NAG C 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

```c
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, \(x\), and degree \(l\) in the other, \(y\). The range of both variables is \(-1\) to \(+1\). However, these normalised variables will usually have been derived (as when the polynomial has been computed by `nag_2d_cheb_fit_lines (e02cac)`, for example) from the user’s original variables \(x\) and \(y\) by the transformations

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

(Here \(x_{\text{min}}\) and \(x_{\text{max}}\) are the ends of the range of \(x\) which has been transformed to the range \(-1\) to \(+1\) of \(\bar{x}\). \(y_{\text{min}}\) and \(y_{\text{max}}\) are correspondingly for \(y\). See Section 8). 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 \(x_{\text{min}}, \ldots, x_{\text{max}}\), etc. to be supplied by the user. 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 (Clenshaw (1955)) with modifications due to Reinsch and Gentleman (1969) is used to form the sums.

4 References


5 Parameters

1: \( m\text{first} \) – Integer \( \text{Input} \)
2: \( m\text{last} \) – Integer \( \text{Input} \)

On entry: the index of the first and last \( x \) value in the array \( x \) at which the evaluation is required respectively (see Section 8).

Constraint: \( m\text{last} \geq m\text{first} \).

3: \( k \) – Integer \( \text{Input} \)
4: \( l \) – Integer \( \text{Input} \)

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

Constraint: \( k \geq 0 \) and \( l \geq 0 \).

5: \( x[m\text{last}] \) – const double \( \text{Input} \)

On entry: \( x[i - 1] \), for \( i = m\text{first}, m\text{first} + 1, \ldots, m\text{last} \), must contain the \( x \) values at which the evaluation is required.

Constraint: \( \text{xmin} \leq x[i - 1] \leq \text{xmax} \), for all \( i \).

6: \( \text{xmin} \) – double \( \text{Input} \)
7: \( \text{xmax} \) – double \( \text{Input} \)

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

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

Constraint: \( \text{xmax} > \text{xmin} \).

8: \( y \) – double \( \text{Input} \)

On entry: the value of the \( y \) co-ordinate of all the points at which the evaluation is required.

Constraint: \( \text{ymin} \leq y \leq \text{ymax} \).

9: \( \text{ymin} \) – double \( \text{Input} \)
10: \( \text{ymax} \) – double \( \text{Input} \)

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

Constraint: \( \text{ymax} > \text{ymin} \).

11: \( ff[m\text{last}] \) – double \( \text{Output} \)

On exit: \( ff[i - 1] \) gives the value of the polynomial at the point \( (x_i, y) \), for \( i = m\text{first}, m\text{first} + 1, \ldots, m\text{last} \).

12: \( a[dim] \) – const double \( \text{Input} \)

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

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

13: \( \text{fail} \) – NagError * \( \text{Input/Output} \)

The NAG error parameter (see the Essential Introduction).
6 Error Indicators and Warnings

**NE_INT_2**
On entry, \( k = \langle \text{value} \rangle \), \( l = \langle \text{value} \rangle \).
Constraint: \( k \geq 0 \) and \( l \geq 0 \).
On entry, \( \text{mfirst} > \text{mlast} \): \( \text{mfirst} = \langle \text{value} \rangle \), \( \text{mlast} = \langle \text{value} \rangle \).

**NE_INTERNAL_ERROR**
Unexpected failure in internal call to nag_1d_cheb_eval (e02aec).

**NE_REAL_2**
On entry, \( \text{xmin} \leq \text{xmax} \): \( \text{xmin} = \langle \text{value} \rangle \), \( \text{xmax} = \langle \text{value} \rangle \).
On entry, \( y > \text{ymax} \): \( y = \langle \text{value} \rangle \), \( \text{ymax} = \langle \text{value} \rangle \).
On entry, \( y < \text{ymin} \): \( y = \langle \text{value} \rangle \), \( \text{ymin} = \langle \text{value} \rangle \).
On entry, \( \text{ymin} \leq \text{ymax} \): \( \text{ymin} = \langle \text{value} \rangle \), \( \text{ymax} = \langle \text{value} \rangle \).

**NE_REAL_ARRAY**
On entry, \( x = \langle \text{value} \rangle \) \( \forall i \), \( x = \langle \text{value} \rangle \), \( \text{xmin} = \langle \text{value} \rangle \).
On entry, \( x > \text{xmax} \): \( x = \langle \text{value} \rangle \), \( \text{xmax} = \langle \text{value} \rangle \).

**NE_ALLOC_FAIL**
Memory allocation failed.

**NE_BAD_PARAM**
On entry, parameter \( \langle \text{value} \rangle \) 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

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 Further Comments

The time taken is approximately proportional to \((k + 1) \times (m + l + 1)\), where \(m = \text{mlast} - \text{mfirst} + 1\), 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 \(y_{\text{min}}\) and \(y_{\text{max}}\) supplied to the present function must be the same as those supplied to nag_2d_cheb_fit_lines (e02cac). The same applies to \(x_{\text{min}}\) and \(x_{\text{max}}\) 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 8 of the document for nag_2d_cheb_fit_lines (e02cac)).

