# nag_monotonic_interpolant (e01bec)

## 1. Purpose

**nag_monotonic_interpolant (e01bec)** computes a monotonicity-preserving piecewise cubic Hermite interpolant to a set of data points.

## 2. Specification

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

void nag_monotonic_interpolant(Integer n, double x[], double f[],
     double d[], NagError *fail)
```

## 3. Description

This function estimates first derivatives at the set of data points $(x_r, f_r)$, for $r = 0, 1, \ldots, n-1$, which determine a piecewise cubic Hermite interpolant to the data, that preserves monotonicity over ranges where the data points are monotonic. If the data points are only piecewise monotonic, the interpolant will have an extremum at each point where monotonicity switches direction. The estimates of the derivatives are computed by a formula due to Brodlie, which is described in Fritsch and Butland (1984), with suitable changes at the boundary points.

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

Values of the computed interpolant can subsequently be computed by calling nag_monotonic_evaluate (e01bfc).

## 4. Parameters

**n**
Input: $n$, the number of data points.
Constraint: $\mathbf{n} \geq 2$.

**x[n]**
Input: $\mathbf{x}[r]$ must be set to $x_r$, the $r$th value of the independent variable (abscissa), for $r = 0, 1 \ldots, n-1$.
Constraint: $\mathbf{x}[r] < \mathbf{x}[r+1]$.

**f[n]**
Input: $\mathbf{f}[r]$ must be set to $f_r$, the $r$th value of the dependent variable (ordinate), for $r = 0, 1, \ldots, n-1$.

**d[n]**
Output: estimates of derivatives at the data points. $\mathbf{d}[r]$ contains the derivative at $\mathbf{x}[r]$.

**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: $\mathbf{n} = \langle value \rangle$.

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

## 6. Further Comments

The time taken by the function is approximately proportional to $n$.

The values of the computed interpolant at the points px[$i$], for $i = 0, 1, \ldots, m-1$, may be obtained in the real array pf, of length at least $m$, by the call:

```
e01bfc(n,x,f,d,m,px,pf,&fail)
```

where **n**, **x** and **f** are the input parameters to nag_monotonic_interpolant and **d** is the output parameter from nag_monotonic_interpolant.

### 6.1. Accuracy

The computational errors in the array **d** 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.

Fritsch F N and Butland J (1984) A method for constructing local monotone piecewise cubic interpolants *SIAM J. Sci. Stat. Comput.* **5** 300–304.

## 7. See Also

nag_monotonic_evaluate (e01bfc)

## 8. Example

This example program reads in a set of data points, calls nag_monotonic_interpolant to compute a piecewise monotonic interpolant, and then calls nag_monotonic_evaluate (e01bfc) to evaluate the interpolant at equally spaced points.

### 8.1. Program Text

```
/* nag_monotonic_interpolant(e01bec) 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("e01bec Example Program Results\n");
  Vscanf("%*[^\n]"); /* Skip to end of line */
  Vscanf("%ld",&n);
  if (n>=1 && n<NMAX)
    {
      for (r = 0; r < n; r++)
        Vscanf("%lf%lf",&x[r],&f[r]);
      /* Abort on error in e01bec */
      e01bec(n, x, f, d, NAGERR_DEFAULT);
      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

```
e01bec Example Program Data
   9
  7.99   0.00000E+0
  8.09   0.27643E-4
  8.19   0.43750E-1
  8.70   0.16918E+0
  9.20   0.46943E+0
 10.00   0.94374E+0
 12.00   0.99864E+0
 15.00   0.99992E+0
 20.00   0.99999E+0
  11
```

### 8.3. Program Results

```
e01bec 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
```