

JDEP 384H: Numerical Methods in Business

Instructor: Thomas Shores
Department of Mathematics

Lecture 21, April 3, 2007
110 Kaufmann Center

Outline

- 1 Chapter 4: Numerical Integration: Deterministic and Monte Carlo Methods
 - BT 4.1: Numerical Integration
 - BT 4.2: Monte Carlo Integration
 - BT 4.3: Generating Pseudorandom Variates

Outline

- 1 Chapter 4: Numerical Integration: Deterministic and Monte Carlo Methods
 - BT 4.1: Numerical Integration
 - BT 4.2: Monte Carlo Integration
 - BT 4.3: Generating Pseudorandom Variates

Outline

- 1 Chapter 4: Numerical Integration: Deterministic and Monte Carlo Methods
 - BT 4.1: Numerical Integration
 - BT 4.2: Monte Carlo Integration
 - BT 4.3: Generating Pseudorandom Variates

The Basic Idea

Monte Carlo Simulation:

- Create a quantitative model of a process.
- Treat the events that constitute the process as random.
- Generate random variables to simulate the events.
- Use these values to compute the outcome of the process.

A Guiding Example is Monte Carlo Integration:

We want to approximate $\int_a^b g(x) dx$. For convenience, assume $g(x) \geq 0$, so that this integral represents (positive) area. Let's use $\int_0^1 e^x dx = e - 1 \approx 1.7183$ as a test case.

The Basic Idea

Monte Carlo Simulation:

- Create a quantitative model of a process.
- Treat the events that constitute the process as random.
- Generate random variables to simulate the events.
- Use these values to compute the outcome of the process.

A Guiding Example is Monte Carlo Integration:

We want to approximate $\int_a^b g(x) dx$. For convenience, assume $g(x) \geq 0$, so that this integral represents (positive) area. Let's use $\int_0^1 e^x dx = e - 1 \approx 1.7183$ as a test case.

The Basic Idea

Monte Carlo Simulation:

- Create a quantitative model of a process.
- Treat the events that constitute the process as random.
- Generate random variables to simulate the events.
- Use these values to compute the outcome of the process.

A Guiding Example is Monte Carlo Integration:

We want to approximate $\int_a^b g(x) dx$. For convenience, assume $g(x) \geq 0$, so that this integral represents (positive) area. Let's use $\int_0^1 e^x dx = e - 1 \approx 1.7183$ as a test case.

The Basic Idea

Monte Carlo Simulation:

- Create a quantitative model of a process.
- Treat the events that constitute the process as random.
- Generate random variables to simulate the events.
- Use these values to compute the outcome of the process.

A Guiding Example is Monte Carlo Integration:

We want to approximate $\int_a^b g(x) dx$. For convenience, assume $g(x) \geq 0$, so that this integral represents (positive) area. Let's use $\int_0^1 e^x dx = e - 1 \approx 1.7183$ as a test case.

The Basic Idea

Monte Carlo Simulation:

- Create a quantitative model of a process.
- Treat the events that constitute the process as random.
- Generate random variables to simulate the events.
- Use these values to compute the outcome of the process.

A Guiding Example is Monte Carlo Integration:

We want to approximate $\int_a^b g(x) dx$. For convenience, assume $g(x) \geq 0$, so that this integral represents (positive) area. Let's use $\int_0^1 e^x dx = e - 1 \approx 1.7183$ as a test case.

The Basic Idea

Monte Carlo Simulation:

- Create a quantitative model of a process.
- Treat the events that constitute the process as random.
- Generate random variables to simulate the events.
- Use these values to compute the outcome of the process.

A Guiding Example is Monte Carlo Integration:

We want to approximate $\int_a^b g(x) dx$. For convenience, assume $g(x) \geq 0$, so that this integral represents (positive) area. Let's use $\int_0^1 e^x dx = e - 1 \approx 1.7183$ as a test case.

Monte Carlo Integration

Hit or Miss Monte Carlo Method:

- Enclose the graph in a box of known area A (in our test case, $0 \leq x \leq 1$, $0 \leq y \leq 3$, so $A = 3$.)
- Throw N random darts at the area, uniformly distributed in x and y directions. Note: the event of a dart throw is represented by a random pair (X_i, Y_i) of independent r.v.'s.
- Count up the number N_H of darts that fall in the area, i.e., for which $Y_i \leq g(X_i)$.

- Proportionately, $\frac{\int_a^b g(x) dx}{A} \approx \frac{N_H}{N}$, so we have

$$\int_a^b g(x) dx \approx \frac{N_H}{N} A .$$

Monte Carlo Integration

Hit or Miss Monte Carlo Method:

- Enclose the graph in a box of known area A (in our test case, $0 \leq x \leq 1$, $0 \leq y \leq 3$, so $A = 3$.)
- Throw N random darts at the area, uniformly distributed in x and y directions. Note: the event of a dart throw is represented by a random pair (X_i, Y_i) of independent r.v.'s.
- Count up the number N_H of darts that fall in the area, i.e., for which $Y_i \leq g(X_i)$.

