NAG Toolbox for MATLAB

e02ca

1 Purpose

e02ca forms an approximation to the weighted, least-squares Chebyshev-series surface fit to data arbitrarily distributed on lines parallel to one independent co-ordinate axis.

2 Syntax

```
[a, ifail] = e02ca(m, k, l, x, y, f, w, xmin, xmax, nux, nuy, 'n', n,
'inuxp1', inuxp1, 'inuyp1', inuyp1)
```

3 Description

This (sub)program determines a bivariate polynomial approximation of degree k in x and l in y to the set of data points $\left(x_{r,s}, y_s, f_{r,s}\right)$, with weights $w_{r,s}$, for $s = 1, 2, \ldots, n$ and $r = 1, 2, \ldots, m_s$. That is, the data points are on lines $y = y_s$, but the x values may be different on each line. The values of k and l are prescribed by you (for guidance on their choice, see Section 8). The (sub)program is based on the method described in Sections 5 and 6 of Clenshaw and Hayes 1965.

The polynomial is represented in double Chebyshev-series form with arguments $\bar{x}$ and $\bar{y}$. The arguments lie in the range -1 to $+1$ and are related to the original variables x and y by the transformations

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

Here y_{max} and y_{min} are set by the (sub)program to, respectively, the largest and smallest value of y_s, but x_{max} and x_{min} are functions of y prescribed by you (see Section 8). For this (sub)program, only their values $x_{\mathrm{max}}^{(s)}$ and $x_{\mathrm{min}}^{(s)}$ at each $y = y_s$ are required. For each $s = 1, 2, \ldots, n$, $x_{\mathrm{max}}^{(s)}$ must not be less than the largest $x_{r,s}$ on the line $y = y_s$, and, similarly, $x_{\mathrm{min}}^{(s)}$ must not be greater than the smallest $x_{r,s}$.

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 with argument $\bar{x}$, and $T_j(y)$ is similarly defined. However, the standard convention, followed in this (sub)program, is that coefficients in the above expression which have either i or j zero are written as $\frac{1}{2}a_{ij}$, instead of simply a_{ij}, and the coefficient with both i and j equal to zero is written as $\frac{1}{4}a_{0,0}$. The series with coefficients output by the (sub)program should be summed using this convention. e02cb is available to compute values of the fitted function from these coefficients.

The (sub)program first obtains Chebyshev-series coefficients $c_{s,i}$, for $i = 0, 1, \ldots, k$, of the weighted least-squares polynomial curve fit of degree k in $\bar{x}$ to the data on each line $y = y_s$, for $s = 1, 2, \ldots, n$ in turn, using an auxiliary (sub)program. The same (sub)program is then called $k + 1$ times to fit $c_{s,i}$, for $s = 1, 2, \ldots, n$ by a polynomial of degree l in $\bar{y}$, for each $i = 0, 1, \ldots, k$. The resulting coefficients are the required a_{ij}.

You can be force the fit to contain a given polynomial factor. This allows for the surface fit to be constrained to have specified values and derivatives along the boundaries $x = x_{\mathrm{min}}$, $x = x_{\mathrm{max}}$, $y = y_{\mathrm{min}}$ and $y = y_{\mathrm{max}}$ or indeed along any lines $\bar{x} =$ constant or $\bar{y} =$ constant (see Section 8 of Clenshaw and Hayes 1965).

4 References

Clenshaw C W and Hayes J G 1965 Curve and surface fitting *J. Inst. Math. Appl.* **1** 164–183

Hayes J G (ed.) 1970 *Numerical Approximation to Functions and Data* Athlone Press, London

5 Parameters

5.1 Compulsory Input Parameters

1: **m(n) – int32 array**

m(s) must be set to m_s, the number of data x values on the line $y = y_s$, for $s = 1, 2, \dots, n$.

Constraint: **m**$(s) > 0$, for $s = 1, 2, \dots, \mathbf{n}$.

2: **k – int32 scalar**

k, the required degree of x in the fit.

Constraint: for $s = 1, 2, \dots, n$, $\mathbf{inuxp1} - 1 \le \mathbf{k} < mdist(s) + \mathbf{inuxp1} - 1$, where $mdist(s)$ is the number of distinct x values with nonzero weight on the line $y = y_s$. See Section 8, paragraph 3.

