2
Numerical integration

2.1 Introduction

Numerical integration is a problem that is part of many problems in the economics and econometrics literature. The organization of this chapter is as follows. The first section covers quadrature procedures, which are the dominant way to solve models. The second section covers (pseudo) Monte Carlo integration techniques. The last section discusses quasi Monte Carlo integration.

2.2 Quadrature techniques

Suppose we want to calculate

$$I = \int_a^b f(x)dx, \tag{2.1}$$

where $f(x)$ is a scalar function. This could be a difficult problem, e.g., because the functional form is nasty or because we do not even have a functional form, but only a set of function values.

Quadrature techniques are numerical integration techniques for which the formula of the numerical integral can be written as

$$I = \int_a^b f(x)dx \approx \sum_{i=1}^{n} w_i f_i, \tag{2.2}$$

where f_i is the function value of f at node x_i and w_i is a weight. We will discuss two types of quadrature techniques. The first is Newton-Cotes. Newton-Cotes is not very careful about choosing the location of the nodes, but is clever about choosing the weights. The second is Gaussian Quadrature. This procedure is clever about choosing the weights as well as the nodes. To implement quadrature methods you can forget the details of the derivation. All you have to remember is how to construct what kind of weights and this is easy.

2.2.1 Newton-Cotes Quadrature

Consider the integration problem given in Equation 2.1 and suppose that one has three function values at three nodes. Given this information, how would one calculate the integral? Well, one could calculate an approximating function and calculate the integral for this approximating function. Since we have been given three points, we can calculate a second-order polynomial, $P_2(x)$, and get an estimate for the integral using

$$\int_a^b f(x)dx \approx \int_a^b P_2(x)dx. \tag{2.3}$$

Since integrating polynomials is easy, this procedure is straightforward. But one still has to find the approximation and do the integration. It turns out that these procedures can be standardized. That is, one can find the weights in (2.2) *independent of the functional form of* f. They do depend to some extend on the location of the nodes. We assume that $x_0 = a$, $x_1 = (a+b)/2$, and $x_2 = b$. That is, we have equidistant nodes and the first (last) node is the left (right) boundary. We have two segments of equal length, h and we can write $x_1 = x_0 + h$ and $x_2 = x_0 + 2h$.

Recall from the chapter on function approximation, that using Lagrange interpolation the second-order polynomial can be written as

$$P_2(x) = f_0 L_0(x) + f_1 L_1(x) + f_2 L_2(x). \tag{2.4}$$

This means that our approximating integral is given by

$$\begin{aligned}\int_a^b P_2(x)dx &= \int_a^b \left(f_0 L_0(x) + f_1 L_1(x) + f_2 L_2(x)\right) dx \\ &= f_0 \int_a^b L_0(x)dx + f_1 \int_a^b L_1(x)dx + f_2 \int_a^b L_2(x)dx\end{aligned}$$

The right-hand side already has the quadrature form as in Equation (2.2). The weights are the integrals. Key is that the integrals, i.e., the weights, do *not* depend on the function values. The nodes are pinned down by the value of x_0 and h. The beauty is that the integrals do not depend on x_0

either. Try to do the integration for one of them to ensure yourself that this is true. In particular, it is not difficult to show that

$$\int_a^b L_0(x)dx = \frac{1}{3}h$$
$$\int_a^b L_1(x)dx = \frac{4}{3}h$$
$$\int_a^b L_2(x)dx = \frac{1}{3}h$$

Simpson quadrature

Combining the results we get

$$\int_a^b f(x)dx \approx \int_a^b P_2(x)dx = \left(\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{1}{3}f_2\right)h.$$

This will give you an accurate answer if the function f can be approximated well with a second-order polynomial over the interval $[a, b]$. It will give you an exact answer for *any* second-order polynomial. But clearly this will not give you an accurate answer if the function you are integrating is more complex. Sticking to the original idea of integrating approximating polynomials, there are two ways to proceed. The first is to extend the idea to higher-order polynomials. The other is to use the same idea but to smaller intervals.

