NAG Library Chapter Introduction

d01 – Quadrature

Contents

1 Scope of the Chapter

This chapter provides functions for the numerical evaluation of definite integrals in one or more dimensions and for evaluating weights and abscissae of integration rules.

2 Background to the Problems

The functions in this chapter are designed to estimate:

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

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

where $f(x)$ is defined by you, either at a set of points $(x_i, f(x_i))$, for $i = 1, 2, \ldots, n$, where $a = x_1 < x_2 < \cdots < x_n = b$, 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 factor $w(x)$, called the weight-function, of a specific form. These methods take full account of any peculiar behaviour attributable to the $w(x)$ factor.

(b) the value of a multidimensional definite integral of the form

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

where $f(x_1, x_2, \ldots, x_n)$ is a function defined by you and R_n is some region of n-dimensional space.

The simplest form of R_n is the n-rectangle defined by

$$a_i \leq x_i \leq b_i, \quad i = 1, 2, \ldots, n \tag{4}$$

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 page 266 of Davis and Rabinowitz (1975)). Some of the methods described incorporate the transformation procedure.

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.,

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

The points x_i within the interval $[a, b]$ are known as the abscissae, and the w_i are known as the weights.

More generally, if the integrand has the form (2), the corresponding formula is

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

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 at any abscissa is easily obtained, then a wide variety of quadrature rules are available, each characterised by its choice of abscissae and the corresponding weights.

The appropriate rule to use will depend on the interval $[a, b]$ – whether finite or otherwise – and on the form of any $w(x)$ factor in the integrand. A suitable value of N depends on the general 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.

The choice of basic rules constitutes one of the principles on which methods for one-dimensional integrals may be classified. The other major basis of classification is the implementation strategy, of which some types are now presented.

(a) Single rule evaluation procedures

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.

(b) Automatic procedures

The number of abscissae, N, within $[a, b]$ is gradually increased until consistency is achieved to within a level of accuracy (absolute or relative) you requested. There are essentially two ways of doing this; hybrid forms of these two methods are also possible:

(i) whole interval procedures (non-adaptive)

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 procedures

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 to 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 emphasized that the automatic functions in this chapter cannot be assumed to be 100% reliable. In general, however, the reliability is very high.**

2.2 Multidimensional 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 methods can be used to deal with high dimensions.

(a) Products of one-dimensional rules

Using a two-dimensional integral as an example, we have

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

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

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 functions should be used with caution (see 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) Number theoretic methods

These are based on the work of Korobov and Conroy and operate by exploiting implicitly the properties of the Fourier expansion of the integrand. Special rules, constructed from so-called optimal coefficients, give a particularly uniform distribution of the points throughout n-dimensional space and from their number theoretic properties minimize the error on a prescribed class of integrals. The method can be combined with the Monte–Carlo procedure.

(d) Sag–Szekeres method

By transformation this method seeks to induce properties into the integrand which make it accurately integrable by the trapezoidal rule. The transformation also allows effective control over the number of integrand evaluations.

(e) Sparse grid methods

Given a set of one-dimensional quadrature rules of increasing levels of accuracy, the sparse grid method constructs an approximation to a multidimensional integral using d dimensional tensor products of the differences between rules of adjacent levels. This provides a lower theoretical accuracy than the methods in (a), the full grid approach, which is nonetheless still sufficient for various classes of sufficiently smooth integrands. Furthermore, it requries substantially fewer evaluations than the full grid approach. Specifically, if a one-dimensional quadrature rule has $N \sim O\left(2^{\ell}\right)$ points, the full grid will require $O\left(2^{ld}\right)$ function evaluations, whereas the sparse grid of level ℓ will require $O\left(2^{\ell} d^{\ell-1}\right)$. Hence a sparse grid approach is computationally feasible even for integrals over $d \sim O(100)$.

Sparse grid methods are deterministic, and may be viewed as automatic whole domain procedures if their level ℓ is allowed to increase.

(f) 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 Recommendations on Choice and Use of Available Functions

This section is divided into five subsections. The first subsection illustrates the difference between direct and reverse communication functions. The second subsection highlights the different levels of vectorization provided by different interfaces.

Sections 3.3, 3.4 and 3.5 consider in turn functions for: one-dimensional integrals over a finite interval, and over a semi-infinite or an infinite interval; and multidimensional integrals. Within each sub-section, functions are classified by the type of method, which ranges from simple rule evaluation to automatic adaptive algorithms. The recommendations apply particularly when the primary objective is simply to compute the value of one or more integrals, and in these cases the automatic adaptive functions are generally the most convenient and reliable, although also the most expensive in computing time.

Note however that in some circumstances it may be counter-productive to use an automatic function. If the results of the quadrature are to be used in turn as input to a further computation (e.g., an 'outer' quadrature or an optimization problem), then this further computation may be adversely affected by the 'jagged performance profile' of an automatic function; a simple rule-evaluation function may provide much better overall performance. For further guidance, the article by Lyness (1983) is recommended.

