

Basics on simulation

Christian Lantuéjoul & Thomas Romary

christian.lantuejoul@mines-paristech.fr

thomas.romary@mines-paristech.fr



Outline

- 1 Uniform generation
 - Uniform generation on a unit segment
 - Uniform generation in a low-dimensional domain
- 2 Random Variable Simulation
 - Inversion method
 - Rejection method
 - Ad hoc methods
- 3 Uniform generation in a high-dimensional domain
 - Presentation of the problem
 - Hit-and-run algorithm
 - Gibbs sampler
 - Metropolis-Hastings algorithm

Outline

- 1 Uniform generation
 - Uniform generation on a unit segment
 - Uniform generation in a low-dimensional domain
- 2 Random Variable Simulation
 - Inversion method
 - Rejection method
 - Ad hoc methods
- 3 Uniform generation in a high-dimensional domain
 - Presentation of the problem
 - Hit-and-run algorithm
 - Gibbs sampler
 - Metropolis-Hastings algorithm

Uniform distribution on $]0, 1[$

Definition (Probability density function (p.d.f.))

$$f(u) = \begin{cases} 1 & \text{if } 0 < u < 1 \\ 0 & \text{otherwise} \end{cases}$$

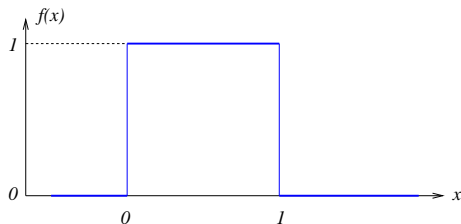
Notation

$U \sim \mathcal{U}$ means that the random variable U is uniformly distributed on $]0, 1[$

Uniform distribution on $]0, 1[$

Definition (Probability density function (p.d.f.))

$$f(u) = \begin{cases} 1 & \text{if } 0 < u < 1 \\ 0 & \text{otherwise} \end{cases}$$



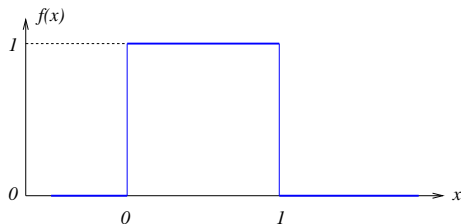
Notation

$U \sim \mathcal{U}$ means that the random variable U is uniformly distributed on $]0, 1[$

Uniform distribution on $]0, 1[$

Definition (Probability density function (p.d.f.))

$$f(u) = \begin{cases} 1 & \text{if } 0 < u < 1 \\ 0 & \text{otherwise} \end{cases}$$



Notation

$U \sim \mathcal{U}$ means that the random variable U is uniformly distributed on $]0, 1[$

Uniform pseudo-random number generator

Definition

Starting from an **initial value** u_0 and a **transformation** T , it produces a sequence (u_n) defined by

$$\begin{aligned}u_1 &= Tu_0 \\u_2 &= Tu_1 = T^2u_0 \\u_n &= Tu_{n-1} = \cdots = T^nu_0 \\&\vdots\end{aligned}$$

u_0 and T are chosen in such a way that (u_1, \dots, u_n) is statistically close to a sequence of independent and uniform values on $]0, 1[$.

Remark (Statistical tests)

Kolmogorov-Smirnov, Lehmann, chi square ...

Uniform pseudo-random number generator

Definition

Starting from an **initial value** u_0 and a **transformation** T , it produces a sequence (u_n) defined by

$$\begin{aligned}u_1 &= Tu_0 \\u_2 &= Tu_1 = T^2u_0 \\u_n &= Tu_{n-1} = \cdots = T^nu_0 \\&\vdots\end{aligned}$$

u_0 and T are chosen in such a way that (u_1, \dots, u_n) is statistically close to a sequence of independent and uniform values on $]0, 1[$.

Remark (Statistical tests)

Kolmogorov-Smirnov, Lehmann, chi square ...

Example: The congruence method (Lehmer 1951)

Ingredient

Two positive integers a and m . a is a **multiplier** and m a **modulo**

Notation

The remainder of the division of x by m is denoted $x \bmod m$

Algorithm (Principle)

- (i) *choose an integer x_0 between 1 and $m - 1$ (the **seed**)*
- (ii) *for $n = 0, 1, 2, \dots$ compute $x_{n+1} = a x_n \bmod m$*
- (iii) *for $n = 0, 1, 2, \dots$ put $u_n = x_n / m$*

Example: The congruence method (Lehmer 1951)

Ingredient

Two positive integers a and m . a is a **multiplier** and m a **modulo**

Notation

The remainder of the division of x by m is denoted $x \bmod m$

Algorithm (Principle)

- (i) *choose an integer x_0 between 1 and $m - 1$ (the **seed**)*
- (ii) *for $n = 0, 1, 2, \dots$ compute $x_{n+1} = a x_n \bmod m$*
- (iii) *for $n = 0, 1, 2, \dots$ put $u_n = x_n/m$*

Random bit generators

Idea

Random bit generators rely on physical processes that are governed by the laws of quantum mechanics. They are completely unpredictable by nature

Property (Pros and cons)

- Provides random numbers with any prespecified precision
- Good statistical properties
- Experiments cannot be repeated

Random bit generators

Idea

Random bit generators rely on physical processes that are governed by the laws of quantum mechanics. They are completely unpredictable by nature



Property (Pros and cons)

- Provides random numbers with any prespecified precision
- Good statistical properties
- Experiments cannot be repeated

Random bit generators

Idea

Random bit generators rely on physical processes that are governed by the laws of quantum mechanics. They are completely unpredictable by nature



Property (Pros and cons)

- Provides random numbers with any prespecified precision
- Good statistical properties
- Experiments cannot be repeated

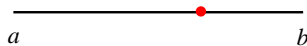
Outline

- 1 Uniform generation
 - Uniform generation on a unit segment
 - Uniform generation in a low-dimensional domain
- 2 Random Variable Simulation
 - Inversion method
 - Rejection method
 - Ad hoc methods
- 3 Uniform generation in a high-dimensional domain
 - Presentation of the problem
 - Hit-and-run algorithm
 - Gibbs sampler
 - Metropolis-Hastings algorithm

Uniform distribution on an interval

Question

How to generate a uniform point on the interval $]a, b[$?



Definition (Probability density function)

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{if } a < x < b \\ 0 & \text{otherwise} \end{cases}$$

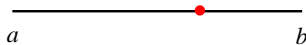
Algorithm

- (i) *generate* $u \sim \mathcal{U}$
- (ii) *return* $a + (b - a)u$

Uniform distribution on an interval

Question

How to generate a uniform point on the interval $]a, b[$?



Definition (Probability density function)

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{if } a < x < b \\ 0 & \text{otherwise} \end{cases}$$

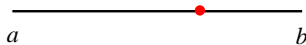
Algorithm

- (i) *generate* $u \sim \mathcal{U}$
- (ii) *return* $a + (b - a)u$

Uniform distribution on an interval

Question

How to generate a uniform point on the interval $]a, b[$?



Definition (Probability density function)

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{if } a < x < b \\ 0 & \text{otherwise} \end{cases}$$

Algorithm

- (i) *generate* $u \sim \mathcal{U}$
- (ii) *return* $a + (b - a)u$

Uniform distribution on a union of intervals

Notation

The objective is to generate a uniform point on the union of pairwise disjoint intervals $]a_1, b_1[, \dots,]a_k, b_k[$. In what follows, $\ell_i = b_i - a_i$ is the length of $]a_i, b_i[$, and $\ell = \sum_{i=1}^k \ell_i$

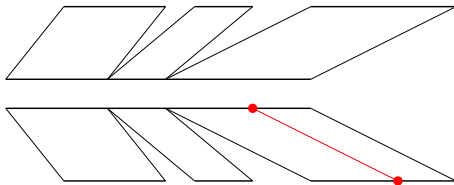
Algorithm

- (i) generate $u \sim \mathcal{U}$
- (ii) find i such that $\sum_{j \leq i-1} \ell_j \leq \ell u < \sum_{j \leq i} \ell_j$
- (iii) return $a_i + \ell u - \sum_{j \leq i-1} \ell_j$

Uniform distribution on a union of intervals

Notation

The objective is to generate a uniform point on the union of pairwise disjoint intervals $]a_1, b_1[, \dots,]a_k, b_k[$. In what follows, $\ell_i = b_i - a_i$ is the length of $]a_i, b_i[$, and $\ell = \sum_{i=1}^k \ell_i$



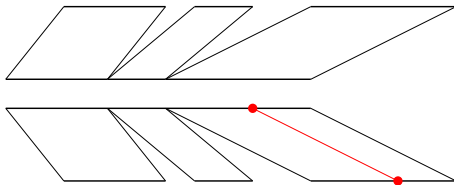
Algorithm

- (i) generate $u \sim \mathcal{U}$
- (ii) find i such that $\sum_{j \leq i-1} \ell_j \leq \ell u < \sum_{j \leq i} \ell_j$
- (iii) return $a_i + \ell u - \sum_{j \leq i-1} \ell_j$

