfill** ≥ 0 its value is the maximum level of fill allowed in the decomposition (see Section 9.2). A negative value of **lfill** indicates that **dtol** will be used to control the fill instead.

6:     **dtol** – REAL (KIND=nag_wp)

If **lfill** < 0, **dtol** is used as a drop tolerance to control the fill-in (see Section 9.2); otherwise **dtol** is not referenced.

Constraint: if **lfill** < 0, **dtol** ≥ 0.0.

7:     **milu** – CHARACTER(1)

Indicates whether or not the factorization should be modified to preserve row-sums (see Section 9.4).

milu = 'M'
        The factorization is modified.

milu = 'N'
        The factorization is not modified.

Constraint: **milu** = 'M' or 'N'.

8:   **ipivp**(**n**) – INTEGER array
9:   **ipivq**(**n**) – INTEGER array

If **pstrat** = 'U', then **ipivp**(k) and **ipivq**(k) must specify the row and column indices of the element used as a pivot at elimination stage k. Otherwise **ipivp** and **ipivq** need not be initialized.

Constraint: if **pstrat** = 'U', **ipivp** and **ipivq** must both hold valid permutations of the integers on [1,**n**].

5.2   Optional Input Parameters

1:   **n** – INTEGER

Default: the dimension of the arrays **ipivp**, **ipivq**. (An error is raised if these dimensions are not equal.)

n, the order of the matrix A.

Constraint: **n** ≥ 1.

2:   **la** – INTEGER

Default: the dimension of the arrays **a**, **irow**, **icol**. (An error is raised if these dimensions are not equal.)

The dimension of the arrays **a**, **irow** and **icol**. these arrays must be of sufficient size to store both A (**nnz** elements) and C (**nnzc** elements).

Constraint: **la** $\geq 2 \times$ **nnz**.

3:   **pstrat** – CHARACTER(1)

Suggested value: **pstrat** = 'C'.

Default: 'C'

Specifies the pivoting strategy to be adopted.

pstrat = 'N'
      No pivoting is carried out.

pstrat = 'U'
      Pivoting is carried out according to the user-defined input values of **ipivp** and **ipivq**.

pstrat = 'P'
      Partial pivoting by columns for stability is carried out.

pstrat = 'C'
      Complete pivoting by rows for sparsity, and by columns for stability, is carried out.

Constraint: **pstrat** = 'N', 'U', 'P' or 'C'.

5.3   Output Parameters

1:   **a**(**la**) – REAL (KIND=nag_wp) array

The first **nnz** entries of **a** contain the nonzero elements of A and the next **nnzc** entries contain the elements of the matrix C. Matrix elements are ordered by increasing row index, and by increasing column index within each row.

2:   **irow**(**la**) – INTEGER array
3:   **icol**(**la**) – INTEGER array

The row and column indices of the nonzero elements returned in **a**.

4:     **ipivp**(**n**) – INTEGER array

5:     **ipivq**(**n**) – INTEGER array

The pivot indices. If **ipivp**$(k) = i$ and **ipivq**$(k) = j$ then the element in row i and column j was used as the pivot at elimination stage k.

6:     **istr**(**n** + **1**) – INTEGER array

istr(i), for $i = 1, 2, \ldots, \mathbf{n}$, is the starting address in the arrays **a**, **irow** and **icol** of row i of the matrix C. **istr**$(\mathbf{n} + 1)$ is the address of the last nonzero element in C plus one.

7:     **idiag**(**n**) – INTEGER array

idiag(i), for $i = 1, 2, \ldots, \mathbf{n}$, holds the index of arrays **a**, **irow** and **icol** which holds the diagonal element in row i of the matrix C.

8:     **nnzc** – INTEGER

The number of nonzero elements in the matrix C.

9:     **npivm** – INTEGER

If **npivm** > 0 it gives the number of pivots which were modified during the factorization to ensure that M exists.

If **npivm** $= -1$ no pivot modifications were required, but a local restart occurred (see Section 9.3). The quality of the preconditioner will generally depend on the returned value of **npivm**.

If **npivm** is large the preconditioner may not be satisfactory. In this case it may be advantageous to call nag_sparse_real_gen_precon_ilu (f11da) again with an increased value of **lfill**, a reduced value of **dtol**, or set **pstrat** $=$ 'C'. See also Section 9.5.

