Monte Carlo and Empirical Methods for Stochastic Inference (MASM11/FMSN50)

Magnus Wiktorsson
Centre for Mathematical Sciences
Lund University, Sweden

Lecture 2
Random number generation
January 23, 2020

Last time: Principal aim

We formulated the main problem of the course, namely to compute some expectation

$$\tau = \mathbb{E}(\phi(X)) = \int_{\mathsf{A}} \phi(x) f(x)\, dx,$$

where

- X is a random variable taking values in $\mathsf{A} \subseteq \mathbb{R}^d$ (where $d \in \mathbb{N}^*$ may be very large),
- $f : \mathsf{A} \to \mathbb{R}_+$ is the probability density (target density) of X, and
- $\phi : \mathsf{A} \to \mathbb{R}$ is a function (objective function) such that the above expectation exists.

This framework covers a large set of fundamental problem in statistics and numerical analysis, and we inspected a few examples.

Last time: Principal aim

We formulated the main problem of the course, namely to compute some expectation

$$\tau = \mathbb{E}(\phi(X)) = \int_{\mathsf{A}} \phi(x) f(x)\, dx,$$

where

- X is a random variable taking values in $\mathsf{A} \subseteq \mathbb{R}^d$ (where $d \in \mathbb{N}^*$ may be very large),
- $f : \mathsf{A} \to \mathbb{R}_+$ is the probability density (target density) of X, and
- $\phi : \mathsf{A} \to \mathbb{R}$ is a function (objective function) such that the above expectation exists.

This framework covers a large set of fundamental problem in statistics and numerical analysis, and we inspected a few examples.

Last time: The MC method in a nutshell

Let $X_1, X_2, \ldots, X_N$ be independent random variables with density f. Then, by the law of large numbers, as N tends to infinity,

$$\tau_N \stackrel{\text{def.}}{=} \frac{1}{N} \sum_{i=1}^{N} \phi(X_i) \to \mathbb{E}(\phi(X)). \quad \text{(a.s.)}$$

Inspired by this result, we formulated the basic MC sampler:

for $i = 1 \to N$ **do**
  draw $X_i \sim f$
end for
set $\tau_N \leftarrow \sum_{i=1}^{N} \phi(X_i)/N$
return τ_N

Last time: The MC method in a nutshell

Let $X_1, X_2, \ldots, X_N$ be independent random variables with density f. Then, by the law of large numbers, as N tends to infinity,

$$\tau_N \stackrel{\text{def.}}{=} \frac{1}{N} \sum_{i=1}^{N} \phi(X_i) \to \mathbb{E}(\phi(X)). \quad \text{(a.s.)}$$

Inspired by this result, we formulated the basic MC sampler:

for $i = 1 \to N$ **do**
  draw $X_i \sim f$
end for
set $\tau_N \leftarrow \sum_{i=1}^{N} \phi(X_i)/N$
return τ_N

What do we need to know?

OK, so what do we need to master for having practical use of the MC method?

We agreed on that, for instance, the following questions should be answered:

1: How do we generate the needed input random variables?

2: How many computer experiments should we do? What can be said about the error?

3: Can we exploit problem structure to speed up the computation

What do we need to know?

OK, so what do we need to master for having practical use of the MC method?

We agreed on that, for instance, the following questions should be answered:

1: How do we generate the needed input random variables?

2: How many computer experiments should we do? What can be said about the error?

3: Can we exploit problem structure to speed up the computation

Plan of today's lecture

1. MC output analysis
   - Confidence bounds
   - The delta method
2. Generating pseudo-random numbers
   - Uniform pseudo-random numbers
   - The inversion method
   - Rejection sampling
3. Summary

Confidence bounds

Last time we noticed that the central limit theorem (CLT) implies

$$\sqrt{N}\,(\tau_N - \tau) \xrightarrow{\text{d.}} \mathcal{N}(0, \sigma^2(\phi)), \quad \text{as } N \to \infty,$$

where

$$\sigma^2(\phi) \stackrel{\text{def.}}{=} \mathbb{V}\left(\phi(X)\right).$$

Consequently, the two-sided confidence interval

$$\mathcal{I}_\alpha = \left(\tau_N - \lambda_{\alpha/2}\frac{\sigma(\phi)}{\sqrt{N}}, \tau_N + \lambda_{\alpha/2}\frac{\sigma(\phi)}{\sqrt{N}}\right),$$

where λ_p denotes the p-quantile of the standard normal distribution, covers τ with (approximate) probability $1 - \alpha$.