Uniform distribution on a union of intervals

Notation

The objective is to generate a uniform point on the union of pairwise disjoint intervals $]a_1, b_1[, \dots,]a_k, b_k[$. In what follows, $\ell_i = b_i - a_i$ is the length of $]a_i, b_i[$, and $\ell = \sum_{i=1}^k \ell_i$



Algorithm

- (i) generate $u \sim \mathcal{U}$
- (ii) find i such that $\sum_{j \leq i-1} \ell_j \leq \ell u < \sum_{j \leq i} \ell_j$
- (iii) return $a_i + \ell u - \sum_{j \leq i-1} \ell_j$

Uniform distribution on a rectangle

Notation

Let R be a rectangle with vertices a, b, c, d

Probability density function

$$f(x) = \begin{cases} \frac{1}{|b-a||c-a|} & \text{if } x \in R \\ 0 & \text{otherwise} \end{cases}$$

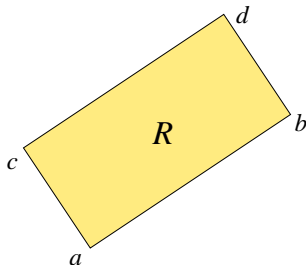
Algorithm

- (i) *generate* $u, v \sim \mathcal{U}$
- (ii) *return* $a + (b-a)u + (c-a)v$

Uniform distribution on a rectangle

Notation

Let R be a rectangle with vertices a, b, c, d



Probability density function

$$f(x) = \begin{cases} \frac{1}{|b-a||c-a|} & \text{if } x \in R \\ 0 & \text{otherwise} \end{cases}$$

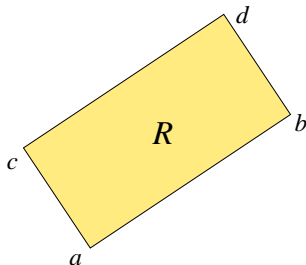
Algorithm

- (i) generate $u, v \sim \mathcal{U}$
- (ii) return $a + (b-a)u + (c-a)v$

Uniform distribution on a rectangle

Notation

Let R be a rectangle with vertices a, b, c, d



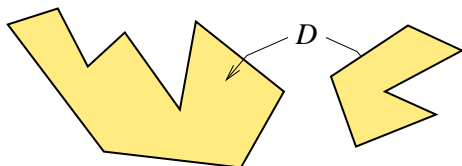
Probability density function

$$f(x) = \begin{cases} \frac{1}{|b-a||c-a|} & \text{if } x \in R \\ 0 & \text{otherwise} \end{cases}$$

Algorithm

- (i) generate $u, v \sim \mathcal{U}$
- (ii) return $a + (b-a)u + (c-a)v$

Uniform distribution on a domain



Definition

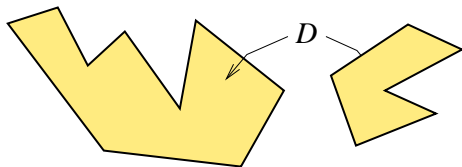
Let D be a subset of \mathbb{R}^2 with **finite area** $|D|$. A random point X is said to be **uniformly distributed on D** if its p.d.f. is equal to

$$f(x) = \begin{cases} \frac{1}{|D|} & \text{if } x \in D \\ 0 & \text{otherwise} \end{cases}$$

Notation

This is usually denoted by $X \sim \mathcal{U}(D)$

Uniform distribution on a domain



Definition

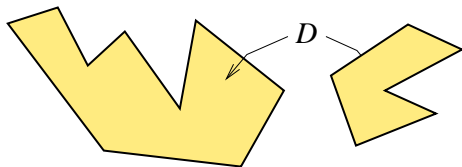
Let D be a subset of \mathbb{R}^2 with **finite area** $|D|$. A random point X is said to be **uniformly distributed on D** if its p.d.f. is equal to

$$f(x) = \begin{cases} \frac{1}{|D|} & \text{if } x \in D \\ 0 & \text{otherwise} \end{cases}$$

Notation

This is usually denoted by $X \sim \mathcal{U}(D)$

Uniform distribution on a domain



Definition

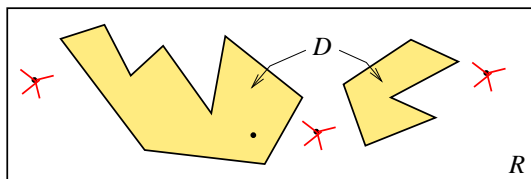
Let D be a subset of \mathbb{R}^2 with **finite area** $|D|$. A random point X is said to be **uniformly distributed on D** if its p.d.f. is equal to

$$f(x) = \begin{cases} \frac{1}{|D|} & \text{if } x \in D \\ 0 & \text{otherwise} \end{cases}$$

Notation

This is usually denoted by $X \sim \mathcal{U}(D)$

Uniform generation on a bounded domain



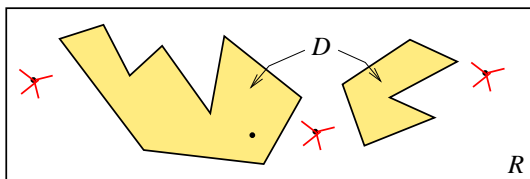
Algorithm (Rejection method)

- (i) generate x from $\mathcal{U}(R)$
- (ii) if $x \notin D$, then goto (i)
- (iii) return x

Remark

The **efficiency** of this algorithm is $\frac{|D|}{|R|} \approx 40\%$

Uniform generation on a bounded domain



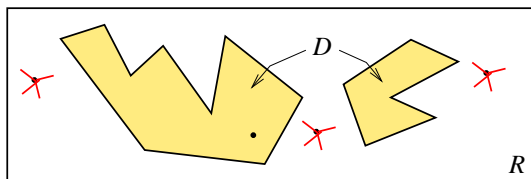
Algorithm (Rejection method)

- (i) *generate x from $\mathcal{U}(R)$*
- (ii) *if $x \notin D$, then goto (i)*
- (iii) *return x*

Remark

The **efficiency** of this algorithm is $\frac{|D|}{|R|} \approx 40\%$

Uniform generation on a bounded domain



Algorithm (Rejection method)

- (i) generate x from $\mathcal{U}(R)$
- (ii) if $x \notin D$, then goto (i)
- (iii) return x

Remark

The **efficiency** of this algorithm is $\frac{|D|}{|R|} \approx 40\%$

Outline

- 1 Uniform generation
 - Uniform generation on a unit segment
 - Uniform generation in a low-dimensional domain
- 2 Random Variable Simulation
 - Inversion method
 - Rejection method
 - Ad hoc methods
- 3 Uniform generation in a high-dimensional domain
 - Presentation of the problem
 - Hit-and-run algorithm
 - Gibbs sampler
 - Metropolis-Hastings algorithm

Definition and problem

Definition

To **simulate** a distribution F means to produce a realization of a random phenomenon that follows the distribution F

The produced realization is said to have been **drawn** or **generated** from distribution F

Problem

How to simulate standard distributions, such as the exponential, Gaussian, Poisson distributions?

Definition and problem

Definition

To **simulate** a distribution F means to produce a realization of a random phenomenon that follows the distribution F

The produced realization is said to have been **drawn** or **generated** from distribution F

Problem

How to simulate standard distributions, such as the exponential, Gaussian, Poisson distributions?

Outline

- 1 Uniform generation
 - Uniform generation on a unit segment
 - Uniform generation in a low-dimensional domain
- 2 Random Variable Simulation
 - Inversion method
 - Rejection method
 - Ad hoc methods
- 3 Uniform generation in a high-dimensional domain
 - Presentation of the problem
 - Hit-and-run algorithm
 - Gibbs sampler
 - Metropolis-Hastings algorithm

The inversion method

Problem

Simulate a distribution with cumulative distribution function (c.d.f.) F

Principle

If $U \sim \mathcal{U}$, then $F^{-1}(U) \sim F$

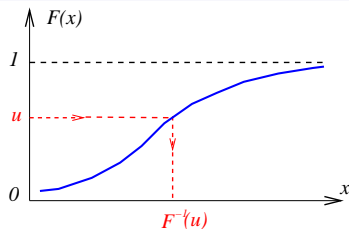
Algorithm

- (i) *generate u from \mathcal{U}*
- (ii) *deliver $F^{-1}(u)$*

The inversion method

Problem

Simulate a distribution with cumulative distribution function (c.d.f.) F



Principle

If $U \sim \mathcal{U}$, then $F^{-1}(U) \sim F$

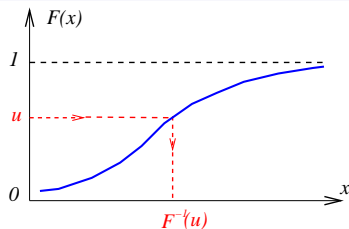
Algorithm

- (i) generate u from \mathcal{U}
- (ii) deliver $F^{-1}(u)$

The inversion method

Problem

Simulate a distribution with cumulative distribution function (c.d.f.) F



Principle

If $U \sim \mathcal{U}$, then $F^{-1}(U) \sim F$

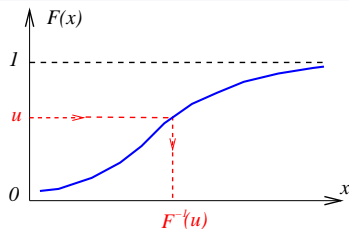
Algorithm

- (i) generate u from \mathcal{U}
- (ii) deliver $F^{-1}(u)$

The inversion method

Problem

Simulate a distribution with cumulative distribution function (c.d.f.) F



Principle

If $U \sim \mathcal{U}$, then $F^{-1}(U) \sim F$

Algorithm

- (i) generate u from \mathcal{U}
- (ii) deliver $F^{-1}(u)$

Example: the exponential distribution

Definition

A random variable follows an **exponential distribution** $\mathcal{E}(a)$ with parameter a (or mean $1/a$) if its p.d.f. is equal to $f(x) = ae^{-ax}$ for $x \geq 0$. Its c.d.f. is $F(x) = 1 - e^{-ax}$.

