NAG Toolbox

nag_sparseig_complex_option (f12ar)

1 Purpose

nag_sparseig_complex_option (f12ar) is an option setting 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), for which it may be used to supply individual optional parameters to nag_sparseig_complex_iter (f12ap) and nag_sparseig_complex_proc (f12aq). nag_sparseig_complex_option (f12ar) is also an option setting function in a suite of functions consisting of nag_sparseig_complex_init (f12an), nag_sparseig_complex_band_init (f12at) and nag_sparseig_complex_band_solve (f12au) for which it may be used to supply individual optional parameters to nag_sparseig_complex_band_solve (f12au).

The initialization function for the appropriate suite, nag_sparseig_complex_init (f12an) or nag_sparseig_complex_band_init (f12at), **must** have been called prior to calling nag_sparseig_complex_option (f12ar).

2 Syntax

```
[icomm, comm, ifail] = nag_sparseig_complex_option(str, icomm, comm)
[icomm, comm, ifail] = f12ar(str, icomm, comm)
```

3 Description

nag_sparseig_complex_option (f12ar) may be used to supply values for optional parameters to nag_sparseig_complex_iter (f12ap) and nag_sparseig_complex_proc (f12aq), or to nag_sparseig_complex_band_solve (f12au). It is only necessary to call nag_sparseig_complex_option (f12ar) for those arguments whose values are to be different from their default values. One call to nag_sparseig_complex_option (f12ar) sets one argument value.

Each optional parameter is defined by a single character string consisting of one or more items. The items associated with a given option must be separated by spaces, or equals signs [=]. Alphabetic characters may be upper or lower case. The string

```
'Pointers = Yes'
```

is an example of a string used to set an optional parameter. For each option the string contains one or more of the following items:

– a mandatory keyword;

– a phrase that qualifies the keyword;

– a number that specifies an integer or double value. Such numbers may be up to 16 contiguous characters in Fortran's I, F, E or D format.

nag_sparseig_complex_option (f12ar) does not have an equivalent function from the ARPACK package which passes options by directly setting values to scalar arguments or to specific elements of array arguments. nag_sparseig_complex_option (f12ar) is intended to make the passing of options more transparent and follows the same principle as the single option setting functions in Chapter E04 (see nag_opt_qpconvex2_sparse_option_string (e04ns) for an example).

The setup function nag_sparseig_complex_init (f12an) must be called prior to the first call to nag_sparseig_complex_option (f12ar) or nag_sparseig_complex_band_init (f12at), and all calls to nag_sparseig_complex_option (f12ar) must precede the first call to nag_sparseig_complex_iter (f12ap) or nag_sparseig_complex_band_solve (f12au).

A complete list of optional parameters, their abbreviations, synonyms and default values is given in Section 11.

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: **str** – CHARACTER(*)

A single valid option string (as described in Section 3 and Section 11).

2: **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).

3: **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).

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **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 options set.

2: **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 options set.

3: **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 $= 1$

The string passed in **str** contains an ambiguous keyword.

ifail $= 2$

The string passed in **str** contains a keyword that could not be recognized.

ifail $= 3$

The string passed in **str** contains a second keyword that could not be recognized.

ifail $= 4$

The initialization function nag_sparseig_complex_init (f12an) or nag_sparseig_complex_band_init (f12at) has not been called or a communication array has become corrupted.

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

Not applicable.

8 Further Comments

None.

9 Example

This example solves $Ax = \lambda Bx$ in shifted-inverse mode, where A and B are derived from the finite element discretization of the one-dimensional convection-diffusion operator $\frac{d^2u}{dx^2} + \rho\frac{du}{dx}$ on the interval $[0, 1]$, with zero Dirichlet boundary conditions.

