NAG Toolbox

nag_matop_real_gen_sparse_lu_reuse (f01bs)

1 Purpose

nag_matop_real_gen_sparse_lu_reuse (f01bs) factorizes a real sparse matrix using the pivotal sequence previously obtained by nag_matop_real_gen_sparse_lu (f01br) when a matrix of the same sparsity pattern was factorized.

2 Syntax

```
[a, w, rpmin, ifail] = nag_matop_real_gen_sparse_lu_reuse(n, a, ivect, jvect, icn, ikeep, grow, idisp, 'nz', nz, 'licn', licn, 'eta', eta, 'abort', abort)
[a, w, rpmin, ifail] = f01bs(n, a, ivect, jvect, icn, ikeep, grow, idisp, 'nz', nz, 'licn', licn, 'eta', eta, 'abort', abort)
```

3 Description

nag_matop_real_gen_sparse_lu_reuse (f01bs) accepts as input a real sparse matrix of the same sparsity pattern as a matrix previously factorized by a call of nag_matop_real_gen_sparse_lu (f01br). It first applies to the matrix the same permutations as were used by nag_matop_real_gen_sparse_lu (f01br), 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 nag_linsys_real_sparse_fac_solve (f04ax) to solve sparse systems of linear equations $Ax = b$ or $A^{\mathrm{T}}x = b$.

nag_matop_real_gen_sparse_lu_reuse (f01bs) is much faster than nag_matop_real_gen_sparse_lu (f01br) and in some applications it is expected that there will be many calls of nag_matop_real_gen_sparse_lu_reuse (f01bs) for each call of nag_matop_real_gen_sparse_lu (f01br).

The method is fully described in Duff (1977).

A more recent algorithm for the same calculation is provided by nag_sparse_direct_real_gen_lu (f11me).

4 References

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

5 Parameters

5.1 Compulsory Input Parameters

1: **n** – INTEGER

n, the order of the matrix A.

Constraint: $\mathbf{n} > 0$.

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

a(i), for $i = 1, 2, \ldots,$ **nz**, must contain the nonzero elements of the sparse matrix A. They can be in any order since nag_matop_real_gen_sparse_lu_reuse (f01bs) will reorder them.

3: **ivect**(**nz**) – INTEGER array

4: **jvect**(**nz**) – INTEGER array

ivect(i) and **jvect**(i), for $i = 1, 2, \ldots,$ **nz**, must contain the row index and the column index respectively of the nonzero element stored in **a**(i).

5: **icn**(**licn**) – INTEGER array

icn contains, on entry, the same information as output by nag_matop_real_gen_sparse_lu (f01br). It must not be changed by you between a call of nag_matop_real_gen_sparse_lu_reuse (f01bs) and a call of nag_linsys_real_sparse_fac_solve (f04ax).

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

6: **ikeep**(**5** × **n**) – INTEGER array

The same indexing information about the factorization as output in **ikeep** by nag_matop_real_gen_sparse_lu (f01br).

You must **not** change **ikeep** between a call of nag_matop_real_gen_sparse_lu_reuse (f01bs) and subsequent calls to nag_linsys_real_sparse_fac_solve (f04ax).

7: **grow** – LOGICAL

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).

8: **idisp**(**2**) – INTEGER array

idisp(1) and **idisp**(2) must be as output in **idisp** by the previous call of nag_matop_real_gen_sparse_lu (f01br).

5.2 Optional Input Parameters

1: **nz** – INTEGER

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

The number of nonzero elements in the matrix A.

Constraint: **nz** > 0.

2: **licn** – INTEGER

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

The dimension of the arrays **a** and **icn**. it should have the same value as it had for nag_matop_real_gen_sparse_lu (f01br).

Constraint: **licn** $\geq$ **nz**.

3: **eta** – REAL (KIND=nag_wp)

Suggested value: **eta** $= 10^{-4}$.

Default: 0.0001

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.

4: **abort** – LOGICAL

Suggested value: **abort** $= true$.

