NAG C Library Function Document

nag_2d_cheb_fit_lines (e02cac)

1 Purpose

nag_2d_cheb_fit_lines (e02cac) 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 Specification

```
void nag_2d_cheb_fit_lines (const Integer m[], Integer n, Integer k, Integer l,
     const double x[], const double y[], const double f[], const double w[],
     double a[], const double xmin[], const double xmax[], const double nux[],
     Integer inuxp1, const double nuy[], Integer inuyp1, NagError *fail)
```

3 Description

This function determines a bivariate polynomial approximation of degree k in x and l in y to the set of data points $(x_{r,s}, y_s, f_{r,s})$, 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 the user (for guidance on their choice, see Section 8). The function is based on the method described in Clenshaw and Hayes (1965), Sections 5 and 6.

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

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

Here $y_{\max}$ and $y_{\min}$ are set by the function to, respectively, the largest and smallest value of y_s, but $x_{\max}$ and $x_{\min}$ are functions of y prescribed by the user (see Section 8). For this function, only their values $x_{\max}^{(s)}$ and $x_{\min}^{(s)}$ at each $y = y_s$ are required. For each $s = 1, 2, \ldots, n$, $x_{\max}^{(s)}$ must not be less than the largest $x_{r,s}$ on the line $y = y_s$, and, similarly, $x_{\min}^{(s)}$ 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 function, 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 function should be summed using this convention. nag_2d_cheb_eval (e02cbc) is available to compute values of the fitted function from these coefficients.

The function 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 function. The same function 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}.

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

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

1: **m**[**n**] – const Integer *Input*

On entry: **m**$[s-1]$ must be set to m_s, the number of data x values on the line $y = y_s$, for $s = 1, 2, \ldots, n$.

Constraint: **m**$[s-1] > 0$ for $s = 1, 2, \ldots,$ **n**.

2: **n** – Integer *Input*

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

Constraint: **n** > 0.

3: **k** – Integer *Input*

On entry: k, the required degree of x in the fit.

Constraint: for $s = 1, 2, \ldots, n$, **inuxp1** $- 1 \leq$ **k** $<$ $mdist(s)$ + **inuxp1** $- 1$, where $mdist(s)$ is the number of distinct x values with non-zero weight on the line $y = y_s$. See Section 8, paragraph 3.

4: **l** – Integer *Input*

On entry: l, the required degree of y in the fit.

Constraints:

l ≥ 0;
inuyp1 $- 1 \leq$ **l** $<$ **n** + **inuyp1** $- 1$.

5: **x**$[dim]$ – const double *Input*

Note: the dimension, dim, of the array **x** must be at least $\sum_{s=1}^{\mathbf{n}}$ **m**$[s-1]$.

On entry: 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 non-decreasing order.

6: **y**[**n**] – const double *Input*

On entry: **y**$[s-1]$ must contain the y value of line $y = y_s$, for $s = 1, 2, \ldots, n$ on which data is given.

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

7: **f**$[dim]$ – const double *Input*

Note: the dimension, dim, of the array **f** must be at least $\sum_{s=1}^{\mathbf{n}}$ **m**$[s-1]$.

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

8: **w**$[dim]$ – const double *Input*

Note: the dimension, dim, of the array **w** must be at least $\sum_{s=1}^{\mathbf{n}}$ **m**$[s-1]$.

On entry: 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.

9: **a**[dim] – double *Output*

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

On exit: **a** contains the Chebyshev coefficients of the fit. **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 nag_2d_cheb_eval (e02cbc) to calculate values of the fitted function.

10: **xmin**[**n**] – const double *Input*

On entry: **xmin**$[s-1]$ must contain $x_{\min}^{(s)}$, 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 $x_{\min}^{(s)}$ is scaled to -1.0 in the fit. (See also Section 8.)

11: **xmax**[**n**] – const double *Input*

On entry: **xmax**$[s-1]$ must contain $x_{\max}^{(s)}$, 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 $x_{\max}^{(s)}$ is scaled to $+1.0$ in the fit. (See also Section 8.)

Constraint: **xmax**$[s] >$ **xmin**$[s]$.

12: **nux**[**inuxp1**] – const double *Input*

On entry: **nux**$[i-1]$ 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 the user requires 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** -1] must be non-zero, unless **inuxp1** $= 1$, in which case **nux** is ignored.

13: **inuxp1** – Integer *Input*

On entry: $inux + 1$, where $inux$ is the degree of a polynomial factor in $\bar{x}$ which the user requires 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** $\leq \mathbf{k}+1$.