In particular, suppose that one has $n+1$ *equidistant* nodes and the distance between the nodes is h. The total number of nodes must be odd so that there are $n/2$ segments of length $2h$. On each of these segments of length $2h$ one then applies the above procedure. This would give

$$\begin{aligned}
\int_a^b f(x)dx &\approx \left(\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{1}{3}f_2\right)h \\
&\quad + \left(\frac{1}{3}f_2 + \frac{4}{3}f_3 + \frac{1}{3}f_4\right)h \\
&\quad + \cdots \\
&\quad + \left(\frac{1}{3}f_{n-2} + \frac{4}{3}f_{n-1} + \frac{1}{3}f_n\right)h \\
&= \left(\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{2}{3}f_2 + \frac{4}{3}f_3 + \frac{2}{3}f_4 + \cdots \frac{2}{3}f_{n-2} + \frac{4}{3}f_{n-1} + \frac{1}{3}f_n\right)h
\end{aligned}$$

2.2.2 Gaussian quadrature

In constructing the Simpson weights no thought went into choosing the location of the nodes. We simply started with equidistant nodes and calculated the formulas for the weights. Writing the code to implement Newton-Cotes is so easy, because the weights only depend on h and not on x_0. But by being smart about choosing the nodes we can do even better. That is we can get a more accurate answer with the same number of points. To be precise, if n is the number of nodes, then the following is true. With Newton-Cotes quadrature we get an exactly correct answer if the function we are integrating is a polynomial of order $n-1$, whereas with Gaussian quadrature we get an exactly correct answer if the function we are interested in is a polynomial of order $2n-1$. For example, if we have 5 nodes then we get an exact answer for all polynomials of order 9 (or less) and we get an accurate answer for functions that can be approximated well with a 9^{th}-order polynomial. Given that we can cover quite a few functions with 9^{th}-order polynomials, you better be impressed about the power and simplicity of Gaussian quadrature.

To understand the procedure, suppose we want to integrate a scalar function defined on $[-1,1]$ using the quadrature formula with n nodes. Thus,

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} \omega_i f(\zeta_i). \tag{2.5}$$

Note that we have $2n$ free parameters, namely the ω_is (the weights) and the ζ_is (the nodes). We want to get the correct answer for any polynomial of order $2n-1$. To accomplish this, we choose the values of ω_i and ζ_i so that by construction we get the correct answer for all the basis functions, that is, for 1, x, x^2, $\cdots$, and x^{2n-1}. But if we get the correct answer for all basis functions, we get the correct answer for any combination. That is, one gets the correct answer for any polynomial of order $2n-1$. To see why, suppose that we have found the ω_is and the ζ_is such that applying the formula in Equation 2.5 for $f(x)=x^4$ gives the right answer, that is

$$\int_{-1}^{1} x^4 dx = \sum_{i=1}^{n} \omega_i \zeta_i^4. \tag{2.6}$$

But this means that we also get the right answer for $f(x)=\alpha x^4$ for *any* value of α. To see why our approximation now would be

$$\sum_{i=1}^{n} \omega_i \alpha \zeta_i^4 = \alpha \sum_{i=1}^{n} \omega_i \zeta_i^4. \tag{2.7}$$

That is our approximation is the answer for $f(x)=x^4$ times α. Since the integral of $\alpha f(x)$ is indeed equal α times the integral of $f(x)$ we get the right answer. Similarly, we get the right answer for any combination of polynomial basis functions.

But we still have to find the ω_is and the ζ_is that give us the correct answer for the basis functions. That will be the case if the following is true:

$$\int_{-1}^{1} x^j dx = \sum_{i=1}^{n} \omega_i \zeta_i^j \quad j=0,1,\cdots,2n-1 \tag{2.8}$$

This is a system of $2n$ equations in $2n$ unknowns. The important thing to realize is that the solution to this system of equations does not depend on f. That is, independent of the particular function one is considering, one uses the same values for the ω_is and the ζ_is. In fact, there are standard subroutines available to solve for the quadrature nodes and weights.

Gauss-Legendre

The procedure discussed above that calculated the integral of a function over the interval $[-1, 1]$ is called Gauss-Legendre. So in practice one would do the following. One would use a numerical procedure to generate the ω_is and the ζ_is. The generated values will satisfy (2.8), but you don't have to worry about how the algorithm makes that happen. Let the solution be ω_i^{GL} and ζ_i^{GL}. The only thing that you have to do is to obtain function values at the indicated nodes and calculate the approximation to the integral using the quadrature formula, that is

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} \omega_i^{GL} f(\zeta_i^{GL}). \tag{2.9}$$

Gauss-Hermite

Gauss-Legendre will give an accurate answer if $f(x)$ can be approximated well with a polynomial. Now suppose that one wants to integrate a function $f(x)$ that can be written as $g(x) * W(x)$ and one knows that $g(x)$ can be approximated well with a polynomial but $g(x) * W(x)$ cannot. In this case, it would not be smart to use Gauss-Legendre. Instead one would want to adjust the procedure to take this into account. There are different Gaussian quadrature procedure that do exactly this for different weighting functions, $W(x)$, and different domains. An important one is Gauss-Hermite for which the weighting function is e^{-x^2} and the domain is the real line. For Gauss-Hermite the weights and the nodes are chosen to satisfy

$$\int_{-\infty}^{\infty} x^j e^{-x^2} dx = \sum_{i=1}^{n} \omega_i \zeta_i^j \quad j = 0, 1, \cdots, 2n-1 \tag{2.10}$$

