NAG Toolbox

nag_sparse_complex_herm_precon_ilu_solve (f11jp)

1 Purpose

nag_sparse_complex_herm_precon_ilu_solve (f11jp) solves a system of complex linear equations involving the incomplete Cholesky preconditioning matrix generated by nag_sparse_complex_herm_precon_ilu (f11jn).

2 Syntax

```
[x, ifail] = nag_sparse_complex_herm_precon_ilu_solve(a, irow, icol, ipiv, istr, check, y, 'n', n, 'la', la)
[x, ifail] = f11jp(a, irow, icol, ipiv, istr, check, y, 'n', n, 'la', la)
```

3 Description

nag_sparse_complex_herm_precon_ilu_solve (f11jp) solves a system of linear equations

$$Mx = y$$

involving the preconditioning matrix $M = PLDL^{\mathrm{H}}P^{\mathrm{T}}$, corresponding to an incomplete Cholesky decomposition of a complex sparse Hermitian matrix stored in symmetric coordinate storage (SCS) format (see Section 2.1.2 in the F11 Chapter Introduction), as generated by nag_sparse_complex_herm_precon_ilu (f11jn).

In the above decomposition L is a complex lower triangular sparse matrix with unit diagonal, D is a real diagonal matrix and P is a permutation matrix. L and D are supplied to nag_sparse_complex_herm_precon_ilu_solve (f11jp) through the matrix

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

which is a lower triangular n by n complex sparse matrix, stored in SCS format, as returned by nag_sparse_complex_herm_precon_ilu (f11jn). The permutation matrix P is returned from nag_sparse_complex_herm_precon_ilu (f11jn) via the array **ipiv**.

nag_sparse_complex_herm_precon_ilu_solve (f11jp) may also be used in combination with nag_sparse_complex_herm_precon_ilu (f11jn) to solve a sparse complex Hermitian positive definite system of linear equations directly (see nag_sparse_complex_herm_precon_ilu (f11jn)). This is illustrated in Section 10.

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

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

The values returned in the array **a** by a previous call to nag_sparse_complex_herm_precon_ilu (f11jn).

2: **irow**(**la**) – INTEGER array
3: **icol**(**la**) – INTEGER array
4: **ipiv**(**n**) – INTEGER array
5: **istr**(**n** + **1**) – INTEGER array

The values returned in arrays **irow**, **icol**, **ipiv** and **istr** by a previous call to nag_sparse_complex_herm_precon_ilu (f11jn).

6: **check** – CHARACTER(1)

Specifies whether or not the input data should be checked.

check = 'C'
Checks are carried out on the values of **n**, **irow**, **icol**, **ipiv** and **istr**.

check = 'N'
None of these checks are carried out.

Constraint: **check** = 'C' or 'N'.

7: **y**(**n**) – COMPLEX (KIND=nag_wp) array

The right-hand side vector y.

5.2 Optional Input Parameters

1: **n** – INTEGER

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

n, the order of the matrix M. This **must** be the same value as was supplied in the preceding call to nag_sparse_complex_herm_precon_ilu (f11jn).

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**. this **must** be the same value supplied in the preceding call to nag_sparse_complex_herm_precon_ilu (f11jn).

5.3 Output Parameters

1: **x**(**n**) – COMPLEX (KIND=nag_wp) array

The solution vector x.

2: **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, **check** $\neq$ 'C' or 'N'.

ifail = 2

On entry, **n** < 1.

ifail $= 3$

On entry, the SCS representation of the preconditioning matrix M is invalid. Further details are given in the error message. Check that the call to nag_sparse_complex_herm_precon_ilu_solve (f11jp) has been preceded by a valid call to nag_sparse_complex_herm_precon_ilu (f11jn) and that the arrays **a**, **irow**, **icol**, **ipiv** and **istr** have not been corrupted between the two calls.

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 computed solution x is the exact solution of a perturbed system of equations $(M + \delta M)x = y$, where

$$|\delta M| \leq c(n)\epsilon P|L||D|\left|L^{\mathrm{H}}\right|P^{\mathrm{T}},$$

$c(n)$ is a modest linear function of n, and ϵ is the ***machine precision***.

8 Further Comments

8.1 Timing

The time taken for a call to nag_sparse_complex_herm_precon_ilu_solve (f11jp) is proportional to the value of **nnzc** returned from nag_sparse_complex_herm_precon_ilu (f11jn).

9 Example

This example reads in a complex sparse Hermitian positive definite matrix A and a vector y. It then calls nag_sparse_complex_herm_precon_ilu (f11jn), with **lfill** $= -1$ and **dtol** $= 0.0$, to compute the **complete** Cholesky decomposition of A:

$$A = PLDL^{\mathrm{H}}P^{\mathrm{T}}.$$

Finally it calls nag_sparse_complex_herm_precon_ilu_solve (f11jp) to solve the system

