NAG Library Function Document

nag_1d_ratnl_eval (e01rbc)

1 Purpose

nag_1d_ratnl_eval (e01rbc) evaluates continued fractions of the form produced by nag_1d_ratnl_interp (e01rac).

2 Specification

```
#include <nag.h>
#include <nage01.h>
void nag_1d_ratnl_eval (Integer m, const double a[], const double u[],
     double x, double *f, NagError *fail)
```

3 Description

nag_1d_ratnl_eval (e01rbc) evaluates the continued fraction

$$R(x) = a_1 + R_m(x)$$

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

$$R_1(x) = 0$$

for a prescribed value of x. nag_1d_ratnl_eval (e01rbc) is intended to be used to evaluate the continued fraction representation (of an interpolatory rational function) produced by nag_1d_ratnl_interp (e01rac).

4 References

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

5 Arguments

1: **m** – Integer *Input*

On entry: m, the number of terms in the continued fraction.

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

2: **a**[**m**] – const double *Input*

On entry: $\mathbf{a}[j-1]$ must be set to the value of the parameter a_j in the continued fraction, for $j = 1, 2, \ldots, m$.

3: **u**[**m**] – const double *Input*

On entry: $\mathbf{u}[j-1]$ must be set to the value of the parameter u_j in the continued fraction, for $j = 1, 2, \ldots, m-1$. (The element $\mathbf{u}[m-1]$ is not used).

4: **x** – double *Input*

On entry: the value of x at which the continued fraction is to be evaluated.

5: **f** – double * *Output*

On exit: the value of the continued fraction corresponding to the value of x.

6: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ 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 contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_POLE_PRESENT

x corresponds to a pole of $R(x)$, or is very close. $\mathbf{x} = \langle value \rangle$.

7 Accuracy

See Section 7 in nag_1d_ratnl_interp (e01rac).

8 Parallelism and Performance

nag_1d_ratnl_eval (e01rbc) is not threaded in any implementation.

9 Further Comments

The time taken by nag_1d_ratnl_eval (e01rbc) is approximately proportional to m.

10 Example

This example reads in the arguments a_j and u_j of a continued fraction (as determined by the example for nag_1d_ratnl_interp (e01rac)) and evaluates the continued fraction at a point x.

10.1 Program Text

```
/* nag_1d_ratnl_eval (e01rbc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

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

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

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

  exit_status = 0;

  INIT_FAIL(fail);

  printf("nag_1d_ratnl_eval (e01rbc) Example Program Results\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  m = 4;

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

  for (i = 1; i <= m; ++i)
#ifdef _WIN32
    scanf_s("%lf", &a[i - 1]);
#else
    scanf("%lf", &a[i - 1]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  for (i = 1; i <= m - 1; ++i)
#ifdef _WIN32
    scanf_s("%lf", &u[i - 1]);
#else
    scanf("%lf", &u[i - 1]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
```

```
#ifdef _WIN32
  scanf_s("%lf%*[^\n] ", &x);
#else
  scanf("%lf%*[^\n] ", &x);
#endif

  printf("\n");
  printf("x = %13.4e\n", x);

  /* nag_1d_ratnl_eval (e01rbc).
   * Interpolated values, evaluate rational interpolant
   * computed by nag_1d_ratnl_interp (e01rac), one variable
   */
  nag_1d_ratnl_eval(m, a, u, x, &f, &fail);
  if (fail.code == NE_NOERROR) {
    printf("\n");
    printf("The value of R(x) is %13.4e\n", f);
  }
  else {
    printf("Error from nag_1d_ratnl_eval (e01rbc).\n%s\n", fail.message);
    exit_status = 1;
  }
END:
  NAG_FREE(a);
  NAG_FREE(u);

  return exit_status;
}
```

10.2 Program Data

```
nag_1d_ratnl_eval (e01rbc) Example Program Data
   4.000   1.000   0.750  -1.000
   0.000   3.000   1.000
   6.000
```

10.3 Program Results

```
nag_1d_ratnl_eval (e01rbc) Example Program Results

x =    6.0000e+00

The value of R(x) is    1.7714e+01
```