NAG Library Routine Document

E02CAF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

E02CAF forms an approximation to the weighted, least squares Chebyshev series surface fit to data arbitrarily distributed on lines parallel to one independent coordinate axis.

2 Specification

```
SUBROUTINE E02CAF (M, N, K, L, X, Y, F, W, MTOT, A, NA, XMIN, XMAX, NUX,    &
                   INUXP1, NUY, INUYP1, WORK, NWORK, IFAIL)
INTEGER            M(N), N, K, L, MTOT, NA, INUXP1, INUYP1, NWORK,          &
                   IFAIL
REAL (KIND=nag_wp) X(MTOT), Y(N), F(MTOT), W(MTOT), A(NA), XMIN(N),         &
                   XMAX(N), NUX(INUXP1), NUY(INUYP1), WORK(NWORK)
```

3 Description

E02CAF 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 9). The subroutine 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

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

Here $y_{\max}$ and $y_{\min}$ are set by the subroutine 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 9). For this subroutine, 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 subroutine, 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 subroutine should be summed using this convention. E02CBF is available to compute values of the fitted function from these coefficients.

The subroutine 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 subroutine. The same subroutine 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 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 $x = x_{\min}$, $x = x_{\max}$, $y = y_{\min}$ and $y = y_{\max}$ 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 Arguments

1: M(N) – INTEGER array *Input*

On entry: M(s) 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) > 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 nonzero weight on the line $y = y_s$. See Section 9.

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(MTOT) – REAL (KIND=nag_wp) array *Input*

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 nondecreasing order.

6: Y(N) – REAL (KIND=nag_wp) array *Input*

On entry: Y(s) 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(MTOT) – REAL (KIND=nag_wp) array *Input*

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

8: W(MTOT) – REAL (KIND=nag_wp) array *Input*

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: MTOT – INTEGER *Input*

On entry: the dimension of the arrays X, F and W as declared in the (sub)program from which E02CAF is called.

Constraint: MTOT $\geq \sum_{s=1}^{\mathrm{N}} \mathrm{M}(s)$.

10: A(NA) – REAL (KIND=nag_wp) array *Output*

On exit: contains the Chebyshev coefficients of the fit. A$(i \times (\mathrm{L}+1)+j)$ is the coefficient a_{ij} of Section 3 defined according to the standard convention. These coefficients are used by E02CBF to calculate values of the fitted function.

11: NA – INTEGER *Input*

On entry: the dimension of the array A as declared in the (sub)program from which E02CAF is called.

Constraint: NA $\geq (\mathrm{K}+1) \times (\mathrm{L}+1)$, the total number of coefficients in the fit.

12: XMIN(N) – REAL (KIND=nag_wp) array *Input*

On entry: XMIN(s) 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 9.)

13: XMAX(N) – REAL (KIND=nag_wp) array *Input*

On entry: XMAX(s) 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 9.)

Constraint: XMAX$(s) >$ XMIN(s).

14: NUX(INUXP1) – REAL (KIND=nag_wp) array *Input*

On entry: 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.

15: INUXP1 – INTEGER *Input*

On entry: $INUX + 1$, 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 $\leq \mathrm{K}+1$.

16: NUY(INUYP1) – REAL (KIND=nag_wp) array *Input*

On entry: 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.

17: INUYP1 – INTEGER *Input*

On entry: $INUY + 1$, 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.

18: WORK(NWORK) – REAL (KIND=nag_wp) array *Workspace*

19: NWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which E02CAF is called.

Constraint: NWORK $\geq 3 \times$ MTOT $+ 2 \times$ N $\times$ (K $+ 2) + 5 \times (1 +$ max(K, L)).

20: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, K or L < 0,
or INUXP1 or INUYP1 < 1,
or INUXP1 $>$ K $+ 1$,
or INUYP1 $>$ L $+ 1$,
or M(i) $<$ K $-$ INUXP1 $+ 2$ for some $i = 1, 2, \ldots,$ N,
or N $<$ L $-$ INUYP1 $+ 2$,
or NA is too small,
or NWORK is too small,
or MTOT is too small.

IFAIL $= 2$

XMIN(i) and XMAX(i) do not span the data X values on Y $=$ Y(i) for some $i = 1, 2, \ldots,$ N, possibly because XMIN(i) $\geq$ XMAX(i).

IFAIL $= 3$

The data X values on $\mathrm{Y} = \mathrm{Y}(i)$ are not nondecreasing for some $i = 1, 2, \ldots, \mathrm{N}$, or the $\mathrm{Y}(i)$ themselves are not strictly increasing.

IFAIL $= 4$

The number of distinct X values with nonzero weight on $\mathrm{Y} = \mathrm{Y}(i)$ is less than $\mathrm{K} - \mathrm{INUXP1} + 2$ for some $i = 1, 2, \ldots, \mathrm{N}$.

