

NAG Toolbox

nag_matop_real_gen_matrix_fun_num (f01e1)

1 Purpose

nag_matop_real_gen_matrix_fun_num (f01e1) computes the matrix function, $f(A)$, of a real n by n matrix A . Numerical differentiation is used to evaluate the derivatives of f when they are required.

2 Syntax

```
[a, user, iflag, imnorm, ifail] = nag_matop_real_gen_matrix_fun_num(a, f, 'n', n,
'user', user)
[a, user, iflag, imnorm, ifail] = f01e1(a, f, 'n', n, 'user', user)
```

3 Description

$f(A)$ is computed using the Schur–Parlett algorithm described in Higham (2008) and Davies and Higham (2003). The coefficients of the Taylor series used in the algorithm are evaluated using the numerical differentiation algorithm of Lyness and Moler (1967).

The scalar function f is supplied via function **f** which evaluates $f(z_i)$ at a number of points z_i .

4 References

Davies P I and Higham N J (2003) A Schur–Parlett algorithm for computing matrix functions. *SIAM J. Matrix Anal. Appl.* **25**(2) 464–485

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

Lyness J N and Moler C B (1967) Numerical differentiation of analytic functions *SIAM J. Numer. Anal.* **4**(2) 202–210

5 Parameters

5.1 Compulsory Input Parameters

- 1: **a**(lda,:) – REAL (KIND=nag_wp) array
 The first dimension of the array **a** must be at least **n**.
 The second dimension of the array **a** must be at least **n**.
 The n by n matrix A .
- 2: **f** – SUBROUTINE, supplied by the user.
 The function **f** evaluates $f(z_i)$ at a number of points z_i .

```
[iflag, fz, user] = f(iflag, nz, z, user)
```

Input Parameters

- 1: **iflag** – INTEGER
iflag will be zero.

- 2: **nz** – INTEGER
 n_z , the number of function values required.
- 3: **z(nz)** – COMPLEX (KIND=nag_wp) array
 The n_z points z_1, z_2, \dots, z_{n_z} at which the function f is to be evaluated.
- 4: **user** – INTEGER array
f is called from nag_matop_real_gen_matrix_fun_num (f01el) with the object supplied to nag_matop_real_gen_matrix_fun_num (f01el).

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 $f(z_i)$; for instance $f(z_i)$ may not be defined. If **iflag** is returned as nonzero then nag_matop_real_gen_matrix_fun_num (f01el) will terminate the computation, with **ifail** = 2.
- 2: **fz(nz)** – COMPLEX (KIND=nag_wp) array
 The n_z function values. **fz**(i) should return the value $f(z_i)$, for $i = 1, 2, \dots, n_z$. If z_i lies on the real line, then so must $f(z_i)$.
- 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: $n \geq 0$.
- 2: **user** – INTEGER array
user is not used by nag_matop_real_gen_matrix_fun_num (f01el), 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,:)** – REAL (KIND=nag_wp) array
 The first dimension of the array **a** will be **n**.
 The second dimension of the array **a** will be **n**.
 The n by n matrix, $f(A)$.
- 2: **user** – INTEGER array
- 3: **iflag** – INTEGER
iflag = 0, unless **iflag** has been set nonzero inside **f**, in which case **iflag** will be the value set and **ifail** will be set to **ifail** = 2.

4: **imnorm** – REAL (KIND=nag_wp)

If A has complex eigenvalues, nag_matop_real_gen_matrix_fun_num (f01el) will use complex arithmetic to compute $f(A)$. The imaginary part is discarded at the end of the computation, because it will theoretically vanish. **imnorm** contains the 1-norm of the imaginary part, which should be used to check that the routine has given a reliable answer.

If a has real eigenvalues, nag_matop_real_gen_matrix_fun_num (f01el) uses real arithmetic and **imnorm** = 0.

5: **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 = 1

A Taylor series failed to converge after 40 terms. Further Taylor series coefficients can no longer reliably be obtained by numerical differentiation.

ifail = 2

iflag has been set nonzero by the user.

ifail = 3

There was an error whilst reordering the Schur form of A .

Note: this failure should not occur and suggests that the function has been called incorrectly.

ifail = 4

The function was unable to compute the Schur decomposition of A .

Note: this failure should not occur and suggests that the function has been called incorrectly.

ifail = 5

An unexpected internal error occurred. Please contact NAG.

ifail = -1

Input argument number $\langle value \rangle$ is invalid.

ifail = -3

On entry, argument lda is invalid.

Constraint: $lda \geq n$.

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

For a normal matrix A (for which $A^T A = A A^T$) the Schur decomposition is diagonal and the algorithm reduces to evaluating f at the eigenvalues of A and then constructing $f(A)$ using the Schur vectors. See Section 9.4 of Higham (2008) for further discussion of the Schur–Parlett algorithm, and Lyness and Moler (1967) for a discussion of numerical differentiation.

8 Further Comments

The integer allocatable memory required is n . If A has real eigenvalues then up to $6n^2$ of double allocatable memory may be required. If A has complex eigenvalues then up to $6n^2$ of complex allocatable memory may be required.

The cost of the Schur–Parlett algorithm depends on the spectrum of A , but is roughly between $28n^3$ and $n^4/3$ floating-point operations. There is an additional cost in numerically differentiating f , in order to obtain the Taylor series coefficients. If the derivatives of f are known analytically, then `nag_matop_real_gen_matrix_fun_usd` (f01em) can be used to evaluate $f(A)$ more accurately. If A is real symmetric then it is recommended that `nag_matop_real_symm_matrix_fun` (f01ef) be used as it is more efficient and, in general, more accurate than `nag_matop_real_gen_matrix_fun_num` (f01el).

For any z on the real line, $f(z)$ must be real. f must also be complex analytic on the spectrum of A . These conditions ensure that $f(A)$ is real for real A .

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

If estimates of the condition number of the matrix function are required then `nag_matop_real_gen_matrix_cond_num` (f01jb) should be used.

`nag_matop_complex_gen_matrix_fun_num` (f01fl) can be used to find the matrix function $f(A)$ for a complex matrix A .

9 Example

This example finds $\cos 2A$ where

$$A = \begin{pmatrix} 3 & 0 & 1 & 2 \\ -1 & 1 & 3 & 1 \\ 0 & 2 & 2 & 1 \\ 2 & 1 & -1 & 1 \end{pmatrix}.$$

9.1 Program Text

```
function f01el_example
fprintf('f01el example results\n\n');

a = [3, 0, 1, 2;
     -1, 1, 3, 1;
      0, 2, 2, 1;
      2, 1, -1, 1];

% Compute cos(2a)
[cos2a, user, iflag, imnorm, ifail] = ...
f01el(a, @f);

disp('f(A) = cos(2A)');
disp(cos2a);

fprintf('Imnorm = %6.2f\n', imnorm);

function [iflag, fz, user] = f(iflag, nz, z, user)
    fz = complex(cos(2*z));
```

9.2 Program Results

f01el example results

f(A) = cos(2A)

-0.1704	-1.1597	-0.1878	-0.7307
-0.3950	-0.4410	0.7606	0.0655
-0.0950	-0.0717	0.0619	-0.4351
-0.1034	0.6424	-1.3964	0.1042

Imnorm = 0.00
