NAG Library Routine Document

F12AQF

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 11 in F12ARF for a detailed description of the specification of the optional parameters.*

1 Purpose

F12AQF is a post-processing routine in a suite of routines consisting of F12ANF, F12APF, F12AQF, F12ARF and F12ASF, that must be called following a final exit from F12AQF.

2 Specification

```
SUBROUTINE F12AQF (NCONV, D, Z, LDZ, SIGMA, RESID, V, LDV, COMM, ICOMM,    &
                   IFAIL)

INTEGER                 NCONV, LDZ, LDV, ICOMM(*), IFAIL
COMPLEX (KIND=nag_wp) D(*), Z(LDZ,*), SIGMA, 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 sparse, complex and nonsymmetric. The suite can also be used to find selected eigenvalues/eigenvectors of smaller scale dense, complex and nonsymmetric problems.

Following a call to F12APF, F12AQF 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 nonsymmetric matrices. There is negligible additional cost to obtain eigenvectors; an orthonormal basis is always computed, but there is an additional storage cost if both are requested.

F12AQF is based on the routine **zneupd** 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 software for complex nonsymmetric problems, but the interface design is quite different in order to make the option setting clearer and to simplify some of the interfaces.

F12AQF is a post-processing routine that must be called following a successful final exit from F12APF. F12AQF uses data returned from F12APF and 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: NCONV – INTEGER *Output*

On exit: the number of converged eigenvalues as found by F12ARF.

2: D($*$) – COMPLEX (KIND=nag_wp) array *Output*

Note: the dimension of the array D must be at least NCV (see F12ANF).

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

3: 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 F12ANF).

On exit: if the default option **Vectors** $=$ RITZ (see F12ADF) has been selected then Z contains the final set of eigenvectors corresponding to the eigenvalues held in D. The complex eigenvector associated with an eigenvalue is stored in the corresponding column of Z.

4: LDZ – INTEGER *Input*

On entry: the first dimension of the array Z as declared in the (sub)program from which F12AQF 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.

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

On entry: if one of the **Shifted Inverse** (see F12ARF) modes has been selected then SIGMA contains the shift used; otherwise SIGMA is not referenced.

6: RESID($*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the dimension of the array RESID must be at least N (see F12ANF).

On entry: must not be modified following a call to F12APF since it contains data required by F12AQF.

7: V(LDV, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array V must be at least $\max(1, \text{NCV})$ (see F12ANF).

On entry: the NCV columns of V contain the Arnoldi basis vectors for OP as constructed by F12APF.

On exit: if the option **Vectors** = SCHUR or RITZ has been set and a separate array Z has been passed (i.e., Z does not equal V), then the first NCONV columns of V will contain approximate Schur vectors that span the desired invariant subspace.

8: LDV – INTEGER *Input*

On entry: the first dimension of the array V as declared in the (sub)program from which F12AQF is called.

Constraint: LDV $\geq$ N.

9: COMM($*$) – COMPLEX (KIND=nag_wp) array *Communication Array*

Note: the dimension of the array COMM must be at least $\max(1, \mathrm{LCOMM})$ (see F12ANF).

On initial entry: must remain unchanged from the prior call to F12ANF.

On exit: contains data on the current state of the solution.

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

Note: the dimension of the array ICOMM must be at least $\max(1, \mathrm{LICOMM})$ (see F12ANF).

On initial entry: must remain unchanged from the prior call to F12ANF.

On exit: contains data on the current state of the solution.

11: 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, LDZ $< \max(1, \mathrm{N})$ or LDZ < 1 when no vectors are required.

IFAIL $= 2$

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

IFAIL $= 3$

The number of eigenvalues found to sufficient accuracy prior to calling F12AQF, as communicated through the parameter ICOMM, is zero.

IFAIL $= 4$

The number of converged eigenvalues as calculated by F12APF differ from the value passed to it through the parameter ICOMM.

IFAIL $= 5$

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

IFAIL $= 6$

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

IFAIL $= 7$

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

IFAIL $= 8$

Either the solver routine F12APF has not been called prior to the call of this routine or a communication array has become corrupted.

IFAIL $= 9$

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

IFAIL $= 10$

An unexpected error has occurred. Please contact NAG.

IFAIL $= -99$

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

See Section 3.8 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

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 Parallelism and Performance

F12AQF is not threaded by NAG in any implementation.

F12AQF 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 solves $Ax = \lambda Bx$ in regular-invert mode, where A and B are derived from the standard central difference discretization of the one-dimensional convection-diffusion operator $\frac{d^2u}{dx^2} + \rho\frac{du}{dx}$ on $[0, 1]$, with zero Dirichlet boundary conditions.

10.1 Program Text

