Numerical Integration

Wouter J. Den Haan
London School of Economics

Quadrature techniques

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

- Nodes: x_i
- Weights: w_i

Quadrature techniques

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

Two versions:

- Newton Cotes:
  - equidistant nodes & "best" choice for the weights w_i
- Gaussian Quadrature:
  - "best" choice for both nodes and weights

Monte Carlo techniques

- pseudo:
    - implemetable version of true Monte Carlo
- quasi:
    - looks like Monte Carlo, but is something different
        - name should have been chosen better

Power

- Newton-Cotes: With n nodes you get
  - exact answer if f is $(n-1)^{\text{th}}$-order polynomial
  - accurate answer f is close to an $(n-1)^{\text{th}}$-order polynomial
- Gaussian: With n nodes you get
  - exact answer if f is $(2n-1)^{\text{th}}$-order polynomial
  - accurate answer f is close to a $(2n-1)^{\text{th}}$-order polynomial

Power

- (Pseudo) Monte Carlo: accuracy requires lots of draws
- Quasi Monte Carlo: definitely better than (pseudo) Monte Carlo and dominates quadrature methods for higher-dimensional problems

Idea behind Newton-Cotes

- function values at n nodes $\Longrightarrow$ you can fit a $(n-1)^{\text{th}}$-order polynomial & integrate the approximating polynomial

$$\int_a^b f(x)dx \approx \int_a^b P_2(x)dx$$

- It turns out that this can be standardized
  - (derivation at the end of these slides)

Simpson with 3 nodes

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

Simpson with n+1 nodes

Implement this idea over many (small) intervals we get:

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

$$= \left(\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{2}{3}f_2 + \frac{4}{3}f_3 + \frac{2}{3}f_4 + \cdots \frac{2}{3}f_{n-2} + \frac{4}{3}f_{n-1} + \frac{1}{3}f_n\right)h$$

Simpson in Matlab

- Integration routine in Matlab

```
quad(@myfun,A,B)
```

- This is an adaptive procedure that adjusts the length of the interval (by looking at changes in derivatives)

Gaussian quadrature

- Could we do better? That is, get better accuracy with same amount of nodes?
- **Answer:** Yes, if you are smart about choosing the nodes
  - This is Gaussian quadrature

Gauss-Legendre quadrature

- Let $[a,b]$ be $[-1,1]$
  - can always be accomplished by scaling
- Quadrature

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

- **Goal:** Get exact answer if $f(x)$ is a polynomial of order $2n-1$
- That is with 5 nodes you get exact answer even if $f(x)$ is a 9th-order polynomial

Implementing Gauss-Legendre quadrature

- Get n nodes and n weights from a computer program
  - ζ_i, $i = 1, \cdots, n$, ω_i, $i = 1, \cdots, n$
- Calculate the function values at the n nodes, f_i $i = 1, \cdots, n$
- Answer is equal to

$$\sum_{i=1}^{n} \omega_i f_i$$

- Anybody could do this
- How does the computer get the nodes and weights?

2n equations for nodes and weights

- To get right answer for $f(x) = 1$

$$\int_{-1}^{1} 1dx = \sum_{i=1}^{n} \omega_i 1$$

- To get right answer for $f(x) = x$

$$\int_{-1}^{1} xdx = \sum_{i=1}^{n} \omega_i \zeta_i$$

- To get right answer for $f(x) = x^2$

$$\int_{-1}^{1} x^2dx = \sum_{i=1}^{n} \omega_i \zeta_i^2$$

- etc

2n equations for nodes and weights

- To get right answer for $f(x) = x^j$ for $j = 0, \cdots, 2n-1$

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

- This is a system of $2n$ equations in $2n$ unknowns.

What has been accomplished so far?

- By construction we get right answer for

$$f(x) = 1, f(x) = x, \cdots, f(x) = x^{2n-1}$$

- But this is enough to get right answer for *any* polynomial of order $2n-1$

$$f(x) = \sum_{i=0}^{2n-1} a_i x^i$$

- Why?

