NAG Library Function Document

nag_2d_cheb_eval (e02cbc)

1 Purpose

nag_2d_cheb_eval (e02cbc) evaluates a bivariate polynomial from the rectangular array of coefficients in its double Chebyshev series representation.

2 Specification

```
#include <nag.h>
#include <nage02.h>
```

```
void nag_2d_cheb_eval (Integer mfirst, Integer mlast, Integer k, Integer l,
     const double x[], double xmin, double xmax, double y, double ymin,
     double ymax, double ff[], const double a[], NagError *fail)
```

3 Description

This function evaluates a bivariate polynomial (represented in double Chebyshev form) of degree k in one variable, $\bar{x}$, and degree l in the other, $\bar{y}$. The range of both variables is -1 to $+1$. However, these normalized variables will usually have been derived (as when the polynomial has been computed by nag_2d_cheb_fit_lines (e02cac), for example) from your original variables x and y by the transformations

$$\bar{x} = \frac{2x - (x_{\max} + x_{\min})}{(x_{\max} - x_{\min})} \quad \text{and} \quad \bar{y} = \frac{2y - (y_{\max} + y_{\min})}{(y_{\max} - y_{\min})}.$$

(Here $x_{\min}$ and $x_{\max}$ are the ends of the range of x which has been transformed to the range -1 to $+1$ of $\bar{x}$. $y_{\min}$ and $y_{\max}$ are correspondingly for y. See Section 9). For this reason, the function has been designed to accept values of x and y rather than $\bar{x}$ and $\bar{y}$, and so requires values of $x_{\min}$, etc. to be supplied by you. In fact, for the sake of efficiency in appropriate cases, the function evaluates the polynomial for a sequence of values of x, all associated with the same value of y.

The double Chebyshev series can be written as

$$\sum_{i=0}^{k} \sum_{j=0}^{l} a_{ij} T_i(\bar{x}) T_j(\bar{y}),$$

where $T_i(\bar{x})$ is the Chebyshev polynomial of the first kind of degree i and argument $\bar{x}$, and $T_j(\bar{y})$ is similarly defined. However the standard convention, followed in this function, is that coefficients in the above expression which have either i or j zero are written $\frac{1}{2}a_{ij}$, instead of simply a_{ij}, and the coefficient with both i and j zero is written $\frac{1}{4}a_{0,0}$.

The function first forms $c_i = \sum_{j=0}^{l} a_{ij} T_j(\bar{y})$, with $a_{i,0}$ replaced by $\frac{1}{2}a_{i,0}$, for each of $i = 0, 1, \ldots, k$. The value of the double series is then obtained for each value of x, by summing $c_i \times T_i(\bar{x})$, with c_0 replaced by $\frac{1}{2}c_0$, over $i = 0, 1, \ldots, k$. The Clenshaw three term recurrence (see Clenshaw (1955)) with modifications due to Reinsch and Gentleman (1969) is used to form the sums.

4 References

Clenshaw C W (1955) A note on the summation of Chebyshev series *Math. Tables Aids Comput.* **9** 118–120

Gentleman W M (1969) An error analysis of Goertzel's (Watt's) method for computing Fourier coefficients *Comput. J.* **12** 160–165

5 Arguments

1: **mfirst** – Integer *Input*
2: **mlast** – Integer *Input*

On entry: the index of the first and last x value in the array x at which the evaluation is required respectively (see Section 9).

Constraint: $\mathbf{mlast} \geq \mathbf{mfirst}$.

3: **k** – Integer *Input*
4: **l** – Integer *Input*

On entry: the degree k of x and l of y, respectively, in the polynomial.

Constraint: $\mathbf{k} \geq 0$ and $\mathbf{l} \geq 0$.

5: **x**[**mlast**] – const double *Input*

On entry: $\mathbf{x}[i-1]$, for $i = \mathbf{mfirst}, \ldots, \mathbf{mlast}$, must contain the x values at which the evaluation is required.

Constraint: $\mathbf{xmin} \leq \mathbf{x}[i-1] \leq \mathbf{xmax}$, for all i.

6: **xmin** – double *Input*
7: **xmax** – double *Input*

On entry: the lower and upper ends, x_{min} and x_{max}, of the range of the variable x (see Section 3).

The values of **xmin** and **xmax** may depend on the value of y (e.g., when the polynomial has been derived using nag_2d_cheb_fit_lines (e02cac)).

Constraint: $\mathbf{xmax} > \mathbf{xmin}$.

8: **y** – double *Input*

On entry: the value of the y coordinate of all the points at which the evaluation is required.

Constraint: $\mathbf{ymin} \leq \mathbf{y} \leq \mathbf{ymax}$.

9: **ymin** – double *Input*
10: **ymax** – double *Input*

On entry: the lower and upper ends, y_{min} and y_{max}, of the range of the variable y (see Section 3).

