NAG Toolbox

nag_eigen_real_symm_sparse_eigsys (f02fj)

1 Purpose

nag_eigen_real_symm_sparse_eigsys (f02fj) finds eigenvalues and eigenvectors of a real sparse symmetric or generalized symmetric eigenvalue problem.

2 Syntax

```
[m, noits, x, d, user, ifail] = nag_eigen_real_symm_sparse_eigsys(m, noits, tol,
dot, image, monit, novecs, x, 'n', n, 'k', k, 'user', user)

[m, noits, x, d, user, ifail] = f02fj(m, noits, tol, dot, image, monit, novecs,
x, 'n', n, 'k', k, 'user', user)
```

Note: the interface to this routine has changed since earlier releases of the toolbox:

At Mark 22: **n** was made optional.

3 Description

nag_eigen_real_symm_sparse_eigsys (f02fj) finds the m eigenvalues of largest absolute value and the corresponding eigenvectors for the real eigenvalue problem

$$Cx = \lambda x \tag{1}$$

where C is an n by n matrix such that

$$BC = C^{\mathrm{T}}B \tag{2}$$

for a given positive definite matrix B. C is said to be B-symmetric. Different specifications of C allow for the solution of a variety of eigenvalue problems. For example, when

$$C = A \quad \text{and} \quad B = I \quad \text{where} \quad A = A^{\mathrm{T}}$$

the function finds the m eigenvalues of largest absolute magnitude for the standard symmetric eigenvalue problem

$$Ax = \lambda x. \tag{3}$$

The function is intended for the case where A is sparse.

As a second example, when

$$C = B^{-1}A$$

where

$$A = A^{\mathrm{T}}$$

the function finds the m eigenvalues of largest absolute magnitude for the generalized symmetric eigenvalue problem

$$Ax = \lambda Bx. \tag{4}$$

The function is intended for the case where A and B are sparse.

The function does not require C explicitly, but C is specified via **image** which, given an n-element vector z, computes the image w given by

$$w = Cz.$$

For instance, in the above example, where $C = B^{-1}A$, **image** will need to solve the positive definite system of equations $Bw = Az$ for w.

To find the m eigenvalues of smallest absolute magnitude of (3) we can choose $C = A^{-1}$ and hence find the reciprocals of the required eigenvalues, so that **image** will need to solve $Aw = z$ for w, and correspondingly for (4) we can choose $C = A^{-1}B$ and solve $Aw = Bz$ for w.

A table of examples of choice of **image** is given in Table 1. It should be remembered that the function also returns the corresponding eigenvectors and that B is positive definite. Throughout A is assumed to be symmetric and, where necessary, nonsingularity is also assumed.

Eigenvalues Required	**Problem**		
	$Ax = \lambda x (B = I)$	$Ax = \lambda Bx$	$ABx = \lambda x$
Largest	Compute $w = Az$	Solve $Bw = Az$	Compute $w = ABz$
Smallest (Find $1/\lambda$)	Solve $Aw = z$	Solve $Aw = Bz$	Solve $Av = z$, $Bw = v$
Furthest from σ (Find $\lambda - \sigma$)	Compute $w = (A - \sigma I)z$	Solve $Bw = (A - \sigma B)z$	Compute $w = (AB - \sigma I)z$
Closest to σ (Find $1/(\lambda - \sigma)$)	Solve $(A - \sigma I)w = z$	Solve $(A - \sigma B)w = Bz$	Solve $(AB - \sigma I)w = z$

Table 1
The Requirement of **image** for Various Problems.

The matrix B also need not be supplied explicitly, but is specified via **dot** which, given n-element vectors z and w, computes the generalized dot product $w^{\mathrm{T}}Bz$.

nag_eigen_real_symm_sparse_eigsys (f02fj) is based upon routine SIMITZ (see Nikolai (1979)), which is itself a derivative of the Algol procedure ritzit (see Rutishauser (1970)), and uses the method of simultaneous (subspace) iteration. (See Parlett (1998) for a description, analysis and advice on the use of the method.)

The function performs simultaneous iteration on $k > m$ vectors. Initial estimates to $p \leq k$ eigenvectors, corresponding to the p eigenvalues of C of largest absolute value, may be supplied to nag_eigen_real_symm_sparse_eigsys (f02fj). When possible k should be chosen so that the kth eigenvalue is not too close to the m required eigenvalues, but if k is initially chosen too small then nag_eigen_real_symm_sparse_eigsys (f02fj) may be re-entered, supplying approximations to the k eigenvectors found so far and with k then increased.

