1. Purpose

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

2. Specification

```c
#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: \(n \geq 2\).

**x[n]**

Input: \(x_r\) must be set to \(x_r\), the \(r\)th value of the independent variable (abscissa), for \(r = 0, 1, \ldots, n - 1\).

Constraint: \(x_r < x_{r+1}\).

**f[n]**

Input: \(f_r\) 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. \(d_r\) contains the derivative at \(x_r\).

**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: \(n = \langle\text{value}\rangle\).

**NE_NOT_MONOTONIC**

On entry, \(x_{r-1} \geq x_r\) for \(r = \langle\text{value}\rangle\): \(x_{r-1}, x_r = \langle\text{values}\rangle\).

The values of \(x_r\), 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:
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.


7. See Also

nag_monotonic_eval (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_eval (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("%*
"); /* 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

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

Example Program Results

<table>
<thead>
<tr>
<th>Interpolated</th>
<th>Abscissa</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.9900</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>9.1910</td>
<td>0.4640</td>
<td></td>
</tr>
<tr>
<td>10.3920</td>
<td>0.9645</td>
<td></td>
</tr>
<tr>
<td>11.5930</td>
<td>0.9965</td>
<td></td>
</tr>
<tr>
<td>12.7940</td>
<td>0.9992</td>
<td></td>
</tr>
<tr>
<td>13.9950</td>
<td>0.9998</td>
<td></td>
</tr>
<tr>
<td>15.1960</td>
<td>0.9999</td>
<td></td>
</tr>
<tr>
<td>16.3970</td>
<td>1.0000</td>
<td></td>
</tr>
<tr>
<td>17.5980</td>
<td>1.0000</td>
<td></td>
</tr>
<tr>
<td>18.7990</td>
<td>1.0000</td>
<td></td>
</tr>
<tr>
<td>20.0000</td>
<td>1.0000</td>
<td></td>
</tr>
</tbody>
</table>