Chapter d01 – Quadrature

1. Scope of the Chapter

This chapter provides routines for the numerical evaluation of definite integrals in one or more dimensions.

2. Background

The routines in this chapter are designed to estimate:

(a) the value of a one-dimensional integral of the form

$$\int_a^b f(x)\ dx \tag{1}$$

where $f(x)$ is defined by the user, either at a set of points $(x_i, f(x_i))$, for $i = 1, 2, ..., n$ where $a = x_1 < x_2 < ... < x_n = b$, or in the form of a function; and the limits of integration a, b may be finite or infinite.
Some methods are specially designed for integrands of the form

$$f(x) = w(x)g(x) \tag{2}$$

which contain a *weight function* $w(x)$ of a specific form. These methods take full account of any particular behaviour attributable to the $w(x)$ factor.

(b) the value of a multi-dimensional definite integral of the form

$$\int_{R_n} f(x_1, x_2, ..., x_n)\ dx_n ... dx_2 dx_1 \tag{3}$$

where $f(x_1, x_2, ..., x_n)$ is a function defined by the user and R_n is some region of n-dimensional space. The simplest form of R_n is the n-rectangle defined by

$$a_i \le x_i \le b_i, \qquad i = 1, 2, ..., n \tag{4}$$

where a_i and b_i are constants. When a_i and b_i are functions of x_j $(j < i)$ the region can easily be transformed to the rectangular form (see Davis and Rabinowitz (1975)).

2.1. One-dimensional Integrals

To estimate the value of a one-dimensional integral, a quadrature rule uses an approximation in the form of a weighted sum of integrand values, i.e.,

$$\int_a^b f(x)\ dx \simeq \sum_{i=1}^{N} w_i f(x_i), \tag{5}$$

where x_i and w_i are the abscissae and the weights of the quadrature rule respectively.
More generally, if the integrand has the form (2), the corresponding formula becomes

$$\int_a^b w(x)g(x)\ dx \simeq \sum_{i=1}^{N} w_i g(x_i). \tag{6}$$

If the integrand is known only at a fixed set of points, these points must be used as the abscissae, and the weighted sum is calculated using finite-difference methods. However, if the functional form of the integrand is known, so that its value can easily be evaluated at any abscissae, then a wide variety of quadrature rules are available, each characterised by its choice of abscissae and the corresponding weights.

The choice of an appropriate rule depends on the interval $[a, b]$ – whether finite, infinite or semi-infinite – and the form of any $w(x)$ factor. A suitable value of N depends on the behaviour of $f(x)$; or of $g(x)$, if there is a $w(x)$ factor present.
Among possible rules, we mention particularly the Gaussian formulae, which employ a distribution of abscissae which is optimal for $f(x)$ or $g(x)$ of polynomial form.
Generally, quadrature algorithms are divided into two catagories; *automatic* and *non-automatic*; the latter is sometimes referred to as the *single rule evalution algorithm*. In a non-automatic algorithm, a fixed number of abscissae, N, is used. This number and the particular rule chosen uniquely determine the weights and abscissae. No estimate is made of the accuracy of the result. However, in an automatic algorithm, the number of abscissae, N, within $[a, b]$ is gradually increased until consistency is achieved to within a level of accuracy (absolute or relative) requested by the user. There are essentially two ways of doing this, *non-adaptive* or *adaptive*; hybrid forms of these two methods are also possible.

(i) Non-adaptive algorithms
A series of rules using increasing values of N are successively applied over the whole interval $[a, b]$. It is clearly more economical if abscissae already used for a lower value of N can be used again as part of a higher-order formula. This principle is known as **optimal extension**. There is no overlap between the abscissae used in Gaussian formulae of different orders. However, the Kronrod formulae are designed to give an optimal $(2N - 1)$-point formula by adding $(N + 1)$ points to an N-point Gauss formula. Further extensions have been developed by Patterson.

(ii) Adaptive algorithms
The interval $[a, b]$ is repeatedly divided into a number of sub-intervals, and integration rules are applied separately to each sub-interval. Typically, the subdivision process will be carried further in the neighbourhood of a sharp peak in the integrand, than where the curve is smooth. Thus, the distribution of abscissae is adapted to the shape of the integrand.
Subdivision raises the problem of what constitutes an acceptable accuracy in each sub-interval. The usual **global acceptability criterion** demands that the sum of the absolute values of the error estimates in the sub-intervals should meet the conditions required of the error over the whole interval. Automatic extrapolation over several levels of subdivision may eliminate the effects of some types of singularities.