Constraint: $\mathbf{ymax} > \mathbf{ymin}$.

11: **ff**[**mlast**] – double *Output*

On exit: $\mathbf{ff}[i-1]$ gives the value of the polynomial at the point (x_i, y), for $i = \mathbf{mfirst}, \ldots, \mathbf{mlast}$.

12: **a**[*dim*] – const double *Input*

Note: the dimension, *dim*, of the array **a** must be at least $((\mathbf{k}+1) \times (\mathbf{l}+1))$.

On entry: the Chebyshev coefficients of the polynomial. The coefficient a_{ij} defined according to the standard convention (see Section 3) must be in $\mathbf{a}[i \times (l+1) + j]$.

13: **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.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT_2

On entry, **k** $=$ ⟨*value*⟩ and **l** $=$ ⟨*value*⟩.
Constraint: **k** ≥ 0 and **l** ≥ 0.

On entry, **mfirst** $=$ ⟨*value*⟩ and **mlast** $=$ ⟨*value*⟩.
Constraint: **mfirst** $\leq$ **mlast**.

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.

Unexpected failure in internal call to nag_1d_cheb_eval (e02aec).

NE_REAL_2

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

On entry, **y** $=$ ⟨*value*⟩ and **ymax** $=$ ⟨*value*⟩.
Constraint: **y** $\leq$ **ymax**.

On entry, **y** $=$ ⟨*value*⟩ and **ymin** $=$ ⟨*value*⟩.
Constraint: **y** $\geq$ **ymin**.

On entry, **ymin** $=$ ⟨*value*⟩ and **ymax** $=$ ⟨*value*⟩.
Constraint: **ymin** $<$ **ymax**.

NE_REAL_ARRAY

On entry, $I =$ ⟨*value*⟩, $\mathbf{x}[I-1] =$ ⟨*value*⟩ and **xmax** $=$ ⟨*value*⟩.
Constraint: $\mathbf{x}[I-1] \leq$ **xmax**.

On entry, $I =$ ⟨*value*⟩, $\mathbf{x}[I-1] =$ ⟨*value*⟩ and **xmin** $=$ ⟨*value*⟩.
Constraint: $\mathbf{x}[I-1] \geq$ **xmin**.

7 Accuracy

The method is numerically stable in the sense that the computed values of the polynomial are exact for a set of coefficients which differ from those supplied by only a modest multiple of ***machine precision***.

8 Parallelism and Performance

nag_2d_cheb_eval (e02cbc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The time taken is approximately proportional to $(k+1) \times (m+l+1)$, where $m = \mathbf{mlast} - \mathbf{mfirst} + 1$, the number of points at which the evaluation is required.

This function is suitable for evaluating the polynomial surface fits produced by the function nag_2d_cheb_fit_lines (e02cac), which provides the array **a** in the required form. For this use, the values of y_{min} and y_{max} supplied to the present function must be the same as those supplied to nag_2d_cheb_fit_lines (e02cac). The same applies to x_{min} and x_{max} if they are independent of y. If they vary with y, their values must be consistent with those supplied to nag_2d_cheb_fit_lines (e02cac) (see Section 9 in nag_2d_cheb_fit_lines (e02cac)).

The arguments **mfirst** and **mlast** are intended to permit the selection of a segment of the array **x** which is to be associated with a particular value of y, when, for example, other segments of **x** are associated with other values of y. Such a case arises when, after using nag_2d_cheb_fit_lines (e02cac) to fit a set of data, you wish to evaluate the resulting polynomial at all the data values. In this case, if the arguments **x**, **y**, **mfirst** and **mlast** of the present function are set respectively (in terms of arguments of nag_2d_cheb_fit_lines (e02cac)) to **x**, $\mathbf{y}(S)$, $1 + \sum_{i=1}^{s-1} \mathbf{m}(i)$ and $\sum_{i=1}^{s} \mathbf{m}(i)$, the function will compute values of the polynomial surface at all data points which have $\mathbf{y}[S-1]$ as their y coordinate (from which values the residuals of the fit may be derived).

10 Example

This example reads data in the following order, using the notation of the argument list above:

N **k** **l**
$\mathbf{a}[i-1]$, for $i = 1, 2, \ldots, (\mathbf{k}+1) \times (\mathbf{l}+1)$
ymin **ymax**
$\mathbf{y}[i-1]$ $M(i-1)$ $\mathbf{xmin}[i-1]$ $\mathbf{xmax}[i-1]$ $\mathrm{X1}(i)$ $\mathrm{XM}(i)$, for $i = 1, 2, \ldots, N$.

For each line $\mathbf{y} = \mathbf{y}[i-1]$ the polynomial is evaluated at $M(i)$ equispaced points between $\mathrm{X1}(i)$ and $\mathrm{XM}(i)$ inclusive.

