NAG Toolbox for MATLAB

f11dk

1 Purpose

f11dk computes the **approximate** solution of a real, symmetric or nonsymmetric, sparse system of linear equations applying a number of Jacobi iterations. It is expected that f11dk will be used as a preconditioner for the iterative solution of real sparse systems of equations.

2 Syntax

```
[x, diag, ifail] = f11dk(store, trans, init, niter, a, irow, icol,
check, b, diag, 'n', n, 'nnz', nnz)
```

3 Description

f11dk computes the **approximate** solution of the real sparse system of linear equations $Ax = b$ using **niter** iterations of the Jacobi algorithm (see also Golub and Van Loan 1996 and Young 1971):

$$x_{k+1} = x_k + D^{-1}(b - Ax_k) \qquad (1)$$

where $k = 1, \ldots,$ **niter** and $x_0 = 0$.

f11dk can be used both for nonsymmetric and symmetric systems of equations. For symmetric matrices, either all nonzero elements of the matrix A can be supplied using co-ordinate storage (CS), or only the nonzero elements of the lower triangle of A, using symmetric co-ordinate storage (SCS) (see the F11 Chapter Introduction).

It is expected that f11dk will be used as a preconditioner for the iterative solution of real sparse systems of equations, using either the suite comprising the functions f11gd, f11ge and f11gf, for symmetric systems, or the suite comprising the functions f11bd, f11be and f11bf, for nonsymmetric systems of equations.

4 References

Golub G H and Van Loan C F 1996 *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

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

5 Parameters

5.1 Compulsory Input Parameters

1: **store – string**

Specifies whether the matrix A is stored using symmetric co-ordinate storage (SCS) (applicable only to a symmetric matrix A) or co-ordinate storage (CS) (applicable to both symmetric and nonsymmetric matrices).

store = 'N'

The complete matrix A is stored in CS format.

store = 'S'

The lower triangle of the symmetric matrix A is stored in SCS format.

Constraint: **store** = 'N' or 'S'.

2: **trans – string**

If **store** = 'N', specifies whether the approximate solution of $Ax = b$ or of $A^{\mathrm{T}}x = b$ is required.

trans = 'N'

The approximate solution of $Ax = b$ is calculated.

trans = 'T'

The approximate solution of $A^{\mathrm{T}}x = b$ is calculated.

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

Suggested value: if the matrix A is symmetric and stored in CS format, it is recommended that **trans** = 'N' for reasons of efficiency.

3: **init – string**

On first entry, **init** should be set to 'I', unless the diagonal elements of A are already stored in the array **diag**. Otherwise, if **diag** already contains the diagonal of A, it can be set to 'N'.

init = 'N'

diag must contain the diagonal of A.

init = 'I'

diag will store the diagonal of A on exit.

Constraint: **init** = 'N' or 'I'.

Suggested value: **init** = 'I' on first entry; **init** = 'N', subsequently, unless **diag** has been overwritten.

4: **niter – int32 scalar**

The number of Jacobi iterations requested.

Constraint: **niter** > 0.

5: **a(nnz) – double array**

If **store** = 'N', the nonzero elements in the matrix A (CS format).

If **store** = 'S', the nonzero elements in the lower triangle of the matrix A (SCS format).

In both cases, the elements of either A or of its lower triangle must be ordered by increasing row index and by increasing column index within each row. Multiple entries for the same row and columns indices are not permitted. The function f11za or f11zb may be used to reorder the elements in this way for CS and SCS storage, respectively.

6: **irow(nnz) – int32 array**
7: **icol(nnz) – int32 array**

If **store** = 'N', the row and column indices of the nonzero elements supplied in **a**.

If **store** = 'S', the row and column indices of the nonzero elements of the lower triangle of the matrix A supplied in **a**.

Constraints:

$1 \leq \mathbf{irow}(i) \leq \mathbf{n}$, for $i = 1, 2, \ldots, \mathbf{nnz}$;
if **store** = 'N', $1 \leq \mathbf{icol}(i) \leq \mathbf{n}$, for $i = 1, 2, \ldots, \mathbf{nnz}$;
if **store** = 'S', $1 \leq \mathbf{icol}(i) \leq \mathbf{irow}(i)$, 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 $i = 2, 3, \ldots, \mathbf{nnz}$.

8: **check – string**

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

check = 'C'

Checks are carried out on the values of **n**, **nnz**, **irow**, **icol**; if **init** = 'N', **diag** is also checked.

check = 'N'

None of these checks are carried out.

See also Section 8.2.

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

9: **b(n) – double array**

The right-hand side vector b.

10: **diag(n) – double array**

If **init** = 'N', the diagonal elements of A.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the arrays **b**, **x**, **diag**. (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.)

if **store** = 'N', the number of nonzero elements in the matrix A.

