

NAG C Library Function Document

nag_1d_cheb_deriv (e02ahc)

1 Purpose

nag_1d_cheb_deriv (e02ahc) determines the coefficients in the Chebyshev-series representation of the derivative of a polynomial given in Chebyshev-series form.

2 Specification

```
void nag_1d_cheb_deriv (Integer n, double xmin, double xmax, const double a[],
    Integer ia1, double *patm1, double adif[], Integer iadif1, NagError *fail)
```

3 Description

nag_1d_cheb_deriv (e02ahc) forms the polynomial which is the derivative of a given polynomial. Both the original polynomial and its derivative are represented in Chebyshev-series form. Given the coefficients a_i , for $i = 0, 1, \dots, n$, of a polynomial $p(x)$ of degree n , where

$$p(x) = \frac{1}{2}a_0 + a_1T_1(\bar{x}) + \dots + a_nT_n(\bar{x})$$

the function returns the coefficients \bar{a}_i , for $i = 0, 1, \dots, n-1$, of the polynomial $q(x)$ of degree $n-1$, where

$$q(x) = \frac{dp(x)}{dx} = \frac{1}{2}\bar{a}_0 + \bar{a}_1T_1(\bar{x}) + \dots + \bar{a}_{n-1}T_{n-1}(\bar{x}).$$

Here $T_j(\bar{x})$ denotes the Chebyshev polynomial of the first kind of degree j with argument \bar{x} . It is assumed that the normalised variable \bar{x} in the interval $[-1, +1]$ was obtained from the user's original variable x in the interval $[x_{\min}, x_{\max}]$ by the linear transformation

$$\bar{x} = \frac{2x - (x_{\max} + x_{\min})}{x_{\max} - x_{\min}}$$

and that the user requires the derivative to be with respect to the variable x . If the derivative with respect to \bar{x} is required, set $x_{\max} = 1$ and $x_{\min} = -1$.

Values of the derivative can subsequently be computed, from the coefficients obtained, by using nag_1d_cheb_eval2 (e02akc).

The method employed is that of Chebyshev-series (see Chapter 8 of Modern Computing Methods (1961)), modified to obtain the derivative with respect to x . Initially setting $\bar{a}_{n+1} = \bar{a}_n = 0$, the function forms successively

$$\bar{a}_{i-1} = \bar{a}_{i+1} + \frac{2}{x_{\max} - x_{\min}} 2ia_i, \quad i = n, n-1, \dots, 1.$$

4 References

Modern Computing Methods (1961) Chebyshev-series *NPL Notes on Applied Science* **16** (2nd Edition) HMSO

5 Parameters

1: **n** – Integer *Input*

On entry: n , the degree of the given polynomial $p(x)$.

Constraint: $n \geq 0$.

- 2: **xmin** – double *Input*
 3: **xmax** – double *Input*

On entry: the lower and upper end-points respectively of the interval $[x_{\min}, x_{\max}]$. The Chebyshev-series representation is in terms of the normalised variable \bar{x} , where

$$\bar{x} = \frac{2x - (x_{\max} + x_{\min})}{x_{\max} - x_{\min}}.$$

Constraint: **xmax** > **xmin**.

- 4: **a**[*dim*] – const double *Input*

Note: the dimension, *dim*, of the array **a** must be at least $1 + \mathbf{n} \times \mathbf{ia1}$.

On entry: the Chebyshev coefficients of the polynomial $p(x)$. Specifically, element $i \times \mathbf{ia1}$ of **a** must contain the coefficient a_i , for $i = 0, 1, \dots, n$. Only these $n + 1$ elements will be accessed.

- 5: **ia1** – Integer *Input*

On entry: the index increment of **a**. Most frequently the Chebyshev coefficients are stored in adjacent elements of **a**, and **ia1** must be set to 1. However, if, for example, they are stored in **a**[0], **a**[3], **a**[6], ..., then the value of **ia1** must be 3. See also Section 8.

Constraint: **ia1** ≥ 1 .

- 6: **patm1** – double * *Output*

On exit: the value of $p(x_{\min})$. If this value is passed to the integration routine nag_1d_cheb_intg (e02ajc) with the coefficients of $q(x)$, then the original polynomial $p(x)$ is recovered, including its constant coefficient.

