

# NAG Library Function Document

## 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, const double x[], const double f[],
    const double d[], Integer m, const 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  $i = 0, 1, \dots, m - 1$ . 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 References

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

### 5 Arguments

- |    |  |               |
|----|--|---------------|
| 1: | <b>n</b> – Integer   | <i>Input</i>  |
|    | <i>On entry:</i> <b>n</b> must be unchanged from the previous call of nag_monotonic_interpolant (e01bec).  |               |
| 2: | <b>x[n]</b> – const double   | <i>Input</i>  |
| 3: | <b>f[n]</b> – const double   | <i>Input</i>  |
| 4: | <b>d[n]</b> – const double   | <i>Input</i>  |
|    | <i>On entry:</i> <b>x</b> , <b>f</b> and <b>d</b> must be unchanged from the previous call of nag_monotonic_interpolant (e01bec).                          |               |
| 5: | <b>m</b> – Integer   | <i>Input</i>  |
|    | <i>On entry:</i> <b>m</b> , the number of points at which the interpolant is to be evaluated.  |               |
|    | <i>Constraint:</i> <b>m</b> $\geq$ 1.  |               |
| 6: | <b>px[m]</b> – const double  | <i>Input</i>  |
|    | <i>On entry:</i> the <b>m</b> values of <i>x</i> at which the interpolant is to be evaluated.  |               |
| 7: | <b>pf[m]</b> – double  | <i>Output</i> |
|    | <i>On exit:</i> <b>pf</b> [ <i>i</i> ] contains the value of the interpolant evaluated at the point <b>px</b> [ <i>i</i> ], for $i = 0, 1, \dots, m - 1$ . |               |

- 8: **fail** – NagError \* Input/Output  
 The NAG error argument (see Section 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_INT\_ARG\_LT

On entry, **m** =  $\langle value \rangle$ .  
 Constraint: **m**  $\geq 1$ .  
 On entry, **n** =  $\langle value \rangle$ .  
 Constraint: **n**  $\geq 2$ .

### 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  $r = 0, 1, \dots, n-1$ , are not in strictly increasing order.

### NW\_EXTRAPOLATE

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

## 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 nag\_monotonic\_evaluate (e01bfc) 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  $[\mathbf{x}(r-1), \mathbf{x}(r)]$ ). A single call of nag\_monotonic\_evaluate (e01bfc) with  $m > 1$  is more efficient than several calls with  $m = 1$ .

## 10 Example

This example program reads in values of **n**, **x**, **f**, **d** and **m**, and then calls nag\_monotonic\_evaluate (e01bfc) to evaluate the interpolant at equally spaced points.

### 10.1 Program Text

```
/* nag_monotonic_evaluate (e01bfc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group
 *
 * Mark 2 revised, 1992.
 * Mark 5 revised, 1998.
 * Mark 8 revised, 2004.
 */

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

int main(void)
```

```

{
  Integer  exit_status = 0, i, m, n, r;
  NagError fail;
  double   *d = 0, *f = 0, *pf = 0, *px = 0, step, *x = 0;

  INIT_FAIL(fail);

  printf("nag_monotonic_evaluate (e01bfc) Example Program Results\n");
#ifdef _WIN32
  scanf_s("%*[\n]"); /* Skip to end of line */
#else
  scanf("%*[\n]"); /* Skip to end of line */
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"", &n);
#else
  scanf("%"NAG_IFMT"", &n);
#endif
  if (n >= 2)
  {
    if (!(d = NAG_ALLOC(n, double)) ||
        !(f = NAG_ALLOC(n, double)) ||
        !(x = NAG_ALLOC(n, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  }
  else
  {
    printf("Invalid n.\n");
    exit_status = 1;
    return exit_status;
  }
  for (r = 0; r < n; r++)
#ifdef _WIN32
    scanf_s("%lf%lf%lf", &x[r], &f[r], &d[r]);
#else
    scanf("%lf%lf%lf", &x[r], &f[r], &d[r]);
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"", &m);
#else
  scanf("%"NAG_IFMT"", &m);
#endif
  if (m >= 1)
  {
    if (!(pf = NAG_ALLOC(m, double)) ||
        !(px = NAG_ALLOC(m, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  }
  else
  {
    printf("Invalid m.\n");
    exit_status = 1;
    return exit_status;
  }
  /* 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]);
  /* nag_monotonic_evaluate (e01bfc).
   * Evaluation of interpolant computed by
   * nag_monotonic_interpolant (e01bec), function only
   */
  nag_monotonic_evaluate(n, x, f, d, m, px, pf, &fail);
}

```

```

if (fail.code != NE_NOERROR)
{
    printf("Error from nag_monotonic_evaluate (e01bfc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
}
printf("          Interpolated\n");
printf("    Abscissa      Value\n");
for (i = 0; i < m; i++)
    printf("%13.4f%13.4f\n", px[i], pf[i]);
END:
NAG_FREE(d);
NAG_FREE(f);
NAG_FREE(pf);
NAG_FREE(px);
NAG_FREE(x);
return exit_status;
}

```

## 10.2 Program Data

nag\_monotonic\_evaluate (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

```

## 10.3 Program Results

nag\_monotonic\_evaluate (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

---