BYU-MCL Boot Camp
Numerical Integration
Professor Richard W. Evans

1 Introduction

Integrals of the form $\int_a^b g(x)dx$ arise often in economic models. One example is the aggregating of consumption amounts of a continuum of differentiated goods $c_t(i)$,

$$C_t = \left(\int_0^1 \alpha_i^{\frac{1}{\varepsilon}} c_t(i)^{\frac{\varepsilon-1}{\varepsilon}} di\right)^{\frac{\varepsilon}{\varepsilon-1}} \tag{1}$$

where C_t is aggregate consumption, α_i is a weight on the particular amount of consumption of good i, ε is the constant elasticity of substitution between different goods i, and the measure of goods is normalized to be between 0 and 1, without loss of generality.[1] Another key example of an integral that often occurs in macroeconomics is the expectations operator on the right-hand-side of the standard intertermporal Euler equation,

$$\begin{aligned} & u'(c_t)=\beta E_{z_{t+1}|z_t}\Big[(1+r_{t+1}-\delta)u'(c_{t+1})\Big] \\ \Rightarrow \quad & u'(c_t)=\beta\int_a^b \Big(1+r_{t+1}(z_{t+1})-\delta\Big)u'\Big(c_{t+1}(z_{t+1})\Big)f(z_{t+1}|z_t)dz_{t+1} \end{aligned} \tag{2}$$

where a and b are the bounds of the support of z_{t+1} and $f(z_{t+1}|z_t)$ is the pdf of z_{t+1} that could potentially be conditional on z_t.

It is a rare convenience when these integrals can be evaluated analytically. However, it does not take much richness in functional form to render analytical solutions impossible for many integrals in economic models. In these cases, the integral must be computed numerically. The following discussion of numerical integration draws from the great treatments of the subject in Heer and Maussner (2009, pp. 598-603), Judd (1998, Ch. 7), and Adda and Cooper (2003, pp.55-60).

2 Newton-Cotes Quadrature

Newton-Cotes quadrature forumalas approximate the integral of a function $\int_a^b g(x)dx$ by evaluating the function at N nodes $\{x_1, x_2, ...x_N\}$ and weighting those nodes with N weights $\{\omega_1, \omega_2, ...\omega_N\}$. The general form of Newton-Cotes quadrature forumulas is

$$\int_a^b g(x)dx \approx \sum_{n=1}^N \omega_n g(x_n) \tag{3}$$

[1]The consumption aggregator in (1) is often called the Armington aggregator as it was first proposed in Armington (1969). It is also known as a Dixit-Stiglitz aggregator after its use in Dixit and Stiglitz (1977).

2.1 Midpoint rule (1 node)

The midpoint rule is the simplest Newton-Cotes formula and uses only one node or evaluation of the function. The midpoint formula simply evaluates the function at the midpoint of the domain of $x = \frac{a+b}{2}$ and assumes that the function is a constant at that level over the entire domain of $x \in [a, b]$.

$$\int_a^b g(x)dx \approx (b-a)g\left(\frac{a+b}{2}\right) \tag{4}$$

A more sophisticated midpoint rule is the composite midpoint rule, which breaks up the domain of the function $g(x)$ into N intervals and applies the midpoint rule to each interval.

2.2 Trapezoid rule (2 nodes)

The trapezoid rule is the area under a line that connects the function $g(x)$ at the two endpoints a and b.

$$\int_a^b g(x)dx \approx \frac{b-a}{2}\left[g(a) + g(b)\right] \tag{5}$$

A more sophisticated trapezoid rule is the composite trapezoid rule, which breaks up the domain of the function $g(x)$ into N intervals and applies the trapezoid rule to each interval.

2.3 Simpson's rule (3 nodes)

Simpson's rule offers a smooth nonlinear (quadratic) alternative the linear approximations of the midpoint and trapezoid rules. Simpson's rule finds a quadratic function in x that passes through the end points and the midpoint of the function $g(a)$, $g\left(\frac{a+b}{2}\right)$, and $g(b)$, which produces the following weights and values.

$$\int_a^b g(x)dx \approx \frac{b-a}{6}\left[g(a) + 4g\left(\frac{a+b}{2}\right) + g(b)\right] \tag{6}$$

Again, a more sophisticated Simpson's rule is the composite Simpson's rule, which breaks up the domain of the function $g(x)$ into N intervals and applies the Simpson's rule to each interval.

