NAG Library Routine Document

F01GBF

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

F01GBF computes the action of the matrix exponential e^{tA}, on the matrix B, where A is a real n by n matrix, B is a real n by m matrix and t is a real scalar. It uses reverse communication for evaluating matrix products, so that the matrix A is not accessed explicitly.

2 Specification

```
SUBROUTINE F01GBF (IREVCM, N, M, B, LDB, T, TR, B2, LDB2, X, LDX, Y,        &
                   LDY, P, R, Z, COMM, ICOMM, IFAIL)
INTEGER            IREVCM, N, M, LDB, LDB2, LDX, LDY, ICOMM(2*N+40),        &
                   IFAIL
REAL (KIND=nag_wp) B(LDB,*), T, TR, B2(LDB2,*), X(LDX,*), Y(LDY,*),         &
                   P(N), R(N), Z(N), COMM(N*M+3*N+12)
```

3 Description

$e^{tA}B$ is computed using the algorithm described in Al–Mohy and Higham (2011) which uses a truncated Taylor series to compute the $e^{tA}B$ without explicitly forming e^{tA}.

The algorithm does not explicity need to access the elements of A; it only requires the result of matrix multiplications of the form AX or $A^{\mathrm{T}}Y$. A reverse communication interface is used, in which control is returned to the calling program whenever a matrix product is required.

4 References

Al–Mohy A H and Higham N J (2011) Computing the action of the matrix exponential, with an application to exponential integrators *SIAM J. Sci. Statist. Comput.* **33(2)** 488-511

Higham N J (2008) *Functions of Matrices: Theory and Computation* SIAM, Philadelphia, PA, USA

5 Parameters

Note: this routine uses **reverse communication.** Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the parameter **IREVCM**. Between intermediate exits and re-entries, **all parameters other than B2, X, Y, P and R must remain unchanged**.

1: IREVCM – INTEGER *Input/Output*

On initial entry: must be set to 0.

On intermediate exit: IREVCM = 1, 2, 3, 4 or 5. The calling program must:

(a) if IREVCM = 1: evaluate $B_2 = AB$, where B_2 is an n by m matrix, and store the result in B2;
if IREVCM = 2: evaluate $Y = AX$, where X and Y are n by 2 matrices, and store the result in Y;
if IREVCM = 3: evaluate $X = A^{\mathrm{T}}Y$ and store the result in X;
if IREVCM = 4: evaluate $p = Az$ and store the result in P;
if IREVCM = 5: evaluate $r = A^{\mathrm{T}}z$ and store the result in R.

(b) call F01GBF again with all other parameters unchanged.

On final exit: IREVCM $= 0$.

2: N – INTEGER *Input*

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

Constraint: $N \geq 0$.

3: M – INTEGER *Input*

On entry: the number of columns of the matrix B.

Constraint: $M \geq 0$.

4: $B(LDB, *)$ – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array B must be at least M.

On initial entry: the n by m matrix B.

On intermediate exit: if IREVCM $= 1$, contains the n by m matrix B.

On intermediate re-entry: must not be changed.

On final exit: the n by m matrix $e^{tA}B$.

5: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F01GBF is called.

Constraint: $LDB \geq N$.

6: T – REAL (KIND=nag_wp) *Input*

On entry: the scalar t.

7: TR – REAL (KIND=nag_wp) *Input*

On entry: the trace of A. If this is not available then any number can be supplied (0 is a reasonable default); however, in the trivial case, $n = 1$, the result $e^{\mathrm{TR}t}B$ is immediately returned in the first row of B. See Section 9.

8: $B2(LDB2, *)$ – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array B2 must be at least M.

On initial entry: need not be set.

On intermediate re-entry: if IREVCM $= 1$, must contain AB.

On final exit: the array is undefined.

9: LDB2 – INTEGER *Input*

On initial entry: the first dimension of the array B2 as declared in the (sub)program from which F01GBF is called.

Constraint: $LDB2 \geq N$.

10: $X(LDX, *)$ – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array X must be at least 2.

On initial entry: need not be set.

On intermediate exit: if IREVCM $= 2$, contains the current n by 2 matrix X.

On intermediate re-entry: if IREVCM $= 3$, must contain $A^{\mathrm{T}}Y$.

On final exit: the array is undefined.

11: LDX – INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F01GBF is called.

Constraint: LDX $\geq$ N.

12: Y(LDY, $*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array Y must be at least 2.

On initial entry: need not be set.

On intermediate exit: if IREVCM $=$ 3, contains the current n by 2 matrix Y.

