NAG Toolbox for MATLAB

f12fb

1 Purpose

f12fb is an iterative solver 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.

2 Syntax

```
[irevcm, resid, v, x, mx, nshift, comm, icomm, ifail] = f12fb(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, real and symmetric. The suite can also be used to find selected eigenvalues/eigenvectors of smaller scale dense, real and symmetric problems.

f12fb is a **reverse communication** function, based on the ARPACK routine **dsaupd**, using the Implicitly Restarted Arnoldi iteration method, which for symmetric problems reduces to a variant of the Lanczos 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 symmetric matrices is provided in Lehoucq and Scott 1996. This suite of functions offers the same functionality as the ARPACK software for real symmetric problems, but the interface design is quite different in order to make the option setting clearer to you and to simplify the interface of f12fb.

The setup function f12fa must be called before f12fb, the reverse communication iterative solver. Options may be set for f12fb by prior calls to the option setting function f12fd and a post-processing function f12fc must be called following a successful final exit from f12fb. f12fe, may be called following certain flagged, intermediate exits from f12fb to provide additional monitoring information about the computation.

f12fb uses **reverse communication**, i.e., it returns repeatedly to the calling program with the parameter **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, Buckling or Cayley) and other options can all be set using the option setting function 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

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 **parameter IREVCM**. Between intermediate exits and re-entries, **all parameters other than x, mx and comm must remain unchanged**.

5.1 Compulsory Input Parameters

1: **irevcm – int32 scalar**

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**$(*)$ **– double array**

Note: the dimension of the array **resid** must be at least **n** (see f12fa).

On initial entry: need not be set unless the option **Initial Residual** has been set in a prior call to f12fd 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**,$*$) **– double array**

The first dimension of the array **v** must be at least **n**

The second dimension of the array 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**$(*)$ **– double array**

Note: the dimension of the array **x** must be at least **n** (see f12fa).

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 real parts of the computed shifts when **irevcm** returns the value 3.

5: **mx**$(*)$ **– double array**

Note: the dimension of the array **mx** must be at least **n** (see f12fa).

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. It must return the imaginary parts of the computed shifts when **irevcm** returns the value 3.

6: **comm**$(*)$ **– double array**

Note: the dimension of the array **comm** must be at least $\max(1, \mathbf{lcomm})$ (see f12fa).

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

7: **icomm**$(*)$ **– int32 array**

Note: the dimension of the array **icomm** must be at least $\max(1, \mathbf{licomm})$ (see f12fa).

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

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

ldv

5.4 Output Parameters

1: **irevcm – int32 scalar**

On intermediate exit: has the following meanings.

-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 f12fd. 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.

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 $y = Bx$ (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.

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.

3 Compute the **nshift** real and imaginary parts of the shifts where the real parts are to be returned in the first **nshift** locations of the array **x** and the imaginary parts are to be returned in the first **nshift** locations of the array **mx**. Only complex conjugate pairs of shifts may be applied and the pairs must be placed in consecutive locations. This value of **irevcm** will only arise if the optional parameter **Supplied Shifts** is set in a prior call to f12fd which is intended for experienced users only; the default and recommended option is to use exact shifts (see Lehoucq *et al.* 1998 for details and guidance on the choice of shift strategies).

4 Monitoring step: a call to f12fe can now be made to return the number of Arnoldi iterations, the number of converged Ritz values, their real and imaginary parts, and the corresponding Ritz estimates.

On final exit: **irevcm** = 5: f12fb 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 f12fc must be called to return the requested eigenvalues and eigenvectors (and/or Schur vectors).

2: **resid**$(*)$ **– double array**

Note: the dimension of the array **resid** must be at least **n** (see f12fa).

On intermediate exit: contains the current residual vector.

On final exit: contains the final residual vector.

3: **v(ldv,∗) – double array**

The first dimension of the array **v** must be at least **n**

The second dimension of the array must be at least $\max(1, \mathbf{ncv})$

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

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

4: **x(∗) – double array**

Note: the dimension of the array **x** must be at least **n** (see f12fa).

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(∗) – double array**

Note: the dimension of the array **mx** must be at least **n** (see f12fa).

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 – int32 scalar**

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(∗) – double array**

Note: the dimension of the array **comm** must be at least $\max(1, \mathbf{lcomm})$ (see f12fa).

Contains data defining the current state of the iterative process.

8: **icomm(∗) – int32 array**

Note: the dimension of the array **icomm** must be at least $\max(1, \mathbf{licomm})$ (see f12fa).

Contains data defining the current state of the iterative process.

9: **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:

ifail = 1

On initial entry, the maximum number of iterations ≤ 0, the option **Iteration Limit** has been set to a nonpositive value.

ifail = 2

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

ifail = 3

Eigenvalues from both ends of the spectrum were requested, but the number of eigenvalues requested is one.

ifail $= 4$

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

ifail $= 5$

The maximum number of iterations has been reached. Some Ritz values may have converged; a subsequent call to f12fc will return the number of converged values and the converged values.

ifail $= 6$

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 of the document for f12fa for details of these parameters).

ifail $= 7$

Could not build a Lanczos factorization. Consider changing **ncv** or **nev** in the initialization function (see Section 5 of the document for f12fa for details of these parameters).

ifail $= 8$

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

ifail $= 9$

An unexpected error has occurred. Please contact NAG.

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 x02aj.

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
```