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 formulæ. If the accuracy criterion is not met, formulæ 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

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: \textbf{comm} – Nag_Comm *  
Pointer to structure of type Nag_Comm; the following members are relevant to f.

\textbf{user} – double * 
\textbf{iuser} – Integer * 
\textbf{p} – Pointer  
The type Pointer will be \texttt{void *}. Before calling \texttt{nag_quad_1d_fin_smooth (d01bdc)} you may allocate memory and initialize these pointers with various quantities for use by f when called from \texttt{nag_quad_1d_fin_smooth (d01bdc)} (see Section 3.2.1.1 in the Essential Introduction).

2: \textit{a} – double \textit{Input}  
On entry: \textit{a}, the lower limit of integration.

3: \textit{b} – double \textit{Input}  
On entry: \textit{b}, the upper limit of integration. It is not necessary that \textit{a} < \textit{b}.

4: \textit{epsabs} – double \textit{Input}  
On entry: the absolute accuracy required. If \textit{epsabs} is negative, the absolute value is used. See Section 7.

5: \textit{epsrel} – double \textit{Input}  
On entry: the relative accuracy required. If \textit{epsrel} is negative, the absolute value is used. See Section 7.

6: \textit{result} – double * \textit{Output}  
On exit: the approximation to the integral \textit{I}.

7: \textit{abserr} – double * \textit{Output}  
On exit: an estimate of the modulus of the absolute error, which should be an upper bound for \(|I - \textit{result}|\).

8: \textbf{comm} – Nag_Comm *  
The NAG communication argument (see Section 3.2.1.1 in the Essential Introduction).

6 \textbf{Error Indicators and Warnings}  
There are no specific errors detected by \texttt{nag_quad_1d_fin_smooth (d01bdc)}. However, if \textit{abserr} is greater than \[
\max\{\textit{epsabs}, \textit{epsrel} \times |\textit{result}|\}
\]
this indicates that the function has probably failed to achieve the requested accuracy within 87 function evaluations.

7 \textbf{Accuracy}  
\texttt{nag_quad_1d_fin_smooth (d01bdc)} attempts to compute an approximation, \textit{result}, such that: \[
|I - \textit{result}| \leq \textit{tol},
\]
where \[
\textit{tol} = \max\{|\textit{epsabs}|, |\textit{epsrel}| \times |I|\}.
\]
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 `abser` contains an estimated bound on $|I - \text{result}|$.

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

```c
#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

typedef void (*fptr)(double x, Nag_Comm *comm);

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 */
#ifdef _WIN32
 scanf_s("%*[\n] ");
#else
 scanf("%*[\n] ");
#endif

 /* Input arguments */
#ifdef _WIN32
 scanf_s("%lf %lf", &epsabs, &epsrel);
#else
 scanf("%lf %lf", &epsabs, &epsrel);
#endif

```
```c
scanf("%lf %lf", &epsabs, &epsrel);
#endif
#ifdef _WIN32
scanf_s("%lf %lf", &a, &b);
#else
scanf("%lf %lf", &a, &b);
#endif

/* 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("a - lower limit of integration = %10.4f\n", a);
printf("b - upper limit of integration = %10.4f\n", b);
printf("epsabs - absolute accuracy requested = %9.2e\n", epsabs);
printf("epsrel - relative accuracy requested = %9.2e\n", epsrel);
printf("result - approximation to the integral = %9.5f\n", result);
printf("abserr - estimate to the absolute error = %9.2e\n", 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