A problem with this approach is that $\sigma^2(\phi)$ is in general not known.

Confidence bounds

Last time we noticed that the central limit theorem (CLT) implies

$$\sqrt{N}\,(\tau_N - \tau) \xrightarrow{\text{d.}} \mathcal{N}(0, \sigma^2(\phi)), \quad \text{as } N \to \infty,$$

where

$$\sigma^2(\phi) \stackrel{\text{def.}}{=} \mathbb{V}\,(\phi(X))\,.$$

Consequently, the two-sided confidence interval

$$\mathcal{I}_\alpha = \left(\tau_N - \lambda_{\alpha/2}\frac{\sigma(\phi)}{\sqrt{N}}, \tau_N + \lambda_{\alpha/2}\frac{\sigma(\phi)}{\sqrt{N}}\right),$$

where λ_p denotes the p-quantile of the standard normal distribution, covers τ with (approximate) probability $1-\alpha$.

A problem with this approach is that $\sigma^2(\phi)$ is in general not known.

Confidence bounds (cont.)

Quick fix: $\sigma^2(\phi)$ is again an expectation that can be estimated using the already generated MC sample $(X_i)_{i=1}^N$! More specifically, for large N's,

$$\sigma^2(\phi) = \mathbb{E}\left(\phi^2(X)\right) - \mathbb{E}^2\left(\phi(X)\right) = \mathbb{E}\left(\phi^2(X)\right) - \tau^2$$
$$\approx \frac{1}{N}\sum_{i=1}^N \phi^2(X_i) - \tau_N^2 = \frac{1}{N}\sum_{i=1}^N \left(\phi(X_i) - \frac{1}{N}\sum_{\ell=1}^N \phi(X_\ell)\right)^2.$$

This estimator is not unbiased, and one often uses instead the bias-corrected estimator

$$\sigma_N^2(\phi) \stackrel{\text{def.}}{=} \frac{1}{N-1}\sum_{i=1}^N \left(\phi(X_i) - \frac{1}{N}\sum_{\ell=1}^N \phi(X_\ell)\right)^2.$$

In Matlab, this estimator is pre-implemented in the routine `var` (see also `std`).

Confidence bounds (cont.)

Quick fix: $\sigma^2(\phi)$ is again an expectation that can be estimated using the already generated MC sample $(X_i)_{i=1}^N$! More specifically, for large N's,

$$\sigma^2(\phi) = \mathbb{E}\left(\phi^2(X)\right) - \mathbb{E}^2\left(\phi(X)\right) = \mathbb{E}\left(\phi^2(X)\right) - \tau^2$$
$$\approx \frac{1}{N}\sum_{i=1}^N \phi^2(X_i) - \tau_N^2 = \frac{1}{N}\sum_{i=1}^N \left(\phi(X_i) - \frac{1}{N}\sum_{\ell=1}^N \phi(X_\ell)\right)^2.$$

This estimator is not unbiased, and one often uses instead the bias-corrected estimator

$$\sigma_N^2(\phi) \stackrel{\text{def.}}{=} \frac{1}{N-1}\sum_{i=1}^N \left(\phi(X_i) - \frac{1}{N}\sum_{\ell=1}^N \phi(X_\ell)\right)^2.$$

In Matlab, this estimator is pre-implemented in the routine `var` (see also `std`).

Confidence bounds (cont.)

