NAG Library Function Document

nag_quad_1d_fin_gonnet_vec (d01rgc)

1 Purpose

nag_quad_1d_fin_gonnet_vec (d01rgc) is a general purpose integrator which calculates an approximation to the integral of a function $f(x)$ over a finite interval $[a, b]$:

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

The function is suitable as a general purpose integrator, and can be used when the integrand has singularities and infinities. In particular, the function can continue if the function **f** explicitly returns a quiet or signalling NaN or a signed infinity.

2 Specification

```
#include <nag.h>
#include <nagd01.h>
void nag_quad_1d_fin_gonnet_vec (double a, double b,
     void (*f)(const double x[], Integer nx, double fv[], Integer *iflag,
          Nag_Comm *comm),
     double epsabs, double epsrel, double *dinest, double *errest,
     Integer *nevals, Nag_Comm *comm, NagError *fail)
```

3 Description

nag_quad_1d_fin_gonnet_vec (d01rgc) uses the algorithm described in Gonnet (2010). It is an adaptive algorithm, similar to the QUADPACK routine QAGS (see Piessens *et al.* (1983), see also nag_quad_1d_gen_vec_multi_rcomm (d01rac)) but includes significant differences regarding how the integrand is represented, how the integration error is estimated and how singularities and divergent integrals are treated. The local error estimation is described in Gonnet (2010).

nag_quad_1d_fin_gonnet_vec (d01rgc) requires a function to evaluate the integrand at an array of different points and is therefore amenable to parallel execution.

4 References

Gonnet P (2010) Increasing the reliability of adaptive quadrature using explicit interpolants *ACM Trans. Math. software* **37** 26

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

5 Arguments

1: **a** – double *Input*

On entry: a, the lower limit of integration.

2: **b** – double *Input*

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

Note: if **a** = **b**, the function will immediately return **dinest** = 0.0, **errest** = 0.0 and **nevals** = 0.

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

f must return the value of the integrand f at a set of points.

The specification of **f** is:

```
void f (const double x[], Integer nx, double fv[], Integer *iflag,
     Nag_Comm *comm)
```

1: **x**[**nx**] – const double *Input*

On entry: the abscissae, x_i, for $i = 1, 2, \ldots,$ **nx**, at which function values are required.

2: **nx** – Integer *Input*

On entry: the number of abscissae at which a function value is required.

3: **fv**[**nx**] – double *Output*

On exit: **fv** must contain the values of the integrand f. **fv**$[i-1] = f(x_i)$ for all $i = 1, 2, \ldots,$ **nx**.

4: **iflag** – Integer * *Input/Output*

On entry: **iflag** $= 0$.

On exit: set **iflag** < 0 to force an immediate exit with **fail.code** = NE_USER_STOP.

5: **comm** – Nag_Comm *

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_gonnet_vec (d01rgc) you may allocate memory and initialize these pointers with various quantities for use by **f** when called from nag_quad_1d_fin_gonnet_vec (d01rgc) (see Section 2.3.1.1 in How to Use the NAG Library and its Documentation).

4: **epsabs** – double *Input*

On entry: the absolute accuracy required.

If **epsabs** is negative, |**epsabs**| is used. See Section 7.

If **epsabs** $= 0.0$, only the relative error will be used.

5: **epsrel** – double *Input*

On entry: the relative accuracy required.

If **epsrel** is negative, |**epsrel**| is used. See Section 7.

If **epsrel** $= 0.0$, only the absolute error will be used otherwise the actual value of **epsrel** used by nag_quad_1d_fin_gonnet_vec (d01rgc) is max(***machine precision***, |**epsrel**|).

Constraint: at least one of **epsabs** and **epsrel** must be nonzero.

6: **dinest** – double * *Output*

On exit: the estimate of the definite integral **f**.

7: **errest** – double * *Output*

On exit: the error estimate of the definite integral **f**.

8: **nevals** – Integer * *Output*

On exit: the total number of points at which the integrand, f, has been evaluated.

9: **comm** – Nag_Comm *

The NAG communication argument (see Section 2.3.1.1 in How to Use the NAG Library and its Documentation).

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

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ACCURACY

The requested accuracy was not achieved. Consider using larger values of **epsabs** and **epsrel**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

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

NE_CONVERGENCE

The integral is probably divergent or slowly convergent.

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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_TOO_SMALL

Both **epsabs** $= 0.0$ and **epsrel** $= 0.0$.

NE_USER_STOP

Exit requested from **f** with **iflag** $=$ ⟨*value*⟩.

7 Accuracy

nag_quad_1d_fin_gonnet_vec (d01rgc) cannot guarantee, but in practice usually achieves, the following accuracy:

$$|I - \mathbf{dinest}| \leq tol,$$

where

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

and **epsabs** and **epsrel** are user-specified absolute and relative error tolerances. Moreover, it returns the quantity **errest** which, in normal circumstances, satisfies

$$|I - \mathbf{dinest}| \leq \mathbf{errest} \leq tol.$$

8 Parallelism and Performance

nag_quad_1d_fin_gonnet_vec (d01rgc) is currently neither directly nor indirectly threaded. In particular, the user-supplied function **f** is not called from within a parallel region initialized inside nag_quad_1d_fin_gonnet_vec (d01rgc).

