NAG Library Routine Document

E02DFF

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

E02DFF calculates values of a bicubic spline from its B-spline representation. The spline is evaluated at all points on a rectangular grid.

2 Specification

```
SUBROUTINE E02DFF (MX, MY, PX, PY, X, Y, LAMDA, MU, C, FF, WRK, LWRK, IWRK,     &
                   LIWRK, IFAIL)

INTEGER            MX, MY, PX, PY, LWRK, IWRK(LIWRK), LIWRK, IFAIL
REAL (KIND=nag_wp) X(MX), Y(MY), LAMDA(PX), MU(PY), C((PX-4)*(PY-4)),           &
                   FF(MX*MY), WRK(LWRK)
```

3 Description

E02DFF calculates values of the bicubic spline $s(x,y)$ on a rectangular grid of points in the x-y plane, from its augmented knot sets $\{\lambda\}$ and $\{\mu\}$ and from the coefficients c_{ij}, for $i = 1,2,\ldots,\mathrm{PX}-4$ and $j = 1,2,\ldots,\mathrm{PY}-4$, in its B-spline representation

$$s(x,y) = \sum_{ij} c_{ij} M_i(x) N_j(y).$$

Here $M_i(x)$ 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}.

The points in the grid are defined by coordinates x_q, for $q = 1,2,\ldots,m_x$, along the x axis, and coordinates y_r, for $r = 1,2,\ldots,m_y$, along the y axis.

This routine may be used to calculate values of a bicubic spline given in the form produced by E01DAF, E02DAF, E02DCF and E02DDF. It is derived from the routine B2VRE in Anthony *et al.* (1982).

4 References

Anthony G T, Cox M G and Hayes J G (1982) *DASL – Data Approximation Subroutine Library* National Physical Laboratory

Cox M G (1978) The numerical evaluation of a spline from its B-spline representation *J. Inst. Math. Appl.* **21** 135–143

5 Parameters

1: MX – INTEGER *Input*
2: MY – INTEGER *Input*

On entry: MX and MY must specify m_x and m_y respectively, the number of points along the x and y axis that define the rectangular grid.

Constraint: MX ≥ 1 and MY ≥ 1.

3: PX – INTEGER *Input*
4: PY – INTEGER *Input*

On entry: PX and PY must specify the total number of knots associated with the variables x and y respectively. They are such that $\mathrm{PX} - 8$ and $\mathrm{PY} - 8$ are the corresponding numbers of interior knots.

Constraint: $\mathrm{PX} \geq 8$ and $\mathrm{PY} \geq 8$.

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

On entry: X and Y must contain x_q, for $q = 1, 2, \ldots, m_x$, and y_r, for $r = 1, 2, \ldots, m_y$, respectively. These are the x and y coordinates that define the rectangular grid of points at which values of the spline are required.

Constraint: X and Y must satisfy

$$\mathrm{LAMDA}(4) \leq \mathrm{X}(q) < \mathrm{X}(q+1) \leq \mathrm{LAMDA}(\mathrm{PX}-3), \qquad q = 1, 2, \ldots, m_x - 1$$

and

$$\mathrm{MU}(4) \leq \mathrm{Y}(r) < \mathrm{Y}(r+1) \leq \mathrm{MU}(\mathrm{PY}-3), \qquad r = 1, 2, \ldots, m_y - 1.$$

.The spline representation is not valid outside these intervals.

7: LAMDA(PX) – REAL (KIND=nag_wp) array *Input*
8: MU(PY) – REAL (KIND=nag_wp) array *Input*

On entry: LAMDA and MU must contain the complete sets of knots $\{\lambda\}$ and $\{\mu\}$ associated with the x and y variables respectively.

Constraint: the knots in each set must be in nondecreasing order, with $\mathrm{LAMDA}(\mathrm{PX}-3) > \mathrm{LAMDA}(4)$ and $\mathrm{MU}(\mathrm{PY}-3) > \mathrm{MU}(4)$.

9: C((PX – 4) × (PY – 4)) – REAL (KIND=nag_wp) array *Input*

On entry: $\mathrm{C}((\mathrm{PY}-4) \times (i-1) + j)$ must contain the coefficient c_{ij} described in Section 3, for $i = 1, 2, \ldots, \mathrm{PX} - 4$ and $j = 1, 2, \ldots, \mathrm{PY} - 4$.

10: FF(MX × MY) – REAL (KIND=nag_wp) array *Output*

On exit: $\mathrm{FF}(\mathrm{MY} \times (q-1) + r)$ contains the value of the spline at the point (x_q, y_r), for $q = 1, 2, \ldots, m_x$ and $r = 1, 2, \ldots, m_y$.

11: WRK(LWRK) – REAL (KIND=nag_wp) array *Workspace*
12: LWRK – INTEGER *Input*

On entry: the dimension of the array WRK as declared in the (sub)program from which E02DFF is called.

Constraint: $\mathrm{LWRK} \geq \min(4 \times \mathrm{MX} + \mathrm{PX}, 4 \times \mathrm{MY} + \mathrm{PY})$.

13: IWRK(LIWRK) – INTEGER array *Workspace*
14: LIWRK – INTEGER *Input*

On entry: the dimension of the array IWRK as declared in the (sub)program from which E02DFF is called.

Constraints:

if $4 \times \mathrm{MX} + \mathrm{PX} > 4 \times \mathrm{MY} + \mathrm{PY}$, $\mathrm{LIWRK} \geq \mathrm{MY} + \mathrm{PY} - 4$;
otherwise $\mathrm{LIWRK} \geq \mathrm{MX} + \mathrm{PX} - 4$.

15: 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, MX < 1,
or MY < 1,
or PY < 8,
or PX < 8.

