NAG Toolbox

nag_sparseig_real_band_init (f12af)

1 Purpose

nag_sparseig_real_band_init (f12af) is a setup function for nag_sparseig_real_band_solve (f12ag) which may be used for finding some eigenvalues (and optionally the corresponding eigenvectors) of a standard or generalized eigenvalue problem defined by real, banded, nonsymmetric matrices. The banded matrix must be stored using the LAPACK column ordered storage format for real banded nonsymmetric matrices (see Section 3.2.4 in the F07 Chapter Introduction).

2 Syntax

```
[icomm, comm, ifail] = nag_sparseig_real_band_init(n, nev, ncv)
[icomm, comm, ifail] = f12af(n, nev, ncv)
```

3 Description

The pair of functions nag_sparseig_real_band_init (f12af) and nag_sparseig_real_band_solve (f12ag) together with the option setting function nag_sparseig_real_option (f12ad) are 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 banded real and nonsymmetric.

nag_sparseig_real_band_init (f12af) is a setup function which must be called before the option setting function nag_sparseig_real_option (f12ad) and the solver function nag_sparseig_real_band_solve (f12ag). Internally, nag_sparseig_real_band_solve (f12ag) makes calls to nag_sparseig_real_iter (f12ab) and nag_sparseig_real_proc (f12ac); the function documents for nag_sparseig_real_iter (f12ab) and nag_sparseig_real_proc (f12ac) should be consulted for details of the algorithm used.

This setup function initializes the communication arrays, sets (to their default values) all options that can be set by you via the option setting function nag_sparseig_real_option (f12ad), and checks that the lengths of the communication arrays as passed by you are of sufficient length. For details of the options available and how to set them, see Section 11.1 in nag_sparseig_real_option (f12ad).

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: **n** – INTEGER

The order of the matrix A (and the order of the matrix B for the generalized problem) that defines the eigenvalue problem.

Constraint: $\mathbf{n} > 0$.

2: **nev** – INTEGER

The number of eigenvalues to be computed.

Constraint: $0 < \mathbf{nev} < \mathbf{n} - 1$.

3: **ncv** – INTEGER

The number of Lanczos basis vectors to use during the computation.

At present there is no *a priori* analysis to guide the selection of **ncv** relative to **nev**. However, it is recommended that $\mathbf{ncv} \geq 2 \times \mathbf{nev} + 1$. If many problems of the same type are to be solved, you should experiment with increasing **ncv** while keeping **nev** fixed for a given test problem. This will usually decrease the required number of matrix-vector operations but it also increases the work and storage required to maintain the orthogonal basis vectors. The optimal 'cross-over' with respect to CPU time is problem dependent and must be determined empirically.

Constraint: $\mathbf{nev} + 1 < \mathbf{ncv} \leq \mathbf{n}$.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **icomm**(**max**(**1**, $licomm$)) – INTEGER array

Contains data to be communicated to nag_sparseig_real_band_solve (f12ag).

2: **comm**(**max**(**1**, $lcomm$)) – REAL (KIND=nag_wp) array

Contains data to be communicated to nag_sparseig_real_band_solve (f12ag).

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 $= 1$

On entry, $\mathbf{n} \leq 0$.

ifail $= 2$

On entry, $\mathbf{nev} \leq 0$.

ifail $= 3$

On entry, $\mathbf{ncv} < \mathbf{nev} + 2$ or $\mathbf{ncv} > \mathbf{n}$.

ifail $= 4$

On entry, $licomm < 140$ and $licomm \neq -1$.

ifail $= 5$

On entry, $lcomm < 60$ and $lcomm \neq -1$.

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

The use of nag_sparseig_real_band_init (f12af) is illustrated in Section 10 in nag_sparseig_real_band_solve (f12ag).

9.1 Program Text

```
function f12af_example

fprintf('f12af example results\n\n');

nx  = nag_int(10);
n   = nx*nx;
nev = nag_int(4);
ncv = nx;
kl  = nx;
ku  = nx;

% Construct ab and mb
ab = zeros(2*kl+ku+1,n);
mb = zeros(2*kl+ku+1,n);

% Main diagonal of A.
idiag = kl + ku + 1;
for j=1:n
  ab(idiag,j) = 4;
  mb(idiag,j) = 4;
end

% First subdiagonal and superdiagonal of A.
isup = kl + ku;
isub = kl + ku + 2;
rho = 100;
h = 1/double(nx+1);
for i=1:nx
  lo = (i-1)*nx;
  for j=lo+1:lo+nx-1
    ab(isub,j+1) = -1 + 0.5*h*rho;
```

```
    ab(isup,j)   = -1 - 0.5*h*rho;
  end
end
for j = 1:n - 1
  mb(isub,j+1) = 1;
  mb(isup,j) = 1;
end

% kl-th subdiagonal and ku-th super-diagonal.
isup = kl + 1;
isub = 2*kl + ku + 1;
for i = 1:nx - 1
  lo = (i-1)*nx;
  for j = lo + 1:lo + nx
    ab(isup,nx+j) = -1;
    ab(isub,j)    = -1;
  end
end

sigmar = 0.4;
sigmai = 0.6;
resid = zeros(100,1);

[icomm, comm, ifail] = f12af( ...
                              n, nev, ncv);
[icomm, comm, ifail] = f12ad( ...
                              'Shifted imaginary', icomm, comm);
[icomm, comm, ifail] = f12ad( ...
                              'Generalized', icomm, comm);
[nconv, dr, di, z, resid, v, comm, icomm, ifail] = ...
  f12ag( ...
         kl, ku, ab, mb, sigmar, sigmai, resid, comm, icomm);

fprintf('The %4d Ritz values closest to %8.2f %+8.2fi are:\n\n', ...
        nconv, sigmar, sigmai);
fprintf('%9.4f %+9.4fi\n', [dr(1:nconv) di(1:nconv)]');
```

9.2 Program Results

```
     f12af example results

The    4 Ritz values closest to     0.40    +0.60i are:

   0.3610   +0.7223i
   0.3610   -0.7223i
   0.4598   -0.7199i
   0.4598   +0.7199i
```