10:    **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 $= 1$

On entry, $\mathbf{n} < 1$,
or $\mathbf{nnz} < 1$,
or $\mathbf{nnz} > \mathbf{n}^2$,
or $\mathbf{la} < 2 \times \mathbf{nnz}$,
or $\mathbf{lfill} < 0$ and $\mathbf{dtol} < 0.0$,
or $\mathbf{pstrat} \neq$ 'N', 'U', 'P' or 'C',
or $\mathbf{milu} \neq$ 'M' or 'N',
or $liwork < 7 \times \mathbf{n} + 2$.

ifail $= 2$

On entry, the arrays **irow** and **icol** fail to satisfy the following constraints:

$1 \leq \mathbf{irow}(i) \leq \mathbf{n}$ and $1 \leq \mathbf{icol}(i) \leq \mathbf{n}$, for $i = 1, 2, \ldots, \mathbf{nnz}$;

$\mathbf{irow}(i-1) < \mathbf{irow}(i)$ o r $\mathbf{irow}(i-1) = \mathbf{irow}(i)$ a n d $\mathbf{icol}(i-1) < \mathbf{icol}(i)$,   f o r $i = 2, 3, \ldots, \mathbf{nnz}$.

Therefore a nonzero element has been supplied which does not lie within the matrix A, is out of order, or has duplicate row and column indices. Call nag_sparse_real_gen_sort (f11za) to reorder and sum or remove duplicates.

ifail $= 3$

> On entry, **pstrat** $=$ 'U', but one or both of **ipivp** and **ipivq** does not represent a valid permutation of the integers in [1,**n**]. An input value of **ipivp** or **ipivq** is either out of range or repeated.

ifail $= 4$

> **la** is too small, resulting in insufficient storage space for fill-in elements. The decomposition has been terminated before completion. Either increase **la** or reduce the amount of fill by reducing **lfill**, or increasing **dtol**.

ifail $= 5$ (nag_sparse_real_gen_sort (f11za))

> A serious error has occurred in an internal call to the specified function. Check all function calls and array sizes. Seek expert help.

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

The accuracy of the factorization 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. The factorization can generally be made more accurate by increasing **lfill**, or by reducing **dtol** with **lfill** < 0.

If nag_sparse_real_gen_precon_ilu (f11da) is used in combination with nag_sparse_real_gen_basic_solver (f11be) or nag_sparse_real_gen_solve_ilu (f11dc), the more accurate the factorization the fewer iterations will be required. However, the cost of the decomposition will also generally increase.

8   Further Comments

8.1   Timing

The time taken for a call to nag_sparse_real_gen_precon_ilu (f11da) is roughly proportional to $(\mathbf{nnzc})^2/\mathbf{n}$.

8.2   Control of Fill-in

If **lfill** ≥ 0 the amount of fill-in occurring in the incomplete factorization is controlled by limiting the maximum **level** of fill-in to **lfill**. The original nonzero elements of A are defined to be of level 0. The fill level of a new nonzero location occurring during the factorization is defined as:

$$k = \max(k_\mathrm{e}, k_\mathrm{c}) + 1,$$

where k_e is the level of fill of the element being eliminated, and k_c is the level of fill of the element causing the fill-in.

If **lfill** < 0 the fill-in is controlled by means of the **drop tolerance dtol**. A potential fill-in element a_{ij} occurring in row i and column j will not be included if:

$$\left| a_{ij} \right| < \mathbf{dtol} \times \alpha,$$

where α is the maximum absolute value element in the matrix A.

For either method of control, any elements which are not included are discarded unless **milu** = 'M', in which case their contributions are subtracted from the pivot element in the relevant elimination row, to preserve the row-sums of the original matrix.

Should the factorization process break down a local restart process is implemented as described in Section 9.3. This will affect the amount of fill present in the final factorization.

8.3   Algorithmic Details

The factorization is constructed row by row. At each elimination stage a row index is chosen. In the case of complete pivoting this index is chosen in order to reduce fill-in. Otherwise the rows are treated in the order given, or some user-defined order.

The chosen row is copied from the original matrix A and modified according to those previous elimination stages which affect it. During this process any fill-in elements are either dropped or kept according to the values of **lfill** or **dtol**. In the case of a modified factorization (**milu** = 'M') the sum of the dropped terms for the given row is stored.

Finally the pivot element for the row is chosen and the multipliers are computed for this elimination stage. For partial or complete pivoting the pivot element is chosen in the interests of stability as the element of largest absolute value in the row. Otherwise the pivot element is chosen in the order given, or some user-defined order.