Quick fix: $\sigma^2(\phi)$ is again an expectation that can be estimated using the already generated MC sample $(X_i)_{i=1}^N$! More specifically, for large N's,

$$\sigma^2(\phi) = \mathbb{E}\left(\phi^2(X)\right) - \mathbb{E}^2\left(\phi(X)\right) = \mathbb{E}\left(\phi^2(X)\right) - \tau^2$$
$$\approx \frac{1}{N}\sum_{i=1}^{N}\phi^2(X_i) - \tau_N^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\phi(X_i) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(X_\ell)\right)^2.$$

This estimator is not unbiased, and one often uses instead the bias-corrected estimator

$$\sigma_N^2(\phi) \stackrel{\text{def.}}{=} \frac{1}{N-1}\sum_{i=1}^{N}\left(\phi(X_i) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(X_\ell)\right)^2.$$

In Matlab, this estimator is pre-implemented in the routine `var` (see also `std`).

The delta method

For a given estimand τ, one is often interested in estimating $\varphi(\tau)$ for some function $\varphi : \mathbb{R} \to \mathbb{R}$.

Question: Is it OK to simply estimate $\varphi(\tau)$ by $\varphi(\tau_N)$?

The estimator $\varphi(\tau_N)$ is not unbiased; indeed, under suitable assumptions on φ it holds that

$$\begin{aligned}\mathbb{E}\left(\varphi(\tau) - \varphi(\tau_N)\right) &= \frac{\varphi''(\tau)}{2}\mathbb{V}(\tau_N) + \mathcal{O}\left(N^{-2}\right) \\ &= \frac{\varphi''(\tau)\sigma^2(\phi)}{2N} + \mathcal{O}\left(N^{-2}\right),\end{aligned}$$

verifying that $\varphi(\tau_N)$ is asymptotically unbiased (consistent). In addition, one may establish the CLT

$$\sqrt{N}\left(\varphi(\tau_N) - \varphi(\tau)\right) \xrightarrow{\text{d.}} \mathcal{N}(0, \varphi'(\tau)^2\sigma^2(\phi)), \quad \text{as } N \to \infty,$$

which can by used for constructing CBs in the usual manner.

The delta method

For a given estimand τ, one is often interested in estimating $\varphi(\tau)$ for some function $\varphi : \mathbb{R} \to \mathbb{R}$.

Question: Is it OK to simply estimate $\varphi(\tau)$ by $\varphi(\tau_N)$?

The estimator $\varphi(\tau_N)$ is not unbiased; indeed, under suitable assumptions on φ it holds that

$$\begin{aligned}\mathbb{E}\left(\varphi(\tau) - \varphi(\tau_N)\right) &= \frac{\varphi''(\tau)}{2}\mathbb{V}(\tau_N) + \mathcal{O}\left(N^{-2}\right) \\ &= \frac{\varphi''(\tau)\sigma^2(\phi)}{2N} + \mathcal{O}\left(N^{-2}\right),\end{aligned}$$

verifying that $\varphi(\tau_N)$ is asymptotically unbiased (consistent). In addition, one may establish the CLT

$$\sqrt{N}\left(\varphi(\tau_N) - \varphi(\tau)\right) \xrightarrow{\text{d.}} \mathcal{N}(0, \varphi'(\tau)^2\sigma^2(\phi)), \quad \text{as } N \to \infty,$$

which can by used for constructing CBs in the usual manner.

The delta method

For a given estimand τ, one is often interested in estimating $\varphi(\tau)$ for some function $\varphi : \mathbb{R} \to \mathbb{R}$.

Question: Is it OK to simply estimate $\varphi(\tau)$ by $\varphi(\tau_N)$?