3: **l – int32 scalar**

l, the required degree of y in the fit.

Constraints:

$\mathbf{l} \ge 0$;
$\mathbf{inuyp1} - 1 \le \mathbf{l} < \mathbf{n} + \mathbf{inuyp1} - 1$.

4: **x(mtot) – double array**

The x values of the data points. The sequence must be

all points on $y = y_1$, followed by

all points on $y = y_2$, followed by

$\vdots$

all points on $y = y_n$.

Constraint: for each y_s, the x values must be in nondecreasing order.

5: **y(n) – double array**

y(s) must contain the y value of line $y = y_s$, for $s = 1, 2, \dots, n$ on which data is given.

Constraint: the y_s values must be in strictly increasing order.

6: **f(mtot) – double array**

f, the data values of the dependent variable in the same sequence as the x values.

7: **w(mtot) – double array**

The weights to be assigned to the data points, in the same sequence as the x values. These weights should be calculated from estimates of the absolute accuracies of the f_r, expressed as standard deviations, probable errors or some other measure which is of the same dimensions as f_r. Specifically, each w_r should be inversely proportional to the accuracy estimate of f_r. Often weights all equal to unity will be satisfactory. If a particular weight is zero, the corresponding data point is omitted from the fit.

8: **xmin(n) – double array**

xmin(s) must contain $x_{\text{min}}^{(s)}$, the lower end of the range of x on the line $y = y_s$, for $s = 1, 2, \ldots, n$. It must not be greater than the lowest data value of x on the line. Each $x_{\text{min}}^{(s)}$ is scaled to -1.0 in the fit. (See also Section 8.)

9: **xmax(n) – double array**

xmax(s) must contain $x_{\text{max}}^{(s)}$, the upper end of the range of x on the line $y = y_s$, for $s = 1, 2, \ldots, n$. It must not be less than the highest data value of x on the line. Each $x_{\text{max}}^{(s)}$ is scaled to $+1.0$ in the fit. (See also Section 8.)

Constraint: **xmax**(s) > **xmin**(s).

10: **nux(inuxp1) – double array**

nux(i) must contain the coefficient of the Chebyshev polynomial of degree $(i - 1)$ in $\bar{x}$, in the Chebyshev-series representation of the polynomial factor in $\bar{x}$ which you require the fit to contain, for $i = 1, 2, \ldots,$ **inuxp1**. These coefficients are defined according to the standard convention of Section 3.

Constraint: **nux**(**inuxp1**) must be nonzero, unless **inuxp1** = 1, in which case **nux** is ignored.

11: **nuy(inuyp1) – double array**

nuy(i) must contain the coefficient of the Chebyshev polynomial of degree $(i - 1)$ in $\bar{y}$, in the Chebyshev-series representation of the polynomial factor which you require the fit to contain, for $i = 1, 2, \ldots,$ **inuyp1**. These coefficients are defined according to the standard convention of Section 3.

Constraint: **nuy**(**inuyp1**) must be nonzero, unless **inuyp1** = 1, in which case **nuy** is ignored.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the arrays **m**, **y**, **xmin**, **xmax**. (An error is raised if these dimensions are not equal.)

the number of lines $y =$ constant on which data points are given.

Constraint: **n** > 0.

2: **inuxp1 – int32 scalar**

Default: The dimension of the array **nux**.

INUX + 1, where INUX is the degree of a polynomial factor in $\bar{x}$ which you require the fit to contain. (See Section 3, last paragraph.)

If this option is not required, **inuxp1** should be set equal to 1.

Constraint: $1 \leq$ **inuxp1** $\leq$ **k** $+ 1$.

3: **inuyp1 – int32 scalar**

Default: The dimension of the array **nuy**.

INUY + 1, where INUY is the degree of a polynomial factor in $\bar{y}$ which you require the fit to contain. (See Section 3, last paragraph.) If this option is not required, **inuyp1** should be set equal to 1.

