NAG Toolbox

nag_sparseig_complex_monit (f12as)

1 Purpose

nag_sparseig_complex_monit (f12as) can be used to return additional monitoring information during computation. It is in a suite of functions consisting of nag_sparseig_complex_init (f12an), nag_sparseig_complex_iter (f12ap), nag_sparseig_complex_proc (f12aq), nag_sparseig_complex_option (f12ar) and nag_sparseig_complex_monit (f12as).

2 Syntax

```
[niter, nconv, ritz, rzest] = nag_sparseig_complex_monit(icomm, comm)
[niter, nconv, ritz, rzest] = f12as(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 complex eigenvalue problem $Ax = \lambda x$, or of a generalized complex eigenvalue problem $Ax = \lambda Bx$ of order n, where n is large and the coefficient matrices A and B are sparse and complex. The suite can also be used to find selected eigenvalues/eigenvectors of smaller scale dense complex problems.

On an intermediate exit from nag_sparseig_complex_iter (f12ap) with **irevcm** $= 4$, nag_sparseig_complex_monit (f12as) may be called to return monitoring information on the progress of the Arnoldi iterative process. The information returned by nag_sparseig_complex_monit (f12as) is:

– the number of the current Arnoldi iteration;

– the number of converged eigenvalues at this point;

– the converged eigenvalues;

– the error bounds on the converged eigenvalues.

nag_sparseig_complex_monit (f12as) 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 argument value (see Lehoucq *et al.* (1998) for details of ARPACK routines). nag_sparseig_complex_monit (f12as) should not be called at any time other than immediately following an **irevcm** $= 4$ return from nag_sparseig_complex_iter (f12ap).

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**(:) – INTEGER array

The dimension of the array **icomm** must be at least $\max(1, \mathbf{licomm})$, where **licomm** is passed to the setup function (see nag_sparseig_complex_init (f12an))

The array **icomm** output by the preceding call to nag_sparseig_complex_iter (f12ap).

2: **comm**(:) – COMPLEX (KIND=nag_wp) array

The dimension of the array **comm** must be at least $\max(1, \mathbf{lcomm})$, where **lcomm** is passed to the setup function (see nag_sparseig_complex_init (f12an))

The array **comm** output by the preceding call to nag_sparseig_complex_iter (f12ap).

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **niter** – INTEGER

The number of the current Arnoldi iteration.

2: **nconv** – INTEGER

The number of converged eigenvalues so far.

3: **ritz**(:) – COMPLEX (KIND=nag_wp) array

The dimension of the array **ritz** will be **ncv** (see nag_sparseig_complex_init (f12an))

The first **nconv** locations of the array **ritz** contain the converged approximate eigenvalues.

4: **rzest**(:) – COMPLEX (KIND=nag_wp) array

The dimension of the array **rzest** will be **ncv** (see nag_sparseig_complex_init (f12an))

The first **nconv** locations of the array **rzest** contain the complex Ritz estimates on the converged approximate eigenvalues.

6 Error Indicators and Warnings

None.

7 Accuracy

A Ritz value, λ, is deemed to have converged if the magnitude of its Ritz estimate $\leq$ **Tolerance** $\times |\lambda|$. The default **Tolerance** used is the ***machine precision*** given by nag_machine_precision (x02aj).

8 Further Comments

None.

9 Example

This example solves $Ax = \lambda Bx$ in shifted-inverse mode, where A and B are obtained 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. The shift, σ, is a complex number, and the operator used in the shifted-inverse iterative process is $\mathrm{OP} = \mathrm{inv}(A - \sigma B) \times B$.

9.1 Program Text

```
     function f12as_example

fprintf('f12as example results\n\n');

nx  = nag_int(16);
n   = nx^2;
nev = nag_int(4);
ncv = nag_int(10);

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

sigma = complex(5000);
rho = complex(10);
h = 1/double(n+1);
s = rho/2;
s1 = -1/h - s - sigma*h/6;
s2 =  2/h - 4*sigma*h/6;
s3 = -1/h + s - sigma*h/6;

dl = complex(repmat(s1, n-1, 1));
dd = complex(repmat(s2, n, 1));
du = complex(repmat(s3, n-1, 1));

% Initialisation Step
[icomm, comm, ifail] = f12an( ...
                              n, nev, ncv);

% Set the mode
[icomm, comm, ifail] = f12ar( ...
                              'SHIFTED INVERSE', icomm, comm);
% Set the problem type
[icomm, comm, ifail] = f12ar( ...
                              'GENERALIZED', icomm, comm);

[dl, dd, du, du2, ipiv, info] = f07cr( ...
                                       dl, dd, du);

% Solve
while (irevcm ~= 5)
  [irevcm, resid, v, x, mx, nshift, comm, icomm, ifail] = ...
      f12ap(...
            irevcm, resid, v, x, mx, comm, icomm);
  if (irevcm == -1)
    x = mv(x);
    [x, info] = f07cs(...
                      'N', dl, dd, du, du2, ipiv, x);
  elseif (irevcm == 1)
    [x, info] = f07cs(...
                      'N', dl, dd, du, du2, ipiv, mx);
  elseif (irevcm == 2)
    x = mv(x);
  elseif (irevcm == 4)
    [niter, nconv, ritz, rzest] = f12as(icomm, comm);
    fprintf('Iteration %d, No. converged = %d, ', niter, nconv);
    fprintf('norm of estimates = %12.2g\n', norm(rzest(1:nev),1));
  end
```

```
end

% Post-process to compute eigenvalues/vectors
[nconv, d, z, v, comm, icomm, ifail] = ...
    f12aq( ...
           sigma, resid, v, comm, icomm);

fprintf('\n\nThe %d generalised Ritz values closest to %7.1f are:\n', ...
        nconv, sigma);
for i=1:nconv
  fprintf('  %8.2f  %+8.2fi\n',real(d(i)),imag(d(i)));
end

function [w] = mv(v)
  n = numel(v);
  w = complex(zeros(n,1));
  h = 1/(n+1);

  w(1) = h*(4*v(1)+v(2))/6;
  for j=2:n-1
    w(j)=h*(v(j-1)+4*v(j)+v(j+1))/6;
  end
  w(n) = h*(v(n-1)+4*v(n))/6;
```

9.2 Program Results

```
     f12as example results

Iteration 1, No. converged = 0, norm of estimates =      7.9e-07
Iteration 2, No. converged = 1, norm of estimates =      1.7e-09
Iteration 3, No. converged = 2, norm of estimates =      3.4e-11
Iteration 4, No. converged = 2, norm of estimates =      6.2e-14
Iteration 5, No. converged = 2, norm of estimates =      8.5e-16
Iteration 6, No. converged = 3, norm of estimates =      8.2e-18

The 4 generalised Ritz values closest to  5000.0 are:
   4829.85     -0.00i
   5279.52     +0.00i
   4400.63     +0.00i
   5749.72     +0.00i
```