On intermediate re-entry: if IREVCM $=$ 2, must contain AX.

On final exit: the array is undefined.

13: LDY – INTEGER *Input*

On entry: the first dimension of the array Y as declared in the (sub)program from which F01GBF is called.

Constraint: LDY $\geq$ N.

14: P(N) – REAL (KIND=nag_wp) array *Input/Output*

On initial entry: need not be set.

On intermediate re-entry: if IREVCM $=$ 4, must contain Az.

On final exit: the array is undefined.

15: R(N) – REAL (KIND=nag_wp) array *Input/Output*

On initial entry: need not be set.

On intermediate re-entry: if IREVCM $=$ 5, must contain $A^{\mathrm{T}}z$.

On final exit: the array is undefined.

16: Z(N) – REAL (KIND=nag_wp) array *Input/Output*

On initial entry: need not be set.

On intermediate exit: if IREVCM $=$ 4 or 5, contains the vector z.

On intermediate re-entry: must not be changed.

On final exit: the array is undefined.

17: COMM(N $\times$ M $+$ 3 $\times$ N $+$ 12) – REAL (KIND=nag_wp) array *Communication Array*

18: ICOMM(2 $\times$ N $+$ 40) – INTEGER array *Communication Array*

19: IFAIL – INTEGER *Input/Output*

On initial 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, because for this routine the values of the output parameters may be useful even if IFAIL $\neq$ 0 on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On final 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 $= 2$

$e^{tA}B$ has been computed using an IEEE double precision Taylor series, although the arithmetic precision is higher than IEEE double precision.

IFAIL $= -1$

On initial entry, IREVCM $= \langle value \rangle$.
Constraint: IREVCM $= 0$.

On intermediate re-entry, IREVCM $= \langle value \rangle$.
Constraint: IREVCM $= 1$, 2, 3, 4 or 5.

IFAIL $= -2$

On initial entry, N $= \langle value \rangle$.
Constraint: N ≥ 0.

IFAIL $= -3$

On initial entry, M $= \langle value \rangle$.
Constraint: M ≥ 0.

IFAIL $= -5$

On initial entry, LDB $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: LDB $\geq$ N.

IFAIL $= -9$

On initial entry, LDB2 $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: LDB2 $\geq$ N.

IFAIL $= -11$

On initial entry, LDX $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: LDX $\geq$ N.

IFAIL $= -13$

On initial entry, LDY $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: LDY $\geq$ N.

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.

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.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

For a symmetric matrix A (for which $A^{\mathrm{T}} = A$) the computed matrix $e^{tA}B$ is guaranteed to be close to the exact matrix, that is, the method is forward stable. No such guarantee can be given for non-symmetric matrices. See Section 4 of Al–Mohy and Higham (2011) for details and further discussion.

8 Parallelism and Performance

F01GBF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

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

9.1 Use of $Tr(A)$

The elements of A are not explicitly required by F01GBF. However, the trace of A is used in the preprocessing phase of the algorithm. If $Tr(A)$ is not available to the calling subroutine then any number can be supplied (0 is recommended). This will not affect the stability of the algorithm, but it may reduce its efficiency.

9.2 When to use F01GBF

F01GBF is designed to be used when A is large and sparse. Whenever a matrix multiplication is required, the routine will return control to the calling program so that the multiplication can be done in the most efficient way possible. Note that $e^{tA}B$ will not, in general, be sparse even if A is sparse.

If A is small and dense then F01GAF can be used to compute $e^{tA}B$ without the use of a reverse communication interface.

The complex analog of F01GBF is F01HBF.

9.3 Use in Conjunction with NAG Library Routines

To compute $e^{tA}B$, the following skeleton code can normally be used:

```
revcm: Do
  Call F01GBF(IREVCM,N,M,B,LDB,T,TR,B2,LDB2,X,LDX,Y,LDY,P,R,Z, &
              COMM,ICOMM,IFAIL)
  If (IREVCM == 0) Then
      Exit revcm
  Else If (IREVCM == 1) Then
      .. Code to compute B2=AB ..
  Else If (IREVCM == 2) Then
      .. Code to compute Y=AX ..
  Else If (IREVCM == 3) Then
      .. Code to compute X=A^T Y ..
  Else If (IREVCM == 4) Then
      .. Code to compute P=AZ ..
  Else If (IREVCM == 5) Then
      .. Code to compute R=A^T Z ..
  End If
End Do revcm
```

The code used to compute the matrix products will vary depending on the way A is stored. If all the elements of A are stored explicitly, then F06YAF (DGEMM)) can be used. If A is triangular then F06YFF (DTRMM) should be used. If A is symmetric, then F06YCF (DSYMM) should be used. For sparse A stored in coordinate storage format F11XAF and F11XEF can be used. Alternatively if A is stored in compressed column format F11MKF can be used.

