NAG Library Routine Document

E02DCF

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

E02DCF computes a bicubic spline approximation to a set of data values, given on a rectangular grid in the x-y plane. The knots of the spline are located automatically, but a single parameter must be specified to control the trade-off between closeness of fit and smoothness of fit.

2 Specification

```
SUBROUTINE E02DCF (START, MX, X, MY, Y, F, S, NXEST, NYEST, NX, LAMDA,      &
                   NY, MU, C, FP, WRK, LWRK, IWRK, LIWRK, IFAIL)
INTEGER            MX, MY, NXEST, NYEST, NX, NY, LWRK, IWRK(LIWRK),         &
                   LIWRK, IFAIL
REAL (KIND=nag_wp) X(MX), Y(MY), F(MX*MY), S, LAMDA(NXEST), MU(NYEST),      &
                   C((NXEST-4)*(NYEST-4)), FP, WRK(LWRK)
CHARACTER(1)       START
```

3 Description

E02DCF determines a smooth bicubic spline approximation $s(x,y)$ to the set of data points $\left(x_q, y_r, f_{q,r}\right)$, for $q = 1,2,\ldots,m_x$ and $r = 1,2,\ldots,m_y$.

The spline is given in the B-spline representation

$$s(x,y) = \sum_{i=1}^{n_x-4} \sum_{j=1}^{n_y-4} c_{ij} M_i(x) N_j(y), \tag{1}$$

where $M_i(x)$ and $N_j(y)$ denote normalized cubic B-splines, the former defined on the knots λ_i to λ_{i+4} and the latter on the knots μ_j to μ_{j+4}. For further details, see Hayes and Halliday (1974) for bicubic splines and de Boor (1972) for normalized B-splines.

The total numbers n_x and n_y of these knots and their values $\lambda_1,\ldots,\lambda_{n_x}$ and $\mu_1,\ldots,\mu_{n_y}$ are chosen automatically by the routine. The knots $\lambda_5,\ldots,\lambda_{n_x-4}$ and $\mu_5,\ldots,\mu_{n_y-4}$ are the interior knots; they divide the approximation domain $[x_1, x_{m_x}] \times \left[y_1, y_{m_y}\right]$ into $(n_x - 7) \times \left(n_y - 7\right)$ subpanels $[\lambda_i, \lambda_{i+1}] \times \left[\mu_j, \mu_{j+1}\right]$, for $i = 4,5,\ldots,n_x - 4$ and $j = 4,5,\ldots,n_y - 4$. Then, much as in the curve case (see E02BEF), the coefficients c_{ij} are determined as the solution of the following constrained minimization problem:

$$\text{minimize}\, \eta, \tag{2}$$

subject to the constraint

$$\theta = \sum_{q=1}^{m_x} \sum_{r=1}^{m_y} \epsilon_{q,r}^2 \leq S, \tag{3}$$

where η is a measure of the (lack of) smoothness of $s(x,y)$. Its value depends on the discontinuity jumps in $s(x,y)$ across the boundaries of the subpanels. It is zero only when there are no discontinuities and is positive otherwise, increasing with the size of the jumps (see Dierckx (1982) for details).

$\epsilon_{q,r}$ denotes the residual $f_{q,r} - s\left(x_q, y_r\right)$,

and S is a non-negative number specified by you.

By means of the parameter S, 'the smoothing factor', you will then control the balance between smoothness and closeness of fit, as measured by the sum of squares of residuals in (3). If S is too large, the spline will be too smooth and signal will be lost (underfit); if S is too small, the spline will pick up too much noise (overfit). In the extreme cases the routine will return an interpolating spline ($\theta = 0$) if S is set to zero, and the least squares bicubic polynomial ($\eta = 0$) if S is set very large. Experimenting with S-values between these two extremes should result in a good compromise. (See Section 9.3 for advice on choice of S.)

The method employed is outlined in Section 9.5 and fully described in Dierckx (1981) and Dierckx (1982). It involves an adaptive strategy for locating the knots of the bicubic spline (depending on the function underlying the data and on the value of S), and an iterative method for solving the constrained minimization problem once the knots have been determined.

Values and derivatives of the computed spline can subsequently be computed by calling E02DEF, E02DFF or E02DHF as described in Section 9.6.