IFAIL $= 2$

On entry, LWRK is too small,
or LIWRK is too small.

IFAIL $= 3$

On entry, the knots in array LAMDA, or those in array MU, are not in nondecreasing order, or LAMDA(PX – 3) $\leq$ LAMDA(4), or MU(PY – 3) $\leq$ MU(4).

IFAIL $= 4$

On entry, the restriction LAMDA(4) $\leq$ X(1) $< \cdots <$ X(MX) $\leq$ LAMDA(PX – 3), or the restriction MU(4) $\leq$ Y(1) $< \cdots <$ Y(MY) $\leq$ MU(PY – 3), is violated.

7 Accuracy

The method used to evaluate the B-splines is numerically stable, in the sense that each computed value of $s(x_r, y_r)$ can be regarded as the value that would have been obtained in exact arithmetic from slightly perturbed B-spline coefficients. See Cox (1978) for details.

8 Further Comments

Computation time is approximately proportional to $m_x m_y + 4(m_x + m_y)$.

9 Example

This example reads in knot sets LAMDA(1),...,LAMDA(PX) and MU(1),...,MU(PY), and a set of bicubic spline coefficients c_{ij}. Following these are values for m_x and the x coordinates x_q, for $q = 1, 2, \ldots, m_x$, and values for m_y and the y coordinates y_r, for $r = 1, 2, \ldots, m_y$, defining the grid of points on which the spline is to be evaluated.

9.1 Program Text

```
    Program e02dffe

!     E02DFF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: e02dff, nag_wp, x04cbf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: indent = 0, ncols = 80, nin = 5,     &
                                          nout = 6
      Character (1), Parameter         :: chlabel = 'C', diag = 'N', matrix =  &
                                          'G'
      Character (4), Parameter         :: form = 'F8.3'
!     .. Local Scalars ..
      Integer                          :: ifail, liwrk, lwrk, mx, my, px, py
      Character (48)                   :: title
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: c(:), ff(:), lamda(:), mu(:),        &
                                          wrk(:), x(:), y(:)
      Integer, Allocatable             :: iwrk(:)
      Character (10), Allocatable      :: clabs(:), rlabs(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: min
!     .. Executable Statements ..
      Write (nout,*) 'E02DFF Example Program Results'
      Flush (nout)

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

!     Read PX and PY, the number of knots in the X and Y directions.

      Read (nin,*) px, py
      Allocate (lamda(px),mu(py),c((px-4)*(py-4)))

!     Read the knots LAMDA(1) .. LAMDA(PX) and MU(1) .. MU(PY).

      Read (nin,*) lamda(1:px)
      Read (nin,*) mu(1:py)

!     Read C, the bicubic spline coefficients.

      Read (nin,*) c(1:(px-4)*(py-4))

!     Read MX and MY, the number of grid points in the X and Y
!     directions respectively.

      Read (nin,*) mx, my
      lwrk = min(4*mx+px,4*my+py)
      liwrk = mx + px - 4
      Allocate (clabs(mx),rlabs(my),x(mx),y(my),ff(mx*my),wrk(lwrk), &
        iwrk(liwrk))

!     Read the X and Y co-ordinates defining the evaluation grid.

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

!     Evaluate the spline at the MX by MY points.

      ifail = 0
      Call e02dff(mx,my,px,py,x,y,lamda,mu,c,ff,wrk,lwrk,iwrk,liwrk,ifail)

!     Generate column and row labels to print the results with.

      Write (clabs(1:mx),99999) x(1:mx)
      Write (rlabs(1:my),99999) y(1:my)

      Write (nout,*)
      Flush (nout)

!     Print the result array.
      title = 'Spline evaluated on X-Y grid (X across, Y down):'
      Call x04cbf(matrix,diag,my,mx,ff,my,form,title,chlabel,rlabs,chlabel, &
        clabs,ncols,indent,ifail)

99999 Format ((F5.1))
    End Program e02dffe
```

9.2 Program Data

```
E02DFF Example Program Data
11  10                                                   PX  PY
1.0  1.0  1.0  1.0  1.3  1.5  1.6  2.0  2.0  2.0  2.0   LAMDA(1) .. LAMDA(PX)
0.0  0.0  0.0  0.0  0.4  0.7  1.0  1.0  1.0  1.0        MU(1) .. MU(PY)
1.0000   1.1333   1.3667   1.7000   1.9000   2.0000
1.2000   1.3333   1.5667   1.9000   2.1000   2.2000
1.5833   1.7167   1.9500   2.2833   2.4833   2.5833
2.1433   2.2767   2.5100   2.8433   3.0433   3.1433
2.8667   3.0000   3.2333   3.5667   3.7667   3.8667
3.4667   3.6000   3.8333   4.1667   4.3667   4.4667
4.0000   4.1333   4.3667   4.7000   4.9000   5.0000    Spline coefficients, C
7  6                                                     MX  MY
1.0  1.1  1.3  1.4  1.5  1.7  2.0                        X(1) .. X(MX)
0.0  0.2  0.4  0.6  0.8  1.0                             Y(1) .. Y(MY)
```

9.3 Program Results

```
 E02DFF Example Program Results

 Spline evaluated on X-Y grid (X across, Y down):
          1.0     1.1     1.3     1.4     1.5     1.7     2.0
 0.0    1.000   1.210   1.690   1.960   2.250   2.890   4.000
 0.2    1.200   1.410   1.890   2.160   2.450   3.090   4.200
 0.4    1.400   1.610   2.090   2.360   2.650   3.290   4.400
 0.6    1.600   1.810   2.290   2.560   2.850   3.490   4.600
 0.8    1.800   2.010   2.490   2.760   3.050   3.690   4.800
 1.0    2.000   2.210   2.690   2.960   3.250   3.890   5.000
```