3.1 Direct and Reverse Communication

Functions in this chapter which evaluate an integral value may be classified as either direct communication or reverse communication. See Section 3.2.2 in the Essential Introduction for a description of these terms.

Currently in this chapter the only function explicitly using reverse communication is nag_quad_1d_gen_vec_multi_rcomm (d01rac).

3.2 Choice of Interface

This section concerns the design of the interface for the provision of abscissae, and the subsequent collection of calculated information, typically integrand evaluations. Vectorized interfaces typically allow for more efficient operation.

(a) Single abscissa interfaces

The algorithm will provide a single abscissa at which information is required. These are typically the most simple to use, although they may be significantly less efficient than a vectorized equivalent. Most of the algorithms in this chapter are of this type.

Examples of this include nag_quad_md_gauss (d01fbc) and nag_1d_quad_gen_1 (d01sjc).

(b) Vectorized abscissae interfaces

The algorithm will return a set of abscissae, at all of which information is required. While these are more complicated to use, they are typically more efficient than a non-vectorized equivalent. They reduce the overhead of function calls, allow the avoidance of repetition of computations common to each of the integrand evaluations, and offer greater scope for vectorization and parallelization of your code.

Examples include nag_quad_1d_fin_gonnet_vec (d01rgc) and nag_quad_1d_gauss_vec (d01uac).

(c) Multiple integral interfaces

These are functions which allow for multiple integrals to be estimated simultaneously. As with (b) above, these are more complicated to use than single integral functions, however they can provide

higher efficiency, particularly if several integrals require the same subcalculations at the same abscissae. They are most efficient if integrals which are supplied together are expected to have similar behaviour over the domain, particularly when the algorithm is adaptive.

nag_quad_1d_gen_vec_multi_rcomm (d01rac) is an example.

3.3 One-dimensional Integrals over a Finite Interval

(a) Integrand defined at a set of points

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 function may be unreliable. In order to check results independently and so as to provide an alternative technique you may fit the integrand by Chebyshev series using nag_1d_cheb_fit (e02adc) and then use function nag_1d_cheb_intg (e02ajc) to evaluate its integral (which need not be restricted to the range of the integration points, as is the case for nag_1d_quad_vals (d01gac)). A further alternative is to fit a cubic spline to the data using nag_1d_spline_fit_knots (e02bac) and then to 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 you do not wish to make any assumptions about the integrand, the most reliable functions to use will be nag_quad_1d_gen_vec_multi_rcomm (d01rac), nag_quad_1d_fin_gonnet_vec (d01rgc), nag_1d_quad_gen_1 (d01sjc), nag_1d_quad_osc_1 (d01skc) and nag_1d_quad_brkpts_1 (d01slc), although these will in general be less efficient for simple integrals.**

(i) Rule-evaluation functions

If $f(x)$ is known to be sufficiently well behaved (more precisely, can be closely approximated by a polynomial of moderate degree), a Gaussian function with a suitable number of abscissae may be used.

nag_quad_1d_gauss_wset (d01tbc) or nag_quad_1d_gauss_wgen (d01tcc) with nag_quad_md_gauss (d01fbc) may be used if it is required to examine the weights and abscissae.

nag_quad_1d_gauss_wset (d01tbc) is faster and more accurate, whereas nag_quad_1d_gauss_wgen (d01tcc) is more general. nag_quad_1d_gauss_vec (d01uac) uses the same quadrature rules as nag_quad_1d_gauss_wset (d01tbc), and may be used if you do not explicitly require the weights and abscissae.

If $f(x)$ is well behaved, apart from a weight-function of the form

$$\left|x-\frac{a+b}{2}\right|^c \quad \text{or} \quad (b-x)^c(x-a)^d,$$

nag_quad_1d_gauss_wgen (d01tcc) with nag_quad_md_gauss (d01fbc) may be used.

(ii) Automatic whole-interval functions

If $f(x)$ is reasonably smooth, and the required accuracy is not too high, the automatic whole interval function nag_quad_1d_fin_smooth (d01bdc) may be used. Additionally, nag_quad_md_sgq_multi_vec (d01esc) with $d=1$ may be used with an appropriate transformation from the unit interval.

nag_quad_1d_fin_smooth (d01bdc) uses the Gauss 10-point rule, with the 21 point Kronrod extension, and the subsequent 43 and 87 point Patterson extensions if required.

nag_quad_md_sgq_multi_vec (d01esc) supports multiple simultaneous integrals, and has a vectorized interface. Either high order Gauss–Patterson rules (of size $2^\ell-1$, for $\ell=1,\ldots,9$), or high order Clenshaw-Curtis rules (of size $2^{\ell-1}+1$, for $\ell=2,\ldots,12$). Gauss–Patterson rules

possess greater polynomial accuracy, whereas Clenshaw–Curtis rules are often well suited to oscillatory integrals.

