

NAG Toolbox

nag_sparseig_real_symm_option (f12fd)

1 Purpose

nag_sparseig_real_symm_option (f12fd) is an option setting function in a suite of functions consisting of nag_sparseig_real_symm_init (f12fa), nag_sparseig_real_symm_iter (f12fb), nag_sparseig_real_symm_proc (f12fc), nag_sparseig_real_symm_option (f12fd) and nag_sparseig_real_symm_monit (f12fe), and may be used to supply individual optional parameters to nag_sparseig_real_symm_iter (f12fb) and nag_sparseig_real_symm_proc (f12fc). The initialization function nag_sparseig_real_symm_init (f12fa) **must** have been called prior to calling nag_sparseig_real_symm_option (f12fd).

2 Syntax

```
[icomm, comm, ifail] = nag_sparseig_real_symm_option(str, icomm, comm)
[icomm, comm, ifail] = f12fd(str, icomm, comm)
```

3 Description

nag_sparseig_real_symm_option (f12fd) may be used to supply values for optional parameters to nag_sparseig_real_symm_iter (f12fb) and nag_sparseig_real_symm_proc (f12fc). It is only necessary to call nag_sparseig_real_symm_option (f12fd) for those arguments whose values are to be different from their default values. One call to nag_sparseig_real_symm_option (f12fd) 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_real_symm_option (f12fd) 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_real_symm_option (f12fd) is intended to make the passing of options more transparent and follows the same principle as the single option setting functions in Chapter E04.

The setup function nag_sparseig_real_symm_init (f12fa) must be called prior to the first call to nag_sparseig_real_symm_option (f12fd) and all calls to nag_sparseig_real_symm_option (f12fd) must precede the first call to nag_sparseig_real_symm_iter (f12fb), the reverse communication iterative solver.

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, Philadelphia

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_real_symm_init` (f12fa))

On initial entry: must remain unchanged following a call to the setup function `nag_sparseig_real_symm_init` (f12fa).

3: **comm**(:) – REAL (KIND=nag_wp) array

The dimension of the array **comm** must be at least 60

On initial entry: must remain unchanged following a call to the setup function `nag_sparseig_real_symm_init` (f12fa).

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_real_symm_init` (f12fa))

Contains data on the current options set.

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

The dimension of the array **comm** will be 60

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_real_symm_init` (f12fa) 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 obtained from the standard central difference discretization of the one-dimensional Laplacian operator $\frac{\partial^2 u}{\partial x^2}$ on $[0, 1]$, with zero Dirichlet boundary conditions. Data is passed to and from the reverse communication function `nag_sparseig_real_symm_iter` (f12fb) using pointers to the communication array.

9.1 Program Text

```
function f12fd_example

fprintf('f12fd example results\n\n');

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

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

```

sigma = 0;

% Setup and factorize A - sigma*B
h = 1/double(n+1);
ad(1:n) = 2/h - sigma*4*h/6;
adl(1:n) = -1/h - sigma*h/6;
adu(1:n) = adl(1:n);

[adl, ad, adu, adu2, ipiv, info] = f07cd( ...
                                     adl, ad, adu);

% Initialisation Step
[icomm, comm, ifail] = f12fa( ...
                             n, nev, ncv);

% Set Optional Parameters
[icomm, comm, ifail] = f12fd( ...
                             'Generalized', icomm, comm);
[icomm, comm, ifail] = f12fd( ...
                             'Shifted Inverse', icomm, comm);

% Solve
while (irevcm ~= 5)
    [irevcm, resid, v, x, mx, nshift, comm, icomm, ifail] = ...
        f12fb( ...
              irevcm, resid, v, x, mx, comm, icomm);
    if (irevcm == -1)
        % Solve (A-sigma*B)y = Bx
        mx = f12fd_Bx(n, x);
        [x, info] = f07ce( ...
                        'N', adl, ad, adu, adu2, ipiv, mx);
    elseif (irevcm == 1)
        % Solve (A-sigma*B)y = Bx, Bx in mx
        [x, info] = f07ce( ...
                        'N', adl, ad, adu, adu2, ipiv, mx);
    elseif (irevcm == 2)
        % y = Bx
        mx = f12fd_Bx(n, x);
    elseif (irevcm == 4 && imon == 1)
        [niter, nconv, ritz, rzest] = ...
            f12fe(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] = ...
    f12fc( ...
          sigma, resid, v, comm, icomm);