The estimator $\varphi(\tau_N)$ is not unbiased; indeed, under suitable assumptions on φ it holds that

$$\begin{aligned}\mathbb{E}\left(\varphi(\tau) - \varphi(\tau_N)\right) &= \frac{\varphi''(\tau)}{2}\mathbb{V}(\tau_N) + \mathcal{O}\left(N^{-2}\right) \\ &= \frac{\varphi''(\tau)\sigma^2(\phi)}{2N} + \mathcal{O}\left(N^{-2}\right),\end{aligned}$$

verifying that $\varphi(\tau_N)$ is asymptotically unbiased (consistent). In addition, one may establish the CLT

$$\sqrt{N}\left(\varphi(\tau_N) - \varphi(\tau)\right) \xrightarrow{\text{d.}} \mathcal{N}(0, \varphi'(\tau)^2\sigma^2(\phi)), \quad \text{as } N \to \infty,$$

which can by used for constructing CBs in the usual manner.

The delta method

For a given estimand τ, one is often interested in estimating $\varphi(\tau)$ for some function $\varphi : \mathbb{R} \to \mathbb{R}$.

Question: Is it OK to simply estimate $\varphi(\tau)$ by $\varphi(\tau_N)$?

The estimator $\varphi(\tau_N)$ is not unbiased; indeed, under suitable assumptions on φ it holds that

$$\begin{aligned}\mathbb{E}\left(\varphi(\tau) - \varphi(\tau_N)\right) &= \frac{\varphi''(\tau)}{2}\mathbb{V}(\tau_N) + \mathcal{O}\left(N^{-2}\right) \\ &= \frac{\varphi''(\tau)\sigma^2(\phi)}{2N} + \mathcal{O}\left(N^{-2}\right),\end{aligned}$$

verifying that $\varphi(\tau_N)$ is asymptotically unbiased (consistent). In addition, one may establish the CLT

$$\sqrt{N}\left(\varphi(\tau_N) - \varphi(\tau)\right) \xrightarrow{\text{d.}} \mathcal{N}(0, \varphi'(\tau)^2\sigma^2(\phi)), \quad \text{as } N \to \infty,$$

which can by used for constructing CBs in the usual manner.

Example: Buffon's needle (Simulation without computer)

Consider a wooden floor with parallel boards of width d on which we randomly drop a needle with length ℓ, with $\ell \le d$. Let

$$\begin{cases} X = \text{distance from the lower needlepoint to the upper board edge} \in \mathsf{U}(0,d) \\ \theta = \text{angle between the needle and the board edge normal} \in \mathsf{U}(-\pi/2,\pi/2). \end{cases}$$

Then

$$\tau = \mathbb{P}\,(\text{needle intersects board edge}) = \mathbb{P}(X \le \ell\cos\theta) = \ldots = \frac{2\ell}{\pi d}.$$

so we can estimate π as,

$$\pi = \frac{2\ell}{\tau d}.$$

This may not look well suited for computer implementation since the simulation of θ seems to need the value of π.

Example: Buffon's needle (Simulation without computer)

Consider a wooden floor with parallel boards of width d on which we randomly drop a needle with length ℓ, with $\ell \leq d$. Let

$$\begin{cases} X = \text{distance from the lower needlepoint to the upper board edge} \in \mathsf{U}(0,d) \\ \theta = \text{angle between the needle and the board edge normal} \in \mathsf{U}(-\pi/2,\pi/2). \end{cases}$$

Then

$$\tau = \mathbb{P}\left(\text{needle intersects board edge}\right) = \mathbb{P}(X \leq \ell\cos\theta) = \ldots = \frac{2\ell}{\pi d}.$$

so we can estimate π as,

$$\pi = \frac{2\ell}{\tau d}.$$

This may not look well suited for computer implementation since the simulation of θ seems to need the value of π.

Example: Buffon's needle (cont.)

However if $U_1, U_2 \in \mathsf{U}(0,1)$ then

$$Y = \frac{U_1}{\sqrt{U_1^2 + U_2^2}} | \{U_1^2 + U_2^2 \leq 1\} \stackrel{d}{=} \cos(\theta).$$

So we can draw U_1 and U_2 and generate Y if $U_1^2 + U_2^2 \leq 1$. We will soon talk more about this (rejection sampling).
But if we analyse the probability for this event to happen we see that

$$\mathbb{P}(U_1^2 + U_2^2 \leq 1) = \pi/4.$$

So this actually suggests a better way to estimate π directly.

