NAG Toolbox for MATLAB

f12ar

1 Purpose

f12ar is an option setting function in a suite of functions consisting of f12an, f12ap, f12aq, f12ar and f12as, and may be used to supply individual optional parameters to f12ap and f12aq. The initialization function f12an **must** have been called prior to calling f12ar.

2 Syntax

```
[icomm, comm, ifail] = f12ar(str, icomm, comm)
```

3 Description

f12ar may be used to supply values for optional parameters to f12ap and f12aq. It is only necessary to call f12ar for those parameters whose values are to be different from their default values. One call to f12ar 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

```
'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.

f12ar 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. f12ar is intended to make the passing of options more transparent and follows the same principle as the single option setting functions in Chapter E04 (see e04ns for an example).

The setup function f12an must be called prior to the first call to f12ar and all calls to f12ar must precede the first call to f12ap, 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 f12an).

On initial entry: must remain unchanged following a call to the setup function f12an.

3: **comm**$(*)$ **– complex array**

Note: the dimension of the array **comm** must be at least $\max(1, 3 \times \mathbf{n} + 3 \times \mathbf{ncv} \times \mathbf{ncv} + 5 \times \mathbf{ncv} + 60)$ (see f12an).

On initial entry: must remain unchanged following a call to the setup function f12an.

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 f12an).

Contains data on the current options set.

2: **comm**$(*)$ **– complex array**

Note: the dimension of the array **comm** must be at least $\max(1, 3 \times \mathbf{n} + 3 \times \mathbf{ncv} \times \mathbf{ncv} + 5 \times \mathbf{ncv} + 60)$ (see f12an).

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 f12an has not been called or a communication array has become corrupted.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

```
n = int32(100);
nx = int32(10);
nev = int32(4);
ncv = int32(20);

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

sigma = complex(500);
rho = complex(10);
h = 1/double(n+1);
s = rho/2;
s1 = -1/h - s - sigma*h/6;
s2 =  2/h - 4*sigma*h/6;
s3 = -1/h + s - sigma*h/6;

dl = complex(repmat(s1, double(n-1), 1));
dd = complex(repmat(s2, double(n), 1));
du = complex(repmat(s3, double(n-1), 1));



% Initialisation Step
[icomm, comm, ifail] = f12an(n, nev, ncv);

% Set the mode
[icomm, comm, ifail] = f12ar('SHIFTED INVERSE', icomm, comm);
% Set the problem type
[icomm, comm, ifail] = f12ar('GENERALIZED', icomm, comm);

[dl, dd, du, du2, ipiv, info] = f07cr(dl, dd, du);

% Solve
while (irevcm ~= 5)
  [irevcm, resid, v, x, mx, nshift, comm, icomm, ifail] = ...
    f12ap(irevcm, resid, v, x, mx, comm, icomm);
  if (irevcm == -1)
    x = f12_mv(x);
    [x, info] = f07cs('N', dl, dd, du, du2, ipiv, x);
  elseif (irevcm == 1)
    [x, info] = f07cs('N', dl, dd, du, du2, ipiv, mx);
  elseif (irevcm == 2)
    x = f12_mv(x);
  elseif (irevcm == 4)
    [niter, nconv, ritz, rzest] = f12as(icomm, comm);
     fprintf('\nIteration %d, No. converged = %d, norm of estimates =
%16.8g', niter, nconv, norm(rzest(1:nev),1));
```

```
  end
end

% Post-process to compute eigenvalues/vectors
[nconv, d, z, v, comm, icomm, ifail] = f12aq(sigma, resid, v, comm,
icomm);
fprintf('\n\nThe %d generalised Ritz values closest to %s are:\n', nconv,
num2str(sigma));
for i=1:nconv
  fprintf('%d    %s\n', i, num2str(d(i)));
end
```

```
Iteration 1, No. converged = 3, norm of estimates =    3.0560047e-17
The 4 generalised Ritz values closest to 500+0i are:
1    509.939+4.832173e-15i
2    380.9092+3.750444e-12i
3    659.1558-1.202553e-12i
4    271.9412-3.031939e-12i
```

10 Optional Parameters

Several optional parameters for the computational functions f12ap and f12aq define choices in the problem specification or the algorithm logic. In order to reduce the number of formal parameters of f12ap and f12aq 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 f12ar prior to a call to f12ap, but after a call to f12an. 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 f12ap and f12aq (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
Generalized	See **Standard**
Initial Residual	See **Random Residual**
Iteration Limit	Default $= 300$
Largest Imaginary	See **Largest Magnitude**
Largest Magnitude	Default
Largest Real	See **Largest Magnitude**
List	See **Nolist**
Monitoring	Default $= -1$
Nolist	Default
Pointers	Default $=$ No
Print Level	Default $= 0$
Random Residual	Default
Regular	Default
Regular Inverse	See **Regular**

Shifted Inverse	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 = Schur

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
Supplied Shifts

During the Arnoldi iterative process, shifts are applied as part of the implicit restarting scheme. The shift strategy used by default and selected by the optional parameter **Exact Shifts** is strongly recommended over the alternative **Supplied Shifts**.

If **Exact Shifts** are used then these are computed internally by the algorithm in the implicit restarting scheme. This strategy is generally effective and cheaper to apply in terms of number of operations than using explicit shifts.

If **Supplied Shifts** are used then, during the Arnoldi iterative process, you must supply shifts through array arguments of f12ap when f12ap returns with **irevcm** = 3; the complex shifts are returned in **x** (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 f12ap exits. If not all requested eigenvalues have converged to within **Tolerance** and the number of Arnoldi iterations has reached this limit then f12ap exits with an error; f12aq 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 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 f12ap 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 f12ap, required data can be communicated to and from f12ap 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 f12ap. For example, when **irevcm** $= 1$ f12ap returns the vector x in **x** and the matrix-vector product Bx in **mx** and expects the result of the linear operation $\mathrm{OP}(x)$ 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 f12ap 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 f12ar 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 f12ap.

≥ 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 f12ap. 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 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.

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

Random Residual Default
Initial Residual

To begin the Arnoldi iterative process, f12ap requires an initial residual vector. By default f12ap 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 f12ap through the array argument **resid**; this option can be set using optional parameter **Random Residual**.

Regular Default
Regular Inverse
Shifted Inverse

These options define the computational mode which in turn defines the form of operation $\mathrm{OP}(x)$ to be performed when f12ap returns with **irevcm** $= -1$ or 1 and the matrix-vector product Bx when f12ap 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 $\mathrm{OP} = (A - \sigma I)^{-1}$

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 $\mathrm{OP} = (A - \sigma B)^{-1}B$

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 = Schur

The function f12aq 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 f12aq. 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 f12aq, if **z** is set equal to **v** (as in Section 9) then the Schur vectors in **v** are overwritten by the eigenvectors computed by f12aq.