(iii) Automatic adaptive functions

Firstly, several functions 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.

1. if $w(x) = (b-x)^{\alpha}(x-a)^{\beta}(\log(b-x))^{k}(\log(x-a))^{l}$, where $k, l = 0$ or 1, $\alpha, \beta > -1$: use nag_1d_quad_wt_alglog_1 (d01spc);
2. if $w(x) = \frac{1}{x-c}$: use nag_1d_quad_wt_cauchy_1 (d01sqc) (this integral is called the Hilbert transform of g);
3. if $w(x) = \cos(\omega x)$ or $\sin(\omega x)$: use nag_1d_quad_wt_trig_1 (d01snc) (this function can also handle certain types of singularities in $g(x)$).

Secondly, there are multiple routines for general $f(x)$, using different strategies.

nag_1d_quad_gen_1 (d01sjc) and nag_1d_quad_osc_1 (d01skc) use the strategy of Piessens *et al.* (1983), using repeated bisection of the interval, and in the first case the ϵ-algorithm (Wynn (1956)), to improve the integral estimate. This can cope with singularities away from the end points, provided singular points do not occur as abscissae, nag_1d_quad_osc_1 (d01skc) tends to perform better than nag_1d_quad_gen_1 (d01sjc) on more oscillatory integrals.

nag_1d_quad_brkpts_1 (d01slc) uses the same subdivision strategy as nag_1d_quad_gen_1 (d01sjc) over a set of initial interval segments determined by supplied break-points. It is hence suitable for integrals with discontinuities (including switches in definition) or sharp peaks occuring at known points. Such integrals may also be approximated using other functions which do not allow break-points, although such integrals should be evaluated over each of the sub-intervals seperately.

nag_quad_1d_gen_vec_multi_rcomm (d01rac) again uses the strategy of Piessens *et al.* (1983), and provides the functionality of nag_1d_quad_gen_1 (d01sjc), nag_1d_quad_osc_1 (d01skc) and nag_1d_quad_brkpts_1 (d01slc) in a reverse communication framework. It also supports multiple integrals and uses a vectorized interface for the abscissae. Hence it is likely to be more efficient if several similar integrals are required to be evaluated over the same domain. Furthermore, its behaviour can be tailored through the use of optional arguments.

nag_quad_1d_fin_gonnet_vec (d01rgc) uses another adaptive scheme due to Gonnet (2010). This attempts to match the quadrature rule to the underlying integrand as well as subdividing the domain. Further, it can explicitly deal with singular points at abscissae, should NaN's or ∞ be returned by the user-supplied function, provided the generation of these does not cause the program to halt (see Chapter x07).

3.4 One-dimensional Integrals over a Semi-infinite or Infinite Interval

(a) Integrand defined at 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 functions

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

1. $e^{-\beta x}, \beta > 0$ (semi-infinite interval, lower limit finite); or
2. $e^{-\beta x}, \beta < 0$ (semi-infinite interval, upper limit finite); or
3. $e^{-\beta(x-\alpha)^2}, \beta > 0$ (infinite interval),

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

nag_quad_1d_gauss_vec (d01uac) may be used if it is not required to examine the weights and abscissae.

nag_quad_1d_gauss_wset (d01tbc) or nag_quad_1d_gauss_wgen (d01tcc) with nag_quad_md_gauss (d01fbc) may be used if it is required to examine the weights and abscissae.

nag_quad_1d_gauss_wset (d01tbc) is faster and more accurate, whereas nag_quad_1d_gauss_wgen (d01tcc) is more general.

(ii) Automatic adaptive functions

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 range. 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).

The following alternative procedures are mentioned for completeness, though their use will rarely be necessary.

1. If the integrand decays rapidly towards an infinite end point, a finite cut-off may be chosen, and the finite range methods applied.
2. If the only irregularities occur in the finite part (apart from a singularity at the finite limit, with which nag_1d_quad_inf_1 (d01smc) can cope), the range may be divided, with nag_1d_quad_inf_1 (d01smc) used on the infinite part.
3. A transformation to finite range may be employed, e.g.,

$$x = \frac{1-t}{t} \quad \text{or} \quad x = -\log_{\mathrm{e}} t$$

will transform $(0,\infty)$ to $(1,0)$ while for infinite ranges we have

$$\int_{-\infty}^{\infty} f(x)\,dx = \int_{0}^{\infty} (f(x)+f(-x))\,dx.$$

If the integrand behaves badly on $(-\infty,0)$ and well on $(0,\infty)$ or vice versa it is better to compute it as $\displaystyle\int_{-\infty}^{0} f(x)\,dx + \int_{0}^{\infty} f(x)\,dx$. This saves computing unnecessary function values in the semi-infinite range where the function is well behaved.

