Stochastic Simulation of Processes, Fields and Structures

Ulm University
Institute of Stochastics

Lecture Notes
Dr. Tim Brereton
Summer Term 2014

Ulm, 2014

Contents

Chapter 1

Random Walks, Estimators and Markov Chains

1.1 Stochastic Processes

To begin, we need to define the basic objects we will be learning about.

Definition 1.1.1 (Stochastic Process)**.** A stochastic process is a set of random variables $\{X_i\}_{i \in I}$, taking values in a *state space* $\mathcal{X}$, with index sex $I \subset \mathbb{R}$.

In general, i represents a point in time. However, it could represent a point in 1D space as well.

We will say a process is *discrete time* if I is discrete. For example, I could be $\mathbb{N}$ or $\{1, 2, 3, 10, 20\}$. Normally, when we talk about discrete time stochastic processes, we will use the index n (e.g., $\{X_n\}_{n \in \mathbb{N}}$).

We will say a process is *continuous time* if I is an interval. For example $[0, \infty)$ or $[1, 2]$. Normally, when we talk about continuous time stochastic processes, we will use the index t (e.g. $\{X_t\}_{t \in [0,T]}$).

1.2 Random Walks

1.2.1 Bernoulli Processes

One of the simplest stochastic processes is a *random walk*. However, even though the random walk is very simple, it has a number of properties that will be important when we think about more complicated processes. To define a random walk, we begin with an even simpler process called a *Bernoulli process*. A Bernoulli process is a discrete time process taking values in the state space $\{0, 1\}$.

Definition 1.2.1 (Bernoulli Process)**.** A Bernoulli process with parameter $p \in [0,1]$ is a sequence of *independent and identically distributed* (i.i.d.) random variables, $\{Y_n\}_{n\geq 1}$, such that

$$\mathbb{P}(Y_1 = 1) = 1 - \mathbb{P}(Y_1 = 0) = p. \tag{1.1}$$

A variable with the *probability mass function* (pmf) described by (1.1) is called a Bernoulli random variable with distribution $\mathsf{Ber}(p)$.

In the case where $p = 1/2$, you can think of a Bernoulli process as a sequence of fair coin tosses. In the case where $p \neq 1/2$, you can think of a Bernoulli process as a sequence of tosses of an unfair coin.

Note that if $U \sim \mathsf{U}(0,1)$, then $\mathbb{P}(U \leq p) = p$ for $p \in [0,1]$. This means if we define the random variable $Y = \mathbb{I}(U \leq p)$, where $\mathbb{I}(\cdot)$ is the indicator function and $U \sim \mathsf{U}(0,1)$, we have

$$\mathbb{P}(Y = 1) = 1 - \mathbb{P}(Y = 0) = p.$$

In Matlab, we can make these variables as follows.

Listing 1.1: Generating a Bernoulli Random Variable

```
Y = (rand <= p)
```

If we want to make a realization of the first n steps of a Bernoulli process, we simply make n such variables. One way to do this is the following.

Listing 1.2: Generating a Bernoulli Process 1

```
n = 20; p = 0.6; Y = zeros(n,1);

for i = 1:n
  Y(i) = (rand <= p);
end
```

We can do this more quickly in Matlab though.

Listing 1.3: Generating a Bernoulli Process 2

```
n = 20; p = 0.6;

Y = (rand(n,1) <= p);
```

It is important to be able to visualize stochastic objects. One way to represent / visualize a Bernoulli process is to put $n = 1, 2, \ldots$ on the x-axis and the values of Y_n on the y-axis. I draw a line (almost of length 1) at each value of Y_n, as this is easier to see than dots. Figure 1.2.1 shows a realization of the first 20 steps of a Bernoulli process.

Listing 1.4: Generating and Visualizing a Bernoulli Process

```
n = 20; p = 0.6;

Y = (rand(n,1) <= p);

clf;
axis([1 n+1 -0.5 1.5]);
for i = 1:n
    line([i, i+.9],[Y(i) Y(i)]);
end
```

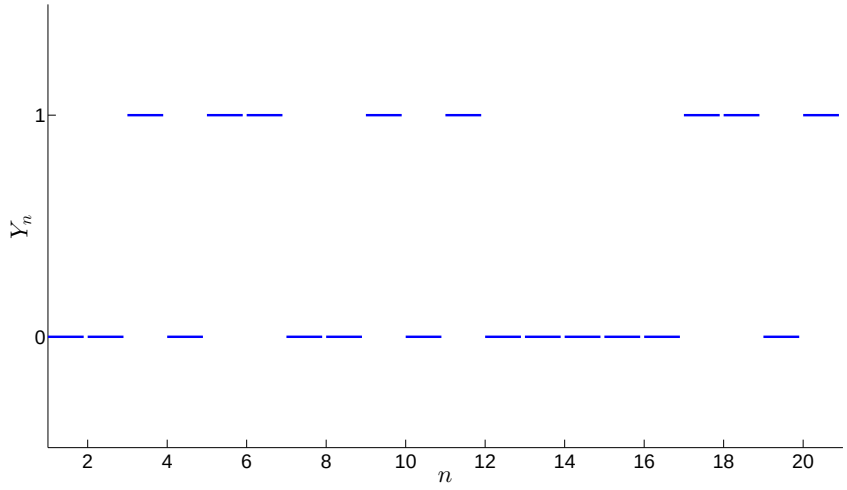

Figure 1.2.1: A realization of the first 20 steps of a Bernoulli process.

Now, it is easy to write out the probability of seeing a particular realization of n steps of a Bernoulli process. This is given by

$$\mathbb{P}(Y_n = y_n, Y_{n-1} = y_{n-1}, \ldots, Y_1 = y_1) = p^{N_U}(1-p)^{n-N_U}.$$

where N_U is the number of times the Bernoulli process takes the value 1. More technically, we define N_U as

$$N_U = \#\{0 \leq i \leq n : y_i = 1\},$$

where $\#$ denotes the cardinality of the set.

1.2.2 Random Walks

Now that we can generate Bernoulli processes, we are ready to consider our main object of interest in this chapter: the random walk. The random walk is a discrete time random process taking values in the state space $\mathbb{Z}$.

Definition 1.2.2 (Random Walk)**.** Given a Bernoulli process $\{Y_n\}_{n\geq 1}$ with parameter p, we define a random walk $\{X_n\}_{n\geq 0}$ with parameter p and initial condition $X_0 = x_0$ by

$$X_{n+1} = X_n + (2Y_{n+1} - 1).$$

Note that $(2Y_n - 1)$ is 1 if $Y_n = 1$ and -1 if $Y_n = 0$. So, essentially, at each step of the process, it goes up or down by 1.

There are lots of ways to think about random walks. One way is in terms of gambling (which is a classical setting for probability). Think of a game which is free to enter. The person running the game flips a coin. With probability p, the coin shows heads and you win €1. With probability $1-p$, you lose €1. If you start with €x_0, and assuming you can go into debt, X_n is the random variable describing how much money you have after n games.

Given that we know how to generate a Bernoulli process, it is straightforward to generate a random walk.

Listing 1.5: Generating a Random Walk 1

```
n = 100; X = zeros(n,1);
p = 0.5; X_0 = 0;

Y = (rand(n,1) <= p);
X(1) = X_0 + 2*Y(1) - 1;
for i = 2:n
    X(i) = X(i-1) + 2*Y(i) - 1;
end
```

A more compact way is as follows.

Listing 1.6: Generating a Random Walk 2

```
n = 100; X = zeros(n,1);
p = 0.5; X_0 = 0;

Y = (rand(n,1) <= p);
X = X_0 + cumsum((2*Y - 1));
```

It is good to be able to plot these. Here, we have at least two options. One is to draw a line for each value of X_n, as we did for the Bernoulli process.

Because the value of $\{X_n\}_{n\geq 0}$ changes at every value of n, we can draw the lines to be length 1. This approach is nice, because it reinforces the idea that $\{X_n\}_{n\geq 0}$ jumps at each value of n (this is made obvious by the gaps between the lines).

Listing 1.7: Plotting a Random Walk 1

```
clf
axis([1 n+1 min(X)-1 max(X)+1]);
for i = 1:n
    line([i,i+1],[X(i) X(i)],'LineWidth',3);
end
```

For larger values of n, it is help to draw vertical lines as well as horizontal lines (otherwise, things get a bit hard to see). We can do this as follows.

Listing 1.8: Plotting a Random Walk 2

```
clf
stairs(X);
axis([1 n+1 min(X)-1 max(X)+1]);
```

Below are a number of realizations of random walks. Figures 1.2.2 and 1.2.3 are generated using lines. Figures 1.2.4 and 1.2.5 are generated using the 'stairs' function.

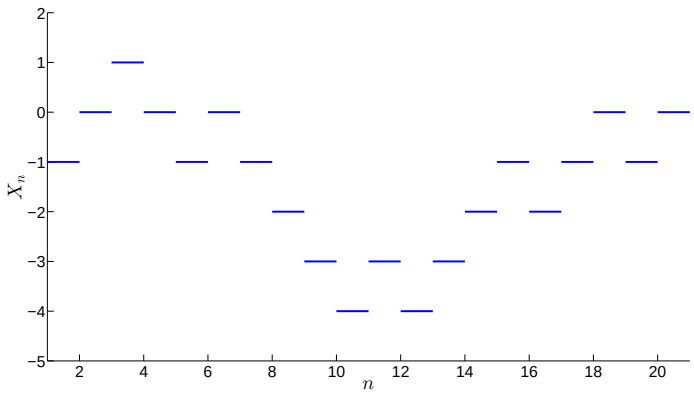

Figure 1.2.2: A realization of the first 20 steps of a random walk with $p = 0.5$.

Note how, as the number of steps simulated gets bigger, the sample paths look wilder and wilder.