- Proportionately, $\frac{\int_a^b g(x) dx}{A} \approx \frac{N_H}{N}$, so we have

$$\int_a^b g(x) dx \approx \frac{N_H}{N} A .$$

Monte Carlo Integration

Hit or Miss Monte Carlo Method:

- Enclose the graph in a box of known area A (in our test case, $0 \leq x \leq 1$, $0 \leq y \leq 3$, so $A = 3$.)
- Throw N random darts at the area, uniformly distributed in x and y directions. Note: the event of a dart throw is represented by a random pair (X_i, Y_i) of independent r.v.'s.
- Count up the number N_H of darts that fall in the area, i.e., for which $Y_i \leq g(X_i)$.
- Proportionately, $\frac{\int_a^b g(x) dx}{A} \approx \frac{N_H}{N}$, so we have

$$\int_a^b g(x) dx \approx \frac{N_H}{N} A .$$

Monte Carlo Integration

Hit or Miss Monte Carlo Method:

- Enclose the graph in a box of known area A (in our test case, $0 \leq x \leq 1$, $0 \leq y \leq 3$, so $A = 3$.)
- Throw N random darts at the area, uniformly distributed in x and y directions. Note: the event of a dart throw is represented by a random pair (X_i, Y_i) of independent r.v.'s.
- Count up the number N_H of darts that fall in the area, i.e., for which $Y_i \leq g(X_i)$.

- Proportionately, $\frac{\int_a^b g(x) dx}{A} \approx \frac{N_H}{N}$, so we have

$$\int_a^b g(x) dx \approx \frac{N_H}{N} A .$$

Monte Carlo Integration

Hit or Miss Monte Carlo Method:

- Enclose the graph in a box of known area A (in our test case, $0 \leq x \leq 1$, $0 \leq y \leq 3$, so $A = 3$.)
- Throw N random darts at the area, uniformly distributed in x and y directions. Note: the event of a dart throw is represented by a random pair (X_i, Y_i) of independent r.v.'s.
- Count up the number N_H of darts that fall in the area, i.e., for which $Y_i \leq g(X_i)$.

- Proportionately, $\frac{\int_a^b g(x) dx}{A} \approx \frac{N_H}{N}$, so we have

$$\int_a^b g(x) dx \approx \frac{N_H}{N} A .$$

Example Calculation

Carry out the following steps in Matlab

```
> help rand
> format
> rand('seed',0)
> A = 3
> N = 10
> X = rand(N,1);
> Y = (3-0)*rand(N,1);
> hits = sum(Y <= exp(X))
> area = A*(hits/N)
> Itrue = exp(1)-1 % now try to improve accuracy
```

Monte Carlo Integration

Sample Mean Monte Carlo Method:

- Write integral as $\int_a^b g(x) dx = \int_a^b \frac{g(x)}{f(x)} f(x) dx$ where $f(x)$ is known positive p.d.f. which vanishes outside $[a, b]$.
- Interpret $\int_a^b g(x) dx = E \left[\frac{g(X)}{f(X)} \right]$ where X is r.v. with p.d.f. $f(x)$.
- Take N independent samples of X and deduce

$$\int_a^b g(x) dx \approx \frac{1}{N} \sum_{i=1}^N \frac{g(X_i)}{f(X_i)}.$$

Monte Carlo Integration

Sample Mean Monte Carlo Method:

- Write integral as $\int_a^b g(x) dx = \int_a^b \frac{g(x)}{f(x)} f(x) dx$ where $f(x)$ is known positive p.d.f. which vanishes outside $[a, b]$.

- Interpret $\int_a^b g(x) dx = E \left[\frac{g(X)}{f(X)} \right]$ where X is r.v. with p.d.f. $f(x)$.

- Take N independent samples of X and deduce

$$\int_a^b g(x) dx \approx \frac{1}{N} \sum_{i=1}^N \frac{g(X_i)}{f(X_i)}.$$

Monte Carlo Integration

Sample Mean Monte Carlo Method:

- Write integral as $\int_a^b g(x) dx = \int_a^b \frac{g(x)}{f(x)} f(x) dx$ where $f(x)$ is known positive p.d.f. which vanishes outside $[a, b]$.
- Interpret $\int_a^b g(x) dx = E \left[\frac{g(X)}{f(X)} \right]$ where X is r.v. with p.d.f. $f(x)$.
- Take N independent samples of X and deduce

$$\int_a^b g(x) dx \approx \frac{1}{N} \sum_{i=1}^N \frac{g(X_i)}{f(X_i)}.$$

Monte Carlo Integration

Sample Mean Monte Carlo Method:

- Write integral as $\int_a^b g(x) dx = \int_a^b \frac{g(x)}{f(x)} f(x) dx$ where $f(x)$ is known positive p.d.f. which vanishes outside $[a, b]$.
- Interpret $\int_a^b g(x) dx = E \left[\frac{g(X)}{f(X)} \right]$ where X is r.v. with p.d.f. $f(x)$.
- Take N independent samples of X and deduce

$$\int_a^b g(x) dx \approx \frac{1}{N} \sum_{i=1}^N \frac{g(X_i)}{f(X_i)}.$$

Example Calculation

For our example, take $f(x) = 1$. Carry out the following steps in Matlab

```
> rand('state',0)
> N = 10
> X = rand(N,1);
> expectI = sum(exp(X))/N
> Itrue = exp(1)-1 % now try to improve accuracy
```

Outline

- 1 Chapter 4: Numerical Integration: Deterministic and Monte Carlo Methods
 - BT 4.1: Numerical Integration
 - BT 4.2: Monte Carlo Integration
 - BT 4.3: Generating Pseudorandom Variates

Pseudo-Random Variables

About “random” numbers:

They aren't really random. That's why we call them “pseudo-random.” Moreover,

- They are completely deterministic, given an initial “seed” number, a uniform distribution is usually generated by a congruential formula (in Matlab, `rand` function.)
- As such, they repeat values according to a “period” which is to be avoided, as that repeating the numbers introduces discernable bias (in Matlab, old `rand` has period $2^{31} - 2$, new has period $(2^{19937} - 1) / 2$.)

Pseudo-Random Variables

About “random” numbers:

They aren't really random. That's why we call them “pseudo-random.” Moreover,

- They are completely deterministic, given an initial “seed” number, a uniform distribution is usually generated by a congruential formula (in Matlab, `rand` function.)
- As such, they repeat values according to a “period” which is to be avoided, as that repeating the numbers introduces discernable bias (in Matlab, old `rand` has period $2^{31} - 2$, new has period $(2^{19937} - 1) / 2$.)

Pseudo-Random Variables

About “random” numbers:

They aren't really random. That's why we call them “pseudo-random.” Moreover,

- They are completely deterministic, given an initial “seed” number, a uniform distribution is usually generated by a congruential formula (in Matlab, `rand` function.)
- As such, they repeat values according to a “period” which is to be avoided, as that repeating the numbers introduces discernable bias (in Matlab, old `rand` has period $2^{31} - 2$, new has period $(2^{19937} - 1) / 2$.)

Pseudo-Random Variables

About “random” numbers (continued):

They aren't really random. That's why we call them “pseudo-random.” Moreover,

- Most other distributions are simulated by various tricks applied to uniformly generated r.v.'s.
- A common method is inverse transform, which uses the fact that if $F(X)$ is the c.d.f. for r.v. X , and the inverse function F^{-1} can be found, then $U = F(X)$ is uniformly distributed on $[0, 1]$, and $X = F^{-1}(U)$ (verify this and use it in an exponential example.)
- Also used are an “acceptance-rejection” method and Box-Mueller for normal distributions (in Matlab, `randn` function) – both depend on uniform r.v.'s.

Pseudo-Random Variables

About “random” numbers (continued):

They aren't really random. That's why we call them “pseudo-random.” Moreover,

- Most other distributions are simulated by various tricks applied to uniformly generated r.v.'s.
- A common method is inverse transform, which uses the fact that if $F(X)$ is the c.d.f. for r.v. X , and the inverse function F^{-1} can be found, then $U = F(X)$ is uniformly distributed on $[0, 1]$, and $X = F^{-1}(U)$ (verify this and use it in an exponential example.)
- Also used are an “acceptance-rejection” method and Box-Mueller for normal distributions (in Matlab, `randn` function) – both depend on uniform r.v.'s.

Pseudo-Random Variables

About “random” numbers (continued):

They aren't really random. That's why we call them “pseudo-random.” Moreover,

- Most other distributions are simulated by various tricks applied to uniformly generated r.v.'s.
- A common method is inverse transform, which uses the fact that if $F(X)$ is the c.d.f. for r.v. X , and the inverse function F^{-1} can be found, then $U = F(X)$ is uniformly distributed on $[0, 1]$, and $X = F^{-1}(U)$ (verify this and use it in an exponential example.)
- Also used are an “acceptance-rejection” method and Box-Mueller for normal distributions (in Matlab, `randn` function) – both depend on uniform r.v.'s.

Pseudo-Random Variables

About “random” numbers (continued):

They aren't really random. That's why we call them “pseudo-random.” Moreover,

- Most other distributions are simulated by various tricks applied to uniformly generated r.v.'s.
- A common method is inverse transform, which uses the fact that if $F(X)$ is the c.d.f. for r.v. X , and the inverse function F^{-1} can be found, then $U = F(X)$ is uniformly distributed on $[0, 1]$, and $X = F^{-1}(U)$ (verify this and use it in an exponential example.)
- Also used are an “acceptance-rejection” method and Box-Mueller for normal distributions (in Matlab, `randn` function) – both depend on uniform r.v.'s.