Example: Buffon's needle (cont.)

Since

$$\tau = \mathbb{P}\left(\text{needle intersects board edge}\right) = \mathbb{E}\left(\mathbb{1}_{\{X \leq \ell \cos\theta\}}\right),$$

an MC approximation of π is obtained by using the delta method by first estimating τ via

```
X = d*rand(1,N);
for i=1:N
 ready=0;
 while ~ready
   U=rand(2,1);
   ready=sum(U.^2)<=1;
 end
 costheta(i)=U(1)/sqrt(sum(U.^2));
end
tau = mean(X <= L*costheta);
```

and then letting

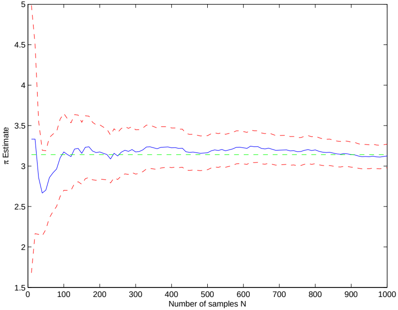
```
pi_est = 2*L./(tau*d);
```

Example: Buffon's needle (cont.)

The delta method provides a 95% confidence interval through

```
sigma = std(X <= L*costheta);
LB = pi_est - norminv(0.975)*2*L/(d*tau^2*sqrt(N))*sigma;
UB = pi_est + norminv(0.975)*2*L/(d*tau^2*sqrt(N))*sigma;
```

Executing this code (and the previous) for `N = 1:10:1000` yields the following graph:



Generating pseudo-random numbers

Pseudo-random numbers = Numbers exhibiting statistical randomness while being generated by a deterministic process.

We will discuss

- how to generate pseudo-random $\mathcal{U}(0,1)$ numbers,
- inversion and transformation methods,
- rejection sampling, and
- conditional methods.

Good pseudo-random numbers

"Good" pseudo-random numbers

- appear to come from the correct distribution (also in the tails),
- have long periodicity,
- are "independent" and
- fast to generate.

Most standard computing languages have packages or functions that generate either $\mathcal{U}(0,1)$ random numbers or integers on $\mathcal{U}(0,2^{32}-1)$:

- `rand` and `unifrnd` in matlab
- `rand` in C/C++
- `Random` in Java

Good pseudo-random numbers

"Good" pseudo-random numbers

- appear to come from the correct distribution (also in the tails),
- have long periodicity,
- are "independent" and
- fast to generate.

Most standard computing languages have packages or functions that generate either $\mathcal{U}(0,1)$ random numbers or integers on $\mathcal{U}(0,2^{32}-1)$:

- `rand` and `unifrnd` in matlab
- `rand` in C/C++
- `Random` in Java

Linear congruential generator (LCG)

The linear congruential generator is a simple, fast, and popular way of generating random numbers:

$$X_n = (a \cdot X_{n-1} + c) \mod m,$$

where a, c, and m are integers. This recursion generates integer numbers (X_n) in $[0, m-1]$. These are mapped to $(0, 1)$ through division by m. It turns out that the period of the generator is m if (for $c > 0$)

(i) c and m are relatively prime,

(ii) $a - 1$ is divisible by all prime factors of m, and

(iii) $a - 1$ is divisible by 4 if m is divisible by 4.

This is known as the Hull-Dobell Theorem (1962). "Thus with m as a power of 2, as natural on a binary machine, we need only to have c odd and $a \mod 4 = 1$" (Hull-Dobell,1962).

Linear congruential generator (LCG)

The linear congruential generator is a simple, fast, and popular way of generating random numbers:

$$X_n = (a \cdot X_{n-1} + c) \mod m,$$

where a, c, and m are integers. This recursion generates integer numbers (X_n) in $[0, m-1]$. These are mapped to $(0, 1)$ through division by m. It turns out that the period of the generator is m if (for $c > 0$)

(i) c and m are relatively prime,

(ii) $a - 1$ is divisible by all prime factors of m, and

(iii) $a - 1$ is divisible by 4 if m is divisible by 4.