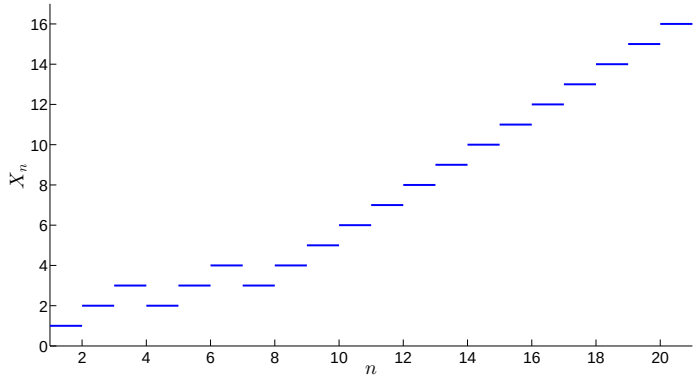

Figure 1.2.3: A realization of the first 20 steps of a random walk with $p = 0.9$.

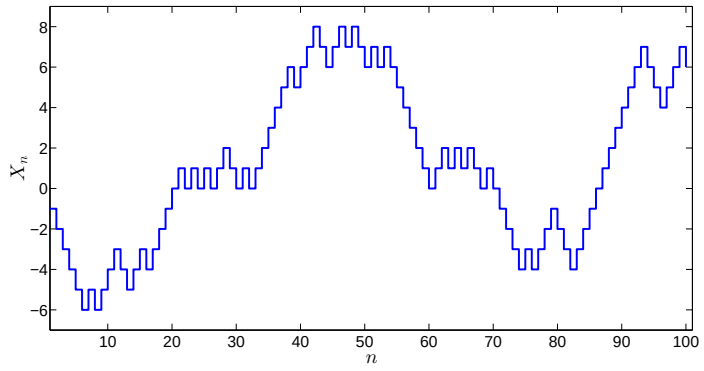

Figure 1.2.4: A realization of the first 100 steps of a random walk with $p = 0.5$.

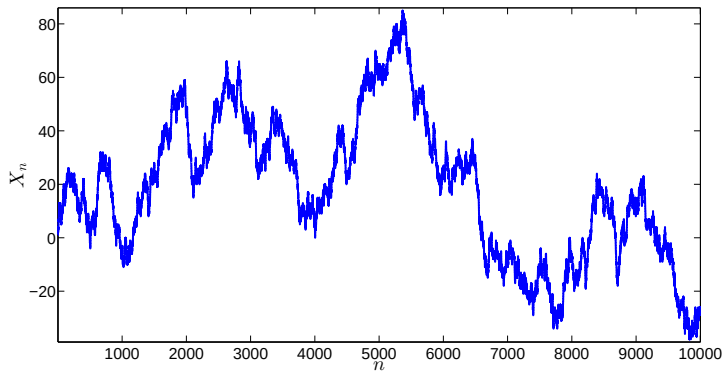

Figure 1.2.5: A realization of the first 10000 steps of a random walk with $p = 0.5$.

1.2.3 Probabilities of Random Walks

A random walk is a special kind of process, called a *Markov process*. Discrete time discrete state space Markov processes (called *Markov chains*) are defined by the following property.

Definition 1.2.3 (Markov Property: Discrete Time Discrete State Space Version)**.** We say a discrete time, discrete state space stochastic process, $\{X_n\}_{n\geq 0}$, has the Markov property if

$$\mathbb{P}(X_{n+1} = x_{n+1} \,|\, X_n = x_n, \ldots, X_0 = x_0) = \mathbb{P}(X_{n+1} = x_{n+1} \,|\, X_n = x_n),$$

for all $n \geq 0$ and $x_0, \ldots, x_n \in \mathcal{X}$.

If we think about how a random walk can move after n steps (assuming we know the values of the first n steps), we have

$$\mathbb{P}(X_{n+1} = x_{n+1} \,|\, X_n = x_n, \ldots, X_0 = x_0) = \begin{cases} p & \text{if } x_{n+1} = x_n + 1, \\ 1-p & \text{if } x_{n+1} = x_n - 1, \\ 0 & \text{otherwise.} \end{cases}$$

Note that these probabilities do not depend on any value other than X_n. That is, it is always the case that $\mathbb{P}(X_{n+1} = x_{n+1} \,|\, X_n = x_n, \ldots, X_0 = x_0) = \mathbb{P}(X_{n+1} = x_{n+1} \,|\, X_n = x_n)$, so $\{X_n\}_{n\geq 0}$ is a Markov chain.

Using this result, we can assign probabilities to various paths of the random walk. That is, we can write

$$\begin{aligned} &\mathbb{P}(X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1) \\ &= \mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1}, \ldots, X_1 = x_1) \cdots \mathbb{P}(X_2 = x_2 | X_1 = x_1)\mathbb{P}(X_1 = x_1) \\ &= \mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1}) \cdots \mathbb{P}(X_2 = x_2 | X_1 = x_1)\mathbb{P}(X_1 = x_1) \end{aligned}$$

Now, each of these probabilities is either p or $1-p$. So, we can write

$$\mathbb{P}(X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1) = p^{N_U}(1-p)^{n-N_U}.$$

where N_U is the number of times the random walk goes up. More precisely,

$$N_U = \#\{1 \leq i \leq n : x_i - x_{i-1} = 1\}$$

1.2.4 Distribution of X_n

One thing we might be interested in is the distribution of X_n, given we only know the initial value of the random walk, x_0. We can make this a bit

simpler by thinking about the distribution of $X_n - x_0$ (this always starts at 0). Notice that X_n cannot be more than n steps from x_0. That is, $|X_n - x_0| \leq n$. Also, if x_0 is even then X_n must be even if n is even and odd if n is odd. If x_0 is odd, it is the other way around.

Now, $X_n - x_0 = n$ can only happen if all n of the steps are up. There is only one possible path for which this is true. This path has probability p^n. Likewise, $X_n - x_0 = n-2$ can only happen if all but one of the steps are up. Any path for which this is true has probability $p^{n-1}(1-p)^1$. However, there is more than one path that has $n-1$ up steps and 1 down step (the first step could be the down one or the second step could be the down one, etc.). In fact, there are $\binom{n}{n-1}$ possible paths. Continuing with this pattern, we have

$$\mathbb{P}(X_n - x_0 = x) = \begin{cases} \binom{n}{(n+x)/2} p^{(n+x)/2}(1-p)^{(n-x)/2} & \text{if } |x| \leq n \text{ and } x \text{ have the same parity as } n \\ 0 & \text{otherwise.} \end{cases}$$

Having the same parity means both n and x are even or both n and x are odd.

1.2.5 First Passage Time

A quantity we are often very interested in is the *first passage time* of $\{X_n\}_{n\geq 0}$ into the set A.

Definition 1.2.4 (First Passage Time: Discrete Time Version)**.** Given a stochastic process $\{X_n\}_{n\geq 0}$ we define the first passage time of $\{X_n\}_{n\geq 0}$ into the set A by

$$\tau_A = \begin{cases} \inf\{n \geq 0 : X_n \in A\} & \text{if } X_n \in A \text{ for some } n \geq 0 \\ \infty & \text{if } X_n \notin A \text{ for all } n \geq 0 \end{cases}.$$

Note that a first passage time is a random variable.

In general, for random walks, we want to calculate the first time the process hits a certain level (say 10). In this case, $A = \{10, 11, \ldots\}$. In a random walk context, a first passage time could correspond to the first time a queue gets too long, or (in a very simplistic model) the first time a stock price hits a certain level.

It is possible to work out the distribution of the first passage time using a beautiful and simple piece of mathematics called the *reflection principle*. However, I will just give you the result.

Lemma 1.2.5 (First Passage Time Distribution for a Random Walk)**.** The distribution of the first passage time for a random walk, $\{X_n\}_{n\geq 0}$ with $x_0 = 0$, where $a \neq 0$ and $A = \{a, a+1, \ldots\}$ if $a > 0$ or $A = \{a, a-1, a-2, \ldots\}$ if $a < 0$, is given by

$$\mathbb{P}(\tau_A = n) = \begin{cases} \frac{|a|}{n}\mathbb{P}(X_n = a) & \text{if } a \text{ and } n \text{ have the same parity and } n \geq |a| \\ 0 & \text{otherwise} \end{cases}.$$

Working out first passage time probabilities for a random walk hitting a with $x_0 \neq 0$ is the same as working out first passage time probabilities for a random walk starting from 0 hitting $a - x_0$.

Example 1.2.6 (Probability a random walk returns to 0 in 4 steps)**.** We can make everything a bit clearer with an example. What is the probability a random walker returns to zero in 4 steps? Our formula for first passage time probabilities only helps with levels other than 0. However, we can make it work by realising that, at the first step, the random walk must step up to 1 (with probability p) or step down to -1 (with probability $1-p$). Now, the probability a random walk starting at 1 first hits 0 in 3 steps is the same as the probability a random walk starting at 0 first hits -1 in 3 steps and the probability a random walk starting at -1 first hits 0 in 3 steps is the same as the probability a random walk starting at 0 first hits 1 in 3 steps. So our answer is of the form

$$\begin{aligned} &p\,\mathbb{P}\left(\tau_{\{-1,-2,\ldots\}} = 3\right) + (1-p)\mathbb{P}\left(\tau_{\{1,2,\ldots\}} = 3\right) \\ &= p\frac{1}{3}\mathbb{P}(X_3 = -1) + (1-p)\frac{1}{3}\mathbb{P}(X_3 = 1) \\ &= p\frac{1}{3}\binom{3}{1}p^1(1-p)^2 + (1-p)\frac{1}{3}\binom{3}{2}p^2(1-p)^1 \\ &= p^2(1-p)^2 + (1-p)^2p^2 = 2p^2(1-p)^2 \end{aligned}$$

1.3 Properties of Estimators

When we generate stochastic processes, we are usually interested in calculating actual numbers. These numbers are almost always in the form of an expectation. Sometimes we just want to know $\mathbb{E}X_n$ for some value of n but usually it is something slightly more complicated. Often we interested in a probability, for example $\mathbb{P}(X_5 > 10)$ or $\mathbb{P}(\tau_A < 10)$. A probability can be written as an expectation using the indicator function. For example $\mathbb{P}(X_5 > 3) = \mathbb{E}\mathbb{I}(X_5 > 3)$ and $\mathbb{P}(\tau_A < 10) = \mathbb{E}\mathbb{I}(\tau_A < 10)$.

