NAG Library Routine Document

G05PXF

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

G05PXF generates a random orthogonal matrix.

2 Specification

```
SUBROUTINE G05PXF (SIDE, INIT, M, N, STATE, A, LDA, IFAIL)
INTEGER            M, N, STATE(*), LDA, IFAIL
REAL (KIND=nag_wp) A(LDA,N)
CHARACTER(1)       SIDE, INIT
```

3 Description

G05PXF pre- or post-multiplies an m by n matrix A by a random orthogonal matrix U, overwriting A. The matrix A may optionally be initialized to the identity matrix before multiplying by U, hence U is returned. U is generated using the method of Stewart (1980). The algorithm can be summarised as follows.

Let $x_1, x_2, \ldots, x_{n-1}$ follow independent multinormal distributions with zero mean and variance $I\sigma^2$ and dimensions $n, n-1, \ldots, 2$; let $H_j = \text{diag}\left(I_{j-1}, H_j^*\right)$, where I_{j-1} is the identity matrix and H_j^* is the Householder transformation that reduces x_j to $r_{jj}e_1$, e_1 being the vector with first element one and the remaining elements zero and r_{jj} being a scalar, and let $D = \text{diag}(\text{sign}(r_{11}), \text{sign}(r_{22}), \ldots, \text{sign}(r_{nn}))$. Then the product $U = DH_1H_2 \ldots H_{n-1}$ is a random orthogonal matrix distributed according to the Haar measure over the set of orthogonal matrices of n. See Theorem 3.3 in Stewart (1980).

One of the initialization routines G05KFF (for a repeatable sequence if computed sequentially) or G05KGF (for a non-repeatable sequence) must be called prior to the first call to G05PXF.

4 References

Stewart G W (1980) The efficient generation of random orthogonal matrices with an application to condition estimates *SIAM J. Numer. Anal.* **17** 403–409

5 Arguments

1: SIDE – CHARACTER(1) *Input*

On entry: indicates whether the matrix A is multiplied on the left or right by the random orthogonal matrix U.

SIDE = 'L'
The matrix A is multiplied on the left, i.e., premultiplied.

SIDE = 'R'
The matrix A is multiplied on the right, i.e., post-multiplied.

Constraint: SIDE = 'L' or 'R'.

2: INIT – CHARACTER(1) *Input*

On entry: indicates whether or not A should be initialized to the identity matrix.

INIT = 'I'
A is initialized to the identity matrix.

INIT = 'N'
A is not initialized and the matrix A must be supplied in A.

Constraint: INIT = 'I' or 'N'.

3: M – INTEGER *Input*

On entry: m, the number of rows of the matrix A.

Constraints:

if SIDE = 'L', M $>$ 1;
otherwise M $\geq$ 1.

4: N – INTEGER *Input*

On entry: n, the number of columns of the matrix A.

Constraints:

if SIDE = 'R', N $>$ 1;
otherwise N $\geq$ 1.

5: STATE($*$) – INTEGER array *Communication Array*

Note: the actual argument supplied **must** be the array STATE supplied to the initialization routines G05KFF or G05KGF.

On entry: contains information on the selected base generator and its current state.

On exit: contains updated information on the state of the generator.

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

On entry: if INIT = 'N', A must contain the matrix A.

On exit: the matrix UA when SIDE = 'L' or the matrix AU when SIDE = 'R'.

7: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which G05PXF is called.

Constraint: LDA $\geq$ M.

8: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation 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 argument, 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, SIDE is not valid: SIDE $= \langle value \rangle$.

IFAIL $= 2$

On entry, INIT is not valid: INIT $= \langle value \rangle$.

IFAIL $= 3$

On entry, SIDE $= \langle value \rangle$, M $= \langle value \rangle$.
Constraint: if SIDE $=$ 'L', M > 1; otherwise M ≥ 1.

IFAIL $= 4$

On entry, SIDE $= \langle value \rangle$, N $= \langle value \rangle$.
Constraint: if SIDE $=$ 'R', N > 1; otherwise N ≥ 1.

IFAIL $= 5$

On entry, STATE vector has been corrupted or not initialized.

IFAIL $= 7$

On entry, LDA $= \langle value \rangle$ and M $= \langle value \rangle$.
Constraint: LDA $\geq$ M.

IFAIL $= 8$

On entry, SIDE $= \langle value \rangle$, M $= \langle value \rangle$.
Constraint: if SIDE $=$ 'L', M > 1; otherwise M ≥ 1.

On entry, SIDE $= \langle value \rangle$, N $= \langle value \rangle$.
Constraint: if SIDE $=$ 'R', N > 1; otherwise N ≥ 1.

IFAIL $= -99$

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

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

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

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The maximum error in $U^{\mathrm{T}}U$ should be a modest multiple of ***machine precision*** (see Chapter X02).

8 Parallelism and Performance

G05PXF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

G05PXF 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

None.

10 Example

Following initialization of the pseudorandom number generator by a call to G05KFF, a 4 by 4 orthogonal matrix is generated using the INIT = 'I' option and the result printed.

10.1 Program Text

```
    Program g05pxfe

!     G05PXF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g05kff, g05pxf, nag_wp, x04caf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: lseed = 1, nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: genid, i, ifail, lda, lstate, m, n,  &
                                          subid
      Character (1)                    :: init, side
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:)
      Integer                          :: seed(lseed)
      Integer, Allocatable             :: state(:)
!     .. Executable Statements ..
      Write (nout,*) 'G05PXF Example Program Results'
      Write (nout,*)
      Flush (nout)

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

!     Read in the base generator information and seed
      Read (nin,*) genid, subid, seed(1)

!     Initial call to initializer to get size of STATE array
      lstate = 0
      Allocate (state(lstate))
      ifail = 0
      Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

!     Reallocate STATE
      Deallocate (state)
      Allocate (state(lstate))

!     Initialize the generator to a repeatable sequence
      ifail = 0
```

```
      Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

!     Read in the problem size
      Read (nin,*) n, m

!     Read in control parameters
      Read (nin,*) side, init

      lda = m
      Allocate (a(lda,n))

!     Read in A if required
      If (init=='N' .Or. init=='n') Then
        Read (nin,*)(a(i,1:n),i=1,m)
      End If

!     Generate the random orthogonal matrix
      ifail = 0
      Call g05pxf(side,init,m,n,state,a,lda,ifail)

!     Display the results
      ifail = 0
      Call x04caf('General',' ',m,n,a,lda,'Random Matrix',ifail)

    End Program g05pxfe
```

10.2 Program Data

```
G05PXF Example Program Data
1  1  1762543      :: GENID,SUBID,SEED(1)
4  4               :: N,M
'R'  'I'           :: SIDE,INIT
```

10.3 Program Results

```
 G05PXF Example Program Results

 Random Matrix
          1       2       3       4
 1   0.1756  0.7401 -0.3067 -0.5722
 2   0.6593 -0.5781 -0.2191 -0.4279
 3   0.6680  0.3172  0.6077  0.2895
 4  -0.2971 -0.1323  0.6990 -0.6369
```