This is known as the Hull-Dobell Theorem (1962). "Thus with m as a power of 2, as natural on a binary machine, we need only to have c odd and $a \mod 4 = 1$" (Hull-Dobell,1962).

Multiplicative congruential generator (MCG)

The multiplicative congruential generator is special case of LCG:s where $c = 0$:

$$X_n = a \cdot X_{n-1} \mod m,$$

where a, m are integers. This recursion generates integer numbers (X_n) in $[1, m-1]$. These are mapped to $(0, 1)$ through division by m. It turns out that the period of the generator is $m - 1$ if

(i) The number m is a prime,

(ii) The multiplier a is is primitive root of m, and

(iii) $x_0 \in [1, m-1]$.

The number a is a primitive root of m if and only if $a \mod m \neq 0$ and $a^{(m-1)/q} \mod m \neq 1$, for any prime divisor q of $m - 1$.

As an example, MATLAB (pre v. 5) uses $m = 2^{31} - 1$, $a = 7^5 = 16807$. Now MATLAB uses the Mersenne Twister alogorithm.

Multiplicative congruential generator (MCG)

The multiplicative congruential generator is special case of LCG:s where $c = 0$:

$$X_n = a \cdot X_{n-1} \mod m,$$

where a, m are integers. This recursion generates integer numbers (X_n) in $[1, m-1]$. These are mapped to $(0, 1)$ through division by m. It turns out that the period of the generator is $m - 1$ if

(i) The number m is a prime,

(ii) The multiplier a is is primitive root of m, and

(iii) $x_0 \in [1, m-1]$.

The number a is a primitive root of m if and only if $a \mod m \neq 0$ and $a^{(m-1)/q} \mod m \neq 1$, for any prime divisor q of $m - 1$.

As an example, MATLAB (pre v. 5) uses $m = 2^{31} - 1$, $a = 7^5 = 16807$. Now MATLAB uses the Mersenne Twister alogorithm.

Multiplicative congruential generator (MCG)

The multiplicative congruential generator is special case of LCG:s where $c = 0$:

$$X_n = a \cdot X_{n-1} \mod m,$$

where a, m are integers. This recursion generates integer numbers (X_n) in $[1, m-1]$. These are mapped to $(0,1)$ through division by m. It turns out that the period of the generator is $m-1$ if

(i) The number m is a prime,

(ii) The multiplier a is is primitive root of m, and

(iii) $x_0 \in [1, m-1]$.

The number a is a primitive root of m if and only if $a \mod m \neq 0$ and $a^{(m-1)/q} \mod m \neq 1$, for any prime divisor q of $m-1$.

As an example, MATLAB (pre v. 5) uses $m = 2^{31} - 1$, $a = 7^5 = 16807$. Now MATLAB uses the Mersenne Twister alogorithm.

Multiplicative congruential generator (MCG)

The multiplicative congruential generator is special case of LCG:s where $c = 0$:

$$X_n = a \cdot X_{n-1} \mod m,$$

where a, m are integers. This recursion generates integer numbers (X_n) in $[1, m-1]$. These are mapped to $(0, 1)$ through division by m. It turns out that the period of the generator is $m - 1$ if

(i) The number m is a prime,

(ii) The multiplier a is is primitive root of m, and

(iii) $x_0 \in [1, m-1]$.

The number a is a primitive root of m if and only if $a \mod m \neq 0$ and $a^{(m-1)/q} \mod m \neq 1$, for any prime divisor q of $m - 1$.

As an example, MATLAB (pre v. 5) uses $m = 2^{31} - 1$, $a = 7^5 = 16807$. Now MATLAB uses the Mersenne Twister alogorithm.

The inversion method

We will now assume that we have access to $\mathcal{U}(0,1)$ pseudo-random numbers U and want to generate random numbers X from a univariate distribution with distribution function F.

