NAG Library Routine Document

E01DAF

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

E01DAF computes a bicubic spline interpolating surface through a set of data values, given on a rectangular grid in the x-y plane.

2 Specification

```
SUBROUTINE E01DAF (MX, MY, X, Y, F, PX, PY, LAMDA, MU, C, WRK, IFAIL)
INTEGER             MX, MY, PX, PY, IFAIL
REAL (KIND=nag_wp) X(MX), Y(MY), F(MX*MY), LAMDA(MX+4), MU(MY+4),        &
                    C(MX*MY), WRK((MX+6)*(MY+6))
```

3 Description

E01DAF determines a bicubic spline interpolant to the set of data points $(x_q, y_r, f_{q,r})$, for $q = 1, 2, \ldots, m_x$ and $r = 1, 2, \ldots, m_y$. The spline is given in the B-spline representation

$$s(x,y) = \sum_{i=1}^{m_x} \sum_{j=1}^{m_y} c_{ij} M_i(x) N_j(y),$$

such that

$$s(x_q, y_r) = f_{q,r},$$

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}, and the c_{ij} are the spline coefficients. These knots, as well as the coefficients, are determined by the routine, which is derived from the routine B2IRE in Anthony *et al.* (1982). The method used is described in Section 9.2.

For further information on splines, see Hayes and Halliday (1974) for bicubic splines and de Boor (1972) for normalized B-splines.

Values and derivatives of the computed spline can subsequently be computed by calling E02DEF, E02DFF or E02DHF as described in Section 9.3.

4 References

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

Cox M G (1975) An algorithm for spline interpolation *J. Inst. Math. Appl.* **15** 95–108

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

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

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: $\text{MX} \geq 4$ and $\text{MY} \geq 4$.

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

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

On entry: $\text{X}(q)$ and $\text{Y}(r)$ must contain x_q, for $q = 1, 2, \ldots, m_x$, and y_r, for $r = 1, 2, \ldots, m_y$, respectively.

Constraints:

$\text{X}(q) < \text{X}(q+1)$, for $q = 1, 2, \ldots, m_x - 1$;
$\text{Y}(r) < \text{Y}(r+1)$, for $r = 1, 2, \ldots, m_y - 1$.

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

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

6: PX – INTEGER *Output*

7: PY – INTEGER *Output*

On exit: PX and PY contain $m_x + 4$ and $m_y + 4$, the total number of knots of the computed spline with respect to the x and y variables, respectively.

8: LAMDA(MX + 4) – REAL (KIND=nag_wp) array *Output*

9: MU(MY + 4) – REAL (KIND=nag_wp) array *Output*

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

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

and

$$\text{LAMDA}(\text{PX}-3) = \text{LAMDA}(\text{PX}-2) = \text{LAMDA}(\text{PX}-1) = \text{LAMDA}(\text{PX}) = \text{X}(\text{MX})$$

needed for the B-spline representation.

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

On exit: the coefficients of the spline interpolant. $\text{C}(m_y \times (i-1) + j)$ contains the coefficient c_{ij} described in Section 3.

11: WRK((MX + 6) × (MY + 6)) – REAL (KIND=nag_wp) array *Workspace*

12: 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: $\text{IFAIL} = 0$ 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 < 4,
or MY < 4.

IFAIL $= 2$

On entry, either the values in the X array or the values in the Y array are not in increasing order if not already there.

IFAIL $= 3$

A system of linear equations defining the B-spline coefficients was singular; the problem is too ill-conditioned to permit solution.

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.

7 Accuracy

The main sources of rounding errors are in steps 2, 3, 6 and 7 of the algorithm described in Section 9.2. It can be shown (see Cox (1975)) that the matrix A_x formed in step 2 has elements differing relatively from their true values by at most a small multiple of 3ϵ, where ϵ is the ***machine precision***. A_x is 'totally positive', and a linear system with such a coefficient matrix can be solved quite safely by elimination without pivoting. Similar comments apply to steps 6 and 7. Thus the complete process is numerically stable.

8 Parallelism and Performance

E01DAF is not threaded by NAG in any implementation.

E01DAF 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 by E01DAF is approximately proportional to $m_x m_y$.