4 References

de Boor C (1972) On calculating with B-splines *J. Approx. Theory* **6** 50–62

Dierckx P (1981) An improved algorithm for curve fitting with spline functions *Report TW54* Department of Computer Science, Katholieke Univerciteit Leuven

Dierckx P (1982) A fast algorithm for smoothing data on a rectangular grid while using spline functions *SIAM J. Numer. Anal.* **19** 1286–1304

Hayes J G and Halliday J (1974) The least squares fitting of cubic spline surfaces to general data sets *J. Inst. Math. Appl.* **14** 89–103

Reinsch C H (1967) Smoothing by spline functions *Numer. Math.* **10** 177–183

5 Parameters

1: START – CHARACTER(1) *Input*

On entry: determines whether calculations are to be performed afresh (Cold Start) or whether knots found in previous calls are to be used as an initial estimate of knot placement (Warm Start).

START = 'C'
: The routine will build up the knot set starting with no interior knots. No values need be assigned to the parameters NX, NY, LAMDA, MU, WRK or IWRK.

START = 'W'
: The routine will restart the knot-placing strategy using the knots found in a previous call of the routine. In this case, the parameters NX, NY, LAMDA, MU, WRK and IWRK must be unchanged from that previous call. This warm start can save much time in determining a satisfactory set of knots for the given value of S. This is particularly useful when different smoothing factors are used for the same dataset.

Constraint: START = 'C' or 'W'.

2: MX – INTEGER *Input*

On entry: m_x, the number of grid points along the x axis.

Constraint: MX $\geq$ 4.

3: X(MX) – REAL (KIND=nag_wp) array *Input*

On entry: X(q) must be set to x_q, the x coordinate of the qth grid point along the x axis, for $q = 1, 2, \ldots, m_x$.

Constraint: $x_1 < x_2 < \cdots < x_{m_x}$.

4: MY – INTEGER *Input*

On entry: m_y, the number of grid points along the y axis.

Constraint: MY ≥ 4.

5: Y(MY) – REAL (KIND=nag_wp) array *Input*

On entry: $\mathrm{Y}(r)$ must be set to y_r, the y coordinate of the rth grid point along the y axis, for $r = 1, 2, \ldots, m_y$.

Constraint: $y_1 < y_2 < \cdots < y_{m_y}$.

6: F(MX × MY) – REAL (KIND=nag_wp) array *Input*

On entry: $\mathrm{F}(m_y \times (q-1) + r)$ must contain the data value $f_{q,r}$, for $q = 1, 2, \ldots, m_x$ and $r = 1, 2, \ldots, m_y$.

7: S – REAL (KIND=nag_wp) *Input*

On entry: the smoothing factor, S.

If $\mathrm{S} = 0.0$, the routine returns an interpolating spline.

If S is smaller than ***machine precision***, it is assumed equal to zero.

For advice on the choice of S, see Sections 3 and 9.3.

Constraint: $\mathrm{S} \geq 0.0$.

8: NXEST – INTEGER *Input*
9: NYEST – INTEGER *Input*

On entry: an upper bound for the number of knots n_x and n_y required in the x- and y-directions respectively.

In most practical situations, $\mathrm{NXEST} = m_x/2$ and $\mathrm{NYEST} = m_y/2$ is sufficient. NXEST and NYEST never need to be larger than $m_x + 4$ and $m_y + 4$ respectively, the numbers of knots needed for interpolation ($\mathrm{S} = 0.0$). See also Section 9.4.

Constraints:

NXEST ≥ 8;
NYEST ≥ 8.

10: NX – INTEGER *Input/Output*

On entry: if the warm start option is used, the value of NX must be left unchanged from the previous call.

On exit: the total number of knots, n_x, of the computed spline with respect to the x variable.

11: LAMDA(NXEST) – REAL (KIND=nag_wp) array *Input/Output*

On entry: if the warm start option is used, the values $\mathrm{LAMDA}(1), \mathrm{LAMDA}(2), \ldots, \mathrm{LAMDA}(\mathrm{NX})$ must be left unchanged from the previous call.

