Golub, G.H. 1973, *SIAM Review*, vol. 15, pp. 318–334. [10]

Kronrod, A.S. 1964, *Doklady Akademii Nauk SSSR*, vol. 154, pp. 283–286 (in Russian). [11]

Patterson, T.N.L. 1968, *Mathematics of Computation*, vol. 22, pp. 847–856 and C1–C11; 1969, *op. cit.*, vol. 23, p. 892. [12]

Piessens, R., de Doncker, E., Uberhuber, C.W., and Kahaner, D.K. 1983, *QUADPACK: A Subroutine Package for Automatic Integration* (New York: Springer-Verlag). [13]

Stoer, J., and Bulirsch, R. 1980, *Introduction to Numerical Analysis* (New York: Springer-Verlag), §3.6.

Johnson, L.W., and Riess, R.D. 1982, *Numerical Analysis*, 2nd ed. (Reading, MA: Addison-Wesley), §6.5.

Carnahan, B., Luther, H.A., and Wilkes, J.O. 1969, *Applied Numerical Methods* (New York: Wiley), §§2.9–2.10.

Ralston, A., and Rabinowitz, P. 1978, *A First Course in Numerical Analysis*, 2nd ed. (New York: McGraw-Hill), §§4.4–4.8.

4.6 Multidimensional Integrals

Integrals of functions of several variables, over regions with dimension greater than one, are *not easy*. There are two reasons for this. First, the number of function evaluations needed to sample an N-dimensional space increases as the Nth power of the number needed to do a one-dimensional integral. If you need 30 function evaluations to do a one-dimensional integral crudely, then you will likely need on the order of 30000 evaluations to reach the same crude level for a three-dimensional integral. Second, the region of integration in N-dimensional space is defined by an $N - 1$ dimensional boundary which can itself be terribly complicated: It need not be convex or simply connected, for example. By contrast, the boundary of a one-dimensional integral consists of two numbers, its upper and lower limits.

The first question to be asked, when faced with a multidimensional integral, is, "can it be reduced analytically to a lower dimensionality?" For example, so-called *iterated integrals* of a function of one variable $f(t)$ can be reduced to one-dimensional integrals by the formula

$$\begin{aligned}\int_0^x dt_n \int_0^{t_n} dt_{n-1} \cdots \int_0^{t_3} dt_2 \int_0^{t_2} f(t_1)dt_1 \\ = \frac{1}{(n-1)!}\int_0^x (x-t)^{n-1} f(t)dt\end{aligned} \tag{4.6.1}$$

Alternatively, the function may have some special symmetry in the way it depends on its independent variables. If the boundary also has this symmetry, then the dimension can be reduced. In three dimensions, for example, the integration of a spherically symmetric function over a spherical region reduces, in polar coordinates, to a one-dimensional integral.

The next questions to be asked will guide your choice between two entirely different approaches to doing the problem. The questions are: Is the shape of the boundary of the region of integration simple or complicated? Inside the region, is the integrand smooth and simple, or complicated, or locally strongly peaked? Does the problem require high accuracy, or does it require an answer accurate only to a percent, or a few percent?

If your answers are that the boundary is complicated, the integrand is *not* strongly peaked in very small regions, and relatively low accuracy is tolerable, then your problem is a good candidate for *Monte Carlo integration*. This method is very straightforward to program, in its cruder forms. One needs only to know a region with simple boundaries that *includes* the complicated region of integration, plus a method of determining whether a random point is inside or outside the region of integration. Monte Carlo integration evaluates the function at a random sample of points, and estimates its integral based on that random sample. We will discuss it in more detail, and with more sophistication, in Chapter 7.

If the boundary is simple, and the function is very smooth, then the remaining approaches, breaking up the problem into repeated one-dimensional integrals, or multidimensional Gaussian quadratures, will be effective and relatively fast [1]. If you require high accuracy, these approaches are in any case the *only* ones available to you, since Monte Carlo methods are by nature asymptotically slow to converge.

For low accuracy, use repeated one-dimensional integration or multidimensional Gaussian quadratures when the integrand is slowly varying and smooth in the region of integration, Monte Carlo when the integrand is oscillatory or discontinuous, but not strongly peaked in small regions.

If the integrand *is* strongly peaked in small regions, and you know where those regions are, break the integral up into several regions so that the integrand is smooth in each, and do each separately. If you don't know where the strongly peaked regions are, you might as well (at the level of sophistication of this book) quit: It is hopeless to expect an integration routine to search out unknown pockets of large contribution in a huge N-dimensional space. (But see §7.8.)

If, on the basis of the above guidelines, you decide to pursue the repeated one-dimensional integration approach, here is how it works. For definiteness, we will consider the case of a three-dimensional integral in x, y, z-space. Two dimensions, or more than three dimensions, are entirely analogous.