14: **nuy**[**inuyp1**] – const double *Input*

On entry: **nuy**$[i-1]$ 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 the user requires 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** -1] must be non-zero, unless **inuyp1** $= 1$, in which case **nuy** is ignored.

15: **inuyp1** – Integer *Input*

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

16: **fail** – NagError * *Input/Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} > 0$.

On entry, **inuyp1** $= \langle value \rangle$.
Constraint: **inuyp1** ≥ 1.

On entry, **inuxp1** $= \langle value \rangle$.
Constraint: **inuxp1** ≥ 1.

On entry, $\mathbf{l} = \langle value \rangle$.
Constraint: $\mathbf{l} \geq 0$.

On entry, $\mathbf{k} = \langle value \rangle$.
Constraint: $\mathbf{k} \geq 0$.

NE_INT_2

On entry, **inuyp1** $> \mathbf{l} + 1$: **inuyp1** $= \langle value \rangle$, $\mathbf{l} = \langle value \rangle$.

On entry, **inuxp1** $> \mathbf{k} + 1$: **inuxp1** $= \langle value \rangle$, $\mathbf{k} = \langle value \rangle$.

NE_INT_3

On entry, $\mathbf{n} = \langle value \rangle$, $\mathbf{l} = \langle value \rangle$, **inuyp1** $= \langle value \rangle$.
Constraint: $\mathbf{l} \geq 0$ and **inuyp1** $- 1 \leq \mathbf{l} < \mathbf{n} +$ **inuyp1** $- 1$.

On entry, $\mathbf{n} <$ **inuyp1** $- \mathbf{l} + 2$: $\mathbf{n} = \langle value \rangle$, $\mathbf{l} = \langle value \rangle$, **inuyp1** $= \langle value \rangle$.

NE_INT_ARRAY

On entry, $\mathbf{m}[s-1] = \langle value \rangle$.
Constraint: $\mathbf{m}[s-1] > 0$ for $s = 1, \ldots, \mathbf{n}$.

On entry, $\mathbf{m}[i-1] <$ **inuxp1** $- \mathbf{k} + 2$: $i = \langle value \rangle$, $\mathbf{m}[i-1] = \langle value \rangle$ $\mathbf{k} = \langle value \rangle$, **inuxp1** $= \langle value \rangle$.

On entry, **nux**[**inuxp1** $- 1] = 0.0$ and **inuxp1** is not equal to 1, or **nuy**[**inuyp1** $- 1] = 0.0$ and **inuyp1** is not equal to 1: **inuxp1** $= \langle value \rangle$, **nux**[**inuxp1** $- 1] = \langle value \rangle$ **inuyp1** $= \langle value \rangle$, **nuy**[**inuyp1** $- 1] = \langle value \rangle$.

NE_NON_ZERO_WEIGHTS

On entry, the number of distinct **x** values with non-zero weight on $\mathbf{y} = \mathbf{y}[i-1]$ is less than **k** $-$ **inuxp1** $+ 2$: $i = \langle value \rangle$, $\mathbf{y}[i-1] = \langle value \rangle$ $\mathbf{k} = \langle value \rangle$, **inuxp1** $= \langle value \rangle$.

NE_NOT_NON_DECREASING

On entry, the data **x** values are not non-decreasing for $\mathbf{y} = \mathbf{y}[i-1]$: $i = \langle value \rangle$, $\mathbf{y}[i-1] = \langle value \rangle$.

NE_NOT_STRICTLY_INCREASING

On entry, $\mathbf{y}[i-1] \leq \mathbf{y}[i-2]$: $i = \langle value \rangle$, $\mathbf{y}[i-1] = \langle value \rangle$, $\mathbf{y}[i-2] = \langle value \rangle$.

NE_REAL_ARRAY

On entry, **xmin**$[i-1]$ and **xmax**$[i-1]$ do not span the data **x** values on $\mathbf{y} = \mathbf{y}[i-1]$: $i = \langle value \rangle$, **xmin**$[i-1] = \langle value \rangle$, **xmax**$[i-1] = \langle value \rangle$ $\mathbf{y}[i-1] = \langle value \rangle$.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter ⟨*value*⟩ 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

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 $k \times (k \times \sum_{s=1}^{\mathbf{n}} \mathbf{m}[s-1] + n \times l^2)$.

The reason for allowing $x_{\max}$ and $x_{\min}$ (which are used to normalise the range of x) to vary with y is that unsatisfactory fits can result if the highest (or lowest) data values of the normalised 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 outine nag_1d_cheb_fit (e02adc), 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 function, 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 function 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 Clenshaw and Hayes (1965), Section 6), 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 function, 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 Hayes (1970), Section 8.) In practice, therefore, to establish suitable values of k' and l', the user 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 function when testing that the degrees k and l have not been chosen too large.