In general, we will not know how to calculate the expectations we are interested in explicitly. In the first few weeks we will practice by calculating expectations of things we know exactly, but soon the objects we are considering will become too complicated for us to do this. We will need to estimate things.

Monte Carlo methods are based on one very important result in probability theory: the *strong law of large numbers*. In simple terms, this says we can estimate an expected value using a sample (or empirical) average.

Theorem 1.3.1 (Strong Law of Large Numbers)**.**
Let $\{X^{(i)}\}_{i\geq 0}$ be a sequence of i.i.d. random variables with values in $\mathbb{R}$ such that $\mathbb{E}|X^{(1)}| \leq \infty$. Let S_N, $N > 0$, be the sample average defined by

$$S_N = \frac{1}{N}\sum_{i=1}^{N} X^{(i)}.$$

Then,

$$\lim_{N\to\infty} S_N = \mathbb{E}X^{(1)} \text{ almost surely.}$$

This means we can estimate $\ell = \mathbb{E}X$ by simulating a sequence of random variables with the same distribution as X and taking their average value. That is, if we can produce a sequence of i.i.d. random variables $\left\{X^{(i)}\right\}_{i=1}^{N}$ with the same distribution as X, then the *estimator*

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N} X^{(i)},$$

is approximately equal to ℓ for large enough N.

Example 1.3.2 (Estimating $\mathbb{P}(X_5 > 3)$ for a random walk)**.** To estimate $\ell = \mathbb{P}(X_5 > 3) = \mathbb{E}\mathbb{I}(X_5 > 3)$ we can simulate N random walks until the 5th step. We then look at the N values at the 5th step, $X_5^{(1)}, \ldots, X_5^{(N)}$, and check how many of them have a value bigger than 3. Our estimator is

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(X_5^{(i)} > 3).$$

We can easily do this in Matlab.

Listing 1.9: Estimating $\mathbb{P}(X_5 > 3)$

```
N = 10^3;
n = 5; p = 0.5; X_0 = 0;
```

```
3
4  X = zeros(N,5);
5
6  Y = (rand(N,n) <= p);
7  X = X_0 + cumsum((2*Y - 1),2);
8  X_5 = X(:,5);
9
10 ell_est = mean(X_5 > 3)
```

In this simple example, we can calculate the probability exactly. It is only possible for X_5 to be greater than 3 if all the steps of the random walk are up. That is, $\ell = \mathbb{P}(X_5 > 3) = p^5$. In the case where $p = 0.5$, $\ell = 0.03125$. We can look at example output from our estimator: for $N = 10^1$, we have $\widehat{\ell} = 0$; for $N = 10^2$, we have $\widehat{\ell} = 0.04$; for $N = 10^3$, we have $\widehat{\ell} = 0.026$; for $N = 10^4$, we have $\widehat{\ell} = 0.0343$; and so on. On average, our estimator will become more accurate each time we increase N.

Example 1.3.3 (Estimating $\mathbb{E}\tau_{\{4,5,\ldots\}}$)**.** We can estimate expected hitting times as well. For example, we might want to estimate $\mathbb{E}\tau_{\{4,5,\ldots\}}$ for a random walk with $x_0 = 0$. We should do this for a random walk with $p > (1-p)$, so we can be certain the random walk hits 4. In Matlab, we would do the following.

Listing 1.10: Estimating $\mathbb{E}\tau_{\{4,5,\ldots\}}$

```
1  N = 10^4;
2  n = 5; p = 0.8; X_0 = 0;
3
4  tau_4 = zeros(N,1);
5
6  for i = 1:N
7      X = X_0; n = 0;
8      while(X ~= 4)
9          X = X + 2*(rand <= p) - 1;
10         n = n + 1;
11     end
12     tau_4(i) = n;
13 end
14
15 est_expec_tau_4 = mean(tau_4)
```

Notice that we have to use a 'while' loop instead of a 'for' loop, as we do not know how long we will have to simulate $\{X_n\}_{n\geq 0}$ for.

1.3.1 Bias, Variance, the Central Limit Theorem and Mean Square Error

The strong law of large numbers tells that, if we use an infinite sample, a Monte Carlo estimator will give us the right answer (with probability 1). This is not really a practical result. In practice, we can only run a computer for a finite amount of time. What we are really interested in are the properties of our estimator for a fixed N.

It is comforting if our estimator on average gives the right answer. The *bias* of an estimator is the amount by which, on average, it deviates from the value it is supposed to estimate.

Definition 1.3.4 (Bias)**.** The bias of an estimator, $\widehat{\ell}$, for the value ℓ is given by

$$\mathsf{Bias}(\widehat{\ell}) = \mathbb{E}[\widehat{\ell} - \ell] = \mathbb{E}\widehat{\ell} - \ell.$$

An estimator without bias is called *unbiased*.

Definition 1.3.5 (Unbiased Estimator)**.** We say an estimator, $\widehat{\ell}$, of ℓ is unbiased if $\mathsf{Bias}(\widehat{\ell}) = 0$ or, equivalently,

$$\mathbb{E}\widehat{\ell} = \ell.$$

Many Monte Carlo estimators are unbiased. It is usually pretty easy to prove this.

Example 1.3.6 (Our estimator of $\mathbb{P}(X_5 > 3)$ is unbiased)**.** Our estimator of $\ell = \mathbb{P}(X_5 > 3)$ is

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(X_5^{(i)} > 3),$$

where the $\{X_5^i\}_{i=1}^N$ are i.i.d. Now,

$$\mathbb{E}\widehat{\ell} = \mathbb{E}\frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(X_5^{(i)} > 3) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}\mathbb{I}(X_5^{(i)} > 3)$$

(we can justify exchanging expectation and summation using the fact that the sum is finite) and, because the $\{X_5^{(i)}\}_{i=1}^N$ are i.i.d. with the same distribution as X_5,

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{E}\mathbb{I}(X_5^{(i)} > 3) = \frac{1}{N} N \mathbb{E}\mathbb{I}(X_5^{(1)} > 3) = \mathbb{E}\mathbb{I}(X_5 > 3) = \mathbb{P}(X_5 > 3) = \ell.$$

Because Monte Carlo estimators are random (they are averages of random variables), their errors (deviations from the true value) are random as well. This means we should use probabilistic concepts to describe these errors. For unbiased estimators, variance / standard deviation is an obvious choice of error measure. This is because variance is a measure of the deviation of a random variable from the mean of its distribution (which, for unbiased estimators, is the value which is being estimated). For an estimator of the form $\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N} X^{(i)}$, with the $\{X^{(i)}\}_{i=1}^{N}$ i.i.d., the variance is given by

$$\mathrm{Var}(\widehat{\ell}) = \mathrm{Var}\left(\frac{1}{N}\sum_{i=1}^{N} X^{(i)}\right) = \frac{1}{N^2} N \mathrm{Var}(X^{(1)}) = \frac{1}{N}\mathrm{Var}(X^{(1)}). \tag{1.2}$$

It should be clear that, as N gets larger, the variance of Monte Carlo estimators gets smaller. The standard deviation of the estimator is given by

$$\mathrm{Std}(\widehat{\ell}) = \sqrt{\mathrm{Var}(\widehat{\ell})} = \frac{\sqrt{\mathrm{Var}(X^{(1)})}}{\sqrt{N}}.$$

Example 1.3.7 (The variance of our estimator of $\mathbb{P}(X_5 > 3)$)**.** The variance of

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(X_5^{(i)} > 3),$$

is given by

$$\mathrm{Var}(\widehat{\ell}) = \frac{1}{N}\mathrm{Var}(\mathbb{I}(X_5 > 3)).$$

Observe that $\mathbb{I}(X_5 > 3)$ takes the value 1 with probability $\ell = \mathbb{P}(X_5 > 3)$ and the value 0 with probability $1 - \ell$. That is, it is a Bernoulli random variable with parameter $p = \ell$. Now, the variance of a $\mathsf{Ber}(p)$ random variable is given by $p(1-p)$. So, $\mathrm{Var}(\mathbb{I}(X_5 > 3)) = \ell(1-\ell)$. In example 1.3.2 we observed that $\ell = p^5$. So,

$$\mathrm{Var}(\widehat{\ell}) = \frac{\ell(1-\ell)}{N} = \frac{p^5(1-p^5)}{N}.$$

In order to calculate variances of the form (1.2), we need to know $\mathrm{Var}(X^{(1)})$. Unfortunately, we usually do not know this. In the example above, calculating $\mathrm{Var}(\mathbb{I}(X_5 > 3))$ required us to know ℓ, which was the very quantity we were trying to estimate. In practice, we usually estimate the variance (or, more meaningfully, standard deviation) instead. This is easy to do in Matlab.

Listing 1.11: Estimating the mean and variance of our estimator of $\mathbb{P}(X_5 > 3)$

```
N = 10^4;
n = 5; p = 0.5; X_0 = 0;

X = zeros(N,5);

Y = (rand(N,n) <= p);
X = X_0 + cumsum((2*Y - 1),2);
X_5 = X(:,5);

ell_est = mean(X_5 > 3)
var_est = var(X_5 > 3) / N
std_est = std(X_5 > 3) / sqrt(N)
```

Knowing the variance / standard deviation is very useful, because it allows us to make confidence intervals for estimators. This is because of an even more famous result in probability theory than the strong law of large numbers: the central limit theorem.