- 7: **adif**[*dim*] – double *Output*

Note: the dimension, *dim*, of the array **adif** must be at least $1 + \mathbf{n} \times \mathbf{iadif1}$.

On exit: the Chebyshev coefficients of the derived polynomial $q(x)$. (The differentiation is with respect to the variable x .) Specifically, element $i \times \mathbf{iadif1}$ of **adif** contains the coefficient \bar{a}_i , $i = 0, 1, \dots, n - 1$. Additionally, element $n \times \mathbf{iadif1}$ is set to zero.

- 8: **iadif1** – Integer *Input*

On entry: the index increment of **adif**. Most frequently the Chebyshev coefficients are required in adjacent elements of **adif**, and **iadif1** must be set to 1. However, if, for example, they are to be stored in **adif**[0], **adif**[3], **adif**[6], ..., then the value of **iadif1** must be 3. See Section 8.

Constraint: **iadif1** ≥ 1 .

- 9: **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 .

On entry, **ia1** = $\langle value \rangle$.

Constraint: **ia1** ≥ 1 .

On entry, **iadif1** = $\langle value \rangle$.

Constraint: **iadif1** ≥ 1 .

NE_REAL_2

On entry, $\mathbf{xmax} \leq \mathbf{xmin}$: $\mathbf{xmax} = \langle value \rangle$, $\mathbf{xmin} = \langle value \rangle$.

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

There is always a loss of precision in numerical differentiation, in this case associated with the multiplication by $2i$ in the formula quoted in Section 3.

8 Further Comments

The time taken is approximately proportional to $n + 1$.

The increments **ia1**, **iadif1** are included as parameters to give a degree of flexibility which, for example, allows a polynomial in two variables to be differentiated with respect to either variable without rearranging the coefficients.

9 Example

Suppose a polynomial has been computed in Chebyshev-series form to fit data over the interval $[-0.5, 2.5]$. The following program evaluates the first and second derivatives of this polynomial at 4 equally spaced points over the interval. (For the purposes of this example, **xmin**, **xmax** and the Chebyshev coefficients are simply supplied. Normally a program would first read in or generate data and compute the fitted polynomial.)

9.1 Program Text

```

/* nag_ld_cheb_deriv (e02ahc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

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

int main(void)
{
    /* Initialized data */
    const double xmin = -0.5;
    const double xmax = 2.5;
    const double a[7] = { 2.53213, 1.13032, 0.2715, 0.04434, 0.00547, 5.4e-4, 4e-5 };

    /* Scalars */
    double deriv, deriv2, patm1, x;
    Integer exit_status, i, m, n, one;
    NagError fail;

    /* Arrays */
    double *adif = 0, *adif2 = 0;

    INIT_FAIL(fail);
    exit_status = 0;

```

```

Vprintf("e02ahc Example Program Results\n");

n = 6;
one = 1;

/* Allocate memory */
if ( !(adif = NAG_ALLOC(n + 1, double)) ||
      !(adif2 = NAG_ALLOC(n + 1, double)) )
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

e02ahc(n, xmin, xmax, a, one, &patm1, adif, one, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from e02ahc call 1.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

e02ahc(n, xmin, xmax, adif, one, &patm1, adif2, one, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from e02ahc call 2.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

m = 4;
Vprintf("\n");
Vprintf("   i  Argument      1st deriv      2nd deriv\n");

for (i = 1; i <= m; ++i)
{
    x = (xmin * (double)(m - i) + xmax * (double)(i - 1)) / (double)(m - 1);
    e02akc(n, xmin, xmax, adif, one, x, &deriv, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from e02akc call 1.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
    e02akc(n, xmin, xmax, adif2, one, x, &deriv2, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from e02akc call 2.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
    Vprintf("%4ld%9.4f      %9.4f      %9.4f      \n", i, x, deriv, deriv2);
}

END:
if (adif) NAG_FREE(adif);
if (adif2) NAG_FREE(adif2);

return exit_status;
}

```

9.2 Program Data

None.

9.3 Program Results

e02ahc Example Program Results

i	Argument	1st deriv	2nd deriv
1	-0.5000	0.2453	0.1637
2	0.5000	0.4777	0.3185
3	1.5000	0.9304	0.6203
4	2.5000	1.8119	1.2056