Let the solution be ω_i^{GH} and ζ_i^{GH}. So the approximation would be given by

$$\int_{-\infty}^{\infty} g(x) e^{-x^2} dx \approx \sum_{i=1}^{n} \omega_i^{GH} g(\zeta_{gh,i}). \tag{2.11}$$

Make sure you understand why there is an "=" in Equation (2.10) and an "$\approx$" in Equation (2.11). In the first equation we are choosing the ω_is and the ζ_is so that our approximating formula, i.e. Equation (2.11) will give the correct answer for particular choices of $g(x)$, namely polynomial basis functions. But unless $g(x)$ is a polynomial, the quadrature formula is an approximation.

Gauss-Chebyshev

Another Gaussian quadrature procedure is Gauss-Chebyshev that deals with

$$\int_{-1}^{1} g(x) \frac{1}{(1-x^2)^{1/2}} dx$$

Quadrature nodes

The nodes that solve the problems discussed here turn out to be the zeros of the basis functions of the corresponding Orthogonal polynomial. That is, the Chebyshev nodes that solve

$$\int_{-1}^{1} x^j \frac{1}{(1-x^2)^{1/2}} dx = \sum_{i=1}^{n} \omega_i \zeta_i^j \quad j = 0, 1, \cdots, 2n-1$$

are exactly the same as the Chebyshev nodes discussed in the chapter on approximating functions, although it goes a bit to far to explain why.

2.2.3 Change in variable

Suppose one wants to calculate the expectation of $h(y)$, i.e., $\mathrm{E}[h(y)]$, where y is a random variable with a $N(\mu, \sigma^2)$ distribution. That is one wants to calculate

$$\int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} h(y) \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right) dy$$

Also, suppose that $h(y)$ is a function that one expects can be approximated well with a polynomial. This problem resembles a Gauss-Hermite quadrature problem but not exactly. One might be tempted to make it a Gauss-Hermite problem simply by defining

$$\overline{h}(y) = \frac{h(y)}{\sigma\sqrt{2\pi}} \frac{\exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)}{\exp\left(-y^2\right)}$$

and considering the identical integral

$$\int_{-\infty}^{\infty} \overline{h}(y) \exp\left(-y^2\right) dy.$$

But note that it was given that $h(y)$ could be approximated well with a polynomial, not that $\overline{h}(y)$ can be. In fact, given that $\exp(-y^2)$ is not like a polynomial, $\overline{h}(y)$ may be approximated very poorly with a polynomial.

So the appropriate way to go is to do a change of variables. This is very easy but don't forget the Jacobian. That is, if $y = \phi(x)$ then

$$\int_a^b g(y) dy = \int_{\phi^{-1}(a)}^{\phi^{-1}(b)} g(\phi(x)) \phi'(x) dx$$

The transformation we use here is

$$x = \frac{y-\mu}{\sigma\sqrt{2}} \text{ or } y = \sigma\sqrt{2}x + \mu$$

This gives

$$\begin{aligned}
\mathrm{E}\left[h(y)\right] &= \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} h(y) \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right) dy \\
&= \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} h(\sqrt{2}\sigma x + \mu) \exp\left(-x^2\right) \sigma\sqrt{2} dy \\
&= \int_{-\infty}^{\infty} \frac{1}{\sqrt{\pi}} h(\sqrt{2}\sigma x + \mu) \exp\left(-x^2\right) dy
\end{aligned}$$

What to do in practice?

So what would you do in practice if one wants to evaluate $\mathrm{E}[h(y)]$. First, one obtains n Gauss-Hermite quadrature weights and nodes using a numerical algorithm. Second, one gets an approximation using

$$\mathrm{E}\left[h(y)\right] \approx \sum_{i=1}^{n} \frac{1}{\sqrt{\pi}} \omega_i^{GH} h\left(\sqrt{2}\sigma\zeta_i^{GH} + \mu\right) \tag{2.12}$$

and do not forget to divide by $\sqrt{\pi}$! Well, how often do you get something in life so complex as an integral so easily?

