

## NAG C Library Function Document

### nag\_1d\_ratnl\_interp (e01rac)

#### 1 Purpose

nag\_1d\_ratnl\_interp (e01rac) produces, from a set of function values and corresponding abscissae, the coefficients of an interpolating rational function expressed in continued fraction form.

#### 2 Specification

```
void nag_1d_ratnl_interp (Integer n, const double x[], const double f[],
    Integer *m, double a[], double u[], NagError *fail)
```

#### 3 Description

nag\_1d\_ratnl\_interp (e01rac) 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  $i = 1, 2, \dots, n$ . More specifically, nag\_1d\_ratnl\_interp (e01rac) determines the parameters  $a_j$ , for  $j = 1, 2, \dots, m$  and  $u_j$ ,  $j = 1, 2, \dots, m - 1$ , in the continued fraction

$$R(x) = a_1 + R_m(x) \quad (1)$$

where

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

and

$$R_1(x) = 0,$$

such that  $R(x_i) = f_i$ , for  $i = 1, 2, \dots, n$ . The value of  $m$  in (1) is determined by the function; normally  $m = n$ . The values of  $u_j$  form a re-ordered 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 nag\_1d\_ratnl\_eval (e01rbc).

The computational method employed in nag\_1d\_ratnl\_interp (e01rac) 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

- 1: **n** – Integer *Input*  
*On entry:*  $n$ , the number of data points.  
*Constraint:*  $n > 0$ .
- 2: **x[n]** – const double *Input*  
*On entry:*  $x[i - 1]$  must be set to the value of the  $i$ th data abscissa,  $x_i$ , for  $i = 1, 2, \dots, n$ .  
*Constraint:* the  $x[i]$  must be distinct.

- 3: **f[n]** – const double *Input*  
*On entry:* **f**[ $i - 1$ ] must be set to the value of the data ordinate,  $f_i$ , corresponding to  $x_i$ , for  $i = 1, 2, \dots, n$ .
- 4: **m** – Integer \* *Output*  
*On exit:*  $m$ , the number of terms in the continued fraction representation of  $R(x)$ .
- 5: **a[n]** – double *Output*  
*On exit:* **a**[ $j - 1$ ] contains the value of the parameter  $a_j$  in  $R(x)$ , for  $j = 1, 2, \dots, m$ . The remaining elements of **a**, if any, are set to zero.
- 6: **u[n]** – double *Output*  
*On exit:* **u**[ $j - 1$ ] contains the value of the parameter  $u_j$  in  $R(x)$ , for  $j = 1, 2, \dots, m - 1$ . 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.
- 7: **fail** – NagError \* *Input/Output*  
The NAG error parameter (see the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_INT

On entry, **n** =  $\langle value \rangle$ .  
Constraint: **n** > 0.

### NE\_CONT\_FRAC

A continued fraction of the required form does not exist.

### NE\_REAL\_ARRAY

On entry, **x**[ $i - 1$ ] is very close to **x**[ $j - 1$ ]:  $i = \langle value \rangle$ , **x**[ $i - 1$ ] =  $\langle value \rangle$ ,  $j = \langle value \rangle$ , **x**[ $j - 1$ ] =  $\langle value \rangle$ .

### NE\_ALLOC\_FAIL

Memory allocation failed.

### NE\_BAD\_PARAM

On entry, parameter  $\langle value \rangle$  had an illegal value.

### NE\_INTERNAL\_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

## 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 `nag_1d_ratnl_eval` (e01rbc) 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 the user 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, normalisation 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, `nag_1d_ratnl_interp` (e01rac) 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 re-ordered vector  $u$ , but any change in the value of  $R(x)$  subsequently produced by `nag_1d_ratnl_eval` (e01rbc) 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, the user is 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 the function 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 `nag_1d_ratnl_eval` (e01rbc).

## 9 Example

This example program reads in the abscissae and ordinates of 5 data points and prints the parameters  $a_j$  and  $u_j$  of a rational function which interpolates them.

### 9.1 Program Text

```

/* nag_1d_ratnl_interp (e01rac) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nage01.h>

int main(void)
{
    /* Scalars */
    Integer exit_status, i, m, n;
    NagError fail;

    /* Arrays */
    double *a = 0, *f = 0, *u = 0, *x = 0;

    exit_status = 0;
    INIT_FAIL(fail);

```

```

Vprintf("e01rac Example Program Results\n");

/* Skip heading in data file */
Vscanf("%*[\n] ");
n = 5;

/* Allocate memory */
if ( !(a = NAG_ALLOC(n, double)) ||
      !(f = NAG_ALLOC(n, double)) ||
      !(u = NAG_ALLOC(n, double)) ||
      !(x = NAG_ALLOC(n, double)) )
{
  Vprintf("Allocation failure\n");
  exit_status = -1;
  goto END;
}

for (i = 1; i <= n; ++i)
  Vscanf("%lf", &x[i-1]);
Vscanf("%*[\n] ");

for (i = 1; i <= n; ++i)
  Vscanf("%lf", &f[i-1]);
Vscanf("%*[\n] ");

e01rac(n, x, f, &m, a, u, &fail);

Vprintf("\n");
Vprintf("The values of u[j] are\n");
for (i = 1; i <= m - 1; ++i)
{
  Vprintf("%12.4e", u[i-1]);
  Vprintf(i%4 == 0 || i == m - 1 ? "\n" : " ");
}
Vprintf("\n");

Vprintf("The Thiele coefficients a[j] are\n");
for (i = 1; i <= m; ++i)
{
  Vprintf("%12.4e", a[i-1]);
  Vprintf(i%4 == 0 || i == m ? "\n" : " ");
}

END:
if (a) NAG_FREE(a);
if (f) NAG_FREE(f);
if (u) NAG_FREE(u);
if (x) NAG_FREE(x);

return exit_status;
}

```

## 9.2 Program Data

```

e01rac Example Program Data
  0.0    1.0    2.0    3.0    4.0
  4.0    2.0    4.0    7.0   10.4

```

## 9.3 Program Results

e01rac Example Program Results

```

The values of u[j] are
  0.0000e+00  3.0000e+00  1.0000e+00

```

```

The Thiele coefficients a[j] are
  4.0000e+00  1.0000e+00  7.5000e-01 -1.0000e+00

```