# NAG Library Routine Document

# F12AGF

**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.

**Note**: *this routine uses* **optional parameters** *to define choices in the problem specification. If you wish to use* default *settings for all of the optional parameters, then the option setting routine F12ADF need not be called. If, however, you wish to reset some or all of the settings please refer to Section 11 in F12ADF for a detailed description of the specification of the optional parameters.*

## 1 Purpose

F12AGF is the main solver routine in a suite of routines consisting of F12ADF, F12AFF and F12AGF. It must be called following an initial call to F12AFF and following any calls to F12ADF.

F12AGF returns approximations to selected eigenvalues, and (optionally) the corresponding eigenvectors, of a standard or generalized eigenvalue problem defined by real banded nonsymmetric matrices. The banded matrix must be stored using the LAPACK storage format for real banded nonsymmetric matrices.

## 2 Specification

```
SUBROUTINE F12AGF (KL, KU, AB, LDAB, MB, LDMB, SIGMAR, SIGMAI, NCONV,      &
                   DR, DI, Z, LDZ, RESID, V, LDV, COMM, ICOMM, IFAIL)

INTEGER            KL, KU, LDAB, LDMB, NCONV, LDZ, LDV, ICOMM(*), IFAIL
REAL (KIND=nag_wp) AB(LDAB,*), MB(LDMB,*), SIGMAR, SIGMAI, DR(*),          &
                   DI(*), Z(LDZ,*), RESID(*), V(LDV,*), COMM(*)
```

## 3 Description

The suite of routines is designed to calculate some of the eigenvalues, $\lambda$, (and optionally the corresponding eigenvectors, $x$) of a standard eigenvalue problem $Ax = \lambda x$, or of a generalized eigenvalue problem $Ax = \lambda Bx$ of order $n$, where $n$ is large and the coefficient matrices $A$ and $B$ are banded, real and nonsymmetric.

Following a call to the initialization routine F12AFF, F12AGF returns the converged approximations to eigenvalues and (optionally) the corresponding approximate eigenvectors and/or an orthonormal basis for the associated approximate invariant subspace. The eigenvalues (and eigenvectors) are selected from those of a standard or generalized eigenvalue problem defined by real banded nonsymmetric matrices. There is negligible additional computational cost to obtain eigenvectors; an orthonormal basis is always computed, but there is an additional storage cost if both are requested.

The banded matrices $A$ and $B$ must be stored using the LAPACK column ordered storage format for banded nonsymmetric matrices; please refer to Section 3.3.2 in the F07 Chapter Introduction for details on this storage format.

F12AGF is based on the banded driver routines **dnbdr1** to **dnbdr6** from the ARPACK package, which uses the Implicitly Restarted Arnoldi iteration method. The method is described in Lehoucq and Sorensen (1996) and Lehoucq (2001) while its use within the ARPACK software is described in great detail in Lehoucq *et al.* (1998). An evaluation of software for computing eigenvalues of sparse nonsymmetric matrices is provided in Lehoucq and Scott (1996). This suite of routines offers the same functionality as the ARPACK banded driver software for real nonsymmetric problems, but the interface design is quite different in order to make the option setting clearer and to combine the different drivers into a general purpose routine.

F12AGF, is a general purpose routine that must be called following initialization by F12AFF. F12AGF uses options, set either by default or explicitly by calling F12ADF, to return the converged approximations to selected eigenvalues and (optionally):

– the corresponding approximate eigenvectors;

– an orthonormal basis for the associated approximate invariant subspace;

– both.

Please note that for **Generalized** problems, the **Shifted Inverse Imaginary** and **Shifted Inverse Real** inverse modes are only appropriate if either $A$ or $B$ is symmetric semidefinite. Otherwise, if $A$ or $B$ is non-singular, the **Standard** problem can be solved using the matrix $B^{-1}A$ (say).