Default: $true$

If **abort** $= true$, nag_matop_real_gen_sparse_lu_reuse (f01bs) exits immediately (with **ifail** $= 8$) if it finds duplicate elements in the input matrix.

If **abort** $= false$, nag_matop_real_gen_sparse_lu_reuse (f01bs) 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 nag_file_set_unit_advisory (x04ab)), unless suppressed by the value of **ifail** on entry.

5.3 Output Parameters

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

The nonzero elements in the LU factorization. The array must **not** be changed by you between a call of nag_matop_real_gen_sparse_lu_reuse (f01bs) and a call of nag_linsys_real_sparse_fac_solve (f04ax).

2: **w**(**n**) – REAL (KIND=nag_wp) array

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.

3: **rpmin** – REAL (KIND=nag_wp)

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 nag_matop_real_gen_sparse_lu (f01br).

4: **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, **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 nag_matop_real_gen_sparse_lu (f01br) (see Section 9).

ifail $= 6$

The input matrix is numerically singular.

ifail $= 7$ (*warning*)

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.

ifail $= -399$

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

ifail $= -999$

Dynamic memory allocation failed.

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 $\mathbf{w}(1)$ if **grow** $= true$, and m_{ij} the number of Gaussian elimination operations applied to element (i, j).

If $\rho = \mathbf{w}(1)$ is very large or **rpmin** is very small, then a fresh call of nag_matop_real_gen_sparse_lu (f01br) is recommended.

8 Further Comments

If you have a sequence of problems with the same sparsity pattern then nag_matop_real_gen_sparse_lu_reuse (f01bs) is recommended after nag_matop_real_gen_sparse_lu (f01br) 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 nag_matop_real_gen_sparse_lu (f01br).

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 function to be entered with a matrix having the same pattern of nonzeros as was earlier presented to nag_matop_real_gen_sparse_lu (f01br). 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.

9 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 $\mathbf{w}(1)$ and **rpmin** returned by nag_matop_real_gen_sparse_lu_reuse (f01bs). Normally the calls of nag_matop_real_gen_sparse_lu (f01br) and nag_matop_real_gen_sparse_lu_reuse (f01bs) would be followed by calls of nag_linsys_real_sparse_fac_solve (f04ax).

9.1 Program Text

```
function f01bs_example

fprintf('f01bs example results\n\n');

% Define sparse matrix A
n  = nag_int(6);
nz = nag_int(15);
z  = nag_int(15);
a  = zeros(150,1);
a(1:15)   = [5;  2;-1;2;  3;  -2;1;1;  -1;-1;2;-3;  -1;-1;6];

irn = zeros(75,1,nag_int_name);
icn = zeros(150,1,nag_int_name);
irn(1:15) = [nag_int(1); 2;2;2; 3; 4;4;4; 5;5;5;5; 6;6;6];
icn(1:15) = [nag_int(1); 2;3;4; 3; 1;4;5; 1;4;5;6; 1;2;6];

abort     = [true;     true;     false;     true];

% second matrix has same sparsity
ivect = irn(1:15);
jvect = icn(1:15);
grow = true;

[a, irn, icn, ikeep, w, idisp, ifail] = ...
f01br( ...
       n, nz, a, irn, icn, abort);

fprintf('For first matrix:\nValue of w(1)  = %7.4f\n\n', w(1));

% overwrite nz elements of A by second matrix with same sparsity
a(1:15) = [10; 12;-3;-1; 15; -2;10;-1; -1;-5;1;-1; -1;-2;6 ];
eta = 0.1;
[a, w, rpmin, ifail] = f01bs( ...
                             n, a, ivect, jvect, icn, ikeep, grow, idisp, ...
                             'eta',eta);

fprintf('For second matrix:\nValue of w(1)  = %7.4f\n\n', w(1));
fprintf('Value of rpmin = %7.4f\n', rpmin);
```

9.2 Program Results

```
     f01bs example results

For first matrix:
Value of w(1)  = 18.0000

For second matrix:
Value of w(1)  = 51.0000

Value of rpmin =  0.1000
```