

NAG Library Routine Document

F11JSF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details.

1 Purpose

F11JSF solves a complex sparse Hermitian system of linear equations, represented in symmetric coordinate storage format, using a conjugate gradient or Lanczos method, without preconditioning, with Jacobi or with SSOR preconditioning.

2 Specification

```

SUBROUTINE F11JSF (METHOD, PRECON, N, NNZ, A, IROW, ICOL, OMEGA, B, TOL,      &
                  MAXITN, X, RNORM, ITN, RDIAG, WORK, LWORK, IWORK, IFAIL)
INTEGER           N, NNZ, IROW(NNZ), ICOL(NNZ), MAXITN, ITN, LWORK,      &
                  IWORK(N+1), IFAIL
REAL (KIND=nag_wp) OMEGA, TOL, RNORM, RDIAG(N)
COMPLEX (KIND=nag_wp) A(NNZ), B(N), X(N), WORK(LWORK)
CHARACTER(*)      METHOD
CHARACTER(1)      PRECON

```

3 Description

F11JSF solves a complex sparse Hermitian linear system of equations

$$Ax = b,$$

using a preconditioned conjugate gradient method (see Barrett *et al.* (1994)), or a preconditioned Lanczos method based on the algorithm SYMMLQ (see Paige and Saunders (1975)). The conjugate gradient method is more efficient if A is positive definite, but may fail to converge for indefinite matrices. In this case the Lanczos method should be used instead. For further details see Barrett *et al.* (1994).

F11JSF allows the following choices for the preconditioner:

- no preconditioning;
- Jacobi preconditioning (see Young (1971));
- symmetric successive-over-relaxation (SSOR) preconditioning (see Young (1971)).

For incomplete Cholesky (IC) preconditioning see F11JQF.

The matrix A is represented in symmetric coordinate storage (SCS) format (see Section 2.1.2 in the F11 Chapter Introduction) in the arrays A, IROW and ICOL. The array A holds the nonzero entries in the lower triangular part of the matrix, while IROW and ICOL hold the corresponding row and column indices.

4 References

Barrett R, Berry M, Chan T F, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C and Van der Vorst H (1994) *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* SIAM, Philadelphia

Paige C C and Saunders M A (1975) Solution of sparse indefinite systems of linear equations *SIAM J. Numer. Anal.* **12** 617–629

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

5 Parameters

- 1: METHOD – CHARACTER(*) *Input*
On entry: specifies the iterative method to be used.
 METHOD = 'CG'
 Conjugate gradient method.
 METHOD = 'SYMMLQ'
 Lanczos method (SYMMLQ).
Constraint: METHOD = 'CG' or 'SYMMLQ'.
- 2: PRECON – CHARACTER(1) *Input*
On entry: specifies the type of preconditioning to be used.
 PRECON = 'N'
 No preconditioning.
 PRECON = 'J'
 Jacobi.
 PRECON = 'S'
 Symmetric successive-over-relaxation (SSOR).
Constraint: PRECON = 'N', 'J' or 'S'.
- 3: N – INTEGER *Input*
On entry: n , the order of the matrix A .
Constraint: $N \geq 1$.
- 4: NNZ – INTEGER *Input*
On entry: the number of nonzero elements in the lower triangular part of the matrix A .
Constraint: $1 \leq \text{NNZ} \leq N \times (N + 1)/2$.
- 5: A(NNZ) – COMPLEX (KIND=nag_wp) array *Input*
On entry: the nonzero elements of the lower triangular part of 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 routine F11ZPF may be used to order the elements in this way.
- 6: IROW(NNZ) – INTEGER array *Input*
 7: ICOL(NNZ) – INTEGER array *Input*
On entry: the row and column indices of the nonzero elements supplied in array A .
Constraints:
 IROW and ICOL must satisfy these constraints (which may be imposed by a call to F11ZPF):
 $1 \leq \text{IROW}(i) \leq N$ and $1 \leq \text{ICOL}(i) \leq \text{IROW}(i)$, for $i = 1, 2, \dots, \text{NNZ}$;
 $\text{IROW}(i - 1) < \text{IROW}(i)$ or $\text{IROW}(i - 1) = \text{IROW}(i)$ and $\text{ICOL}(i - 1) < \text{ICOL}(i)$, for $i = 2, 3, \dots, \text{NNZ}$.
- 8: OMEGA – REAL (KIND=nag_wp) *Input*
On entry: if PRECON = 'S', OMEGA is the relaxation parameter ω to be used in the SSOR method. Otherwise OMEGA need not be initialized.
Constraint: $0.0 < \text{OMEGA} < 2.0$.