If **store** = 'S', the number of nonzero elements in the lower triangle of the matrix A.

Constraints:

if **store** = 'N', $1 \leq \mathbf{nnz} \leq \mathbf{n}^2$;
if **store** = 'S', $1 \leq \mathbf{nnz} \leq [\mathbf{n}(\mathbf{n}+1)]/2$.

5.3 Input Parameters Omitted from the MATLAB Interface

work

5.4 Output Parameters

1: **x(n) – double array**

The approximate solution vector $x_{\mathbf{niter}}$.

2: **diag(n) – double array**

If **init** = 'N', unchanged on exit.

If **init** = 'I', the diagonal elements of A.

3: **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, **store** $\neq$ 'N' or 'S',
or **trans** $\neq$ 'N' or 'T',
or **init** $\neq$ 'N' or 'I',
or **check** $\neq$ 'C' or 'N',
or **niter** ≤ 0.

ifail $= 2$

On entry, **n** < 1,
or **nnz** < 1,
or **nnz** $> \mathbf{n}^2$, if **store** = 'N',
or $1 \leq \mathbf{nnz} \leq [\mathbf{n}(\mathbf{n}+1)]/2$, if **store** = 'S'.

ifail $= 3$

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

$1 \leq \mathbf{irow}(i) \leq \mathbf{n}$ and

if **store** = 'N', then $1 \leq \mathbf{icol}(i) \leq \mathbf{n}$, or
if **store** = 'S', then $1 \leq \mathbf{icol}(i) \leq \mathbf{irow}(i)$, 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 within the matrix A, is out of order, or has duplicate row and column indices. Call either f11za or f11zb to reorder and sum or remove duplicates when **store** = 'N' or **store** = 'S', respectively.

ifail $= 4$

On entry, **init** = 'N' and some diagonal elements of A stored in **diag** are zero.

ifail $= 5$

On entry, **init** = 'I' and some diagonal elements of A are zero.

7 Accuracy

In general, the Jacobi method cannot be used on its own to solve systems of linear equations. The rate of convergence is bound by its spectral properties (see, for example, Golub and Van Loan 1996) and as a solver, the Jacobi method can only be applied to a limited set of matrices. One condition that guarantees convergence is strict diagonal dominance.

However, the Jacobi method can be used successfully as a preconditioner to a wider class of systems of equations. The Jacobi method has good vector/parallel properties, hence it can be applied very efficiently. Unfortunately, it is not possible to provide criteria which define the applicability of the Jacobi method as a preconditioner, and its usefulness must be judged for each case.

8 Further Comments

8.1 Timing

The time taken for a call to f11dk is proportional to **niter** $\times$ **nnz**.

8.2 Use of check

It is expected that a common use of f11dk will be as preconditioner for the iterative solution of real, symmetric or nonsymmetric, linear systems. In this situation, f11dk is likely to be called many times. In the interests of both reliability and efficiency, you are recommended to set **check** = 'C' for the first of such calls, and to set **check** = 'N' for all subsequent calls.

9 Example

```
store = 'Non symmetric';
trans = 'N';
init = 'I';
niter = int32(4);
a = [4;
     -1;
     1;
     4;
     -5;
     2;
     -7;
     2;
     2;
     -1;
     6;
     2;
     -1;
     8;
     -2;
     -2;
     5;
     8;
     -2;
     -1;
     7;
     -1;
     2;
     6];
irow = [int32(1);
     int32(1);
     int32(1);
     int32(2);
     int32(2);
     int32(2);
     int32(3);
     int32(3);
     int32(4);
     int32(4);
     int32(4);
     int32(4);
     int32(5);
     int32(5);
     int32(5);
     int32(6);
     int32(6);
     int32(6);
     int32(7);
     int32(7);
     int32(7);
     int32(8);
     int32(8);
     int32(8)];
icol = [int32(1);
     int32(4);
     int32(8);
     int32(1);
     int32(2);
```

```
     int32(5);
     int32(3);
     int32(6);
     int32(1);
     int32(3);
     int32(4);
     int32(7);
     int32(2);
     int32(5);
     int32(7);
     int32(1);
     int32(3);
     int32(6);
     int32(3);
     int32(5);
     int32(7);
     int32(2);
     int32(6);
     int32(8)];
check = 'Check';
b = [6;
     8;
     -9;
     46;
     17;
     21;
     22;
     34];
diag = [0;
     0;
     0;
     0;
     0;
     0;
     0;
     0];
[x, diagOut, ifail] = f11dk(store, trans, init, niter, a, irow, icol,
check, b, diag)
```

```
x =
    1.7646
    1.2155
    1.8151
    5.9643
    3.2885
    1.9172
    4.1373
    5.2298
diagOut =
     4
    -5
    -7
     6
     8
     8
     7
     6
ifail =
           0
```