9.2 Outline of Method Used

The process of computing the spline consists of the following steps:

1. choice of the interior x-knots λ_5, $\lambda_6, \ldots, \lambda_{m_x}$ as $\lambda_i = x_{i-2}$, for $i = 5, 6, \ldots, m_x$,
2. formation of the system

$$A_x E = F,$$

   where A_x is a band matrix of order m_x and bandwidth 4, containing in its qth row the values at x_q of the B-splines in x, F is the m_x by m_y rectangular matrix of values $f_{q,r}$, and E denotes an m_x by m_y rectangular matrix of intermediate coefficients,
3. use of Gaussian elimination to reduce this system to band triangular form,
4. solution of this triangular system for E,
5. choice of the interior y knots μ_5, $\mu_6, \ldots, \mu_{m_y}$ as $\mu_i = y_{i-2}$, for $i = 5, 6, \ldots, m_y$,
6. formation of the system

$$A_y C^{\mathrm{T}} = E^{\mathrm{T}},$$

   where A_y is the counterpart of A_x for the y variable, and C denotes the m_x by m_y rectangular matrix of values of c_{ij},
7. use of Gaussian elimination to reduce this system to band triangular form,
8. solution of this triangular system for C^{T} and hence C.

For computational convenience, steps 2 and 3, and likewise steps 6 and 7, are combined so that the formation of A_x and A_y and the reductions to triangular form are carried out one row at a time.

9.3 Evaluation of Computed Spline

The values of the computed spline at the points (x_k, y_k), for $k = 1, 2, \ldots, m$, 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 $\mathtt{M} = m$ and the coordinates x_k, y_k are stored in $\mathtt{X}(k)$, $\mathtt{Y}(k)$. PX and PY, LAMDA, MU and C have the same values as PX and PY LAMDA, MU and C output from E01DAF. WRK is a real workspace array of length at least PY, and IWRK is an integer workspace array of length at least $\mathtt{PY} - 4$. (See E02DEF.)

To evaluate the computed spline on an m_x by m_y rectangular grid of points in the x-y plane, which is defined by the x coordinates stored in $\mathtt{X}(j)$, for $j = 1, 2, \ldots, m_x$, and the y coordinates stored in $\mathtt{Y}(k)$, for $k = 1, 2, \ldots, m_y$, 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(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, LAMDA, MU and C have the same values as PX, PY, LAMDA, MU and C output from E01DAF. WRK is a real workspace array of length at least $\mathtt{LWRK} = \min(nwrk1, nwrk2)$, for $nwrk1 = \mathtt{MX} \times 4 + \mathtt{PX}$, $nwrk2 = \mathtt{MY} \times 4 + \mathtt{PY}$, and IWRK is an integer workspace array of length at least $\mathtt{LIWRK} = \mathtt{MY} + \mathtt{PY} - 4$ if $nwrk1 > nwrk2$, or $\mathtt{MX} + \mathtt{PX} - 4$ otherwise.

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

10 Example

This example reads in values of m_x, x_q, for $q = 1, 2, \ldots, m_x$, m_y and y_r, for $r = 1, 2, \ldots, m_y$, followed by values of the ordinates $f_{q,r}$ defined at the grid points (x_q, y_r).

It then calls E01DAF to compute a bicubic spline interpolant of the data values, and prints the values of the knots and B-spline coefficients. Finally it evaluates the spline at a small sample of points on a rectangular grid.