At each major iteration nag_eigen_real_symm_sparse_eigsys (f02fj) solves an r by r ($r \leq k$) eigenvalue sub-problem in order to obtain an approximation to the eigenvalues for which convergence has not yet occurred. This approximation is refined by Chebyshev acceleration.

4 References

Nikolai P J (1979) Algorithm 538: Eigenvectors and eigenvalues of real generalized symmetric matrices by simultaneous iteration *ACM Trans. Math. Software* **5** 118–125

Parlett B N (1998) *The Symmetric Eigenvalue Problem* SIAM, Philadelphia

Rutishauser H (1969) Computational aspects of F L Bauer's simultaneous iteration method *Numer. Math.* **13** 4–13

Rutishauser H (1970) Simultaneous iteration method for symmetric matrices *Numer. Math.* **16** 205–223

5 Parameters

5.1 Compulsory Input Parameters

1: **m** – INTEGER

m, the number of eigenvalues required.

Constraint: **m** ≥ 1.

2: **noits** – INTEGER

The maximum number of major iterations (eigenvalue sub-problems) to be performed. If **noits** ≤ 0, the value 100 is used in place of **noits**.

3: **tol** – REAL (KIND=nag_wp)

A relative tolerance to be used in accepting eigenvalues and eigenvectors. If the eigenvalues are required to about t significant figures, **tol** should be set to about 10^{-t}. d_i is accepted as an eigenvalue as soon as two successive approximations to d_i differ by less than $\left(\left|\tilde{d}_i\right| \times \textbf{tol}\right)/10$, where $\tilde{d}_i$ is the latest approximation to d_i. Once an eigenvalue has been accepted, an eigenvector is accepted as soon as $(d_i f_i)/(d_i - d_k) < \textbf{tol}$, where f_i is the normalized residual of the current approximation to the eigenvector (see Section 9 for further information). The values of the f_i and d_i can be printed from **monit**. If **tol** is supplied outside the range $(\epsilon, 1.0)$, where ϵ is the ***machine precision***, the value ϵ is used in place of **tol**.

4: **dot** – REAL (KIND=nag_wp) FUNCTION, supplied by the user.

dot must return the value $w^{\mathrm{T}}Bz$ for given vectors w and z. For the standard eigenvalue problem, where $B = I$, **dot** must return the dot product $w^{\mathrm{T}}z$.

```
[result, iflag, user] = dot(iflag, n, z, w, user)
```

Input Parameters

1: **iflag** – INTEGER

Is always non-negative.

2: **n** – INTEGER

The number of elements in the vectors z and w and the order of the matrix B.

3: **z**(**n**) – REAL (KIND=nag_wp) array

The vector z for which $w^{\mathrm{T}}Bz$ is required.

4: **w**(**n**) – REAL (KIND=nag_wp) array

The vector w for which $w^{\mathrm{T}}Bz$ is required.

5: **user** – REAL (KIND=nag_wp) array

dot is called from nag_eigen_real_symm_sparse_eigsys (f02fj) with the object supplied to nag_eigen_real_symm_sparse_eigsys (f02fj).

Output Parameters

1: **result**

result returns the value $w^{\mathrm{T}}Bz$ for given vectors w and z.

2: **iflag** – INTEGER

May be used as a flag to indicate a failure in the computation of $w^{\mathrm{T}}Bz$. If **iflag** is negative on exit from **dot**, nag_eigen_real_symm_sparse_eigsys (f02fj) will exit immediately with **ifail** set to **iflag**. Note that in this case **dot** must still be assigned a value.

3: **user** – REAL (KIND=nag_wp) array

5: **image** – SUBROUTINE, supplied by the user.

image must return the vector $w = Cz$ for a given vector z.

```
[iflag, w, user] = image(iflag, n, z, user)
```

Input Parameters

1: **iflag** – INTEGER

Is always non-negative.

2: **n** – INTEGER

n, the number of elements in the vectors w and z, and the order of the matrix C.

3: **z**(**n**) – REAL (KIND=nag_wp) array

The vector z for which Cz is required.