IFAIL $= 5$

On entry, $\mathrm{NUX}(\mathrm{INUXP1}) = 0.0$ and $\mathrm{INUXP1} \neq 1$,
or $\mathrm{NUY}(\mathrm{INUYP1}) = 0.0$ and $\mathrm{INUYP1} \neq 1$.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

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

8 Parallelism and Performance

E02CAF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

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

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 routine E02ADF, 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 subroutine, 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 datasets 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 subroutine 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 dataset 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 subroutine, 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 subroutine when testing that the degrees k and l have not been chosen too large.

10 Example

This example reads data in the following order, using the notation of the argument list for E02CAF above:

N	K	L		
Y(i)	M(i)	XMIN(i)	XMAX(i),	for $i = 1, 2, \ldots,$ N
X(i)	F(i)	W(i),		for $i = 1, 2, \ldots,$ MTOT.

The data points are fitted using E02CAF, and then the fitting polynomial is evaluated at the data points using E02CBF.

The output is:

the data points and their fitted values;

the Chebyshev coefficients of the fit.

10.1 Program Text

```
    Program e02cafe

!     E02CAF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: e02caf, e02cbf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: ymax
      Integer                          :: i, ifail, inuxp1, inuyp1, j, k, l,   &
                                          mi, mtot, n, na, nwork, r, t
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:), f(:), ff(:), nux(:), nuy(:),   &
                                          w(:), work(:), x(:), xmax(:),        &
                                          xmin(:), y(:)
      Integer, Allocatable             :: m(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max, sum
!     .. Executable Statements ..
      Write (nout,*) 'E02CAF Example Program Results'

!     Skip heading in data file
      Read (nin,*)

!     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

      Read (nin,*) n, k, l
      inuxp1 = 1
      inuyp1 = 1
      na = (k+1)*(l+1)
      Allocate (a(na),m(n),y(n),xmin(n),xmax(n),nux(inuxp1),nuy(inuyp1))

!     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

      Do i = 1, n
        Read (nin,*) y(i), m(i), xmin(i), xmax(i)
      End Do

      mtot = sum(m(1:n))
      nwork = 3*mtot + 2*n*(k+2) + 5*(1+max(k,l))
      Allocate (x(mtot),f(mtot),w(mtot),ff(mtot),work(nwork))

!     Input the X-values and function values, F, together with
!     their weights, W.

      Read (nin,*)(x(i),f(i),w(i),i=1,mtot)

!     Evaluate the coefficients, A, of the fit to this set of data

      ifail = 0
      Call e02caf(m,n,k,l,x,y,f,w,mtot,a,na,xmin,xmax,nux,inuxp1,nuy,inuyp1,   &
        work,nwork,ifail)

      mi = 0

      Write (nout,*)
      Write (nout,*) '     Data Y     Data X     Data F   Fitted F   Residual'
```

```
      Write (nout,*)

      Do r = 1, n
        t = mi + 1
        mi = mi + m(r)
        ymax = y(n)

        If (n==1) Then
          ymax = ymax + 1.0E0_nag_wp
        End If

!       Evaluate the fitted polynomial at each of the data points
!       on the line Y = Y(R)

        ifail = 0
        Call e02cbf(t,mi,k,l,x,xmin(r),xmax(r),y(r),y(1),ymax,ff,a,na,work,    &
          nwork,ifail)

!       Output the data and fitted values on the line Y = Y(R)

        Do i = t, mi
          Write (nout,99999) y(r), x(i), f(i), ff(i), ff(i) - f(i)
        End Do

        Write (nout,*)
      End Do

!     Output the Chebyshev coefficients of the fit

      Write (nout,*) 'Chebyshev coefficients of the fit'
      Write (nout,*)

      Do j = 1, k + 1
        Write (nout,99998)(a(i),i=1+(j-1)*(l+1),j*(l+1))
      End Do

99999 Format (3X,4F11.4,E11.2)
99998 Format (1X,6F11.4)
    End Program e02cafe
```