Theorem 1.3.8 (Central Limit Theorem)**.** Let $\left\{X^{(i)}\right\}_{i=1}^{N}$ be a sequence of i.i.d. random values taking values in $\mathbb{R}$ such that $\mathbb{E}(X^{(1)}) = \mu$ and $\mathrm{Var}(X^{(1)}) = \sigma^2$, with $0 < \sigma^2 < \infty$. Then, the random variables

$$Z_n = \frac{\sum_{i=1}^{N} X^{(i)} - N\mu}{\sigma\sqrt{N}}$$

converge in distribution to a random variable $Z \sim \mathsf{N}(0,1)$ as $N \to \infty$.

This is important because it implies that, for large enough N,

$$\frac{1}{N}\sum_{i=1}^{N} X^{(i)} - \mathbb{E}X^{(1)}$$

is approximately Normally distributed with mean 0 and variance $\mathrm{Var}(X^{(1)})/N$. As a result, we can make *confidence intervals* for our estimators. For example, a 95% confidence interval would be of the form

$$\left(\widehat{\ell} - 1.96\frac{\sqrt{\mathrm{Var}(X^{(1)})}}{\sqrt{N}}, \widehat{\ell} + 1.96\frac{\sqrt{\mathrm{Var}(X^{(1)})}}{\sqrt{N}}\right).$$

Sometimes, for whatever reason, we are unable to find an unbiased estimator, or our unbiased estimator is not very good. For biased estimators (and, arguably, for unbiased estimators) a good measure of the error is *mean squared error*.

Definition 1.3.9 (Mean Squared Error)**.** The mean square error of the estimator, $\widehat{\ell}$, of ℓ is given by

$$\mathsf{MSE}(\widehat{\ell}) = \mathbb{E}\left(\widehat{\ell} - \ell\right)^2.$$

The mean square error can be decomposed into variance and bias terms.

Lemma 1.3.10. It holds true that $\mathsf{MSE}(\widehat{\ell}) = \mathrm{Var}(\widehat{\ell}) + \mathsf{Bias}(\widehat{\ell})^2$.

Proof. We have

$$\begin{aligned}
\mathsf{MSE}(\widehat{\ell}) &= \mathbb{E}\left(\widehat{\ell} - \ell\right)^2 = \mathbb{E}\widehat{\ell}^2 - 2\ell\mathbb{E}\widehat{\ell} + \ell^2 \\
&= \mathbb{E}\widehat{\ell}^2 + \ell^2 - 2\ell\mathbb{E}\widehat{\ell} + \left(\mathbb{E}\widehat{\ell}\right)^2 - \left(\mathbb{E}\widehat{\ell}\right)^2 \\
&= \mathbb{E}\widehat{\ell}^2 - \left(\mathbb{E}\widehat{\ell}\right)^2 + (\ell - \mathbb{E}\widehat{\ell})^2 \\
&= \mathrm{Var}(\widehat{\ell}) + \mathsf{Bias}(\widehat{\ell})^2
\end{aligned}$$

□

1.3.2 Non-Asymptotic Error Bounds

Another way to measure the error of an estimator is by $|\ell - \widehat{\ell}|$, the distance between the estimator and the value it is trying to estimate. Pretty obviously, we want this to be as small as possible. We can get bounds on this error using some famous inequalities from probability theory.

Theorem 1.3.11 (Markov's inequality)**.** Given a random variable X taking values in $\mathbb{R}$, a function $g : \mathbb{R} \to [0, \infty)$ (that is, a function that never returns negative values), and $a > 0$, we have

$$\mathbb{P}(g(X) \geq a) \leq \frac{\mathbb{E}g(X)}{a}.$$

Proof. It is clear that $g(X) \geq a\mathbb{I}(g(X) \geq a)$, so $\mathbb{E}g(X) \geq \mathbb{E}a\mathbb{I}(g(X) \geq a) = a\mathbb{E}\mathbb{I}(g(X) \geq a) = a\mathbb{P}(g(X) \geq a)$. □

If we set $g(x) = (x - \mathbb{E}X)^2$ and $a = \epsilon^2$, where $\epsilon > 0$, we have

$$\mathbb{P}((X - \mathbb{E}X)^2 \geq \epsilon^2) \leq \frac{(X - \mathbb{E}X)^2}{\epsilon^2} \Rightarrow \mathbb{P}(|X - \mathbb{E}X| \geq \epsilon) \leq \frac{\mathrm{Var}(X)}{\epsilon^2}.$$

This is Chebyshev's inequality.

Theorem 1.3.12 (Chebyshev's inequality)**.** Given a random variable X taking values in $\mathbb{R}$ with $\mathrm{Var}(X) < \infty$ and $\epsilon > 0$, we have

$$\mathbb{P}(|X - \mathbb{E}X| \geq \epsilon) \leq \frac{\mathrm{Var}(X)}{\epsilon^2}.$$

Example 1.3.13 (Error Bounds on Probability Estimators)**.** Consider the standard estimator of $\ell = \mathbb{P}(X > \gamma)$,

$$\widehat{\ell} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(X_i > \gamma).$$

We know the variance of this estimator is $N^{-1}\mathbb{P}(X > \gamma)(1 - \mathbb{P}(X > \gamma))$. So, we have the error bound

$$\mathbb{P}(|\widehat{\ell} - \ell| > \epsilon) \leq \frac{\mathbb{P}(X > \gamma)(1 - \mathbb{P}(X > \gamma))}{N\epsilon^2}.$$

1.3.3 Big O and Little o Notation

Although it is not always sensible to concentrate on how things behave asymptotically (for example, how an estimator behaves as $N \to \infty$), it often difficult to get meaningful non-asymptotic results. We will use two types of asymptotic notation in this course. The first is what is called big O notation (or, sometimes, Landau notation).

Definition 1.3.14 (Big O)**.** We say $f(x) = O(g(x))$, or f is of order $g(x)$, if there exists $C > 0$ and $x_0 > 0$ such that

$$|f(x)| \leq Cg(x)$$

for all $x \geq x_0$ as $x \to \infty$ (or, sometimes, as $x \to 0$).

Example 1.3.15. The quadratic $x^2 + 3x + 1$ is $O(x^2)$, as, for $x \geq x_0 = 1$, we have $x^2 + 3x + 1 \leq x^2 + 3x^2 + x^2 = 5x^2$, so $x^2 + 3x + 1 \leq Cx^2$ where $C = 5$.

If we can break a function into a sum of other functions (e.g., $f(x) = x^2 + 3x = f_1(x) + f_2(x)$, where $f_1(x) = x^2$ and $f_2(x) = 3x$), then the order of f is the order of the component function with the biggest order. In our example, $f_1(x) = O(x^2)$ and $f_2(x) = O(x)$, so $f(x) = O(x^2)$.

We use big O notation to describe the behavior of algorithms. If we measure the dimension of a problem by n — for example, the number of items in a list that we have to sort — then the work done by the algorithm will usually be a function of n. Usually, we prefer algorithms with smaller growth

rates for the work they have to do. For example, if algorithm 1 accomplishes a task with $O(n)$ work and another algorithm needs $O(n^2)$ work, then we will tend to prefer algorithm 1. However, if the work done by algorithm 1 is $f_1(n) = 10^6 n$ and the work done by algorithm 2 is $f_2(n) = n^2$, then the second algorithm is actually better if $n < 10^6$, even though its order is worse.

It is worth noting that $x^2 = O(x^2)$, but it is also true that $x^2 = O(x^3)$, so the equals sign is something of an abuse of notation.

Example 1.3.16 (The Standard Deviation of the Monte Carlo Estimator)**.** The standard deviation of the Monte Carlo estimator

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N} X^{(i)},$$

is

$$\frac{1}{\sqrt{N}}\sqrt{\mathrm{Var}(X^{(1)})} = O(N^{-1/2}).$$

Definition 1.3.17 (Small o Notation)**.** We say $f(x) = o(g(x))$ if

$$\frac{f(x)}{g(x)} \to 0$$

as $x \to \infty$ (or, sometimes, as $x \to 0$).

Basically, $f(x) = o(g(x))$ means that $f(x)$ is growing more slowly than $g(x)$ as x gets large (or small).

1.4 Markov Chains

The following material is closely based on the book "Markov Chains" by James Norris ([3]). This is really a wonderful book and well worth buying if you are interested in Markov chains.

As stated above, random walks are examples of Markov Chains, discrete time discrete state space stochastic processes with the Markov property. While random walks can only move up or down by 1 at each step, there is no such restriction on Markov chains in general.

On a finite state space (i.e., $|\mathcal{X}| < \infty$) a Markov chain can be represented by a transition matrix, P. The element in the ith row and jth column, $P_{i,j}$, describes the probability of going from state i to state j in one step. That is $P_{i,j} = \mathbb{P}(X_1 = j \,|\, X_0 = i)$. We will always work with homogenous Markov chains (that is, the transition probabilities will never depend on n), so we have that $P_{i,j} = \mathbb{P}(X_1 = j \,|\, X_0 = i) = \mathbb{P}(X_{n+1} = j \,|\, X_n = i)$ for all $n \geq 0$.

Example 1.4.1. Consider the Markov chain with the following graphical representation (the nodes are the states and the arrows represent possible transitions, with the probabilities attached).

PICTURE HERE

We can write this in matrix form as

$$P = \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 0 & 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 0 \end{bmatrix}$$

The convention when working with Markov chains is to describe probability distributions using row vectors. So, for example, $\boldsymbol{\mu} = (1/4, 1/4, 1/4, 1/4)$ would be a possible probability distribution for 4 states.