3.5 Multidimensional Integrals

A number of techniques are available in this area and the choice depends to a large extent on the dimension and the required accuracy. It can be advantageous to use more than one technique as a confirmation of accuracy, particularly for high-dimensional integrations. Several functions include a transformation procedure, using a user-supplied function, which allows general product regions to be easily dealt with in terms of conversion to the standard n-cube region.

(a) Products of one-dimensional rules (suitable for up to about 5 dimensions)

If $f(x_1, x_2, \ldots, x_n)$ is known to be a sufficiently well behaved function of each variable x_i, apart possibly from weight functions of the types provided, a product of Gaussian rules may be used. These are provided by nag_quad_1d_gauss_wset (d01tbc) or nag_quad_1d_gauss_wgen (d01tcc) with nag_quad_md_gauss (d01fbc). Rules for finite, semi-infinite and infinite ranges are included.

For two-dimensional integrals only, unless the integrand is very badly behaved, the automatic whole-interval product procedure of nag_quad_2d_fin (d01dac) may be used. The limits of the inner integral may be user-specified functions of the outer variable. Infinite limits may be handled by

transformation (see Section 3.4); end point singularities introduced by transformation should not be troublesome, as the integrand value will not be required on the boundary of the region.

If none of these functions proves suitable and convenient, the one-dimensional functions may be used recursively. For example, the two-dimensional integral

$$I = \int_{a_1}^{b_1} \int_{a_2}^{b_2} f(x, y)\, dy\, dx$$

may be expressed as

$$I = \int_{a_1}^{b_1} F(x)\, dx, \qquad \text{where} \qquad F(x) = \int_{a_2}^{b_2} f(x, y)\, dy.$$

The user-supplied code to evaluate $F(x)$ will call the integration function for the y-integration, which will call more user-supplied code for $f(x, y)$ as a function of y (x being effectively a constant).

From Mark 24 onwards, all direct communication functions may be called recursively. As such, **you may use any function, including the same function, for each dimension**. Note however, **in previous releases**, some direct communication functions, specifically, nag_quad_1d_fin_smooth (d01bdc), nag_quad_2d_fin (d01dac), nag_quad_md_gauss (d01fbc), nag_quad_md_sphere (d01fdc), nag_quad_md_numth_vec (d01gdc) and nag_quad_md_simplex (d01pac), could not be called recursively.

The reverse communication function nag_quad_1d_gen_vec_multi_rcomm (d01rac) may be used by itself in a pseudo-recursive manner, in that it may be called to evaluate an inner integral for the integrand value of an outer integral also being calculated by nag_quad_1d_gen_vec_multi_rcomm (d01rac).

(b) Sag–Szekeres method

nag_quad_md_sphere (d01fdc) is particularly suitable for integrals of very large dimension although the accuracy is generally not high. It allows integration over either the general product region (with built-in transformation to the n-cube) or the n-sphere. Although no error estimate is provided, two adjustable arguments may be varied for checking purposes or may be used to tune the algorithm to particular integrals.

(c) Number Theoretic method

Algorithms of this type carry out multidimensional integration using the Korobov–Conroy method over a product region with built-in transformation to the n-cube. A stochastic modification of this method is incorporated into the functions in this library, hybridising the technique with the Monte–Carlo procedure. An error estimate is provided in terms of the statistical standard error. A number of pre-computed optimal coefficient rules for up to 20 dimensions are provided; others can be computed using nag_quad_md_numth_coeff_prime (d01gyc) and nag_quad_md_numth_coeff_2prime (d01gzc). Like the Sag–Szekeres method it is suitable for large dimensional integrals although the accuracy is not high.

nag_quad_md_numth_vec (d01gdc) has a vectorized interface which can result in faster execution, especially on vector-processing machines. You are required to provide two functions, the first to return an array of values of the integrand at each of an array of points, and the second to evaluate the limits of integration at each of an array of points. This reduces the overhead of function calls, avoids repetitions of computations common to each of the evaluations of the integral and limits of integration, and offers greater scope for vectorization of your code.

(d) A combinatorial extrapolation method

nag_quad_md_simplex (d01pac) computes a sequence of approximations and an error estimate to the integral of a function over a multidimensional simplex using a combinatorial method with extrapolation.

(e) Sparse Grid method

nag_quad_md_sgq_multi_vec (d01esc) implements a sparse grid quadrature scheme for the integration of a vector of multidimensional integrals over the unit hypercube,

$$\mathbf{F} \approx \int_{[0,1]^d} \mathbf{f}(\mathbf{x}) d\mathbf{x}.$$

The function uses a vectorized interface, which returns a set of points at which the integrands must be evaluated in a sparse storage format for efficiency.

