NAG Toolbox

nag_matop_complex_herm_matrix_fun (f01ff)

1 Purpose

nag_matop_complex_herm_matrix_fun (f01ff) computes the matrix function, $f(A)$, of a complex Hermitian n by n matrix A. $f(A)$ must also be a complex Hermitian matrix.

2 Syntax

```
[a, user, iflag, ifail] = nag_matop_complex_herm_matrix_fun(uplo, a, f, 'n', n, 'user', user)
[a, user, iflag, ifail] = f01ff(uplo, a, f, 'n', n, 'user', user)
```

3 Description

$f(A)$ is computed using a spectral factorization of A

$$A = QDQ^{\mathrm{H}},$$

where D is the real diagonal matrix whose diagonal elements, d_i, are the eigenvalues of A, Q is a unitary matrix whose columns are the eigenvectors of A and Q^{H} is the conjugate transpose of Q. $f(A)$ is then given by

$$f(A) = Qf(D)Q^{\mathrm{H}},$$

where $f(D)$ is the diagonal matrix whose ith diagonal element is $f(d_i)$. See for example Section 4.5 of Higham (2008). $f(d_i)$ is assumed to be real.

4 References

Higham N J (2008) *Functions of Matrices: Theory and Computation* SIAM, Philadelphia, PA, USA

5 Parameters

5.1 Compulsory Input Parameters

1: **uplo** – CHARACTER(1)

If **uplo** = 'U', the upper triangle of the matrix A is stored.

If **uplo** = 'L', the lower triangle of the matrix A is stored.

Constraint: **uplo** = 'U' or 'L'.

2: **a**$(lda, :)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **a** must be at least $\max(1, \mathbf{n})$.

The second dimension of the array **a** must be at least **n**.

The n by n Hermitian matrix A.

If **uplo** = 'U', the upper triangular part of a must be stored and the elements of the array below the diagonal are not referenced.

If **uplo** = 'L', the lower triangular part of a must be stored and the elements of the array above the diagonal are not referenced.

3: **f** – SUBROUTINE, supplied by the user.

The function **f** evaluates $f(z_i)$ at a number of points z_i.

```
[iflag, fx, user] = f(iflag, n, x, user)
```

Input Parameters

1: **iflag** – INTEGER

iflag will be zero.

2: **n** – INTEGER

n, the number of function values required.

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

The n points $x_1, x_2, \ldots, x_n$ at which the function f is to be evaluated.

4: **user** – INTEGER array

f is called from nag_matop_complex_herm_matrix_fun (f01ff) with the object supplied to nag_matop_complex_herm_matrix_fun (f01ff).

Output Parameters

1: **iflag** – INTEGER

iflag should either be unchanged from its entry value of zero, or may be set nonzero to indicate that there is a problem in evaluating the function $f(x)$; for instance $f(x)$ may not be defined, or may be complex. If **iflag** is returned as nonzero then nag_matop_complex_herm_matrix_fun (f01ff) will terminate the computation, with **ifail** $= -6$.

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

The n function values. **fx**(i) should return the value $f(x_i)$, for $i = 1, 2, \ldots, n$.

3: **user** – INTEGER array

5.2 Optional Input Parameters

1: **n** – INTEGER

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

n, the order of the matrix A.

Constraint: **n** ≥ 0.

2: **user** – INTEGER array

user is not used by nag_matop_complex_herm_matrix_fun (f01ff), but is passed to **f**. 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: **a**$(lda, :)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **a** will be $\max(1, \mathbf{n})$.

The second dimension of the array **a** will be **n**.

If **ifail** $= 0$, the upper or lower triangular part of the n by n matrix function, $f(A)$.

2: **user** – INTEGER array

3: **iflag** – INTEGER

iflag $= 0$, unless you have set **iflag** nonzero inside **f**, in which case **iflag** will be the value you set and **ifail** will be set to **ifail** $= -6$.

4: **ifail** – INTEGER

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

The computation of the spectral factorization failed to converge.

ifail $= -1$

Constraint: **uplo** = 'L' or 'U'.

ifail $= -2$

Constraint: **n** ≥ 0.

ifail $= -3$

An internal error occurred when computing the spectral factorization. Please contact NAG.

ifail $= -4$

Constraint: $lda \geq$ **n**.

ifail $= -6$

iflag was set to a nonzero value in **f**.

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

Provided that $f(D)$ can be computed accurately then the computed matrix function will be close to the exact matrix function. See Section 10.2 of Higham (2008) for details and further discussion.

8 Further Comments

The integer allocatable memory required is **n**, the double allocatable memory required is $4 \times \mathbf{n} - 2$ and the complex allocatable memory required is approximately $(\mathbf{n} + nb + 1) \times \mathbf{n}$, where nb is the block size required by nag_lapack_zheev (f08fn).

The cost of the algorithm is $O\left(n^3\right)$ plus the cost of evaluating $f(D)$. If $\hat{\lambda}_i$ is the ith computed eigenvalue of A, then the user-supplied function **f** will be asked to evaluate the function f at $f\left(\hat{\lambda}_i\right)$, for $i = 1, 2, \ldots, n$.

For further information on matrix functions, see Higham (2008).

nag_matop_real_symm_matrix_fun (f01ef) can be used to find the matrix function $f(A)$ for a real symmetric matrix A.

9 Example

This example finds the matrix cosine, $\cos(A)$, of the Hermitian matrix

$$A = \begin{pmatrix} 1 & 2+i & 3+2i & 4+3i \\ 2-i & 1 & 2+i & 3+2i \\ 3-2i & 2-i & 1 & 2+i \\ 4-3i & 3-2i & 2-i & 1 \end{pmatrix}.$$

9.1 Program Text

```
function f01ff_example

fprintf('f01ff example results\n\n');

uplo = 'u';
a = [ 1,  2 + 1i,  3 + 2i,  4 + 3i;
      0,  1 + 0i,  2 + 1i,  3 + 2i;
      0,       0,  1 + 0i,  2 + 1i;
      0,       0,       0,  1 + 0i];

% Compute f(a)
[cosa, user, iflag, ifail] = ...
f01ff(uplo, a, @f);

% Display results
[ifail] = x04da( ...
                uplo, 'n', cosa, 'Hermitian f(A) = cos(A)');

function [iflag, fx, user] = f(iflag, n, x, user)
  fx = cos(x);
```

9.2 Program Results

```
     f01ff example results

 Hermitian f(A) = cos(A)
          1       2       3       4
 1   0.0904 -0.3377 -0.1009 -0.1092
     0.0000 -0.0273 -0.0594 -0.1586

 2           0.4265 -0.3139 -0.1009
             0.0000 -0.0273 -0.0594

 3                  0.4265 -0.3377
                    0.0000 -0.0273

 4                         0.0904
                           0.0000
```