5.3 Input Parameters Omitted from the MATLAB Interface

mtot, na, work, nwork

5.4 Output Parameters

1: **a(na) – double array**

Contains the Chebyshev coefficients of the fit. $\mathbf{a}(i \times (\mathbf{l}+1)+j)$ is the coefficient a_{ij} of Section 3 defined according to the standard convention. These coefficients are used by e02cb to calculate values of the fitted function.

2: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, **k** or $\mathbf{l} < 0$,
or **inuxp1** or **inuyp1** < 1,
or **inuxp1** $> \mathbf{k}+1$,
or **inuyp1** $> \mathbf{l}+1$,
or $\mathbf{m}(i) < \mathbf{k} - \mathbf{inuxp1} + 2$ for some $i = 1, 2, \ldots, \mathbf{n}$,
or $\mathbf{n} < \mathbf{l} - \mathbf{inuyp1} + 2$,
or **na** is too small,
or **nwork** is too small,
or **mtot** is too small.

ifail $= 2$

$\mathbf{xmin}(i)$ and $\mathbf{xmax}(i)$ do not span the data **x** values on $\mathbf{y} = \mathbf{y}(i)$ for some $i = 1, 2, \ldots, \mathbf{n}$, possibly because $\mathbf{xmin}(i) \geq \mathbf{xmax}(i)$.

ifail $= 3$

The data **x** values on $\mathbf{y} = \mathbf{y}(i)$ are not nondecreasing for some $i = 1, 2, \ldots, \mathbf{n}$, or the $\mathbf{y}(i)$ themselves are not strictly increasing.

ifail $= 4$

The number of distinct **x** values with nonzero weight on $\mathbf{y} = \mathbf{y}(i)$ is less than $\mathbf{k} - \mathbf{inuxp1} + 2$ for some $i = 1, 2, \ldots, \mathbf{n}$.

ifail $= 5$

On entry, $\mathbf{nux}(\mathbf{inuxp1}) = 0$ and $\mathbf{inuxp1} \neq 1$,
or $\mathbf{nuy}(\mathbf{inuyp1}) = 0$ and $\mathbf{inuyp1} \neq 1$.

7 Accuracy

No error analysis for this method has been published. Practical experience with the method, however, is generally extremely satisfactory.

8 Further Comments

The time taken is approximately proportional to $k \times \left(k \times \mathbf{mtot} + n \times l^2\right)$.

The reason for allowing $x_{\max}$ and $x_{\min}$ (which are used to normalize the range of x) to vary with y is that unsatisfactory fits can result if the highest (or lowest) data values of the normalized x on each line $y = y_s$ are not approximately the same. (For an explanation of this phenomenon, see page 176 of Clenshaw and Hayes 1965.) Commonly in practice, the lowest (for example) data values $x_{1,s}$, while not being approximately constant, do lie close to some smooth curve in the (x,y) plane. Using values from this curve as the values of $x_{\min}$, different in general on each line, causes the lowest transformed data values $\bar{x}_{1,s}$ to be approximately constant. Sometimes, appropriate curves for $x_{\max}$ and $x_{\min}$ will be clear from the context of the problem (they need not be polynomials). If this is not the case, suitable curves can often be obtained by fitting to the lowest data values $x_{1,s}$ and to the corresponding highest data values of x, low degree polynomials in y, using function e02ad, and then shifting the two curves outwards by a small amount so that they just contain all the data between them. The complete curves are not in fact supplied to the present (sub)program, only their values at each y_s; and the values simply need to lie on smooth curves. More values on the complete curves will be required subsequently, when computing values of the fitted surface at arbitrary y values.