Other domains can be readily integrated over by using an appropriate mapping inside the provided function for evaluating the integrands. It is suitable for d up to $O(100)$, although no upper bound on the number of dimensions is enforced. It will also evaluate one-dimensional integrals, although in this case the sparse grid used is in fact the full grid.

The function uses optional arguments, set and queried using the functions nag_quad_opt_set (d01zkc) and nag_quad_opt_get (d01zlc) respectively. Amongst other options, these allow the parallelization of the function to be controlled.

(f) Automatic functions (nag_multid_quad_adapt_1 (d01wcc) and nag_multid_quad_monte_carlo_1 (d01xbc))

Both functions are for integrals of the form

$$\int_{a_1}^{b_1} \int_{a_2}^{b_2} \cdots \int_{a_n}^{b_n} f(x_1, x_2, \ldots, x_n)\, dx_n dx_{n-1} \cdots dx_1.$$

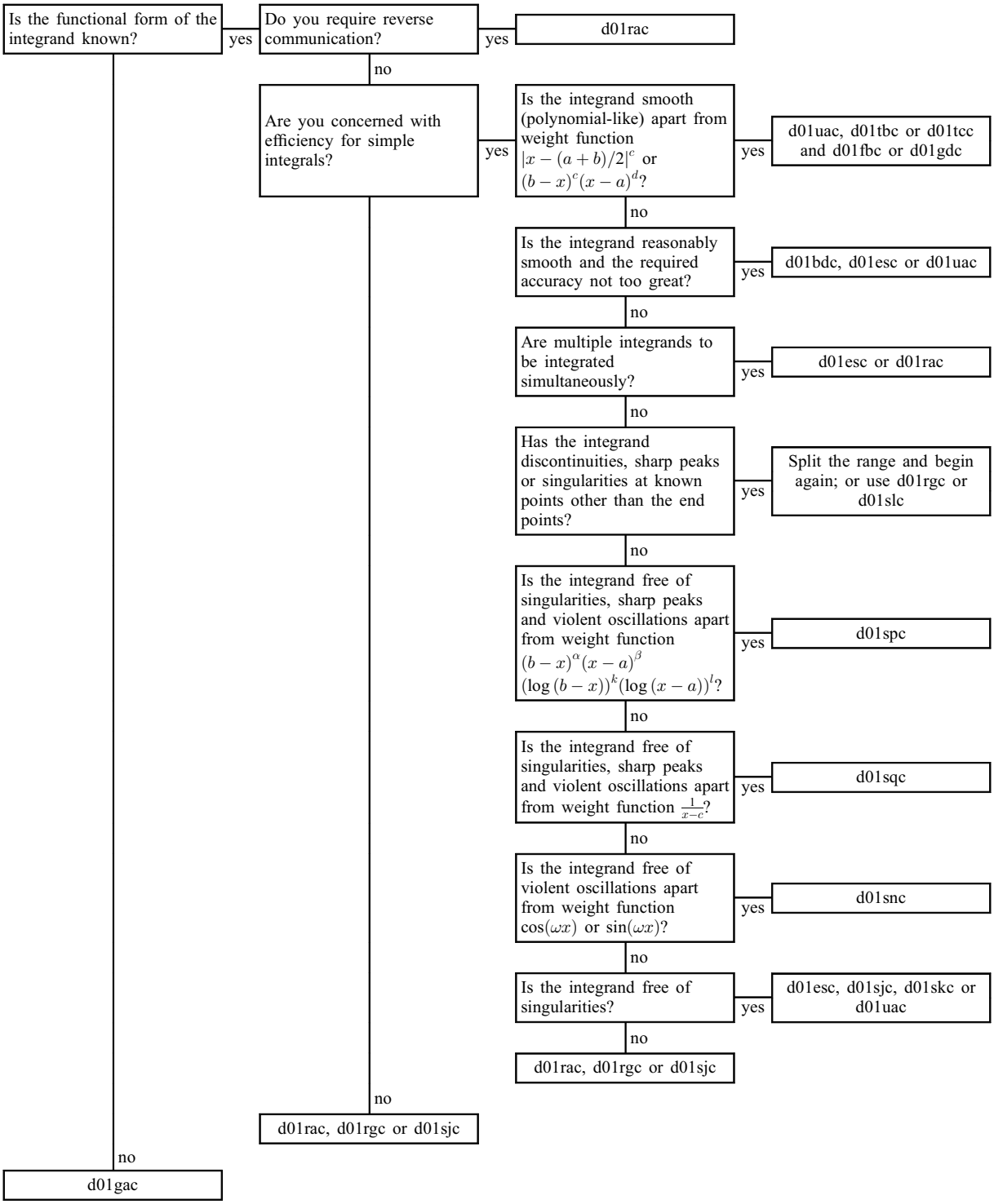
nag_multid_quad_monte_carlo_1 (d01xbc) is an adaptive Monte–Carlo function. This function is usually slow and not recommended for high-accuracy work. It is a robust function that can often be used for low-accuracy results with highly irregular integrands or when n is large.

nag_multid_quad_adapt_1 (d01wcc) is an adaptive deterministic function. 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 the Monte–Carlo function nag_multid_quad_monte_carlo_1 (d01xbc). The function 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 from at least two different functions.