An ideal general-purpose method would be an automatic method which could be used for a wide variety of integrands, was efficient (i.e., required the use of as few abscissae as possible), and was reliable (i.e., always gave results within the requested accuracy). Complete reliability is unobtainable, and generally higher reliability is obtained at the expense of efficiency, and vice versa. **It must therefore be emphasised that the automatic routines in this chapter cannot be assumed to be 100% reliable**. In general, however, the reliability is very high.

2.2. Multi-dimensional Integrals

A distinction must be made between cases of moderately low dimensionality (say, up to 4 or 5 dimensions), and those of higher dimensionality. Where the number of dimensions is limited, a one-dimensional method may be applied to each dimension, according to some suitable strategy, and high accuracy may be obtainable (using product rules). However, the number of integrand evaluations rises very rapidly with the number of dimensions, so that the accuracy obtainable with an acceptable amount of computational labour is limited; for example a product of 3-point rules in 20 dimensions would require more than 10^9 integrand evaluations. Special techniques such as the Monte Carlo, number theoretic and Sag-Szekeres methods can be used to deal with high dimensions (see Davis and Rabinowitz (1975)).

(a) Products of one-dimensional rules
Using a two-dimensional integral as an example, we have

$$\int_{a_1}^{b_1} \int_{a_2}^{b_2} f(x, y)\ dy\ dx \simeq \sum_{i=1}^{N} w_i \Big(\int_{a_2}^{b_2} f(x_i, y) dy \Big) \tag{7}$$

$$\int_{a_1}^{b_1} \int_{a_2}^{b_2} f(x, y)\ dy\ dx \simeq \sum_{i=1}^{N} \sum_{j=1}^{N} w_i v_j f(x_i, y_j) \tag{8}$$

where (w_i, x_i) and (v_i, y_i) are the weights and abscissae of the rules used in the respective dimensions.

A different one-dimensional rule may be used for each dimension, as appropriate to the range and any weight function present, and a different strategy may be used, as appropriate to the integrand behaviour as a function of each independent variable.

For a rule-evaluation strategy in all dimensions, the formula (8) is applied in a straightforward manner. For automatic strategies (i.e., attempting to attain a requested accuracy), there is a problem in deciding what accuracy must be requested in the inner integral(s). Reference to formula (7) shows that the presence of a limited but random error in the y-integration for different values of x_i can produce a 'jagged' function of x, which may be difficult to integrate to the desired accuracy and for this reason products of automatic one-dimensional routines should be used with caution (see also Lyness (1983)).

(b) Monte Carlo methods
These are based on estimating the mean value of the integrand sampled at points chosen from an appropriate statistical distribution function. Usually a variance reducing procedure is incorporated to combat the fundamentally slow rate of convergence of the rudimentary form of the technique. These methods can be effective by comparison with alternative methods when the integrand contains singularities or is erratic in some way, but they are of quite limited accuracy.

(c) Automatic adaptive procedures
An automatic adaptive strategy in several dimensions normally involves division of the region into subregions, concentrating the divisions in those parts of the region where the integrand is worst behaved. It is difficult to arrange with any generality for variable limits in the inner integral(s). For this reason, some methods use a region where all the limits are constants; this is called a hyper-rectangle. Integrals over regions defined by variable or infinite limits may be handled by transformation to a hyper-rectangle. Integrals over regions so irregular that such a transformation is not feasible may be handled by surrounding the region by an appropriate hyper-rectangle and defining the integrand to be zero outside the desired region. Such a technique should always be followed by a Monte Carlo method for integration.
The method used locally in each subregion produced by the adaptive subdivision process is usually one of three types: Monte Carlo, number theoretic or deterministic. Deterministic methods are usually the most rapidly convergent but are often expensive to use for high dimensionality and not as robust as the other techniques.

3. References

Davis P J and Rabinowitz P (1975) *Methods of Numerical Integration* Academic Press.
Lyness J N (1983) When not to use an automatic quadrature routine? *SIAM Review* **25** 63–87.
Piessens R, De Doncker-Kapenga E, Überhuber C and Kahaner D K (1983) *QUADPACK, A Subroutine Package for Automatic Integration* Springer-Verlag.
Sobol I M (1974) *The Monte Carlo Method* The University of Chicago Press.
Stroud A H (1971) *Approximate Calculation of Multiple Integrals* Prentice-Hall.

4. Recommendations on Choice and Use of Available Routines

The following three sub-sections consider in turn routines for: one-dimensional integrals over a finite interval, and over a semi-finite or an infinite interval; and multi-dimensional integrals.