9.1 Program Text

```
function f12ar_example

fprintf('f12ar 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));
```

```
sigma = complex(500);
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('\nIteration %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

```
     f12ar example results

Iteration 1, No. converged = 3, norm of estimates =      3.8e-17

The 4 generalised Ritz values closest to   500.0 are:
    509.94     -0.00i
    380.91     +0.00i
    659.16     +0.00i
    271.94     -0.00i
```

10 Optional Parameters

Several optional parameters for the computational suite functions nag_sparseig_complex_iter (f12ap) and nag_sparseig_complex_proc (f12aq), and for the banded driver nag_sparseig_complex_band_solve (f12au), define choices in the problem specification or the algorithm logic. In order to reduce the number of formal arguments of nag_sparseig_complex_iter (f12ap), nag_sparseig_complex_proc (f12aq) and nag_sparseig_complex_band_solve (f12au) these optional parameters have associated *default values* that are appropriate for most problems. Therefore, you need only specify those optional parameters whose values are to be different from their default values.

The remainder of this section can be skipped if you wish to use the default values for all optional parameters.

The following is a list of the optional parameters available. A full description of each optional parameter is provided in Section 11.1.

Advisory
Defaults
Exact Shifts
Generalized
Initial Residual
Iteration Limit
Largest Imaginary
Largest Magnitude
Largest Real
List
Monitoring
Nolist
Pointers
Print Level
Random Residual
Regular
Regular Inverse
Shifted Inverse
Smallest Imaginary
Smallest Magnitude
Smallest Real
Standard
Supplied Shifts
Tolerance
Vectors

Optional parameters may be specified by calling nag_sparseig_complex_option (f12ar) before a call to nag_sparseig_complex_iter (f12ap) or nag_sparseig_complex_band_init (f12at), but after a

corresponding call to nag_sparseig_complex_init (f12an) or nag_sparseig_complex_band_solve (f12au). One call is necessary for each optional parameter. Any optional parameters you do not specify are set to their default values. Optional parameters you do specify are unaltered by nag_sparseig_complex_iter (f12ap), nag_sparseig_complex_proc (f12aq) and nag_sparseig_complex_band_solve (f12au) (unless they define invalid values) and so remain in effect for subsequent calls unless you alter them.

10.1 Description of the Optional Parameters

For each option, we give a summary line, a description of the optional parameter and details of constraints.

The summary line contains:

the keywords, where the minimum abbreviation of each keyword is underlined;

a parameter value, where the letters a, i and r denote options that take character, integer and real values respectively;

the default value, where the symbol ϵ is a generic notation for ***machine precision*** (see nag_machine_precision (x02aj)).

Keywords and character values are case and white space insensitive.

Advisory i Default = the value returned by nag_file_set_unit_advisory (x04ab)

The destination for advisory messages.

Defaults

This special keyword may be used to reset all optional parameters to their default values.

Exact Shifts Default
Supplied Shifts

During the Arnoldi iterative process, shifts are applied as part of the implicit restarting scheme. The shift strategy used by default and selected by the optional parameter **Exact Shifts** is strongly recommended over the alternative **Supplied Shifts** and will always be used by nag_sparseig_complex_band_solve (f12au).

If **Exact Shifts** are used then these are computed internally by the algorithm in the implicit restarting scheme. This strategy is generally effective and cheaper to apply in terms of number of operations than using explicit shifts.

If **Supplied Shifts** are used then, during the Arnoldi iterative process, you must supply shifts through array arguments of nag_sparseig_complex_iter (f12ap) when nag_sparseig_complex_iter (f12ap) returns with **irevcm** = 3; the complex shifts are returned in **x** (or in **comm** when the option **Pointers** = YES is set). This option should only be used if you are an experienced user since this requires some algorithmic knowledge and because more operations are usually required than for the implicit shift scheme. Details on the use of explicit shifts and further references on shift strategies are available in Lehoucq *et al.* (1998).

