NAG Library Function Document

nag_sum_cheby_series (c06dcc)

1 Purpose

nag_sum_cheby_series (c06dcc) evaluates a polynomial from its Chebyshev series representation at a set of points.

2 Specification

```
#include <nag.h>
#include <nagc06.h>
void nag_sum_cheby_series (const double x[], Integer lx, double xmin,
     double xmax, const double c[], Integer n, Nag_Series s, double res[],
     NagError *fail)
```

3 Description

nag_sum_cheby_series (c06dcc) evaluates, at each point in a given set X, the sum of a Chebyshev series of one of three forms according to the value of the parameter **s**:

s = Nag_SeriesGeneral:

$$0.5c_1 + \sum_{j=2}^{n} c_j T_{j-1}(\bar{x})$$

s = Nag_SeriesEven:

$$0.5c_1 + \sum_{j=2}^{n} c_j T_{2j-2}(\bar{x})$$

s = Nag_SeriesOdd:

$$\sum_{j=1}^{n} c_j T_{2j-1}(\bar{x})$$

where $\bar{x}$ lies in the range $-1.0 \leq \bar{x} \leq 1.0$. Here $T_r(x)$ is the Chebyshev polynomial of order r in $\bar{x}$, defined by $\cos(ry)$ where $\cos y = \bar{x}$.

It is assumed that the independent variable $\bar{x}$ in the interval $[-1.0, +1.0]$ was obtained from your original variable $x \in X$, a set of real numbers in the interval $[x_{\min}, x_{\max}]$, by the linear transformation

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

The method used is based upon a three-term recurrence relation; for details see Clenshaw (1962).

The coefficients c_j are normally generated by other functions, for example they may be those returned by the interpolation function nag_1d_cheb_interp (e01aec) (in vector **a**), by a least squares fitting function in Chapter e02, or as the solution of a boundary value problem by nag_ode_bvp_ps_lin_solve (d02uec).

4 References

Clenshaw C W (1962) Chebyshev Series for Mathematical Functions *Mathematical tables* HMSO

5 Arguments

1: **x**[**lx**] – const double *Input*

On entry: $x \in X$, the set of arguments of the series.

Constraint: **xmin** $\leq$ **x**$[i-1] \leq$ **xmax**, for $i = 1, 2, \ldots,$ **lx**.

2: **lx** – Integer *Input*

On entry: the number of evaluation points in X.

Constraint: **lx** ≥ 1.

3: **xmin** – double *Input*

4: **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: **xmin** $<$ **xmax**.

5: **c**[**n**] – const double *Input*

On entry: **c**$[j-1]$ must contain the coefficient c_j of the Chebyshev series, for $j = 1, 2, \ldots, n$.

6: **n** – Integer *Input*

On entry: n, the number of terms in the series.

Constraint: **n** ≥ 1.

7: **s** – Nag_Series *Input*

On entry: determines the series (see Section 3).

s = Nag_SeriesGeneral
The series is general.

s = Nag_SeriesEven
The series is even.

s = Nag_SeriesOdd
The series is odd.

Constraint: **s** = Nag_SeriesGeneral, Nag_SeriesEven or Nag_SeriesOdd.

8: **res**[**lx**] – double *Output*

On exit: the Chebyshev series evaluated at the set of points X.

9: **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, **lx** = ⟨*value*⟩.
Constraint: **lx** ≥ 1.

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 1.

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_REAL_2

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

NE_REAL_3

On entry, element **x**[⟨*value*⟩] = ⟨*value*⟩, **xmin** = ⟨*value*⟩ and **xmax** = ⟨*value*⟩.
Constraint: **xmin** $\leq$ **x**[i] $\leq$ **xmax**, for all i.

7 Accuracy

There may be a loss of significant figures due to cancellation between terms. However, provided that n is not too large, nag_sum_cheby_series (c06dcc) yields results which differ little from the best attainable for the available ***machine precision***.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken increases with n.

10 Example

This example evaluates

$$0.5 + T_1(x) + 0.5T_2(x) + 0.25T_3(x)$$

at the points $X = [0.5, 1.0, -0.2]$.

10.1 Program Text

```
/* nag_sum_cheby_series (c06dcc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */

#include <nag.h>
#include <nag_stdlib.h>
#include <nagc06.h>

int main(void)
{
  /* Scalars */
  Integer    exit_status = 0, i, lx, n;
  double     xmax, xmin;
  /* Arrays */
  char       sstr[30];
  double     *c = 0, *res = 0, *x = 0;
  /* Nag Types */
  Nag_Series s;
  NagError   fail;

  INIT_FAIL(fail);

  printf("nag_sum_cheby_series (c06dcc) Example Program Results\n");
  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%*[^\n]%"NAG_IFMT"%*[^\n]", &n, &lx);
#else
  scanf("%"NAG_IFMT"%*[^\n]%"NAG_IFMT"%*[^\n]", &n, &lx);
#endif

  if (!(c = NAG_ALLOC(n, double)) ||
      !(res = NAG_ALLOC(lx, double)) ||
      !(x = NAG_ALLOC(lx, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

#ifdef _WIN32
  for (i = 0; i < lx; i++) scanf_s("%lf", &x[i]);
#else
  for (i = 0; i < lx; i++) scanf("%lf", &x[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  scanf_s("%lf%lf%*[^\n]", &xmin, &xmax);
#else
  scanf("%lf%lf%*[^\n]", &xmin, &xmax);
#endif
#ifdef _WIN32
  scanf_s("%29s%*[^\n]", sstr, _countof(sstr));
#else
  scanf("%29s%*[^\n]", sstr);
#endif
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
```

```
   */
  s = (Nag_Series) nag_enum_name_to_value(sstr);
#ifdef _WIN32
  for (i = 0; i < n; i++) scanf_s("%lf", &c[i]);
#else
  for (i = 0; i < n; i++) scanf("%lf", &c[i]);
#endif

  /* nag_sum_cheby_series (c06dcc).
   * Evaluates a polynomial from its Chebyshev series representation.
   */
  nag_sum_cheby_series(x,lx,xmin,xmax,c,n,s,res,&fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_sum_cheby_series (c06dcc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  printf("\n%8s%16s\n","x","sum at x");
  for (i = 0; i < lx; i++) printf("%11.4f%13.4f\n", x[i], res[i]);

 END:
  NAG_FREE(c);
  NAG_FREE(res);
  NAG_FREE(x);

  return exit_status;
}
```

10.2 Program Data

```
nag_sum_cheby_series (c06dcc) Example Program Data
   4                      : n,   length of series
   3                      : lx,  number of evaluation points
   0.5  1.0  -0.2         : x[], evaluation points
  -1.0  1.0               : xmin xmax, range for x
Nag_SeriesGeneral         : s,   form of series
   1.0  1.0   0.5  0.25   : c[], the series coefficients
```

10.3 Program Results

```
nag_sum_cheby_series (c06dcc) Example Program Results

       x        sum at x
     0.5000       0.5000
     1.0000       2.2500
    -0.2000      -0.0180
```