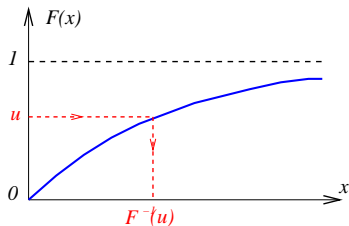
Algorithm

- (i) *generate u from \mathcal{U}*
- (ii) *deliver $-\frac{1}{a} \ln u$*

Example: the exponential distribution

Definition

A random variable follows an **exponential distribution** $\mathcal{E}(a)$ with parameter a (or mean $1/a$) if its p.d.f. is equal to $f(x) = ae^{-ax}$ for $x \geq 0$. Its c.d.f. is $F(x) = 1 - e^{-ax}$.



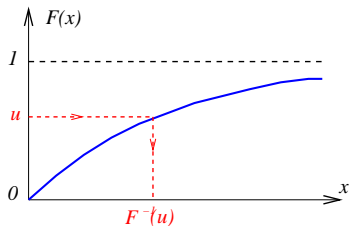
Algorithm

- (i) generate u from \mathcal{U}
- (ii) deliver $-\frac{1}{a} \ln u$

Example: the exponential distribution

Definition

A random variable follows an **exponential distribution** $\mathcal{E}(a)$ with parameter a (or mean $1/a$) if its p.d.f. is equal to $f(x) = ae^{-ax}$ for $x \geq 0$. Its c.d.f. is $F(x) = 1 - e^{-ax}$.



Algorithm

- (i) generate u from \mathcal{U}
- (ii) deliver $-\frac{1}{a} \ln u$

Example: the Bernoulli distribution

Definition

A random variable X follows a Bernoulli distribution $\mathcal{B}(p)$ with mean p if it is equal to **1** with probability p , and to **0** with the complementary probability $q = 1 - p$. Its c.d.f. $F(x)$ is **0** if $x \leq 0$, **q** if $0 < x \leq 1$ and **1** if $x > 1$

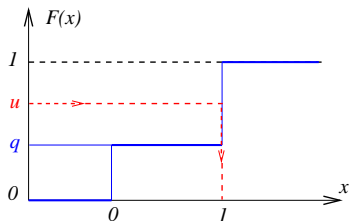
Algorithm

- (i) *generate u from \mathcal{U}*
- (ii) *deliver 0 if $u \leq q$ and 1 otherwise*

Example: the Bernoulli distribution

Definition

A random variable X follows a Bernoulli distribution $\mathcal{B}(p)$ with mean p if it is equal to **1** with probability p , and to **0** with the complementary probability $q = 1 - p$. Its c.d.f. $F(x)$ is **0** if $x \leq 0$, **q** if $0 < x \leq 1$ and **1** if $x > 1$



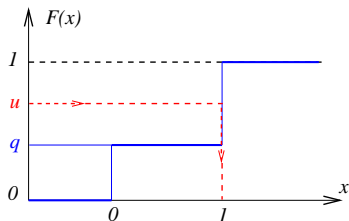
Algorithm

- (i) generate u from \mathcal{U}
- (ii) deliver 0 if $u \leq q$ and 1 otherwise

Example: the Bernoulli distribution

Definition

A random variable X follows a Bernoulli distribution $\mathcal{B}(p)$ with mean p if it is equal to **1** with probability p , and to **0** with the complementary probability $q = 1 - p$. Its c.d.f. $F(x)$ is **0** if $x \leq 0$, **q** if $0 < x \leq 1$ and **1** if $x > 1$



Algorithm

- (i) generate u from \mathcal{U}
- (ii) deliver 0 if $u \leq q$ and 1 otherwise

Outline

- 1 Uniform generation
 - Uniform generation on a unit segment
 - Uniform generation in a low-dimensional domain
- 2 Random Variable Simulation
 - Inversion method
 - Rejection method
 - Ad hoc methods
- 3 Uniform generation in a high-dimensional domain
 - Presentation of the problem
 - Hit-and-run algorithm
 - Gibbs sampler
 - Metropolis-Hastings algorithm

Why considering uniform distributions?

Notation

Let f be a **p.d.f.** defined on \mathbb{R}^d and let S_f be its **subgraph**

$$S_f = \{(x, y) \in \mathbb{R}^d \times \mathbb{R} : 0 \leq y \leq f(x)\}$$

Property

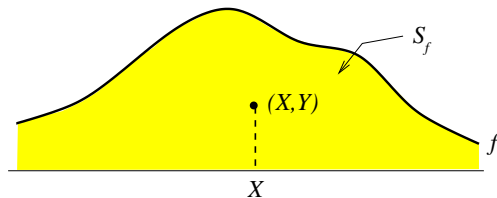
$$(X, Y) \sim \mathcal{U}(S_f) \implies X \sim f$$

Why considering uniform distributions?

Notation

Let f be a **p.d.f.** defined on \mathbb{R}^d and let S_f be its **subgraph**

$$S_f = \{(x, y) \in \mathbb{R}^d \times \mathbb{R} : 0 \leq y \leq f(x)\}$$



Property

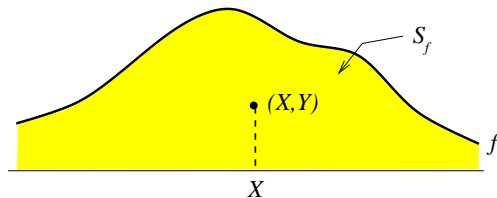
$$(X, Y) \sim \mathcal{U}(S_f) \implies X \sim f$$

Why considering uniform distributions?

Notation

Let f be a **p.d.f.** defined on \mathbb{R}^d and let S_f be its **subgraph**

$$S_f = \{(x, y) \in \mathbb{R}^d \times \mathbb{R} : 0 \leq y \leq f(x)\}$$



Property

$$(X, Y) \sim \mathcal{U}(S_f) \implies X \sim f$$

Rejection method

Problem

Simulate a distribution with probability density function (p.d.f.) f

Context

$f \leq Cg$ where $C > 1$ and g is another p.d.f. that can be simulated

Algorithm

- (i) *generate x from g and u from \mathcal{U}*
- (ii) *if $uCg(x) > f(x)$, goto (i)*
- (iii) *deliver x*

Rejection method

Problem

Simulate a distribution with probability density function (p.d.f.) f

Context

$f \leq Cg$ where $C > 1$ and g is another p.d.f. that can be simulated

Algorithm

- (i) *generate x from g and u from \mathcal{U}*
- (ii) *if $uCg(x) > f(x)$, goto (i)*
- (iii) *deliver x*

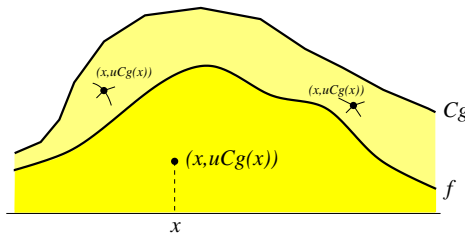
Rejection method

Problem

Simulate a distribution with probability density function (p.d.f.) f

Context

$f \leq Cg$ where $C > 1$ and g is another p.d.f. that can be simulated



Algorithm

- (i) generate x from g and u from \mathcal{U}
- (ii) if $uCg(x) > f(x)$, goto (i)
- (iii) deliver x

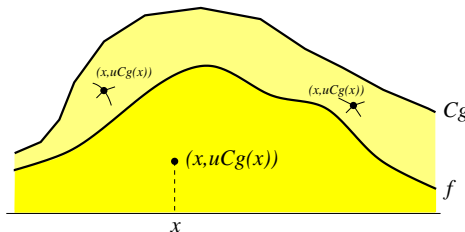
Rejection method

Problem

Simulate a distribution with probability density function (p.d.f.) f

Context

$f \leq Cg$ where $C > 1$ and g is another p.d.f. that can be simulated



Algorithm

- (i) generate x from g and u from \mathcal{U}
- (ii) if $uCg(x) > f(x)$, goto (i)
- (iii) deliver x

