NAG Toolbox for MATLAB

f11dr

1 Purpose

f11dr solves a system of linear equations involving the preconditioning matrix corresponding to SSOR applied to a complex sparse non-Hermitian matrix, represented in co-ordinate storage format.

2 Syntax

```
[x, ifail] = f11dr(trans, a, irow, icol, rdiag, omega, check, y, 'n', n, 'nnz', nnz)
```

3 Description

f11dr solves a system of linear equations

$$Mx = y, \qquad \text{or} \qquad M^{\mathrm{H}}x = y,$$

according to the value of the parameter **trans**, where the matrix

$$M = \frac{1}{\omega(2-\omega)}(D+\omega L)D^{-1}(D+\omega U)$$

corresponds to symmetric successive-over-relaxation (SSOR) Young 1971 applied to a linear system $Ax = b$, where A is a complex sparse non-Hermitian matrix stored in co-ordinate storage (CS) format (see Section 2.1.1 in the F11 Chapter Introduction).

In the definition of M given above D is the diagonal part of A, L is the strictly lower triangular part of A, U is the strictly upper triangular part of A, and ω is a user-defined relaxation parameter.

It is envisaged that a common use of f11dr will be to carry out the preconditioning step required in the application of f11bs to sparse linear systems. For an illustration of this use of f11dr see the example program given in Section 9. f11dr is also used for this purpose by the Black Box function f11ds.

4 References

Young D 1971 *Iterative Solution of Large Linear Systems* Academic Press, New York

5 Parameters

5.1 Compulsory Input Parameters

1: **trans – string**

Specifies whether or not the matrix M is transposed.

trans = 'N'

$Mx = y$ is solved.

trans = 'T'

$M^{\mathrm{H}}x = y$ is solved.

Constraint: **trans** = 'N' or 'T'.

2: **a**(**nnz**) **– complex 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 f11zn may be used to order the elements in this way.

3: **irow**(**nnz**) – **int32 array**

4: **icol**(**nnz**) – **int32 array**

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

Constraints:

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

5: **rdiag**(**n**) – **complex array**

The elements of the diagonal matrix D^{-1}, where D is the diagonal part of A.

6: **omega** – **double scalar**

The relaxation parameter ω.

Constraint: $0.0 < \mathbf{omega} < 2.0$.

7: **check** – **string**

Specifies whether or not the CS representation of the matrix M should be checked.

check = 'C'

Checks are carried on the values of **n**, **nnz**, **irow**, **icol** and **omega**.

check = 'N'

None of these checks are carried out.

See also Section 8.2.

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

8: **y**(**n**) – **complex array**

The right-hand side vector y.

5.2 Optional Input Parameters

1: **n** – **int32 scalar**

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

n, the order of the matrix A.

Constraint: $\mathbf{n} \geq 1$.

2: **nnz** – **int32 scalar**

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

the number of nonzero elements in the matrix A.

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

5.3 Input Parameters Omitted from the MATLAB Interface

iwork

5.4 Output Parameters

1: **x(n) – complex array**

The solution vector x.

2: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, **trans** $\neq$ 'N' or 'T',
or **check** $\neq$ 'C' or 'N'.

ifail $= 2$

On entry, **n** < 1,
or **nnz** < 1,
or **nnz** $> \mathbf{n}^2$,
or **omega** lies outside the interval $(0.0, 2.0)$,

ifail $= 3$

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 $i = 1, 2, \ldots, \mathbf{nnz}$;

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

Therefore a nonzero element has been supplied which does not lie in the matrix A, is out of order, or has duplicate row and column indices. Call f11zn to reorder and sum or remove duplicates.

ifail $= 4$

On entry, the matrix A has a zero diagonal element. The SSOR preconditioner is not appropriate for this problem.

7 Accuracy

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

$$|\delta M| \leq c(n)\epsilon |D + \omega L| \left|D^{-1}\right| |D + \omega U|,$$

$c(n)$ is a modest linear function of n, and ϵ is the ***machine precision***. An equivalent result holds when **trans** $=$ 'T'.

8 Further Comments

8.1 Timing

The time taken for a call to f11dr is proportional to **nnz**.

8.2 Use of check

It is expected that a common use of f11dr will be to carry out the preconditioning step required in the application of f11bs to sparse linear systems. In this situation f11dr is likely to be called many times with

the same matrix M. In the interests of both reliability and efficiency, you are recommended to set **check** to 'C' for the first of such calls, and to 'N' for all subsequent calls.

9 Example

```
trans = 'N';
a = [complex(2, +3);
     complex(1, -1);
     complex(-1, +0);
     complex(0, +2);
     complex(-2, +1);
     complex(1, +0);
     complex(0, -1);
     complex(5, +4);
     complex(3, -1);
     complex(1, +0);
     complex(-2, +2);
     complex(-3, +1);
     complex(0, +3);
     complex(4, -2);
     complex(-2, +0);
     complex(-6, +1)];
irow = [int32(1);
     int32(1);
     int32(1);
     int32(2);
     int32(2);
     int32(2);
     int32(3);
     int32(3);
     int32(3);
     int32(3);
     int32(4);
     int32(4);
     int32(4);
     int32(5);
     int32(5);
     int32(5)];
icol = [int32(1);
     int32(2);
     int32(4);
     int32(2);
     int32(3);
     int32(5);
     int32(1);
     int32(3);
     int32(4);
     int32(5);
     int32(1);
     int32(4);
     int32(5);
     int32(2);
     int32(3);
     int32(5)];
rdiag = [complex(0.1538461538461539, -0.2307692307692308);
     complex(0, -0.5);
     complex(0.1219512195121951, -0.0975609756097561);
     complex(-0.3, -0.1);
     complex(-0.1621621621621622, -0.02702702702702703)];
omega = 1.4;
check = 'C';
y = [complex(-3, +3);
     complex(-11, +5);
     complex(23, +48);
     complex(-41, +2);
     complex(-28, -31)];
[x, ifail] = f11dr(trans, a, irow, icol, rdiag, omega, check, y)
```

```
x =
  -0.9414 + 0.6309i
  -3.3782 + 3.6938i
   2.3340 + 0.6792i
  -0.5814 + 4.1560i
   4.0239 + 6.8504i
ifail =
           0
```