On exit: contains the complete set of knots λ_i associated with the x variable, i.e., the interior knots $\mathrm{LAMDA}(5), \mathrm{LAMDA}(6), \ldots, \mathrm{LAMDA}(\mathrm{NX}-4)$ as well as the additional knots

$$\mathrm{LAMDA}(1) = \mathrm{LAMDA}(2) = \mathrm{LAMDA}(3) = \mathrm{LAMDA}(4) = \mathrm{X}(1)$$

and

$$\mathrm{LAMDA}(\mathrm{NX}-3) = \mathrm{LAMDA}(\mathrm{NX}-2) = \mathrm{LAMDA}(\mathrm{NX}-1) = \mathrm{LAMDA}(\mathrm{NX}) = \mathrm{X}(\mathrm{MX})$$

needed for the B-spline representation.

12: NY – INTEGER *Input/Output*

On entry: if the warm start option is used, the value of NY must be left unchanged from the previous call.

On exit: the total number of knots, n_y, of the computed spline with respect to the y variable.

13: MU(NYEST) – REAL (KIND=nag_wp) array *Input/Output*

On entry: if the warm start option is used, the values $\mathrm{MU}(1), \mathrm{MU}(2), \ldots, \mathrm{MU}(\mathrm{NY})$ must be left unchanged from the previous call.

On exit: contains the complete set of knots μ_i associated with the y variable, i.e., the interior knots $\mathrm{MU}(5), \mathrm{MU}(6), \ldots, \mathrm{MU}(\mathrm{NY}-4)$ as well as the additional knots

$$\mathrm{MU}(1) = \mathrm{MU}(2) = \mathrm{MU}(3) = \mathrm{MU}(4) = \mathrm{Y}(1)$$

and

$$\mathrm{MU}(\mathrm{NY}-3) = \mathrm{MU}(\mathrm{NY}-2) = \mathrm{MU}(\mathrm{NY}-1) = \mathrm{MU}(\mathrm{NY}) = \mathrm{Y}(\mathrm{MY})$$

needed for the B-spline representation.

14: $\mathrm{C}((\mathrm{NXEST}-4) \times (\mathrm{NYEST}-4))$ – REAL (KIND=nag_wp) array *Output*

On exit: the coefficients of the spline approximation. $\mathrm{C}\left(\left(n_y-4\right) \times (i-1)+j\right)$ is the coefficient c_{ij} defined in Section 3.

15: FP – REAL (KIND=nag_wp) *Output*

On exit: the sum of squared residuals, θ, of the computed spline approximation. If $\mathrm{FP} = 0.0$, this is an interpolating spline. FP should equal S within a relative tolerance of 0.001 unless $\mathrm{NX} = \mathrm{NY} = 8$, when the spline has no interior knots and so is simply a bicubic polynomial. For knots to be inserted, S must be set to a value below the value of FP produced in this case.

16: WRK(LWRK) – REAL (KIND=nag_wp) array *Communication Array*

If the warm start option is used, on entry, the values $\mathrm{WRK}(1), \ldots, \mathrm{WRK}(4)$ must be left unchanged from the previous call.

This array is used as workspace.

17: LWRK – INTEGER *Input*

On entry: the dimension of the array WRK as declared in the (sub)program from which E02DCF is called.

Constraint: $\mathrm{LWRK} \geq 4 \times (\mathrm{MX}+\mathrm{MY}) + 11 \times (\mathrm{NXEST}+\mathrm{NYEST}) + \mathrm{NXEST} \times \mathrm{MY} + \max(\mathrm{MY}, \mathrm{NXEST}) + 54$.

18: IWRK(LIWRK) – INTEGER array *Communication Array*

If the warm start option is used, on entry, the values $\mathrm{IWRK}(1), \ldots, \mathrm{IWRK}(3)$ must be left unchanged from the previous call.

This array is used as workspace.

19: LIWRK – INTEGER *Input*

On entry: the dimension of the array IWRK as declared in the (sub)program from which E02DCF is called.

Constraint: $\mathrm{LIWRK} \geq 3 + \mathrm{MX} + \mathrm{MY} + \mathrm{NXEST} + \mathrm{NYEST}$.

20: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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 parameter, 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, START $\neq$ 'C' or 'W',
or MX < 4,
or MY < 4,
or S < 0.0,
or S $= 0.0$ and NXEST $<$ MX $+ 4$,
or S $= 0.0$ and NYEST $<$ MY $+ 4$,
or NXEST < 8,
or NYEST < 8,
or LWRK $< 4 \times (\mathrm{MX} + \mathrm{MY}) + 11 \times (\mathrm{NXEST} + \mathrm{NYEST}) + \mathrm{NXEST} \times \mathrm{MY} + \max(\mathrm{MY}, \mathrm{NXEST}) + 54$,
or LIWRK $< 3 + \mathrm{MX} + \mathrm{MY} + \mathrm{NXEST} + \mathrm{NYEST}$.

