NAG Toolbox for MATLAB

e01ra

1 Purpose

e01ra produces, from a set of function values and corresponding abscissae, the coefficients of an interpolating rational function expressed in continued fraction form.

2 Syntax

```
[m, a, u, ifail] = e01ra(x, f, 'n', n)
```

3 Description

e01ra produces the parameters of a rational function $R(x)$ which assumes prescribed values f_i at prescribed values x_i of the independent variable x, for $i = 1, 2, \ldots, n$. More specifically, e01ra determines the parameters a_j, for $j = 1, 2, \ldots, m$ and u_j, $j = 1, 2, \ldots, m - 1$, in the continued fraction

$$R(x) = a_1 + R_m(x) \tag{1}$$

where

$$R_i(x) = \frac{a_{m-i+2}(x - u_{m-i+1})}{1 + R_{i-1}(x)}, \qquad \text{for } i = m, m-1, \ldots, 2,$$

and

$$R_1(x) = 0,$$

such that $R(x_i) = f_i$, for $i = 1, 2, \ldots, n$. The value of m in (1) is determined by the function; normally $m = n$. The values of u_j form a reordered subset of the values of x_i and their ordering is designed to ensure that a representation of the form (1) is determined whenever one exists.

The subsequent evaluation of (1) for given values of x can be carried out using e01rb.

The computational method employed in e01ra is the modification of the Thacher–Tukey algorithm described in Graves–Morris and Hopkins 1981.

4 References

Graves–Morris P R and Hopkins T R 1981 Reliable rational interpolation *Numer. Math.* **36** 111–128

5 Parameters

5.1 Compulsory Input Parameters

1: **x(n) – double array**

x(i) must be set to the value of the ith data abscissa, x_i, for $i = 1, 2, \ldots, n$.

Constraint: the **x**(i) must be distinct.

2: **f(n) – double array**

f(i) must be set to the value of the data ordinate, f_i, corresponding to x_i, for $i = 1, 2, \ldots, n$.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the arrays **x**, **f**, **a**, **u**. (An error is raised if these dimensions are not equal.)

n, the number of data points.

Constraint: $\mathbf{n} > 0$.

5.3 Input Parameters Omitted from the MATLAB Interface

iw

5.4 Output Parameters

1: **m – int32 scalar**

m, the number of terms in the continued fraction representation of $R(x)$.

2: **a(n) – double array**

$\mathbf{a}(j)$ contains the value of the parameter a_j in $R(x)$, for $j = 1, 2, \ldots, m$. The remaining elements of **a**, if any, are set to zero.

3: **u(n) – double array**

$\mathbf{u}(j)$ contains the value of the parameter u_j in $R(x)$, for $j = 1, 2, \ldots, m - 1$. The u_j are a permuted subset of the elements of **x**. The remaining $n - m + 1$ locations contain a permutation of the remaining x_i, which can be ignored.

4: **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 $= 1$

On entry, $\mathbf{n} \leq 0$.

ifail $= 2$

At least one pair of the values $\mathbf{x}(i)$ are equal (or so nearly so that a subsequent division will inevitably cause overflow).

ifail $= 3$

A continued fraction of the required form does not exist.

7 Accuracy

Usually, it is not the accuracy of the coefficients produced by this function which is of prime interest, but rather the accuracy of the value of $R(x)$ that is produced by the associated function e01rb when subsequently it evaluates the continued fraction (1) for a given value of x. This final accuracy will depend mainly on the nature of the interpolation being performed. If interpolation of a ‘well-behaved smooth’ function is attempted (and provided the data adequately represents the function), high accuracy will normally ensue, but, if the function is not so ‘smooth’ or extrapolation is being attempted, high accuracy is much less likely. Indeed, in extreme cases, results can be highly inaccurate.

There is no built-in test of accuracy but several courses are open to you to prevent the production or the acceptance of inaccurate results.

1. If the origin of a variable is well outside the range of its data values, the origin should be shifted to correct this; and, if the new data values are still excessively large or small, scaling to make the largest value of the order of unity is recommended. Thus, normalization to the range -1.0 to $+1.0$ is ideal. This applies particularly to the independent variable; for the dependent variable, the removal of leading figures which are common to all the data values will usually suffice.
2. To check the effect of rounding errors engendered in the functions themselves, e01ra should be re-entered with x_1 interchanged with x_i and f_1 with f_i, $(i \neq 1)$. This will produce a completely different vector a and a reordered vector u, but any change in the value of $R(x)$ subsequently produced by e01rb will be due solely to rounding error.
3. Even if the data consist of calculated values of a formal mathematical function, it is only in exceptional circumstances that bounds for the interpolation error (the difference between the true value of the function underlying the data and the value which would be produced by the two functions if exact arithmetic were used) can be derived that are sufficiently precise to be of practical use. Consequently, you are recommended to rely on comparison checks: if extra data points are available, the calculation may be repeated with one or more data pairs added or exchanged, or alternatively, one of the original data pairs may be omitted. If the algorithms are being used for extrapolation, the calculations should be performed repeatedly with the $2, 3, \ldots$ nearest points until, hopefully, successive values of $R(x)$ for the given x agree to the required accuracy.

8 Further Comments

The time taken by e01ra is approximately proportional to n^2.

The continued fraction (1) when expanded produces a rational function in x, the degree of whose numerator is either equal to or exceeds by unity that of the denominator. Only if this rather special form of interpolatory rational function is needed explicitly, would this function be used without subsequent entry (or entries) to e01rb.

9 Example

```
x = [0;
     1;
     2;
     3;
     4];
f = [4;
     2;
     4;
     7;
     10.4];
[m, a, u, ifail] = e01ra(x, f)
```

```
m =
           4
a =
    4.0000
    1.0000
    0.7500
   -1.0000
         0
u =
     0
     3
     1
     2
     4
ifail =
                0
```