Define the general inverse $F^{\leftarrow}(u) \stackrel{\text{def.}}{=} \inf\{x \in \mathbb{R} : F(x) \geq u\}$ and

draw $U \sim \mathcal{U}(0,1)$
set $X \leftarrow F^{\leftarrow}(U)$
return X

One may now prove the following.

Theorem (Inverse method)

The output X has distribution function F.

The inversion method

We will now assume that we have access to $\mathcal{U}(0,1)$ pseudo-random numbers U and want to generate random numbers X from a univariate distribution with distribution function F.

Define the general inverse $F^{\leftarrow}(u) \overset{\text{def.}}{=} \inf\{x \in \mathbb{R} : F(x) \geq u\}$ and

draw $U \sim \mathcal{U}(0,1)$
set $X \leftarrow F^{\leftarrow}(U)$
return X

One may now prove the following.

Theorem (Inverse method)

The output X has distribution function F.

The inversion method

We will now assume that we have access to $\mathcal{U}(0,1)$ pseudo-random numbers U and want to generate random numbers X from a univariate distribution with distribution function F.

Define the general inverse $F^{\leftarrow}(u) \stackrel{\text{def.}}{=} \inf\{x \in \mathbb{R} : F(x) \geq u\}$ and

draw $U \sim \mathcal{U}(0,1)$
set $X \leftarrow F^{\leftarrow}(U)$
return X

One may now prove the following.

Theorem (Inverse method)

The output X has distribution function F.

The inversion method (cont.)

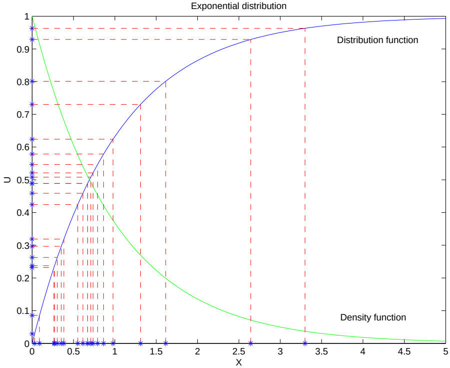
Some remarks:

- If F is strictly monotone, then $F^{\leftarrow} = F^{-1}$.
- The method is limited to cases where
  - we want to generate univariate random numbers and
  - the generalized inverse $F^{\leftarrow}$ is easy to evaluate (which is far from always the case).

Example: exponential distribution

$$F(x) = 1 - e^{-x}, x \in \mathbb{R}_+ \Leftrightarrow F^{\leftarrow}(u) = -\log(1-u), u \in (0,1)$$

```
F_inv = @(y) - log(1 - y);
U = rand(1,20);
X = F_inv(U);
```



Rejection sampling

The inversion method looks promising, but what do we do if, e.g., $f(x) \propto \exp(\cos^2(x))$, $x \in (-\pi/2, \pi/2)$? Here we cannot find an inverse and even the normalizing constant is hard to calculate. ☹

In the continuous case the following (somewhat magic!) algorithm saves the day. Let f and g be densities on $\mathbb{R}^d$ for which there exists a constant $K < \infty$ such that $f(x) \leq Kg(x)$ for all $x \in \mathbb{R}^d$; then

repeat
  draw $X^* \sim g$
  draw $U \sim \mathcal{U}(0, 1)$
until $U \leq \frac{f(X^*)}{Kg(X^*)}$
$X \leftarrow X^*$
return X

Rejection sampling

The inversion method looks promising, but what do we do if, e.g., $f(x) \propto \exp(\cos^2(x))$, $x \in (-\pi/2, \pi/2)$? Here we cannot find an inverse and even the normalizing constant is hard to calculate. ☹

In the continuous case the following (somewhat magic!) algorithm saves the day. Let f and g be densities on $\mathbb{R}^d$ for which there exists a constant $K < \infty$ such that $f(x) \le Kg(x)$ for all $x \in \mathbb{R}^d$; then

repeat
  draw $X^* \sim g$
  draw $U \sim \mathcal{U}(0,1)$
until $U \le \frac{f(X^*)}{Kg(X^*)}$
$X \leftarrow X^*$
return X

Rejection sampling (cont.)

