NAG Toolbox for MATLAB

f12ae

1 Purpose

f12ae can be used to return additional monitoring information during computation. It is in a suite of functions consisting of f12aa, f12ab, f12ac, f12ad and f12ae.

2 Syntax

```
[niter, nconv, ritzr, ritzi, rzest] = f12ae(icomm, comm)
```

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.

On an intermediate exit from f12ab with **irevcm** $= 4$, f12ae may be called to return monitoring information on the progress of the Arnoldi iterative process. The information returned by f12ae is:

– the number of the current Arnoldi iteration;

– the number of converged eigenvalues at this point;

– the real and imaginary parts of the converged eigenvalues;

– the error bounds on the converged eigenvalues.

f12ae does not have an equivalent function from the ARPACK package which prints various levels of detail of monitoring information through an output channel controlled via a parameter value (see Lehoucq *et al.* 1998 for details of ARPACK routines). f12ae should not be called at any time other than immediately following an **irevcm** $= 4$ return from f12ab.

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: **icomm**$(*)$ **– int32 array**

Note: the dimension of the array **icomm** must be at least $\max(1, \mathbf{licomm})$, where **licomm** is passed to the setup function f12aa (see f12aa).

The array **icomm** output by the preceding call to f12ab.

2: **comm**$(*)$ **– double array**

Note: the dimension of the array **comm** must be at least $\max(1, 3 \times \mathbf{n} + 3 \times \mathbf{ncv} \times \mathbf{ncv} + 6 \times \mathbf{ncv} + 60)$ (see f12aa).

The array **comm** output by the preceding call to f12ab.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **niter – int32 scalar**

The number of the current Arnoldi iteration.

2: **nconv – int32 scalar**

The number of converged eigenvalues so far.

3: **ritzr**$(*)$ **– double array**

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

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

4: **ritzi**$(*)$ **– double array**

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

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

5: **rzest**$(*)$ **– double array**

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

The first **nconv** locations of the array **rzest** contain the Ritz estimates (error bounds) on the converged approximate eigenvalues.

6 Error Indicators and Warnings

None.

7 Accuracy

A Ritz value, λ, is deemed to have converged 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));
    converged(niter,1) = nconv;
    rzestNorms(niter,1) = 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
```