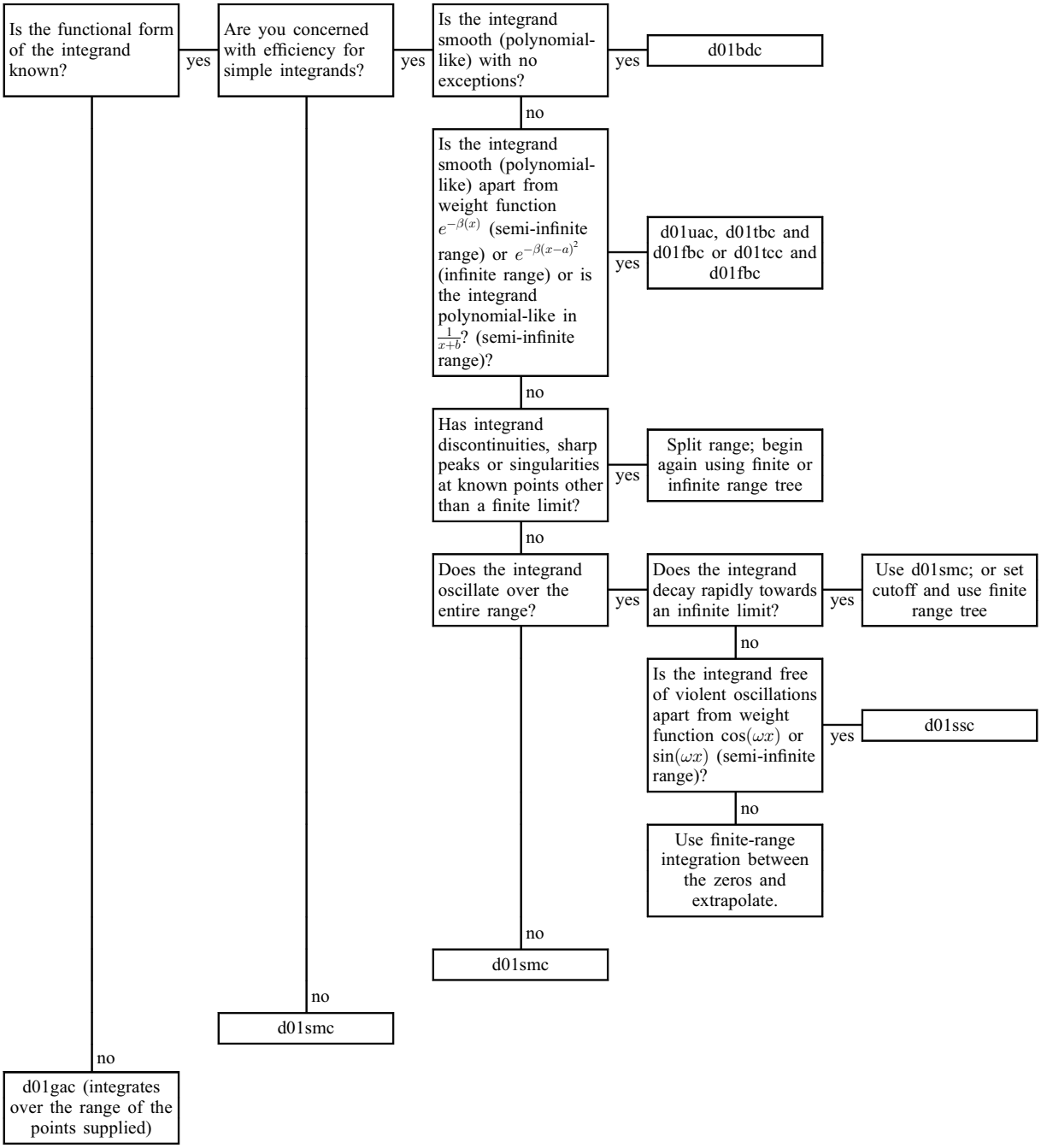
There are many problems for which one or both of the functions will require large amounts of computing time to obtain even moderately accurate results. The amount of computing time is controlled by the number of integrand evaluations you have allowed, and you should set this argument carefully, with reference to the time available and the accuracy desired.