4: **user** – REAL (KIND=nag_wp) array

image is called from nag_eigen_real_symm_sparse_eigsys (f02fj) with the object supplied to nag_eigen_real_symm_sparse_eigsys (f02fj).

Output Parameters

1: **iflag** – INTEGER

May be used as a flag to indicate a failure in the computation of w. If **iflag** is negative on exit from **image**, nag_eigen_real_symm_sparse_eigsys (f02fj) will exit immediately with **ifail** set to **iflag**.

2: **w**(**n**) – REAL (KIND=nag_wp) array

The vector $w = Cz$.

3: **user** – REAL (KIND=nag_wp) array

6: **monit** – SUBROUTINE, supplied by the NAG Library or the user.

monit is used to monitor the progress of nag_eigen_real_symm_sparse_eigsys (f02fj). **monit** may be the dummy function nag_eigen_real_symm_sparse_eigsys_dummy_monit (f02fjz) if no monitoring is actually required. (nag_eigen_real_symm_sparse_eigsys_dummy_monit (f02fjz) is included in the NAG Toolbox.) **monit** is called after the solution of each eigenvalue sub-problem and also just prior to return from nag_eigen_real_symm_sparse_eigsys (f02fj). The arguments **istate** and **nextit** allow selective printing by **monit**.

```
monit(istate, nextit, nevals, nevecs, k, f, d)
```

Input Parameters

1: **istate** – INTEGER

Specifies the state of nag_eigen_real_symm_sparse_eigsys (f02fj).

istate $= 0$
: No eigenvalue or eigenvector has just been accepted.

istate $= 1$
: One or more eigenvalues have been accepted since the last call to **monit**.

istate $= 2$
: One or more eigenvectors have been accepted since the last call to **monit**.

istate $= 3$
: One or more eigenvalues and eigenvectors have been accepted since the last call to **monit**.

istate $= 4$
: Return from nag_eigen_real_symm_sparse_eigsys (f02fj) is about to occur.

2: **nextit** – INTEGER

The number of the next iteration.

3: **nevals** – INTEGER

The number of eigenvalues accepted so far.

4: **nevecs** – INTEGER

The number of eigenvectors accepted so far.

5: **k** – INTEGER

k, the number of simultaneous iteration vectors.

6: **f**(**k**) – REAL (KIND=nag_wp) array

A vector of error quantities measuring the state of convergence of the simultaneous iteration vectors. See **tol** and Section 9 for further details. Each element of **f** is initially set to the value 4.0 and an element remains at 4.0 until the corresponding vector is tested.

7: **d**(**k**) – REAL (KIND=nag_wp) array

d(i) contains the latest approximation to the absolute value of the ith eigenvalue of C.

7: **novecs** – INTEGER

The number of approximate vectors that are being supplied in **x**. If **novecs** is outside the range $(0, \mathbf{k})$, the value 0 is used in place of **novecs**.

8: **x**$(ldx, \mathbf{k})$ – REAL (KIND=nag_wp) array

ldx, the first dimension of the array, must satisfy the constraint $ldx \geq \mathbf{n}$.

If $0 < \mathbf{novecs} \leq \mathbf{k}$, the first **novecs** columns of **x** must contain approximations to the eigenvectors corresponding to the **novecs** eigenvalues of largest absolute value of C. Supplying approximate eigenvectors can be useful when reasonable approximations are known, or when nag_eigen_real_symm_sparse_eigsys (f02fj) is being restarted with a larger value of **k**. Otherwise it is not necessary to supply approximate vectors, as simultaneous iteration vectors will be generated randomly by nag_eigen_real_symm_sparse_eigsys (f02fj).

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the first dimension of the array **x**.

n, the order of the matrix C.

Constraint: $\mathbf{n} \geq 1$.

2: **k** – INTEGER

Suggested value: $\mathbf{k} = \mathbf{m} + 4$ will often be a reasonable choice in the absence of better information.

Default: $\mathbf{k} = \mathbf{m} + 4$

Default: the second dimension of the array **x**.

The number of simultaneous iteration vectors to be used. Too small a value of **k** may inhibit convergence, while a larger value of **k** incurs additional storage and additional work per iteration.

Constraint: $\mathbf{m} < \mathbf{k} \leq \mathbf{n}$.