IFAIL $= 2$

The values of X(q), for $q = 1, 2, \ldots,$ MX, are not in strictly increasing order.

IFAIL $= 3$

The values of Y(r), for $r = 1, 2, \ldots,$ MY, are not in strictly increasing order.

IFAIL $= 4$

The number of knots required is greater than allowed by NXEST and NYEST. Try increasing NXEST and/or NYEST and, if necessary, supplying larger arrays for the parameters LAMDA, MU, C, WRK and IWRK. However, if NXEST and NYEST are already large, say NXEST $>$ MX$/2$ and NYEST $>$ MY$/2$, then this error exit may indicate that S is too small.

IFAIL $= 5$

The iterative process used to compute the coefficients of the approximating spline has failed to converge. This error exit may occur if S has been set very small. If the error persists with increased S, contact NAG.

IFAIL $= -99$

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

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

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

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

If IFAIL $= 4$ or 5, a spline approximation is returned, but it fails to satisfy the fitting criterion (see (2) and (3) in Section 3) – perhaps by only a small amount, however.

7 Accuracy

On successful exit, the approximation returned is such that its sum of squared residuals FP is equal to the smoothing factor S, up to a specified relative tolerance of 0.001 – except that if $n_x = 8$ and $n_y = 8$, FP may be significantly less than S: in this case the computed spline is simply the least squares bicubic polynomial approximation of degree 3, i.e., a spline with no interior knots.

8 Parallelism and Performance

E02DCF is not threaded by NAG in any implementation.

E02DCF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

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

9.1 Timing

The time taken for a call of E02DCF depends on the complexity of the shape of the data, the value of the smoothing factor S, and the number of data points. If E02DCF is to be called for different values of S, much time can be saved by setting START $=$ 'W' after the first call.

9.2 Weighting of Data Points

E02DCF does not allow individual weighting of the data values. If these were determined to widely differing accuracies, it may be better to use E02DDF. The computation time would be very much longer, however.

9.3 Choice of S

If the standard deviation of $f_{q,r}$ is the same for all q and r (the case for which E02DCF is designed – see Section 9.2.) and known to be equal, at least approximately, to σ, say, then following Reinsch (1967) and choosing the parameter S in the range $\sigma^2\left(m \pm \sqrt{2m}\right)$, where $m = m_x m_y$, is likely to give a good start in the search for a satisfactory value. If the standard deviations vary, the sum of their squares over all the data points could be used. Otherwise experimenting with different values of S will be required from the start, taking account of the remarks in Section 3.

In that case, in view of computation time and memory requirements, it is recommended to start with a very large value for S and so determine the least squares bicubic polynomial; the value returned for FP, call it FP_0, gives an upper bound for S. Then progressively decrease the value of S to obtain closer fits – say by a factor of 10 in the beginning, i.e., $\mathrm{S} = \mathrm{FP}_0/10$, $\mathrm{S} = \mathrm{FP}_0/100$, and so on, and more carefully as the approximation shows more details.

The number of knots of the spline returned, and their location, generally depend on the value of S and on the behaviour of the function underlying the data. However, if E02DCF is called with START = 'W', the knots returned may also depend on the smoothing factors of the previous calls. Therefore if, after a number of trials with different values of S and START = 'W', a fit can finally be accepted as satisfactory, it may be worthwhile to call E02DCF once more with the selected value for S but now using START = 'C'. Often, E02DCF then returns an approximation with the same quality of fit but with fewer knots, which is therefore better if data reduction is also important.

9.4 Choice of NXEST and NYEST

The number of knots may also depend on the upper bounds NXEST and NYEST. Indeed, if at a certain stage in E02DCF the number of knots in one direction (say n_x) has reached the value of its upper bound (NXEST), then from that moment on all subsequent knots are added in the other (y) direction. Therefore you have the option of limiting the number of knots the routine locates in any direction. For example, by setting NXEST = 8 (the lowest allowable value for NXEST), you can indicate that you want an approximation which is a simple cubic polynomial in the variable x.