10.1 Program Text

```
    Program e01dafe

!     E01DAF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: e01daf, 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 ..
      Real (Kind=nag_wp)              :: step, xhi, xlo, yhi, ylo
      Integer                         :: i, ifail, j, liwrk, lwrk, mx, my,    &
                                         nx, ny, px, py
      Character (54)                  :: title
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable :: c(:), f(:,:), fg(:), lamda(:),       &
                                         mu(:), tx(:), ty(:), wrk(:), x(:),   &
                                         y(:)
      Integer, Allocatable            :: iwrk(:)
      Character (10), Allocatable     :: clabs(:), rlabs(:)
!     .. Intrinsic Procedures ..
      Intrinsic                       :: min, real
!     .. Executable Statements ..
      Write (nout,*) 'E01DAF Example Program Results'

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

!     Read the number of X points, MX, and the values of the
!     X co-ordinates.

      Read (nin,*) mx
      Allocate (x(mx),lamda(mx+4))

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

!     Read the number of Y points, MY, and the values of the
!     Y co-ordinates.

      Read (nin,*) my
      Allocate (y(my),mu(my+4),c(mx*my),f(my,mx),wrk((mx+6)*(my+6)))

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

!     Read the function values at the grid points.

      Do j = 1, my
        Read (nin,*) f(j,1:mx)
      End Do

!     Generate the (X,Y,F) interpolating bicubic B-spline.
```

```
      ifail = 0
      Call e01daf(mx,my,x,y,f,px,py,lamda,mu,c,wrk,ifail)

!     Print the knot sets, LAMDA and MU.

      Write (nout,*)
      Write (nout,*) '               I   Knot LAMDA(I)      J     Knot MU(J)'
      Write (nout,99997)(j,lamda(j),j,mu(j),j=4,min(px,py)-3)
      If (px>py) Then
        Write (nout,99997)(j,lamda(j),j=py-2,px-3)
      Else If (px<py) Then
        Write (nout,99996)(j,mu(j),j=px-2,py-3)
      End If

!     Print the spline coefficients.

      Write (nout,*)
      Write (nout,*) 'The B-Spline coefficients:'
      Write (nout,99999)(c(i),i=1,mx*my)
      Write (nout,*)
      Flush (nout)

!     Evaluate the spline on a regular rectangular grid at nx*ny points
!     over the domain [xlo,xhi] x [ylo,yhi].

      Deallocate (wrk)

      Read (nin,*) nx, xlo, xhi
      Read (nin,*) ny, ylo, yhi
      lwrk = min(4*nx+px,4*ny+py)

      If (4*nx+px>4*ny+py) Then
        liwrk = ny + py - 4
      Else
        liwrk = nx + px - 4
      End If

      Allocate (tx(nx),ty(ny),fg(nx*ny),wrk(lwrk),iwrk(liwrk))

!     Generate nx/ny equispaced x/y co-ordinates.

      step = (xhi-xlo)/real(nx-1,kind=nag_wp)
      tx(1) = xlo
      Do i = 2, nx - 1
        tx(i) = tx(i-1) + step
      End Do
      tx(nx) = xhi

      step = (yhi-ylo)/real(ny-1,kind=nag_wp)
      ty(1) = ylo
      Do i = 2, ny - 1
        ty(i) = ty(i-1) + step
      End Do
      ty(ny) = yhi

!     Evaluate the spline.
      ifail = 0
      Call e02dff(nx,ny,px,py,tx,ty,lamda,mu,c,fg,wrk,lwrk,iwrk,liwrk,ifail)

!     Generate row and column labels and title for printing results.
      Allocate (clabs(nx),rlabs(ny))
      Do i = 1, nx
        Write (clabs(i),99998) tx(i)
      End Do
      Do i = 1, ny
        Write (rlabs(i),99998) ty(i)
        Flush (nout)
      End Do
      title = 'Spline evaluated on a regular mesh (X across, Y down):'
```

```
!     Print the results.
      ifail = 0
      Call x04cbf(matrix,diag,ny,nx,fg,ny,form,title,chlabel,rlabs,chlabel, &
        clabs,ncols,indent,ifail)

99999 Format (1X,8F9.4)
99998 Format (F5.2)
99997 Format (1X,I16,F12.4,I11,F12.4)
99996 Format (1X,I39,F12.4)
    End Program e01dafe
```

10.2 Program Data

```
E01DAF Example Program Data
  7                                               MX
  1.00  1.10  1.30  1.50  1.60  1.80  2.00        X(1) .. X(MX)
  6                                               MY
  0.00  0.10  0.40  0.70  0.90  1.00              Y(1) .. Y(MY)
  1.00  1.21  1.69  2.25  2.56  3.24  4.00        (F(MY*(I-1)+J),I=1..MX),J=1..MY
  1.10  1.31  1.79  2.35  2.66  3.34  4.10
  1.40  1.61  2.09  2.65  2.96  3.64  4.40
  1.70  1.91  2.39  2.95  3.26  3.94  4.70
  1.90  2.11  2.59  3.15  3.46  4.14  4.90
  2.00  2.21  2.69  3.25  3.56  4.24  5.00
  6  1.0  2.0                                     NX  XLO  XHI
  6  0.0  1.0                                     NY  YLO  YHI
```

10.3 Program Results

```
 E01DAF Example Program Results

               I   Knot LAMDA(I)      J     Knot MU(J)
               4      1.0000          4      0.0000
               5      1.3000          5      0.4000
               6      1.5000          6      0.7000
               7      1.6000          7      1.0000
               8      2.0000

 The B-Spline coefficients:
    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 evaluated on a regular mesh (X across, Y down):
         1.00     1.20     1.40     1.60     1.80     2.00
 0.00    1.000    1.440    1.960    2.560    3.240    4.000
 0.20    1.200    1.640    2.160    2.760    3.440    4.200
 0.40    1.400    1.840    2.360    2.960    3.640    4.400
 0.60    1.600    2.040    2.560    3.160    3.840    4.600
 0.80    1.800    2.240    2.760    3.360    4.040    4.800
 1.00    2.000    2.440    2.960    3.560    4.240    5.000
```