NAG Library Routine Document

E02DDF

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

E02DDF computes a bicubic spline approximation to a set of scattered data. 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 E02DDF (START, M, X, Y, F, W, S, NXEST, NYEST, NX, LAMDA, NY,    &
                   MU, C, FP, RANK, WRK, LWRK, IWRK, LIWRK, IFAIL)
INTEGER            M, NXEST, NYEST, NX, NY, RANK, LWRK, IWRK(LIWRK),        &
                   LIWRK, IFAIL
REAL (KIND=nag_wp) X(M), Y(M), F(M), W(M), S, LAMDA(NXEST), MU(NYEST),      &
                   C((NXEST-4)*(NYEST-4)), FP, WRK(LWRK)
CHARACTER(1)       START
```

3 Description

E02DDF determines a smooth bicubic spline approximation $s(x,y)$ to the set of data points (x_r, y_r, f_r) with weights w_r, for $r = 1, 2, \ldots, m$.

The approximation domain is considered to be the rectangle $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$, where $x_{\min}\,(y_{\min})$ and $x_{\max}\,(y_{\max})$ denote the lowest and highest data values of $x(y)$.

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 $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ into $(n_x - 7) \times (n_y - 7)$ subpanels $[\lambda_i, \lambda_{i+1}] \times [\mu_j, \mu_{j+1}]$, 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:

minimize

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

subject to the constraint

$$\theta = \sum_{r=1}^{m} \epsilon_r^2 \leq S \tag{3}$$

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 (1981b) for details).
ϵ_r denotes the weighted residual $w_r(f_r - s(x_r,y_r))$,
and S is a non-negative number to be 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 method would return an interpolating spline $(\theta = 0)$ if S were set to zero, and returns 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 9.2 for advice on choice of S.) Note however, that this routine, unlike E02BEF and E02DCF, does not allow S to be set exactly to zero: to compute an interpolant to scattered data, E01SAF or E01SGF should be used.

The method employed is outlined in Section 9.5 and fully described in Dierckx (1981a) and Dierckx (1981b). 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 (1981a) An improved algorithm for curve fitting with spline functions *Report TW54* Department of Computer Science, Katholieke Univerciteit Leuven

Dierckx P (1981b) An algorithm for surface fitting with spline functions *IMA J. Numer. Anal.* **1** 267–283

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

Peters G and Wilkinson J H (1970) The least squares problem and pseudo-inverses *Comput. J.* **13** 309–316

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 or WRK.

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 and WRK 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: M – INTEGER *Input*

On entry: m, the number of data points.

The number of data points with nonzero weight (see W) must be at least 16.

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

4: Y(M) – REAL (KIND=nag_wp) array *Input*

5: F(M) – REAL (KIND=nag_wp) array *Input*

On entry: X(r), Y(r), F(r) must be set to the coordinates of (x_r, y_r, f_r), the rth data point, for $r = 1, 2, \ldots, m$. The order of the data points is immaterial.

6: W(M) – REAL (KIND=nag_wp) array *Input*

On entry: W(r) must be set to w_r, the rth value in the set of weights, for $r = 1, 2, \ldots, m$. Zero weights are permitted and the corresponding points are ignored, except when determining $x_{\min}$, $x_{\max}$, $y_{\min}$ and $y_{\max}$ (see Section 9.4). For advice on the choice of weights, see Section 2.1.2 in the E02 Chapter Introduction.

Constraint: the number of data points with nonzero weight must be at least 16.

7: S – REAL (KIND=nag_wp) *Input*

On entry: the smoothing factor, S.

For advice on the choice of S, see Sections 3 and 9.2.

Constraint: S > 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, NXEST = NYEST $= 4 + \sqrt{m/2}$ is sufficient. See also Section 9.3.

Constraint: NXEST ≥ 8 and 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 LAMDA(1), LAMDA(2), …, LAMDA(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 LAMDA(5), LAMDA(6), …, LAMDA(NX $-$ 4) as well as the additional knots

$$\text{LAMDA}(1) = \text{LAMDA}(2) = \text{LAMDA}(3) = \text{LAMDA}(4) = x_{\min}$$

and

$$\text{LAMDA}(\text{NX}-3) = \text{LAMDA}(\text{NX}-2) = \text{LAMDA}(\text{NX}-1) = \text{LAMDA}(\text{NX}) = x_{\max}$$