- 9: B(N) – COMPLEX (KIND=nag_wp) array Input
On entry: the right-hand side vector b .
- 10: TOL – REAL (KIND=nag_wp) Input
On entry: the required tolerance. Let x_k denote the approximate solution at iteration k , and r_k the corresponding residual. The algorithm is considered to have converged at iteration k if
- $$\|r_k\|_\infty \leq \tau \times (\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty).$$
- If $\text{TOL} \leq 0.0$, $\tau = \max(\sqrt{\epsilon}, \sqrt{n}\epsilon)$ is used, where ϵ is the *machine precision*. Otherwise $\tau = \max(\text{TOL}, 10\epsilon, \sqrt{n}\epsilon)$ is used.
Constraint: $\text{TOL} < 1.0$.
- 11: MAXITN – INTEGER Input
On entry: the maximum number of iterations allowed.
Constraint: $\text{MAXITN} \geq 1$.
- 12: X(N) – COMPLEX (KIND=nag_wp) array Input/Output
On entry: an initial approximation to the solution vector x .
On exit: an improved approximation to the solution vector x .
- 13: RNORM – REAL (KIND=nag_wp) Output
On exit: the final value of the residual norm $\|r_k\|$, where k is the output value of ITN.
- 14: ITN – INTEGER Output
On exit: the number of iterations carried out.
- 15: RDIAG(N) – REAL (KIND=nag_wp) array Output
On exit: the elements of the diagonal matrix D^{-1} , where D is the diagonal part of A . Note that since A is Hermitian the elements of D^{-1} are necessarily real.
- 16: WORK(LWORK) – COMPLEX (KIND=nag_wp) array Workspace
 17: LWORK – INTEGER Input
On entry: the dimension of the array WORK as declared in the (sub)program from which F11JSF is called.
Constraints:
- if METHOD = 'CG', LWORK $\geq 6 \times N + 120$;
 if METHOD = 'SYMMLQ', LWORK $\geq 7 \times N + 120$.
- 18: IWORK(N + 1) – INTEGER array Workspace
- 19: IFAIL – INTEGER Input/Output
On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.
- For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**
- On exit:* IFAIL = 0 unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry $IFAIL = 0$ or -1 , explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

$IFAIL = 1$

On entry, $METHOD \neq 'CG'$ or $'SYMMMLQ'$,
 or $PRECON \neq 'N', 'J'$ or $'S'$,
 or $N < 1$,
 or $NNZ < 1$,
 or $NNZ > N \times (N + 1)/2$,
 or $OMEGA$ lies outside the interval $(0.0, 2.0)$,
 or $TOL \geq 1.0$,
 or $MAXITN < 1$,
 or $LWORK$ is too small.

$IFAIL = 2$

On entry, the arrays $IROW$ and $ICOL$ fail to satisfy the following constraints:

$1 \leq IROW(i) \leq N$ and $1 \leq ICOL(i) \leq IROW(i)$, for $i = 1, 2, \dots, NNZ$;

$IROW(i - 1) < IROW(i)$, or $IROW(i - 1) = IROW(i)$ and $ICOL(i - 1) < ICOL(i)$, for $i = 2, 3, \dots, NNZ$.

Therefore a nonzero element has been supplied which does not lie in the lower triangular part of A , is out of order, or has duplicate row and column indices. Call F11ZPF to reorder and sum or remove duplicates.

$IFAIL = 3$

On entry, the matrix A has a zero diagonal element. Jacobi and SSOR preconditioners are not appropriate for this problem.

$IFAIL = 4$

The required accuracy could not be obtained. However, a reasonable accuracy has been obtained and further iterations could not improve the result.

$IFAIL = 5$

Required accuracy not obtained in $MAXITN$ iterations.

$IFAIL = 6$

The preconditioner appears not to be positive definite.

$IFAIL = 7$

The matrix of the coefficients appears not to be positive definite (conjugate gradient method only).

$IFAIL = 8$

A serious error has occurred in an internal call to an auxiliary routine. Check all subroutine calls and array sizes. Seek expert help.

$IFAIL = 9$

The matrix of the coefficients has a non-real diagonal entry, and is therefore not Hermitian.

7 Accuracy

On successful termination, the final residual $r_k = b - Ax_k$, where $k = \text{ITN}$, satisfies the termination criterion

$$\|r_k\|_\infty \leq \tau \times (\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty).$$

The value of the final residual norm is returned in RNORM.

8 Further Comments

The time taken by F11JSF for each iteration is roughly proportional to NNZ. One iteration with the Lanczos method (SYMMLQ) requires a slightly larger number of operations than one iteration with the conjugate gradient method.

The number of iterations required to achieve a prescribed accuracy cannot easily be determined *a priori*, as it can depend dramatically on the conditioning and spectrum of the preconditioned matrix of the coefficients $\bar{A} = M^{-1}A$.