10 Example

This example computes $e^{tA}B$, where

$$A = \begin{pmatrix} 0.4 & -0.2 & 1.3 & 0.6 \\ 0.3 & 0.8 & 1.0 & 1.0 \\ 3.0 & 4.8 & 0.2 & 0.7 \\ 0.5 & 0.0 & -5.0 & 0.7 \end{pmatrix},$$

$$B = \begin{pmatrix} 0.1 & 1.1 \\ 1.7 & -0.2 \\ 0.5 & 1.0 \\ 0.4 & -0.2 \end{pmatrix},$$

and

$$t = -0.2.$$

10.1 Program Text

```
    Program f01gbfe

!     F01GBF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: dgemm, f01gbf, nag_wp, x04caf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)                :: t, tr
      Integer                           :: i, ifail, irevcm, lda, ldb, ldb2,    &
                                           ldx, ldy, m, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable   :: a(:,:), b(:,:), b2(:,:), comm(:),    &
                                           p(:), r(:), x(:,:), y(:,:), z(:)
      Integer, Allocatable              :: icomm(:)
!     .. Executable Statements ..
      Write (nout,*) 'F01GBF Example Program Results'
      Write (nout,*)
      Flush (nout)

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

      lda = n
      ldb = n
      ldb2 = n
      ldx = n
      ldy = n

!     Allocate required memory
      Allocate (a(lda,n))
      Allocate (b(ldb,m))
      Allocate (b2(ldb2,m))
```

```
      Allocate (comm(n*m+3*n+12))
      Allocate (x(ldx,2))
      Allocate (y(ldy,2))
      Allocate (icomm(2*n+40))
      Allocate (p(n))
      Allocate (r(n))
      Allocate (z(n))

!     Read A from data file
      Read (nin,*)(a(i,1:n),i=1,n)

!     Read B from data file
      Read (nin,*)(b(i,1:m),i=1,n)

!     Compute the trace of A
      tr = 0.0_nag_wp
      Do i = 1, n
        tr = tr + a(i,i)
      End Do

!     Find exp(tA)B
      irevcm = 0
      ifail = 0

!      Initial call to f01gbf reverse communication interface
revcm: Do
        Call f01gbf(irevcm,n,m,b,ldb,t,tr,b2,ldb2,x,ldx,y,ldy,p,r,z,comm, &
          icomm,ifail)
        If (irevcm==0) Then
          Exit revcm

        Else If (irevcm==1) Then
!         Compute AB and store in B2
          Call dgemm('N','N',n,m,n,1.0_nag_wp,a,lda,b,ldb,0.0_nag_wp,b2,ldb2)

        Else If (irevcm==2) Then
!         Compute AX and store in Y
          Call dgemm('N','N',n,2,n,1.0_nag_wp,a,lda,x,ldx,0.0_nag_wp,y,ldy)

        Else If (irevcm==3) Then
!         Compute A^T Y and store in X
          Call dgemm('T','N',n,2,n,1.0_nag_wp,a,lda,y,ldy,0.0_nag_wp,x,ldx)

        Else If (irevcm==4) Then
!         Compute AZ and store in P
          Call dgemm('N','N',n,1,n,1.0_nag_wp,a,lda,z,n,0.0_nag_wp,p,n)

        Else
!         Compute A^T Z and store in R
          Call dgemm('T','N',n,1,n,1.0_nag_wp,a,lda,z,n,0.0_nag_wp,r,n)
        End If

      End Do revcm

      If (ifail==0) Then
!       Print solution
        ifail = 0
        Call x04caf('G','N',n,m,b,ldb,'exp(tA)B',ifail)
      End If

    End Program f01gbfe
```

10.2 Program Data

```
F01GBF Example Program Data

4   2  -0.2                :Values of N, M, and T

0.4   -0.2    1.3    0.6
0.3    0.8    1.0    1.0
```

```
3.0    4.8    0.2    0.7
0.5    0.0   -5.0    0.7   :End of matrix A

0.1    1.1
1.7   -0.2
0.5    1.0
0.4   -0.2                 :End of matrix B
```

10.3 Program Results

```
 F01GBF Example Program Results

 exp(tA)B
             1          2
 1      0.1933     0.7812
 2      1.4423    -0.4055
 3     -1.0756     0.6686
 4      0.0276     0.4900
```