4.1. One-dimensional Integral over a Finite Interval

(a) If $f(x)$ is defined numerically at four or more points, then the Gill-Miller finite difference method, nag_1d_quad_vals (d01gac) should be used. The interval of integration is taken to coincide with the range of x-values of the points supplied. It is in the nature of this problem that any routine may be unreliable. In order to check results independently and so as to provide an alternative technique the user may fit the integrand with a cubic spline to the data using nag_1d_spline_fit_knots (e02bac) and then evaluate its integral using nag_1d_spline_intg (e02bdc).

(b) Integrand defined as a function
If the functional form of $f(x)$ is known, then one of the following approaches should be taken. They are arranged in the order from most specific to most general, hence the first applicable procedure in the list will be the most efficient. **However, if the user does not wish to make any assumptions about the integrand, the most reliable routines to use will be nag_1d_quad_gen_1 (d01sjc), although it will in general be less efficient for simple integrals.**

(i) Rule-evaluation routines
If $f(x)$ is known to be sufficiently well-behaved (more precisely, can be closely approximated by a polynomial of moderate degree), nag_1d_quad_gauss_1 (d01tac) (which is based on a family of interlacing Gaussian quadrature) may be used.
nag_1d_quad_gauss_1 (d01tac) may also be used if it is not required to examine the weights and abscissae.

(ii) Automatic adaptive routines
Firstly, several routines are available for integrands of the form $w(x)g(x)$ where $g(x)$ is a 'smooth' function (i.e., has no singularities, sharp peaks or violent oscillations in the interval of integration) and $w(x)$ is a weight function of one of the following forms:
if $w(x) = (b-x)^{\alpha}(x-a)^{\beta}(\log(b-x))^{k}(\log(x-a))^{l}$ where $k, l = 0$ or 1, $\alpha, \beta > -1$, then use nag_1d_quad_wt_alglog_1 (d01spc);
if $w(x) = 1/(x-c)$, then use nag_1d_quad_wt_cauchy_1 (d01sqc) (this integral is called the Hilbert transform of $g(x)$);
if $w(x) = \cos(\omega x)$ or $\sin(\omega x)$, then use nag_1d_quad_wt_trig_1 (d01snc) (this routine can also handle certain types of singularities in $g(x)$).
Secondly, there are some routines for general $f(x)$. If $f(x)$ is known to be free of singularities, though it may be oscillatory, nag_1d_quad_osc_1 (d01skc) may be used.
The most powerful of the finite interval integration routines is nag_1d_quad_gen_1 (d01sjc) (which can cope with singularities of several types). nag_1d_quad_gen_1 (d01sjc) is very reliable, particularly where the integrand has singularities other than at an end-point, or has discontinuities or cusps, and is therefore recommended where the integrand is known to be badly-behaved, or where its nature is completely unknown.
Most of the routines in this chapter require the user to supply a function to evaluate the integrand at a single point.
If $f(x)$ has singularities of certain types, discontinuities or sharp peaks **occurring at known points**, the integral should be evaluated separately over each of the subranges or nag_1d_quad_brkpts_1 (d01slc) may be used.

4.2. One-dimensional Integrals over a Semi-infinite or Infinite Interval

(a) Integrand defined as a set of points
If $f(x)$ is defined numerically at four or more points, and the portion of the integral lying outside the range of the points supplied may be neglected, then the Gill-Miller finite difference method, nag_1d_quad_vals (d01gac), should be used.

(b) Integrand defined as a function

(i) Rule evaluation routines
If $f(x)$ behaves approximately like a polynomial in x, apart from a weight function of the form:

$e^{-\beta x}, \beta > 0$ (semi-infinite interval, lower limit finite);

$e^{-\beta x}, \beta < 0$ (semi-infinite interval, upper limit finite);

$e^{-\beta(x-\alpha)^2}, \beta > 0$ (infinite interval);

or if $f(x)$ behaves approximately like a polynomial in $(x+b)^{-1}$ (semi-infinite range), then nag_1d_quad_gauss_1 (d01tac) may be used.

(ii) Automatic adaptive routines
nag_1d_quad_inf_1 (d01smc) may be used, except for integrands which decay slowly towards an infinite end-point, and oscillate in sign over the entire interval. For this class, it may be possible to calculate the integral by integrating between the zeros and invoking some extrapolation process.

nag_1d_quad_inf_wt_trig_1 (d01ssc) may be used for integrals involving weight functions of the form $\cos(\omega x)$ and $\sin(\omega x)$ over a semi-infinite interval (lower limit finite).