If the factorization breaks down because the chosen pivot element is zero, or there is no nonzero pivot available, a local restart recovery process is implemented. The modification of the given pivot row according to previous elimination stages is repeated, but this time keeping all fill. Note that in this case the final factorization will include more fill than originally specified by the user-supplied value of **lfill** or **dtol**. The local restart usually results in a suitable nonzero pivot arising. The original criteria for dropping fill-in elements is then resumed for the next elimination stage (hence the **local** nature of the restart process). Should this restart process also fail to produce a nonzero pivot element an arbitrary unit pivot is introduced in an arbitrarily chosen column. nag_sparse_real_gen_precon_ilu (f11da) returns an integer argument **npivm** which gives the number of these arbitrary unit pivots introduced. If no pivots were modified but local restarts occurred **npivm** $= -1$ is returned.

8.4   Choice of Arguments

There is unfortunately no choice of the various algorithmic arguments which is optimal for all types of matrix, and some experimentation will generally be required for each new type of matrix encountered.

If the matrix A is not known to have any particular special properties the following strategy is recommended. Start with **lfill** $= 0$ and **pstrat** = 'C'. If the value returned for **npivm** is significantly larger than zero, i.e., a large number of pivot modifications were required to ensure that M existed, the preconditioner is not likely to be satisfactory. In this case increase **lfill** until **npivm** falls to a value close to zero.

If A has non-positive off-diagonal elements, is nonsingular, and has only non-negative elements in its inverse, it is called an 'M-matrix'. It can be shown that no pivot modifications are required in the incomplete LU factorization of an M-matrix (see Meijerink and Van der Vorst (1977)). In this case a good preconditioner can generally be expected by setting **lfill** $= 0$, **pstrat** = 'N' and **milu** = 'M'.

Some illustrations of the application of nag_sparse_real_gen_precon_ilu (f11da) to linear systems arising from the discretization of two-dimensional elliptic partial differential equations, and to random-valued randomly structured linear systems, can be found in Salvini and Shaw (1996).

8.5   Direct Solution of Sparse Linear Systems

Although it is not their primary purpose nag_sparse_real_gen_precon_ilu (f11da) and nag_sparse_real_gen_precon_ilu_solve (f11db) may be used together to obtain a **direct** solution to a nonsingular sparse linear system. To achieve this the call to nag_sparse_real_gen_precon_ilu_solve (f11db) should be preceded by a **complete** LU factorization

$$A = PLDUQ = M.$$

a complete factorization is obtained from a call to nag_sparse_real_gen_precon_ilu (f11da) with **lfill** < 0 and **dtol** $= 0.0$, provided **npivm** ≤ 0 on exit. A positive value of **npivm** indicates that A is singular, or ill-conditioned. A factorization with positive **npivm** may serve as a preconditioner, but will not result in a direct solution. It is therefore **essential** to check the output value of **npivm** if a direct solution is required.

The use of nag_sparse_real_gen_precon_ilu (f11da) and nag_sparse_real_gen_precon_ilu_solve (f11db) as a direct method is illustrated in Section 10 in nag_sparse_real_gen_precon_ilu_solve (f11db).

9   Example

This example reads in a sparse matrix A and calls nag_sparse_real_gen_precon_ilu (f11da) to compute an incomplete LU factorization. It then outputs the nonzero elements of both A and $C = L + D^{-1} + U - 2I$.

The call to nag_sparse_real_gen_precon_ilu (f11da) has **lfill** $= 0$, and **pstrat** $= $ 'C', giving an unmodified zero-fill LU factorization, with row pivoting for sparsity and column pivoting for stability.