Exercise 1. You can verify that the analytical solution to the integral of the function

$$g(x) = 0.1x^4 - 1.5x^3 + 0.53x^2 + 2x + 1$$

between $x = -10$ and $x = 10$ is $\int_{-10}^{10} g(x)dx = 4,373.3\bar{3}$. Write a Python function that will take as arguments an anonymous function that the user specifies representing $g(x)$, integration bounds a and b, and `method = {'midpoint','trapezoid','Simpsons'}`. For this exercise, just do the simple methods (nodes equal 1, 2, and 3), not the composit methods. Evaluate the numerical approximations of the integral $\int_a^b g(x)dx$ using all three Newton-Cotes methods in your function and compare the difference between the values of these integrals to the true analytical value of the integral.

Exercise 2. Write a Python function that makes a Newton-Cotes discrete approximation of the distribution of the normally distributed variable $Z \sim N(\mu, \sigma)$. Let this function take as arguments the mean μ, the standard deviation σ, the number of equally spaced nodes N to estimate the distribution, and the number of standard deviations k away from μ to make the furthest nodes on either side of μ. Have this function return a vector of nodes of $[Z_1, Z_2, ...Z_N]$ and a vector of weights $[\omega_1, \omega_2, ...\omega_N]$ such that ω_i is given by the integral under the normal distribution between the midpoints of the two closest nodes. Define $f(Z; \mu, \sigma)$ as the pdf of the normal distribution and $F(Z; \mu, \sigma)$ as the cdf.

$$\omega_i = \begin{cases} F\left(\frac{Z_1+Z_2}{2}; \mu, \sigma\right) & \text{if} \quad i = 1 \\ \int_{Z_{min}}^{Z_{max}} f(Z; \mu, \sigma) dZ & \text{if} \quad 1 < i < N \\ 1 - F\left(\frac{Z_{N-1}+Z_N}{2}; \mu, \sigma\right) & \text{if} \quad i = N \end{cases}$$

$$\text{where} \quad Z_{min} = \frac{Z_{i-1} + Z_i}{2} \quad \text{and} \quad Z_{max} = \frac{Z_i + Z_{i+1}}{2}$$

What are the weights and nodes $\{\omega_n, Z_n\}_{n=1}^N$ for $N = 11$?

Exercise 3. If $Z \sim N(\mu, \sigma)$, then $A \equiv e^Z \sim LN(\mu, \sigma)$ is distributed lognormally and $\log(A) \sim N(\mu, \sigma)$. Use your knowledge that $A \equiv e^Z$, $\log(A) \sim N(\mu, \sigma)$, and your function from Exercise 2 to write a function that gives a discrete approximation to the lognormal distribution. Note: You will not end up with evenly spaced nodes $[A_1, A_2, ...A_N]$, but your weights should be the same as in Exercise 2.

Exercise 4. Let Y_i represent the income of individual i in the United States for all individuals i. Assume that income Y_i is lognormally distributed in the U.S. according to $Y_i \sim LN(\mu, \sigma)$, where the mean of log income is $\mu = 10.5$ and the standard deviation of log income is $\sigma = 0.8$. Use your function from Exercise 3 to compute an approximation of the expected value of income or average income in the U.S. How does your approximation compare to the exact expected value of $E[Y] = e^{\mu + \frac{\sigma^2}{2}}$?

3 Gaussian Quadrature

Gaussian quadrature formulas for approximating an integral take the Newton-Cotes approximation form $\int_a^b g(x)dx \approx \sum_{n=1}^N \omega_n g(x_n)$ and optimally choose the weights ω_n and nodes x_n given the total number of nodes N and some approximating polynomial class $h_i(x)$. The N weights and nodes are chosen to make an *exact integration* relationship hold. That is, the N weights and nodes must hold exactly for polynomials of order $2N - 1$ in the following way,

$$\int_a^b h_i(x)dx = \sum_{n=1}^N \omega_n h_i(x_n) \quad \text{for} \quad i = 0, 1, ...2N - 1 \tag{7}$$

where $h_i(x)$ is an i-order polynomial in x.

As a simple example, suppose we want to approximate an arbitrary function $g(x)$ with Gaussian quadrature using a simple class of polynomials $h_i(x) = x^i$ and only $N = 2$ weights and nodes. (7) implies a system of four equations used to determine the four variables $(\omega_1, \omega_2, x_1, x_2)$ to approximate the integral $\int_a^b g(x)dx \approx \sum_{n=1}^N \omega_n g(x_n)$.

$$\begin{aligned}
\int_a^b k dx &= \omega_1 k + \omega_2 k \quad \text{where} \quad h_0(x) = k \\
\int_a^b x dx &= \omega_1 x_1 + \omega_2 x_2 \\
\int_a^b x^2 dx &= \omega_1 x_1^2 + \omega_2 x_2^2 \\
\int_a^b x^3 dx &= \omega_1 x_1^3 + \omega_2 x_2^3
\end{aligned} \tag{8}$$

For $N = 2$, the optimal weights and nodes that solve the system (8) are $(\omega_1, \omega_2, x_1, x_2) = (1, 1, -0.578, 0.578)$. The Python code to solve this nonlinear system could be a simple root finder such as `scipy.optimize.root` or one of the constrained minimizers in `scipy.optimize.minimize`. In general, the spacing of the nodes will not be uniform.

