NAG Library Routine Document

E01BFF

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

E01BFF evaluates a piecewise cubic Hermite interpolant at a set of points.

2 Specification

```
SUBROUTINE E01BFF (N, X, F, D, M, PX, PF, IFAIL)
INTEGER                 N, M, IFAIL
REAL (KIND=nag_wp) X(N), F(N), D(N), PX(M), PF(M)
```

3 Description

E01BFF evaluates a piecewise cubic Hermite interpolant, as computed by E01BEF, at the points PX(i), for $i = 1, 2, \ldots, m$. If any point lies outside the interval from X(1) to X(N), a value is extrapolated from the nearest extreme cubic, and a warning is returned.

The routine is derived from routine PCHFE in Fritsch (1982).

4 References

Fritsch F N (1982) PCHIP final specifications *Report UCID-30194* Lawrence Livermore National Laboratory

5 Parameters

1: N – INTEGER *Input*

2: X(N) – REAL (KIND=nag_wp) array *Input*

3: F(N) – REAL (KIND=nag_wp) array *Input*

4: D(N) – REAL (KIND=nag_wp) array *Input*

On entry: N, X, F and D must be unchanged from the previous call of E01BEF.

5: M – INTEGER *Input*

On entry: m, the number of points at which the interpolant is to be evaluated.

Constraint: M ≥ 1.

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

On entry: the m values of x at which the interpolant is to be evaluated.

7: PF(M) – REAL (KIND=nag_wp) array *Output*

On exit: PF(i) contains the value of the interpolant evaluated at the point PX(i), for $i = 1, 2, \ldots, m$.

8: 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, N < 2.

IFAIL $= 2$

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

IFAIL $= 3$

On entry, M < 1.

IFAIL $= 4$

At least one of the points PX(i), for $i = 1, 2, \ldots,$ M, lies outside the interval [X(1), X(N)], and extrapolation was performed at all such points. Values computed at such points may be very unreliable.

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 computational errors in the array PF should be negligible in most practical situations.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by E01BFF is approximately proportional to the number of evaluation points, m. The evaluation will be most efficient if the elements of PX are in nondecreasing order (or, more generally, if they are grouped in increasing order of the intervals $[\mathrm{X}(r-1), \mathrm{X}(r)]$). A single call of E01BFF with $m > 1$ is more efficient than several calls with $m = 1$.

10 Example

This example reads in values of N, X, F and D, and then calls E01BFF to evaluate the interpolant at equally spaced points.

10.1 Program Text

```
    Program e01bffe

!     E01BFF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: e01bff, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: step
      Integer                          :: i, ifail, m, n, r
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: d(:), f(:), pf(:), px(:), x(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: min, real
!     .. Executable Statements ..
      Write (nout,*) 'E01BFF Example Program Results'

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

      Read (nin,*) n
      Allocate (d(n),f(n),x(n))

      Do r = 1, n
        Read (nin,*) x(r), f(r), d(r)
      End Do

      Read (nin,*) m
      Allocate (pf(m),px(m))

!     Compute M equally spaced points from X(1) to X(N).

      step = (x(n)-x(1))/real(m-1,kind=nag_wp)

      Do i = 1, m
        px(i) = min(x(1)+real(i-1,kind=nag_wp)*step,x(n))
      End Do

      ifail = 0
      Call e01bff(n,x,f,d,m,px,pf,ifail)

      Write (nout,*)
      Write (nout,*) '                  Interpolated'
      Write (nout,*) '       Abscissa          Value'

      Do i = 1, m
```

```
        Write (nout,99999) px(i), pf(i)
      End Do

99999 Format (1X,3F15.4)
    End Program e01bffe
```

10.2 Program Data

```
E01BFF Example Program Data
   9                              N, the number of data points
  7.990  0.00000E+0  0.00000E+0  X(R), F(R), D(R)
  8.090  0.27643E-4  5.52510E-4
  8.190  0.43749E-1  0.33587E+0
  8.700  0.16918E+0  0.34944E+0
  9.200  0.46943E+0  0.59696E+0
  10.00  0.94374E+0  6.03260E-2
  12.00  0.99864E+0  8.98335E-4
  15.00  0.99992E+0  2.93954E-5
  20.00  0.99999E+0  0.00000E+0  End of data points
   11                            M, the number of evaluation points
```

10.3 Program Results

```
 E01BFF Example Program Results

                  Interpolated
        Abscissa          Value
          7.9900         0.0000
          9.1910         0.4640
         10.3920         0.9645
         11.5930         0.9965
         12.7940         0.9992
         13.9950         0.9998
         15.1960         0.9999
         16.3970         1.0000
         17.5980         1.0000
         18.7990         1.0000
         20.0000         1.0000
```