Numerical Integration and Simulation

Tyler Ransom
Duke University

August 1, 2012

Plan

- Discuss numerical integration
- Examples of when you would need to know how to numerically integrate
- Numerical integration in Matlab
- Monte Carlo
- Simulation
- Relationship between Monte Carlo, simulation and integration

Numerical Integration

- Generally speaking, numerical integration refers to a way of evaluating integrals that don't have a closed form solution
- Simplest example: CDF of the normal distribution
- Requires integration of the normal pdf, which doesn't have a closed-form antiderivative

Integration in Economics

In structural applied micro, you'll find a few situations in which you will need to use integration to recover correct parameter estimates:

- Unobserved Heterogeneity in a non-linear model
- Dynamic models where forward-looking agents aren't able to see future shocks
- Games in IO where opponents' epsilons are unknown to players
- Random utility models where epsilon is assumed to be Normal
- Basically, whenever you need to take an expectation over something unknown (either to the agent or the econometrician), you'll need to integrate

You've already done numerical integration

You may not have recognized so, but you have already estimated models with numerical integration (Matlab just did it for you)

- Simple probit model: P_{i1} is a normal cdf (the integral of a normal pdf)
- Logit models: P_{i1} is technically an integral that has a closed form, thanks to the Gumbel distributional assumption (another reason why logit is so popular)

Example: Random effects in a nonlinear model

Recall the simple binomial probit model:

$$\begin{aligned} y_{it} &= X_{it}\beta + \varepsilon_{it} \text{ with} \\ y_{it} &= 1 \text{ if } X_{it}\beta + \varepsilon_{it} > 0 \\ y_{it} &= 0 \text{ else} \\ \varepsilon_{it} &\sim N(0,1) \end{aligned}$$

The probabilities for our likelihood are then

$$\begin{aligned} \Pr(y_{it} = 1 \mid X_{it}) &= \Phi(X_{it}\beta) \\ \Pr(y_{it} = 0 \mid X_{it}) &= 1 - \Phi(X_{it}\beta) \end{aligned}$$

where $\Phi(\cdot)$ is the CDF of the standard normal

Example: Random effects in a nonlinear model

Now let's add normally-distributed random effects to the model. i.e. α_i is mean-zero normal with variance σ_α^2, and ε_{it} is iid standard normal:

$$\begin{aligned}
\Pr\left(y_{it} = 1 \,|\, X_{it}, \alpha_i\right) &= \Phi\left(X_{it}\beta + \alpha_i\right) \\
\Pr\left(y_{it} = 0 \,|\, X_{it}, \alpha_i\right) &= 1 - \Phi\left(X_{it}\beta + \alpha_i\right)
\end{aligned}$$

If we could observe α_i, the log likelihood function would look like the regular probit log likelihood with the α_i term added in:

$$\ell(X_{it}; \beta, \alpha_i) = \sum_i \sum_t \ln\left(\Phi\left(X_{it}\beta + \alpha_i\right)^{y_{it}}\right) + \ln\left(\left(1 - \Phi\left(X_{it}\beta + \alpha_i\right)\right)^{1-y_{it}}\right)$$

Example: Random effects in a nonlinear model

However, because we can't observe α_i, we need to integrate it out of the likelihood, so the log likelihood function looks like:

$$\ell(X_{it};\beta,\alpha_i) = \sum_i \ln\left(\int_{-\infty}^{\infty}\prod_t \Phi\left(X_{it}\beta+\alpha_i\right)^{y_{it}}\left(1-\Phi\left(X_{it}\beta+\alpha_i\right)\right)^{1-y_{it}} f\left(\alpha_i\right) d\alpha_i\right)$$

where now we will estimate the parameters β and σ_α^2

- Luckily, because of the assumption that $\varepsilon_{it} \mid \alpha_i$ is iid standard normal, the integral is only one-dimensional and much more manageable to compute