The accuracy of the Gaussian quadrature approximation of the integral $\int_a^b g(x)dx$ increases in the number of nodes N. The accuracy of the approximation of the integral can also be improved by the choice of polynomial family $h_i(x)$. In particular, the families of orthonormal polynomials have multiple desirable properties. Because of the orthogonality of their coefficients, the system (8) is easier to solve due to the lack of collinearity.Also, the weights ω_n turn out to be the zeros of the orthogonal polynomial family. Lastly, these orthogonal families of polynomials can give very accurate solutions to integrals of the form $\int_a^b w(x)g(x)dx$, where $w(x)$ is the weighting function of an orthonormal family of polynomials.

For a more detailed discussion of the theory behind Gaussian quadrature, see Judd (1998, pp. 257-265) and Heer and Maussner (2009, 599-601). The general applicability of Gaussian quadrature and its accuracy and efficiency advantage over Newton-Cotes formulas is summarized by Judd (1998, p.265).

> "Even when the asymptotic rate of convergence fore Gaussian quadrature is no better than the comparable Newton-Cotes formula, experience shows that Gaussian formulas often outperform the alternative Newton-Cotes formula [in terms of accuracy]."

Exercise 5. Approximate the integral of the function in Exercise 1 using Gaussian quadrature with $N = 3$, $(\omega_1, \omega_2, \omega_3, x_1, x_2, x_3)$. How does the accuracy of your approximated integral compare to the approximations from Exercise 1 and the true known value of the integral?

Exercise 6. Use the Python Gaussian quadrature command `scipy.integrate.quad` to numerically approximate the integral from Exercise 1.

$$\int_{-10}^{10} g(x)dx \quad \text{where} \quad g(x) = 0.1x^4 - 1.5x^3 + 0.53x^2 + 2x + 1$$

How does the approximated integral from the `scipy.integrate.quad` command compare to the exact value of the function?

4 Monte Carlo Integration

In the Newton-Cotes quadrature methods of approximating an integral nodes and weights for the approximating function $\sum_{n=1}^{N}\omega_n g(x_n)$ are chosen without much attention to the effect of the placement of these nodes or the levels of the weights on the accuracy of the approximation. Newton-Cotes methods are computationally fast, but lack in accuracy. Gaussian quadrature methods spend more computational time choosing "optimal" weights and nodes, which gives an accuracy payoff over Newton-Cotes formulas. Monte Carlo integration methods use the computationally fast method of drawing uniformly from the support of the variable of integration and depends on a lot of draws to get the accuracy close.[2] Although Monte Carlo integration methods do not converge as quickly as Gaussian quadrature methods for functions of one variable, they are especially valuable in integrating over functions of multiple variables.

Let $\mathbf{x} \in \Omega \subset \mathbb{R}^m$ be a vector of length m of variables with domain Ω which is a subset of $\mathbb{R}^m$. We are interested in approximating following integral,

$$\int_{\Omega} g(\mathbf{x})d\mathbf{x} \approx V\frac{1}{N}\sum_{n=1}^{N} g\left(\mathbf{x}_n\right) \quad \text{where} \quad V = \int_{\Omega} d\mathbf{x} \tag{9}$$

Equation (9) essentially says you can approximate the integral of a function $g(\mathbf{x})$ on domain Ω by taking the average of the evaluations of the function g at N random draws of the vector $\mathbf{x}_n$ multiplied by the volume of the domain.

