NAG Library Function Document

nag_1d_quad_vals (d01gac)

1 Purpose

nag_1d_quad_vals (d01gac) integrates a function which is specified numerically at four or more points, over the whole of its specified range, using third-order finite difference formulae with error estimates, according to a method due to Gill and Miller (1972).

2 Specification

```
#include <nag.h>
#include <nagd01.h>
void nag_1d_quad_vals (Integer n, const double x[], const double y[],
     double *ans, double *er, NagError *fail)
```

3 Description

nag_1d_quad_vals (d01gac) evaluates the definite integral

$$I = \int_{x_1}^{x_n} y(x)\,dx,$$

where the function y is specified at the n-points $x_1, x_2, \ldots, x_n$, which should be all distinct, and in either ascending or descending order. The integral between successive points is calculated by a four-point finite difference formula centred on the interval concerned, except in the case of the first and last intervals, where four-point forward and backward difference formulae respectively are employed. If n is less than 4, the function fails. An approximation to the truncation error is integrated and added to the result. It is also returned separately to give an estimate of the uncertainty in the result. The method is due to Gill and Miller (1972).

4 References

Gill P E and Miller G F (1972) An algorithm for the integration of unequally spaced data *Comput. J.* **15** 80–83

5 Arguments

1: **n** – Integer *Input*

On entry: n, the number of points.

Constraint: $\mathbf{n} \geq 4$.

2: **x**[**n**] – const double *Input*

On entry: the values of the independent variable, i.e., the $x_1, x_2, \ldots, x_n$.

Constraint: either $\mathbf{x}[0] < \mathbf{x}[1] < \cdots < \mathbf{x}[\mathbf{n}-1]$ or $\mathbf{x}[0] > \mathbf{x}[1] > \cdots > \mathbf{x}[\mathbf{n}-1]$.

3: **y**[**n**] – const double *Input*

On entry: the values of the dependent variable y_i at the points x_i, for $i = 1, 2, \ldots, n$.

4: **ans** – double * *Output*

On exit: the estimated value of the integral.

5: **er** – double * *Output*

On exit: an estimate of the uncertainty in **ans**.

6: **fail** – NagError * *Input/Output*

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

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 4$.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

NE_NOT_STRICTLY_DECREASING

The sequence **x** is not strictly decreasing: $\mathbf{x}[\langle value \rangle] = \langle value \rangle$ and $\mathbf{x}[\langle value \rangle] = \langle value \rangle$.

NE_NOT_STRICTLY_INCREASING

The sequence **x** is not strictly increasing: $\mathbf{x}[\langle value \rangle] = \langle value \rangle$, $\mathbf{x}[\langle value \rangle] = \langle value \rangle$.

NE_QUAD_FIRST_TWO_PTS_EQL

The sequence **x** has first two points equal: $\mathbf{x}[0] = \langle value \rangle$ and $\mathbf{x}[1] = \langle value \rangle$.

7 Accuracy

No accuracy level is specified by you before calling nag_1d_quad_vals (d01gac) but on return the absolute value of **er** is an approximation to, but not necessarily a bound for, $|I - \mathbf{ans}|$. If on exit **fail.code** = NE_INT, NE_NOT_STRICTLY_DECREASING, NE_NOT_STRICTLY_INCREASING or NE_QUAD_FIRST_TWO_PTS_EQL, both **ans** and **er** are returned as zero.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_1d_quad_vals (d01gac) depends on the number of points supplied, n.

In their paper, Gill and Miller (1972) do not add the quantity **er** to **ans** before return. However, extensive tests have shown that a dramatic reduction in the error often results from such addition. In other cases, it does not make an improvement, but these tend to be cases of low accuracy in which the modified answer is not significantly inferior to the unmodified one. You have the option of recovering the Gill–Miller answer by subtracting **er** from **ans** on return from the function.

10 Example

This example evaluates the integral

$$\int_0^1 \frac{4}{1+x^2}\,dx = \pi$$

reading in the function values at 21 unequally spaced points.

10.1 Program Text

```
/* nag_1d_quad_vals (d01gac) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 * Mark 8 revised, 2004.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagd01.h>

int main(void)
{

  Integer  exit_status = 0, i, n;
  NagError fail;
  double   ans, error, *x = 0, *y = 0;

  INIT_FAIL(fail);

  printf("nag_1d_quad_vals (d01gac) 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 >= 4)
    {
      if (!(x = NAG_ALLOC(n, double)) ||
          !(y = 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 (i = 0; i < n; ++i)
#ifdef _WIN32
```

```
    scanf_s("%lf%lf", &x[i], &y[i]);
#else
    scanf("%lf%lf", &x[i], &y[i]);
#endif
  /* nag_1d_quad_vals (d01gac).
   * One-dimensional integration of a function defined by data
   * values only
   */
  nag_1d_quad_vals(n, x, y, &ans, &error, &fail);
  if (fail.code == NE_NOERROR)
    {
      printf("Integral = %7.4f\n", ans);
      printf("Estimated error = %7.4f\n", error);
    }
  else
    {
      printf("Error from nag_1d_quad_vals (d01gac).\n%s\n",
              fail.message);
      printf("%s\n", fail.message);
      exit_status = 1;
    }
 END:
  NAG_FREE(x);
  NAG_FREE(y);
  return exit_status;
}
```

10.2 Program Data

```
nag_1d_quad_vals (d01gac) Example Program Data
   21
   0.00   4.0000
   0.04   3.9936
   0.08   3.9746
   0.12   3.9432
   0.22   3.8153
   0.26   3.7467
   0.30   3.6697
   0.38   3.4943
   0.39   3.4719
   0.42   3.4002
   0.45   3.3264
   0.46   3.3014
   0.60   2.9412
   0.68   2.7352
   0.72   2.6344
   0.73   2.6094
   0.83   2.3684
   0.85   2.3222
   0.88   2.2543
   0.90   2.2099
   1.00   2.0000
```

10.3 Program Results

```
nag_1d_quad_vals (d01gac) Example Program Results
Integral =  3.1414
Estimated error = -0.0001
```