Quadrature -- 'extras'

Romberg Integration

[This topic is really just so that you know the name in case you meet it elsewhere.]

As previously noted, if you know how the error [in any numerical process] behaves as a function of [eg] the number of strips, then we can firstly estimate it, and secondly use the estimate to get an improved estimate of [in this case] the integral. Romberg's idea was simply that the improved estimate [using Richardson extrapolation] itself has an error, which we can get a formula for, and which can therefore itself be improved; and the improvements have formulaic errors which can be improved; and so on.

More specifically, we start with, for example, the Trapezoidal Rule, $I \approx \frac{1}{2}(b-a)(f(a)+f(b))$, with a single strip. Down the left-hand column, write down successive results of the composite form of the Trapezoidal Rule with 2, 4, 8, ... strips, as far as necessary. In a second column, write down the result of using Richardson extrapolation on successive pairs of values. [This is exactly like what we did with Simpson's Rule except that the divisor $15 = 2^4 - 1$ is replaced by $3 = 2^2 - 1$.] In a third column, write down the result of extrapolating the second column. [The second column *is* Simpson's Rule.] In a fourth column, use extrapolation on the third column [using as divisor $63 = 2^6 - 1$]. Then a fifth column similarly [using divisor $255 = 2^8 - 1$], and so on.

This method is essentially 'free'; the hard work lies in evaluating the integrand at lots of x-values. So it's a Good Idea, but it doesn't get us much further unless we are using 16 or 32 strips, so it's more for computer than hand use. Also, it relies on the errors behaving 'properly'; nothing along these lines can rescue the poor convergence of all quadrature methods when the integrand is not well-behaved.

Gaussian Integration

[This topic is really just so that you know the name in case you meet it elsewhere.] Simpson's Rule and other Newton–Cotes formulas are based on the notion of passing a polynomial of degree n through $n+1$ points. But if we have a slightly more general rule, of form

$$I \approx a_1 f(x_1) + a_2 f(x_2) + a_3 f(x_3) + \ldots + a_n f(x_n),$$

then we can *choose* the a_i and the x_i [which will no longer be equally spaced] so that the formula is exact whenever $f(x)$ is a polynomial of degree $2n-1$. [There are $2n$ a's and x's to choose, and $2n$ arbitrary coefficients in the polynomial, so making the Rule exact gives us $2n$ equations in $2n$ unknowns.]

For each n, this optimal choice only has to be determined once. The resulting formula is the n-point Gauss's Rule. It is usually *much* more accurate than Simpson's Rule or Romberg integration with the same amount of work. *But* it's quite hard to determine the error without doing almost as much work again [which undoes the efficiency]. So the main application is to a series of integrals in which we can estimate in advance how many strips/points will be needed.

In the above form, you will also see the description Gauss–Legendre integration, as the x_i turn out to be the zeros of the so-called Legendre polynomials, which are important in all manner of bits of applied maths. You will also see the same general idea applied to various improper or weighted integrals. For example, we can use the zeros of the Laguerre polynomials [which are important *etc*] to find x_i so that the integral $\int_0^\infty f(x)\,\mathrm{e}^{-x}\mathrm{d}x$ is estimated exactly whenever f is a polynomial of degree $2n-1$ [Gauss–Laguerre integration], and Gauss–Hermite integration for integrals of form $\int_{-\infty}^{\infty} f(x)\,\mathrm{e}^{-x^2}\mathrm{d}x$, and lots of other variants.

Adaptive integration

[This topic is really just so that you know the name in case you meet it elsewhere.]

In integrals like $\int_0^1 \sqrt{x}\,\mathrm{d}x$, the problem was that the integrand behaved badly near $x = 0$; everywhere else, there is no problem, and Simpson's Rule works perfectly well. There is no point compositing the whole integral repeatedly, so that we use thousands or millions of strips between, say, $x = 0.1$ and $x = 1$.