3: **user** – REAL (KIND=nag_wp) array

user is not used by nag_eigen_real_symm_sparse_eigsys (f02fj), but is passed to **dot** and **image**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

5.3 Output Parameters

1: **m** – INTEGER

m', the number of eigenvalues actually found. It is equal to m if $\textbf{ifail} = 0$ on exit, and is less than m if $\textbf{ifail} = 2$, 3 or 4. See Section 6 and Section 9 for further information.

2: **noits** – INTEGER

The number of iterations actually performed.

3: $\mathbf{x}(ldx, \mathbf{k})$ – REAL (KIND=nag_wp) array

If $\textbf{ifail} = 0$, 2, 3 or 4, the first m' columns contain the eigenvectors corresponding to the eigenvalues returned in the first m' elements of **d**; and the next $k - m' - 1$ columns contain approximations to the eigenvectors corresponding to the approximate eigenvalues returned in the next $k - m' - 1$ elements of **d**. Here m' is the value returned in **m**, the number of eigenvalues actually found. The kth column is used as workspace.

4: $\mathbf{d}(\mathbf{k})$ – REAL (KIND=nag_wp) array

If $\textbf{ifail} = 0$, 2, 3 or 4, the first m' elements contain the first m' eigenvalues in decreasing order of magnitude; and the next $k - m' - 1$ elements contain approximations to the next $k - m' - 1$ eigenvalues. Here m' is the value returned in **m**, the number of eigenvalues actually found. $\mathbf{d}(k)$ contains the value e where $(-e, e)$ is the latest interval over which Chebyshev acceleration is performed.

5: **user** – REAL (KIND=nag_wp) array

6: **ifail** – INTEGER

$\textbf{ifail} = 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail < 0 (*warning*)

A negative value of **ifail** indicates an exit from nag_eigen_real_symm_sparse_eigsys (f02fj) because you have set **iflag** negative in **dot** or **image**. The value of **ifail** will be the same as your setting of **iflag**.

ifail $= 1$

On entry, **n** < 1,
or **m** < 1,
or **m** $\geq$ **k**,
or **k** $>$ **n**,
or $ldx <$ **n**,
or $lwork < 3 \times$ **k** $+ \max($**k** $\times$ **k**$, 2 \times$ **n**$)$,
or $lruser < 1$,
or $liuser < 1$.

ifail $= 2$ (*warning*)

Not all the requested eigenvalues and vectors have been obtained. Approximations to the rth eigenvalue are oscillating rapidly indicating that severe cancellation is occurring in the rth eigenvector and so **m** is returned as $(r - 1)$. A restart with a larger value of **k** may permit convergence.

ifail $= 3$ (*warning*)

Not all the requested eigenvalues and vectors have been obtained. The rate of convergence of the remaining eigenvectors suggests that more than **noits** iterations would be required and so the input value of **m** has been reduced. A restart with a larger value of **k** may permit convergence.

ifail $= 4$ (*warning*)

Not all the requested eigenvalues and vectors have been obtained. **noits** iterations have been performed. A restart, possibly with a larger value of **k**, may permit convergence.

ifail $= 5$

This error is very unlikely to occur, but indicates that convergence of the eigenvalue sub-problem has not taken place. Restarting with a different set of approximate vectors may allow convergence. If this error occurs you should check carefully that nag_eigen_real_symm_sparse_eigsys (f02fj) is being called correctly.

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

Eigenvalues and eigenvectors will normally be computed to the accuracy requested by the argument **tol**, but eigenvectors corresponding to small or to close eigenvalues may not always be computed to the accuracy requested by the argument **tol**. Use of the **monit** to monitor acceptance of eigenvalues and eigenvectors is recommended.

8 Further Comments

The time taken by nag_eigen_real_symm_sparse_eigsys (f02fj) will be principally determined by the time taken to solve the eigenvalue sub-problem and the time taken by **dot** and **image**. The time taken to solve an eigenvalue sub-problem is approximately proportional to nk^2. It is important to be aware that several calls to **dot** and **image** may occur on each major iteration.