Iteration Limit i Default = 300

The limit on the number of Arnoldi iterations that can be performed before nag_sparseig_complex_iter (f12ap) or nag_sparseig_complex_band_solve (f12au) exits. If not all requested eigenvalues have converged to within **Tolerance** and the number of Arnoldi iterations has reached this limit then nag_sparseig_complex_iter (f12ap) or nag_sparseig_complex_band_solve (f12au) exits with an error; nag_sparseig_complex_band_solve (f12au) returns the number of converged eigenvalues, the converged eigenvalues and, if requested, the corresponding eigenvectors, while nag_sparseig_complex_proc (f12aq) can be called subsequent to nag_sparseig_complex_iter (f12ap) to do the same.

Largest Magnitude Default
Largest Imaginary
Largest Real
Smallest Imaginary
Smallest Magnitude
Smallest Real

The Arnoldi iterative method converges on a number of eigenvalues with given properties. The default is for nag_sparseig_complex_iter (f12ap) or nag_sparseig_complex_band_solve (f12au) to compute the eigenvalues of largest magnitude using **Largest Magnitude**. Alternatively, eigenvalues may be chosen which have **Largest Real** part, **Largest Imaginary** part,**Smallest Magnitude**, **Smallest Real** part or **Smallest Imaginary** part.

Note that these options select the eigenvalue properties for eigenvalues of OP (and B for **Generalized** problems), the linear operator determined by the computational mode and problem type.

Nolist Default
List

Normally each optional parameter specification is not printed to the advisory channel as it is supplied. Optional parameter **List** may be used to enable printing and optional parameter **Nolist** may be used to suppress the printing.

Monitoring i Default $= -1$

If $i > 0$, monitoring information is output to channel number i during the solution of each problem; this may be the same as the **Advisory** channel number. The type of information produced is dependent on the value of **Print Level**, see the description of the optional parameter **Print Level** for details of the information produced. Please see nag_file_open (x04ac) to associate a file with a given channel number.

Pointers Default $=$ NO

During the iterative process and reverse communication calls to nag_sparseig_complex_iter (f12ap), required data can be communicated to and from nag_sparseig_complex_iter (f12ap) in one of two ways. When **Pointers** $=$ NO is selected (the default) then the array arguments **x** and **mx** are used to supply you with required data and used to return computed values back to nag_sparseig_complex_iter (f12ap). For example, when **irevcm** $= 1$, nag_sparseig_complex_iter (f12ap) returns the vector x in **x** and the matrix-vector product Bx in **mx** and expects the result or the linear operation OP(x) to be returned in **x**.

If **Pointers** $=$ YES is selected then the data is passed through sections of the array argument **comm**. The section corresponding to **x** when **Pointers** $=$ NO begins at a location given by the first element of **icomm**; similarly the section corresponding to **mx** begins at a location given by the second element of **icomm**. This option allows nag_sparseig_complex_iter (f12ap) to perform fewer copy operations on each intermediate exit and entry, but can also lead to less elegant code in the calling program.

This option has no affect on nag_sparseig_complex_band_solve (f12au) which sets **Pointers** $=$ YES internally.

Print Level i Default $= 0$

This controls the amount of printing produced by nag_sparseig_complex_option (f12ar) as follows.

$= 0$ No output except error messages.

> 0 The set of selected options.

$= 2$ Problem and timing statistics on final exit from f12ap or f12au.

≥ 5 A single line of summary output at each Arnoldi iteration.

≥ 10 If **Monitoring** > 0, **Monitoring** is set, then at each iteration, the length and additional steps of the current Arnoldi factorization and the number of converged Ritz values; during re-orthogonalization, the norm of initial/restarted starting vector.

≥ 20 Problem and timing statistics on final exit from f12ap. If **Monitoring** > 0, **Monitoring** is set, then at each iteration, the number of shifts being applied, the eigenvalues and estimates of the Hessenberg matrix H, the size of the Arnoldi basis, the wanted Ritz values and associated Ritz estimates and the shifts applied; vector norms prior to and following re-orthogonalization.

