NAG Library Routine Document

E02DHF

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

E02DHF computes the partial derivative (of order ν_x, ν_y), of a bicubic spline approximation to a set of data values, from its B-spline representation, at points on a rectangular grid in the x-y plane. This routine may be used to calculate derivatives of a bicubic spline given in the form produced by E01DAF, E02DAF, E02DCF and E02DDF.

2 Specification

```
SUBROUTINE E02DHF (MX, MY, PX, PY, X, Y, LAMDA, MU, C, NUX, NUY, Z,        &
                   IFAIL)

INTEGER            MX, MY, PX, PY, NUX, NUY, IFAIL
REAL (KIND=nag_wp) X(MX), Y(MY), LAMDA(PX), MU(PY), C((PX-4)*(PY-4)),      &
                   Z(MX*MY)
```

3 Description

E02DHF determines the partial derivative $\frac{\partial^{\nu_x+\nu_y}}{\partial x^{\nu_x}\partial y^{\nu_y}}$ of a smooth bicubic spline approximation $s(x,y)$ at the set of data points (x_q, y_r).

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}, with n_x and n_y the total numbers of knots of the computed spline with respect to the x and y variables respectively. For further details, see Hayes and Halliday (1974) for bicubic splines and de Boor (1972) for normalized B-splines. This routine is suitable for B-spline representations returned by E01DAF, E02DAF, E02DCF and E02DDF.

The partial derivatives can be up to order 2 in each direction; thus the highest mixed derivative available is $\frac{\partial^4}{\partial x^2 \partial y^2}$.

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.

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: MX – INTEGER *Input*

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

Constraint: MX ≥ 1.

2: MY – INTEGER *Input*

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

Constraint: MY ≥ 1.

3: PX – INTEGER *Input*

On entry: the total number of knots in the x-direction of the bicubic spline approximation, e.g., the value NX as returned by E02DCF.

4: PY – INTEGER *Input*

On entry: the total number of knots in the y-direction of the bicubic spline approximation, e.g., the value NY as returned by E02DCF.

5: 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 $q = 1, 2, \ldots, m_x$, on which values of the partial derivative are sought.

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

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

On entry: Y(r) must be set to y_r, the y coordinate of the rth grid point along the y axis, for $r = 1, 2, \ldots, m_y$ on which values of the partial derivative are sought.

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

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

On entry: contains the position of the knots in the x-direction of the bicubic spline approximation to be differentiated, e.g., LAMDA as returned by E02DCF.

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

On entry: contains the position of the knots in the y-direction of the bicubic spline approximation to be differentiated, e.g., MU as returned by E02DCF.

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

On entry: the coefficients of the bicubic spline approximation to be differentiated, e.g., C as returned by E02DCF.

10: NUX – INTEGER *Input*

On entry: specifies the order, ν_x of the partial derivative in the x-direction.

Constraint: $0 \leq$ NUX ≤ 2.

11: NUY – INTEGER *Input*

On entry: specifies the order, ν_y of the partial derivative in the y-direction.

Constraint: $0 \leq$ NUY ≤ 2.

12: Z(MX × MY) – REAL (KIND=nag_wp) array *Output*

On exit: Z$(m_y \times (q-1)+r)$ contains the derivative $\frac{\partial^{\nu_x+\nu_y}}{\partial x^{\nu_x}\partial y^{\nu_y}}s\left(x_q,y_r\right)$, for $q=1,2,\ldots,m_x$ and $r=1,2,\ldots,m_y$.

13: 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, NUX $= \langle value \rangle$.
Constraint: $0 \leq$ NUX ≤ 2.

IFAIL $= 2$

On entry, NUY $= \langle value \rangle$.
Constraint: $0 \leq$ NUY ≤ 2.

IFAIL $= 3$

On entry, MX $= \langle value \rangle$.
Constraint: MX ≥ 1.

IFAIL $= 4$

On entry, MY $= \langle value \rangle$.
Constraint: MY ≥ 1.

IFAIL $= 5$

On entry, for $i = \langle value \rangle$, X$(i-1) = \langle value \rangle$ and X$(i) = \langle value \rangle$.
Constraint: X$(i-1) \leq$ X(i), for $i = 2,3,\ldots,$MX.

