# NAG 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

```
#include <nag.h>
#include <nage02.h>
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 $i = 0, 1, \ldots, n$, of a polynomial $p(x)$ of degree $n$, where

$$p(x) = \tfrac{1}{2}a_0 + a_1 T_1(\bar{x}) + \cdots + a_n T_n(\bar{x})$$

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

$$q(x) = \frac{dp(x)}{dx} = \tfrac{1}{2}\bar{a}_0 + \bar{a}_1 T_1(\bar{x}) + \cdots + \bar{a}_{n-1} T_{n-1}(\bar{x}).$$

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 normalized variable $\bar{x}$ in the interval $[-1, +1]$ was obtained from your 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 you require 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

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

## 4 References

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

# 5 Arguments

1: **n** – Integer *Input*

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

*Constraint*: $\mathbf{n} \geq 0$.

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 normalized 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 $(1 + (\mathbf{n} + 1 - 1) \times \mathbf{ia1})$.

*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 $i = 0, 1, \ldots, n$. 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 $\mathbf{a}[0], \mathbf{a}[3], \mathbf{a}[6], \ldots$, then the value of **ia1** must be 3. See also Section 9.

*Constraint*: $\mathbf{ia1} \geq 1$.

6: **patm1** – double * *Output*

*On exit*: the value of $p(x_{\min})$. If this value is passed to the integration function 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 $(1 + (\mathbf{n} + 1 - 1) \times \mathbf{iadif1})$.

*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 $\bar{a}_i$, for $i = 0, 1, \ldots, n - 1$. 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 $\mathbf{adif}[0], \mathbf{adif}[3], \mathbf{adif}[6], \ldots$, then the value of **iadif1** must be 3. See Section 9.

*Constraint*: $\mathbf{iadif1} \geq 1$.

9: **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_INT**

On entry, **ia1** = ⟨*value*⟩.
Constraint: **ia1** $\geq 1$.

On entry, **iadif1** = ⟨*value*⟩.
Constraint: **iadif1** $\geq 1$.

On entry, **n** $+ 1$ = ⟨*value*⟩.
Constraint: **n** $+ 1 \geq 1$.

On entry, **n** = ⟨*value*⟩.
Constraint: **n** $\geq 0$.

**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_REAL_2**

On entry, **xmax** = ⟨*value*⟩ and **xmin** = ⟨*value*⟩.
Constraint: **xmax** $>$ **xmin**.

## 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 Parallelism and Performance

nag_1d_cheb_deriv (e02ahc) is not threaded in any implementation.

## 9 Further Comments

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

The increments **ia1**, **iadif1** are included as arguments 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.

# 10 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.)

## 10.1 Program Text

```
/* nag_1d_cheb_deriv (e02ahc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#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;
  printf("nag_1d_cheb_deriv (e02ahc) Example Program Results\n");

  n = 6;
  one = 1;

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

  /* nag_1d_cheb_deriv (e02ahc).
   * Derivative of fitted polynomial in Chebyshev series form
   */
  nag_1d_cheb_deriv(n, xmin, xmax, a, one, &patm1, adif, one, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_1d_cheb_deriv (e02ahc) call 1.\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  /* nag_1d_cheb_deriv (e02ahc), see above. */
```

```
  nag_1d_cheb_deriv(n, xmin, xmax, adif, one, &patm1, adif2, one, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_1d_cheb_deriv (e02ahc) call 2.\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  m = 4;
  printf("\n");
  printf("   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);
    /* nag_1d_cheb_eval2 (e02akc).
     * Evaluation of fitted polynomial in one variable from
     * Chebyshev series form
     */
    nag_1d_cheb_eval2(n, xmin, xmax, adif, one, x, &deriv, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_1d_cheb_eval2 (e02akc) call 1.\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }
    /* nag_1d_cheb_eval2 (e02akc), see above. */
    nag_1d_cheb_eval2(n, xmin, xmax, adif2, one, x, &deriv2, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_1d_cheb_eval2 (e02akc) call 2.\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }
    printf("%4" NAG_IFMT "%9.4f    %9.4f    %9.4f    \n", i, x, deriv,
           deriv2);
  }