Example: the Gamma distribution

Definition

A random variable follows a **gamma distribution** $\mathcal{G}(\alpha)$ with parameter α if its p.d.f. is equal to

$$f(x) = \frac{1}{\Gamma(\alpha)} e^{-x} x^{\alpha-1} \quad x > 0$$

Algorithm

- **Case $\alpha < 1$:** $g(x) = \alpha x^{\alpha-1} e^{-x^\alpha}$, $C = \frac{e^{(1-\alpha)\alpha^{1/(1-\alpha)}}}{\Gamma(\alpha+1)} \leq 3.070$
 - (i) generate x and y from $\mathcal{E}(1)$
 - (ii) if $x + y < (1 - \alpha)e^{\alpha/(1-\alpha)} + x^{1/\alpha}$, then goto (i)
 - (iii) deliver $x^{1/\alpha}$
- **Case $\alpha > 1$:** $g(x) = \frac{1}{\alpha} e^{-x/\alpha}$, $C = \frac{e^{1-\alpha}\alpha^\alpha}{\Gamma(\alpha)}$
 - (i) generate x and y from $\mathcal{E}(1)$
 - (ii) if $y < (\alpha - 1)(x - \ln x - 1)$, then goto (i)
 - (iii) deliver x

Example: the Gamma distribution

Definition

A random variable follows a **gamma distribution** $\mathcal{G}(\alpha)$ with parameter α if its p.d.f. is equal to

$$f(x) = \frac{1}{\Gamma(\alpha)} e^{-x} x^{\alpha-1} \quad x > 0$$

Algorithm

- **Case $\alpha < 1$:** $g(x) = \alpha x^{\alpha-1} e^{-x^\alpha}$, $C = \frac{e^{(1-\alpha)\alpha^{\alpha/(1-\alpha)}}}{\Gamma(\alpha+1)} \leq 3.070$
 - (i) generate x and y from $\mathcal{E}(1)$
 - (ii) if $x + y < (1 - \alpha)e^{\alpha/(1-\alpha)} + x^{1/\alpha}$, then goto (i)
 - (iii) deliver $x^{1/\alpha}$
- **Case $\alpha > 1$:** $g(x) = \frac{1}{\alpha} e^{-x/\alpha}$, $C = \frac{e^{1-\alpha}\alpha^\alpha}{\Gamma(\alpha)}$
 - (i) generate x and y from $\mathcal{E}(1)$
 - (ii) if $y < (\alpha - 1)(x - \ln x - 1)$, then goto (i)
 - (iii) deliver x

Outline

- 1 Uniform generation
 - Uniform generation on a unit segment
 - Uniform generation in a low-dimensional domain
- 2 Random Variable Simulation
 - Inversion method
 - Rejection method
 - Ad hoc methods
- 3 Uniform generation in a high-dimensional domain
 - Presentation of the problem
 - Hit-and-run algorithm
 - Gibbs sampler
 - Metropolis-Hastings algorithm

Gaussian distribution

Definition

A random variable follows a standard **Gaussian** or **normal** distribution \mathcal{N} if its p.d.f. is equal to

$$g(y) = \frac{1}{\sqrt{2\pi}} \exp \left\{ -\frac{y^2}{2} \right\} \quad y \in \mathbb{R}$$

Algorithm (Box and Muller)

- (i) *generate u and v from \mathcal{U}*
- (ii) *return $\sqrt{-2 \ln u} \cos(2\pi v)$*

Remark

Both $\sqrt{-2 \ln U} \cos(2\pi V)$ and $\sqrt{-2 \ln U} \sin(2\pi V)$ are independent and normally distributed

Gaussian distribution

Definition

A random variable follows a standard **Gaussian** or **normal** distribution \mathcal{N} if its p.d.f. is equal to

$$g(y) = \frac{1}{\sqrt{2\pi}} \exp \left\{ -\frac{y^2}{2} \right\} \quad y \in \mathbb{R}$$

Algorithm (Box and Muller)

- (i) *generate u and v from \mathcal{U}*
- (ii) *return $\sqrt{-2 \ln u} \cos(2\pi v)$*

Remark

Both $\sqrt{-2 \ln U} \cos(2\pi V)$ and $\sqrt{-2 \ln U} \sin(2\pi V)$ are independent and normally distributed

Gaussian distribution

Definition

A random variable follows a standard **Gaussian** or **normal** distribution \mathcal{N} if its p.d.f. is equal to

$$g(y) = \frac{1}{\sqrt{2\pi}} \exp \left\{ -\frac{y^2}{2} \right\} \quad y \in \mathbb{R}$$

Algorithm (Box and Muller)

- (i) *generate u and v from \mathcal{U}*
- (ii) *return $\sqrt{-2 \ln u} \cos(2\pi v)$*

Remark

Both $\sqrt{-2 \ln U} \cos(2\pi V)$ and $\sqrt{-2 \ln U} \sin(2\pi V)$ are independent and normally distributed

Poisson distribution

Definition

A random variable N follows a Poisson distribution $\mathcal{P}(\theta)$ with mean θ if its probability mass function (p.m.f.) is equal to

$$P\{N = n\} = e^{-\theta} \frac{\theta^n}{n!} \quad n \in \mathbb{N}$$

Property

If independent $\mathcal{E}(1)$ intervals X_1, X_2, \dots are concatenated starting from 0, then the number of intervals totally contained in $[0, \theta]$ is $\mathcal{P}(\theta)$ distributed

Poisson distribution

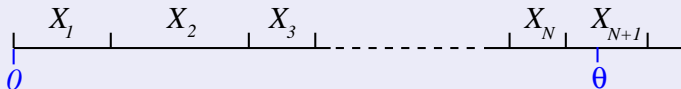
Definition

A random variable N follows a Poisson distribution $\mathcal{P}(\theta)$ with mean θ if its probability mass function (p.m.f.) is equal to

$$P\{N = n\} = e^{-\theta} \frac{\theta^n}{n!} \quad n \in \mathbb{N}$$

Property

If independent $\mathcal{E}(1)$ intervals X_1, X_2, \dots are concatenated starting from 0, then the number of intervals totally contained in $[0, \theta]$ is $\mathcal{P}(\theta)$ distributed



Poisson distribution (2)

Algorithm

- (i) *set $n = 0$ and $A = 1$*
- (ii) *generate u from \mathcal{U} and set $A = Au$*
- (iii) *if $A > e^{-\theta}$, then set $n = n + 1$ and goto ii)*
- (iv) *deliver n .*

Algorithm (case $\theta > 50$)

- (i) *generate y from $\mathcal{N}(\theta, \theta)$*
- (ii) *deliver y rounded off to the nearest integer*

Poisson distribution (2)

Algorithm

- (i) *set $n = 0$ and $A = 1$*
- (ii) *generate u from \mathcal{U} and set $A = Au$*
- (iii) *if $A > e^{-\theta}$, then set $n = n + 1$ and goto ii)*
- (iv) *deliver n .*

Algorithm (case $\theta > 50$)

- (i) *generate y from $\mathcal{N}(\theta, \theta)$*
- (ii) *deliver y rounded off to the nearest integer*

Uniform orientation in \mathbb{R}^d

Definition

A **unit** random vector $U = (U_1, \dots, U_d)$ is uniformly oriented in \mathbb{R}^d if its p.d.f. is **uniform on the d -dimensional unit sphere**

Property

Let $Y = (Y_1, \dots, Y_d)$ a random vector with independent normal components. Then $Y/|Y|$ is uniformly oriented.

Algorithm

- (i) *generate independently y_1, \dots, y_d from \mathcal{N}*
- (ii) *calculate $r = \sqrt{y_1^2 + \dots + y_d^2}$*
- (iii) *deliver $y_1/r, \dots, y_d/r$*

Uniform orientation in \mathbb{R}^d

Definition

A **unit** random vector $U = (U_1, \dots, U_d)$ is uniformly oriented in \mathbb{R}^d if its p.d.f. is **uniform on the d -dimensional unit sphere**

Property

Let $Y = (Y_1, \dots, Y_d)$ a random vector with independent normal components. Then $Y/|Y|$ is uniformly oriented.

Algorithm

- (i) *generate independently y_1, \dots, y_d from \mathcal{N}*
- (ii) *calculate $r = \sqrt{y_1^2 + \dots + y_d^2}$*
- (iii) *deliver $y_1/r, \dots, y_d/r$*

Outline

- 1 Uniform generation
 - Uniform generation on a unit segment
 - Uniform generation in a low-dimensional domain
- 2 Random Variable Simulation
 - Inversion method
 - Rejection method
 - Ad hoc methods
- 3 Uniform generation in a high-dimensional domain
 - Presentation of the problem
 - Hit-and-run algorithm
 - Gibbs sampler
 - Metropolis-Hastings algorithm

Efficiency of the rejection method

Remark

The rejection method becomes less and less efficient as the workspace dimension increases.

Example

Unit ball in \mathbb{R}^d :

The efficiency is $\omega_d/2^d = (\pi/4)^{d/2}/\Gamma(d/2 + 1)$. For $d = 100$, it is around 10^{-70} ...