fprintf('Smallest %d Eigenvalues are:\n', nconv);
disp(d(1:nconv));

function [y] = f12fd_Bx(n,x)

    y = zeros(n,1);
    h = 1/(6*double(n+1));

    y(1) = 4*x(1) + x(2);
    for j=2:n-1
        y(j) = x(j-1) + 4*x(j) + x(j+1);
    end
    y(n) = x(n-1) + 4*x(n);
    y = y*h;

```

9.2 Program Results

```
f12fd example results
```

Smallest 4 Eigenvalues are:

```
  9.8704
 39.4912
 88.8909
158.1175
```

10 Optional Parameters

Several optional parameters for the computational functions `nag_sparseig_real_symm_iter` (f12fb) and `nag_sparseig_real_symm_proc` (f12fc) define choices in the problem specification or the algorithm logic. In order to reduce the number of formal arguments of `nag_sparseig_real_symm_iter` (f12fb) and `nag_sparseig_real_symm_proc` (f12fc) 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

Both Ends

Buckling

Cayley

Defaults

Exact Shifts

Generalized

Initial Residual

Iteration Limit

Largest Algebraic

Largest Magnitude

List

Monitoring

Nolist

Pointers

Print Level

Random Residual

Regular

Regular Inverse

Shifted Inverse

Smallest Algebraic

Smallest Magnitude

Standard

Supplied Shifts

Tolerance

Vectors

Optional parameters may be specified by calling `nag_sparseig_real_symm_option` (f12fd) before a call to `nag_sparseig_real_symm_iter` (f12fb), but after a call to `nag_sparseig_real_symm_init` (f12fa). One call is necessary for each optional parameter.

All optional parameters you do not specify are set to their default values. Optional parameters you do specify are unaltered by `nag_sparseig_real_symm_iter` (f12fb) and `nag_sparseig_real_symm_proc` (f12fc) (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 Lanczos iterative process, shifts are applied internally as part of the implicit restarting scheme. The shift strategy used by default and selected by the **Exact Shifts** is strongly recommended over the alternative **Supplied Shifts** (see Lehoucq *et al.* (1998) for details of shift strategies).

If **Exact Shifts** are used then these are computed internally by the algorithm in the implicit restarting scheme.

If **Supplied Shifts** are used then, during the Lanczos iterative process, you must supply shifts through array arguments of `nag_sparseig_real_symm_iter` (f12fb) when `nag_sparseig_real_symm_iter` (f12fb) returns with **irevcn** = 3; the real and imaginary parts of the shifts are returned in **x** and **mx** respectively (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 Lanczos iterations that can be performed before `nag_sparseig_real_symm_iter` (f12fb) exits. If not all requested eigenvalues have converged to within **Tolerance** and the number of Lanczos iterations has reached this limit then `nag_sparseig_real_symm_iter` (f12fb) exits with an error; `nag_sparseig_real_symm_proc` (f12fc) can still be called subsequently to return the number of converged eigenvalues, the converged eigenvalues and, if requested, the corresponding eigenvectors.

Largest Magnitude

Default

Both Ends**Largest Algebraic****Smallest Algebraic****Smallest Magnitude**

The Lanczos iterative method converges on a number of eigenvalues with given properties. The default is for `nag_sparseig_real_symm_iter` (f12fb) to compute the eigenvalues of largest magnitude using **Largest Magnitude**. Alternatively, eigenvalues may be chosen which have **Largest Algebraic** part, **Smallest Magnitude**, or **Smallest Algebraic** part; or eigenvalues which are from **Both Ends** of the algebraic spectrum.

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_real_symm_iter` (f12fb), required data can be communicated to and from `nag_sparseig_real_symm_iter` (f12fb) in one of two ways. When **Pointers** = NO is selected (the default) then the array arguments \mathbf{x} and \mathbf{mx} are used to supply you with required data and used to return computed values back to `nag_sparseig_real_symm_iter` (f12fb). For example, when `irevcn` = 1, `nag_sparseig_real_symm_iter` (f12fb) returns the vector x in \mathbf{x} and the matrix-vector product Bx in \mathbf{mx} and expects the result of the linear operation $OP(x)$ to be returned in \mathbf{x} .