9.5 Outline of Method Used

If $S = 0$, the requisite number of knots is known in advance, i.e., $n_x = m_x + 4$ and $n_y = m_y + 4$; the interior knots are located immediately as $\lambda_i = x_{i-2}$ and $\mu_j = y_{j-2}$, for $i = 5, 6, \ldots, n_x - 4$ and $j = 5, 6, \ldots, n_y - 4$. The corresponding least squares spline is then an interpolating spline and therefore a solution of the problem.

If $S > 0$, suitable knot sets are built up in stages (starting with no interior knots in the case of a cold start but with the knot set found in a previous call if a warm start is chosen). At each stage, a bicubic spline is fitted to the data by least squares, and θ, the sum of squares of residuals, is computed. If $\theta > S$, new knots are added to one knot set or the other so as to reduce θ at the next stage. The new knots are located in intervals where the fit is particularly poor, their number depending on the value of S and on the progress made so far in reducing θ. Sooner or later, we find that $\theta \leq S$ and at that point the knot sets are accepted. The routine then goes on to compute the (unique) spline which has these knot sets and which satisfies the full fitting criterion specified by (2) and (3). The theoretical solution has $\theta = S$. The routine computes the spline by an iterative scheme which is ended when $\theta = S$ within a relative tolerance of 0.001. The main part of each iteration consists of a linear least squares computation of special form, done in a similarly stable and efficient manner as in E02BAF for least squares curve-fitting.

An exception occurs when the routine finds at the start that, even with no interior knots $\left(n_x = n_y = 8\right)$, the least squares spline already has its sum of residuals $\leq S$. In this case, since this spline (which is simply a bicubic polynomial) also has an optimal value for the smoothness measure η, namely zero, it is returned at once as the (trivial) solution. It will usually mean that S has been chosen too large.

For further details of the algorithm and its use see Dierckx (1982).

9.6 Evaluation of Computed Spline

The values of the computed spline at the points (x_r, y_r), for $r = 1, 2, \ldots, m$, may be obtained in the real array FF (see E02DEF), of length at least m, by the following call:

```
IFAIL = 0
CALL E02DEF(M,PX,PY,X,Y,LAMDA,MU,C,FF,WRK,IWRK,IFAIL)
```

where `M` $= m$ and the coordinates x_r, y_r are stored in `X`(k), `Y`(k). `PX` and `PY` have the same values as NX and NY as output from E02DCF, and `LAMDA`, `MU` and `C` have the same values as LAMDA, MU and C output from E02DCF. `WRK` is a real workspace array of length at least `PY` $- 4$, and `IWRK` is an integer workspace array of length at least `PY` $- 4$.

To evaluate the computed spline on a m_x by m_y rectangular grid of points in the x-y plane, which is defined by the x coordinates stored in `X`(q), for $q = 1, 2, \ldots, m_x$, and the y coordinates stored in `Y`(r), for $r = 1, 2, \ldots, m_y$, returning the results in the real array FF (see E02DFF) which is of length at least MX $\times$ MY, the following call may be used:

```
      IFAIL = 0
      CALL E02DFF(MX,MY,PX,PY,X,Y,LAMDA,MU,C,FG,WRK,LWRK,
     *            IWRK,LIWRK,IFAIL)
```

where $\mathtt{MX} = m_x$, $\mathtt{MY} = m_y$. `PX` and `PY` have the same values as NX and NY as output from E02DCF, and`LAMDA`, `MU` and `C` have the same values as LAMDA, MU and C output from E02DCF. `WRK` is a real workspace array of length at least $\mathtt{LWRK} = \min(\mathit{nwrk1}, \mathit{nwrk2})$, where $\mathit{nwrk1} = \mathtt{MX} \times 4 + \mathtt{PX}$ and $\mathit{nwrk2} = \mathtt{MY} \times 4 + \mathtt{PY}$. `IWRK` is an integer workspace array of length at least $\mathtt{LIWRK} = \mathtt{MY} + \mathtt{PY} - 4$ if $\mathit{nwrk1} \geq \mathit{nwrk2}$, or $\mathtt{MX} + \mathtt{PX} - 4$ otherwise.

The result of the spline evaluated at grid point (q, r) is returned in element $(\mathtt{MY} \times (q-1) + r)$ of the array `FG`.