$$PLDL^{\mathrm{H}}P^{\mathrm{T}}x = y.$$

9.1 Program Text

```
function f11jp_example

fprintf('f11jp example results\n\n');

% Solve sparse Hermitian system Ax = b using CG method with
% Incomplete Cholesky preconditioning (IC)

% Define A and b
n  = nag_int(9);
nz = nag_int(23);
a    = zeros(3*nz,1);
irow = zeros(3*nz, 1, nag_int_name);
icol = zeros(3*nz, 1, nag_int_name);
a(1:nz) = [ 6 + 0.i; -1 + 1.i;  6 + 0.i;  0 + 1.i;
            5 + 0.i;  5 + 0.i;  2 - 2.i;  4 + 0.i;
            1 + 1.i;  2 + 0.i;  6 + 0.i; -4 + 3.i;
```

```
            0 + 1.i; -1 + 0.i;  6 + 0.i; -1 - 1.i;
            0 - 1.i;  9 + 0.i;  1 + 3.i;  1 + 2.i;
           -1 + 0.i;  1 + 4.i;  9 + 0.i];
b       = [ 8 + 54i;-10 - 92i; 25 + 27i; 26 - 28i;
           54 + 12i; 26 - 22i; 47 + 65i; 71 - 57i;
           60 + 70i];
irow(1:nz) = nag_int([1;2;2;3;3;4;5;5;6;6;6;7;7;7;7;8;8;8;9;9;9;9;9]);
icol(1:nz) = nag_int([1;1;2;2;3;4;1;5;3;4;6;2;5;6;7;4;6;8;1;5;6;8;9]);

% Setup IC factorization
lfill  = nag_int(0);
dtol   = 0;
mic    = 'N';
dscale = 0;
ipiv   = zeros(n, 1, nag_int_name);

[a, irow, icol, ipiv, istr, nnzc, npivm, ifail] = ...
  f11jn( ...
         nz, a, irow, icol, lfill, dtol, mic, dscale, ipiv);

% Iterative method setup
method = 'CG    ';
precon = 'Preconditioned';
tol    = (x02aj)^(3/8);
maxitn = nag_int(20);
anorm  = 0;
sigmax = 0;
maxits = nag_int(9);
monit  = nag_int(2);

[lwreq, work, ifail] = ...
  f11gr( ...
         method, precon, nag_int(n), tol, maxitn, anorm, sigmax, ...
         maxits, monit, 'sigcmp', 's', 'norm_p', '1');

% Reverse communication loop calling f11ge
irevcm = nag_int(0);
u      = complex(zeros(n,1));
v      = b;
wgt    = zeros(n,1);

while (irevcm ~= 4)
  [irevcm, u, v, work, ifail] = ...
    f11gs( ...
           irevcm, u, v, wgt, work);

  if (irevcm == 1)
    % v = Au
    [v, ifail] = f11xs( ...
                        a(1:nz), irow(1:nz), icol(1:nz), 'N', u);
  elseif (irevcm == 2)
    % Solve (IC)v = u
    [v, ifail] = f11jp( ...
                        a, irow, icol, ipiv, istr, 'N', u);
  elseif (irevcm == 3)
    % Monitoring
    [itn, stplhs, stprhs, anorm, sigmax, its, sigerr, ifail] = ...
        f11gt(work);
    fprintf('\nMonitoring at iteration number %2d\n',itn);
    fprintf('residual norm:              %14.4e\n', stplhs);
    fprintf('\n   Solution Vector\n');
    disp(u);
    fprintf('\n   Residual Vector\n');
    disp(v);
  end
end

% Get information about the computation
[itn, stplhs, stprhs, anorm, sigmax, its, sigerr, ifail] = ...
    f11gt(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\n');
disp(u);
fprintf('\n   Residual Vector\n');
disp(v);
```

9.2 Program Results

```
     f11jp example results

Monitoring at iteration number  2
residual norm:                  1.4937e+01

   Solution Vector
   0.2142 + 4.5333i
  -1.6589 -12.6722i
   2.4101 + 7.4551i
   4.4400 - 6.4174i
   9.1135 + 3.7812i
   4.4419 - 4.0382i
   1.4757 + 1.2662i
   8.4872 - 3.5347i
   5.9948 + 0.9685i

   Residual Vector
  -1.8370 + 3.6956i
  -0.6501 + 0.2546i
  -0.1262 - 0.1362i
  -0.1312 + 0.1413i
  -1.1471 + 0.7339i
  -0.5505 - 1.0535i
   1.7165 - 1.4614i
  -0.3583 + 0.2876i
  -0.3028 - 0.3532i

Monitoring at iteration number  4
residual norm:                  1.4602e+00

   Solution Vector
   1.0061 + 8.9847i
   1.9637 - 7.9768i
   3.0067 + 7.0285i
   3.9830 - 5.9636i
   5.0390 + 5.0432i
   6.0488 - 4.0771i
   6.9710 + 3.0168i
   8.0118 - 1.9806i
   9.0074 + 0.9646i

   Residual Vector
   0.0115 - 0.0282i
   0.0135 - 0.1734i
   0.0182 + 0.0196i
   0.0189 - 0.0204i
  -0.0909 - 0.1090i
  -0.2389 + 0.3244i
   0.1903 - 0.0155i
   0.0516 - 0.0414i
   0.0436 + 0.0509i

Number of iterations for convergence:       5
Residual norm:                              9.0594e-14
Right-hand side of termination criteria:    2.8433e-03
i-norm of matrix a:                         2.2000e+01

   Solution Vector
   1.0000 + 9.0000i
```

```
   2.0000 - 8.0000i
   3.0000 + 7.0000i
   4.0000 - 6.0000i
   5.0000 + 5.0000i
   6.0000 - 4.0000i
   7.0000 + 3.0000i
   8.0000 - 2.0000i
   9.0000 + 1.0000i

   Residual Vector
   1.0e-13 *

  -0.0178 + 0.0000i
   0.0355 - 0.2842i
  -0.0355 + 0.0355i
   0.0355 - 0.0711i
  -0.0711 + 0.0355i
  -0.0711 + 0.0000i
   0.0000 + 0.0000i
   0.0000 - 0.0711i
   0.0000 - 0.1421i
```