NAG Toolbox for MATLAB

f12ac

1 Purpose

f12ac is a post-processing function in a suite of functions consisting of f12aa, f12ab, f12ac, f12ad and f12ae, that must be called following a final exit from f12ab.

2 Syntax

```
[nconv, dr, di, z, v, comm, icomm, ifail] = f12ac(sigmar, sigmai, resid,
v, comm, icomm)
```

3 Description

The suite of functions 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, real and nonsymmetric. The suite can also be used to find selected eigenvalues/eigenvectors of smaller scale dense, real and nonsymmetric problems.

Following a call to f12ab, f12ac 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 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.

f12ac is based on the function **dneupd** 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 functions offers the same functionality as the ARPACK software for real nonsymmetric problems, but the interface design is quite different in order to make the option setting clearer to you and to simplify some of the interfaces.

f12ac, is a post-processing function that must be called following a successful final exit from f12ab. f12ac uses data returned from f12ab and options, set either by default or explicitly by calling f12ad, 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

5.1 Compulsory Input Parameters

1: **sigmar – double scalar**

If one of the **Shifted** modes have been selected then **sigmar** contains the real part of the shift used; otherwise **sigmar** is not referenced.

2: **sigmai – double scalar**

If one of the **Shifted** modes have been selected then **sigmai** contains the imaginary part of the shift used; otherwise **sigmai** is not referenced.

3: **resid**$(*)$ **– double array**

Note: the dimension of the array **resid** must be at least **n** (see f12aa).

Must not be modified following a call to f12ab since it contains data required by f12ac.

4: **v**(**ldv**,$*$) **– double array**

The first dimension of the array **v** must be at least **n**

The second dimension of the array must be at least max(1, **ncv**)

The **ncv** columns of **v** contain the Arnoldi basis vectors for OP as constructed by f12ab.

5: **comm**$(*)$ **– double array**

Note: the dimension of the array **comm** must be at least max(1, **lcomm**) (see f12aa).

On initial entry: must remain unchanged from the prior call to f12ab.

6: **icomm**$(*)$ **– int32 array**

Note: the dimension of the array **icomm** must be at least max(1, **licomm**) (see f12aa).

On initial entry: must remain unchanged from the prior call to f12ab.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

ldz, ldv

5.4 Output Parameters

1: **nconv – int32 scalar**

The number of converged eigenvalues as found by f12ab.

2: **dr**$(*)$ **– double array**

Note: the dimension of the array **dr** must be at least **nev** (see f12aa).

The first **nconv** locations of the array **dr** contain the real parts of the converged approximate eigenvalues.

3: **di**$(*)$ **– double array**

Note: the dimension of the array **di** must be at least **nev** (see f12aa).

The first **nconv** locations of the array **di** contain the imaginary parts of the converged approximate eigenvalues.

4: **z(n × (nev + 1)) – double array**

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.

5: **v(ldv,∗) – double array**

The first dimension of the array **v** must be at least **n**

The second dimension of the array must be at least max(1, **ncv**)

If the option **Vectors** = Schur has been set, or the option **Vectors** = 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.

6: **comm(∗) – double array**

Note: the dimension of the array **comm** must be at least max(1, **lcomm**) (see f12aa).

Contains data on the current state of the solution.

7: **icomm(∗) – int32 array**

Note: the dimension of the array **icomm** must be at least max(1, **licomm**) (see f12aa).

Contains data on the current state of the solution.

8: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

On entry, **ldz** < max(1, **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 f12ac, as communicated through the parameter **icomm**, is zero.

ifail = 4

The number of converged eigenvalues as calculated by f12ab differ from the value passed to it through the parameter **icomm**.

ifail = 5

Unexpected error during calculation of a real 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 function f12ab has not been called prior to the call of this function or a communication array has become corrupted.

ifail = 9

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

ifail = 10

An unexpected error has occurred. Please contact NAG.

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 x02aj.

8 Further Comments

None.

9 Example

```
n = int32(100);
nev = int32(4);
ncv = int32(20);

h = 1/(double(n)+1);
rho = 10;
md = repmat(4*h, double(n), 1);
me = repmat(h, double(n-1), 1);

irevcm = int32(0);
resid = zeros(n,1);
v = zeros(n, ncv);
x = zeros(n, 1);
mx = zeros(n);

dd = 2/h;
dl = -1/h - rho/2;
du = -1/h + rho/2;
y = zeros(n,1);

[icomm, comm, ifail] = f12aa(n, nev, ncv);
[icomm, comm, ifail] = f12ad('REGULAR INVERSE', icomm, comm);
[icomm, comm, ifail] = f12ad('GENERALIZED', icomm, comm);

% Construct m and factorise
[md, me, info] = f07jd(md, me);

while (irevcm ~= 5)
  [irevcm, resid, v, x, mx, nshift, comm, icomm, ifail] = ...
    f12ab(irevcm, resid, v, x, mx, comm, icomm);
    if (irevcm == -1 || irevcm == 1)
      y(1) = dd*x(1) + du*x(2);
      for i = 2:n-1
        y(i) = dl*x(i-1) + dd*x(i) + du*x(i+1);
      end
      y(n) = dl*x(n-1) + dd*x(n);
      [x, info] = f07je(md, me, y);
    elseif (irevcm == 2)
      y(1) = 4*x(1) + x(2);
      for i=2:n-1
        y(i) = x(i-1) + 4*x(i) + x(i+1);
      end
      y(n) = x(n-1) + 4*x(n);
      x = h*y;
    elseif (irevcm == 4)
      [niter, nconv, ritzr, ritzi, rzest] = f12ae(icomm, comm);
      if (niter == 1)
        fprintf('\n');
      end
        fprintf('Iteration  %2d  No.  converged  =  %d  Norm  of  estimates  =
%16.8e\n', niter, nconv, norm(rzest));
    end
  end
  [nconv, dr, di, z, v, comm, icomm, ifail] = f12ac(0, 0, resid, v, comm,
  icomm)
```

```
Iteration  1 No. converged = 0 Norm of estimates =   5.56325675e+03
Iteration  2 No. converged = 0 Norm of estimates =   5.44836588e+03
Iteration  3 No. converged = 0 Norm of estimates =   5.30320774e+03
Iteration  4 No. converged = 0 Norm of estimates =   6.24234186e+03
Iteration  5 No. converged = 0 Norm of estimates =   7.15674705e+03
Iteration  6 No. converged = 0 Norm of estimates =   5.45460864e+03
Iteration  7 No. converged = 0 Norm of estimates =   6.43147571e+03
Iteration  8 No. converged = 0 Norm of estimates =   5.11241161e+03
Iteration  9 No. converged = 0 Norm of estimates =   7.19327824e+03
Iteration 10 No. converged = 1 Norm of estimates =   5.77945489e+03
Iteration 11 No. converged = 2 Norm of estimates =   4.73125738e+03
Iteration 12 No. converged = 3 Norm of estimates =   5.00078500e+03
nconv =
           4
dr =
   1.0e+04 *
    2.0383
    2.0339
    2.0265
    2.0163
di =
     0
     0
     0
     0
z =
     array elided
v =
     array elided
comm =
     array elided
icomm =
     array elided
ifail =
           0
```