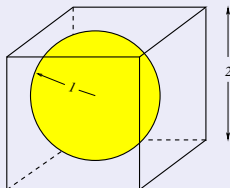
Efficiency of the rejection method

Remark

The rejection method becomes less and less efficient as the workspace dimension increases.

Example

Unit ball in \mathbb{R}^d :

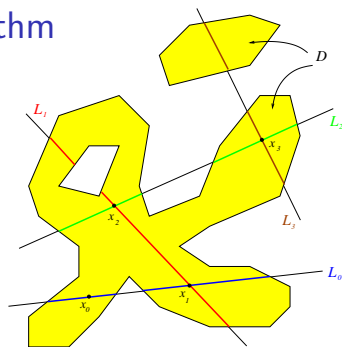


The efficiency is $\omega_d/2^d = (\pi/4)^{d/2}/\Gamma(d/2 + 1)$. For $d = 100$, it is around 10^{-70} ...

Outline

- 1 Uniform generation
 - Uniform generation on a unit segment
 - Uniform generation in a low-dimensional domain
- 2 Random Variable Simulation
 - Inversion method
 - Rejection method
 - Ad hoc methods
- 3 Uniform generation in a high-dimensional domain
 - Presentation of the problem
 - Hit-and-run algorithm
 - Gibbs sampler
 - Metropolis-Hastings algorithm

Hit-and-run algorithm



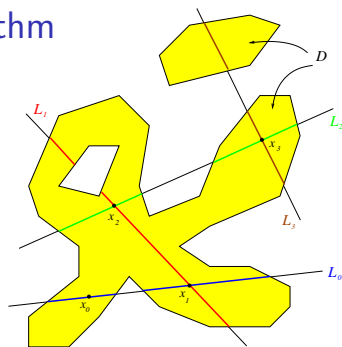
Notation

$$S_{d-1} = \{u \in \mathbb{R}^d : |u| = 1\}$$

Algorithm

- (i) *reset $x \in D$*
- (ii) *generate $u \sim \mathcal{U}(S_{d-1})$ and put $L = x + \mathbb{R}u$*
- (iii) *generate $x \sim \mathcal{U}(D \cap L)$*
- (iv) *goto (ii)*

Hit-and-run algorithm



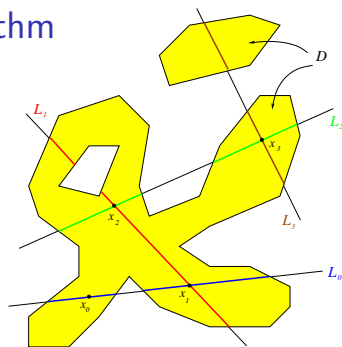
Notation

$$S_{d-1} = \{u \in \mathbb{R}^d : |u| = 1\}$$

Algorithm

- (i) *reset $x \in D$*
- (ii) *generate $u \sim \mathcal{U}(S_{d-1})$ and put $L = x + \mathbb{R}u$*
- (iii) *generate $x \sim \mathcal{U}(D \cap L)$*
- (iv) *goto (ii)*

Hit-and-run algorithm



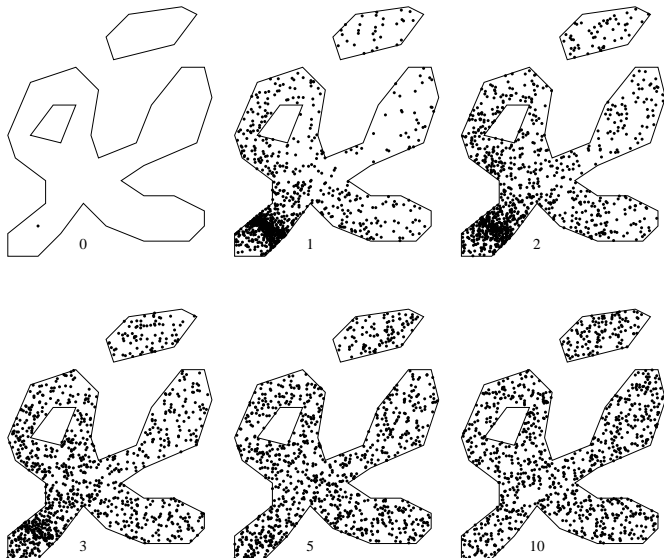
Notation

$$S_{d-1} = \{u \in \mathbb{R}^d : |u| = 1\}$$

Algorithm

- (i) *reset* $x \in D$
- (ii) *generate* $u \sim \mathcal{U}(S_{d-1})$ *and put* $L = x + \mathbb{R}u$
- (iii) *generate* $x \sim \mathcal{U}(D \cap L)$
- (iv) *goto* (ii)

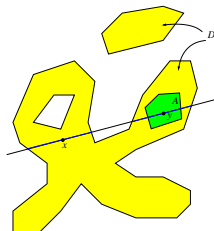
1000 simulations by hit-and-run



Hit-and-run algorithm: Rate of convergence

Definition

The **transition kernel** $P(x, A)$ is the probability of jumping from $x \in D$ to $A \in \mathcal{B}(D)$ in one step.



Result

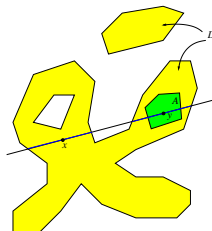
$$P(x, A) = \frac{2}{d\omega_d} \int_A \frac{dy}{\ell(x, y, D)|x - y|^{d-1}}$$

where $\ell(x, y, D)$ is the length of the intersection between D and the line supporting both points x and y .

Hit-and-run algorithm: Rate of convergence

Definition

The **transition kernel** $P(x, A)$ is the probability of jumping from $x \in D$ to $A \in \mathcal{B}(D)$ in one step.



Result

$$P(x, A) = \frac{2}{d\omega_d} \int_A \frac{dy}{\ell(x, y, D)|x - y|^{d-1}}$$

where $\ell(x, y, D)$ is the length of the intersection between D and the line supporting both points x and y .

Hit-and-run algorithm: rate of convergence (2)

Notation

u_D is the uniform pdf on D .

Property (minorization)

If D is **bounded**, then there exists $0 < \varepsilon < 1$ such that

$$P(x, A) \geq \varepsilon u_D(A) \quad x \in D, A \in \mathcal{B}(D)$$

Consequence (uniform ergodicity)

The probability $P^n(x, A)$ of jumping from x to A in n steps satisfies

$$|P^n(x, A) - u_D(A)| < (1 - \varepsilon)^n \quad x \in D, A \in \mathcal{B}(D)$$

Hit-and-run algorithm: rate of convergence (2)

Notation

u_D is the uniform pdf on D .

Property (minorization)

If D is **bounded**, then there exists $0 < \varepsilon < 1$ such that

$$P(x, A) \geq \varepsilon u_D(A) \quad x \in D, A \in \mathcal{B}(D)$$

Consequence (uniform ergodicity)

The probability $P^n(x, A)$ of jumping from x to A in n steps satisfies

$$|P^n(x, A) - u_D(A)| < (1 - \varepsilon)^n \quad x \in D, A \in \mathcal{B}(D)$$

Hit-and-run algorithm: rate of convergence (2)

Notation

u_D is the uniform pdf on D .

Property (minorization)

If D is **bounded**, then there exists $0 < \varepsilon < 1$ such that

$$P(x, A) \geq \varepsilon u_D(A) \quad x \in D, A \in \mathcal{B}(D)$$

Consequence (uniform ergodicity)

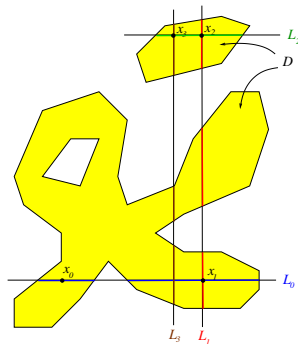
The probability $P^n(x, A)$ of jumping from x to A in n steps satisfies

$$|P^n(x, A) - u_D(A)| < (1 - \varepsilon)^n \quad x \in D, A \in \mathcal{B}(D)$$

Outline

- 1 Uniform generation
 - Uniform generation on a unit segment
 - Uniform generation in a low-dimensional domain
- 2 Random Variable Simulation
 - Inversion method
 - Rejection method
 - Ad hoc methods
- 3 Uniform generation in a high-dimensional domain
 - Presentation of the problem
 - Hit-and-run algorithm
 - Gibbs sampler
 - Metropolis-Hastings algorithm

Gibbs sampler



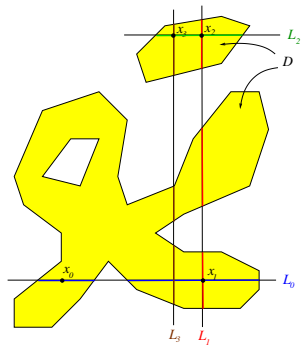
Notation

u_1, \dots, u_d are the unit vectors supported by the coordinate axes.

