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, const double x[], const double f[],
const double d[], Integer m, const double px[], double pf[],
double pd[], NagError *fail)

3 Description
nag_monotonic_deriv (e01bgc) 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 function is derived from routine PCHFD in Fritsch (1982).

4 References
Laboratory

5 Arguments
1: n – Integer
   On entry: n must be unchanged from the previous call of nag_monotonic_interpolant (e01bec).

2: x[n] – const double
   Input

3: f[n] – const double
   Input

4: d[n] – const double
   Input

   On entry: x, f and d must be unchanged from the previous call of nag_monotonic_interpolant
   (e01bec).

5: m – Integer
   Input

   On entry: m, the number of points at which the interpolant is to be evaluated.

   Constraint: $m \geq 1$.

6: px[m] – const double
   Input

   On entry: the $m$ values of $x$ at which the interpolant is to be evaluated.
7: \( \text{pf}[m] \) – double

\textit{Output}

On exit: \( \text{pf}[i] \) contains the value of the interpolant evaluated at the point \( \text{px}[i] \), for \( i = 0, 1, \ldots, m - 1 \).

8: \( \text{pd}[m] \) – double

\textit{Output}

On exit: \( \text{pd}[i] \) contains the first derivative of the interpolant evaluated at the point \( \text{px}[i] \), for \( i = 0, 1, \ldots, m - 1 \).

9: \( \text{fail} \) – NagError

\textit{Input/Output}

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE_INT_ARG_LT}

On entry, \( m = \langle \text{value} \rangle \).

Constraint: \( m \geq 1 \).

On entry, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 2 \).

\textbf{NE_NOT_MONOTONIC}

On entry, \( x[r - 1] \geq x[r] \) for \( r = \langle \text{value} \rangle : x[r - 1] = \langle \text{value} \rangle, x[r] = \langle \text{value} \rangle \).

The values of \( x[r] \), for \( r = 0, 1, \ldots, n - 1 \), are not in strictly increasing order.

\textbf{NW_EXTRAPOLATE}

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

7 Accuracy

The computational errors in the arrays \( \text{pf} \) and \( \text{pd} \) should be negligible in most practical situations.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by \texttt{nag_monotonic_deriv (e01bgc)} is approximately proportional to the number of evaluation points, \( m \). The evaluation will be most efficient if the elements of \( \text{px} \) are in nondecreasing order (or, more generally, if they are grouped in increasing order of the intervals \( [x[r - 1], x[r]] \)). A single call of \texttt{nag_monotonic_deriv (e01bgc)} with \( m > 1 \) is more efficient than several calls with \( m = 1 \).

10 Example

This example program reads in values of \( n, x, f \) and \( d \) and calls \texttt{nag_monotonic_deriv (e01bgc)} to compute the values of the interpolant and its derivative at equally spaced points.
#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, *pd = 0, *pf = 0, *px = 0, step, *x = 0;
    INIT_FAIL(fail);

    printf("nag_monotonic_deriv (e01bgc) Example Program Results\n");
    #ifdef _WIN32
        scanf_s("%*[\n]"); /* Skip heading in data file */
    #else
        scanf("%*[\n]"); /* Skip heading in data file */
    #endif

    #ifdef _WIN32
        scanf_s("%"NAG_IFMT"", &n);
    #else
        scanf("%"NAG_IFMT"", &n);
    #endif
    if (n >= 2)
    {
        if (!(x = NAG_ALLOC(n, double)) ||
            !(f = NAG_ALLOC(n, double)) ||
            !(d = NAG_ALLOC(n, double)))
        {
            printf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }
    }
    else
    {
        printf("Invalid n.\n");
        exit_status = 1;
        return exit_status;
    }

    #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 (!(pd = NAG_ALLOC(m, double)) ||
            !(pf = NAG_ALLOC(m, double)) ||
            !(px = NAG_ALLOC(m, double)))
        {
            printf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }
    }
    ...........

Mark 25
/* 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_deriv (e01bgc). */
Evaluation of interpolant computed by
/nag_monotonic_interpolant (e01bec), function and first
derivative */
nag_monotonic_deriv(n, x, f, d, m, px, pf, pd, &fail);
if (fail.code != NE_NOERROR)
{
  printf("Error from nag_monotonic_deriv (e01bgc).\n%", fail.message);
  exit_status = 1;
  goto END;
}
else
{
  printf("Invalid m.\n");
  exit_status = 1;
  return exit_status;
}

END:
NAG_FREE(x);
NAG_FREE(pd);
NAG_FREE(pf);
NAG_FREE(px);
NAG_FREE(f);
NAG_FREE(d);
return exit_status;

10.2 Program Data
nag_monotonic_deriv (e01bgc) Example Program Data

<table>
<thead>
<tr>
<th>Abscissa</th>
<th>Interpolated Value</th>
<th>Interpolated Derivative</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.990</td>
<td>0.0000E+0</td>
<td>0.0000E+0</td>
</tr>
<tr>
<td>8.090</td>
<td>0.27643E-4</td>
<td>5.52510E-4</td>
</tr>
<tr>
<td>8.190</td>
<td>0.43749E-1</td>
<td>0.33587E+0</td>
</tr>
<tr>
<td>8.700</td>
<td>0.16918E+0</td>
<td>0.34944E+0</td>
</tr>
<tr>
<td>9.200</td>
<td>0.46943E+0</td>
<td>0.59696E+0</td>
</tr>
<tr>
<td>10.00</td>
<td>0.94374E+0</td>
<td>6.03260E-2</td>
</tr>
<tr>
<td>11.500</td>
<td>0.99864E+0</td>
<td>8.98335E-4</td>
</tr>
<tr>
<td>15.00</td>
<td>0.99992E0</td>
<td>2.93954E-5</td>
</tr>
<tr>
<td>20.00</td>
<td>0.99999E+0</td>
<td>0.00000E+0</td>
</tr>
</tbody>
</table>

10.3 Program Results
nag_monotonic_deriv (e01bgc) Example Program Results

<table>
<thead>
<tr>
<th>Abscissa</th>
<th>Interpolated Abscissa Value</th>
<th>Interpolated Interpolated Derivative</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.990</td>
<td>0.0000</td>
<td>0.0000e+0</td>
</tr>
<tr>
<td>9.1910</td>
<td>0.4640</td>
<td>6.060e-01</td>
</tr>
<tr>
<td>10.3920</td>
<td>0.9645</td>
<td>4.569e-02</td>
</tr>
<tr>
<td>11.5930</td>
<td>0.9965</td>
<td>9.917e-03</td>
</tr>
<tr>
<td>12.7940</td>
<td>0.9992</td>
<td>6.249e-04</td>
</tr>
<tr>
<td>13.9950</td>
<td>0.9998</td>
<td>2.708e-04</td>
</tr>
<tr>
<td>Value</td>
<td>Weight</td>
<td>Error</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>15.1960</td>
<td>0.9999</td>
<td>2.809e-05</td>
</tr>
<tr>
<td>16.3970</td>
<td>1.0000</td>
<td>2.034e-05</td>
</tr>
<tr>
<td>17.5980</td>
<td>1.0000</td>
<td>1.308e-05</td>
</tr>
<tr>
<td>18.7990</td>
<td>1.0000</td>
<td>6.297e-06</td>
</tr>
<tr>
<td>20.0000</td>
<td>1.0000</td>
<td>-9.529e-22</td>
</tr>
</tbody>
</table>