NAG Toolbox

nag_sparseig_complex_iter (f12ap)

1 Purpose

nag_sparseig_complex_iter (f12ap) is an iterative solver 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). It is used to find some of the eigenvalues (and optionally the corresponding eigenvectors) of a standard or generalized eigenvalue problem defined by complex nonsymmetric matrices.

2 Syntax

```
[irevcm, resid, v, x, mx, nshift, comm, icomm, ifail] =
nag_sparseig_complex_iter(irevcm, resid, v, x, mx, comm, icomm)
```

```
[irevcm, resid, v, x, mx, nshift, comm, icomm, ifail] = f12ap(irevcm, resid, v,
x, mx, 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.

nag_sparseig_complex_iter (f12ap) is a **reverse communication** function, based on the ARPACK routine **znaupd**, using 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 the interface of nag_sparseig_complex_iter (f12ap).

The setup function nag_sparseig_complex_init (f12an) must be called before nag_sparseig_complex_iter (f12ap), the reverse communication iterative solver. Options may be set for nag_sparseig_complex_iter (f12ap) by prior calls to the option setting function nag_sparseig_complex_option (f12ar) and a post-processing function nag_sparseig_complex_proc (f12aq) must be called following a successful final exit from nag_sparseig_complex_iter (f12ap). nag_sparseig_complex_monit (f12as) may be called following certain flagged intermediate exits from nag_sparseig_complex_iter (f12ap) to provide additional monitoring information about the computation.

nag_sparseig_complex_iter (f12ap) uses **reverse communication**, i.e., it returns repeatedly to the calling program with the argument **irevcm** (see Section 5) set to specified values which require the calling program to carry out one of the following tasks:

– compute the matrix-vector product $y = \mathrm{OP}x$, where OP is defined by the computational mode;

– compute the matrix-vector product $y = Bx$;

– notify the completion of the computation;

– allow the calling program to monitor the solution.

The problem type to be solved (standard or generalized), the spectrum of eigenvalues of interest, the mode used (regular, regular inverse, shifted inverse, shifted real or shifted imaginary) and other options can all be set using the option setting function nag_sparseig_complex_option (f12ar) (see Section 11.1 in nag_sparseig_complex_option (f12ar) for details on setting options and of the default settings).

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

Note: this function uses **reverse communication.** Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the argument **irevcm**. Between intermediate exits and re-entries, **all arguments other than x, mx and comm must remain unchanged**.

5.1 Compulsory Input Parameters

1: **irevcm** – INTEGER

On initial entry: **irevcm** $= 0$, otherwise an error condition will be raised.

On intermediate re-entry: must be unchanged from its previous exit value. Changing **irevcm** to any other value between calls will result in an error.

Constraint: on initial entry, **irevcm** $= 0$; on re-entry **irevcm** must remain unchanged.

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

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

On initial entry: need not be set unless the option **Initial Residual** has been set in a prior call to nag_sparseig_complex_option (f12ar) in which case **resid** should contain an initial residual vector, possibly from a previous run.

On intermediate re-entry: must be unchanged from its previous exit. Changing **resid** to any other value between calls may result in an error exit.

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 $\max(1,\mathbf{ncv})$.

On initial entry: need not be set.

On intermediate re-entry: must be unchanged from its previous exit.

4: **x**$(:)$ – COMPLEX (KIND=nag_wp) array

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

On initial entry: need not be set, it is used as a convenient mechanism for accessing elements of **comm**.

On intermediate re-entry: if **Pointers** $=$ YES, **x** need not be set.

If **Pointers** $=$ NO, **x** must contain the result of $y = \mathrm{OP}x$ when **irevcm** returns the value -1 or $+1$. It must return the computed shifts when **irevcm** returns the value 3.

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

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

On initial entry: need not be set, it is used as a convenient mechanism for accessing elements of **comm**.

On intermediate re-entry: if **Pointers** = YES, **mx** need not be set.

If **Pointers** = NO, **mx** must contain the result of $y = Bx$ when **irevcm** returns the value 2.

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

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

On initial entry: must remain unchanged following a call to the setup function nag_sparseig_complex_init (f12an).

7: **icomm**(:) – INTEGER array

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

On initial entry: must remain unchanged following a call to the setup function nag_sparseig_complex_init (f12an).

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **irevcm** – INTEGER

On intermediate exit: has the following meanings.

irevcm $= -1$
The calling program must compute the matrix-vector product $y = \mathrm{OP}x$, where x is stored in **x** (by default) or in the array **comm** (starting from the location given by the first element of **icomm**) when the option **Pointers** = YES is set in a prior call to nag_sparseig_complex_option (f12ar). The result y is returned in **x** (by default) or in the array **comm** (starting from the location given by the second element of **icomm**) when the option **Pointers** = YES is set.

irevcm $= 1$
The calling program must compute the matrix-vector product $y = \mathrm{OP}x$. This is similar to the case **irevcm** $= -1$ except that the result of the matrix-vector product Bx (as required in some computational modes) has already been computed and is available in **mx** (by default) or in the array **comm** (starting from the location given by the third element of **icomm**) when the option **Pointers** = YES is set.

irevcm $= 2$
The calling program must compute the matrix-vector product $y = Bx$, where x is stored in **x** and y is returned in **mx** (by default) or in the array **comm** (starting from the location given by the second element of **icomm**) when the option **Pointers** = YES is set.

irevcm $= 3$
Compute the **nshift** complex shifts. This value of **irevcm** will only arise if the optional parameter **Supplied Shifts** is set in a prior call to nag_sparseig_complex_option (f12ar) which is intended for experienced users only; the default and recommended option is to use exact shifts (see Lehoucq *et al.* (1998) for details).

irevcm $= 4$
Monitoring step: a call to nag_sparseig_complex_monit (f12as) can now be made to return the number of Arnoldi iterations, the number of converged Ritz values, the array of converged values, and the corresponding Ritz estimates.

On final exit: **irevcm** $= 5$: nag_sparseig_complex_iter (f12ap) has completed its tasks. The value of **ifail** determines whether the iteration has been successfully completed, or whether errors have been detected. On successful completion nag_sparseig_complex_proc (f12aq) must be called to return the requested eigenvalues and eigenvectors (and/or Schur vectors).

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

The dimension of the array **resid** will be **n** (see nag_sparseig_complex_init (f12an))

On intermediate exit: contains the current residual vector.

On final exit: contains the final residual vector.

3: **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 $\max(1, \mathbf{ncv})$.

On intermediate exit: contains the current set of Arnoldi basis vectors.

On final exit: contains the final set of Arnoldi basis vectors.

4: **x**$(:)$ – COMPLEX (KIND=nag_wp) array

The dimension of the array **x** will be **n** (see nag_sparseig_complex_init (f12an))

On intermediate exit: if **Pointers** $=$ YES, **x** is not referenced.

If **Pointers** $=$ NO, **x** contains the vector x when **irevcm** returns the value -1 or $+1$.

On final exit: does not contain useful data.

5: **mx**$(:)$ – COMPLEX (KIND=nag_wp) array

The dimension of the array **mx** will be **n** (see nag_sparseig_complex_init (f12an))

On intermediate exit: if **Pointers** $=$ YES, **mx** is not referenced.

If **Pointers** $=$ NO, **mx** contains the vector Bx when **irevcm** returns the value $+1$.

On final exit: does not contain any useful data.

6: **nshift** – INTEGER

On intermediate exit: if the option **Supplied Shifts** is set and **irevcm** returns a value of 3, **nshift** returns the number of complex shifts required.

7: **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 defining the current state of the iterative process.

8: **icomm**$(:)$ – INTEGER array

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

Contains data defining the current state of the iterative process.

9: **ifail** – INTEGER

On final exit: **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 initial entry, the maximum number of iterations ≤ 0, the option **Iteration Limit** has been set to a non-positive value.

ifail $= 2$

The options **Generalized** and **Regular** are incompatible.

ifail $= 3$

The option **Initial Residual** was selected but the starting vector held in **resid** is zero.

ifail $= 4$ (*warning*)

The maximum number of iterations has been reached. Some Ritz values may have converged; a subsequent call to nag_sparseig_complex_proc (f12aq) will return the number of converged values and the converged values.

ifail $= 5$

No shifts could be applied during a cycle of the implicitly restarted Arnoldi iteration. One possibility is to increase the size of **ncv** relative to **nev** (see Section 5 in nag_sparseig_complex_init (f12an) for details of these arguments).

ifail $= 6$

Could not build an Arnoldi factorization. Consider changing **ncv** or **nev** in the initialization function (see Section 5 in nag_sparseig_complex_init (f12an) for details of these arguments).

ifail $= 7$

Unexpected error in internal call to compute eigenvalues and corresponding error bounds of the current upper Hessenberg matrix. Please contact NAG.

ifail $= 8$

Either the initialization function nag_sparseig_complex_init (f12an) has not been called prior to the first call of this function or a communication array has become corrupted.

ifail $= 9$

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 $Ax = \lambda x$ in shift-invert mode, where A is obtained from the standard central difference discretization of the convection-diffusion operator $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \rho\frac{\partial u}{\partial x}$ on the unit square, with zero Dirichlet boundary conditions. The shift used is a complex number.

9.1 Program Text