IFAIL $= 6$

On entry, for $i = \langle value \rangle$, Y$(i-1) = \langle value \rangle$ and Y$(i) = \langle value \rangle$.
Constraint: Y$(i-1) \leq$ Y(i), for $i = 2,3,\ldots,$MY.

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

On successful exit, the partial derivatives on the given mesh are accurate to ***machine precision*** with respect to the supplied bicubic spline. Please refer to Section 7 in E01DAF, E02DAF, E02DCF and E02DDF of the routine document for the respective routine which calculated the spline approximant for details on the accuracy of that approximation.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

This example reads in values of m_x, m_y, x_q, for $q = 1, 2, \ldots, m_x$, 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 E02DCF to compute a bicubic spline approximation for one specified value of S. Finally it evaluates the spline and its first x derivative at a small sample of points on a rectangular grid by calling E02DHF.

10.1 Program Text

```
!   E02DHF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.
    Module e02dhfe_mod

!     E02DHF 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                                :: print_spline
!     .. Parameters ..
      Integer, Parameter, Public            :: nin = 5, nout = 6
    Contains
      Subroutine print_spline(ngx,gridx,ngy,gridy,z,zder)

!       Print spline function and spline derivative evaluation

!       .. Use Statements ..
        Use nag_library, Only: x04cbf
!       .. Parameters ..
        Integer, Parameter                  :: indent = 0, ncols = 80
        Character (1), Parameter            :: chlabel = 'C', diag = 'N',     &
                                               matrix = 'G'
        Character (4), Parameter            :: form = 'F8.3'
```

```
!       .. Scalar Arguments ..
        Integer, Intent (In)                 :: ngx, ngy
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: gridx(ngx), gridy(ngy),        &
                                                z(ngx*ngy), zder(ngx*ngy)
!       .. Local Scalars ..
        Integer                              :: i, ifail
        Character (48)                       :: title
!       .. Local Arrays ..
        Character (10), Allocatable          :: clabs(:), rlabs(:)
!       .. Executable Statements ..
!       Allocate for row and column label
        Allocate (clabs(ngx),rlabs(ngy))
!       Generate column and row labels to print the results with.
        Do i = 1, ngx
          Write (clabs(i),99999) gridx(i)
        End Do
        Do i = 1, ngy
          Write (rlabs(i),99999) gridy(i)
        End Do

!       Print the spline evaluations.
        title = 'Spline evaluated on X-Y grid (X across, Y down):'
        Write (nout,*)
        Flush (nout)
        ifail = 0
        Call x04cbf(matrix,diag,ngy,ngx,z,ngy,form,title,chlabel,rlabs, &
          chlabel,clabs,ncols,indent,ifail)

!       Print the spline derivative evaluations.
        title = 'Spline derivative evaluated on X-Y grid:'
        Write (nout,*)
        Flush (nout)
        ifail = 0
        Call x04cbf(matrix,diag,ngy,ngx,zder,ngy,form,title(1:40),chlabel, &
          rlabs,chlabel,clabs,ncols,indent,ifail)

        Deallocate (clabs,rlabs)

99999   Format (F5.2)
      End Subroutine print_spline
    End Module e02dhfe_mod
    Program e02dhfe

!     E02DHF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e02dcf, e02dhf, nag_wp
      Use e02dhfe_mod, Only: nin, nout, print_spline
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                     :: delta, fp, s, xhi, xlo, yhi, ylo
      Integer                                :: i, ifail, liwrk, lwrk, mx, my,   &
                                                nc, ngx, ngy, nux, nuy, nx,      &
                                                nxest, ny, nyest
      Character (1)                          :: start
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable        :: c(:), f(:), gridx(:), gridy(:),  &
                                                lamda(:), mu(:), wrk(:), x(:),   &
                                                y(:), z(:), zder(:)
      Integer, Allocatable                   :: iwrk(:)
!     .. Intrinsic Procedures ..
      Intrinsic                              :: max, real
!     .. Executable Statements ..
      Write (nout,*) 'E02DHF 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
      nc = (nxest-4)*(nyest-4)

!     Allocations for spline fit
      Allocate (lamda(nxest),mu(nyest),c(nc))

!     Allocations for e02dcf only
      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),wrk(lwrk),iwrk(liwrk))

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

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

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

!     Determine the spline approximation.
      start = 'C'
      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 (x,y,f,wrk,iwrk)

      Write (nout,*)
      Write (nout,99999) 'Spline fit used smoothing factor S =', s, '.'
      Write (nout,99998) 'Number of knots in each direction  =', nx, ny
      Write (nout,*)
      Write (nout,99999) 'Sum of squared residuals           =', fp, '.'

!     Spline and its derivative to be evaluated on rectangular grid with
!     ngx*ngy points on the domain [xlo,xhi]x[ylo,yhi].

      Read (nin,*) ngx, xlo, xhi
      Read (nin,*) ngy, ylo, yhi

!     Allocations for e02dhf (spline evaluation).
      Allocate (gridx(ngx),gridy(ngy),z(ngx*ngy),zder(ngx*ngy))

      delta = (xhi-xlo)/real(ngx-1,kind=nag_wp)
      gridx(1) = xlo
      Do i = 2, ngx - 1
        gridx(i) = gridx(i-1) + delta
      End Do
      gridx(ngx) = xhi

      delta = (yhi-ylo)/real(ngy-1,kind=nag_wp)
      gridy(1) = ylo
      Do i = 2, ngy - 1
        gridy(i) = gridy(i-1) + delta
      End Do
      gridy(ngy) = yhi

!     Evaluate spline (nux=nuy=0)
      nux = 0
      nuy = 0
      ifail = 0
      Call e02dhf(ngx,ngy,nx,ny,gridx,gridy,lamda,mu,c,nux,nuy,z,ifail)
!     Evaluate spline partial derivative of order (nux,nuy)
      Read (nin,*) nux, nuy
      Write (nout,*)
      Write (nout,99998) 'Derivative of spline has order nux, nuy =', nux, nuy
      ifail = 0
      Call e02dhf(ngx,ngy,nx,ny,gridx,gridy,lamda,mu,c,nux,nuy,zder,ifail)
```

```
!     Print tabulated spline and derivative evaluations.
      Call print_spline(ngx,gridx,ngy,gridy,z,zder)

99999 Format (1X,A,1P,E13.4,A)
99998 Format (1X,A,I5,',',I5,'.')
    End Program e02dhfe
```