4.3. Multi-dimensional Integrals

At present, there are only two automatic routines available in this area. The algorithms used in these routines are based on a Monte Carlo method and an adaptive sub-division strategy.
Both routines are for integrals of the form

$$\int_{a_1}^{b_1} \int_{a_2}^{b_2} \ldots \int_{a_n}^{b_n} f(x_1, x_2, ..., x_n) dx_n ... dx_2 dx_1.$$

nag_multid_quad_monte_carlo_1 (d01xbc) is an adaptive Monte Carlo routine. This routine is usually slow and not recommended for high accuracy work. It is a robust routine that can often be used for low accuracy results with highly irregular integrands or when n is large.
nag_multid_quad_adapt_1 (d01wcc) is an adaptive deterministic routine. Convergence is fast for well-behaved integrands. Highly accurate results can often be obtained for n between 2 and 5, using significantly fewer integrand evaluations than would be required by nag_multid_quad_monte_carlo_1 (d01xbc). The routine will usually work when the integrand is mildly singular and, for $n \leq 10$, should be used before nag_multid_quad_monte_carlo_1 (d01xbc). If it is known in advance that the integrand is highly irregular, it is best to compare results using both routines.
There are many problems for which one or both of the routines will require large amounts of computing time to obtain even moderately accurate results. The amount of computing time is controlled by the number of integrand evaluations allowed by the user, and users should set this parameter carefully, with reference to the time available and the accuracy desired.

5. Available Functions

At Mark 5, we have introduced thread-safe versions of the functions requiring communication between the calling program and the user-defined function to evaluate the integrand. This has been implemented by introducing a `void *` parameter both in the calling sequence of the NAG function and the user-defined function. Note that the functionality of the thread-safe version is essentially the same as the corresponding thread-unsafe version. However, it is recommended that the thread-safe version be used in preference to the thread-unsafe versions, particularly in view of the fact that the thread-unsafe version may be removed at a later mark of the C Library.

5.1. Thread-unsafe functions

1-D quadrature, adaptive, finite interval, allowing for badly-behaved integrands	`d01ajc`
1-D quadrature, adaptive, finite interval, method suitable for oscilating functions	`d01akc`
1-D quadrature, adaptive, finite interval, allowing for singularities at user-specified break-points	`d01alc`
1-D quadrature, adaptive, infinite or semi-infinite interval	`d01amc`
1-D quadrature, adaptive, finite interval, weight function $\cos(\omega x)$ or $\sin(\omega x)$	`d01anc`
1-D quadrature, adaptive, finite interval, weight function with end-point singularities of algebraico-logarithmic	`d01apc`
1-D quadrature, adaptive, finite interval, weight function $1/(x-c)$, Cauchy principal value (Hilbert transform)	`d01aqc`
1-D quadrature, adaptive, semi-finite interval, weight function $\cos(\omega x)$ or $\sin(\omega x)$	`d01asc`
1-D Guassian quadrature	`d01bac`
Multi-dimensional adaptive quadrature over hyper-rectangle	`d01fcc`
1-D quadrature, integration of function defined by data values, Gill-Miller method	`d01gac`
Multi-dimensional quadrature over hyper-rectangle, Monte Carlo method	`d01gbc`

5.2. Thread-safe functions

1-D quadrature, adaptive, finite interval, allowing for badly-behaved integrands	`d01sjc`
1-D quadrature, adaptive, finite interval, method suitable for oscilating functions	`d01skc`
1-D quadrature, adaptive, finite interval, allowing for singularities at user-specified break-points	`d01slc`
1-D quadrature, adaptive, infinite or semi-infinite interval	`d01smc`

1-D quadrature, adaptive, finite interval, weight function $\cos(\omega x)$ or $\sin(\omega x)$	`d01snc`
1-D quadrature, adaptive, finite interval, weight function with end-point singularities of algebraico-logarithmic	`d01spc`
1-D quadrature, adaptive, finite interval, weight function $1/(x-c)$, Cauchy principal value (Hilbert transform)	`d01sqc`
1-D quadrature, adaptive, semi-finite interval, weight function $\cos(\omega x)$ or $\sin(\omega x)$	`d01ssc`
1-D Guassian quadrature	`d01tac`
Multi-dimensional adaptive quadrature over hyper-rectangle	`d01wcc`
Multi-dimensional quadrature over hyper-rectangle, Monte Carlo method	`d01xbc`