```
function f12ap_example

fprintf('f12ap example results\n\n');

n   = nag_int(100);
nx  = nag_int(10);
nev = nag_int(4);
ncv = nag_int(20);

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

sig = complex(0);

rho = 10;
h   = 1/double(n+1);
h2 = h*h;
s = rho/2;
s/h;
s1 = -1/h2 - s/h;
s2 =  2/h2;
s3 = -1/h2 + s/h;
cl = s1*ones(n-1,1);  cl = complex(cl);
cd = s2*ones(n,1);    cd = complex(cd);
cu = s3*ones(n-1,1);  cu = complex(cu);
% cd(1:n-1) = cd(1:n-1) - sig;

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

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

% 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)
    [x, info] = f07cs( ...
                       'N', cl, cd, cu, cu2, ipiv, x);
  elseif (irevcm == 4)
    [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
[nconv, d, z, v, comm, icomm, ifail] = ...
  f12aq( ...
         sig, resid, v, comm, icomm);

fprintf('\nThe %d Eigenvalues of smallest magnitude are:\n',nconv);
fprintf('%11.4f%+11.4fi\n',[real(d(1:nconv)) imag(d(1:nconv))]');
```

9.2 Program Results

```
     f12ap example results

Iteration  1, No. converged = 3, norm of estimates =   9.10e-18

The 4 Eigenvalues of smallest magnitude are:
    34.8720    -0.0000i
    64.4326    -0.0000i
   113.6685    +0.0000i
   182.5320    -0.0000i
```