4 Decision Trees

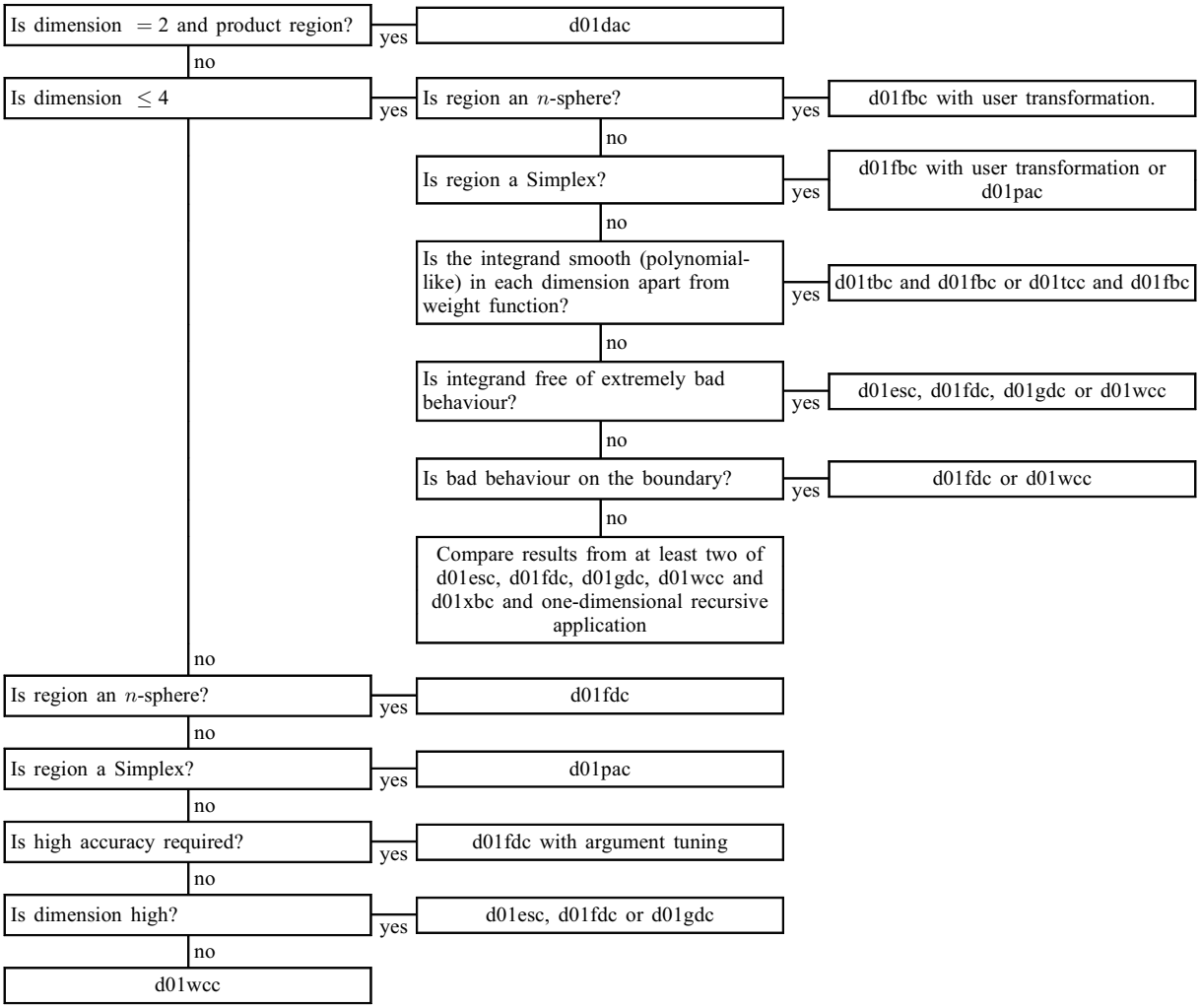
Tree 1: One-dimensional integrals over a finite interval



Tree 2: One-dimensional integrals over a semi-infinite or infinite interval



Tree 3: Multidimensional integrals



5 Functionality Index

Korobov optimal coefficients for use in nag_quad_md_numth_vec (d01gdc):
  when number of points is a product of 2 primes ..... nag_quad_md_numth_coeff_2prime (d01gzc)
  when number of points is prime ..... nag_quad_md_numth_coeff_prime (d01gyc)

Multidimensional quadrature,
  over a finite two-dimensional region ..... nag_quad_2d_fin (d01dac)
  over a general product region,
    Korobov–Conroy number-theoretic method ..... nag_quad_md_numth_vec (d01gdc)
    Sag–Szekeres method (also over n-sphere) ..... nag_quad_md_sphere (d01fdc)
  over a hyper-rectangle,
    adaptive method ..... nag_multid_quad_adapt_1 (d01wcc)
    Gaussian quadrature rule-evaluation ..... nag_quad_md_gauss (d01fbc)
    Monte–Carlo method ..... nag_multid_quad_monte_carlo_1 (d01xbc)
    sparse grid method (with user transformation),
      muliple integrands, vectorized interface ..... nag_quad_md_sgq_multi_vec (d01esc)
  over an n-simplex ..... nag_quad_md_simplex (d01pac)