## 4 References

Lehoucq R B (2001) Implicitly restarted Arnoldi methods and subspace iteration *SIAM Journal on Matrix Analysis and Applications* **23** 551–562

Lehoucq R B and Scott J A (1996) An evaluation of software for computing eigenvalues of sparse nonsymmetric matrices *Preprint MCS-P547-1195* Argonne National Laboratory

Lehoucq R B and Sorensen D C (1996) Deflation techniques for an implicitly restarted Arnoldi iteration *SIAM Journal on Matrix Analysis and Applications* **17** 789–821

Lehoucq R B, Sorensen D C and Yang C (1998) *ARPACK Users' Guide: Solution of Large-scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods* SIAM, Philidelphia

## 5 Arguments

1: KL – INTEGER *Input*

*On entry*: the number of subdiagonals of the matrices $A$ and $B$.

*Constraint*: KL $\geq 0$.

2: KU – INTEGER *Input*

*On entry*: the number of superdiagonals of the matrices $A$ and $B$.

*Constraint*: KU $\geq 0$.

3: AB(LDAB, $*$) – REAL (KIND=nag_wp) array *Input*

**Note**: the second dimension of the array AB must be at least $\max(1, \mathrm{N})$ (see F12AFF).

*On entry*: must contain the matrix $A$ in LAPACK banded storage format for nonsymmetric matrices (see Section 3.3.4 in the F07 Chapter Introduction).

4: LDAB – INTEGER *Input*

*On entry*: the first dimension of the array AB as declared in the (sub)program from which F12AGF is called.

*Constraint*: LDAB $\geq 2 \times \mathrm{KL} + \mathrm{KU} + 1$.

5: MB(LDMB, $*$) – REAL (KIND=nag_wp) array *Input*

**Note**: the second dimension of the array MB must be at least $\max(1, \mathrm{N})$ (see F12AFF).

*On entry*: must contain the matrix $B$ in LAPACK banded storage format for nonsymmetric matrices (see Section 3.3.4 in the F07 Chapter Introduction).

6: LDMB – INTEGER *Input*

*On entry*: the first dimension of the array MB as declared in the (sub)program from which F12AGF is called.

*Constraint*: LDMB $\geq 2 \times \mathrm{KL} + \mathrm{KU} + 1$.

7: SIGMAR – REAL (KIND=nag_wp) *Input*

*On entry*: if one of the **Shifted Inverse Real** modes (see F12ADF) have been selected then SIGMAR must contain the real part of the shift used; otherwise SIGMAR is not referenced. Section 4.3.4 in the F12 Chapter Introduction describes the use of shift and inverse transformations.

8: SIGMAI – REAL (KIND=nag_wp) *Input*

*On entry*: if one of the **Shifted Inverse Real** modes (see F12ADF) have been selected then SIGMAI must contain the imaginary part of the shift used; otherwise SIGMAI is not referenced. Section 4.3.4 in the F12 Chapter Introduction describes the use of shift and inverse transformations.

9: NCONV – INTEGER *Output*

*On exit*: the number of converged eigenvalues.

10: DR$(*)$ – REAL (KIND=nag_wp) array *Output*

**Note**: the dimension of the array DR must be at least NEV $+ 1$ (see F12AFF).

*On exit*: the first NCONV locations of the array DR contain the real parts of the converged approximate eigenvalues. The number of eigenvalues returned may be one more than the number requested by NEV since complex values occur as conjugate pairs and the second in the pair can be returned in position NEV $+ 1$ of the array.

11: DI$(*)$ – REAL (KIND=nag_wp) array *Output*

**Note**: the dimension of the array DI must be at least NEV $+ 1$ (see F12AFF).

*On exit*: the first NCONV locations of the array DI contain the imaginary parts of the converged approximate eigenvalues. The number of eigenvalues returned may be one more than the number requested by NEV since complex values occur as conjugate pairs and the second in the pair can be returned in position NEV $+ 1$ of the array.

12: Z(LDZ, $*$) – REAL (KIND=nag_wp) array *Output*

**Note**: the second dimension of the array Z must be at least NEV $+ 1$ if the default option **Vectors** = Ritz has been selected and at least 1 if the option **Vectors** = None or Schur has been selected (see F12AAF).

*On exit*: if the default option **Vectors** = Ritz has been selected then Z contains the final set of eigenvectors corresponding to the eigenvalues held in DR and DI. The complex eigenvector associated with the eigenvalue with positive imaginary part is stored in two consecutive columns. The first column holds the real part of the eigenvector and the second column holds the imaginary part. The eigenvector associated with the eigenvalue with negative imaginary part is simply the complex conjugate of the eigenvector associated with the positive imaginary part.