Numerical integration methods

- “Quadrature” and “numerical integration” can be used interchangeably
- Examples of quadrature that you probably already know:
  - Rectangle rule
  - Trapezoidal rule
  - Simpson’s rule
- These are very simple, but may not be most computationally efficient or accurate
- There are many fancier methods that also work well when the integral is multi-dimensional

Quadrature in Matlab

Matlab has a few built-in routines to approximate $\int_a^b f(x)dx$

- 1-dimensional integrals: `quad`, `quadl`, `quadgk`
- 2-dimensional integrals: `dblquad`, `quad2d`
- 3-dimensional integrals: `triplequad`

Note: Matlab vectorizes these over the unknown (x), so f can't take data as inputs

- Hence, these are not very useful when integrating a likelihood function because you want to evaluate something like the random effects equation two slides ago [which would require looping over values]
- `quadv` is a vectorized version of quad that would help with this
- When integrating a likelihood function, I usually write my own Simpson's rule code

Quadrature in Matlab

The MathWorks file exchange has a great function to approximate $\int_a^b f(x)dx$

- `lgwt` computes Gauss-Legendre quadrature weights on a grid of points (i.e. $b - a$ divided by some step h)
- These weights don't depend on the function f
- To compute the integral, first generate, e.g. `f=normpdf(x,0,1);` for each grid point, then simply type `int=sum(f.*w);` where `w` is the vector of Gauss-Legendre quadrature weights

Note: `lgwt` works like Matlab's `quad` functions (i.e. f can't take data as inputs).

- Need to use `bsxfun` or `arrayfun` to vectorize it for use in a likelihood function
- `lgwt` can be used for multidimensional integrals (e.g. $\int_a^b \int_c^d f(x,y)dydx$) as well (see later):

Syntax for Matlab's quadrature functions

The general syntax for these is

```
integral_approximation = quad(@(x) f(x),a,b,tol);
```

where $f(x)$ is a function handle for the function of interest, a is the lower bound of integration, and b is the upper bound of integration. Note that a and b need not be finite. *Tol* is the numerical tolerance (default is 10^{-6}).

Syntax for `lgwt`

The general syntax for this is

```
[x, w] = lgwt(N,a,b); %get weights and grid points
     f = normpdf(x,0,sigma); %evaluate f(x) where f is normal pdf
integral_approximation = sum(f.*w);
```

where N is the desired number of grid points and everything else is the same as before. Multidimensional example:

```
[nx, wx] = lgwt(N,a,b); %get weights and grid points for 1st dimension
[ny, wy] = lgwt(N,a,b); %get weights and grid points for 2nd dimension
[nx, ny] = ndgrid(nx, ny); %form two matrices of grid points
[wx, wy] = ndgrid(wx, wy); %form two matrices of quadrature weights
int = sum(sum(wx .* wy .* f(nx) .* f(ny)));
```

`lgwt` vs. `dblquad` example

Let's compute $\int_{\pi}^{2\pi} \int_{0}^{\pi} [y \sin(x) + x \cos(y)]\, dydx$ using two different methods:

```
Q = dblquad(@(x,y)y*sin(x)+x*cos(y),pi,2*pi,0,pi);
```

The answer given by Matlab is -9.8696. Now let's compare it with `lgwt`:

```
[nx, wx] = lgwt(50,pi,2*pi);
[ny, wy] = lgwt(50,0,pi);
[nx, ny] = ndgrid(nx, ny);
[wx, wy] = ndgrid(wx, wy);
int = sum(sum(wx .* wy .* (ny.*sin(nx)+nx.*cos(ny))));
```

The answer is the same to within `3e-8` of `dblquad`

Computing the binomial probit random effects estimator

Our log likelihood function looks like:

$$\ell(X_{it};\beta,\alpha_i) = \sum_i \ln\left(\int_{-\infty}^{\infty}\prod_t \Phi\left(X_{it}\beta+\alpha_i\right)^{y_{it}}\left(1-\Phi\left(X_{it}\beta+\alpha_i\right)\right)^{1-y_{it}} f\left(\alpha_i\right)d\alpha_i\right)$$

How do we evaluate this integral?