needed for the B-spline representation (where $x_{\min}$ and $x_{\max}$ are as described in Section 3).

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

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

and

$$\mathrm{MU}(\mathrm{NY}-3) = \mathrm{MU}(\mathrm{NY}-2) = \mathrm{MU}(\mathrm{NY}-1) = \mathrm{MU}(\mathrm{NY}) = y_{\max}$$

needed for the B-spline representation (where $y_{\min}$ and $y_{\max}$ are as described in Section 3).

14: C((NXEST − 4) × (NYEST − 4)) – REAL (KIND=nag_wp) array *Output*

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

15: FP – REAL (KIND=nag_wp) *Output*

On exit: the weighted sum of squared residuals, θ, of the computed spline approximation. 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: RANK – INTEGER *Output*

On exit: gives the rank of the system of equations used to compute the final spline (as determined by a suitable machine-dependent threshold). When $\mathrm{RANK} = (\mathrm{NX}-4)\times(\mathrm{NY}-4)$, the solution is unique; otherwise the system is rank-deficient and the minimum-norm solution is computed. The latter case may be caused by too small a value of S.

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

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

This array is used as workspace.

18: LWRK – INTEGER *Input*

On entry: the dimension of the array WRK as declared in the (sub)program from which E02DDF is called.

Constraint: $\mathrm{LWRK} \geq (7\times u\times v+25\times w)\times(w+1)+2\times(u+v+4\times\mathrm{M})+23\times w+56$, where $u = \mathrm{NXEST}-4$, $v = \mathrm{NYEST}-4$ and $w = \max(u, v)$..

For some problems, the routine may need to compute the minimal least squares solution of a rank-deficient system of linear equations (see Section 3). The amount of workspace required to solve such problems will be larger than specified by the value given above, which must be increased by an amount, *lwrk2* say. An upper bound for *lwrk2* is given by $4\times u\times v\times w+2\times u\times v+4\times w$, where u, v and w are as above. However, if there are enough data points, scattered uniformly over the approximation domain, and if the smoothing factor S is not too small, there is a good chance that this extra workspace is not needed. A lot of memory might therefore be saved by assuming that no additional workspace is required (*lwrk2* = 0).

19: IWRK(LIWRK) – INTEGER array *Workspace*

20: LIWRK – INTEGER *Input*

On entry: the dimension of the array IWRK as declared in the (sub)program from which E02DDF is called.

Constraint: LIWRK $\geq$ M + 2 × (NXEST − 7) × (NYEST − 7).

21: 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 the number of data points with nonzero weight $<$ 16,
or S $\leq$ 0.0,
or NXEST $<$ 8,
or NYEST $<$ 8,
or LWRK $<$ $(7 \times u \times v + 25 \times w) \times (w + 1) + 2 \times (u + v + 4 \times \mathrm{M}) + 23 \times w + 56$, where $u = \mathrm{NXEST} - 4$, $v = \mathrm{NYEST} - 4$ and $w = \max(u, v)$,
or LIWRK $<$ M + 2 × (NXEST − 7) × (NYEST − 7).

IFAIL = 2

On entry, either all the X(r), for $r = 1, 2, \ldots, \mathrm{M}$, are equal, or all the Y(r), for $r = 1, 2, \ldots, \mathrm{M}$, are equal.

IFAIL = 3

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, NYEST $> 4 + \sqrt{\mathrm{M}/2}$, then this error exit may indicate that S is too small.

IFAIL = 4

No more knots can be added because the number of B-spline coefficients (NX − 4) × (NY − 4) already exceeds the number of data points M. This error exit may occur if either of S or M is too small.

IFAIL = 5

No more knots can be added because the additional knot would (quasi) coincide with an old one. This error exit may occur if too large a weight has been given to an inaccurate data point, or if S is too small.

IFAIL $= 6$

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 $= 7$

LWRK is too small; the routine needs to compute the minimal least squares solution of a rank-deficient system of linear equations, but there is not enough workspace. There is no approximation returned but, having saved the information contained in NX, LAMDA, NY, MU and WRK, and having adjusted the value of LWRK and the dimension of array WRK accordingly, you can continue at the point the program was left by calling E02DDF with START = 'W'. Note that the requested value for LWRK is only large enough for the current phase of the algorithm. If the routine is restarted with LWRK set to the minimum value requested, a larger request may be made at a later stage of the computation. See Section 5 for the upper bound on LWRK. On soft failure, the minimum requested value for LWRK is returned in IWRK(1) and the safe value for LWRK is returned in IWRK(2).

