NAG Library Routine Document

F05AAF

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

F05AAF applies the Schmidt orthogonalization process to n vectors in m-dimensional space, $n \leq m$.

2 Specification

```
SUBROUTINE F05AAF (A, LDA, M, N1, N2, S, CC, ICOL, IFAIL)

INTEGER            LDA, M, N1, N2, ICOL, IFAIL
REAL (KIND=nag_wp) A(LDA,N2), S(N2), CC
```

3 Description

F05AAF applies the Schmidt orthogonalization process to n linearly independent vectors in m-dimensional space, $n \leq m$. The effect of this process is to replace the original n vectors by n orthonormal vectors which have the property that the rth vector is linearly dependent on the first r of the original vectors, and that the sum of squares of the elements of the rth vector is equal to 1, for $r = 1, 2, \ldots, n$. Inner-products are accumulated using ***additional precision***.

4 References

None.

5 Parameters

1: A(LDA,N2) – REAL (KIND=nag_wp) array *Input/Output*

On entry: columns N1 to N2 contain the vectors to be orthogonalized. The vectors are stored by columns in elements 1 to m.

On exit: these vectors are overwritten by the orthonormal vectors.

2: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F05AAF is called.

Constraint: LDA $\geq$ M.

3: M – INTEGER *Input*

On entry: m, the number of elements in each vector.

4: N1 – INTEGER *Input*

5: N2 – INTEGER *Input*

On entry: the indices of the first and last columns of A to be orthogonalized.

Constraint: N1 $\leq$ N2.

6: S(N2) – REAL (KIND=nag_wp) array *Workspace*

7: CC – REAL (KIND=nag_wp) *Output*

On exit: is used to indicate linear dependence of the original vectors. The nearer CC is to 1.0, the more likely vector ICOL is dependent on vectors N1 to ICOL − 1. See Section 8.

8: ICOL – INTEGER *Output*

On exit: the column number corresponding to CC. See Section 8.

9: 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, N1 > N2.

7 Accuracy

Innerproducts are accumulated using ***additional precision*** arithmetic and full machine accuracy should be obtained except when CC > 0.99999. (See Section 8.)

8 Further Comments

The time taken by F05AAF is approximately proportional to nm^2, where $n = \text{N2} - \text{N1} + 1$.

Parameters CC and ICOL have been included to give some indication of whether or not the vectors are nearly linearly independent, and their values should always be tested on exit from the routine. CC will be in the range $[0.0, 1.0]$ and the closer CC is to 1.0, the more likely the vector ICOL is to be linearly dependent on vectors N1 to ICOL − 1. Theoretically, when the vectors are linearly dependent, CC should be exactly 1.0. In practice, because of rounding errors, it may be difficult to decide whether or not a value of CC close to 1.0 indicates linear dependence. As a general guide a value of CC > 0.99999 usually indicates linear dependence, but examples exist which give CC > 0.99999 for linearly independent vectors. If one of the original vectors is zero or if, possibly due to rounding errors, an exactly zero vector is produced by the Gram–Schmidt process, then CC is set exactly to 1.0 and the vector is not, of course, normalized. If more than one such vector occurs then ICOL references the last of these vectors.

If you are concerned about testing for near linear dependence in a set of vectors you may wish to consider using routine F08KBF (DGESVD).

9 Example

This example orthonormalizes columns 2, 3 and 4 of the matrix:

$$\begin{pmatrix} 1 & -2 & 3 & 1 \\ -2 & 1 & -2 & -1 \\ 3 & -2 & 1 & 5 \\ 4 & 1 & 5 & 3 \end{pmatrix}.$$

9.1 Program Text

```
    Program f05aafe

!     F05AAF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: f05aaf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter              :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)              :: cc
      Integer                         :: i, icol, ifail, lda, m, n1, n2
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:), s(:)
!     .. Executable Statements ..
      Write (nout,*) 'F05AAF Example Program Results'

!     Skip heading in data file
      Read (nin,*)

      Read (nin,*) m, n1, n2
      lda = m
      Allocate (a(lda,n2),s(n2))

      Read (nin,*)(a(i,1:n2),i=1,m)

      ifail = 0
      Call f05aaf(a,lda,m,n1,n2,s,cc,icol,ifail)

      Write (nout,*)
      Write (nout,99999) 'N1 = ', n1, '  N2 = ', n2
      Write (nout,*)
      Write (nout,99998) 'CC = ', cc, ' ICOL = ', icol
      Write (nout,*)
      Write (nout,*) 'Final vectors'
      Write (nout,99997)(a(i,n1:n2),i=1,m)

99999 Format (1X,A,I2,A,I2)
99998 Format (1X,A,F7.4,A,I2)
99997 Format (1X,3F9.4)
    End Program f05aafe
```

9.2 Program Data

```
F05AAF Example Program Data
 4   2    4
 1  -2    3    1
-2   1   -2   -1
 3  -2    1    5
 4   1    5    3
```

9.3 Program Results

```
 F05AAF Example Program Results

 N1 =  2  N2 =  4

 CC =  0.5822 ICOL =  4

 Final vectors
   -0.6325   0.3310  -0.5404
    0.3162  -0.2483   0.2119
   -0.6325  -0.0000   0.7735
    0.3162   0.9104   0.2543
```