The parameters \(\text{mfirst}\) and \(\text{mlast}\) 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, the user wishes to evaluate the resulting polynomial at all the data values. In this case, if the parameters \(x\), \(y\), \(\text{mfirst}\) and \(\text{mlast}\) of the present routine are set respectively (in terms of parameters of
nag_2d_cheb_fit_lines (e02cac)) to \( x, y(s), 1 + \sum_{i=1}^{s-1} m(i) \) and \( \sum_{i=1}^{s} m(i) \), the function will compute values of
the polynomial surface at all data points which have \( y(s-1) \) as their \( y \) co-ordinate (from which values the residuals of the fit may be derived).

9  Example

The example program reads data in the following order, using the notation of the parameter list above:

\[
n, k, l
\]

\[
a[i-1], \quad \text{for } i = 1, 2, \ldots, (k+1) \times (l+1)
\]

\[
ymin, ymax
\]

\[
y[i-1], m(i-1), xmin[i-1], xmax[i-1], x1(i), xm(i), \quad \text{for } i = 1, 2, \ldots, n.
\]

For each line \( y = y[i-1] \) the polynomial is evaluated at \( m(i) \) equispaced points between \( x1(i) \) and \( xm(i) \) inclusive.

9.1  Program Text

/* nag_2d_cheb_eval (e02cbc) 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)
{
    /* Scalars */
    double x1, xm, xmax, xmin, y, ymax, ymin;
    Integer exit_status, i, ifail, j, k, l, m, n, ncoef, one;
    NagError fail;

    /* Arrays */
    double *a = 0, *ff = 0, *x = 0;
    INIT_FAIL(fail);
    exit_status = 0;
    Vprintf("e02cbc Example Program Results\n");

    /* Skip heading in data file */
    Vscanf("%*[\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)))
        {
            Vprintf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }

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

        for (i = 0; i < n; ++i)
        {
            Vscanf("%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)) )
{
    Vprintf("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;
ifail = 0;
e02cbc(one, m, k, l, x, xmin, xmax, y, ymin, ymax,
        ff, a, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from e02cbc.\n", fail.message);
    exit_status = 1;
    goto END;
}

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

if (ff) NAG_FREE(ff);
if (x) NAG_FREE(x);
}

if (a) NAG_FREE(a);

END:
if (a) NAG_FREE(a);
if (ff) NAG_FREE(ff);
if (x) NAG_FREE(x);
return exit_status;
}

9.2 Program Data

e02cbc Example Program Data

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>15.34820</td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>9</td>
<td>5.15073</td>
</tr>
<tr>
<td>2.0</td>
<td>8</td>
<td>0.10140</td>
</tr>
<tr>
<td>1.0</td>
<td>8</td>
<td>1.14719</td>
</tr>
<tr>
<td>1.5</td>
<td>8</td>
<td>0.04901</td>
</tr>
<tr>
<td>2.0</td>
<td>8</td>
<td>-0.10464</td>
</tr>
<tr>
<td>-0.10464</td>
<td>0.10140</td>
<td>0.04901</td>
</tr>
<tr>
<td>-0.00022</td>
<td>4.0</td>
<td>0.04901</td>
</tr>
</tbody>
</table>

[NP3645/7] e02cbc.5
9.3 Program Results

e02cbc Example Program Results

\[ y = 1.0000e+00 \]

\[ \begin{array}{lll}
  i & x(i) & \text{Poly}(x(i),y) \\
  0 & 5.0000e-01 & 2.0812e+00 \\
  1 & 1.0000e+00 & 2.1888e+00 \\
  2 & 1.5000e+00 & 2.3018e+00 \\
  3 & 2.0000e+00 & 2.4204e+00 \\
  4 & 2.5000e+00 & 2.5450e+00 \\
  5 & 3.0000e+00 & 2.6758e+00 \\
  6 & 3.5000e+00 & 2.8131e+00 \\
  7 & 4.0000e+00 & 2.9572e+00 \\
  8 & 4.5000e+00 & 3.1084e+00 \\
\end{array} \]

\[ y = 1.5000e+00 \]

\[ \begin{array}{lll}
  i & x(i) & \text{Poly}(x(i),y) \\
  0 & 5.0000e-01 & 2.6211e+00 \\
  1 & 1.0000e+00 & 2.7553e+00 \\
  2 & 1.5000e+00 & 2.8963e+00 \\
  3 & 2.0000e+00 & 3.0444e+00 \\
  4 & 2.5000e+00 & 3.2002e+00 \\
  5 & 3.0000e+00 & 3.3639e+00 \\
  6 & 3.5000e+00 & 3.5359e+00 \\
  7 & 4.0000e+00 & 3.7166e+00 \\
\end{array} \]

\[ y = 2.0000e+00 \]

\[ \begin{array}{lll}
  i & x(i) & \text{Poly}(x(i),y) \\
  0 & 5.0000e-01 & 3.1700e+00 \\
  1 & 1.0000e+00 & 3.3315e+00 \\
  2 & 1.5000e+00 & 3.5015e+00 \\
  3 & 2.0000e+00 & 3.6806e+00 \\
  4 & 2.5000e+00 & 3.8692e+00 \\
  5 & 3.0000e+00 & 4.0678e+00 \\
  6 & 3.5000e+00 & 4.2769e+00 \\
  7 & 4.0000e+00 & 4.4971e+00 \\
\end{array} \]