9 Example

The example program reads data in the following order, using the notation of the parameter list for nag_2d_cheb_fit_lines (e02cac) above:

n **k** **l**
$\mathbf{y}[i-1]$ $\mathbf{m}[i-1]$ $\mathbf{xmin}[i-1]$ $\mathbf{xmax}[i-1]$, for $i = 1, 2, \ldots, \mathbf{n}$
$\mathbf{x}[i-1]$ $\mathbf{f}[i-1]$ $\mathbf{w}[i-1]$, for $i = 1, 2, \ldots, \sum_{s=1}^{\mathbf{n}} \mathbf{m}[s-1]$.

The data points are fitted using nag_2d_cheb_fit_lines (e02cac), and then the fitting polynomial is evaluated at the data points using nag_2d_cheb_eval (e02cbc).

The output is:

the data points and their fitted values;

the Chebyshev coefficients of the fit.

9.1 Program Text

```
/* nag_2d_cheb_fit_lines (e02cac) 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 ymax;
  Integer exit_status, i, j, k, l, mi, mj, n, r, t,
          na, one;
  NagError fail;

  /* Arrays */
  double *a = 0, *f = 0, *ff = 0, *w = 0,
         *x = 0, *xmax = 0, *xmin = 0, *y = 0;
  Integer *m = 0;

  INIT_FAIL(fail);
  exit_status = 0;
  Vprintf("e02cac Example Program Results\n");

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

```
  Vscanf("%*[^\n] ");

  /* Input the number of lines Y = Y(I) on which data is given, */
  /* and the required degree of fit in the X and Y directions */
  while (scanf("%ld%ld%ld%*[^\n] ", &n, &k, &l) != EOF)
    {
      Vprintf("\n");
      if (n > 0)
        {
          /* Allocate arrays m, y, xmin and xmax */
          if ( !(m = NAG_ALLOC(n, Integer)) ||
               !(y = NAG_ALLOC(n, double)) ||
               !(xmin = NAG_ALLOC(n, double)) ||
               !(xmax = NAG_ALLOC(n, double)))
            {
              Vprintf("Allocation failure\n");
              exit_status = -1;
              goto END;
            }

          mj = 0;
          /* Input Y(I), the number of data points on Y = Y(I) and the */
          /* range of X-values on this line, for I = 1,2,...N */
          for (i = 0; i < n; ++i)
            {
              Vscanf("%lf%ld%lf%lf%*[^\n] ", &y[i], &mi, &xmin[i], &xmax[i]);
              m[i] = mi;
              mj += mi;
            }

          /* Allocate arrays x, f, ff, w and a */
          na = (k + 1) * (l + 1);
          if ( !(x = NAG_ALLOC(mj, double)) ||
               !(f = NAG_ALLOC(mj, double)) ||
               !(ff = NAG_ALLOC(mj, double)) ||
               !(w = NAG_ALLOC(mj, double)) ||
               !(a = NAG_ALLOC(na, double)) )
            {
              Vprintf("Allocation failure\n");
              exit_status = -1;
              goto END;
            }

          /* Input the X-values and function values, F, together with */
          /* their weights, W. */
          for (i = 0; i < mj; ++i)
            Vscanf("%lf%lf%lf", &x[i], &f[i], &w[i]);
          Vscanf("%*[^\n] ");

          /* Evaluate the coefficients, A, of the fit to this set of data */
          one = 1;
          e02cac(m, n, k, l, x, y, f, w, a, xmin, xmax, y,  one, y, one, &fail);
          if (fail.code != NE_NOERROR)
            {
              Vprintf("Error from e02cac.\n%s\n", fail.message);
              exit_status = 1;
              goto END;
            }

          Vprintf("     Data Y     Data X     Data F   Fitted F   Residual\n");
          Vprintf("\n");

          mi = 0;
          for (r = 1; r <= n; ++r)
            {
              t = mi + 1;
              mi += m[r-1];
              ymax = y[n-1];
              if (n == 1)
                ymax += 1.0;
```

```
              /* Evaluate the fitted polynomial at each of the data points */
              /* on the line Y = Y(R) */
             e02cbc(t, mi, k, l, x, xmin[r-1], xmax[r-1],
                    y[r-1], y[0], ymax, ff, a, &fail);
             if (fail.code != NE_NOERROR)
               {
                 Vprintf("Error from e02cbc.\n%s\n", fail.message);
                 exit_status = 1;
                 goto END;
               }
             /* Output the data and fitted values on the line Y = Y(R) */
              for (i = t-1; i < mi; ++i)
                {
                  Vprintf("%11.4f%11.4f%11.4f%11.4f",
                          y[r-1], x[i], f[i], ff[i]);
                  Vprintf("%11.2e\n", ff[i] - f[i]);
                }
              Vprintf("\n");
            }

