nag_monotonic_deriv (e01bgc)

1. Purpose

nag_monotonic_deriv (e01bgc) evaluates a piecewise cubic Hermite interpolant and its first derivative at a set of points.

2. Specification

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

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

3. Description

This function evaluates a piecewise cubic Hermite interpolant, as computed by the NAG function nag_monotonic_interpolant (e01bec), at the points **px**$[i]$, for $i = 0, 1, \ldots, m-1$. The first derivatives at the points are also computed. If any point lies outside the interval from **x**$[0]$ to **x**$[n-1]$, values of the interpolant and its derivative are extrapolated from the nearest extreme cubic, and a warning is returned.

If values of the interpolant only, and not of its derivative, are required, nag_monotonic_evaluate (e01bfc) should be used.

The routine is derived from routine PCHFD 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: **m** $\geq$ 1.

px[m]

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

pf[m]

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

pd[m]

Output: **pd**$[i]$ contains the first derivative of the interpolant evaluated at the point **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: **n** = ⟨*value*⟩.
On entry, **m** must not be less than 1: **m** = ⟨*value*⟩.

NE_NOT_MONOTONIC

On entry, **x**$[r-1] \geq$ **x**$[r]$ for r = ⟨*value*⟩: **x**$[r-1]$ = ⟨*value*⟩, **x**$[r]$ = ⟨*value*⟩.
The values of **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_deriv with $m > 1$ is more efficient than several calls with $m = 1$.

6.1. Accuracy

The computational errors in the arrays **pf** and **pd** 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)
nag_monotonic_evaluate (e01bfc)

8. Example

This example program reads in values of **n**, **x**, **f** and **d** and calls nag_monotonic_deriv to compute the values of the interpolant and its derivative at equally spaced points.

8.1. Program Text

```
/* nag_monotonic_deriv(e01bgc) Example Program
 *
 * Copyright 1991 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 */

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

#define MMAX 21
#define NMAX 50

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

  Vprintf("e01bgc Example Program Results\n");
  Vscanf("%*[^\n]");  /* Skip heading in data file */
  Vscanf("%ld",&n);
  if (n>0 && n<=NMAX)
    {
      for (r=0; r<n; r++)
        Vscanf("%lf%lf%lf",&x[r], &f[r], &d[r]);
      Vscanf("%ld",&m);
      if (m>0 && 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]);
          e01bgc(n, x, f, d, m, px, pf, pd, NAGERR_DEFAULT);
          Vprintf("                           Interpolated");
```

```
              Vprintf("      Interpolated\n");
              Vprintf("       Abscissa                Value");
              Vprintf("           Derivative\n");
              for (i=0; i<m; i++)
                Vprintf("%15.4f      %15.4f      %15.3e\n",px[i],pf[i],pd[i]);
              exit(EXIT_SUCCESS);
            }
          else
            {
              Vfprintf(stderr, "m is out of range: m = %ld\n",m);
              exit(EXIT_FAILURE);
            }
        }
      else
        {
          Vfprintf(stderr, "n is out of range: n = %ld\n",n);
          exit(EXIT_FAILURE);
        }
    }
```

8.2. Program Data

```
e01bgc 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

```
e01bgc Example Program Results
                               Interpolated      Interpolated
       Abscissa                   Value           Derivative
         7.9900                  0.0000            0.000e+00
         9.1910                  0.4640            6.060e-01
        10.3920                  0.9645            4.569e-02
        11.5930                  0.9965            9.917e-03
        12.7940                  0.9992            6.249e-04
        13.9950                  0.9998            2.708e-04
        15.1960                  0.9999            2.809e-05
        16.3970                  1.0000            2.034e-05
        17.5980                  1.0000            1.308e-05
        18.7990                  1.0000            6.297e-06
        20.0000                  1.0000           -7.627e-22
```