One-dimensional quadrature,
  adaptive integration of a function over a finite interval,
    strategy due to Gonnet,
      suitable for badly behaved integrals,
        vectorized interface ..... nag_quad_1d_fin_gonnet_vec (d01rgc)

strategy due to Piessens and de Doncker,
allowing for badly behaved integrands ..... nag_1d_quad_gen_1 (d01sjc)
allowing for singularities at user-specified break-points ..... nag_1d_quad_brkpts_1 (d01slc)
suitable for highly oscillatory integrals ..... nag_1d_quad_osc_1 (d01skc)
weight function $1/(x-c)$ Cauchy principal value (Hilbert transform) ..... nag_1d_quad_wt_cauchy_1 (d01sqc)
weight function $\cos(\omega x)$ or $\sin(\omega x)$ ..... nag_1d_quad_wt_trig_1 (d01snc)
weight function with end-point singularities of algebraico-logarithmic type ..... nag_1d_quad_wt_alglog_1 (d01spc)
adaptive integration of a function over an infinite interval or semi-infinite interval,
no weight function ..... nag_1d_quad_inf_1 (d01smc)
weight function $\cos(\omega x)$ or $\sin(\omega x)$ ..... nag_1d_quad_inf_wt_trig_1 (d01ssc)
integration of a function defined by data values only,
Gill–Miller method ..... nag_1d_quad_vals (d01gac)
non-adaptive integration over a finite, semi-infinite or infinite interval,
using pre-computed weights and abscissae
vectorized interface ..... nag_quad_1d_gauss_vec (d01uac)
non-adaptive integration over a finite interval ..... nag_quad_1d_fin_smooth (d01bdc)
reverse communication,
adaptive integration over a finite interval,
multiple integrands,
efficient on vector machines ..... nag_quad_1d_gen_vec_multi_rcomm (d01rac)

Service functions,
array size query for nag_quad_1d_gen_vec_multi_rcomm (d01rac) ..... nag_quad_1d_gen_vec_multi_dimreq (d01rcc)
general option getting ..... nag_quad_opt_get (d01zlc)
general option setting and initialization ..... nag_quad_opt_set (d01zkc)

Weights and abscissae for Gaussian quadrature rules,
more general choice of rule,
calculating the weights and abscissae ..... nag_quad_1d_gauss_wgen (d01tcc)
restricted choice of rule,
using pre-computed weights and abscissae ..... nag_quad_1d_gauss_wset (d01tbc)

6 Auxiliary Functions Associated with Library Function Arguments

None.

7 Functions Withdrawn or Scheduled for Withdrawal

The following lists all those functions that have been withdrawn since Mark 23 of the Library or are scheduled for withdrawal at one of the next two marks.

Withdrawn Function	Mark of Withdrawal	Replacement Function(s)
nag_1d_quad_gen (d01ajc)	24	nag_1d_quad_gen_1 (d01sjc)
nag_1d_quad_osc (d01akc)	24	nag_1d_quad_osc_1 (d01skc)
nag_1d_quad_brkpts (d01alc)	24	nag_1d_quad_brkpts_1 (d01slc)
nag_1d_quad_inf (d01amc)	24	nag_1d_quad_inf_1 (d01smc)
nag_1d_quad_wt_trig (d01anc)	24	nag_1d_quad_wt_trig_1 (d01snc)
nag_1d_quad_wt_alglog (d01apc)	24	nag_1d_quad_wt_alglog_1 (d01spc)
nag_1d_quad_wt_cauchy (d01aqc)	24	nag_1d_quad_wt_cauchy_1 (d01sqc)
nag_1d_quad_inf_wt_trig (d01asc)	24	nag_1d_quad_inf_wt_trig_1 (d01ssc)
nag_1d_quad_gauss (d01bac)	24	nag_quad_1d_gauss_vec (d01uac)
nag_multid_quad_adapt (d01fcc)	25	nag_multid_quad_adapt_1 (d01wcc)
nag_multid_quad_monte_carlo (d01gbc)	25	nag_multid_quad_monte_carlo_1 (d01xbc)
nag_1d_quad_gauss_1 (d01tac)	27	nag_quad_1d_gauss_vec (d01uac)

8 References

Davis P J and Rabinowitz P (1975) *Methods of Numerical Integration* Academic Press

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

Lyness J N (1983) When not to use an automatic quadrature routine *SIAM Rev.* **25** 63–87

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

Sobol I M (1974) *The Monte Carlo Method* The University of Chicago Press

Stroud A H (1971) *Approximate Calculation of Multiple Integrals* Prentice–Hall

Wynn P (1956) On a device for computing the $e_m(S_n)$ transformation *Math. Tables Aids Comput.* **10** 91–96