If **Pointers** = YES is selected then the data is passed through sections of the array argument **comm**. The section corresponding to \mathbf{x} when **Pointers** = NO begins at a location given by the first element of **icomm**; similarly the section corresponding to \mathbf{mx} begins at a location given by the second element of **icomm**. This option allows `nag_sparseig_real_symm_iter` (f12fb) to perform fewer copy operations on each intermediate exit and entry, but can also lead to less elegant code in the calling program.

Print Level i

Default = 0

This controls the amount of printing produced by `nag_sparseig_real_symm_option` (f12fd) as follows.

- = 0 No output except error messages. If you want to suppress all output, set **Print Level** = 0.
- > 0 The set of selected options.
- = 2 Problem and timing statistics on final exit from f12fb.
- ≥ 5 A single line of summary output at each Lanczos iteration.
- ≥ 10 If **Monitoring** > 0, **Monitoring** is set, then at each iteration, the length and additional steps of the current Lanczos factorization and the number of converged Ritz values; during re-orthogonalization, the norm of initial/restarted starting vector; on a final Lanczos iteration, the number of update iterations taken, the number of converged eigenvalues, the converged eigenvalues and their Ritz estimates.

- ≥ 20 Problem and timing statistics on final exit from f12fb. If **Monitoring** > 0 , **Monitoring** is set, then at each iteration, the number of shifts being applied, the eigenvalues and estimates of the symmetric tridiagonal matrix H , the size of the Lanczos 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 symmetric tridiagonal matrix H ; during re-orthogonalization, the initial/restarted starting vector; during the Lanczos iteration loop, a restart is flagged and the number of the residual requiring iterative refinement; while applying shifts, some indices.
- ≥ 40 If **Monitoring** > 0 , **Monitoring** is set, then during the Lanczos iteration loop, the Lanczos 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 Lanczos iteration loop: norms of key components and the active column of H , norms of residuals during iterative refinement, the final symmetric tridiagonal matrix H ; while applying shifts: number of shifts, shift values, block indices, updated tridiagonal matrix H ; while computing eigenvalues of H : the diagonals of H , the computed eigenvalues and Ritz estimates.

Note that setting **Print Level** ≥ 30 can result in very lengthy **Monitoring** output.

Note that setting **Print Level** ≥ 30 can result in very lengthy **Monitoring** output.

Random Residual Initial Residual

Default

To begin the Lanczos iterative process, nag_sparseig_real_symm_iter (f12fb) requires an initial residual vector. By default nag_sparseig_real_symm_iter (f12fb) 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_real_symm_iter (f12fb) through the array argument **resid**; this option can be set using optional parameter **Initial Residual**.

Regular Regular Inverse Shifted Inverse Buckling Cayley

Default

These options define the computational mode which in turn defines the form of operation $OP(x)$ to be performed when nag_sparseig_real_symm_iter (f12fb) returns with **irevcn** = -1 or 1 and the matrix-vector product Bx when nag_sparseig_real_symm_iter (f12fb) returns with **irevcn** = 2.

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

Regular $OP = A$
Shifted Inverse $OP = (A - \sigma I)^{-1}$ where σ is real

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

Regular Inverse $OP = B^{-1}A$
Shifted Inverse $OP = (A - \sigma B)^{-1}B$, where σ is real

Buckling $OP = (B - \sigma A)^{-1}A$, where σ is real
Cayley $OP = (A - \sigma B)^{-1}(A + \sigma B)$, where σ is real

Standard
Generalized

Default

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 = ϵ

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_real_symm_proc` (f12fc) 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_real_symm_proc` (f12fc). 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_real_symm_proc` (f12fc), 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_real_symm_proc` (f12fc).
