nag_monotonic_evaluate (e01bfc)

1. Purpose

nag_monotonic_evaluate (e01bfc) evaluates a piecewise cubic Hermite interpolant at a set of points.

2. Specification

```
#include <nag.h>
#include <nage01.h>

void nag_monotonic_evaluate(Integer n, double x[], double f[],
     double d[], Integer m, double px[], double pf[], NagError *fail)
```

3. Description

A piecewise cubic Hermite interpolant, as computed by nag_monotonic_interpolant (e01bec), is evaluated at the points $\mathbf{px}[i]$, for $i = 0, 1, \ldots, m-1$. If any point lies outside the interval from $\mathbf{x}[0]$ to $\mathbf{x}[n-1]$, a value is extrapolated from the nearest extreme cubic, and a warning is returned.

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

4. Parameters

n
x[n]
f[n]
d[n]
Input: **n**, **x**, **f** and **d** must be unchanged from the previous call of nag_monotonic_interpolant (e01bec).

m
Input: m, the number of points at which the interpolant is to be evaluated.
Constraint: $\mathbf{m} \geq 1$.

px[m]
Input: the m values of x at which the interpolant is to be evaluated.

pf[m]
Output: $\mathbf{pf}[i]$ contains the value of the interpolant evaluated at the point $\mathbf{px}[i]$, for $i = 0, 1 \ldots, m-1$.

fail
The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

NE_INT_ARG_LT
On entry, **n** must not be less than 2: $\mathbf{n} = \langle value \rangle$.
On entry, **m** must not be less than 1: $\mathbf{m} = \langle value \rangle$.

NE_NOT_MONOTONIC
On entry, $\mathbf{x}[r-1] \geq \mathbf{x}[\mathrm{r}]$ for r = ⟨*value*⟩: $\mathbf{x}[r-1], \mathbf{x}[r] = \langle values \rangle$.
The values of $\mathbf{x}[r]$, for $r = 0, 1, \ldots, n-1$, are not in strictly increasing order.

NW_EXTRAPOLATE
Warning – some points in array PX lie outside the range $\mathbf{x}[0] \ldots \mathbf{x}[n-1]$. Values at these points are unreliable as they have been computed by extrapolation.

6. Further Comments

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

6.1. Accuracy

The computational errors in the array **pf** should be negligible in most practical situations.

6.2. References

Fritsch F N (August 1982) *PCHIP Final Specifications* Lawrence Livermore National Laboratory report UCID-30194.

7. See Also

nag_monotonic_interpolant (e01bec)

8. Example

This example program reads in values of **n**, **x**, **f**, **d** and **m**, and then calls nag_monotonic_evaluate to evaluate the interpolant at equally spaced points.

8.1. Program Text

```
/* nag_monotonic_evaluate(e01bfc) Example Program
 *
 * Copyright 1990 Numerical Algorithms Group
 *
 * Mark 2 revised, 1992.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nage01.h>

#define MMAX 50
#define NMAX 50

main()
{
  Integer i, m, n, r;
  double step, d[NMAX], f[NMAX], pf[MMAX], px[MMAX], x[NMAX];
  static NagError fail;

  fail.print = TRUE;
  Vprintf("e01bfc Example Program Results\n");
  Vscanf("%*[^\n]"); /* Skip to end of line */
  Vscanf("%ld",&n);
  if (n>=1 && n<NMAX)
    {
      for (r = 0; r < n; Vscanf("%lf%lf%lf",&x[r],&f[r],&d[r]), r++);
      Vscanf("%ld",&m);
      if (m>=1 && m<MMAX)
        {
          /* Compute M Equally spaced points from x[0] to x[n-1]. */
          step = (x[n-1] - x[0]) / (double)(m-1);
          for (i = 0; i < m; i++)
            px[i] = MIN(x[0]+ i*step,x[n-1]);
          e01bfc(n, x, f, d, m, px, pf, &fail);
          Vprintf("                 Interpolated\n");
          Vprintf("      Abscissa      Value\n");
          for (i = 0; i < m; i++)
            Vprintf("%13.4f%13.4f\n", px[i], pf[i]);
        }
      exit(EXIT_SUCCESS);
    }
  else
    {
      Vfprintf(stderr, "n is out of range: n = %5ld\n", n);
      exit(EXIT_FAILURE);
    }
}
```

8.2. Program Data

```
e01bfc Example Program Data
   9
  7.990  0.00000E+0  0.00000E+0
  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
   11
```

8.3. Program Results

```
e01bfc 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
```