The initial state of the Markov chain, x_0, will be given by a probability distribution. I will try to use $\boldsymbol{\lambda}$ for this. So, for example, $\mathbb{P}(X_0 = 1) = \lambda_1$. When the Markov chain starts at a fixed state, x_0, this will be represented by the distribution $\boldsymbol{\delta}_i$ which is 0 for all states except the ith one, where it is 1. Because the Markov property tells us that it is sufficient to know the current state in order to calculate the probability of the next state, the transition probability matrix P and the initial distribution λ fully specify the Markov chain. We will call a Markov chain a (P, λ) Markov chain if it has transition probability matrix P and the initial distribution λ.

The path probabilities of a Markov chain are straightforward to calculate. We have

$$\begin{aligned} &\mathbb{P}(X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1, X_0 = x_0) \\ &= \mathbb{P}(X_n = x_n \,|\, X_{n-1} = x_{n-1}) \cdots \mathbb{P}(X_1 = x_1 \,|\, X_0 = x_0)\mathbb{P}(X_0 = x_0) \\ &= P_{x_{n-1},x_n} P_{x_{n-2},x_{n-1}} \cdots P_{x_0,x_1} \lambda_{x_0} \end{aligned}$$

We can also work out the distribution of X_n quite easily. Labeling the states from $1, \ldots, L$, the probability that we are in state 1 in the first step is given by

$$\mathbb{P}(X_1 = 1) = \lambda_1 P_{1,1} + \lambda_2 P_{2,1} + \cdots + \lambda_L P_{L,1}.$$

Likewise,

$$\mathbb{P}(X_1 = 2) = \lambda_1 P_{1,2} + \lambda_2 P_{2,2} + \cdots + \lambda_L P_{L,2}.$$

and,

$$\mathbb{P}(X_1 = L) = \lambda_1 P_{1,L} + \lambda_2 P_{2,L} + \cdots + \lambda_L P_{L,L}.$$

It is easy to see, in fact, that the distribution

$$(\mathbb{P}(X_1 = 1), \mathbb{P}(X_1 = 2), \ldots, \mathbb{P}(X_1 = L))$$

is given by $\boldsymbol{\lambda}P$. If we consider $\mathbb{P}(X_2 = x_2)$, we have

$$\mathbb{P}(X_2 = 1) = \mathbb{P}(X_1 = 1)P_{1,1} + \mathbb{P}(X_1 = 2)P_{2,1} + \cdots + \mathbb{P}(X_1 = L)P_{L,1},$$

and so on. This implies that the distribution of X_2 is given by $\boldsymbol{\lambda}P^2$. If we keep on going, we have the general result that, for a $(P, \boldsymbol{\lambda})$ Markov chain,

$$\mathbb{P}(X_n = j) = (\boldsymbol{\lambda}P^n)_j.$$

We call P^n the *n-step transition matrix*.

Example 1.4.2 (Calculating the Distribution of X_n)**.** Consider the Markov chain from example 1.4.1, with transition matrix

$$P = \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 0 & 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 0 \end{bmatrix}.$$

Given an initial distribution, $\boldsymbol{\lambda} = (1/4, 1/4, 1/4, 1/4)$, we can calculate the distribution of X_n in Matlab as follows.

Listing 1.12: Finding the Distribution of X_n

```
n = 2;
P = [0 1/3 1/3 1/3; 0 1/2 1/2 0; 1/2 0 0 1/2; 1/2 0 1/2 0];
lambda = [1/4 1/4 1/4 1/4];

X_n_dist = lambda * P^n
```

For $n = 1$, we get

$$\boldsymbol{\lambda}P = (0.2500, 0.2083, 0.3333, 0.2083).$$

For $n = 2$, we have

$$\boldsymbol{\lambda}P^2 = (0.2708, 0.1875, 0.2917, 0.2500).$$

For $n = 20$, we have

$$\boldsymbol{\lambda}P^{20} = (0.2727, 0.1818, 0.3030, 0.2424).$$

And, for $n = 1000$, we have

$$\boldsymbol{\lambda} P^{1000} = (0.2727, 0.1818, 0.3030, 0.2424).$$

If we use the initial distribution $\lambda = (1, 0, 0, 0)$, then, for $n = 1$, we have

$$\boldsymbol{\lambda} P = (0, 0.3333, 0.3333, 0.3333).$$

For $n = 2$, we have

$$\boldsymbol{\lambda} P^2 = (0.3333, 0.1667, 0.3333, 0.1667).$$

For $n = 20$, we have

$$\boldsymbol{\lambda} P^{20} = (0.2727, 0.1818, 0.3030, 0.2424).$$

And, for $n = 1000$, we have

$$\boldsymbol{\lambda} P^{1000} = (0.2727, 0.1818, 0.3030, 0.2424).$$

Notice that the distributions appears to converge to the same distribution regardless of the choice of initial distribution.

When we define Markov chains on infinite (but countable) state spaces, we can still keep lots of the formalism from the finite state space setting. Because the state space is countable, it still makes sense to talk about $P_{i,j}$. However, now the matrix is infinite, so calculating values like P^n may be a bit more difficult (we cannot just enter the matrix into Matlab and take the nth power).

Example 1.4.3 (Transition Matrix for a Random Walk)**.** For a random walk, we have the transition matrix $(P_{i,j})_{i \in \mathbb{Z}, j \in \mathbb{Z}}$, where

$$P_{i,j} = \begin{cases} p & \text{if } j = i + 1 \\ (1 - p) & \text{if } j = i - 1 \\ 0 & \text{otherwise} \end{cases}.$$

1.4.1 Simulating Markov Chains

We need some basic techniques for drawing from discrete distributions before we can start simulating Markov chains.

Drawing from a Discrete Uniform Distribution

We can simulate a random variables from the discrete uniform distribution on $\{1, \ldots, L\}$ (i.e., $\boldsymbol{\mu} = (1/L, \ldots, 1/L)$) by observing that if $U \sim \mathcal{U}(0,1)$, then

$$\mathbb{P}(\lceil LU \rceil = 1) = \mathbb{P}(LU \leq 1) = \mathbb{P}(U \leq 1/L) = 1/L,$$
$$\mathbb{P}(\lceil LU \rceil = 2) = \mathbb{P}(1 < LU \leq 2) = \mathbb{P}(1/L < U \leq 2/L) = 2/L.$$

and so on. This suggests that $\lceil LU \rceil$ is a random variable distributed uniformly on $\{1, \ldots, L\}$.

Listing 1.13: Drawing uniformly from L values.

```
X = ceil(L * rand);
```

Drawing From A Discrete Distribution on a Small State Space

Simulating a Markov Chain

It is pretty straightforward to simulate finite state space Markov chains (provided that the state space is not too big).

Algorithm 1.4.1 (Simulating a Markov Chain)**.**

(i) Draw X_0 from $\boldsymbol{\lambda}$. Set $i = 1$.

(ii) Set $X_{i+1} = j$ with probability $P_{X_i,j}$.

(iii) Set $i = i + 1$. If $i < n$ repeat from step 2.

Example 1.4.4. We can simulate the following Markov chain, where $\boldsymbol{\lambda} = (1/3, 1/3, 1/3)$ and

$$P = \begin{bmatrix} 1/2 & 1/4 & 1/4 \\ 0 & 0 & 1 \\ 2/3 & 1/3 & 0 \end{bmatrix}.$$

Listing 1.14: Matlab Code

```
n = 10^3; X = zeros(n,1);
X(1) = ceil(rand*3); i =1;
P = [1/2 1/4 1/4; 0 0 1; 2/3 1/3 0];
while i<n
    X(i+1) = min(find(rand<cumsum(P(X(i),:))));
    i = i+1;
end
```

Unfortunately, in the case of Markov chains with big (or infinite) state spaces, this approach does not work. However, there is often another way to simulate such Markov chains. We have already seen one such example for random walks.

1.4.2 Communication

It is very often the case that the distribution of X_n settles down to some fixed value as n grows large. We saw this in example 1.4.2. In order to talk about limiting behavior (and stationary distributions, which are closely related) we need the transition matrix to posses certain properties. For this reason, we introduce a number of definitions. We say that state i *leads to* state j, written $i \to j$, if

$$\mathbb{P}_i(X_n = j \text{ for some } n \geq 0) > 0.$$

That is, if it is possible to get to j from i (though not necessarily in a single step). Assuming that $\lambda_i > 0$, $\mathbb{P}_i(A) = \mathbb{P}(A \,|\, X_0 = i)$. If no $\boldsymbol{\lambda}$ is specified, then assume that $\lambda_i = 1$.

Definition 1.4.5 (Communication)**.** We say two states $i, j \in \mathcal{X}$ *communicate*, denoted $i \leftrightarrow j$, if $i \to j$ and $j \to i$.

Obviously, communicating is reflexive. That is, $i \to i$ for all $i \in \mathcal{X}$. We can partition the state space $\mathcal{X}$ into *communicating classes* as follows. We say i and j are in the communicating class C if they communicate.

Example 1.4.6 (Communicating Classes)**.** Consider the Markov chain with the following graphical representation.

PICTURE HERE

There are two communicating classes $C_1 = \{1, 2\}$ and $C_2 = \{3\}$. In full, we have the following relationships. $1 \leftrightarrow 2$, $1 \to 3$ and $2 \to 3$.

Definition 1.4.7 (Irreducibility)**.** We say a transition matrix P is *irreducible* if $\mathcal{X}$ is a single communicating class (if it is always possible to get from one state to another, though not always in just 1 step).

1.4.3 The Strong Markov Property

An important and (slightly) stronger property than the normal Markov property is the *strong Markov property*. To introduce this, we give our first definition of a stopping time.

Definition 1.4.8 (Stopping Time: Discrete Time Version)**.** A random variable $\tau \to \{0, 1, ...\} \cup \{\infty\}$ is a stopping time if the event $\{\tau = n\}$ only depends on $X_0, X_1, \ldots, X_n$, for $n \geq 0$.