- Could use one of Matlab's built-in quadrature routines and loop over the i's
- Could use `lgwt` with `arrayfun` to generate the grid and weights for each person, then add everything up in the likelihood
- Could also discretize α_i and use Simpson's rule on a grid. The approximate integral is then a weighted sum over these grid values.
- Could compute the integral by simulation (see last part of slides)

Simulation

In economics, “simulation” can mean a lot of different things:

1. simulate comparative statics in a theoretical model with no closed-form solution
2. Monte Carlo simulation to test the asymptotic properties of an econometric estimator
3. counterfactual simulation of a structural model to show the effects of a hypothetical policy change
4. simulation of a numerical integral than doesn’t have a closed-form solution

We’ll focus on numbers 2 and 4 today

Monte Carlo: background

- Monte Carlo is a phrase coined by John von Neumann while working on the Manhattan Project. (Monte Carlo is the name of the casino in Monaco where the uncle of von Neumann's co-worker, Stanisław Ulam, would borrow money to gamble) It was a code name for Ulam and von Neumann's early use of what we now call the Monte Carlo method
- The method in general refers to the repetition of random processes to approximate the probability of certain outcomes
- Example: What are the odds of being dealt a flush in 5-card poker? Could use combinatorics to figure it out, or you could shuffle the deck, deal the cards and repeat this process (many times!), counting the number of times a flush happens. This is what is known as the "Monte Carlo method."

Monte Carlos in theoretical econometrics

- In theoretical econometrics, Monte Carlo simulation is used to assess the asymptotic properties of estimators when the data generating process (DGP) is known
- The DGP is known because the researcher created it
- The researcher can then compare how the estimator performs when the sample size increases. She can also repeat the estimation many times with the same n to see what the distribution of estimates looks like. This informs her whether of the degree to which the estimation procedure is biased and/or inefficient

Monte Carlos in applied micro

In applied micro, Monte Carlo simulations can be useful in a variety of ways:

- Researcher can create simulated ("fake") data on which to test her code
- This is a good way to check for bugs in the code
- It's also a good way to figure out the properties of a novel estimation algorithm
- In this way, the Monte Carlo acts like a laboratory
- If a researcher is implementing a new estimation algorithm, Monte Carlo results are usually required for it to get published (i.e. proof that the algorithm actually works)

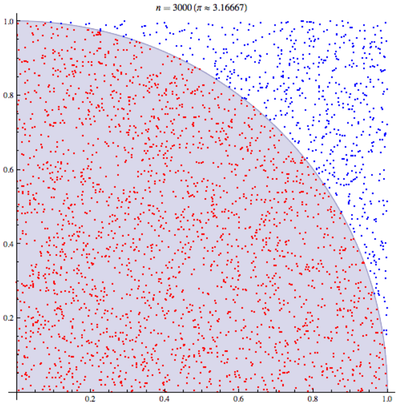
Approximating an integral with simulation

How can we use simulation (or the Monte Carlo method) to approximate an integral? Let's look at an example (from Wikipedia)

- Suppose we want to find the area under the unit circle from $x = 0$ to $x = 1$
- The function being integrated is $f(x) = \sqrt{1 - x^2}$
- We can approximate the area under the curve by generating thousands of random points in the area $(0, 1) \times (0, 1)$
- Then we can count the proportion of points below the curve $f(x)$. This is an estimate of the integral.

Approximating an integral with simulation

Visually, we can see how well the integral performs under different numbers of points:



General steps for simulating an integral

1. Calculate function bounds over the bounds of integration (e.g. $\overline{y} = \max_x \; : \; \underline{x} \leq x \leq \overline{x}$) where $\overline{x}$ is the upper bound of integration)
2. Set up an $N \times 2$ grid matrix of random draws: $\left[U(\underline{x}, \overline{x}) \;\; U(\underline{y}, \overline{y}) \right]$
3. Calculate the area of the grid: $(\overline{x} - \underline{x}) \times \left(\overline{y} - \underline{y}\right)$
4. Create an indicator for whether or not a point is below $f(x)$ and above 0
5. Create an indicator for whether or not a point is above $f(x)$ and below 0
6. integral_estimate $= \left(\text{area} * \left(\sum_i \text{below} - \sum_i \text{above}\right)\right) / N$

