NAG Toolbox

nag_sparseig_complex_proc (f12aq)

1 Purpose

nag_sparseig_complex_proc (f12aq) is a post-processing function 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), that must be called following a final exit from nag_sparseig_complex_proc (f12aq).

2 Syntax

```
[nconv, d, z, v, comm, icomm, ifail] = nag_sparseig_complex_proc(sigma, resid, v, comm, icomm)
[nconv, d, z, v, comm, icomm, ifail] = f12aq(sigma, 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, 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 nag_sparseig_complex_iter (f12ap), nag_sparseig_complex_proc (f12aq) 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.

nag_sparseig_complex_proc (f12aq) is based on the function **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 functions 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.

nag_sparseig_complex_proc (f12aq) is a post-processing function that must be called following a successful final exit from nag_sparseig_complex_iter (f12ap). nag_sparseig_complex_proc (f12aq) uses data returned from nag_sparseig_complex_iter (f12ap) and options set either by default or explicitly by calling nag_sparseig_complex_option (f12ar), 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: **sigma** – COMPLEX (KIND=nag_wp)

If one of the **Shifted Inverse** (see nag_sparseig_complex_option (f12ar)) modes has been selected then **sigma** contains the shift used; otherwise **sigma** is not referenced.

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

The dimension of the array **resid** must be at least **n** (see nag_sparseig_complex_init (f12an))

Must not be modified following a call to nag_sparseig_complex_iter (f12ap) since it contains data required by nag_sparseig_complex_proc (f12aq).

3: **v**(ldv,:) – COMPLEX (KIND=nag_wp) array

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

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

The **ncv** columns of **v** contain the Arnoldi basis vectors for OP as constructed by nag_sparseig_complex_iter (f12ap).

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

The dimension of the array **comm** must be at least max(1, **lcomm**) (see nag_sparseig_complex_init (f12an))

On initial entry: must remain unchanged from the prior call to nag_sparseig_complex_init (f12an).

5: **icomm**(:) – INTEGER array

The dimension of the array **icomm** must be at least max(1, **licomm**) (see nag_sparseig_complex_init (f12an))

On initial entry: must remain unchanged from the prior call to nag_sparseig_complex_init (f12an).

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **nconv** – INTEGER

The number of converged eigenvalues as found by nag_sparseig_complex_option (f12ar).

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

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

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

3: **z**($\mathbf{n} \times \mathbf{ncv}$) – COMPLEX (KIND=nag_wp) array

If the default option **Vectors** = RITZ (see nag_sparseig_real_option (f12ad)) 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: **v**(ldv, :) – COMPLEX (KIND=nag_wp) array

The first dimension of the array **v** will be **n**.

The second dimension of the array **v** will be $\max(1, \mathbf{ncv})$.

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.

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

The dimension of the array **comm** will be $\max(1, \mathbf{lcomm})$ (see nag_sparseig_complex_init (f12an))

Contains data on the current state of the solution.

6: **icomm**(:) – INTEGER array

The dimension of the array **icomm** will be $\max(1, \mathbf{licomm})$ (see nag_sparseig_complex_init (f12an))

Contains data on the current state of the solution.

7: **ifail** – INTEGER

ifail = 0 unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

On entry, $ldz < \max(1, \mathbf{n})$ 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 nag_sparseig_complex_proc (f12aq), as communicated through the argument **icomm**, is zero.

ifail = 4

The number of converged eigenvalues as calculated by nag_sparseig_complex_iter (f12ap) differ from the value passed to it through the argument **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 function nag_sparseig_complex_iter (f12ap) 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.

ifail $= -99$

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

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

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 nag_machine_precision (x02aj).

8 Further Comments

None.

9 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.

9.1 Program Text

```
function f12aq_example

fprintf('f12aq example results\n\n');

global rho;

n   = nag_int(100);
nev = nag_int(4);
ncv = nag_int(20);
rho = 10;
imon = 0;

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

```
z  = complex(zeros(n,1));

% Initialisation Step
[icomm, comm, ifail] = f12an( ...
                              n, nev, ncv);
[icomm, comm, ifail] = f12ar( ...
                              'Regular Inverse', icomm, comm);
[icomm, comm, ifail] = f12ar( ...
                              'Generalized', icomm, comm);

% Factorize B
h  = 1/double(n+1);
cl = complex(h*ones(n-1,1));
cd = complex(4*h*ones(n,1));
cu = cl;

[cl, cd, cu, cu2, ipiv, info] = f07cr( ...
                                       cl, cd, cu);

% Solve
while (irevcm ~= 5)
  [irevcm, resid, v, x, mx, nshift, comm, icomm, ifail] = ...
    f12ap( ...
           irevcm, resid, v, x, mx, comm, icomm);
  if (irevcm == 1 || irevcm == -1)
    % Solve By = Ax;
    ax = f12aq_Ax(n, x);
    [x, info] = f07cs( ...
                       'N', cl, cd, cu, cu2, ipiv, ax);
  elseif (irevcm == 2)
    % y = Bx
    x = f12aq_Ax(n, x);
  elseif (irevcm == 4 && imon==1)
    [niter, nconv, ritz, rzest] = f12as( ...
                                         icomm, comm);
    fprintf(['Iteration %2d, No. converged = %d, ', ...
             'norm of estimates = %10.2e\n'], ...
            niter, nconv, norm(rzest(1:nev),2));
  end
end

% Post-process to compute eigenvalues/vectors
sigma = complex(0);
[nconv, d, z, v, comm, icomm, ifail] = ...
f12aq( ...
       sigma, resid, v, comm, icomm);

fprintf('Largest %d Eigenvalues are:\n',nconv);
fprintf('%12.4f%+12.4fi\n',[real(d(1:nconv)) imag(d(1:nconv))]');

function [y] = f12aq_Ax(n, x)

  global rho;

  y = complex(ones(n,1));

  h = 1/double(n+1);
  s = rho/2;
  dd = complex(2/h);
  dl = complex(-1/h - s);
  du = complex(-1/h + s);

  y(1) = dd*x(1) + du*x(2);
  for j=2:n-1
    y(j) = dl*x(j-1) + dd*x(j) +du*x(j+1);
  end
  y(n) = dl*x(n-1) + dd*x(n);

function [y] = f12aq_Bx(n,x)

  y = complex(ones(n,1));
```

```
  h = 1/double(n+1);
  dd = complex(4*h);
  dl = complex(h);
  du = complex(h);
  y(1) = dd*x(1) +du*x(2);
  for j=2:n-1
    y(j) = dl*x(j-1) + dd*x(j) +du*x(j+1);
  end
  y(n) = dl*x(n-1) + dd*x(n);
```

9.2 Program Results

```
     f12aq example results

Largest 4 Eigenvalues are:
  20383.0384     -0.0000i
  20338.7563     +0.0000i
  20265.2844     -0.0000i
  20163.1142     +0.0000i
```