Basically, in order for something to be a stopping time, we have to decide it has happen or not without knowing the future. So, for example $\tau = \inf\{n \geq 0 : X_n = 1\}$ is a stopping time, because we can tell when it occurs without seeing into the future. On the other hand, $\tau = \sup\{n \geq 0 : X_n = 1\}$ is not a stopping time for a random walk (we cannot tell when it happens unless we know the future). Another example of something that is not a stopping time is $\tau = \inf\{n \geq 0 : X_{n+1} = i\}$.

A stopping time that we have already encountered was the first passage time for the random walk.

Theorem 1.4.9 (Strong Markov Property)**.** Let $\{X_n\}_{n\geq 0}$ be a $(\boldsymbol{\lambda}, P)$ Markov chain and let τ be a stopping time of $\{X_n\}_{n\geq 0}$. Then, conditional on $\tau < \infty$ and $X_\tau = i$, $\{X_{\tau+n}\}_{n\geq 0}$ is a $(\boldsymbol{\delta}_i, P)$ Markov chain (δ_i is a distribution with 1 at state i and 0 everywhere else) and is independent of $X_0, X_1, \ldots, X_\tau$.

Proof. See [3] for a proof. □

1.4.4 Recurrence and Transience

All states in a Markov chain have the property of being either *recurrent* or *transient*.

Definition 1.4.10 (Recurrent)**.** Given a Markov chain $\{X_n\}_{n\geq 0}$ with transition matrix P, we say that a state i is recurrent if

$$\mathbb{P}_i(X_n = i \text{ for infinitely many } n) = 1.$$

Definition 1.4.11 (Transient)**.** Given a Markov chain $\{X_n\}_{n\geq 0}$ with transition matrix P, we say that a state i is transient if

$$\mathbb{P}_i(X_n = i \text{ for infinitely many } n) = 0.$$

In order to establish some important facts about recurrence and transience, we need the following result about expectations of non-negative integer valued random variables (random variables taking values in the natural numbers).

Theorem 1.4.12. Given a random variable X taking values in $\mathbb{N}$,

$$\mathbb{E}X = \sum_{i=0}^{\infty} \mathbb{P}(X > i).$$

Proof. We have

$$\begin{aligned}\mathbb{E}X &= \sum_{x=1}^{\infty} x\mathbb{P}(X=x) = \sum_{x=1}^{\infty}\sum_{i=0}^{x-1}\mathbb{P}(X=x)\\ &= \sum_{i=0}^{\infty}\sum_{x=i+1}^{\infty}\mathbb{P}(X=x) = \sum_{i=0}^{\infty}\mathbb{P}(X>i),\end{aligned}$$

where we can swap sums because of the non-negative summands. □

We also need to introduce a few stopping times.

Definition 1.4.13 (First Passage Time Including Return)**.** We define the first passage time into the state i (including the possibility of starting in i) as

$$\widetilde{\tau}_i = \inf\{n \geq 1 : X_n = i\}$$

Definition 1.4.14 (rth Passage Time)**.** We define the *rth passage time* by $\widetilde{\tau}_i^{(0)} = 0$ and

$$\widetilde{\tau}_i^{(r+1)} = \inf\{n < \tau_i^{(r)} : X_n = i\} \quad \text{for } r \geq 0.$$

We also define a sequence of random variables that are not stopping times but describe the times between visits to state i.

Definition 1.4.15 (rth Excursion Length)**.** We define the rth excursion length as

$$S_i^{(r)} = \begin{cases}\widetilde{\tau}_i^{(r)} - \widetilde{\tau}_i^{(r-1)} & \text{if } \widetilde{\tau}_i^{(r-1)} < \infty\\ 0 & \text{otherwise}\end{cases}.$$

Lemma 1.4.16. For $r \geq 2$, conditional on $\widetilde{\tau}_i^{(r-1)} < \infty$, $S_i^{(r)}$ is independent of $\{X_n\}_{n\geq 0}^{\widetilde{\tau}_i^{(r-1)}}$ and

$$\mathbb{P}(S_i^{(r)} = n \,|\, \widetilde{\tau}_i^{(r-1)} < \infty) = \mathbb{P}_i(\widetilde{\tau}_i = n).$$

Proof. If we use the strong Markov property with the stopping time $\tau = \widetilde{\tau}_i^{(r-1)}$, we get that $\{X_{\tau+n}\}_{n\geq 0}$ is a $(\boldsymbol{\delta}_i, P)$ Markov property that is independent of $X_0, \ldots, X_\tau$. Now, we can write $S_i^{(r)}$ as

$$S_i^{(r)} = \inf\{n > 0 : X_{\tau+n} = i\},$$

so $S_i^{(r)}$ is the first passage time, including return, to state i for $\{X_{\tau+n}\}_{n\geq 0}$. □

We define the *number of visits* to state i by

$$V_i = \sum_{n=0}^{\infty} \mathbb{I}(X_n = i).$$

Let $\mathbb{E}_i$ be the expectation given that the process starts in state i.

Lemma 1.4.17. It holds that $\mathbb{E}_i V_i = \sum_{n=0}^{\infty} P_{i,i}^n$.

Proof.

$$\mathbb{E}_i V_i = \mathbb{E}_i \sum_{n=0}^{\infty} \mathbb{I}(X_n = i) = \sum_{n=0}^{\infty} \mathbb{E}_i \mathbb{I}(X_n = i) = \sum_{n=0}^{\infty} \mathbb{P}_i(X_n = i) = \sum_{n=0}^{\infty} P_{i,i}^n.$$

□

Lemma 1.4.18. For $r \geq 0$, $\mathbb{P}_i(V_i > r) = \left(\mathbb{P}_i(\widetilde{\tau}_i < \infty)\right)^r$.

Proof. When $r = 0$, this is clearly true. If it is true for r, then

$$\begin{aligned}&\mathbb{P}_i(V_i > r+1) = \mathbb{P}_i(\widetilde{\tau}_i^{(r+1)} < \infty) = \mathbb{P}_i(\widetilde{\tau}_i^{(r)} < \infty \text{ and } S_i^{(r+1)} < \infty)\\ &= \mathbb{P}_i(S_i^{(r+1)} < \infty \,|\, \widetilde{\tau}_i^{(r)} < \infty)\mathbb{P}_i(\widetilde{\tau}_i^{(r)} < \infty) = \mathbb{P}_i(\widetilde{\tau}_i < \infty)\mathbb{P}_i(\widetilde{\tau}_i^{(r)} < \infty).\end{aligned}$$

□

Theorem 1.4.19. The following holds

(i) If $\mathbb{P}_i(\widetilde{\tau}_{\{i\}} < \infty) = 1$ then i is recurrent and $\sum_{n=0}^{\infty} P_{i,i}^n = \infty$.

(ii) If $\mathbb{P}_i(\widetilde{\tau}_{\{i\}} < \infty) < 1$ then i is transient and $\sum_{n=0}^{\infty} P_{i,i}^n < \infty$.

Proof. We prove the two statements separately.

Part 1. If $\mathbb{P}_i(\widetilde{\tau}_i < \infty) = 1$ then $\mathbb{P}_i(V_i = \infty) = \lim_{r\to\infty} \mathbb{P}_i(V_i > r)$. By lemma 1.4.18,

$$\lim_{r\to\infty} \mathbb{P}_i(V_i > r) = \lim_{r\to\infty} \left(\mathbb{P}_i(\widetilde{\tau}_i < \infty)\right)^r = 1,$$

so i is recurrent. If $\mathbb{P}_i(V_i = \infty) = 1$ then $\mathbb{E}_i V_i = \infty$. Now, by lemma 1.4.17,

$$\sum_{n=0}^{\infty} P_{i,i}^n = \mathbb{E}_i V_i = \infty.$$

Part 2. If $\mathbb{P}_i(\widetilde{\tau}_i < \infty) < 1$ then

$$\mathbb{P}_i(V_i = \infty) = \lim_{r\to\infty} \mathbb{P}_i(V_i > r) = \lim_{r\to\infty} \left(\mathbb{P}_i(\widetilde{\tau}_i < \infty)\right)^r = 0,$$

so i is transient. Now,

$$\sum_{n=0}^{\infty} P_{i,i}^n = \mathbb{E}_i V_i = \sum_{r=0}^{\infty} \mathbb{P}_i(V_i > r) = \sum_{r=0}^{\infty} \left(\mathbb{P}_i(\widetilde{\tau}_i < \infty)\right)^r = \frac{1}{1 - \mathbb{P}_i(\widetilde{\tau} < \infty)} < \infty.$$

□

Theorem 1.4.20. Let C be a communicating class. Then either all states in C are transient or all are recurrent.

Proof. Take a pair of states i and j in C and assume i is transient. Because i and j are in the same communicating class, there must exist $n \geq 0$ and $m \geq 0$ so that $P_{i,j}^n > 0$ and $P_{j,i}^m > 0$. Now, it must be the case that

$$P_{i,i}^{n+r+m} \geq P_{i,j}^n P_{j,j}^r P_{j,i}^m$$

as this only describes the probability of one possible path from i back to i (such a path need not pass through j). Rearranging, we have

$$P_{j,j}^r \leq \frac{P_{i,i}^{n+r+m}}{P_{i,j}^n P_{j,i}^m}.$$

Summing over r we get

$$\sum_{r=0}^{\infty} P_{j,j}^r \leq \frac{\sum_{r=0}^{\infty} P_{i,i}^{n+r+m}}{P_{i,j}^n P_{j,i}^m}.$$

Now, because i is assumed to be transient, $\sum_{r=0}^{\infty} P_{i,i}^{n+r+m} < \infty$. As a result, $\sum_{r=0}^{\infty} P_{j,j}^r < \infty$, implying j is also transient. Thus, the only way a state can be recurrent is if all states are recurrent. □

Definition 1.4.21 (Closed Class)**.** A communicating class C is closed if $i \in C$ and $i \to j$ implies $j \in C$.