Algorithm

- (i) reset $x \in D$
- (ii) generate $u \sim \mathcal{U}\{u_1, \dots, u_d\}$ and put $L = x + \mathbb{R}u$
- (iii) generate x from $\mathcal{U}(D \cap L)$
- (iv) goto (ii)

Gibbs sampler



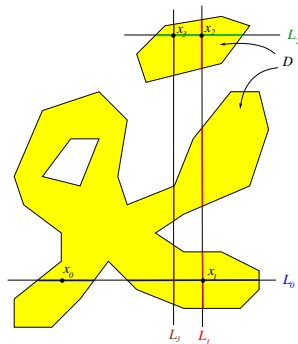
Notation

u_1, \dots, u_d are the unit vectors supported by the coordinate axes.

Algorithm

- (i) reset $x \in D$
- (ii) generate $u \sim \mathcal{U}\{u_1, \dots, u_d\}$ and put $L = x + \mathbb{R}u$
- (iii) generate x from $\mathcal{U}(D \cap L)$
- (iv) goto (ii)

Gibbs sampler



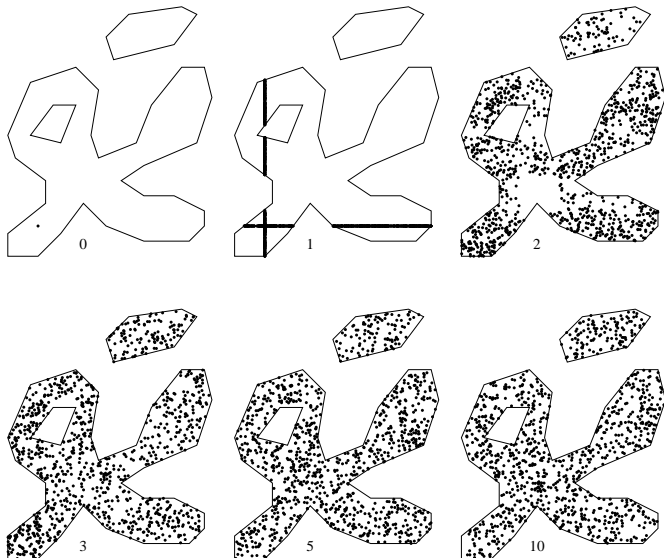
Notation

u_1, \dots, u_d are the unit vectors supported by the coordinate axes.

Algorithm

- (i) reset $x \in D$
- (ii) generate $u \sim \mathcal{U}\{u_1, \dots, u_d\}$ and put $L = x + \mathbb{R}u$
- (iii) generate x from $\mathcal{U}(D \cap L)$
- (iv) goto (ii)

1000 simulations using the Gibbs sampler



Irreducibility problems

Remark

Communication between different connected components of D may not be possible.

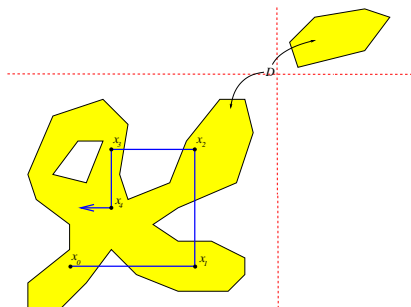
Definition

The transition kernel P is **irreducible** if, for any $x \in D$ and any $A \in \mathcal{B}(D)$ of positive volume, there exists $n > 0$ such that $P^n(x, A) > 0$.

Irreducibility problems

Remark

Communication between different connected components of D may not be possible.



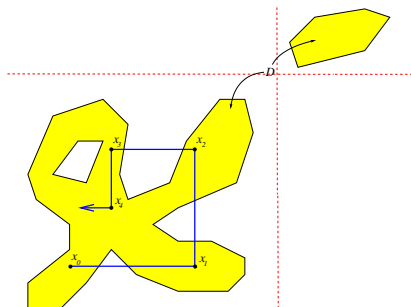
Definition

The transition kernel P is **irreducible** if, for any $x \in D$ and any $A \in \mathcal{B}(D)$ of positive volume, there exists $n > 0$ such that $P^n(x, A) > 0$.

Irreducibility problems

Remark

Communication between different connected components of D may not be possible.



Definition

The transition kernel P is **irreducible** if, for any $x \in D$ and any $A \in \mathcal{B}(D)$ of positive volume, there exists $n > 0$ such that $P^n(x, A) > 0$.

Gibbs sampler: Rate of convergence

Theorem (Roberts and Rosenthal, 1998)

If D is **bounded** and its boundary **twice continuously differentiable**, and if the transition kernel is **irreducible**, then the Gibbs sampler is **uniformly ergodic**, i.e. there exist $M > 0$ and $0 < \epsilon < 1$ such that

$$|P^n(x, A) - u_D(A)| \leq M\epsilon^n \quad x \in D, A \in \mathcal{B}(D).$$

Remark

If the boundary of D is not differentiable, then uniform ergodicity may not necessarily hold.

Example (Vicinity of a vertex)

uniformly ergodic

non uniformly ergodic

Gibbs sampler: Rate of convergence

Theorem (Roberts and Rosenthal, 1998)

If D is **bounded** and its boundary **twice continuously differentiable**, and if the transition kernel is **irreducible**, then the Gibbs sampler is **uniformly ergodic**, i.e. there exist $M > 0$ and $0 < \epsilon < 1$ such that

$$|P^n(x, A) - u_D(A)| \leq M\epsilon^n \quad x \in D, A \in \mathcal{B}(D).$$

Remark

If the boundary of D is not differentiable, then uniform ergodicity may not necessarily hold.

Example (Vicinity of a vertex)

uniformly ergodic

non uniformly ergodic

Gibbs sampler: Rate of convergence

Theorem (Roberts and Rosenthal, 1998)

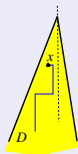
If D is **bounded** and its boundary **twice continuously differentiable**, and if the transition kernel is **irreducible**, then the Gibbs sampler is **uniformly ergodic**, i.e. there exist $M > 0$ and $0 < \epsilon < 1$ such that

$$|P^n(x, A) - u_D(A)| \leq M\epsilon^n \quad x \in D, A \in \mathcal{B}(D).$$

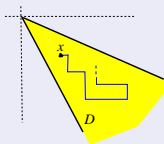
Remark

If the boundary of D is not differentiable, then uniform ergodicity may not necessarily hold.

Example (Vicinity of a vertex)



uniformly ergodic



non uniformly ergodic

Case of an unbounded domain with finite volume

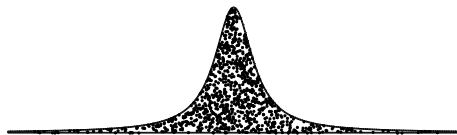
Notation

$$D = \{(x, y) : 0 < y < (1 + x^2)^{-1}\}$$

Case of an unbounded domain with finite volume

Notation

$$D = \{(x, y) : 0 < y < (1 + x^2)^{-1}\}$$

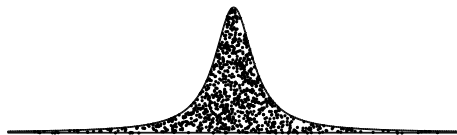


10 iterations

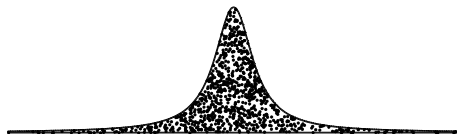
Case of an unbounded domain with finite volume

Notation

$$D = \{(x, y) : 0 < y < (1 + x^2)^{-1}\}$$



10 iterations

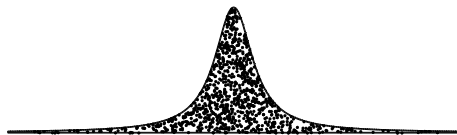


100 iterations

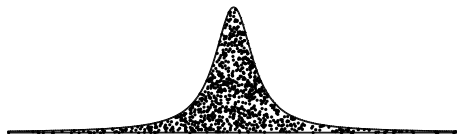
Case of an unbounded domain with finite volume

Notation

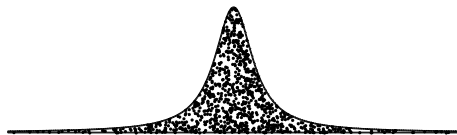
$$D = \{(x, y) : 0 < y < (1 + x^2)^{-1}\}$$



10 iterations



100 iterations



1000 iterations

Case of an unbounded domain with infinite volume

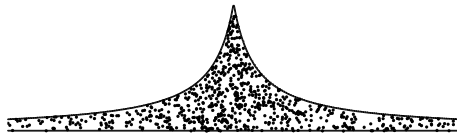
Notation

$$D = \{(x, y) : 0 < y < (1 + |x|)^{-1}\}$$

Case of an unbounded domain with infinite volume

Notation

$$D = \{(x, y) : 0 < y < (1 + |x|)^{-1}\}$$



10 iterations

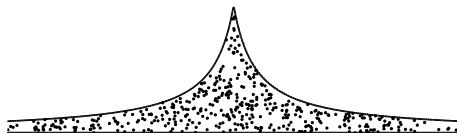
Case of an unbounded domain with infinite volume

Notation

$$D = \{(x, y) : 0 < y < (1 + |x|)^{-1}\}$$



10 iterations

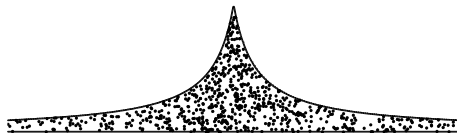


100 iterations

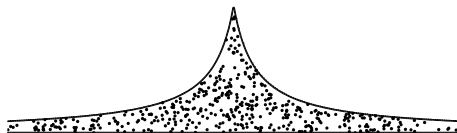
Case of an unbounded domain with infinite volume

Notation

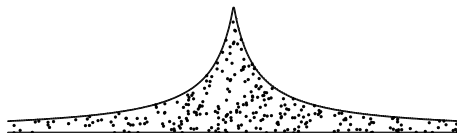
$$D = \{(x, y) : 0 < y < (1 + |x|)^{-1}\}$$



10 iterations



100 iterations



1000 iterations

General Gibbs sampler

Problem

Generate the multivariate pdf $f(x_1, \dots, x_d)$.