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 $= 3$, 4, 5 or 6, a spline approximation is returned, but it fails to satisfy the fitting criterion (see (2) and (3) in Section 3 – perhaps only by a small amount, however.

7 Accuracy

On successful exit, the approximation returned is such that its weighted 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

E02DDF is not threaded by NAG in any implementation.

E02DDF 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 E02DDF depends on the complexity of the shape of the data, the value of the smoothing factor S, and the number of data points. If E02DDF is to be called for different values of S, much time can be saved by setting START = 'W' after the first call.

It should be noted that choosing S very small considerably increases computation time.

9.2 Choice of *S*

If the weights have been correctly chosen (see Section 2.1.2 in the E02 Chapter Introduction), the standard deviation of $w_r f_r$ would be the same for all r, equal to σ, say. In this case, choosing the smoothing factor S in the range $\sigma^2\left(m \pm \sqrt{2m}\right)$, as suggested by Reinsch (1967), is likely to give a good start in the search for a satisfactory value. Otherwise, experimenting with different values of S will be required from the start.

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.

To choose S very small is strongly discouraged. This considerably increases computation time and memory requirements. It may also cause rank-deficiency (as indicated by the parameter RANK) and endanger numerical stability.

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 E02DDF 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 E02DDF once more with the selected value for S but now using START = 'C'. Often, E02DDF then returns an approximation with the same quality of fit but with fewer knots, which is therefore better if data reduction is also important.

9.3 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 E02DDF 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. This may indicate that the value of NXEST is too small. On the other hand, it gives you 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.

9.4 Restriction of the approximation domain

The fit obtained is not defined outside the rectangle $[\lambda_4, \lambda_{n_x-3}] \times \left[\mu_4, \mu_{n_y-3}\right]$. The reason for taking the extreme data values of x and y for these four knots is that, as is usual in data fitting, the fit cannot be expected to give satisfactory values outside the data region. If, nevertheless, you require values over a larger rectangle, this can be achieved by augmenting the data with two artificial data points $(a, c, 0)$ and $(b, d, 0)$ with zero weight, where $[a, b] \times [c, d]$ denotes the enlarged rectangle.

9.5 Outline of method used

First 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 $\theta > \mathrm{S}$, a new knot is added to one knot set or the other so as to reduce θ at the next stage. The new knot is located in an interval where the fit is particularly poor. Sooner or later, we find that $\theta \leq \mathrm{S}$ and at that point the knot sets are accepted. The routine then goes on to compute a spline which has these knot sets and which

satisfies the full fitting criterion specified by (2) and (3). The theoretical solution has $\theta = \mathrm{S}$. The routine computes the spline by an iterative scheme which is ended when $\theta = \mathrm{S}$ 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 E02DAF. As there also, the minimal least squares solution is computed wherever the linear system is found to be rank-deficient.

An exception occurs when the routine finds at the start that, even with no interior knots ($\mathrm{N} = 8$), the least squares spline already has its sum of squares of residuals $\leq \mathrm{S}$. 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 (1981b).

9.6 Evaluation of Computed Spline

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