Simulated MLE (Greene, pp. 593, 799)

Recall the ugly expression for the log likelihood of the simple probit with random effects:

$$\ell(X_{it};\beta,\alpha_i) = \sum_i \ln\left(\int_{-\infty}^{\infty}\prod_t \Phi\left(X_{it}\beta+\alpha_i\right)^{y_{it}}\left(1-\Phi\left(X_{it}\beta+\alpha_i\right)\right)^{1-y_{it}} f\left(\alpha_i\right)d\alpha_i\right)$$

Another way to compute this integral is to notice that

$$\begin{aligned}
&\int_{-\infty}^{\infty}\prod_t \Phi\left(X_{it}\beta+\alpha_i\right)^{y_{it}}\left(1-\Phi\left(X_{it}\beta+\alpha_i\right)\right)^{1-y_{it}} f\left(\alpha_i\right)d\alpha_i \\
= &\int_{-\infty}^{\infty} g\left(\alpha_i\right) f\left(\alpha_i\right)d\alpha_i
\end{aligned}$$

is an expectation over α_i, so can be re-written as

$$\mathbb{E}_{\alpha_i}\left[\prod_t \Phi\left(X_{it}\beta+\alpha_i\right)^{y_{it}}\left(1-\Phi\left(X_{it}\beta+\alpha_i\right)\right)^{1-y_{it}}\right] = \mathbb{E}_{\alpha_i}\left[g\left(\alpha_i\right)\right]$$

Simulated MLE

If this expectation exists, we can think of it in a different way:

$$\frac{1}{R}\sum_{r=1}^{R} g\left(\alpha_{ir}\right) \xrightarrow{p} \mathbb{E}_{\alpha_i}\left[g\left(\alpha_i\right)\right]$$

where the researcher takes R draws from a random number generator for each person. We can plug this in to our original likelihood to get

$$\ell(X_{it}; \beta, \alpha_{ir}) = \sum_i \ln\left(\frac{1}{R}\sum_{r=1}^{R}\left[\prod_t \Phi\left(X_{it}\beta + \sigma_\alpha \alpha_{ir}\right)^{y_{it}}\left(1 - \Phi\left(X_{it}\beta + \sigma_\alpha \alpha_{ir}\right)\right)^{1-y_{it}}\right]\right)$$

where α_{ir} is a draw from a standard normal random number generator

Pros and Cons of simulation methods

Pros:

- Doesn't require strict assumptions on the distribution of the heterogeneity (e.g. random effects can be distributed uniform, logistic, etc.)
- Very intuitive and easy to compute — it's just counting!

Cons:

- Can be very computationally intensive — need to take R draws for each person
- Simulation induces a bias, so R needs to be of the order N^2 (if doing MLE — if doing GMM, no bias)
- Inference is complicated by the fact that the asymptotic variance matrix is a function of R

Approaching integration

Here are some things to think about when doing integration

- Why do I need to integrate? Is it because of an unobserved heterogeneity assumption? If so, what is this assumption buying me above and beyond a closed-form integral assumption?
- Do I have to integrate because part of my model doesn't have a closed-form solution for an expectation?
- Can I get around the nasty computations and "integrate" by doing a discrete summation?
- Would things be easier if I just used simulation methods?

Take-home message

We got started with numerical integration because there are a few instances in economics when integration is necessary to estimate a model

- Most of the time, integration is done to correct for unobserved heterogeneity
- There are other instances, such as dynamic models (where agents can't see the future) and games (where opponents don't have full information on each other), where integration comes into play
- Multinomial probit models also require integration because the normal distribution doesn't have a closed-form solution

Whatever the reason for the numerical integration, make sure that you always have in mind why the integration is happening so you don't get confused or intimidated