Notation

$$x_{1:d} = (x_1, \dots, x_d)$$

$$x_{-i} = x_{1:d \setminus i}$$

Algorithm

- (i) *reset* $x_{1:d}$
- (ii) *generate* $i \sim \mathcal{U}(\{1, \dots, d\})$
- (iii) *generate* $x_i \sim f(\cdot | x_{-i})$
- (iv) *goto* (ii)

General Gibbs sampler

Problem

Generate the multivariate pdf $f(x_1, \dots, x_d)$.

Notation

$$x_{1:d} = (x_1, \dots, x_d)$$

$$x_{-i} = x_{1:d \setminus i}$$

Algorithm

- (i) *reset* $x_{1:d}$
- (ii) *generate* $i \sim \mathcal{U}(\{1, \dots, d\})$
- (iii) *generate* $x_i \sim f(\cdot | x_{-i})$
- (iv) *goto* (ii)

General Gibbs sampler

Problem

Generate the multivariate pdf $f(x_1, \dots, x_d)$.

Notation

$$x_{1:d} = (x_1, \dots, x_d)$$

$$x_{-i} = x_{1:d \setminus i}$$

Algorithm

- (i) *reset* $x_{1:d}$
- (ii) *generate* $i \sim \mathcal{U}(\{1, \dots, d\})$
- (iii) *generate* $x_i \sim f(\cdot | x_{-i})$
- (iv) *goto* (ii)

Outline

- 1 Uniform generation
 - Uniform generation on a unit segment
 - Uniform generation in a low-dimensional domain
- 2 Random Variable Simulation
 - Inversion method
 - Rejection method
 - Ad hoc methods
- 3 Uniform generation in a high-dimensional domain
 - Presentation of the problem
 - Hit-and-run algorithm
 - Gibbs sampler
 - Metropolis-Hastings algorithm

Metropolis-Hastings algorithm

Problem

Simulate a distribution p on some state space, say Ω .

Ingredient and notation

- An **instrumental** transition kernel Q on Ω that is **irreducible**;
- An **acceptance criterion** for each $x, y \in \Omega$

$$\alpha(x, y) = \frac{p(y)Q(y, x)}{p(x)Q(x, y)} \wedge 1 \qquad \alpha(x, y) = \frac{p(y)Q(y, x)}{p(x)Q(x, y) + p(y)Q(y, x)}$$

Algorithm

- reset $x \in \Omega$*
- generate $y \sim Q(x, \cdot)$ and $u \sim \mathcal{U}$*
- if $u < \alpha(x, y)$, then put $x = y$*
- goto (ii)*

Metropolis-Hastings algorithm

Problem

Simulate a distribution p on some state space, say Ω .

Ingredient and notation

- An **instrumental** transition kernel Q on Ω that is **irreducible**;
- An **acceptance criterion** for each $x, y \in \Omega$

$$\alpha(x, y) = \frac{p(y)Q(y, x)}{p(x)Q(x, y)} \wedge 1 \qquad \alpha(x, y) = \frac{p(y)Q(y, x)}{p(x)Q(x, y) + p(y)Q(y, x)}$$

Algorithm

- (i) *reset $x \in \Omega$*
- (ii) *generate $y \sim Q(x, \cdot)$ and $u \sim \mathcal{U}$*
- (iii) *if $u < \alpha(x, y)$, then put $x = y$*
- (iv) *goto (ii)*

Metropolis-Hastings algorithm

Problem

Simulate a distribution p on some state space, say Ω .

Ingredient and notation

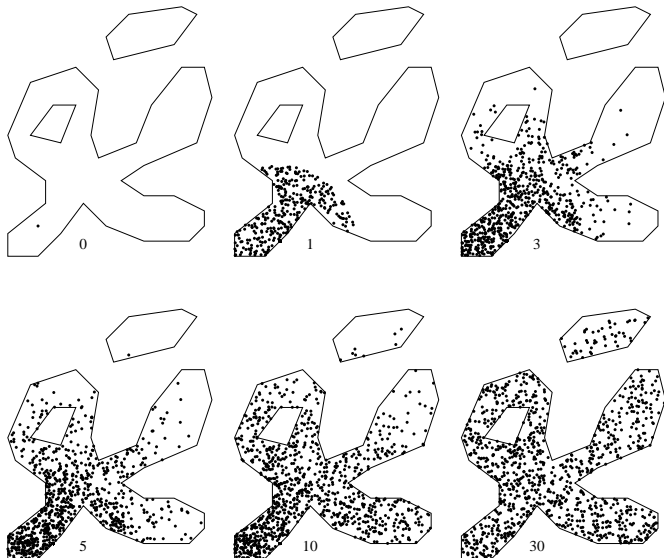
- An **instrumental** transition kernel Q on Ω that is **irreducible**;
- An **acceptance criterion** for each $x, y \in \Omega$

$$\alpha(x, y) = \frac{p(y)Q(y, x)}{p(x)Q(x, y)} \wedge 1 \qquad \alpha(x, y) = \frac{p(y)Q(y, x)}{p(x)Q(x, y) + p(y)Q(y, x)}$$

Algorithm

- (i) *reset* $x \in \Omega$
- (ii) *generate* $y \sim Q(x, \cdot)$ and $u \sim \mathcal{U}$
- (iii) *if* $u < \alpha(x, y)$, *then put* $x = y$
- (iv) *goto* (ii)

1000 simulations using M-H algorithm



Metropolis-Hastings algorithm: why does it work?

Notation

Let $P(x, y)$ the transition kernel induced by the M-H algorithm.

$$P(x, y) = Q(x, y)\alpha(x, y) \quad x \neq y$$

Theorem (Tierney, 1993; Meyn and Tweedie, 1994)

If the kernel P satisfies the following two properties:

- **positivity**: For each $x, y \in \Omega$, there exists $n_0 \in \mathbb{N}$ such that $P^n(x, y) > 0$ for each $n \geq n_0$
- **invariance**: $X_n \sim p$ implies $X_{n+1} \sim p$
then $P^n(x, y) \rightarrow p(y)$ for any $x, y \in \Omega$.

Property

The Metropolis-Hastings kernel is positive and leaves p invariant.

Metropolis-Hastings algorithm: why does it work?

Notation

Let $P(x, y)$ the transition kernel induced by the M-H algorithm.

$$P(x, y) = Q(x, y)\alpha(x, y) \quad x \neq y$$

Theorem (Tierney, 1993; Meyn and Tweedie, 1994)

If the kernel P satisfies the following two properties:

- **positivity**: For each $x, y \in \Omega$, there exists $n_0 \in \mathbb{N}$ such that $P^n(x, y) > 0$ for each $n \geq n_0$
 - **invariance**: $X_n \sim p$ implies $X_{n+1} \sim p$
- then $P^n(x, y) \rightarrow p(y)$ for any $x, y \in \Omega$.

Property

The Metropolis-Hastings kernel is positive and leaves p invariant.

P

Metropolis-Hastings algorithm: why does it work?

Notation

Let $P(x, y)$ the transition kernel induced by the M-H algorithm.

$$P(x, y) = Q(x, y)\alpha(x, y) \quad x \neq y$$

Theorem (Tierney, 1993; Meyn and Tweedie, 1994)

If the kernel P satisfies the following two properties:

- **positivity**: For each $x, y \in \Omega$, there exists $n_0 \in \mathbb{N}$ such that $P^n(x, y) > 0$ for each $n \geq n_0$
 - **invariance**: $X_n \sim p$ implies $X_{n+1} \sim p$
- then $P^n(x, y) \rightarrow p(y)$ for any $x, y \in \Omega$.

Property

The Metropolis-Hastings kernel is positive and leaves p invariant.