An easy example of a univariate integral is $\int_0^1 x\,dx$ (where $g(\mathbf{x}) = x$). The Monte Carlo approximation formula for this integral is the following.

$$\int_0^1 x\,dx \approx V\frac{1}{N}\sum_{n=1}^{N} x_n = \frac{1}{N}\sum_{n=1}^{N} x_n \tag{10}$$

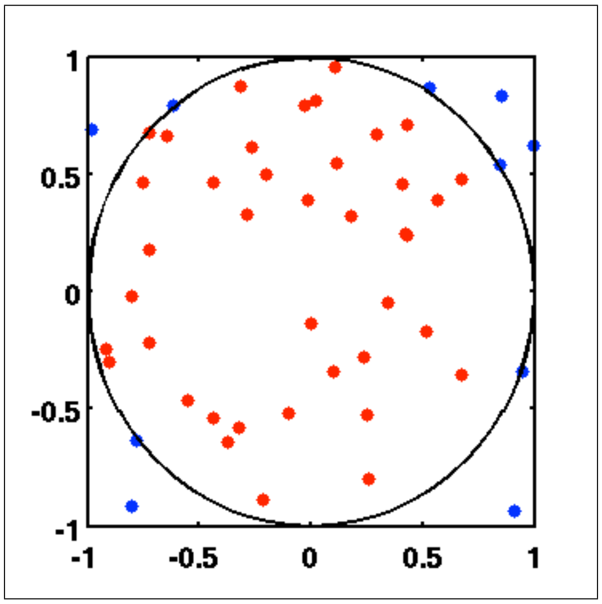
It is easy to see that the answer to the exact integral on the left-hand-side of (10) is 1/2. In the approximation on the right-hand-side of (10), V is the volume of the

[2] Judd (1998, pp. 309-311) spends significant time explaining that these methods are more correctly called "pseudo-Monte Carlo methods" because they make use of pseudo random number generators, the use of which cannot invoke the law of large numbers or the central limit theorem. He advocates the use of equidistributed sequences rather than pseudo random number generators from the uniform distribution to create "quasi-Monte Carlo integral estimation". However, the biases introduced by pseudo random number generators are rarely significant in practice.

domain of $x \in [0,1]$, which is 1. It is straightforward to see that the average of N draws from a uniform distribution between 0 and 1 will converge quickly to 1/2.

Exercise 7 lets you try your hand at coding a classic Monte Carlo integration approximation of the integral of a function of two variables to approximate the value of π. The area of a circle with radius $r = 1$ is π. A way to visualize the Monte Carlo approximation of the area of that circle, or π, is to enclose the circle in a square with sides of length 2, in which the x-axis goes from -1 to 1 and the y-axis goes from -1 to 1. The points in Figure 1 are the uniformly distributed random draws from $(x, y) \in [-1, 1] \times [-1, 1]$. Intuitively, the area of the circle is the fraction of the dots (red dots divided by total dots) that are inside the circle or on the boundary of the circle, multiplied by the area or volume of the square in which the circle lies.

Figure 1: Monte Carlo integral estimation of area of unit radius circle



Following the intuition of the previous paragraph and of Figure 1, the exact area of the circle can be written as an integral of the indicator function of coordinate variables x and y in the following way.

$$\begin{aligned} &\text{Area} = \int_\Omega g(x,y)dx\,dy = \pi \\ &\qquad \text{where} \quad g(x,y) = \begin{cases} 1 & \text{if} \quad x^2+y^2 \leq 1 \\ 0 & \text{else} \end{cases} \qquad \text{and} \quad \Omega = [-1,1]\times[-1,1] \end{aligned} \tag{11}$$

The exact integral (11) for the area of a unit radius circle can be Monte Carlo approximated using the form (9) resulting in the following function.

$$\int_\Omega g(x,y)dx\,dy \approx 4\frac{1}{N}\sum_{n=1}^{N} g\left(x_n, y_n\right) \tag{12}$$

Exercise 7. Use Monte Carlo integration to approximate the value of π. Define a function in Python that takes as arguments an anonymous function $g(\mathbf{x})$ of a vector of variables $\mathbf{x}$, the domain Ω of $\mathbf{x}$, and the number of random draws N and returns the Monte Carlo approximation of the integral $\int_\Omega g(\mathbf{x})d\mathbf{x}$. In order to approximate π, let the functional form of the anonymous function be $g(x,y)$ from (11) with domain $\Omega = [-1,1]\times[-1,1]$. What is the smallest number of random draws N from Ω that matches the true value of π to the 10th decimal 3.1415926535? [Your answers may differ slightly depending on your draws of (x_n,y_n) from the bivariate uniform distribution.]

5 Discrete Markov Approximation of Continuous AR(1) Process

Suppose you have a random shock z_t in your model that has some persistence according to the following AR(1) process.

$$z_{t+1} = \rho z_t + (1-\rho)\mu + \varepsilon_{t+1} \quad \text{where} \quad \varepsilon_t \sim N(0,\sigma) \quad \text{and} \quad \rho\in[0,1) \tag{13}$$

The expected value of z_{t+1} is conditional on the current realization of the shock $E[z_{t+1}|z_t] = \rho z_t + (1-\rho)\mu$ but the variance of z_{t+1} is unconditional $Var[z_{t+1}] = \sigma^2$. The AR(1) process in (13) generates a variable that fluctuates around its mean μ, and the expected value of the variable tomorrow $E[z_{t+1}|z_t]$ is some convex combination of the variable today z_t and the mean μ.