Theorem 1.4.22. Every finite closed class is recurrent.

Proof. See [3]. □

As an irreducible Markov chain consists of one single closed class, this implies that all irreducible Markov chains on finite state spaces are recurrent.

Recurrence of Random Walks

We can use the criteria given in theorem 1.4.19 to establish facts about the recurrence properties of random walks. In order to establish recurrence, we would need $\sum_{n=0}^{\infty} P_{0,0}^n = \infty$. Thus, a first step is to find an expression for $P_{0,0}^n$. As we have already discussed, a random walk starting at 0 can only return to 0 in an even number of steps. Thus, it is sufficient for us to consider $P_{0,0}^{2n}$. Now in order for a random walk to end up at 0 after $2n$ steps, it needs to take exactly n up steps and n down steps. There are $\binom{2n}{n}$ ways of taking this many steps. This gives

$$P_{0,0}^{2n} = \binom{2n}{n} p^n (1-p)^n.$$

Lemma 1.4.23. The following bounds on $n!$ hold of $n \geq 1$.

$$\sqrt{2\pi}\, n^{n+1/2}\, e^{-n} \leq n! \leq e\, n^{n+1/2}\, e^{-n}$$

Given these bounds on $n!$ it is relatively straightforward to get bounds on $P_{0,0}^n$ that allow us to establish the transience or recurrence of the random walk.

Theorem 1.4.24. A symmetric random walk (i.e., a random walk with $p = (1-p) = 1/2$) is recurrent.

Proof. To show the random walk is recurrent, we need to show

$$\sum_{n=0}^{\infty} P_{0,0}^{2n} = \infty.$$

The first step is to get a bound on $\binom{2n}{n}$. By lemma 1.4.23, we have

$$\binom{2n}{n} = \frac{(2n)!}{n!n!} \geq \frac{\sqrt{2\pi}\, 2n^{2n+1/2}\, e^{-2n}}{e\, n^{n+1/2}\, e^{-n} e\, n^{n+1/2}\, e^{-n}} = \frac{2\sqrt{\pi} 4^n}{e^2 \sqrt{n}}.$$

So, we can bound the probabilities from below by

$$P_{0,0}^{2n} \geq \frac{2\sqrt{\pi}}{e^2} \frac{(4p(1-p))^n}{\sqrt{n}}.$$

This implies that

$$\sum_{n=0}^{\infty} P_{0,0}^{2n} \geq \frac{2\sqrt{\pi}}{e^2} \sum_{n=0}^{\infty} \frac{(4p(1-p))^n}{\sqrt{n}}.$$

If $p = (1-p)$ then $4p(1-p) = 1$, so

$$\sum_{n=0}^{\infty} P_{0,0}^{2n} \geq C \sum_{n=0}^{\infty} \frac{1}{\sqrt{n}} = \infty,$$

where $C > 0$ is a constant. □

1.4.5 Invariant Distributions

A topic of enormous interest, especially from a simulation perspective, is the behavior of the distribution of a Markov chain, $\{X_n\}_{n\geq 0}$ as n becomes large. It turns out there are actually a few different things we can mean by this. The best possible situation is that a Markov chain can have a *limiting distribution*. That is, there can exist a distribution, $\boldsymbol{\pi}$, such that $\{X_n\}_{n\geq 0} \to \boldsymbol{\pi}$ as $n \to \infty$ no matter what initial distribution, $\boldsymbol{\lambda}$, we choose. We will discuss the conditions for a limiting distribution to exist later. It turns out, however, that even if a limiting distribution does not exist, the Markov chain may have a unique *stationary distribution*, sometimes also called an *invariant distribution* or an *equilibrium distribution*. This is a distribution, $\boldsymbol{\pi}$, such that $\boldsymbol{\pi}P = \boldsymbol{\pi}$.

Definition 1.4.25 (Stationary Distribution)**.** An invariant distribution of a Markov chain with transition matrix P is a distribution that satisfies

$$\boldsymbol{\pi}P = \boldsymbol{\pi}.$$

The importance of stationary distributions is made clear by the following lemma.

Lemma 1.4.26. Let $\{X_n\}_{n\geq 0}$ be Markov $(\boldsymbol{\pi}, P)$ and suppose $\boldsymbol{\pi}$ is a stationary distribution for P, then $X_n \sim \pi$ for all $n \geq 0$.

Proof. The distribution of X_n is given by πP^n. Now, for $n = 0$ (where we define $P^0 = I$) it is clearly true that $\boldsymbol{\pi}P^n = \boldsymbol{\pi}$. We just need to show $\boldsymbol{\pi}P^n = \boldsymbol{\pi}$ for $n > 1$. We do this by induction. That is, assume $\boldsymbol{\pi}P^n = \boldsymbol{\pi}$. Then

$$\boldsymbol{\pi}P^{n+1} = (\boldsymbol{\pi}P)P^n = \boldsymbol{\pi}P^n = \boldsymbol{\pi}$$

by assumption. Thus, as this is true for $n = 0$, the result follows. □

The obvious questions to ask when faced with a nice and interesting object like a stationary distribution are: does one exist? and, if so, is it unique?. The simple answer is that it is often the case that there is a unique stationary distribution (at least for many of the Markov chains we might be interested in), but that a number of conditions need to be fulfilled. One of these is *positive recurrence*.

Recurrent implies that if the Markov chain starts in state i it will return to state i with probability 1. However, this is no guarantee that the expected return time is finite. Recall that $m_i = \mathbb{E}_i\widetilde{\tau}_i$, where $\widetilde{\tau}_i = \inf\{n > 0 : X_n = i\}$.

Definition 1.4.27 (Positive Recurrence)**.** We say a Markov chain is positive recurrent if $m_i < \infty$.

Definition 1.4.28 (Null Recurrence)**.** We say a Markov chain is *null recurrent* if $m_i = \infty$.

It is straightforward to establish that if a Markov chain is irreducible and has a finite state space (i.e. $|\mathcal{X}| < \infty$), then every state is positive recurrent. It is not always easy to show that an infinite state space Markov chain has such a property.

Now that we have defined positive recurrence, we are able to give conditions that guarantee a Markov chain has a unique stationary distribution.

Theorem 1.4.29. Let P be irreducible. Then the following are equivalent:

(i) Every state is positive recurrent.

(ii) Some state i is positive recurrent.

(iii) P has a (unique) invariant distribution, $\boldsymbol{\pi}$, and $m_i = 1/\pi_i$.

Proof. See [3]. □

Note that it is possible for a Markov chain to have a stationary distribution but not satisfy these conditions.

Example 1.4.30 (Calculating a stationary distribution)**.** We can find a stationary distribution by solving $\boldsymbol{\pi}P = \boldsymbol{\pi}$. For example, consider the following Markov chain

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \end{bmatrix}.$$

So, we need to solve

$$(\pi_1, \pi_2, \pi_2) \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \end{bmatrix} = (\pi_1, \pi_2, \pi_2). \tag{1.3}$$

This gives the following linear system

$$\begin{aligned} \frac{1}{2}\pi_2 + \frac{1}{2}\pi_3 &= \pi_1 \\ \frac{1}{2}\pi_1 + \frac{1}{2}\pi_3 &= \pi_2 \\ \frac{1}{2}\pi_1 + \frac{1}{2}\pi_2 &= \pi_3. \end{aligned}$$

Solving this, we have $\pi_1 = \pi_2 = \pi_3$. As we need $\pi_1 + \pi_2 + \pi_3 = 1$ if $\boldsymbol{\pi}$ is to be a distribution, we have $\pi_1 = \pi_2 = \pi_3 = 1/3$.

1.4.6 Limiting Distribution

The fact that a Markov chain has a unique stationary distribution does not guarantee that it has a limiting distribution. This can be shown using a simple example.

Example 1.4.31 (A Markov chain with a stationary distribution but no limiting distribution)**.** Consider the Markov chain with graph

PICTURE HERE.

The transition matrix is given by

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

This chain is clearly irreducible and positive recurrent, so it has a unique stationary distribution, $\boldsymbol{\pi} = (1/2, 1/2)$. However, $P^{2n} = I$ and $P^{2n+1} = P$. This means that if we start with certainty in a given state the distribution will not converge. For example, if $\boldsymbol{\lambda} = \boldsymbol{\delta}_i = (1, 0)$, then $\lim_{n\to\infty} \boldsymbol{\lambda} P^{2n} = (1, 0)$ and $\lim_{n\to\infty} \boldsymbol{\lambda} P^{2n+1} = (0, 1)$. Thus, no limit exists.

The problem with the Markov chain in example 1.4.31 is that it is periodic: it is only possible to get from state 1 back to state 1 in an even number of steps. In order for a Markov chain to have a limiting distribution, it should not have any periodicity. Unsurprisingly, this requirement is called *aperiodicity*.

Definition 1.4.32. A state is said to be aperiodic if $P^n_{i,i} > 0$ for all sufficiently large n. Equivalently, the set $\{n \geq 0 : P^n_{i,i} > 0\}$ has no common divisor other than 1.

In a irreducible Markov chain, all states are either aperiodic or periodic.

Theorem 1.4.33. Suppose P is irreducible and has an aperiodic state i. Then, all states are aperiodic.

A random walk is periodic with period 2.

Example 1.4.34.

PICTURE HERE.

PICTURE HERE.

If a Markov chain is irreducible, positive recurrent and aperiodic, then it has a limiting distribution.

Theorem 1.4.35. Let P be irreducible and aperiodic and suppose P has an invariant distribution $\boldsymbol{\pi}$. Let $\boldsymbol{\lambda}$ be any distribution. and suppose $\{X_n\}_{n\geq 0}$ is Markov $(\boldsymbol{\lambda}, P)$. Then, $\mathbb{P}(X_n = j) \to \pi_j$ as $n \to \infty$ for all $j \in \mathcal{X}$. In particular, $P^n_{i,j} \to \pi_j$ as $n \to \infty$ for all $i, j \in \mathcal{X}$.

