NAG Toolbox for MATLAB

f12fa

1 Purpose

f12fa is a setup function in a suite of functions consisting of f12fa, f12fb, f12fc, f12fd and f12fe. It is used to find some of the eigenvalues (and optionally the corresponding eigenvectors) of a standard or generalized eigenvalue problem defined by real symmetric matrices.

The suite of functions is suitable for the solution of large sparse, standard or generalized, symmetric eigenproblems where only a few eigenvalues from a selected range of the spectrum are required.

2 Syntax

```
[icomm, comm, ifail] = f12fa(n, nev, ncv)
```

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, real and symmetric. The suite can also be used to find selected eigenvalues/eigenvectors of smaller scale dense, real and symmetric problems.

f12fa is a setup function which must be called before f12fb, the reverse communication iterative solver, and before f12fd, the options setting function. f12fc, is a post-processing function that must be called following a successful final exit from f12fb, while f12fe can be used to return additional monitoring information during the computation.

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 f12fd, 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 10.2 of the document for f12fd.

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 – int32 scalar**

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 – int32 scalar**

The number of eigenvalues to be computed.

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

3: **ncv – int32 scalar**

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 varying **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: $\mathbf{nev} < \mathbf{ncv} \leq \mathbf{n}$.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

licomm, lcomm

5.4 Output Parameters

1: **icomm**$(*)$ **– int32 array**

Note: the dimension of the array **icomm** must be at least $\max(1, \mathbf{licomm})$.

Contains data to be communicated to the other functions in the suite.

2: **comm**$(*)$ **– double array**

Note: the dimension of the array **comm** must be at least $\max(1, \mathbf{lcomm})$.

Contains data to be communicated to the other functions in the suite.

3: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

$\mathbf{ifail} = 1$

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

$\mathbf{ifail} = 2$

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

$\mathbf{ifail} = 3$

On entry, $\mathbf{ncv} \leq \mathbf{nev}$ or $\mathbf{ncv} > \mathbf{n}$.

$\mathbf{ifail} = 4$

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

ifail = 5

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

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

```
n = int32(100);
nx = int32(10);
nev = int32(4);
ncv = int32(10);

irevcm = int32(0);
resid = zeros(100,1);
v = zeros(100,20);
x = zeros(100,1);
mx = zeros(100,1);

sigma = 0;


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

% Set Optional Parameters
[icomm, comm, ifail] = f12fd('SMALLEST MAGNITUDE', 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 || irevcm == -1)
    x = f12f_av(nx, x);
  elseif (irevcm == 4)
    [niter, nconv, ritz, rzest] = f12fe(icomm, comm);
      fprintf('Iteration %d, No. converged = %d, norm of estimates =
%16.8g\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)
```

```
Iteration 1, No. converged = 0, norm of estimates =        81.010211
Iteration 2, No. converged = 0, norm of estimates =        45.634095
Iteration 3, No. converged = 0, norm of estimates =        42.747772
Iteration 4, No. converged = 0, norm of estimates =        8.6106757
Iteration 5, No. converged = 0, norm of estimates =       0.71330195
Iteration 6, No. converged = 0, norm of estimates =       0.15050738
Iteration 7, No. converged = 0, norm of estimates =      0.015776765
Iteration 8, No. converged = 0, norm of estimates =     0.0038996544
Iteration 9, No. converged = 0, norm of estimates =     0.0004324447
Iteration 10, No. converged = 0, norm of estimates =    0.00011026365
Iteration 11, No. converged = 0, norm of estimates =    1.2358564e-05
Iteration 12, No. converged = 0, norm of estimates =    3.1712516e-06
Iteration 13, No. converged = 1, norm of estimates =    3.5633917e-07
Iteration 14, No. converged = 1, norm of estimates =    4.1370551e-08
```

```
Iteration 15, No. converged = 2, norm of estimates =    5.3820859e-09
Iteration 16, No. converged = 1, norm of estimates =     7.557502e-10
Iteration 17, No. converged = 1, norm of estimates =    4.0268792e-11
Iteration 18, No. converged = 2, norm of estimates =    1.4048495e-11
Iteration 19, No. converged = 2, norm of estimates =     1.774664e-10
Iteration 20, No. converged = 2, norm of estimates =    1.1754284e-09
Iteration 21, No. converged = 2, norm of estimates =     7.782271e-09
Iteration 22, No. converged = 2, norm of estimates =    5.2551985e-10
Iteration 23, No. converged = 2, norm of estimates =    1.8326785e-10
Iteration 24, No. converged = 2, norm of estimates =    7.8753868e-11
Iteration 25, No. converged = 2, norm of estimates =    2.2420618e-11
Iteration 26, No. converged = 2, norm of estimates =    8.5727233e-13
Iteration 27, No. converged = 2, norm of estimates =    3.4872081e-13
Iteration 28, No. converged = 2, norm of estimates =    2.6274361e-14
Iteration 29, No. converged = 2, norm of estimates =    2.1845559e-14
Iteration 30, No. converged = 3, norm of estimates =     8.670653e-16
Iteration 31, No. converged = 3, norm of estimates =    2.5548816e-16
Iteration 32, No. converged = 3, norm of estimates =    4.9338532e-18
nconv =
           4
d =
   19.6054
   48.2193
   48.2193
   76.8333
         0
         0
         0
         0
         0
         0
z =
     array elided
v =
     array elided
comm =
     array elided
icomm =
     array elided
ifail =
           0
```