13: LDZ – INTEGER *Input*

*On entry*: the first dimension of the array Z as declared in the (sub)program from which F12AGF is called.

*Constraints*:

if the default option **Vectors** = Ritz has been selected, LDZ $\geq$ N;
if the option **Vectors** = None or Schur has been selected, LDZ $\geq 1$.

14: RESID$(*)$ – REAL (KIND=nag_wp) array *Input/Output*

**Note**: the dimension of the array RESID must be at least N (see F12AFF).

*On entry*: need not be set unless the option **Initial Residual** has been set in a prior call to F12ADF in which case RESID must contain an initial residual vector.

*On exit*: contains the final residual vector.

15: V(LDV, $*$) – REAL (KIND=nag_wp) array *Output*

**Note**: the second dimension of the array V must be at least max(1, NCV) (see F12AFF).

*On exit*: if the option **Vectors** (see F12ADF) has been set to Schur or Ritz then the first NCONV $\times n$ elements of V will contain approximate Schur vectors that span the desired invariant subspace.

The $i$th Schur vector is stored in the $i$th column of V.

16: LDV – INTEGER *Input*

*On entry*: the first dimension of the array V as declared in the (sub)program from which F12AGF is called.

*Constraint*: LDV $\geq$ N.

17: COMM($*$) – REAL (KIND=nag_wp) array *Communication Array*

*On entry*: must remain unchanged from the prior call to F12ADF and F12AFF.

*On exit*: contains no useful information.

18: ICOMM($*$) – INTEGER array *Communication Array*

*On entry*: must remain unchanged from the prior call to F12ADF and F12AFF.

*On exit*: contains no useful information.

19: 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, KL $< 0$.

IFAIL $= 2$

On entry, KU $< 0$.

IFAIL $= 3$

On entry, LDAB $< 2 \times$ KL $+$ KU $+ 1$.

IFAIL $= 4$

On entry, the option **Shifted Inverse Imaginary** was selected, and SIGMAI $=$ zero, but SIGMAI must be nonzero for this computational mode.

IFAIL $= 5$

**Iteration Limit** $< 0$.

IFAIL $= 6$

The options **Generalized** and **Regular** are incompatible.

IFAIL $= 7$

The **Initial Residual** was selected but the starting vector held in RESID is zero.

IFAIL $= 8$

Either the initialization routine F12AFF has not been called prior to the first call of this routine or a communication array has become corrupted.

IFAIL $= 9$

On entry, LDZ $<$ N or LDZ $< 1$ when no vectors are required.

IFAIL $= 10$

On entry, the option **Vectors** $=$ Select was selected, but this is not yet implemented.

IFAIL $= 11$

The number of eigenvalues found to sufficient accuracy is zero.

IFAIL $= 12$

Could not build an Arnoldi factorization. Consider changing NCV or NEV in the initialization routine (see Section 5 in F12AFF for details of these arguments).

IFAIL $= 13$

Unexpected error in internal call to compute eigenvalues and corresponding error bounds of the current upper Hessenberg matrix. Please contact NAG.

IFAIL $= 14$

Unexpected error during calculation of a real Schur form: there was a failure to compute all the converged eigenvalues. Please contact NAG.

IFAIL $= 15$

Unexpected error: the computed Schur form could not be reordered by an internal call. Please contact NAG.

IFAIL $= 16$

Unexpected error in internal call while calculating eigenvectors. Please contact NAG.

IFAIL $= 17$

Failure during internal factorization of real banded matrix. Please contact NAG.

IFAIL $= 18$

Failure during internal solution of real banded system. Please contact NAG.

IFAIL $= 19$

Failure during internal factorization of complex banded matrix. Please contact NAG.

IFAIL $= 20$

Failure during internal solution of complex banded system. Please contact NAG.

IFAIL $= 21$

The maximum number of iterations has been reached. Some Ritz values may have converged; NCONV returns the number of converged values.