10.1 Program Text

```
/* nag_2d_cheb_eval (e02cbc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 * Mark 7b revised, 2004.
 */

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

int main(void)
{
  /* Scalars */
  double   x1, xm, xmax, xmin, y, ymax, ymin;
  Integer  exit_status, i, j, k, l, m, n, ncoef, one;
  NagError fail;

  /* Arrays */
  double   *a = 0, *ff = 0, *x = 0;

  INIT_FAIL(fail);

  exit_status = 0;
  printf("nag_2d_cheb_eval (e02cbc) Example Program Results\n");

  /* Skip heading in data file */
```

```
  scanf("%*[^\n] ");
  while (scanf("%ld%ld%ld%*[^\n] ", &n, &k, &l) != EOF)
    {
      /* Allocate array a */
      ncoef = (k + 1) * (l + 1);
      if (!(a = NAG_ALLOC(ncoef, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }

      for (i = 0; i < ncoef; ++i)
        scanf("%lf", &a[i]);
      scanf("%*[^\n] ");
      scanf("%lf%lf%*[^\n] ", &ymin, &ymax);

      for (i = 0; i < n; ++i)
        {
          scanf("%lf%ld%lf%lf%lf%lf%*[^\n] ",
                 &y, &m, &xmin, &xmax, &x1, &xm);

          /* Allocate arrays x and ff */
          if (!(x = NAG_ALLOC(m, double)) ||
              !(ff = NAG_ALLOC(m, double)))
            {
              printf("Allocation failure\n");
              exit_status = -1;
              goto END;
            }

          for (j = 0; j < m; ++j)
            x[j] = x1 + (xm - x1) * (double) j / (double)(m - 1);

          one = 1;
          /* nag_2d_cheb_eval (e02cbc).
           * Evaluation of fitted polynomial in two variables
           */
          nag_2d_cheb_eval(one, m, k, l, x, xmin, xmax, y, ymin, ymax,
                           ff, a, &fail);
          if (fail.code != NE_NOERROR)
            {
              printf("Error from nag_2d_cheb_eval (e02cbc).\n%s\n",
                      fail.message);
              exit_status = 1;
              goto END;
            }

          printf("\n");
          printf("y = %13.4e\n", y);
          printf("\n");
          printf("  i     x(i)      Poly(x(i),y)\n");
          for (j = 0; j < m; ++j)
            printf("%3ld%13.4e%13.4e\n", j, x[j], ff[j]);

          NAG_FREE(ff);
          NAG_FREE(x);
        }

      NAG_FREE(a);
    }

 END:
  NAG_FREE(a);
  NAG_FREE(ff);
  NAG_FREE(x);

  return exit_status;
}
```

10.2 Program Data

```
nag_2d_cheb_eval (e02cbc) Example Program Data
   3   3   2
15.34820
 5.15073
 0.10140
 1.14719
 0.14419
-0.10464
 0.04901
-0.00314
-0.00699
 0.00153
-0.00033
-0.00022
   0.0          4.0
   1.0          9   0.1          4.5          0.5          4.5
   1.5          8   0.225        4.25         0.5          4.0
   2.0          8   0.4          4.0          0.5          4.0
```

10.3 Program Results

```
nag_2d_cheb_eval (e02cbc) Example Program Results

y =    1.0000e+00

  i      x(i)      Poly(x(i),y)
  0   5.0000e-01   2.0812e+00
  1   1.0000e+00   2.1888e+00
  2   1.5000e+00   2.3018e+00
  3   2.0000e+00   2.4204e+00
  4   2.5000e+00   2.5450e+00
  5   3.0000e+00   2.6758e+00
  6   3.5000e+00   2.8131e+00
  7   4.0000e+00   2.9572e+00
  8   4.5000e+00   3.1084e+00

y =    1.5000e+00

  i      x(i)      Poly(x(i),y)
  0   5.0000e-01   2.6211e+00
  1   1.0000e+00   2.7553e+00
  2   1.5000e+00   2.8963e+00
  3   2.0000e+00   3.0444e+00
  4   2.5000e+00   3.2002e+00
  5   3.0000e+00   3.3639e+00
  6   3.5000e+00   3.5359e+00
  7   4.0000e+00   3.7166e+00

y =    2.0000e+00

  i      x(i)      Poly(x(i),y)
  0   5.0000e-01   3.1700e+00
  1   1.0000e+00   3.3315e+00
  2   1.5000e+00   3.5015e+00
  3   2.0000e+00   3.6806e+00
  4   2.5000e+00   3.8692e+00
  5   3.0000e+00   4.0678e+00
  6   3.5000e+00   4.2769e+00
  7   4.0000e+00   4.4971e+00
```

Example Program
Evaluation of Least-squares Bi-variate Polynomial Fit