≥ 30 If **Monitoring** > 0, **Monitoring** is set, then on final iteration, the norm of the residual; when computing the Schur form, the eigenvalues and Ritz estimates both before and after sorting; for each iteration, the norm of residual for compressed factorization and the compressed upper Hessenberg matrix H; during re-orthogonalization, the initial/restarted starting vector; during the Arnoldi iteration loop, a restart is flagged and the number of the residual requiring iterative refinement; while applying shifts, the indices of the shifts being applied.

≥ 40 If **Monitoring** > 0, **Monitoring** is set, then during the Arnoldi iteration loop, the Arnoldi vector number and norm of the current residual; while applying shifts, key measures of progress and the order of H; while computing eigenvalues of H, the last rows of the Schur and eigenvector matrices; when computing implicit shifts, the eigenvalues and Ritz estimates of H.

≥ 50 If **Monitoring** is set, then during Arnoldi iteration loop: norms of key components and the active column of H, norms of residuals during iterative refinement, the final upper Hessenberg matrix H; while applying shifts: number of shifts, shift values, block indices, updated matrix H; while computing eigenvalues of H: the matrix H, the computed eigenvalues and Ritz estimates.

Random **Residual** Default
Initial **Residual**

To begin the Arnoldi iterative process, nag_sparseig_complex_iter (f12ap) and nag_sparseig_complex_band_solve (f12au) requires an initial residual vector. By default nag_sparseig_complex_iter (f12ap) and nag_sparseig_complex_band_solve (f12au) provides its own random initial residual vector; this option can also be set using optional parameter **Random Residual**. Alternatively, you can supply an initial residual vector (perhaps from a previous computation) to nag_sparseig_complex_iter (f12ap) and nag_sparseig_complex_band_solve (f12au) through the array argument **resid**; this option can be set using optional parameter **Initial Residual**.

Regular Default
Regular **Inverse**
Shifted **Inverse**

These options define the computational mode which in turn defines the form of operation $\mathrm{OP}(x)$ to be performed by nag_sparseig_complex_band_solve (f12au) or when nag_sparseig_complex_iter (f12ap) returns with **irevcm** $= -1$ or 1 and the matrix-vector product Bx when nag_sparseig_complex_iter (f12ap) returns with **irevcm** $= -2$.

Given a **Standard** eigenvalue problem in the form $Ax = \lambda x$ then the following modes are available with the appropriate operator $\mathrm{OP}(x)$.

Regular $\mathrm{OP} = A$
Shifted Inverse $\mathrm{OP} = (A - \sigma I)^{-1}$

Given a **Generalized** eigenvalue problem in the form $Ax = \lambda Bx$ then the following modes are available with the appropriate operator $\mathrm{OP}(x)$.

Regular Inverse $\mathrm{OP} = B^{-1}A$
Shifted Inverse $\mathrm{OP} = (A - \sigma B)^{-1}B$

Standard Default
Generalized

The problem to be solved is either a standard eigenvalue problem, $Ax = \lambda x$, or a generalized eigenvalue problem, $Ax = \lambda Bx$. The optional parameter **Standard** should be used when a standard

eigenvalue problem is being solved and the optional parameter **Generalized** should be used when a generalized eigenvalue problem is being solved.

Tolerance r Default $= \epsilon$

An approximate eigenvalue has deemed to have converged when the corresponding Ritz estimate is within **Tolerance** relative to the magnitude of the eigenvalue.

Vectors Default = RITZ

The function nag_sparseig_complex_proc (f12aq) or nag_sparseig_complex_band_solve (f12au) can optionally compute the Schur vectors and/or the eigenvectors corresponding to the converged eigenvalues. To turn off computation of any vectors the option **Vectors** = NONE should be set. To compute only the Schur vectors (at very little extra cost), the option **Vectors** = SCHUR should be set and these will be returned in the array argument **v** of nag_sparseig_complex_proc (f12aq) or nag_sparseig_complex_band_solve (f12au). To compute the eigenvectors (Ritz vectors) corresponding to the eigenvalue estimates, the option **Vectors** = RITZ should be set and these will be returned in the array argument **z** of nag_sparseig_complex_proc (f12aq) or nag_sparseig_complex_band_solve (f12au), if **z** is set equal to **v** (as in Section 10) then the Schur vectors in **v** are overwritten by the eigenvectors computed by nag_sparseig_complex_proc (f12aq) or nag_sparseig_complex_band_solve (f12au).