10 Example

This example reads in values of MX, MY, x_q, for $q = 1, 2, \ldots,$MX, and y_r, for $r = 1, 2, \ldots,$MY, followed by values of the ordinates $f_{q,r}$ defined at the grid points $\left(x_q, y_r\right)$. It then calls E02DCF to compute a bicubic spline approximation for one specified value of S, and prints the values of the computed knots and B-spline coefficients. Finally it evaluates the spline at a small sample of points on a rectangular grid.

10.1 Program Text

```
!   E02DCF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.
    Module e02dcfe_mod

!     E02DCF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                               :: cprint
!     .. Parameters ..
      Integer, Parameter, Public           :: nin = 5, nout = 6
    Contains
      Subroutine cprint(c,ny,nx,nout)

!       .. Scalar Arguments ..
        Integer, Intent (In)               :: nout, nx, ny
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In)    :: c(ny-4,nx-4)
!       .. Local Scalars ..
        Integer                            :: i
!       .. Executable Statements ..
        Write (nout,*)
        Write (nout,*) 'The B-spline coefficients:'
        Write (nout,*)

        Do i = 1, ny - 4
          Write (nout,99999) c(i,1:(nx-4))
        End Do

        Return

99999   Format (1X,8F9.4)
      End Subroutine cprint
    End Module e02dcfe_mod
    Program e02dcfe

!     E02DCF Example Main Program

!     .. Use Statements ..
```

```
      Use nag_library, Only: e02dcf, e02dff, nag_wp
      Use e02dcfe_mod, Only: cprint, nin, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                 :: delta, fp, s, xhi, xlo, yhi, ylo
      Integer                            :: i, ifail, j, liwrk, lwrk, mx,    &
                                            my, npx, npy, nx, nxest, ny, nyest
      Character (1)                      :: start
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable    :: c(:), f(:), fg(:), lamda(:),     &
                                            mu(:), px(:), py(:), wrk(:),     &
                                            x(:), y(:)
      Integer, Allocatable               :: iwrk(:)
!     .. Intrinsic Procedures ..
      Intrinsic                          :: max, min, real
!     .. Executable Statements ..
      Write (nout,*) 'E02DCF Example Program Results'

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

!     Input the number of X, Y co-ordinates MX, MY.

      Read (nin,*) mx, my
      nxest = mx + 4
      nyest = my + 4
      lwrk = 4*(mx+my) + 11*(nxest+nyest) + nxest*my + max(my,nxest) + 54
      liwrk = 3 + mx + my + nxest + nyest
      Allocate (x(mx),y(my),f(mx*my),lamda(nxest),mu(nyest),wrk(lwrk), &
        iwrk(liwrk),c((nxest-4)*(nyest-4)))

      Read (nin,*) x(1:mx)
      Read (nin,*) y(1:my)

!     Input the MX*MY function values F at the grid points.

      Read (nin,*) f(1:mx*my)

      start = 'C'

      Read (nin,*) s

!     Determine the spline approximation.

      ifail = 0
      Call e02dcf(start,mx,x,my,y,f,s,nxest,nyest,nx,lamda,ny,mu,c,fp,wrk, &
        lwrk,iwrk,liwrk,ifail)

      Deallocate (wrk,iwrk)

      Write (nout,*)
      Write (nout,99999) 'Calling with smoothing factor S =', s, ': NX =', nx, &
        ', NY =', ny, '.'
      Write (nout,*)
      Write (nout,*) '               I   Knot LAMDA(I)      J     Knot MU(J)'
      Write (nout,*)

      Do j = 4, max(nx,ny) - 3

        If (j<=nx-3 .And. j<=ny-3) Then
          Write (nout,99997) j, lamda(j), j, mu(j)
        Else If (j<=nx-3) Then
          Write (nout,99997) j, lamda(j)
        Else If (j<=ny-3) Then
          Write (nout,99996) j, mu(j)
        End If

      End Do

      Call cprint(c,ny,nx,nout)
      Write (nout,*)
      Write (nout,99998) 'Sum of squared residuals FP =', fp

      If (fp==0.0E+0_nag_wp) Then
        Write (nout,*) '(The spline is an interpolating spline)'
      Else If (nx==8 .And. ny==8) Then
        Write (nout,*) '(The spline is the least-squares bi-cubic polynomial)'
      End If

!     Evaluate the spline on a rectangular grid at NPX*NPY points
!     over the domain (XLO to XHI) x (YLO to YHI).

      Read (nin,*) npx, xlo, xhi
      Read (nin,*) npy, ylo, yhi

      lwrk = min(4*npx+nx,4*npy+ny)

      If (4*npx+nx>4*npy+ny) Then
        liwrk = npy + ny - 4
      Else
        liwrk = npx + nx - 4
      End If

      Allocate (px(npx),py(npy),fg(npx*npy),wrk(lwrk),iwrk(liwrk))

      delta = (xhi-xlo)/real(npx-1,kind=nag_wp)

      Do i = 1, npx
        px(i) = min(xlo+real(i-1,kind=nag_wp)*delta,xhi)
      End Do

      delta = (yhi-ylo)/real(npy-1,kind=nag_wp)

      Do i = 1, npy
        py(i) = min(ylo+real(i-1,kind=nag_wp)*delta,yhi)
      End Do

      ifail = 0
      Call e02dff(npx,npy,nx,ny,px,py,lamda,mu,c,fg,wrk,lwrk,iwrk,liwrk,ifail)

      Write (nout,*)
      Write (nout,*) 'Values of computed spline:'
      Write (nout,*)
      Write (nout,99995) '          X', (px(i),i=1,npx)
      Write (nout,*) '     Y'

      Do i = npy, 1, -1
        Write (nout,99994) py(i), (fg(npy*(j-1)+i),j=1,npx)
      End Do

99999 Format (1X,A,1P,E13.4,A,I5,A,I5,A)
99998 Format (1X,A,1P,E13.4)
99997 Format (1X,I16,F12.4,I11,F12.4)
99996 Format (1X,I39,F12.4)
99995 Format (1X,A,7F8.2)
99994 Format (1X,F8.2,3X,7F8.2)
    End Program e02dcfe
```

