1 Purpose

\texttt{nag\_1d\_pade\_eval (e02rbc)} evaluates a rational function at a user-supplied point, given the numerator and denominator coefficients.

2 Specification

\begin{verbatim}
#include <nag.h>
#include <nage02.h>

void nag_1d_pade_eval (const double a[], Integer ia, const double b[],
                        Integer ib, double x, double *ans, NagError *fail)
\end{verbatim}

3 Description

Given a real value $x$ and the coefficients $a_j$, for $j = 0, 1, \ldots, l$ and $b_k$, for $k = 0, 1, \ldots, m$, \texttt{nag\_1d\_pade\_eval (e02rbc)} evaluates the rational function

\[
\frac{\sum_{j=0}^{l} a_j x^j}{\sum_{k=0}^{m} b_k x^k}
\]

using nested multiplication (see Conte and de Boor (1965)).

A particular use of \texttt{nag\_1d\_pade\_eval (e02rbc)} is to compute values of the Padé approximants determined by \texttt{nag\_1d\_pade (e02rac)}.

4 References

Conte S D and de Boor C (1965) \textit{Elementary Numerical Analysis} McGraw–Hill


5 Arguments

1: \(a[\text{ia}]\) – const double

\textit{Input}

\textit{On entry}: \(a[j]\), for \(j = 1, 2, \ldots, l + 1\), must contain the value of the coefficient \(a_j\) in the numerator of the rational function.

2: \(\text{ia}\) – Integer

\textit{Input}

\textit{On entry}: the value of \(l + 1\), where \(l\) is the degree of the numerator.

\textit{Constraint}: \(\text{ia} \geq 1\).

3: \(b[\text{ib}]\) – const double

\textit{Input}

\textit{On entry}: \(b[k]\), for \(k = 1, 2, \ldots, m + 1\), must contain the value of the coefficient \(b_k\) in the denominator of the rational function.

\textit{Constraint}: if \(\text{ib} = 1\), \(b[0] \neq 0.0\).
4:  ib – Integer  
   Input
   
   On entry: the value of $m + 1$, where $m$ is the degree of the denominator.
   
   Constraint: $ib \geq 1$.

5:  x – double  
   Input
   
   On entry: the point $x$ at which the rational function is to be evaluated.

6:  ans – double *  
   Output
   
   On exit: the result of evaluating the rational function at the given point $x$.

7:  fail – NagError *  
   Input/Output
   
   The NAG error argument (see Section 3.6 in the Essential Introduction).

6   Error Indicators and Warnings

NE_ALLOC_FAIL
   Dynamic memory allocation failed.
   See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM
   On entry, argument <value> had an illegal value.

NE_INT
   On entry, $ia = <value>$.
   Constraint: $ia \geq 1$.
   
   On entry, $ib = <value>$.
   Constraint: $ib \geq 1$.

NE_INT_ARRAY
   The first $ib$ entries in b are zero: $ib = <value>$.

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 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE
   Your licence key may have expired or may not have been installed correctly.
   See Section 3.6.5 in the Essential Introduction for further information.

NE_POLE_PRESENT
   Evaluation at or near a pole.

7   Accuracy

A running error analysis for polynomial evaluation by nested multiplication using the recurrence suggested by Kahan (see Peters and Wilkinson (1971)) is used to detect whether you are attempting to evaluate the approximant at or near a pole.
8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken is approximately proportional to $l + m$.

10 Example

This example first calls nag_1d_pade (e02rac) to calculate the $4/4$ Padé approximant to $e^x$, and then uses nag_1d_pade_eval (e02rbc) to evaluate the approximant at $x = 0.1, 0.2, \ldots, 1.0$.

10.1 Program Text

/* nag_1d_pade_eval (e02rbc) Example Program. *
 * Copyright 2014 Numerical Algorithms Group.
 */

#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nage02.h>

int main(void) {
    /* Scalars */
    double ans, tval, x;
    Integer exit_status, i, l, m, ia, ib, ic;
    NagError fail;
    /* Arrays */
    double *aa = 0, *bb = 0, *cc = 0;

    INIT_FAIL(fail);
    exit_status = 0;
    printf("nag_1d_pade_eval (e02rbc) Example Program Results\n");
    l=4;
    m=4;
    ia = 1 + 1;
    ib = m + 1;
    ic = ia + ib - 1;
    /* Allocate memory */
    if (!(aa = NAG_ALLOC(ia, double)) ||
        !(bb = NAG_ALLOC(ib, double)) ||
        !(cc = NAG_ALLOC(ic, double)))
        {
            printf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }
    cc[0] = 1.0;
    for (i = 1; i <= ic - 1; ++i)
        cc[i] = cc[i-1] / (double) i;
    /* nag_1d_pade (e02rac).
     * Pade-approximants */
    nag_1d_pade(ia, ib, cc, aa, bb, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_1d_pade (e02rac).\n\n", fail.message);
    exit_status = 1;
    goto END;
}

printf("\n");
printf(" x  Pade True\n");

for (i = 0; i < 10; ++i)
{
    x = (double)(i + 1) / 10.0;
    /* nag_1d_pade_eval (e02rbc).
     * Evaluation of fitted rational function as computed by
     * nag_1d_pade (e02rac)
     */
    nag_1d_pade_eval(aa, ia, bb, ib, x, &ans, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_1d_pade_eval (e02rbc).\n\n", fail.message);
        exit_status = 1;
        goto END;
    }
    tval = exp(x);
    printf("%6.1f%15.5e%15.5e\n", x, ans, tval);
}

END:
NAG_FREE(aa);
NAG_FREE(bb);
NAG_FREE(cc);

return exit_status;

10.2 Program Data
None.

10.3 Program Results
nag_1d_pade_eval (e02rbc) Example Program Results

<table>
<thead>
<tr>
<th>x</th>
<th>Pade</th>
<th>True</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1.10517e+00</td>
<td>1.10517e+00</td>
</tr>
<tr>
<td>0.2</td>
<td>1.22140e+00</td>
<td>1.22140e+00</td>
</tr>
<tr>
<td>0.3</td>
<td>1.34986e+00</td>
<td>1.34986e+00</td>
</tr>
<tr>
<td>0.4</td>
<td>1.49182e+00</td>
<td>1.49182e+00</td>
</tr>
<tr>
<td>0.5</td>
<td>1.64872e+00</td>
<td>1.64872e+00</td>
</tr>
<tr>
<td>0.6</td>
<td>1.82212e+00</td>
<td>1.82212e+00</td>
</tr>
<tr>
<td>0.7</td>
<td>2.01375e+00</td>
<td>2.01375e+00</td>
</tr>
<tr>
<td>0.8</td>
<td>2.22554e+00</td>
<td>2.22554e+00</td>
</tr>
<tr>
<td>0.9</td>
<td>2.45960e+00</td>
<td>2.45960e+00</td>
</tr>
<tr>
<td>1.0</td>
<td>2.71828e+00</td>
<td>2.71828e+00</td>
</tr>
</tbody>
</table>
Example Program
The [4|4] Pade Approximant of exp(x)

![Graph showing the Pade approximant of exp(x)](image-url)