```
!   F12AQF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

    Module f12aqfe_mod

!     F12AQF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                               :: av, mv
!     .. Parameters ..
      Complex (Kind=nag_wp), Parameter, Public :: four = (4.0_nag_wp,          &
                                              0.0_nag_wp)
      Complex (Kind=nag_wp), Parameter, Public :: one = (1.0E+0_nag_wp,       &
                                              0.0E+0_nag_wp)
      Complex (Kind=nag_wp), Parameter     :: two = (2.0_nag_wp,0.0_nag_wp)
      Integer, Parameter, Public           :: imon = 0, licomm = 140,         &
                                             nerr = 6, nin = 5, nout = 6
    Contains
      Subroutine av(nx,v,w)

!       .. Parameters ..
        Complex (Kind=nag_wp), Parameter     :: rho = (10.0_nag_wp,0.0_nag_wp)
!       .. Scalar Arguments ..
        Integer, Intent (In)                 :: nx
!       .. Array Arguments ..
        Complex (Kind=nag_wp), Intent (In)   :: v(nx*nx)
        Complex (Kind=nag_wp), Intent (Out)  :: w(nx*nx)
!       .. Local Scalars ..
        Complex (Kind=nag_wp)                :: dd, dl, du, h, s
        Integer                              :: j, n
!       .. Intrinsic Procedures ..
        Intrinsic                            :: cmplx
!       .. Executable Statements ..
        n = nx*nx
        h = one/cmplx(n+1,kind=nag_wp)
        s = rho/two
        dd = two/h
        dl = -one/h - s
        du = -one/h + s
        w(1) = dd*v(1) + du*v(2)
        Do j = 2, n - 1
          w(j) = dl*v(j-1) + dd*v(j) + du*v(j+1)
        End Do
        w(n) = dl*v(n-1) + dd*v(n)
        Return
      End Subroutine av

      Subroutine mv(nx,v,w)

!       .. Use Statements ..
        Use nag_library, Only: zscal
!       .. Scalar Arguments ..
        Integer, Intent (In)                 :: nx
!       .. Array Arguments ..
        Complex (Kind=nag_wp), Intent (In)   :: v(nx*nx)
```

```
        Complex (Kind=nag_wp), Intent (Out)  :: w(nx*nx)
!       .. Local Scalars ..
        Complex (Kind=nag_wp)                :: h
        Integer                              :: j, n
!       .. Intrinsic Procedures ..
        Intrinsic                            :: cmplx
!       .. Executable Statements ..
        n = nx*nx
        w(1) = four*v(1) + one*v(2)
        Do j = 2, n - 1
          w(j) = one*v(j-1) + four*v(j) + one*v(j+1)
        End Do
        w(n) = one*v(n-1) + four*v(n)
        h = one/cmplx(n+1,kind=nag_wp)
!       The NAG name equivalent of zscal is f06gdf
        Call zscal(n,h,w,1)
        Return
      End Subroutine mv
    End Module f12aqfe_mod
    Program f12aqfe