Typical examples of these types of shocks in economics are shocks to ability, health status, and productivity shocks—all of which exhibit persistence or dependence on recent values. If the shock must be strictly positive, as is the case with productivity shocks, the variable z_t is simply exponentiated.

$$Y_t = A_t K_t^\alpha L_t^{1-\alpha} \quad \text{where} \quad A_t = e^{z_t} \tag{14}$$

Notice that the variable A_t is lognormally distributed $A_t \sim LN\big(\rho z_{t-1} + (1-\rho)\mu, \sigma\big)$ because $\log(A_t) = z_t$ and $z_t \sim N\big(\rho z_{t-1} + (1-\rho)\mu,\sigma\big)$. You made a discretized approximation of the i.i.d. (no persistence) version of this distribution in Exercise 3 and estimated average income in the U.S. using it in Exercise 4.

Tauchen and Hussey (1991) describe a quadrature-based method for producing efficient nodes and probabilities of a discrete first-order Markov process to approximate a continuous AR(1) random variable.[3] A classic example of where this discretization is extremely valuable is in the stochastic intertemporal Euler equation (2) from Section 1.

$$\begin{aligned} u'(c_t) &= \beta E_{z_{t+1}|z_t}\Big[(1+r_{t+1}-\delta)u'(c_{t+1})\Big] \\ \Rightarrow \quad u'(c_t) &= \beta\int_a^b \Big(1 + r_{t+1}(z_{t+1}) - \delta\Big)u'\Big(c_{t+1}(z_{t+1})\Big)f(z_{t+1}|z_t)dz_{t+1} \end{aligned} \tag{2}$$

[3] Tauchen (1986) details a simpler non-quadrature based method for producing efficient nodes and probabilities of a discrete first-order Markov process to approximate a continuous AR(1) random variable.

The expectation on the right-hand-side of (2) is over z_{t+1} given z_t, where z_{t+1} is the AR(1) process described in (13). One of the most common nonlinear solution techniques for the functional equations of the model characterized by (2) is value function iteration on the following recursive Bellman equation.

$$V(k,z) = \max_{k'} \; u(k,z,k') + \beta E_{z'|z}\big[V(k',z')\big] \tag{15}$$

The expectation on the right-hand-side of the Bellman equation (15) is simply an integral of the form $E_{z'|z}\big[V(k',z')\big] = \int_{z'} V(k',z')f(z'|z)dz'$. However, it is difficult to use standard Gaussian quadrature or Monte Carlo integration methods because the value function $V(k',z')$ is often only known at a few points.

One solution to this problem is to interpolate or fit some continuous function $\tilde{V}(k',z')$ to the known points of $V(k',z')$ and then use Gaussian quadrature or Monte Carlo integration to approximate the integral $\int_{z'} \tilde{V}(k',z')f(z'|z)dz'$. Heer and Maussner (2008) and Heer and Maussner (2009, p. 237) find that the errors in the extrapolated values of the interpolated function $\tilde{V}$ beyond the bounds of the known points of V cause the solution to be less accurate than using the Tauchen-Hussey method of approximating $f(z'|z)$ with a discrete first order Markov process.

References

Adda, Jérôme and Russell Cooper, *Dynamic Economics: Quantitative Methods and Applications*, MIT Press, 2003.

Armington, Paul S., "A Theory of Demand for Products Distinguished by Place of Production," *IMF Staff Papers*, March 1969, *16* (1).

Dixit, Avinash K. and Joseph E. Stiglitz, "Monopolistic Competition and Optimum Product Diversity," *American Economic Review*, June 1977, *67* (3), 297–308.

Heer, Burkhard and Alfred Maussner, "Computation of Business Cycle Models: A Comparison of Numerical Methods," *Macroeconomic Dynamics*, November 2008, *12* (5), 641–663.

_ **and** _ , *Dynamic General Equilibrium Modeling: Computational Methods and Applications*, second ed., Springer, 2009.

Judd, Kenneth L., *Numerical Methods in Economics*, MIT Press, 1998.

Tauchen, George, "Finite State Markov-chain Approximation to Univariate and Vector Autoregressions," *Economics Letters*, 1986, *20* (2), 177–181.

_ **and Robert Hussey**, "Quadrature-based Methods for Obtaining Approximate Solutions to Nonlinear Asset Pricing Models," *Econometrica*, March 1991, *59* (2), 371–396.