NAG Toolbox for MATLAB

f12ad

1 Purpose

f12ad is an option setting function in a suite of functions consisting of f12aa, f12ab, f12ac, f12ad and f12ae, and may be used to supply individual optional parameters to f12ab and f12ac. The initialization function f12aa **must** have been called prior to calling f12ad.

2 Syntax

```
[icomm, comm, ifail] = f12ad(str, icomm, comm)
```

3 Description

f12ad may be used to supply values for optional parameters to f12ab and f12ac. It is only necessary to call f12ad for those parameters whose values are to be different from their default values. One call to f12ad sets one parameter 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

```
'Vectors = None'
```

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.

f12ad does not have an equivalent function from the ARPACK package which passes options by directly setting values to scalar parameters or to specific elements of array arguments. f12ad 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 f12aa must be called prior to the first call to f12ad and all calls to f12ad must precede the first call to f12ab, the reverse communication iterative solver.

A complete list of optional parameters, their abbreviations, synonyms and default values is given in Section 10.

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 – string**

A single valid option string (as described in Section 3 and Section 10).

2: **icomm**$(*)$ **– int32 array**

Note: the dimension of the array **icomm** must be at least $\max(1, \mathbf{licomm})$ (see f12aa).

On initial entry: must remain unchanged following a call to the setup function f12aa.

3: **comm**$(*)$ **– double array**

Note: the dimension of the array **comm** must be at least 60.

On initial entry: must remain unchanged following a call to the setup function f12aa.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **icomm**$(*)$ **– int32 array**

Note: the dimension of the array **icomm** must be at least $\max(1, \mathbf{licomm})$ (see f12aa).

Contains data on the current options set.

2: **comm**$(*)$ **– double array**

Note: the dimension of the array **comm** must be at least 60.

Contains data on the current options set.

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:

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 f12aa has not been called or a communication array has become corrupted.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

```
n = int32(100);
nev = int32(4);
ncv = int32(20);

h = 1/(double(n)+1);
rho = 10;
md = repmat(4*h, double(n), 1);
me = repmat(h, double(n-1), 1);

irevcm = int32(0);
resid = zeros(n,1);
v = zeros(n, ncv);
x = zeros(n, 1);
mx = zeros(n);

dd = 2/h;
dl = -1/h - rho/2;
du = -1/h + rho/2;
y = zeros(n,1);

[icomm, comm, ifail] = f12aa(n, nev, ncv);
[icomm, comm, ifail] = f12ad('REGULAR INVERSE', icomm, comm);
[icomm, comm, ifail] = f12ad('GENERALIZED', icomm, comm);

% Construct m and factorise
[md, me, info] = f07jd(md, me);

while (irevcm ~= 5)
  [irevcm, resid, v, x, mx, nshift, comm, icomm, ifail] = ...
    f12ab(irevcm, resid, v, x, mx, comm, icomm);
  if (irevcm == -1 || irevcm == 1)
    y(1) = dd*x(1) + du*x(2);
    for i = 2:n-1
      y(i) = dl*x(i-1) + dd*x(i) + du*x(i+1);
    end
    y(n) = dl*x(n-1) + dd*x(n);
    [x, info] = f07je(md, me, y);
  elseif (irevcm == 2)
    y(1) = 4*x(1) + x(2);
    for i=2:n-1
      y(i) = x(i-1) + 4*x(i) + x(i+1);
    end
    y(n) = x(n-1) + 4*x(n);
    x = h*y;
  elseif (irevcm == 4)
    [niter, nconv, ritzr, ritzi, rzest] = f12ae(icomm, comm);
    if (niter == 1)
      fprintf('\n');
    end
       fprintf('Iteration %2d No. converged = %d Norm of estimates =
%16.8e\n', niter, nconv, norm(rzest));
  end
end
[nconv, dr, di, z, v, comm, icomm, ifail] = f12ac(0, 0, resid, v, comm,
icomm)
```

```
Iteration  1 No. converged = 0 Norm of estimates =   5.56325675e+03
Iteration  2 No. converged = 0 Norm of estimates =   5.44836588e+03
Iteration  3 No. converged = 0 Norm of estimates =   5.30320774e+03
Iteration  4 No. converged = 0 Norm of estimates =   6.24234186e+03
Iteration  5 No. converged = 0 Norm of estimates =   7.15674705e+03
Iteration  6 No. converged = 0 Norm of estimates =   5.45460864e+03
Iteration  7 No. converged = 0 Norm of estimates =   6.43147571e+03
Iteration  8 No. converged = 0 Norm of estimates =   5.11241161e+03
Iteration  9 No. converged = 0 Norm of estimates =   7.19327824e+03
Iteration 10 No. converged = 1 Norm of estimates =   5.77945489e+03
Iteration 11 No. converged = 2 Norm of estimates =   4.73125738e+03
Iteration 12 No. converged = 3 Norm of estimates =   5.00078500e+03
nconv =
           4
dr =
   1.0e+04 *
    2.0383
    2.0339
    2.0265
    2.0163
di =
     0
     0
     0
     0
z =
     array elided
v =
     array elided
comm =
     array elided
icomm =
     array elided
ifail =
           0
```

10 Optional Parameters

Several optional parameters for the computational functions f12ab and f12ac define choices in the problem specification or the algorithm logic. In order to reduce the number of formal parameters of f12ab and f12ac 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. A complete list of optional parameters and their default values is given in Section 10.1.