9 Example

This example solves a complex sparse Hermitian positive definite system of equations using the conjugate gradient method, with SSOR preconditioning.

9.1 Program Text

```

Program f11jsfe

!      F11JSF Example Program Text

!      Mark 24 Release. NAG Copyright 2012.

!      .. Use Statements ..
Use nag_library, Only: f11jsf, nag_wp
!      .. Implicit None Statement ..
Implicit None
!      .. Parameters ..
Integer, Parameter          :: nin = 5, nout = 6
!      .. Local Scalars ..
Real (Kind=nag_wp)         :: omega, rnorm, tol
Integer                    :: i, ifail, itn, lwork, maxitn, n, nnz
Character (6)              :: method
Character (1)              :: precon
!      .. Local Arrays ..
Complex (Kind=nag_wp), Allocatable :: a(:), b(:), work(:), x(:)
Real (Kind=nag_wp), Allocatable   :: rdiag(:)
Integer, Allocatable           :: icol(:), irow(:), iwork(:)
!      .. Executable Statements ..
Write (nout,*) 'F11JSF Example Program Results'
!      Skip heading in data file
Read (nin,*)

!      Read algorithmic parameters

Read (nin,*) n
Read (nin,*) nnz
lwork = 7*n + 120
Allocate (a(nnz),b(n),work(lwork),x(n),rdiag(n),icol(nnz),irow(nnz), &
         iwork(n+1))
Read (nin,*) method, precon
Read (nin,*) omega
Read (nin,*) tol, maxitn

!      Read the matrix A

Do i = 1, nnz
  Read (nin,*) a(i), irow(i), icol(i)

```

```

      End Do

!      Read rhs vector b and initial approximate solution x

      Read (nin,*) b(1:n)
      Read (nin,*) x(1:n)

!      Solve Ax = b using F11JSF

!      ifail: behaviour on error exit
!      =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call f11jsf(method,precon,n,nnz,a,irow,icol,omega,b,tol,maxitn,x,rnorm, &
        itn,rdiag,work,lwork,iwork,ifail)

      Write (nout,99999) 'Converged in', itn, ' iterations'
      Write (nout,99998) 'Final residual norm =', rnorm

!      Output x

      Write (nout,99997) x(1:n)

99999 Format (1X,A,I10,A)
99998 Format (1X,A,1P,E16.3)
99997 Format (1X,'(',E16.4,',',E16.4,')')
      End Program f11jsfe

```

9.2 Program Data

F11JSF Example Program Data

```

  9          N
 23         NNZ
'CG' 'SSOR' METHOD, PRECON
 1.1       OMEGA
1.0D-6 100 TOL, MAXITN
( 6., 0.)  1   1
(-1., 1.)  2   1
( 6., 0.)  2   2
( 0., 1.)  3   2
( 5., 0.)  3   3
( 5., 0.)  4   4
( 2.,-2.)  5   1
( 4., 0.)  5   5
( 1., 1.)  6   3
( 2., 0.)  6   4
( 6., 0.)  6   6
(-4., 3.)  7   2
( 0., 1.)  7   5
(-1., 0.)  7   6
( 6., 0.)  7   7
(-1.,-1.)  8   4
( 0.,-1.)  8   6
( 9., 0.)  8   8
( 1., 3.)  9   1
( 1., 2.)  9   5
(-1., 0.)  9   6
( 1., 4.)  9   8
( 9., 0.)  9   9      A(I), IROW(I), ICOL(I), I=1,...,NNZ
( 8., 54.)
(-10., -92.)
( 25., 27.)
( 26., -28.)
( 54., 12.)
( 26., -22.)
( 47., 65.)
( 71., -57.)
( 60., 70.)          B(I), I=1,...,N
( 0., 0.)
( 0., 0.)
( 0., 0.)

```

```
( 0., 0.)  
( 0., 0.)  
( 0., 0.)  
( 0., 0.)  
( 0., 0.)  
( 0., 0.)  
X(I), I=1,...,N
```

9.3 Program Results

```
F11JSF Example Program Results  
Converged in      7 iterations  
Final residual norm =      1.477E-05  
( 0.1000E+01, 0.9000E+01)  
( 0.2000E+01, -0.8000E+01)  
( 0.3000E+01, 0.7000E+01)  
( 0.4000E+01, -0.6000E+01)  
( 0.5000E+01, 0.5000E+01)  
( 0.6000E+01, -0.4000E+01)  
( 0.7000E+01, 0.3000E+01)  
( 0.8000E+01, -0.2000E+01)  
( 0.9000E+01, 0.1000E+01)
```