Gauss-Hermite Quadrature

- Suppose we want to approximate

$$\int_{-\infty}^{\infty} f(x) e^{-x^2} dx \text{ with } \sum_{i=1}^{n} \omega_i f(\zeta_i)$$

- The function e^{-x^2} is the *weighting function*, it is not used in the approximation but is captured by the ω_i coefficients

Gauss-Hermite Quadrature

- We can use the same procedure to find the weights and the nodes, that is we solve them from the system:

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

- Note that $e^{-\zeta_i^2}$ is *not* on the right-hand side

Implementing Gauss-Hermite Quadrature

- Get n nodes, ζ_i, $i = 1, \cdots, n$, and n weights, ω_i, $i = 1, \cdots, n$, from a computer program
- Calculate the function values at the n nodes, f_i $i = 1, \cdots, n$
- Answer is equal to

$$\sum_{i=1}^{n} \omega_i f_i$$

Expectation of Normally distributed variable

- How to calculate

$$\mathsf{E}\left[h(y)\right] \text{ with } y \sim N(\mu, \sigma^2)$$

- That is, we have to calculate

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

- Unfortunately, this does not exactly fit the Hermite weighting function, but a change in variable will do the trick

Change of variables

- If $y = \phi(x)$ then

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

- Note the Jacobian is added

Change of variables

The transformation we use here is

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

Change of variables

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

$$= \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} h(\sqrt{2}\sigma x + \mu) \exp\left(-x^2\right) \sigma\sqrt{2} dx$$

$$= \int_{-\infty}^{\infty} \frac{1}{\sqrt{\pi}} h(\sqrt{2}\sigma x + \mu) \exp\left(-x^2\right) dx$$

What to do in practice?

- Obtains n Gauss-Hermite quadrature weights and nodes using a numerical algorithm.
- Calculate the approximation using

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

- Do not forget to divide by $\sqrt{\pi}$!
- Is this amazingly simple or what?

Extra material

- Derivation Simpson formula
- Monte Carlo integration

Lagrange interpolation

Let

$$L_i(x) = \frac{(x - x_0)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n)}{(x_i - x_0)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)}$$

$$f(x) \approx f_0 L_0(x) + \cdots + f_n L_n(x).$$

- What is the right-hand side?
- Do I have a perfect fit at the $n+1$ nodes?

Simpson: 2nd-order Newton-Cotes

- $x_0 = a$, $x_1 = (a+b)/2$, $x_2 = b$, or
- $x_1 = x_0 + h$, $x_2 = x_0 + 2h$

Using the Lagrange way of writing the 2nd-order polynomial, we get

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

Amazing algebra

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

- Why amazing?
  - formula only depends on h, not on values x_i and f_i
- Combining gives

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

True and pseudo Monte Carlo

To calculate an expectation

- Let x be a random variable with CDF $F(x)$
- Monte Carlo integration:

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

- Use random number generator to implement this in practice

True and pseudo Monte Carlo

What if integral is not an expectation

$$\int_a^b h(x)dx = (b-a)\int_a^b h(x) f_{\mathsf{ab}}(x)dx,$$

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

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

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

Quasi Monte Carlo

- Monte Carlo integration has very slow convergence properties
- In higher dimensional problems, however, it does better than quadrature (it seems to avoid the curse of dimensionality)
- But why? Pseudo MC is simply a deterministic way to go through the state space
- Quasi MC takes that idea and improves upon it

Quasi Monte Carlo

- Idea: Fill the space in an *efficient* way
- *Equidistributed* series: A scalar sequence $\{x_t\}_{t=1}^{T}$ is equidistributed over $[a,b]$ iff

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

  for all Rieman-integrable $f(x)$.
- Equidistributed takes the place of uniform

Quasi Monte Carlo

.

- Examples
  - $\xi, 2\xi, 3\xi, 4\xi, \cdots$ is equidistributed modulo 1 for any irrational number ξ.[1]
  - The sequence of prime numbers multiplied by an irrational number $(2\xi, 3\xi, 5\xi, 7\xi, \cdots)$

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

Multidimensional

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

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

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

Multidimensional equidistributed vectors

Examples for the d-dimensional unit hypercube:

Weyl:

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

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

Neiderreiter:

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

References

- Den Haan, W.J., Numerical Integration, online lecture notes.
- Heer, B., and A. Maussner, 2009, Dynamic General Equilibrium Modeling.
- Judd, K. L., 1998, Numerical Methods in Economics.
- Miranda, M.J, and P.L. Fackler, 2002, Applied Computational Economics and Finance.