2.3 Monte Carlo Integration

The idea behind Monte Carlo integration is very simple. Consider a random variable x with CDF $F(x)$. Then one can approximate the integral of the function $h(x)$ with

$$\int_a^b h(x) dF(x) \approx \frac{\sum_{t=1}^{T} h(x_t)}{T}, \tag{2.13}$$

where $\{x_t\}_{t=1}^{T}$ is a series drawn from a random number generator corresponding to the distribution of x. Although very simple there is one important disadvantage: it is not very accurate. Above we saw that we can get an accurate answer with just a few quadrature nodes for a large class of functions. Monte Carlo is subject to sampling variation and this only disappears at root n. Suppose we calculate the mean of a random variable with a uniform distribution on the unit interval. With $T = 100$ the standard error is 0.029 which is 5.8% of the true mean. Even with $T = 1,000$ we have a standard error that is 1.8% of the true mean.

If one doesn't have a CDF then one can use a uniform distribution. That is,

$$\int_a^b h(x)dx = (b-a)\int_a^b h(x)f^{ab}(x)dx, \tag{2.14}$$

where f^{ab} is the density of a random variable with a uniform distribution over $[a,b]$, that is, $f^{ab} = (b-a)^{-1}$. Thus, one could approximate the integral with

$$\int_a^b h(x)dx \approx (b-a)\frac{\sum_{t=1}^T h(x_t)}{T}, \tag{2.15}$$

where x_t is generated using a random number generator for a variable that is uniform on $[a,b]$.

Typically one doesn't have access to true random numbers and one only has access to a computer program that generates them. Therefore, these procedures are also referred to pseudo random numbers. The computer program generates data that are (if it is a good program) indistinguishable from a true series of random numbers. But the function that generates the series is deterministic (and chaotic) so that one should be careful in using theorems for true random numbers to think about things like rates of convergence.

2.4 Multivariate problems and quasi Monte Carlo integration

It is straightforward to extend the idea of quadrature techniques to higher dimensional problems. The number of nodes increases exponentially, however, which means that it becomes computationally quickly very expensive. With Monte Carlo integration one does not seem to have this problem. That is the mean of $h(x_t)$ and the mean of $h(x_t, z_t)$ both converge towards its mean at rate $\sqrt{T}$.

Think of a numerical integration problem as choosing nodes and then taking a (weighted) average. By extending the quadrature techniques derived for scalar functions to multivariate problems one doesn't fill in the space with nodes in the most efficient way.

Building on the idea of pseudo Monte Carlo new techniques have been developed that fill in the space better. The starting point of quasi Monte Carlo integration is to generate *equidistributed* series. A scalar sequence $\{x_t\}_{t=1}^T$ is equidistributed over $[a,b]$ iff

$$\lim_{T\longrightarrow\infty} \frac{b-a}{T}\sum_{t=1}^T f(x_t) = \int_a^b f(x)dx \tag{2.16}$$

for all Rieman-integrable $f(x)$. Note the similarity with Equation (2.15). There are several examples of equidistributed series. For example the sequence $(\xi, 2\xi, 3\xi, 4\xi, \cdots)$ is equidistributed modulo 1 for any irrational number ξ.[1] Another example is the sequence of prime numbers multiplied by an irrational number $(2\xi, 3\xi, 5\xi, 7\xi, \cdots)$.

For a d-dimensional problem, an equidistributed sequence $\{x_t\}_{t=1}^{T} \subset D \subset R^d$ satisfies

$$\lim_{T \longrightarrow \infty} \frac{\mu(D)}{T} \sum_{t=1}^{T} f(x_t) = \int_D f(x)dx, \tag{2.17}$$

where $\mu(D)$ is the Lebesque measure of D.[2]

Some examples for equidistributed vectors on the d-dimensional unit hypercube are the following.

Weyl:

$$x_t = (t\sqrt{p_1}, t\sqrt{p_2}, \cdots, t\sqrt{p_d}) \text{ modulo } 1, \tag{2.18}$$

where p_i is the i^{th} positive prime number.

Neiderreiter:

$$x_t = (t2^{1/(d+1)}, 2^{2/(d+1)}, \cdots, t2^{d/(d+1)}) \text{ modulo } 1$$

Equidistributed vectors for other hypercubes can be done using linear transformations.

---

[1] $Frac(x)$ (or x Modulo 1) means that we subtract the largest integer that is less than x. For example, $frac(3.564) = 0.564$.

[2] To see why you have to multiply the sum with $\mu(D)$ just consider the case when $f(x) = 1\ \forall x$. Then we know the integral should be equal to $\mu(D)$ which we get because $\sum_{t=1}^{T} f(x)/T = 1$.