Optional parameters may be specified by calling f12ad prior to a call to f12ab, but after a call to f12aa. One call is necessary for each optional parameter.

All optional parameters not specified by you are set to their default values. Optional parameters specified by you are unaltered by f12ab and f12ac (unless they define invalid values) and so remain in effect for subsequent calls unless altered by you.

10.1 Optional Parameter Checklist and Default Values

The following list gives the valid options. For each option, we give the keyword, any essential optional qualifiers and the default value. A definition for each option can be found in Section 10.2. The minimum abbreviation of each keyword is underlined. The qualifier may be omitted. The letters i and r denote integer and double values required with certain options. The number ϵ is a generic notation for ***machine precision*** (see x02aj).

Optional Parameters	**Default Values**
Advisory	Default = the value returned by x04ab
Defaults	
Exact Shifts	Default = **Exact Shifts**

Generalized	See **Standard**
Initial Residual	See **Random Residual**
Iteration Limit	Default $= 300$
Largest Imaginary	See **Largest Magnitude**
Largest Magnitude	Default $=$ **Largest Magnitude**
Largest Real	See **Largest Magnitude**
List	See **Nolist**
Monitoring	Default $= -1$
Nolist	Default
Pointers	Default $=$ No
Print Level	Default $= 0$
Random Residual	Default $=$ **Random Residual**
Regular	Default
Regular Inverse	See **Regular**
Shifted Inverse Imaginary	See **Regular**
Shifted Inverse Real	See **Regular**
Smallest Imaginary	See **Largest Magnitude**
Smallest Magnitude	See **Largest Magnitude**
Smallest Real	See **Largest Magnitude**
Standard	Default
Supplied Shifts	See **Exact Shifts**
Tolerance	Default $= \epsilon$
Vectors	Default $=$ Ritz

10.2 Description of the Optional Parameters

Advisory i Default $=$ the value returned by x04ab

The output channel for advisory messages.

Defaults

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

Exact Shifts Default $=$ **Exact Shifts**
Supplied Shifts

During the Arnoldi 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 Arnoldi iterative process, you must supply shifts through array arguments of f12ab when f12ab returns with **irevcm** $= 3$; the real and imaginary parts of the shifts are supplied 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 Arnoldi iterations that can be performed before f12ab exits. If not all requested eigenvalues have converged to within **Tolerance** and the number of Arnoldi iterations has reached this limit then f12ab exits with an error; f12ac can still be called subsequently to return the number of converged eigenvalues, the converged eigenvalues and, if requested, the corresponding eigenvectors.

Largest Magnitude Default = **Largest Magnitude**
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 f12ab 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 x04ac to associate a file with a given channel number.

Pointers Default = No

During the iterative process and reverse communication calls to f12ab, required data can be communicated to and from f12ab 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 f12ab. For example, when **irevcm** = 1 f12ab returns the vector x in **x** and the matrix-vector product Bx in **mx** and expects the result of 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 f12ab 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 f12ad 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 f12ab.

≥ 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-orthogonalisation, the norm of initial/restarted starting vector.

≥ 20 Problem and timing statistics on final exit from f12ab. 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-orthogonalisation.

≥ 30 If **Monitoring** > 0, **Monitoring** is set, then on final iteration, the norm of the residual; when computing the real 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-orthogonalisation, 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, some indices.

≥ 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 = **Random Residual**
Initial Residual

To begin the Arnoldi iterative process, f12ab requires an initial residual vector. By default f12ab 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 f12ab through the array argument **resid**; this option can be set using optional parameter **Random Residual**.

Regular Default
Regular Inverse
Shifted Inverse Imaginary
Shifted Inverse Real

These options define the computational mode which in turn defines the form of operation $\mathrm{OP}(x)$ to be performed when f12ab returns with **irevcm** $= -1$ or 1 and the matrix-vector product Bx when f12ab 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 Real	$\mathrm{OP} = (A - \sigma I)^{-1}$ 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 $\mathrm{OP}(x)$.

Regular Inverse	$\mathrm{OP} = B^{-1}A$
Shifted Inverse Real with real shift	$\mathrm{OP} = (A - \sigma B)^{-1}B$, where σ is real
Shifted Inverse Real with complex shift	$\mathrm{OP} = \mathrm{Real}\left((A - \sigma B)^{-1}B\right)$, where σ is complex
Shifted Inverse Imaginary	$\mathrm{OP} = \mathrm{Imag}\left((A - \sigma B)^{-1}B\right)$, where σ is complex

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 f12ac 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 f12ac. 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 f12ac, if **z** is set equal to **v** (as in Section 9) then the Schur vectors in **v** are overwritten by the eigenvectors computed by f12ac.