10.2 Program Data

```
E02DHF Example Program Data
 11    9                                                      : MX, MY
 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                                                  : X(1:MX)
 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              : Y(1:MY)
 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              : F(1:MX*MY)
 0.1                                                         : S
 6    0.0   5.0                                             : NGX, XLO, XHI
 5    0.0   4.0                                             : NGY, YLO, YHI
 1    0                                                      : NUX, NUY
```

10.3 Program Results

```
 E02DHF Example Program Results

 Spline fit used smoothing factor S =   1.0000E-01.
 Number of knots in each direction  =   10,   13.

 Sum of squared residuals           =   1.0004E-01.

 Derivative of spline has order nux, nuy =    1,    0.

 Spline evaluated on X-Y grid (X across, Y down):
          0.00    1.00    2.00    3.00    4.00    5.00
 0.00    0.992   2.043   3.029   4.014   5.021   5.997
 1.00    0.541   1.088   1.607   2.142   2.705   3.239
 2.00   -0.417  -0.829  -1.241  -1.665  -2.083  -2.485
 3.00   -0.978  -1.975  -2.914  -3.913  -4.965  -5.924
 4.00   -0.648  -1.363  -1.991  -2.606  -3.251  -3.933

 Spline derivative evaluated on X-Y grid:
          0.00    1.00    2.00    3.00    4.00    5.00
 0.00    1.093   1.013   0.970   1.004   1.001   0.939
 1.00    0.565   0.531   0.515   0.558   0.559   0.499
 2.00   -0.429  -0.404  -0.421  -0.423  -0.412  -0.389
 3.00   -1.060  -0.951  -0.949  -1.048  -1.031  -0.861
 4.00   -0.779  -0.661  -0.608  -0.628  -0.663  -0.701
```