As can be seen from Table 1, many applications of nag_eigen_real_symm_sparse_eigsys (f02fj) will require the **image** to solve a system of linear equations. For example, to find the smallest eigenvalues of $Ax = \lambda Bx$, **image** needs to solve equations of the form $Aw = Bz$ for w and functions from Chapters F01 and F04 will frequently be useful in this context. In particular, if A is a positive definite variable band matrix, nag_linsys_real_posdef_vband_solve (f04mc) may be used after A has been factorized by nag_matop_real_vband_posdef_fac (f01mc). Thus factorization need be performed only once prior to calling nag_eigen_real_symm_sparse_eigsys (f02fj). An illustration of this type of use is given in the example program.

An approximation $\tilde{d}_h$, to the ith eigenvalue, is accepted as soon as $\tilde{d}_h$ and the previous approximation differ by less than $\left|\tilde{d}_h\right| \times \textbf{tol}/10$. Eigenvectors are accepted in groups corresponding to clusters of eigenvalues that are equal, or nearly equal, in absolute value and that have already been accepted. If d_r is the last eigenvalue in such a group and we define the residual r_j as

$$r_j = Cx_j - y_r$$

where y_r is the projection of Cx_j, with respect to B, onto the space spanned by $x_1, x_2, \ldots, x_r$, and x_j is the current approximation to the jth eigenvector, then the value f_i returned in **monit** is given by

$$f_i = \max \left\|r_j\right\|_B / \left\|Cx_j\right\|_B \quad \|x\|_B^2 = x^{\mathrm{T}}Bx$$

and each vector in the group is accepted as an eigenvector if

$$(|d_r|f_r)/(|d_r| - e) < \textbf{tol},$$

where e is the current approximation to $\left|\tilde{d}_k\right|$. The values of the f_i are systematically increased if the convergence criteria appear to be too strict. See Rutishauser (1970) for further details.

The algorithm implemented by nag_eigen_real_symm_sparse_eigsys (f02fj) differs slightly from SIMITZ (see Nikolai (1979)) in that the eigenvalue sub-problem is solved using the singular value decomposition of the upper triangular matrix R of the Gram–Schmidt factorization of Cx_r, rather than forming $R^{\mathrm{T}}R$.

9 Example

This example finds the four eigenvalues of smallest absolute value and corresponding eigenvectors for the generalized symmetric eigenvalue problem $Ax = \lambda Bx$, where A and B are the 16 by 16 matrices

$$A = -\frac{1}{4}\begin{pmatrix}
-4 & 1 & & & 1 & & & & & & & & & & & \\
1 & -4 & 1 & & & 1 & & & & & & & & & & \\
 & 1 & -4 & 1 & & & 1 & & & & & & & & & \\
 & & 1 & -4 & 1 & & & 1 & & & & & & & & \\
1 & & & 1 & -4 & 1 & & & 1 & & & & & & & \\
 & 1 & & & 1 & -4 & 1 & & & 1 & & & & & & \\
 & & 1 & & & 1 & -4 & 1 & & & 1 & & & & & \\
 & & & 1 & & & 1 & -4 & 1 & & & 1 & & & & \\
 & & & & 1 & & & 1 & -4 & 1 & & & 1 & & & \\
 & & & & & 1 & & & 1 & -4 & 1 & & & 1 & & \\
 & & & & & & 1 & & & 1 & -4 & 1 & & & 1 & \\
 & & & & & & & 1 & & & 1 & -4 & 1 & & & 1 \\
 & & & & & & & & 1 & & & 1 & -4 & 1 & & \\
 & & & & & & & & & 1 & & & 1 & -4 & 1 & \\
 & & & & & & & & & & 1 & & & 1 & -4 & 1 \\
 & & & & & & & & & & & 1 & & & 1 & -4
\end{pmatrix}$$

$$B = -\frac{1}{2}\begin{pmatrix}
-2 & 1 & & & & & & & & & & & & & & \\
1 & -2 & 1 & & & & & & & & & & & & & \\
 & 1 & -2 & 1 & & & & & & & & & & & & \\
 & & 1 & -2 & 1 & & & & & & & & & & & \\
 & & & 1 & -2 & 1 & & & & & & & & & & \\
 & & & & 1 & -2 & 1 & & & & & & & & & \\
 & & & & & 1 & -2 & 1 & & & & & & & & \\
 & & & & & & 1 & -2 & 1 & & & & & & & \\
 & & & & & & & 1 & -2 & 1 & & & & & & \\
 & & & & & & & & 1 & -2 & 1 & & & & & \\
 & & & & & & & & & 1 & -2 & 1 & & & & \\
 & & & & & & & & & & 1 & -2 & 1 & & & \\
 & & & & & & & & & & & 1 & -2 & 1 & & \\
 & & & & & & & & & & & & 1 & -2 & 1 & \\
 & & & & & & & & & & & & & 1 & -2 & 1 \\
 & & & & & & & & & & & & & & 1 & -2
\end{pmatrix}$$