Naturally, a satisfactory approximation to the surface underlying the data cannot be expected if the character of the surface is not adequately represented by the data. Also, as always with polynomials, the approximating function may exhibit unwanted oscillations (particularly near the ends of the ranges) if the degrees k and l are taken greater than certain values, generally unknown but depending on the total number of coefficients $(k+1) \times (l+1)$ should be significantly smaller than, say not more than half, the total number of data points. Similarly, $k+1$ should be significantly smaller than most (preferably all) the m_s, and $l+1$ significantly smaller than n. Closer spacing of the data near the ends of the x and y ranges is an advantage. In particular, if $\bar{y}_s = -\cos(\pi(s-1)/(n-1))$, for $s = 1, 2, \ldots, n$ and $\bar{x}_{r,s} = -\cos(\pi(r-1)/(m-1))$, for $r = 1, 2, \ldots, m$, (thus $m_s = m$ for all s), then the values $k = m - 1$ and $l = n - 1$ (so that the polynomial passes exactly through all the data points) should not give unwanted oscillations. Other data sets should be similarly satisfactory if they are everywhere at least as closely spaced as the above cosine values with m replaced by $k+1$ and n by $l+1$ (more precisely, if for every s the largest interval between consecutive values of $\arccos \bar{x}_{r,s}$, for $r = 1, 2, \ldots, m$, is not greater than π/k, and similarly for the $\bar{y}_s$). The polynomial obtained should always be examined graphically before acceptance. Note that, for this purpose it is not sufficient to plot the polynomial only at the data values of x and y: intermediate values should also be plotted, preferably via a graphics facility.

Provided the data are adequate, and the surface underlying the data is of a form that can be represented by a polynomial of the chosen degrees, the (sub)program should produce a good approximation to this surface. It is not, however, the true least-squares surface fit nor even a polynomial in x and y, the original variables (see Section 6 of Clenshaw and Hayes 1965,), except in certain special cases. The most important of these is where the data values of x are the same on each line $y = y_s$, (i.e., the data points lie on a rectangular mesh in the (x, y) plane), the weights of the data points are all equal, and $x_{\max}$ and $x_{\min}$ are both constants (in this case they should be set to the largest and smallest data values of x, respectively).

If the data set is such that it can be satisfactorily approximated by a polynomial of degrees k' and l', say, then if higher values are used for k and l in the (sub)program, all the coefficients a_{ij} for $i > k'$ or $j > l'$ will take apparently random values within a range bounded by the size of the data errors, or rather less. (This behaviour of the Chebyshev coefficients, most readily observed if they are set out in a rectangular array, closely parallels that in curve-fitting, examples of which are given in Section 8 of Hayes 1970.) In practice, therefore, to establish suitable values of k' and l', you should first be seeking (within the limitations discussed above) values for k and l which are large enough to exhibit the behaviour described. Values for k' and l' should then be chosen as the smallest which do not exclude any coefficients significantly larger than the random ones. A polynomial of degrees k' and l' should then be fitted to the data.

If the option to force the fit to contain a given polynomial factor in x is used and if zeros of the chosen factor coincide with data x values on any line, then the effective number of data points on that line is reduced by the number of such coincidences. A similar consideration applies when forcing the y-direction. No account is taken of this by the (sub)program when testing that the degrees k and l have not been chosen too large.

9 Example

```
m = [int32(8);
     int32(7);
     int32(7);
     int32(6)];
```

```
k = int32(3);
l = int32(2);
x = [0.1;
     1;
     1.6;
     2.1;
     3.3;
     3.9;
     4.2;
     4.9;
     0.1;
     1.1;
     1.9;
     2.7;
     3.2;
     4.1;
     4.5;
     0.5;
     1.1;
     1.3;
     2.2;
     2.9;
     3.5;
     3.9;
     1.7;
     2;
     2.4;
     2.7;
     3.1;
     3.5 ];
y = [0;
     1;
     2;
     4];
f = [1.01005;
     1.10517;
     1.17351;
     1.23368;
     1.39097;
     1.47698;
     1.52196;
     1.63232;
     2.0201;
     2.23256;
     2.4185;
     2.61993;
     2.75426;
     3.01364;
     3.13662;
     3.15381;
     3.34883;
     3.41649;
     3.73823;
     4.00928;
     4.2572;
     4.43094;
     5.92652;
     6.10701;
     6.35625;
     6.54982;
     6.81713;
     7.09534 ];
w =  ones(28, 1);
xmin = [0;
     0.1;
     0.4;
     1.6];
xmax = [5;
     4.5;
     4;
     3.5];
nux = [0];
nuy = [0];
[a, ifail] = e02ca(m, k, l, x, y, f, w, xmin, xmax, nux, nuy)
```

```
a =
   15.3482
    5.1507
    0.1014
    1.1472
    0.1442
   -0.1046
    0.0490
   -0.0031
   -0.0070
    0.0015
   -0.0003
   -0.0002
ifail =
           0
```