NAG Library Routine Document

F12AUF

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 F12ARF need not be called. If, however, you wish to reset some or all of the settings please refer to Section 10 in F12ARF for a detailed description of the specification of the optional parameters*.

1 Purpose

F12AUF is the main solver routine in a suite of routines consisting of F12ARF, F12ATF and F12AUF. It must be called following an initial call to F12ATF and following any calls to F12ARF.

F12AUF returns approximations to selected eigenvalues, and (optionally) the corresponding eigenvectors, of a standard or generalized eigenvalue problem defined by complex banded non-Hermitian matrices. The banded matrix must be stored using the LAPACK storage format for complex banded non-Hermitian matrices.

2 Specification

```
SUBROUTINE F12AUF (KL, KU, AB, LDAB, MB, LDMB, SIGMA, NCONV, D, Z, LDZ,         &
                   RESID, V, LDV, COMM, ICOMM, IFAIL)

INTEGER                 KL, KU, LDAB, LDMB, NCONV, LDZ, LDV, ICOMM(*), IFAIL
COMPLEX (KIND=nag_wp) AB(LDAB,*), MB(LDMB,*), SIGMA, D(*), Z(LDZ,*),           &
                      RESID(*), V(LDV,*), COMM(*)
```

3 Description

The suite of routines is designed to calculate some of the eigenvalues, λ, (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, complex and non-Hermitian.

Following a call to the initialization routine F12ATF, F12AUF 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 complex banded non-Hermitian 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 non-Hermitian matrices; please refer to Section 3.3.4 in the F07 Chapter Introduction for details on this storage format.

F12AUF is based on the banded driver routines **znbdr1** to **znbdr4** 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 non-Hermitian matrices is provided in Lehoucq and Scott (1996). This suite of routines offers the same functionality as the ARPACK banded driver software for complex non-Hermitian 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.

F12AUF, is a general purpose routine that must be called following initialization by F12ATF. F12AUF uses options, set either by default or explicitly by calling F12ARF, to return the converged approximations to selected eigenvalues and (optionally):

– the corresponding approximate eigenvectors;

– an orthonormal basis for the associated approximate invariant subspace;

– both.

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 Parameters

1: KL – INTEGER *Input*

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

Constraint: KL ≥ 0.

2: KU – INTEGER *Input*

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

Constraint: KU ≥ 0.

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

Note: the second dimension of the array AB must be at least $\max(1, \mathrm{N})$ (see F12ATF).

On entry: must contain the matrix A in LAPACK banded storage format for non-Hermitian 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 F12AUF is called.

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

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

Note: the second dimension of the array MB must be at least $\max(1, \mathrm{N})$ (see F12ATF).

On entry: must contain the matrix B in LAPACK banded storage format for non-Hermitian 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 F12AUF is called.

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

7: SIGMA – COMPLEX (KIND=nag_wp) *Input*

On entry: if the **Shifted Inverse** mode (see F12ARF) has been selected then SIGMA must contain the shift used; otherwise SIGMA is not referenced. Section 4.2 in the F12 Chapter Introduction describes the use of shift and invert transformations.

8: NCONV – INTEGER *Output*

On exit: the number of converged eigenvalues.

9: D(∗) – COMPLEX (KIND=nag_wp) array *Output*

Note: the dimension of the array D must be at least NEV (see F12ATF).

On exit: the first NCONV locations of the array D contain the converged approximate eigenvalues.

10: Z(LDZ,∗) – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array Z must be at least NEV if the default option **Vectors** = Ritz has been selected and at least 1 if the option **Vectors** = None or Schur has been selected (see F12ARF and F12ATF).

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 D. The complex eigenvector associated with the ith eigenvalue (stored in D(i)) is stored in the ith column of Z.

11: LDZ – INTEGER *Input*

On entry: the first dimension of the array Z as declared in the (sub)program from which F12AUF 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 ≥ 1.

12: RESID(∗) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the dimension of the array RESID must be at least N (see F12ATF).

On entry: need not be set unless the option **Initial Residual** has been set in a prior call to F12ARF in which case RESID must contain an initial residual vector.

On exit: contains the final residual vector.

13: V(LDV,∗) – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array V must be at least max(1, NCV) (see F12ATF).

On exit: if the option **Vectors** (see F12ARF) 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 jth Schur vector is stored in the ith column of V.

14: LDV – INTEGER *Input*

On entry: the first dimension of the array V as declared in the (sub)program from which F12AUF is called.

Constraint: LDV $\geq$ N.

15: COMM(∗) – COMPLEX (KIND=nag_wp) array *Communication Array*

On entry: must remain unchanged from the prior call to F12ARF and F12ATF.

On exit: contains no useful information.

16: ICOMM(∗) – INTEGER array *Communication Array*

On entry: must remain unchanged from the prior call to F12ARF and F12ATF.

On exit: contains no useful information.

17: 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, KL = ⟨*value*⟩.
Constraint: KL ≥ 0.

IFAIL = 2

On entry, KU = ⟨*value*⟩.
Constraint: KU ≥ 0.