The following holds true:

Theorem (Rejection sampling)

The output X of the rejection sampling algorithm has density function f.

Moreover:

Theorem

The expected number of trials needed before acceptance is K.

Consequently, K should be chosen as small as possible.

Rejection sampling (cont.)

The following holds true:

Theorem (Rejection sampling)

The output X of the rejection sampling algorithm has density function f.

Moreover:

Theorem

The expected number of trials needed before acceptance is K.

Consequently, K should be chosen as small as possible.

Example

We wish to simulate $f(x) = \exp(\cos^2(x))/c$, $x \in (-\pi/2, \pi/2)$, where $c = \int_{-\pi/2}^{\pi/2} \exp(\cos^2(z))\, dz = \pi e^{1/2} I_o(1/2)$ is the normalizing constant.

However, since for all $x \in (-\pi/2, \pi/2)$,

$$f(x) = \frac{\exp(\cos^2(x))}{c} \le \frac{e}{c} = \underbrace{\frac{e\pi}{c}}_{K} \times \underbrace{\frac{1}{\pi}}_{g},$$

where g is the density of $\mathcal{U}(-\pi/2, \pi/2)$, we may use rejection sampling where a candidate $X^* \sim \mathcal{U}(-\pi/2, \pi/2)$ is accepted if

$$U \le \frac{f(X^*)}{Kg(X^*)} = \frac{\exp(\cos^2(X^*))/c}{e/c} = \exp(\cos^2(X^*) - 1).$$

Example

We wish to simulate $f(x) = \exp(\cos^2(x))/c$, $x \in (-\pi/2, \pi/2)$, where $c = \int_{-\pi/2}^{\pi/2} \exp(\cos^2(z))\,dz = \pi e^{1/2} I_o(1/2)$ is the normalizing constant.

However, since for all $x \in (-\pi/2, \pi/2)$,

$$f(x) = \frac{\exp(\cos^2(x))}{c} \leq \frac{e}{c} = \underbrace{\frac{e\pi}{c}}_{K} \times \underbrace{\frac{1}{\pi}}_{g},$$

where g is the density of $\mathcal{U}(-\pi/2, \pi/2)$, we may use rejection sampling where a candidate $X^* \sim \mathcal{U}(-\pi/2, \pi/2)$ is accepted if

$$U \leq \frac{f(X^*)}{Kg(X^*)} = \frac{\exp(\cos^2(X^*))/c}{e/c} = \exp(\cos^2(X^*) - 1).$$

```
prob = @(x) exp((cos(x))^2 - 1);
trial = 1;
accepted = false;
while ~accepted,
    Xcand = - pi/2 + pi*rand;
    if rand < prob(Xcand),
        accepted = true;
        X = Xcand;
    else
    trial = trial + 1;
    end
end
```

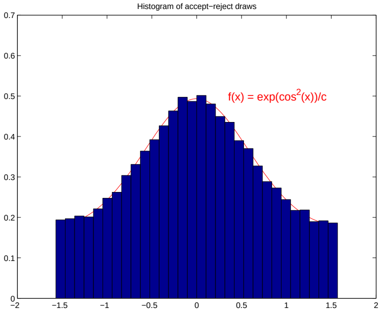


Figure: Plot of a histogram of $20{,}000$ accept-reject draws together with the true density. The average number of trials was $1.5555 (\approx K = e^{1/2}/I_0(1/2) \approx 1.5503)$.

Summary

Today we have

- discussed how to construct confidence intervals of the MC estimates using the CLT,
- shown that natural estimator $\varphi(\tau_N)$ of $\varphi(\tau)$ is asymptotically consistent,
- shown how to generate pseudo-random numbers using
  - the inversion method (when the general inverse $F^{\leftarrow}$ of F is easily obtained),
  - rejection sampling (when $f(x) \leq Kg(x)$ for some density g and constant K),