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

  return exit_status;
}
```

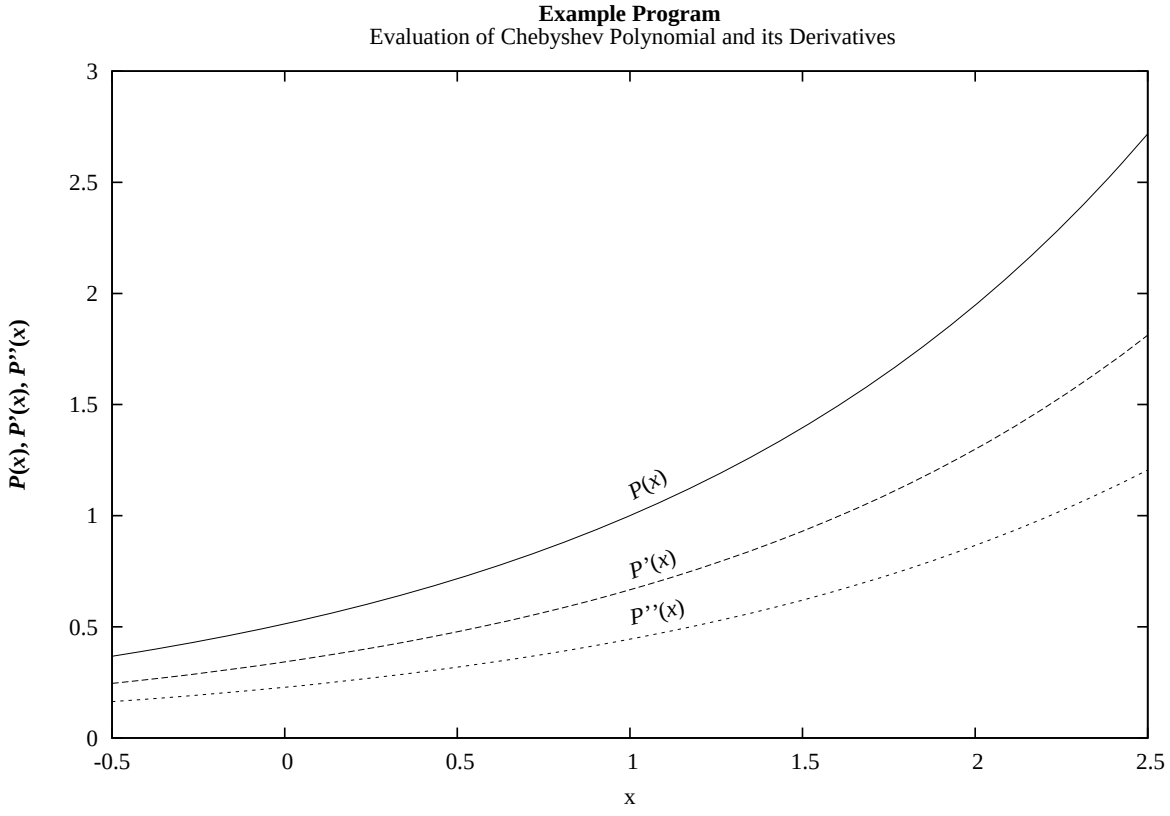
## 10.2 Program Data

None.

## 10.3 Program Results

```
nag_1d_cheb_deriv (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
```


Example Program
Evaluation of Chebyshev Polynomial and its Derivatives
P(x), P'(x), P''(x)
x
P(x)
P'(x)
P''(x)
3
2.5
2
1.5
1
0.5
0
-0.5
0
0.5
1
1.5
2
2.5