NAG Library Function Document

nag_quad_1d_fin_smooth (d01bdc)

1 Purpose

nag_quad_1d_fin_smooth (d01bdc) calculates an approximation to the integral of a function over a finite interval $[a, b]$:

$$I = \int_a^b f(x)\, dx.$$

It is non-adaptive and as such is recommended for the integration of 'smooth' functions. These **exclude** integrands with singularities, derivative singularities or high peaks on $[a, b]$, or which oscillate too strongly on $[a, b]$.

2 Specification

```
#include <nag.h>
#include <nagd01.h>
void nag_quad_1d_fin_smooth (
     double  (*f)(double x, Nag_Comm *comm),
     double a, double b, double epsabs, double epsrel, double *result,
     double *abserr, Nag_Comm *comm)
```

3 Description

nag_quad_1d_fin_smooth (d01bdc) is based on the QUADPACK routine QNG (see Piessens *et al.* (1983)). It is a non-adaptive function which uses as its basic rules, the Gauss 10-point and 21-point formulae. If the accuracy criterion is not met, formulae using 43 and 87 points are used successively, stopping whenever the accuracy criterion is satisfied.

This function is designed for smooth integrands only.

4 References

Patterson T N L (1968) The Optimum addition of points to quadrature formulae *Math. Comput.* **22** 847–856

Piessens R, de Doncker–Kapenga E, Überhuber C and Kahaner D (1983) *QUADPACK, A Subroutine Package for Automatic Integration* Springer–Verlag

5 Arguments

1: **f** – function, supplied by the user *External Function*

f must return the value of the integrand f at a given point.

The specification of **f** is:

```
double  f (double x, Nag_Comm *comm)
```

1: **x** – double *Input*

On entry: the point at which the integrand f must be evaluated.

2: **comm** – Nag_Comm * *Communication Structure*

Pointer to structure of type Nag_Comm; the following members are relevant to **f**.

user – double *
iuser – Integer *
p – Pointer

The type Pointer will be `void *`. Before calling nag_quad_1d_fin_smooth (d01bdc) you may allocate memory and initialize these pointers with various quantities for use by **f** when called from nag_quad_1d_fin_smooth (d01bdc) (see Section 3.2.1.1 in the Essential Introduction).

2: **a** – double *Input*

On entry: a, the lower limit of integration.

3: **b** – double *Input*

On entry: b, the upper limit of integration. It is not necessary that $a < b$.

4: **epsabs** – double *Input*

On entry: the absolute accuracy required. If **epsabs** is negative, the absolute value is used. See Section 7.

5: **epsrel** – double *Input*

On entry: the relative accuracy required. If **epsrel** is negative, the absolute value is used. See Section 7.

6: **result** – double * *Output*

On exit: the approximation to the integral I.

7: **abserr** – double * *Output*

On exit: an estimate of the modulus of the absolute error, which should be an upper bound for $|I - \textbf{result}|$.

8: **comm** – Nag_Comm * *Communication Structure*

The NAG communication argument (see Section 3.2.1.1 in the Essential Introduction).

6 Error Indicators and Warnings

There are no specific errors detected by nag_quad_1d_fin_smooth (d01bdc). However, if **abserr** is greater than

$$\max\{\textbf{epsabs}, \textbf{epsrel} \times |\textbf{result}|\}$$

this indicates that the function has probably failed to achieve the requested accuracy within 87 function evaluations.

7 Accuracy

nag_quad_1d_fin_smooth (d01bdc) attempts to compute an approximation, **result**, such that:

$$|I - \textbf{result}| \leq tol,$$

where

$$tol = \max\{|\textbf{epsabs}|, |\textbf{epsrel}| \times |I|\},$$

and **epsabs** and **epsrel** are user-specified absolute and relative error tolerances. There can be no guarantee that this is achieved, and you are advised to subdivide the interval if you have any doubts about the accuracy obtained. Note that **abserr** contains an estimated bound on $|I - \textbf{result}|$.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_quad_1d_fin_smooth (d01bdc) depends on the integrand and the accuracy required.

10 Example

This example computes

$$\int_0^1 x^2 \sin(10\pi x)\, dx.$$

10.1 Program Text

```
/* nag_quad_1d_fin_smooth (d01bdc) Example Program.
 *
 * Copyright 2011, Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */

#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagd01.h>
#include <nagx01.h>

#ifdef __cplusplus
extern "C" {
#endif
  static double NAG_CALL f(double x, Nag_Comm *comm);
#ifdef __cplusplus
}
#endif

int main(void)
{
  static double ruser[1] = {-1.0};
  Integer exit_status = 0;
  double  a, abserr, b, epsabs, epsrel, result;
  Nag_Comm comm;

  printf("nag_quad_1d_fin_smooth (d01bdc) Example Program Results\n");

  /* For communication with user-supplied functions: */
  comm.user = ruser;

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  /* Input arguments */
  scanf("%lf %lf", &epsabs, &epsrel);
  scanf("%lf %lf", &a, &b);

  /* nag_quad_1d_fin_smooth (d01bdc).
   * One-dimensional quadrature, non-adaptive, finite interval.
   */
  nag_quad_1d_fin_smooth(f, a, b, epsabs, epsrel, &result, &abserr,
```

```
                         &comm);

  printf("\na      - lower limit of integration = %10.4f"
         "\nb      - upper limit of integration = %10.4f"
         "\nepsabs - absolute accuracy requested = %9.2e"
         "\nepsrel - relative accuracy requested = %9.2e\n"
         "\nresult - approximation to the integral = %9.5f"
         "\nabserr - estimate to the absolute error = %9.2e\n\n",
         a, b, epsabs, epsrel, result, abserr);
  if (abserr > MAX(epsabs, epsrel * fabs(result)))
    printf("Warning - requested accuracy may not have been achieved.\n");

  return exit_status;
}

static double NAG_CALL f(double x, Nag_Comm *comm)
{
  if (comm->user[0] == -1.0)
    {
      printf("(User-supplied callback f, first invocation.)\n");
      comm->user[0] = 0.0;
    }
  return (pow(x, 2))*sin(10.0*nag_pi*x);
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_quad_1d_fin_smooth (d01bdc) Example Program Results
(User-supplied callback f, first invocation.)

a      - lower limit of integration =     0.0000
b      - upper limit of integration =     1.0000
epsabs - absolute accuracy requested =  0.00e+00
epsrel - relative accuracy requested =  1.00e-04

result - approximation to the integral =  -0.03183
abserr - estimate to the absolute error =  1.34e-11
```