IFAIL $= 22$

No shifts could be applied during a cycle of the implicitly restarted Arnoldi iteration. One possibility is to increase the size of NCV relative to NEV (see Section 5 in F12AFF for details of these arguments).

IFAIL $= 23$

Overflow occurred during transformation of Ritz values to those of the original problem.

IFAIL $= 24$

The routine was unable to dynamically allocate sufficient internal workspace. Please contact NAG.

IFAIL $= 25$

An unexpected error has occurred. Please contact NAG.

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 relative accuracy of a Ritz value, $\lambda$, is considered acceptable if its Ritz estimate $\leq$ **Tolerance** $\times |\lambda|$. The default **Tolerance** used is the ***machine precision*** given by X02AJF.

## 8 Parallelism and Performance

F12AGF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F12AGF 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

This example constructs the matrices $A$ and $B$ using LAPACK band storage format and solves $Ax = \lambda Bx$ in shifted imaginary mode using the complex shift $\sigma$.

### 10.1 Program Text

```
    Program f12agfe

!     F12AGF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: dgbmv, dnrm2, f06bnf, f12adf, f12aff, f12agf,     &
                             nag_wp, x04abf, x04caf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter    :: zero = 0.0_nag_wp
      Integer, Parameter               :: iset = 1, nin = 5, nout = 6
      Logical, Parameter               :: printr = .False.
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: h, rho, sigmai, sigmar
      Integer                          :: i, idiag, ifail, isub, isup, j, kl,  &
                                          ku, lcomm, ldab, ldmb, ldv, licomm,  &
                                          lo, n, ncol, nconv, ncv, nev, nx,    &
                                          outchn
      Logical                          :: first
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: ab(:,:), ax(:), comm(:), di(:),      &
                                          dr(:), d_print(:,:), mb(:,:), mx(:), &
                                          resid(:), v(:,:)
      Integer, Allocatable             :: icomm(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: abs, int, max, real
!     .. Executable Statements ..
      Write (nout,*) 'F12AGF Example Program Results'
      Write (nout,*)
      Flush (nout)

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

      Read (nin,*) nx, nev, ncv, sigmar, sigmai
      n = nx*nx

!     Initialize communication arrays.
!     Query the required sizes of the communication arrays.

      licomm = -1
      lcomm = -1
      Allocate (icomm(max(1,licomm)),comm(max(1,lcomm)))

      ifail = 0
      Call f12aff(n,nev,ncv,icomm,licomm,comm,lcomm,ifail)

      licomm = icomm(1)
```

```
      lcomm = int(comm(1))
      Deallocate (icomm,comm)
      Allocate (icomm(max(1,licomm)),comm(max(1,lcomm)))

      ifail = 0
      Call f12aff(n,nev,ncv,icomm,licomm,comm,lcomm,ifail)

!     Set the mode.

      ifail = 0
      Call f12adf('SHIFTED IMAGINARY',icomm,comm,ifail)

!     Set problem type

      ifail = 0
      Call f12adf('GENERALIZED',icomm,comm,ifail)

!     Construct the matrix A in banded form and store in AB.
!     KU, KL are number of superdiagonals and subdiagonals within
!     the band of matrices A and M.

      kl = nx
      ku = nx
      ldab = 2*kl + ku + 1
      ldmb = 2*kl + ku + 1
      Allocate (ab(ldab,n),mb(ldmb,n))

!     Zero out AB and MB.