IFAIL = 3

On entry, LDAB = ⟨*value*⟩, 2 × KL + KU + 1 = ⟨*value*⟩.
Constraint: LDAB ≥ 2 × KL + KU + 1.

IFAIL = 5

The maximum number of iterations ≤ 0, the option **Iteration Limit** has been set to ⟨*value*⟩.

IFAIL = 6

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

IFAIL = 7

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

IFAIL = 8

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

IFAIL = 9

On entry, LDZ = ⟨*value*⟩, N = ⟨*value*⟩ in F12AFF.
Constraint: LDZ ≥ N.

IFAIL = 10

On entry, **Vectors** = Select, 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. The size of the current Arnoldi factorization = ⟨*value*⟩.

IFAIL = 13

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

IFAIL = 14

During calculation of a real Schur form, there was a failure to compute a number of eigenvalues Please contact NAG.

IFAIL = 15

The computed Schur form could not be reordered by an internal call. Please contact NAG.

IFAIL = 16

Error in internal call to compute 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 matrix. 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 matrix. Please contact NAG.

IFAIL = 21

The maximum number of iterations has been reached. The maximum number of iterations = ⟨*value*⟩. The number of converged eigenvalues = ⟨*value*⟩.

IFAIL = 22

No shifts could be applied during a cycle of the implicitly restarted Lanczos iteration.

IFAIL = 23

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

IFAIL = −999

Dynamic memory allocation failed.

7 Accuracy

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

8 Further Comments

None.

9 Example

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

9.1 Program Text

```
    Program f12aufe

!     F12AUF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: dznrm2, f12arf, f12atf, f12auf, nag_wp, x04abf,   &
                             x04caf, zaxpy, zgbmv
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Complex (Kind=nag_wp), Parameter :: one = (1.0_nag_wp,0.0_nag_wp)
      Complex (Kind=nag_wp), Parameter :: zero = (0.0_nag_wp,0.0_nag_wp)
      Integer, Parameter               :: iset = 1, nin = 5, nout = 6
      Logical, Parameter               :: printr = .False.
!     .. Local Scalars ..
      Complex (Kind=nag_wp)            :: ch, sigma
      Real (Kind=nag_wp)               :: h, rho
      Integer                          :: i, idiag, ifail, isub, isup, j, kl,  &
                                          ku, lcomm, ldab, ldmb, ldv, licomm,  &
                                          lo, n, ncol, nconv, ncv, nev, nx,    &
                                          outchn
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: ab(:,:), ax(:), comm(:), d(:),     &
                                            mb(:,:), mx(:), resid(:), v(:,:)
      Real (Kind=nag_wp), Allocatable  :: d_print(:,:)
      Integer, Allocatable             :: icomm(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: abs, aimag, cmplx, int, max, real
!     .. Executable Statements ..
      Write (nout,*) 'F12AUF Example Program Results'
      Write (nout,*)
      Flush (nout)

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

      Read (nin,*) nx, nev, ncv
      Read (nin,*) sigma
      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 f12atf(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 f12atf(n,nev,ncv,icomm,licomm,comm,lcomm,ifail)

!     Set the mode.
```

```
      ifail = 0
      Call f12arf('SHIFTED INVERSE',icomm,comm,ifail)

!     Set problem type

      ifail = 0
      Call f12arf('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) = zero
      mb(1:ldmb,1:n) = zero

!     Main diagonal of A.

      idiag = kl + ku + 1
      ab(idiag,1:n) = cmplx(4.0_nag_wp,0.0_nag_wp,kind=nag_wp)
      mb(idiag,1:n) = ab(idiag,1)

!     First subdiagonal and superdiagonal of A.

      isup = kl + ku
      isub = kl + ku + 2
      rho = 100.0_nag_wp
      h = 1._nag_wp/real(nx+1,kind=nag_wp)
      ch = cmplx(0.5_nag_wp*h*rho,0.0_nag_wp,kind=nag_wp)

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

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

      End Do

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

!     KL-th subdiagonal and KU-th super-diagonal.

      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 (d(nev),v(ldv,ncv),resid(n))
```

```
      ifail = -1
      Call f12auf(kl,ku,ab,ldab,mb,ldmb,sigma,nconv,d,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||.

      Allocate (ax(n),mx(n),d_print(nconv,3))
      d_print(1:nconv,1) = real(d(1:nconv))
      d_print(1:nconv,2) = aimag(d(1:nconv))

      Do j = 1, nconv

!       The NAG name equivalent of zgbmv is f06sbf
        Call zgbmv('N',n,n,kl,ku,one,ab(kl+1,1),ldab,v(1,j),1,zero,ax,1)

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

        Call zaxpy(n,-d(j),mx,1,ax,1)
        d_print(j,3) = dznrm2(n,ax,1)/abs(d(j))

      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 f12aufe
```

9.2 Program Data

```
F12AUF Example Program Data
  10    4    10              : nx nev ncv
 ( 0.4, 0.6)                 : sigma
```

9.3 Program Results

```
 F12AUF Example Program Results


  Ritz values closest to sigma
          1       2
 1   0.3610  0.7223
 2   0.4598  0.7199
 3   0.2868  0.7241
 4   0.2410  0.7257
```