```
IFAIL = 0
CALL E02DEF(N,NX,NY,X,Y,LAMDA,MU,C,FF,WRK,IWRK,IFAIL)
```

where $\mathtt{N} = n$ and the coordinates x_r, y_r are stored in $\mathtt{X}(k)$, $\mathtt{Y}(k)$. PX and PY have the same values as NX and NY as output from E02DDF, and LAMDA, MU and C have the same values as LAMDA, MU and C output from E02DDF. WRK is a real workspace array of length at least $\mathtt{PY} - 4$, and IWRK is an integer workspace array of length at least $\mathtt{PY} - 4$.

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

```
 IFAIL = 0
 CALL E02DFF(KX,KY,NX,NY,TX,TY,LAMDA,MU,C,FG,WRK,LWRK,
*            IWRK,LIWRK,IFAIL)
```

where $\mathtt{KX} = k_x$, $\mathtt{KY} = k_y$. NX, NY,LAMDA, MU and C have the same values as NX, NY,LAMDA, MU and C output from E02DDF. WRK is a real workspace array of length at least $\mathtt{LWRK} = \min(\mathit{nwrk1}, \mathit{nwrk2})$, where $\mathit{nwrk1} = \mathtt{KX} \times 4 + \mathtt{PX}$ and $\mathit{nwrk2} = \mathtt{KY} \times 4 + \mathtt{PY}$. IWRK is an integer workspace array of length at least $\mathtt{LIWRK} = \mathtt{KY} + \mathtt{PY} - 4$ if $\mathit{nwrk1} \geq \mathit{nwrk2}$, or $\mathtt{KX} + \mathtt{PX} - 4$ otherwise.

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

10 Example

This example reads in a value of M, followed by a set of M data points (x_r, y_r, f_r) and their weights w_r. It then calls E02DDF 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

```
!   E02DDF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.
    Module e02ddfe_mod

!     E02DDF 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,7F9.2)
      End Subroutine cprint
    End Module e02ddfe_mod
    Program e02ddfe

!     E02DDF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e02ddf, e02dff, nag_wp
      Use e02ddfe_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, m,     &
                                              npx, npy, nx, nxest, ny, nyest,  &
                                              rank, u, v, ww
      Character (1)                        :: start
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: c(:), f(:), fg(:), lamda(:),     &
                                              mu(:), px(:), py(:), w(:),       &
                                              wrk(:), x(:), y(:)
      Integer, Allocatable                 :: iwrk(:)
!     .. Intrinsic Procedures ..
      Intrinsic                            :: max, min, real
!     .. Executable Statements ..
      Write (nout,*) 'E02DDF Example Program Results'

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

      Read (nin,*) m
      nxest = m
      nyest = nxest
      liwrk = m + 2*(nxest-7)*(nyest-7)
      u = nxest - 4
      v = nyest - 4
      ww = max(u,v)
      lwrk = (7*u*v+25*ww)*(ww+1) + 2*(u+v+4*m) + 23*ww + 56
      Allocate (x(m),y(m),f(m),w(m),lamda(nxest),mu(nyest),c((nxest-4)*(nyest- &
        4)),iwrk(liwrk),wrk(lwrk))

!     Input the data-points and the weights.