      ab(1:ldab,1:n) = 0.0_nag_wp
      mb(1:ldmb,1:n) = 0.0_nag_wp

!     Main diagonal of A.

      idiag = kl + ku + 1
      ab(idiag,1:n) = 4.0_nag_wp
      mb(idiag,1:n) = 4.0_nag_wp

!     First subdiagonal and superdiagonal of A.

      isup = kl + ku
      isub = kl + ku + 2
      rho = 100.0_nag_wp
      h = one/real(nx+1,kind=nag_wp)

      Do i = 1, nx
        lo = (i-1)*nx

        Do j = lo + 1, lo + nx - 1
          ab(isub,j+1) = -one + 0.5_nag_wp*h*rho
          ab(isup,j) = -one - 0.5_nag_wp*h*rho
        End Do

      End Do

      mb(isub,2:n) = one
      mb(isup,1:n-1) = one

!     KL-th subdiagonal and KU-th superdiagonal.

      isup = kl + 1
      isub = 2*kl + ku + 1

      Do i = 1, nx - 1
        lo = (i-1)*nx

        Do j = lo + 1, lo + nx
          ab(isup,nx+j) = -one
          ab(isub,j) = -one
        End Do
```

```
      End Do

!     Find eigenvalues closest in value to SIGMA and corresponding
!     eigenvectors.

      ldv = n
      Allocate (dr(nev+1),di(nev+1),v(ldv,ncv),resid(n))

      ifail = -1
      Call f12agf(kl,ku,ab,ldab,mb,ldmb,sigmar,sigmai,nconv,dr,di,v,ldv,resid, &
        v,ldv,comm,icomm,ifail)

      If (ifail/=0) Then
        Go To 100
      End If

!     Compute the residual norm  ||A*x - lambda*x||.

      first = .True.
      Allocate (ax(n),mx(n),d_print(nconv,3))
      d_print(1:nconv,1) = dr(1:nconv)
      d_print(1:nconv,2) = di(1:nconv)

      Do j = 1, nconv

        If (di(j)==zero) Then

!         The NAG name equivalent of dgbmv is f06pbf
          Call dgbmv('N',n,n,kl,ku,one,ab(kl+1,1),ldab,v(1,j),1,zero,ax,1)

          Call dgbmv('N',n,n,kl,ku,one,mb(kl+1,1),ldmb,v(1,j),1,zero,mx,1)

          ax(1:n) = -dr(j)*mx(1:n) + ax(1:n)
          d_print(j,3) = dnrm2(n,ax,1)/abs(dr(j))
        Else If (first) Then

          Call dgbmv('N',n,n,kl,ku,one,ab(kl+1,1),ldab,v(1,j),1,zero,ax,1)

          Call dgbmv('N',n,n,kl,ku,one,mb(kl+1,1),ldmb,v(1,j),1,zero,mx,1)

          ax(1:n) = -dr(j)*mx(1:n) + ax(1:n)

          Call dgbmv('N',n,n,kl,ku,one,mb(kl+1,1),ldmb,v(1,j+1),1,zero,mx,1)

          ax(1:n) = di(j)*mx(1:n) + ax(1:n)
          d_print(j,3) = dnrm2(n,ax,1)

          Call dgbmv('N',n,n,kl,ku,one,ab(kl+1,1),ldab,v(1,j+1),1,zero,ax,1)

          Call dgbmv('N',n,n,kl,ku,one,mb(kl+1,1),ldmb,v(1,j+1),1,zero,mx,1)

          ax(1:n) = -dr(j)*mx(1:n) + ax(1:n)

          Call dgbmv('N',n,n,kl,ku,one,mb(kl+1,1),ldmb,v(1,j),1,zero,mx,1)

          ax(1:n) = -di(j)*mx(1:n) + ax(1:n)
!         The NAG name equivalent of dnrm2 is f06ejf
          d_print(j,3) = f06bnf(d_print(j,3),dnrm2(n,ax,1))
          d_print(j,3) = d_print(j,3)/f06bnf(dr(j),di(j))
          d_print(j+1,3) = d_print(j,3)
          first = .False.
        Else
          first = .True.
        End If

      End Do

      Write (nout,*)
      Flush (nout)

      outchn = nout
```

```
      Call x04abf(iset,outchn)

      If (printr) Then
!       Print residual associated with each Ritz value.
        ncol = 3
      Else
        ncol = 2
      End If
      ifail = 0
      Call x04caf('G','N',nconv,ncol,d_print,nconv,                          &
        ' Ritz values closest to sigma',ifail)

100   Continue
    End Program f12agfe
```

## 10.2 Program Data

```
F12AGF Example Program Data
 10 4 10 0.4 0.6 : Values for NX NEV NCV SIGMAR SIGMAI
```

## 10.3 Program Results

```
 F12AGF Example Program Results


  Ritz values closest to sigma
          1       2
 1   0.3610  0.7223
 2   0.3610 -0.7223
 3   0.4598 -0.7199
 4   0.4598  0.7199
```