tol is taken as 0.0001 and 6 iteration vectors are used. nag_sparse_real_symm_precon_ichol (f11ja) is used to factorize the matrix A, prior to calling nag_eigen_real_symm_sparse_eigsys (f02fj), and nag_sparse_real_symm_solve_ichol (f11jc) is used within **image** to solve the equations $Aw = Bz$ for w.

Output from **monit** occurs each time **istate** is nonzero. Note that the required eigenvalues are the reciprocals of the eigenvalues returned by nag_eigen_real_symm_sparse_eigsys (f02fj).

9.1 Program Text

```
     function f02fj_example

fprintf('f02fj example results\n\n');

n = 16;
m = nag_int(4);
noits = nag_int(1000);
tol = 0.0001;
novecs = nag_int(0);
x = zeros(n, m+2);

% a, b will be passed in user
a = diag(ones(n,1)) + diag(-0.25*ones(n-1,1),1) + ...
    diag(-0.25*ones(n-1,1),-1) + diag(-0.25*ones(n-4,1),4) + ...
    diag(-0.25*ones(n-4,1),-4);
b = diag(ones(n,1)) + diag(-0.5*ones(n-1,1),1) + diag(-0.5*ones(n-1,1),-1);
```

```
[m, noits, x, d, user, ifail] = ...
  f02fj(...
        m, noits, tol, @dot, @image, @monit, novecs, x, 'user', {a, b});

fprintf('\nFinal results\n\n');
disp('Eigenvalues');
disp(1./d(1:m)');
disp('Eigenvectors');
% Normalize eigenvectors before printing
disp(x(:,1:m)/diag(x(1,1:m)));

function [result, iflag, user] = dot(iflag, n, z, w, user)
  b = user{2};
  result=transpose(w)*b*z;

function [iflag, w, user] = image(iflag, n, z, user)

  a=user{1};
  b=user{2};

  w=inv(a)*b*z;

function monit(istate, nextit, nevals, nevecs, k, f, d)

  if (istate ~= 0)
    fprintf('\n  istate = %d nextit = %d\n', istate, nextit);
    fprintf('  nevals = %d nevecs = %d\n', nevals, nevecs);
    fprintf('      f          d \n');
    for i=1:double(k)
      fprintf('%11.3f %11.3f\n',f(i), d(i));
    end
  end
```

9.2 Program Results

```
     f02fj example results

  istate = 3 nextit = 17
  nevals = 1 nevecs = 1
      f           d
    0.000       1.822
    4.000       1.695
    4.000       1.668
    4.000       1.460
    4.000       1.275
    4.000       1.132

  istate = 4 nextit = 30
  nevals = 4 nevecs = 4
      f           d
    0.000       1.822
    0.000       1.695
    0.000       1.668
    0.000       1.460
    4.000       1.275
    4.000       1.153

Final results

Eigenvalues
    0.5488    0.5900    0.5994    0.6850

Eigenvectors
    1.0000    1.0000    1.0000    1.0000
   -1.1586   -0.8089    1.1274   -1.2368
    1.1682   -0.7555   -1.0699    1.9252
   -1.1298    0.7444   -1.3512   -1.3185
    1.6919    1.4943    1.8266    0.8027
   -1.8797   -1.2826    1.7928   -0.4766
```

```
     1.8854   -1.2505   -1.7589    1.4809
    -1.7604    1.3538   -2.0148   -0.6525
     1.7604    1.3538    2.0148   -0.6525
    -1.8854   -1.2505    1.7589    1.4809
     1.8797   -1.2826   -1.7928   -0.4766
    -1.6919    1.4943   -1.8266    0.8027
     1.1298    0.7444    1.3512   -1.3185
    -1.1682   -0.7555    1.0699    1.9252
     1.1586   -0.8089   -1.1274   -1.2368
    -1.0000    1.0000   -1.0000    1.0000
```