The user-supplied function **f** uses a vectorized interface, allowing for the required vector of function values to be evaluated in parallel; for example by placing appropriate OpenMP directives in the code for the user-supplied function **f**.

9 Further Comments

The time taken by nag_quad_1d_fin_gonnet_vec (d01rgc) depends on the integrand and the accuracy required.

nag_quad_1d_fin_gonnet_vec (d01rgc) is suitable for evaluating integrals that have singularities within the requested interval.

In particular, nag_quad_1d_fin_gonnet_vec (d01rgc) accepts non-finite values on return from the user-supplied function **f**, and will adapt the integration rule accordingly to eliminate such points. Non-finite values include NaNs and infinities.

10 Example

This example computes

$$\int_{-1}^{1} \frac{\sin(x)}{x} \ln(10(1-x)).$$

10.1 Program Text

```
/* nag_quad_1d_fin_gonnet_vec (d01rgc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagd01.h>
#include <nagx07.h>

#ifdef __cplusplus
extern "C"
{
#endif
  static void NAG_CALL f(const double x[], Integer nx, double fv[],
                         Integer *iflag, Nag_Comm *comm);
#ifdef __cplusplus
}
#endif

int main(void)
```

```
{
  static double ruser[1] = { -1.0 };
  Integer exit_status = 0;
  double a, b, dinest, epsabs, epsrel, errest;
  Integer nevals;
  Integer exmode[3], exmode_old[3];

  /* Nag types */
  NagError fail;
  Nag_Comm comm;

  INIT_FAIL(fail);

  printf("nag_quad_1d_fin_gonnet_vec (d01rgc) Example Program Results\n\n");

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

  /* The example function can raise various exceptions - it contains
   * a division by zero and a log singularity - although its integral
   * is well behaved.
   */

  /* nag_get_ieee_exception_mode (x07cac).
   * Gets current behaviour of floating point exceptions.
   */
  nag_get_ieee_exception_mode(exmode_old);

  /* Turn exception halting mode off for the three common exceptions:
   * overflow, division-by-zero, and invalid operation.
   */
  exmode[0] = exmode[1] = exmode[2] = 0;
  /* nag_set_ieee_exception_mode (x07cbc). Sets behaviour of
     floating-point exceptions */
  nag_set_ieee_exception_mode(exmode);
  epsabs = 0.0;
  epsrel = 1.0e-04;
  a = -1.0;
  b = 1.0;

  /* Evaluate the integral using the vectorized One-dimensional adaptive
   * quadrature routine nag_quad_1d_fin_gonnet_vec (d01rgc).
   */
  nag_quad_1d_fin_gonnet_vec(a, b, f, epsabs, epsrel, &dinest, &errest,
                             &nevals, &comm, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_quad_1d_fin_gonnet_vec (d01rgc).\n%s\n",
           fail.message);
    exit_status = 1;
    if (fail.code != NE_ACCURACY)
      goto END;
  }

  printf("a      -     lower limit of integration = %11.4f\n", a);
  printf("b      -     upper limit of integration = %11.4f\n", b);
  printf("epsabs -    absolute accuracy requested = %11.2e\n", epsabs);
  printf("epsrel -    relative accuracy requested = %11.2e\n", epsrel);
  printf("\n");
  printf("dinest -  approximation to the integral = %11.4f\n", dinest);
  printf("errest - estimate of the absolute error = %11.2e\n", errest);
  printf("nevals - number of function evaluations = %11" NAG_IFMT "\n",
         nevals);

END:
  /* Restore the original halting mode */
  nag_set_ieee_exception_mode(exmode_old);

  return exit_status;
}

static void NAG_CALL f(const double x[], Integer nx, double fv[],
```

```
                        Integer *iflag, Nag_Comm *comm)
{
  Integer i;
  if (comm->user[0] == -1.0) {
    printf("(User-supplied callback f, first invocation.)\n");
    comm->user[0] = 0.0;
  }
  for (i = 0; i < nx; i++)
    fv[i] = sin(x[i]) / x[i] * log(10.0 * (1.0 - x[i]));
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_quad_1d_fin_gonnet_vec (d01rgc) Example Program Results

(User-supplied callback f, first invocation.)
a      -     lower limit of integration =     -1.0000
b      -     upper limit of integration =      1.0000
epsabs -    absolute accuracy requested =    0.00e+00
epsrel -    relative accuracy requested =    1.00e-04

dinest -  approximation to the integral =      3.8116
errest - estimate of the absolute error =    3.39e-04
nevals - number of function evaluations =         593
```