The first step is to specify the region of integration by (i) its lower and upper limits in x, which we will denote x_1 and x_2; (ii) its lower and upper limits in y at a specified value of x, denoted $y_1(x)$ and $y_2(x)$; and (iii) its lower and upper limits in z at specified x and y, denoted $z_1(x, y)$ and $z_2(x, y)$. In other words, find the numbers x_1 and x_2, and the functions $y_1(x)$, $y_2(x)$, $z_1(x, y)$, and $z_2(x, y)$ such that

$$\begin{aligned} I &\equiv \int\int\int dx\,dy\,dz f(x, y, z) \\ &= \int_{x_1}^{x_2} dx \int_{y_1(x)}^{y_2(x)} dy \int_{z_1(x,y)}^{z_2(x,y)} dz\; f(x, y, z) \end{aligned} \tag{4.6.2}$$

For example, a two-dimensional integral over a circle of radius one centered on the origin becomes

$$\int_{-1}^{1} dx \int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} dy\; f(x, y) \tag{4.6.3}$$

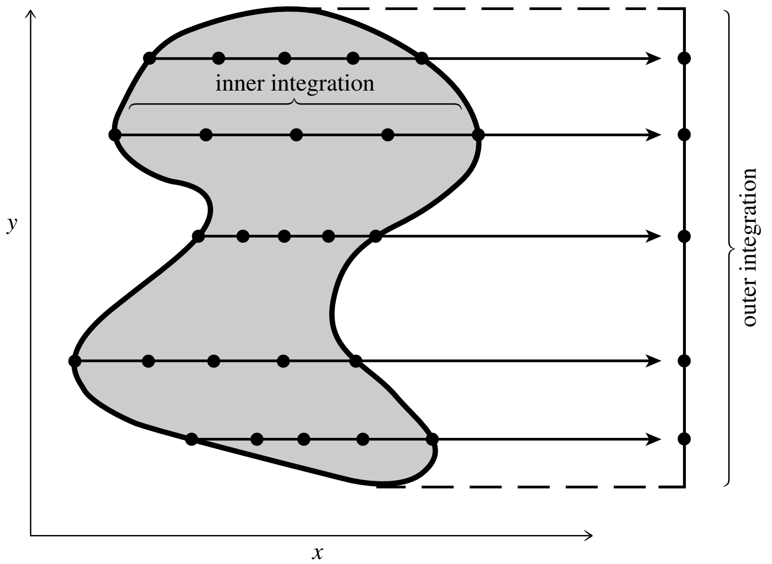

Figure 4.6.1. Function evaluations for a two-dimensional integral over an irregular region, shown schematically. The outer integration routine, in y, requests values of the inner, x, integral at locations along the y axis of its own choosing. The inner integration routine then evaluates the function at x locations suitable to *it*. This is more accurate in general than, e.g., evaluating the function on a Cartesian mesh of points.

Now we can define a function $G(x, y)$ that does the innermost integral,

$$G(x, y) \equiv \int_{z_1(x,y)}^{z_2(x,y)} f(x, y, z)dz \tag{4.6.4}$$

and a function $H(x)$ that does the integral of $G(x, y)$,

$$H(x) \equiv \int_{y_1(x)}^{y_2(x)} G(x, y)dy \tag{4.6.5}$$

and finally our answer as an integral over $H(x)$

$$I = \int_{x_1}^{x_2} H(x)dx \tag{4.6.6}$$

In an implementation of equations (4.6.4)–(4.6.6), some basic one-dimensional integration routine (e.g., `qgaus` in the program following) gets called recursively: once to evaluate the outer integral I, then many times to evaluate the middle integral H, then even more times to evaluate the inner integral G (see Figure 4.6.1). Current values of x and y, and the pointer to your function `func`, are passed "over the head" of the intermediate calls through static top-level variables.

```
static float xsav,ysav;
static float (*nrfunc)(float,float,float);

float quad3d(float (*func)(float, float, float), float x1, float x2)
```
Returns the integral of a user-supplied function `func` over a three-dimensional region specified by the limits `x1`, `x2`, and by the user-supplied functions `yy1`, `yy2`, `z1`, and `z2`, as defined in (4.6.2). (The functions y_1 and y_2 are here called `yy1` and `yy2` to avoid conflict with the names of Bessel functions in some C libraries). Integration is performed by calling `qgaus` recursively.
```
{
    float qgaus(float (*func)(float), float a, float b);
    float f1(float x);

    nrfunc=func;
    return qgaus(f1,x1,x2);
}

float f1(float x)                    This is H of eq. (4.6.5).
{
    float qgaus(float (*func)(float), float a, float b);
    float f2(float y);
    float yy1(float),yy2(float);

    xsav=x;
    return qgaus(f2,yy1(x),yy2(x));
}

float f2(float y)                    This is G of eq. (4.6.4).
{
    float qgaus(float (*func)(float), float a, float b);
    float f3(float z);
    float z1(float,float),z2(float,float);

    ysav=y;
    return qgaus(f3,z1(xsav,y),z2(xsav,y));
}

float f3(float z)                    The integrand f(x,y,z) evaluated at fixed x and y.
{
    return (*nrfunc)(xsav,ysav,z);
}
```

The necessary user-supplied functions have the following prototypes:

```
    float func(float x,float y,float z);      The 3-dimensional function to be inte-
    float yy1(float x);                          grated.
    float yy2(float x);
    float z1(float x,float y);
    float z2(float x,float y);
```

CITED REFERENCES AND FURTHER READING:

Stroud, A.H. 1971, *Approximate Calculation of Multiple Integrals* (Englewood Cliffs, NJ: Prentice-Hall). [1]

Dahlquist, G., and Bjorck, A. 1974, *Numerical Methods* (Englewood Cliffs, NJ: Prentice-Hall), §7.7, p. 318.

Johnson, L.W., and Riess, R.D. 1982, *Numerical Analysis*, 2nd ed. (Reading, MA: Addison-Wesley), §6.2.5, p. 307.

Abramowitz, M., and Stegun, I.A. 1964, *Handbook of Mathematical Functions*, Applied Mathematics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), equations 25.4.58ff.