!     F12AQF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: dznrm2, f12anf, f12apf, f12aqf, f12arf, f12asf,   &
                             nag_wp, zgttrf, zgttrs
      Use f12aqfe_mod, Only: av, four, imon, licomm, mv, nerr, nin, nout, one
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Complex (Kind=nag_wp)                :: h, sigma
      Integer                              :: ifail, ifail1, info, irevcm, j,  &
                                              lcomm, ldv, n, nconv, ncv, nev,  &
                                              niter, nshift, nx
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable   :: comm(:), d(:,:), dd(:), dl(:),   &
                                              du(:), du2(:), mx(:), resid(:),  &
                                              v(:,:), x(:)
      Integer                              :: icomm(licomm)
      Integer, Allocatable                 :: ipiv(:)
!     .. Intrinsic Procedures ..
      Intrinsic                            :: cmplx
!     .. Executable Statements ..
      Write (nout,*) 'F12AQF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) nx, nev, ncv

      n = nx*nx
      lcomm = 3*n + 3*ncv*ncv + 5*ncv + 60
      ldv = n
      Allocate (comm(lcomm),d(ncv,2),dd(n),dl(n),du(n),du2(n),mx(n),resid(n), &
        v(ldv,ncv),x(n),ipiv(n))

      ifail = 0
      Call f12anf(n,nev,ncv,icomm,licomm,comm,lcomm,ifail)

!     Set the mode.
      ifail = 0
      Call f12arf('REGULAR INVERSE',icomm,comm,ifail)
!     Set problem type.
      Call f12arf('GENERALIZED',icomm,comm,ifail)
!     Use pointers to Workspace rather than interfacing through the array X.
      Call f12arf('POINTERS=YES',icomm,comm,ifail)
      h = one/cmplx(n+1,kind=nag_wp)

      dl(1:n-1) = h
      dd(1:n-1) = four*h
      du(1:n-1) = h
      dd(n) = four*h
```

```
!     The NAG name equivalent of zgttrf is f07crf
      Call zgttrf(n,dl,dd,du,du2,ipiv,info)
      If (info/=0) Then
        Write (nerr,99999) info
        Go To 100
      End If

      irevcm = 0
      ifail = -1
revcm: Do
        Call f12apf(irevcm,resid,v,ldv,x,mx,nshift,comm,icomm,ifail)
        If (irevcm==5) Then
          Exit revcm
        Else If (irevcm==-1 .Or. irevcm==1) Then
!         Perform  y <--- OP*x = inv[M]*A*x      |
          Call av(nx,comm(icomm(1)),comm(icomm(2)))
!         The NAG name equivalent of zgttrs is f07csf
          Call zgttrs('N',n,1,dl,dd,du,du2,ipiv,comm(icomm(2)),n,info)
          If (info/=0) Then
            Write (nerr,99998) info
            Exit revcm
          End If
        Else If (irevcm==2) Then
!         Perform  y <--- M*x
          Call mv(nx,comm(icomm(1)),comm(icomm(2)))
        Else If (irevcm==4 .And. imon/=0) Then
!         Output monitoring information
          Call f12asf(niter,nconv,d,d(1,2),icomm,comm)
!         The NAG name equivalent of dznrm2 is f06jjf
          Write (6,99997) niter, nconv, dznrm2(nev,d(1,2),1)
        End If
      End Do revcm

      If (ifail==0 .And. info==0) Then
!       Post-Process using F12AQF to compute eigenvalues/vectors.
        ifail1 = 0
        Call f12aqf(nconv,d,v,ldv,sigma,resid,v,ldv,comm,icomm,ifail1)
        Write (nout,99996) nconv
        Write (nout,99995)(j,d(j,1),j=1,nconv)
      End If
100   Continue

99999 Format (1X,'** Error status returned by ZGTTRF, INFO =',I12)
99998 Format (1X,'** Error status returned by ZGTTRS, INFO =',I12)
99997 Format (1X,'Iteration',1X,I3,', No. converged =',1X,I3,', norm o', &
        'f estimates =',E16.8)
99996 Format (1X/' The ',I4,' Ritz values of largest magnitude are:'/)
99995 Format (1X,I8,5X,'( ',F12.4,' , ',F12.4,' )')
    End Program f12aqfe
```

10.2 Program Data

```
F12AQF Example Program Data
 10 4 20 : Vaues for NX NEV and NCV
```

10.3 Program Results

```
 F12AQF Example Program Results


 The    4 Ritz values of largest magnitude are:

        1     (   20383.0384 ,       0.0000 )
        2     (   20338.7563 ,      -0.0000 )
        3     (   20265.2844 ,       0.0000 )
        4     (   20163.1142 ,      -0.0000 )
```