Simulation of a Poisson distribution

Definition

$$p(x) = \exp\{-\theta\} \frac{\theta^x}{x!} \quad x \in \mathbb{N}$$

Instrumental transition kernel Q

Transition kernel P

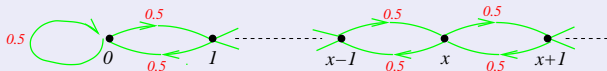
$$P(x, y) = \begin{cases} \frac{x}{\theta + x} & \text{if } y = x - 1 \\ \frac{\theta}{(\theta + x)(\theta + x + 1)} & \text{if } y = x \\ \frac{\theta}{\theta + x + 1} & \text{if } y = x + 1 \end{cases}$$

Simulation of a Poisson distribution

Definition

$$p(x) = \exp\{-\theta\} \frac{\theta^x}{x!} \quad x \in \mathbb{N}$$

Instrumental transition kernel Q



Transition kernel P

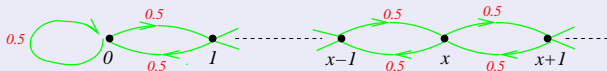
$$P(x, y) = \begin{cases} \frac{x}{\theta + x} & \text{if } y = x - 1 \\ \frac{\theta}{(\theta + x)(\theta + x + 1)} & \text{if } y = x \\ \frac{\theta}{\theta + x + 1} & \text{if } y = x + 1 \end{cases}$$

Simulation of a Poisson distribution

Definition

$$p(x) = \exp\{-\theta\} \frac{\theta^x}{x!} \quad x \in \mathbb{N}$$

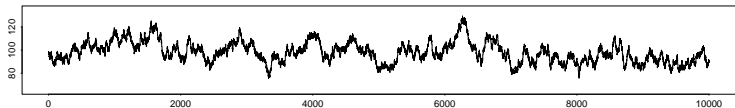
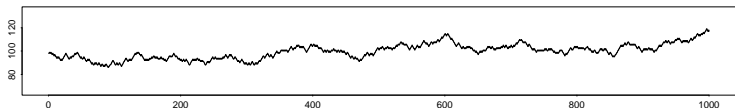
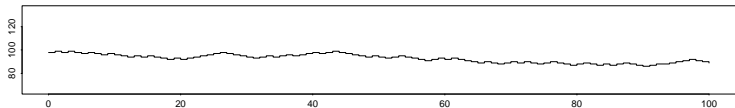
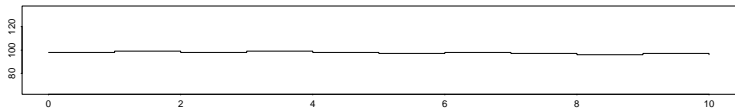
Instrumental transition kernel Q



Transition kernel P

$$P(x, y) = \begin{cases} \frac{x}{\theta + x} & \text{if } y = x - 1 \\ \frac{\theta}{(\theta + x)(\theta + x + 1)} & \text{if } y = x \\ \frac{\theta}{\theta + x + 1} & \text{if } y = x + 1 \end{cases}$$

Simulation of a Poisson distribution with mean 100



Simulation of a Poisson distribution: Rate of convergence

Property

The rate of convergence is governed by the **second largest eigenvalue** of P

Property

The eigenvalues of P are $\lambda_k = \text{sign}(k) \sqrt{\frac{\theta}{\theta + |k|}}$ for each $k \in \mathbb{Z}$.

Consequence

$$|P^n(x, y) - p(y)| = O\left(\frac{\theta}{\theta + 1}\right)^{\frac{n}{2}} \quad n \text{ large}$$

Simulation of a Poisson distribution: Rate of convergence

Property

The rate of convergence is governed by the **second largest eigenvalue** of P

Property

The eigenvalues of P are $\lambda_k = \text{sign}(k) \sqrt{\frac{\theta}{\theta + |k|}}$ for each $k \in \mathbb{Z}$.

Consequence

$$|P^n(x, y) - p(y)| = O\left(\frac{\theta}{\theta + 1}\right)^{\frac{n}{2}} \quad n \text{ large}$$

Simulation of a Poisson distribution: Rate of convergence

Property

The rate of convergence is governed by the **second largest eigenvalue** of P

Property

The eigenvalues of P are $\lambda_k = \text{sign}(k) \sqrt{\frac{\theta}{\theta + |k|}}$ for each $k \in \mathbb{Z}$.

Consequence

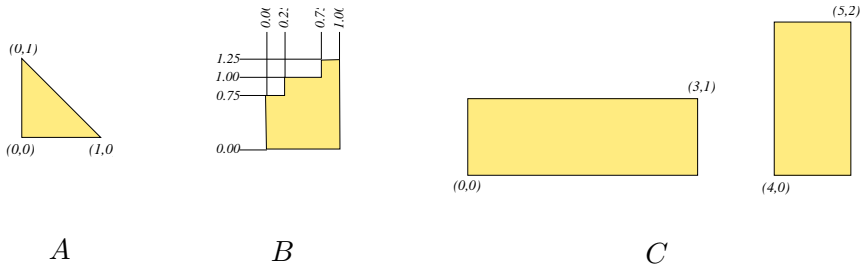
$$|P^n(x, y) - p(y)| = O\left(\frac{\theta}{\theta + 1}\right)^{\frac{n}{2}} \quad n \text{ large}$$

References

- [Hastings W.K.](#) (1970) – "Monte-Carlo sampling methods using Markov chains and their applications". Biometrika, Vol. 57, pp. 97-109.
- [Lantuéjoul C.](#) (2002) – Geostatistical simulation; Models and algorithms. Springer, Berlin.
- [Lehmer D.H.](#) (1949) – Mathematical methods in large scale computing machinery. Harvard University Press (Cambridge).
- [Metropolis N. et al.](#) (1953) – "Equations of state calculations by fast computing machines". The J. of Chem. Physics, Vol. 21-6, pp. 1087-1092.
- [Ripley B.D.](#) (1986) - Statistical simulation. Wiley and Sons (Chichester).
- [Meyn S.P. and Tweedie R.L.](#) (1993) – Markov chains and stochastic stability. Springer (London).
- [Robert C.P. and Casella G.](#) (2004) – Monte Carlo statistical methods. Springer (New York).

Exercise

Write down an algorithm for simulating a uniform point in the following three domains



Compute 100 simulations for each and plot the results

Exercise: Simulation of a Gaussian distribution

Let X be a standardized random normal variable (zero mean and unit variance). Its p.d.f. is

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right), \quad x \in \mathbb{R}$$

- 1 Show that $f(x) \leq C \exp(-|x|)$ with $C = \sqrt{\frac{e}{2\pi}}$
- 2 Consider $g(x) = \frac{1}{2} \exp(-|x|)$, $x \in \mathbb{R}$. Show that g is a p.d.f.
- 3 Therefore, if $X \sim g$, then $|X| \sim \mathcal{E}(1)$. Deduce from this an algorithm for simulating g
- 4 Write down the rejection algorithm for simulating f

Metropolis-Hastings algorithm: why does it work?

$$P(x, y) = Q(x, y)\alpha(x, y) = Q(x, y) \frac{p(y)Q(y, x)}{p(x)Q(x, y) + p(y)Q(y, x)} \quad x \neq y$$

Proof (positivity)

P is positive because Q is irreducible and $\alpha(x, y) > 0$

Proof (invariance)

Note first the **reversibility** property

$$p(x)P(x, y) = \frac{p(x)Q(x, y)p(y)Q(y, x)}{p(x)Q(x, y) + p(y)Q(y, x)} = p(y)P(y, x)$$

from which the invariance is derived at once

$$\sum_x p(x)P(x, y) = \sum_x p(y)P(y, x) = p(y) \sum_x P(y, x) = p(y)$$

Metropolis-Hastings algorithm: why does it work?

$$P(x, y) = Q(x, y)\alpha(x, y) = Q(x, y) \frac{p(y)Q(y, x)}{p(x)Q(x, y) + p(y)Q(y, x)} \quad x \neq y$$

Proof (positivity)

P is positive because Q is irreducible and $\alpha(x, y) > 0$

Proof (invariance)

Note first the **reversibility** property

$$p(x)P(x, y) = \frac{p(x)Q(x, y)p(y)Q(y, x)}{p(x)Q(x, y) + p(y)Q(y, x)} = p(y)P(y, x)$$

from which the invariance is derived at once

$$\sum_x p(x)P(x, y) = \sum_x p(y)P(y, x) = p(y) \sum_x P(y, x) = p(y)$$

Metropolis-Hastings algorithm: why does it work?

$$P(x, y) = Q(x, y)\alpha(x, y) = Q(x, y) \frac{p(y)Q(y, x)}{p(x)Q(x, y) + p(y)Q(y, x)} \quad x \neq y$$

Proof (positivity)

P is positive because Q is irreducible and $\alpha(x, y) > 0$

Proof (invariance)

Note first the **reversibility** property

$$p(x)P(x, y) = \frac{p(x)Q(x, y)p(y)Q(y, x)}{p(x)Q(x, y) + p(y)Q(y, x)} = p(y)P(y, x)$$

from which the invariance is derived at once

$$\sum_x p(x)P(x, y) = \sum_x p(y)P(y, x) = p(y) \sum_x P(y, x) = p(y)$$