          /* Output the Chebyshev coefficients of the fit */
          Vprintf("Chebyshev coefficients of the fit\n");
          Vprintf("\n");

          for (j = 1; j <= k + 1; ++j)
            {
              for (i = (j - 1) * (l + 1); i < j * (l + 1); ++i)
                Vprintf("%11.4f ", a[i]);
              Vprintf("\n");
            }
        }
    }

 END:
  if (a) NAG_FREE(a);
  if (f) NAG_FREE(f);
  if (ff) NAG_FREE(ff);
  if (w) NAG_FREE(w);
  if (x) NAG_FREE(x);
  if (xmax) NAG_FREE(xmax);
  if (xmin) NAG_FREE(xmin);
  if (y) NAG_FREE(y);
  if (m) NAG_FREE(m);

  return exit_status;
}
```

9.2 Program Data

```
e02cac Example Program Data
    4    3    2
          0.0      8          0.0            5.0
          1.0      7          0.1            4.5
          2.0      7          0.4            4.0
          4.0      6          1.6            3.5
          0.1        1.01005          1.0
          1.0        1.10517          1.0
          1.6        1.17351          1.0
          2.1        1.23368          1.0
          3.3        1.39097          1.0
          3.9        1.47698          1.0
          4.2        1.52196          1.0
          4.9        1.63232          1.0
          0.1        2.02010          1.0
          1.1        2.23256          1.0
          1.9        2.41850          1.0
          2.7        2.61993          1.0
          3.2        2.75426          1.0
          4.1        3.01364          1.0
          4.5        3.13662          1.0
```

```
          0.5         3.15381          1.0
          1.1         3.34883          1.0
          1.3         3.41649          1.0
          2.2         3.73823          1.0
          2.9         4.00928          1.0
          3.5         4.25720          1.0
          3.9         4.43094          1.0
          1.7         5.92652          1.0
          2.0         6.10701          1.0
          2.4         6.35625          1.0
          2.7         6.54982          1.0
          3.1         6.81713          1.0
          3.5         7.09534          1.0
```

9.3 Program Results

```
e02cac Example Program Results

     Data Y     Data X     Data F   Fitted F   Residual

     0.0000     0.1000     1.0100     1.0175   7.40e-03
     0.0000     1.0000     1.1052     1.1126   7.39e-03
     0.0000     1.6000     1.1735     1.1809   7.43e-03
     0.0000     2.1000     1.2337     1.2412   7.55e-03
     0.0000     3.3000     1.3910     1.3992   8.19e-03
     0.0000     3.9000     1.4770     1.4857   8.72e-03
     0.0000     4.2000     1.5220     1.5310   9.03e-03
     0.0000     4.9000     1.6323     1.6422   9.83e-03

     1.0000     0.1000     2.0201     1.9987  -2.14e-02
     1.0000     1.1000     2.2326     2.2110  -2.16e-02
     1.0000     1.9000     2.4185     2.3962  -2.23e-02
     1.0000     2.7000     2.6199     2.5966  -2.34e-02
     1.0000     3.2000     2.7543     2.7299  -2.43e-02
     1.0000     4.1000     3.0136     2.9869  -2.68e-02
     1.0000     4.5000     3.1366     3.1084  -2.82e-02

     2.0000     0.5000     3.1538     3.1700   1.62e-02
     2.0000     1.1000     3.3488     3.3648   1.60e-02
     2.0000     1.3000     3.4165     3.4325   1.60e-02
     2.0000     2.2000     3.7382     3.7549   1.66e-02
     2.0000     2.9000     4.0093     4.0272   1.79e-02
     2.0000     3.5000     4.2572     4.2769   1.97e-02
     2.0000     3.9000     4.4309     4.4521   2.12e-02

     4.0000     1.7000     5.9265     5.9231  -3.42e-03
     4.0000     2.0000     6.1070     6.1036  -3.41e-03
     4.0000     2.4000     6.3563     6.3527  -3.50e-03
     4.0000     2.7000     6.5498     6.5462  -3.64e-03
     4.0000     3.1000     6.8171     6.8132  -3.98e-03
     4.0000     3.5000     7.0953     7.0909  -4.49e-03

Chebyshev coefficients of the fit

    15.3482      5.1507      0.1014
     1.1472      0.1442     -0.1046
     0.0490     -0.0031     -0.0070
     0.0015     -0.0003     -0.0002
```