10.2 Program Data

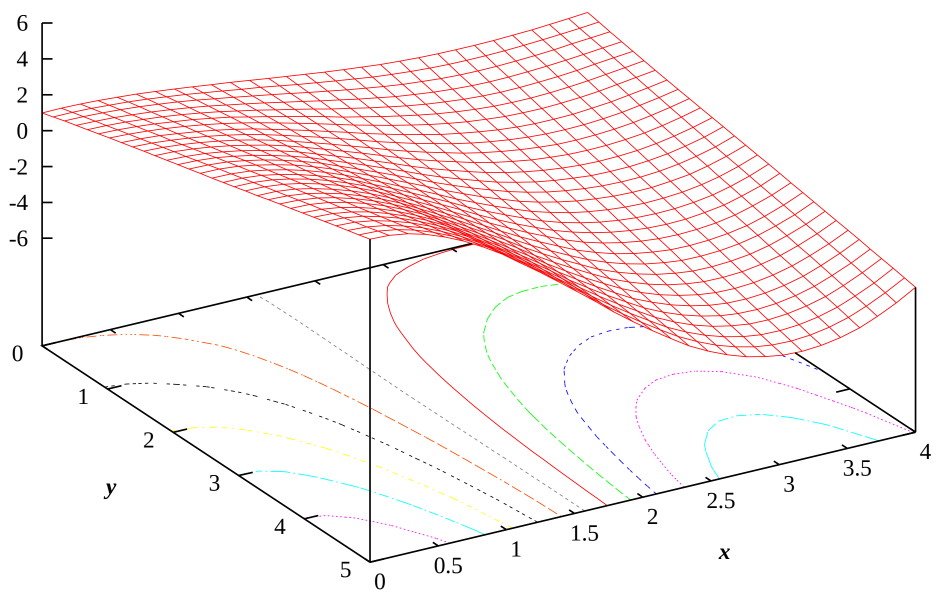
```
E02DCF Example Program Data
 11    9      MX, MY, number of grid points on the X and Y axes
 0.0000E+00  5.0000E-01  1.0000E+00  1.5000E+00  2.0000E+00
 2.5000E+00  3.0000E+00  3.5000E+00  4.0000E+00  4.5000E+00
 5.0000E+00   End of MX grid points
 0.0000E+00  5.0000E-01  1.0000E+00  1.5000E+00  2.0000E+00
 2.5000E+00  3.0000E+00  3.5000E+00  4.0000E+00   End of MY grid points
 1.0000E+00  8.8758E-01  5.4030E-01  7.0737E-02 -4.1515E-01
-8.0114E-01 -9.7999E-01 -9.3446E-01 -6.5664E-01  1.5000E+00
```

```
 1.3564E+00  8.2045E-01  1.0611E-01 -6.2422E-01 -1.2317E+00
-1.4850E+00 -1.3047E+00 -9.8547E-01  2.0600E+00  1.7552E+00
 1.0806E+00  1.5147E-01 -8.3229E-01 -1.6023E+00 -1.9700E+00
-1.8729E+00 -1.4073E+00  2.5700E+00  2.1240E+00  1.3508E+00
 1.7684E-01 -1.0404E+00 -2.0029E+00 -2.4750E+00 -2.3511E+00
-1.6741E+00  3.0000E+00  2.6427E+00  1.6309E+00  2.1221E-01
-1.2484E+00 -2.2034E+00 -2.9700E+00 -2.8094E+00 -1.9809E+00
 3.5000E+00  3.1715E+00  1.8611E+00  2.4458E-01 -1.4565E+00
-2.8640E+00 -3.2650E+00 -3.2776E+00 -2.2878E+00  4.0400E+00
 3.5103E+00  2.0612E+00  2.8595E-01 -1.6946E+00 -3.2046E+00
-3.9600E+00 -3.7958E+00 -2.6146E+00  4.5000E+00  3.9391E+00
 2.4314E+00  3.1632E-01 -1.8627E+00 -3.6351E+00 -4.4550E+00
-4.2141E+00 -2.9314E+00  5.0400E+00  4.3879E+00  2.7515E+00
 3.5369E-01 -2.0707E+00 -4.0057E+00 -4.9700E+00 -4.6823E+00
-3.2382E+00  5.5050E+00  4.8367E+00  2.9717E+00  3.8505E-01
-2.2888E+00 -4.4033E+00 -5.4450E+00 -5.1405E+00 -3.5950E+00
 6.0000E+00  5.2755E+00  3.2418E+00  4.2442E-01 -2.4769E+00
-4.8169E+00 -5.9300E+00 -5.6387E+00 -3.9319E+00   End of data values
 0.1         S, smoothing factor
 6    0.0   5.0
 5    0.0   4.0
```