The basic idea of adaptive integration is that instead of simply doubling the number of strips, we split the integral into two bits, and integrate each separately to within half the permitted error. Suppose we want $\int_0^1 \sqrt{x}\,\mathrm{d}x$ to within 0.000001. Then we find I_2 and I_4 as previously, and discover that the error is too large. So we instead find $\int_0^{\frac{1}{2}} \sqrt{x}\,\mathrm{d}x$ and $\int_{\frac{1}{2}}^1 \sqrt{x}\,\mathrm{d}x$ separately to within 0.0000005, and add the results. We already know I_2 for each of these, so we now work out I_4, estimate the error, and repeat.

In this particular case, each right-hand half will work well, so only a few strips will be needed; the left-hand halves will work badly, so we will continue to need to split them, until indeed the strip width is tiny. But the total work done remains quite small.

This idea is much more use for computer programs, as it provides a reasonable way of evaluating any integral that has only isolated pockets of bad behaviour without the need for thought. It concentrates the effort on the places where f needs lots of evaluations, but leaves the strips wide where possible. For hand/calculator use, keeping track of the calculation is quite hard, and the effort only pays off when the number of strips is quite large; so other strategies are nearly always better.

Line, volume, surface integrals.

[This topic is really just so that you know what to do in case you meet it elsewhere.]

General line and surface integrals can be reduced to 'ordinary' integrals using the parametrisation that describes the line or surface. So we are left with the problem of extending simple quadrature, as previously described, to the case of 'multiple' integrals, such as

$$I = \int\int\int f(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z,$$

with some more-or-less messy limits that depend, in general, on some of the variables.

You *can* use Simpson's Rule [as composited or otherwise adapted above] to evaluate each integral. But note the 'combinatorial explosion' problem: if we need, for example, just 8 strips in each direction, we are doing 9 evaluations of f for each x-integral, and so for each value of y; thus 81 for each y-integral, and so for each value of z; and 729 to integrate out over z. If we have to go to 16 strips, then we will need $17^3 = 4913$ evaluations of f, and the work is rapidly mounting up. Worse, many real-life problems involve more than three variables, and it is common for real-life volumes or surfaces to be rather messy [think, for example, the surface of, say, an aeroplane or a F1 car, or the volume of almost any machine or engineering construction]. Gets decidedly messy! Note that any 'corner' or 'edge' in the surface is equivalent to bad behaviour in the integral, so is likely to result in the need for many strips.

There is no universal *good* answer; a common practical answer on the computer is …

Monte-Carlo integration

[This topic is really just so that you know the name in case you meet it elsewhere.]

This is a generic technique that can be applied to all sorts of numerical problems. Suppose, for definiteness, we are trying to evaluate

$$I = \iiint_V f(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z$$

for some function f over some volume V. Choose a random point inside V. Then the value, call it $\hat{f}$, of f at that point is an *unbiassed estimator* of f inside V. So that value times the volume of V is an unbiassed estimate of I:

$$I \approx \hat{f}\times V.$$

The trouble is—of course it couldn't be *that* simple!—is that although the estimate is unbiassed, it has an outrageously large expected error, so it is essentially useless.

But we know how to reduce errors! We do the experiment lots of times, and average the results. We choose some large number N, choose N random points, then

$$I \approx (\hat{f}_1+\hat{f}_2+\ldots+\hat{f}_N)\times V/N.$$

and the expected error is reduced by a factor $\sqrt{N}$, and so can be made as small as we like. [There are other ways to reduce the error as well, which you can look up in the literature.]

The trouble is that we need N large; you will often need N = 1000 to get even 1sf, and so 100 000 to get 2sf, 10 000 000 to get 3sf, and so on—with care and luck you may be able to squeeze an extra figure. So this is not in any way competitive with ordinary quadrature. *Except* that ordinary quadrature starts to look silly when there are several dimensions and awkward volumes or surfaces, whereas Monte-Carlo doesn't care. It will take pretty much the same amount of work whether there is one dimension, or three or ten or a hundred, and whether or not there are corners. But this is definitely a method for the computer!