      Do i = 1, m
        Read (nin,*) x(i), y(i), f(i), w(i)
      End Do

      start = 'C'
```

```
      Read (nin,*) s

!     Determine the spline approximation.

      ifail = 0
      Call e02ddf(start,m,x,y,f,w,s,nxest,nyest,nx,lamda,ny,mu,c,fp,rank,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,99998) 'rank deficiency =', (nx-4)*(ny-4) - rank
      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,99996) j, lamda(j), j, mu(j)
        Else If (j<=nx-3) Then
          Write (nout,99996) j, lamda(j)
        Else If (j<=ny-3) Then
          Write (nout,99995) j, mu(j)
        End If

      End Do

      Call cprint(c,ny,nx,nout)

      Write (nout,*)
      Write (nout,99997) ' Sum of squared residuals FP =', fp

      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

      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,99994) '          X', (px(i),i=1,npx)
      Write (nout,*) '     Y'

      Do i = npy, 1, -1
        Write (nout,99993) 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,I5)
99997 Format (1X,A,1P,E13.4,A)
99996 Format (1X,I16,F12.4,I11,F12.4)
99995 Format (1X,I39,F12.4)
99994 Format (1X,A,7F8.2)
99993 Format (1X,F8.2,3X,7F8.2)
    End Program e02ddfe
```

10.2 Program Data

```
E02DDF Example Program Data
 30                          M, number of data points
 11.16   1.24  22.15  1.00  X,Y,F,W data point coordinates and weight
 12.85   3.06  22.11  1.00
 19.85  10.72   7.97  1.00
 19.72   1.39  16.83  1.00
 15.91   7.74  15.30  1.00
  0.00  20.00  34.60  1.00
 20.87  20.00   5.74  1.00
  3.45  12.78  41.24  1.00
 14.26  17.87  10.74  1.00
 17.43   3.46  18.60  1.00
 22.80  12.39   5.47  1.00
  7.58   1.98  29.87  1.00
 25.00  11.87   4.40  1.00
  0.00   0.00  58.20  1.00
  9.66  20.00   4.73  1.00
  5.22  14.66  40.36  1.00
 17.25  19.57   6.43  1.00
 25.00   3.87   8.74  1.00
 12.13  10.79  13.71  1.00
 22.23   6.21  10.25  1.00
 11.52   8.53  15.74  1.00
 15.20   0.00  21.60  1.00
  7.54  10.69  19.31  1.00
 17.32  13.78  12.11  1.00
  2.14  15.03  53.10  1.00
  0.51   8.37  49.43  1.00
 22.69  19.63   3.25  1.00
  5.47  17.13  28.63  1.00
 21.67  14.36   5.52  1.00
  3.31   0.33  44.08  1.00  End of data points
 10.0                        S, smoothing factor
   7    3.0   21.0
   6    2.0   17.0
```

10.3 Program Results

```
 E02DDF Example Program Results

 Calling with smoothing factor S =   1.0000E+01: NX =   10, NY =    9,
 rank deficiency =    0

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

                4      0.0000          4      0.0000
                5      9.7575          5      9.0008
                6     18.2582          6     20.0000
                7     25.0000
```

```
 The B-spline coefficients:

    58.16    46.31     6.01    32.00     5.86   -23.78
    63.78    46.74    33.37    18.30    14.36    15.95
    40.84   -33.79     5.17    13.10    -4.13    19.37
    75.44   111.92     6.94    17.33     7.09   -13.24
    34.61   -42.61    25.20    -1.96    10.37    -9.09

  Sum of squared residuals FP =   1.0002E+01

 Values of computed spline:

          X    3.00    6.00    9.00   12.00   15.00   18.00   21.00
    Y
17.00       40.74   28.62   19.84   14.29   11.21    9.46    7.09
14.00       48.34   33.97   21.56   14.71   12.32   10.82    7.15
11.00       37.26   24.46   17.21   14.14   13.02   11.23    7.29
 8.00       30.25   19.66   16.90   16.28   15.21   12.71    8.99
 5.00       36.64   26.75   23.07   21.13   18.97   15.90   11.98
 2.00       45.04   33.70   26.25   22.88   21.62   19.39   13.40
```

Example Program
Calculation and Evaluation of Least-squares Bicubic Spline Fit
from Scattered Data