NAG Library Routine Document

F04ASF

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

F04ASF calculates the accurate solution of a set of real symmetric positive definite linear equations with a single right-hand side, $Ax = b$, using a Cholesky factorization and iterative refinement.

2 Specification

```
SUBROUTINE F04ASF (A, LDA, B, N, C, WK1, WK2, IFAIL)
INTEGER            LDA, N, IFAIL
REAL (KIND=nag_wp) A(LDA,*), B(max(1,N)), C(max(1,N)), WK1(max(1,N)),      &
                   WK2(max(1,N))
```

3 Description

Given a set of real linear equations $Ax = b$, where A is a symmetric positive definite matrix, F04ASF first computes a Cholesky factorization of A as $A = LL^{\mathrm{T}}$ where L is lower triangular. An approximation to x is found by forward and backward substitution. The residual vector $r = b - Ax$ is then calculated using ***additional precision*** and a correction d to x is found by solving $LL^{\mathrm{T}}d = r$. x is then replaced by $x + d$, and this iterative refinement of the solution is repeated until machine accuracy is obtained.

4 References

Wilkinson J H and Reinsch C (1971) *Handbook for Automatic Computation II, Linear Algebra* Springer–Verlag

5 Parameters

1: A(LDA, $*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array A must be at least max(1, N).

On entry: the upper triangle of the n by n positive definite symmetric matrix A. The elements of the array below the diagonal need not be set.

On exit: the elements of the array below the diagonal are overwritten; the upper triangle of A is unchanged.

2: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F04ASF is called.

Constraint: LDA $\geq$ max(1, N).

3: B(max(1, N)) – REAL (KIND=nag_wp) array *Input*

Note: the dimension of the array B must be at least max(1, N).

On entry: the right-hand side vector b.

4: N – INTEGER *Input*

On entry: n, the order of the matrix A.

Constraint: $\mathrm{N} \geq 0$.

5: C(max(1, N)) – REAL (KIND=nag_wp) array *Output*

On exit: the solution vector x.

6: WK1(max(1, N)) – REAL (KIND=nag_wp) array *Workspace*

7: WK2(max(1, N)) – REAL (KIND=nag_wp) array *Workspace*

8: 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: $\mathrm{IFAIL} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

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

Errors or warnings detected by the routine:

$\mathrm{IFAIL} = 1$

The matrix A is not positive definite, possibly due to rounding errors.

$\mathrm{IFAIL} = 2$

Iterative refinement fails to improve the solution, i.e., the matrix A is too ill-conditioned.

$\mathrm{IFAIL} = 3$

On entry, $\mathrm{N} < 0$,
or $\mathrm{LDA} < \max(1, \mathrm{N})$.

$\mathrm{IFAIL} = -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.8 in the Essential Introduction for further information.

$\mathrm{IFAIL} = -399$

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

See Section 3.7 in the Essential Introduction for further information.

$\mathrm{IFAIL} = -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

The computed solutions should be correct to full machine accuracy. For a detailed error analysis see page 39 of Wilkinson and Reinsch (1971).

8 Parallelism and Performance

F04ASF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F04ASF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The time taken by F04ASF is approximately proportional to n^3.

The routine **must not** be called with the same name for parameters B and C.

10 Example

This example solves the set of linear equations $Ax = b$ where

$$A = \begin{pmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 23 \\ 32 \\ 33 \\ 31 \end{pmatrix}.$$

10.1 Program Text

```
    Program f04asfe

!     F04ASF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: f04asf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, lda, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:), b(:), c(:), wk1(:), wk2(:)
!     .. Executable Statements ..
      Write (nout,*) 'F04ASF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n
      lda = n
      Allocate (a(lda,n),b(n),c(n),wk1(n),wk2(n))
      Read (nin,*)(a(i,1:n),i=1,n), b(1:n)

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call f04asf(a,lda,b,n,c,wk1,wk2,ifail)
```

```
      Write (nout,*) ' Solution'
      Write (nout,99999) c(1:n)

99999 Format (1X,F9.4)
    End Program f04asfe
```

10.2 Program Data

```
F04ASF Example Program Data
  4                         : n
    5    7    6    5
    7   10    8    7
    6    8   10    9
    5    7    9   10
   23   32   33   31    : matrices A and B
```

10.3 Program Results

```
 F04ASF Example Program Results

  Solution
    1.0000
    1.0000
    1.0000
    1.0000
```