9.1   Program Text

```
    function f11da_example

fprintf('f11da example results\n\n');

% Solve sparse system Ax = b iteratively using ILU preconditioner

% Define sparse matrix A and RHS B
n    = nag_int(8);
nz   = nag_int(24);
a    = zeros(3*nz, 1);
irow = zeros(3*nz, 1, nag_int_name);
icol = irow;
irow(1:nz) = nag_int(...
          [1; 1; 1; 2; 2; 2; 3; 3; 4; 4; 4; 4;
           5; 5; 5; 6; 6; 6; 7; 7; 7; 8; 8; 8]);
icol(1:nz) = nag_int(...
          [1; 4; 8; 1; 2; 5; 3; 6; 1; 3; 4; 7;
           2; 5; 7; 1; 3; 6; 3; 5; 7; 2; 6; 8]);
a(1:nz)    = [2;-1; 1; 4;-3; 2;-7; 2; 3;-4; 5; 5;
             -1; 8;-3;-6; 5; 2;-5;-1; 6;-1; 2; 3];
b          = [6; 8;-9;46;17;21;22;34];

% ILU preconditioner
%     Input parameters
lfill = nag_int(0);
dtol  = 0;
milu  = 'No modification';
ipivp = zeros(n, 1, nag_int_name);
ipivq = zeros(n, 1, nag_int_name);
% Compute preconditioner
[a, irow, icol, ipivp, ipivq, istr, idiag, nnzc, npivm, ifail] = ...
   f11da(...
         nz, a, irow, icol, lfill, dtol, milu, ipivp, ipivq);

% Iterative Solution Setup
%     Input parameters
method = 'BICGSTAB';
precon = 'Preconditioned';
lpoly  = nag_int(1);
tol    = sqrt(x02aj);
maxitn = nag_int(20);
anorm  = 0;
sigmax = 0;
monit  = nag_int(1);
lwork  = nag_int(6000);
% Setup iterative method
[lwreq, work, ifail] = ...
```

```
     f11bd(...
          method, precon, n, lpoly, tol, maxitn, anorm, sigmax, ...
          monit, lwork, 'norm_p', '1');

% Reverse communication loop calling iterative solver
irevcm = nag_int(0);
u      = zeros(8, 1);
v      = b;
wgt    = zeros(8, 1);

while (irevcm ~= 4)

  [irevcm, u, v, work, ifail] = f11be(...
                                      irevcm, u, v, wgt, work);

  if (irevcm == -1)
    % v = A^Tu
    [v, ifail] = f11xa(...
                       'T', a(1:nz), irow(1:nz), icol(1:nz), 'N', u);
  elseif (irevcm == 1)
    % v = Au
    [v, ifail] = f11xa(...
                       'N', a(1:nz), irow(1:nz), icol(1:nz), 'N', u);
  elseif (irevcm == 2)
    % Solve ILU v = u
    [v, ifail] = f11db('N', a, irow, icol, ipivp, ipivq, istr, idiag, 'N', u);
    if (ifail ~= 0)
      irevcm = 6;
    end
  elseif (irevcm == 3)
    % Monitoring stage
    [itn, stplhs, stprhs, anorm, sigmax, ifail] = ...
      f11bf(work);
    fprintf('\nMonitoring at iteration number %2d\n',itn);
    fprintf('residual norm:                %14.4e\n', stplhs);
    fprintf('\n   Solution Vector  Residual Vector\n');
    for i = 1:n
      fprintf('%16.4f %16.2e\n', u(i), v(i));
    end
  end
end

% Get information about the computation
[itn, stplhs, stprhs, anorm, sigmax, ifail] = ...
      f11bf(work);

fprintf('\nNumber of iterations for convergence:    %4d\n', itn);
fprintf('Residual norm:                           %14.4e\n', stplhs);
fprintf('Right-hand side of termination criteria: %14.4e\n', stprhs);
fprintf('i-norm of matrix a:                      %14.4e\n', anorm);
fprintf('\n   Solution Vector  Residual Vector\n');
for i = 1:n
  fprintf('%16.4f %16.2e\n', u(i), v(i));
end
```

9.2   Program Results

```
     f11da example results

Monitoring at iteration number  1
residual norm:                1.4059e+02

   Solution Vector  Residual Vector
        -4.5858          1.53e+01
         1.0154          2.66e+01
        -2.2234         -8.75e+00
         6.0097          1.86e+01
         1.3827          8.28e+00
        -7.9070          2.04e+01
         0.4427          9.61e+00
```

```
      5.9248            3.31e+01

Monitoring at iteration number  2
residual norm:                 3.2742e+01

   Solution Vector  Residual Vector
         4.1642          -2.96e+00
         4.9370          -5.55e+00
         4.8101           8.21e-01
         5.4324          -1.68e+01
         5.8531           5.60e-01
        11.9250          -1.91e+00
         8.4826           1.01e+00
         6.0625          -3.10e+00

Number of iterations for convergence:       3
Residual norm:                              1.0373e-08
Right-hand side of termination criteria:    5.8900e-06
i-norm of matrix a:                         1.1000e+01

   Solution Vector  Residual Vector
         1.0000          -1.36e-09
         2.0000          -2.61e-09
         3.0000           2.25e-10
         4.0000          -3.22e-09
         5.0000           6.30e-10
         6.0000          -5.24e-10
         7.0000           9.58e-10
         8.0000          -8.49e-10
```