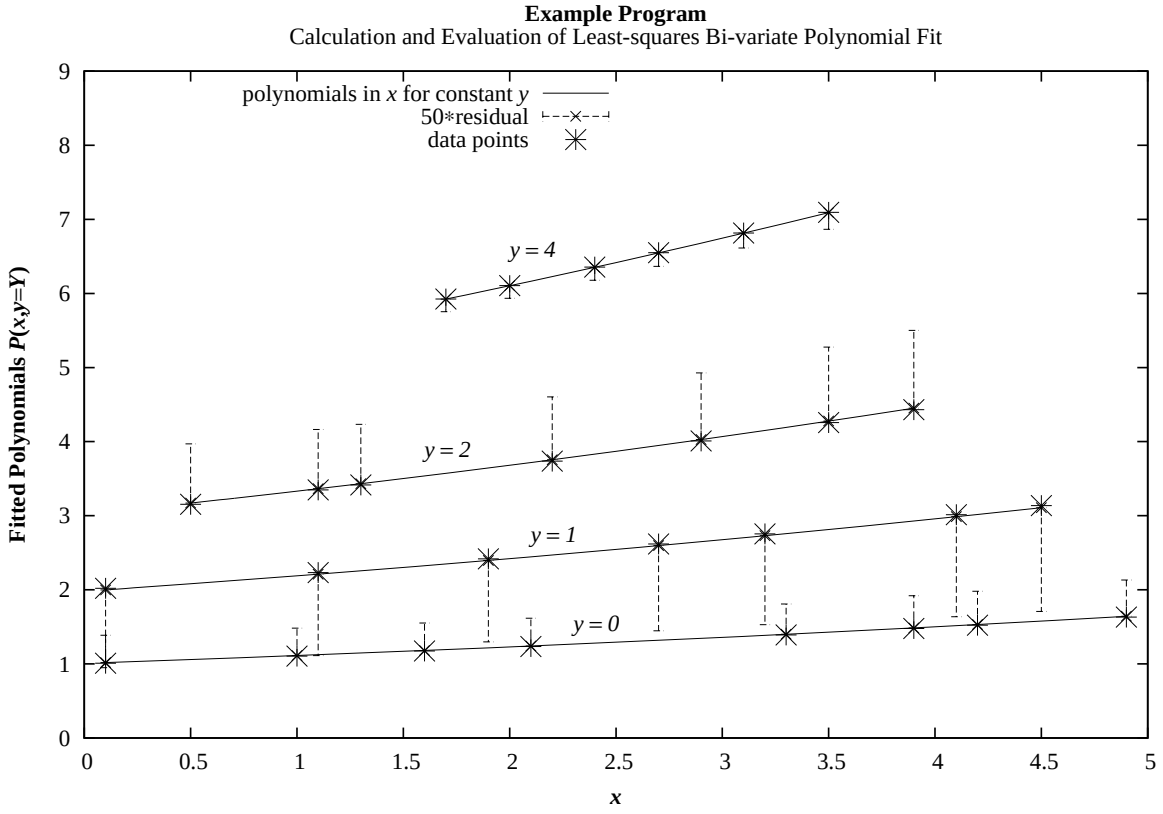
10.2 Program Data

```
E02CAF 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
```

10.3 Program Results

```
 E02CAF Example Program Results

      Data Y     Data X     Data F   Fitted F   Residual

        0.0000     0.1000     1.0100     1.0175   0.74E-02
        0.0000     1.0000     1.1052     1.1126   0.74E-02
        0.0000     1.6000     1.1735     1.1809   0.74E-02
        0.0000     2.1000     1.2337     1.2412   0.75E-02
        0.0000     3.3000     1.3910     1.3992   0.82E-02
        0.0000     3.9000     1.4770     1.4857   0.87E-02
        0.0000     4.2000     1.5220     1.5310   0.90E-02
        0.0000     4.9000     1.6323     1.6422   0.98E-02

        1.0000     0.1000     2.0201     1.9987  -0.21E-01
        1.0000     1.1000     2.2326     2.2110  -0.22E-01
        1.0000     1.9000     2.4185     2.3962  -0.22E-01
        1.0000     2.7000     2.6199     2.5966  -0.23E-01
        1.0000     3.2000     2.7543     2.7299  -0.24E-01
        1.0000     4.1000     3.0136     2.9869  -0.27E-01
        1.0000     4.5000     3.1366     3.1084  -0.28E-01

        2.0000     0.5000     3.1538     3.1700   0.16E-01
        2.0000     1.1000     3.3488     3.3648   0.16E-01
        2.0000     1.3000     3.4165     3.4325   0.16E-01
        2.0000     2.2000     3.7382     3.7549   0.17E-01
        2.0000     2.9000     4.0093     4.0272   0.18E-01
        2.0000     3.5000     4.2572     4.2769   0.20E-01
        2.0000     3.9000     4.4309     4.4521   0.21E-01

        4.0000     1.7000     5.9265     5.9231  -0.34E-02
        4.0000     2.0000     6.1070     6.1036  -0.34E-02
        4.0000     2.4000     6.3563     6.3527  -0.35E-02
        4.0000     2.7000     6.5498     6.5462  -0.36E-02
        4.0000     3.1000     6.8171     6.8132  -0.40E-02
        4.0000     3.5000     7.0953     7.0909  -0.45E-02

 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
```


Example Program
Calculation and Evaluation of Least-squares Bi-variate Polynomial Fit
polynomials in x for constant y
50*residual
data points
y = 4
y = 2
y = 1
y = 0
Fitted Polynomials P(x,y=Y)
x
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
1
2
3
4
5
6
7
8
9