10.3 Program Results

```
 E02DCF Example Program Results

 Calling with smoothing factor S =   1.0000E-01: NX =   10, NY =   13.

               I   Knot LAMDA(I)      J     Knot MU(J)

               4      0.0000          4      0.0000
               5      1.5000          5      1.0000
               6      2.5000          6      2.0000
               7      5.0000          7      2.5000
                                      8      3.0000
                                      9      3.5000
                                     10      4.0000

 The B-spline coefficients:

    0.9918   1.5381   2.3913   3.9845   5.2138   5.9965
    1.0546   1.5270   2.2441   4.2217   5.0860   6.0821
    0.6098   0.9557   1.5587   2.3458   3.3860   3.7716
   -0.2915  -0.4199  -0.7399  -1.1763  -1.5527  -1.7775
   -0.8476  -1.3296  -1.8521  -3.3468  -4.3628  -5.0085
   -1.0168  -1.5952  -2.4022  -3.9390  -5.4680  -6.1656
   -0.9529  -1.3381  -2.2844  -3.9559  -5.0032  -5.8709
   -0.7711  -1.0914  -1.8488  -3.2549  -3.9444  -4.7297
   -0.6476  -1.0373  -1.5936  -2.5887  -3.3485  -3.9330

 Sum of squared residuals FP =   1.0004E-01

 Values of computed spline:

           X    0.00     1.00     2.00     3.00     4.00     5.00
     Y
    4.00       -0.65    -1.36    -1.99    -2.61    -3.25    -3.93
    3.00       -0.98    -1.97    -2.91    -3.91    -4.97    -5.92
    2.00       -0.42    -0.83    -1.24    -1.66    -2.08    -2.48
    1.00        0.54     1.09     1.61     2.14     2.71     3.24
    0.00        0.99     2.04     3.03     4.01     5.02     6.00
```

Example Program
Calculation and Evaluation of Least Squares Bicubic Spline Fit


6
4
2
0
-2
-4
-6
0
1
2
3
4
5
y
0
0.5
1
1.5
2
2.5
3
3.5
4
x