Proof. See [3]. □

1.4.7 Reversibility

Stationary distributions play a number of very important roles in simulation. In particular, we often wish to create a Markov chain that has a specified stationary distribution. One important tool for finding such a Markov chain is to exploit a property called *reversibility*. Not all Markov chains are reversible but, as we shall see, those that are have some nice properties.

Definition 1.4.36. Let $\{X_n\}_{n\geq 0}$ be a Markov $(\boldsymbol{\pi}, P)$ with P irreducible. We say that $\{X_n\}_{n\geq 0}$ is reversible if, for all $N \geq 1$, $\{X_{N-n}\}_{n=0}^N$ is also Markov $(\boldsymbol{\pi}, P)$.

The most important property of reversible Markov chains is that they satisfy the *detailed balance equations*.

Definition 1.4.37. A matrix P and a measure $\boldsymbol{\nu}$ (a vector will all non-negative components) are in detailed balance if

$$\nu_i P_{i,j} = \nu_j P_{j,i} \quad \forall i, j \in \mathcal{X}.$$

The detailed balance equations are important because they allow us to find stationary distributions (and, as we will learn later, construct transition matrices with given stationary distributions).

Lemma 1.4.38. If P and the distribution $\boldsymbol{\pi}$ are in detailed balance, then $\boldsymbol{\pi}$ is invariant for P.

Proof. We have

$$(\boldsymbol{\pi} P)_i = \sum_{j\in\mathcal{X}} \pi_j P_{j,i} = \sum_{j\in\mathcal{X}} \pi_i P_{i,j} = \pi_i.$$

□

The following theorem shows that reversibility is equivalent to satisfying the detailed balance equations.

Theorem 1.4.39. Let P be an irreducible stochastic matrix and let $\boldsymbol{\pi}$ be a distribution. Suppose $\{X_n\}_{n\geq 0}$ is Markov $(\boldsymbol{\pi}, P)$. Then, the following are equivalent.

(i) $\{X_n\}_{n\geq 0}$ is reversible.

(ii) P and $\boldsymbol{\pi}$ are in detailed balance (i.e., $\boldsymbol{\pi}$ is the stationary distribution of P).

Proof. See [3]. □

Random Walks on Graphs

Given a graph with a finite number of vertices, labeled $1, \ldots, L$, each with finite degree (i.e., $d_1 < \infty, \ldots, d_L < \infty$), we can define a random walk with the following transition probabilities

$$P_{i,j} = \begin{cases} \frac{1}{d_i} & \text{if there is an edge between } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}.$$

Example 1.4.40 (A random walk on a graph)**.** Consider the following graph.

PICTURE HERE.

Using the above transition probabilities, we have the Markov chain

PICTURE HERE.

If a graph is connected, then a random walk on it will be irreducible. Given a connected graph, it is easy to establish the stationary distribution of the resulting random walk using the detailed balance equations.

Lemma 1.4.41. For connected graphs with $\sum_i d_i < \infty$ we have

$$\pi_i = \frac{d_i}{\sum_{i\in\mathcal{X}} d_i}.$$

Proof. We just need to confirm that the detailed balance equations hold and that the probabilities sum to 1. First, observe that

$$\frac{d_i}{\sum_{i\in\mathcal{X}} d_i}\frac{1}{d_i} = \frac{d_j}{\sum_{i\in\mathcal{X}} d_j}\frac{1}{d_j} \quad \text{holds for all } i, j \in \mathcal{X}.$$

Now, pretty clearly, $\sum_{i\in\mathcal{X}} \pi_i = 1$, so we are done. □

Example 1.4.42 (A random walk on a graph (cont.))**.** For the graph given in example 1.4.40, we have

$$d_1 + d_2 + d_3 + d_4 = 2 + 3 + 3 + 2 = 10.$$

Thus,

$$\pi_1 = 2/10,\ \pi_2 = 3/10,\ \pi_3 = 3/10,\ \pi_4 = 2/10.$$

1.4.8 The Ergodic Theorem

The ergodic theorem is something like the strong law of large numbers for Markov chains. It tells us the sample averages converge to expected values almost surely. In addition, the second part of the theorem is very useful practically as it is often the case that we are not directly interested in a Markov chain, $\{X_n\}_{n\geq 0}$, but rather in the behavior of some stochastic process $\{Y_n\}_{n\geq 0}$, where $Y_n = f(X_n)$ and f is a deterministic function. For example $\{X_n\}_{n\geq 0}$ could be a Markov chain describing the weather and f could be a function describing how much electricity is consumed by a town (on average) under certain weather conditions. We might then be interested in the average power consumption, which we could estimate by the sample average

$$S_n = \frac{1}{n}\sum_{k=0}^{n} f(X_k).$$

The ergodic theorem guarantees that such sample averages converge to the correct expected values.

Definition 1.4.43 (Number of visits to i before time n)**.** We define the number of visits to i before time n by

$$V_i(n) = \sum_{k=0}^{n-1} \mathbb{I}(X_k = i).$$

Theorem 1.4.44 (Ergodic Theorem)**.** Let $\{X_n\}_{n\geq 0}$ be Markov $(\boldsymbol{\lambda}, P)$, where P is an irreducible transition matrix and $\boldsymbol{\lambda}$ is an arbitrary initial distribution. Then,

$$\mathbb{P}\left(\frac{V_i(n)}{n} \to \frac{1}{m_i} \text{ as } n \to \infty\right) = 1.$$

If, in addition, P is positive recurrent, then for any bounded function $f : \mathcal{X} \to \mathbb{R}$,

$$\mathbb{P}\left(\frac{1}{n}\sum_{k=0}^{n-1} f(X_k) \to \sum_{i\in\mathcal{X}} \pi_i f(i) \text{ as } n \to \infty\right) = 1,$$

where $\boldsymbol{\pi}$ is the stationary distribution of P.

Proof.

Part 1. If P is transient, then the chain will only return to state i a finite number of times. This means that

$$\frac{V_i(n)}{n} \leq \frac{V_i}{n} \to 0 = \frac{1}{m_i}.$$

Now, consider the recurrent case. For a given state i, we have $\mathbb{P}(\widetilde{\tau}_i < \infty) = 1$. Using the strong Markov property, the Markov chain $\{X_{\widetilde{\tau}_i+n}\}_{n\geq 0}$ is Markov $(\boldsymbol{\delta}_i, P)$ and independent of $X_0, \ldots, X_{\widetilde{\tau}_i}$. As the long run proportion of time spend in i is the same for $\{X_n\}_{n\geq 0}$ and $\{X_{\widetilde{\tau}_i+n}\}_{n\geq 0}$, we are safe to assume that the chain starts in i.

Now, by time $n-1$, the chain will have made at most $V(n)$ visits to i. It will certainly have made at least $V(n)-1$ visits. The total length of time required to make all these visits must be less than or equal to $n-1$ (because, by definition, all these visits occur within the first $n-1$ steps). This means that

$$S_i^{(1)} + \cdots + S_i^{(V_i(n)-1)} \leq n-1.$$

By a similar argument,

$$n \leq S_i^{(1)} + \cdots + S_i^{(V_i(n))}.$$

Using these bounds, we have

$$\frac{S_i^{(1)} + \cdots + S_i^{(V_i(n)-1)}}{V_i(n)} \leq \frac{n}{V_i(n)} \leq \frac{S_i^{(1)} + \cdots + S_i^{(V_i(n))}}{V_i(n)}.$$

Now, we know the the excursion lengths are i.i.d. random variables (with finite mean m_i), so, by the large of large numbers,

$$\mathbb{P}\left(\frac{S_i^{(1)} + \cdots + S_i^{(n)}}{n} \to m_i \text{ as } n \to \infty\right) = 1.$$

Now, we know that $\mathbb{P}(V_i(n) \to \infty \text{ as } n \to \infty) = 1$. So, letting $n \to \infty$, we squeeze $n/V_i(n)$ to get

$$\mathbb{P}\left(\frac{n}{V_i(n)} \to m_i \text{ as } n \to \infty\right) = 1.$$

This implies that

$$\mathbb{P}\left(\frac{V_i(n)}{n} \to \frac{1}{m_i} \text{ as } n \to \infty\right) = 1.$$

Part 2. See [3].

□

1.5 Extending the Random Walk Model

1.5.1 Sums of Independent Random Variables

If we consider a random walk with $x_0 = 0$, we can define it by

$$X_n = \sum_{i=1}^{n} Z_i,$$

where the $\{Z_i\}_{i\geq 1}$ are i.i.d. random variables with

$$\mathbb{P}(Z_1 = 1) = 1 - \mathbb{P}(Z_1 = -1) = p.$$

If we replace the $\{Z_i\}_{i\geq 1}$ with an arbitrary sequence of independent random variables, $\{Y_i\}_{i\geq 1}$ we are in the setting of sums of independent random variables. That is, we consider

$$S_n = \sum_{i=1}^{n} Y_i.$$

Now, in the case of a random walk, which is a sum of the $\{Z_i\}_{i\geq 1}$ random variables, we know the distribution of S_n (we calculated this earlier). However, this is not always the case. Normally, in order to find the distribution of a sum of n random variables, we have to calculate an n-fold convolution or use either moment generating functions or characteristic functions and hope that things work out nicely.

Recall, given two independent random variables, X and Y, the convolution of the distributions of X and Y is given by

$$\begin{aligned}\mathbb{P}(X + Y = z) &= \sum_{x=-\infty}^{\infty} \mathbb{P}(X = x)\mathbb{P}(Y = z - x) \\ &= \sum_{y=-\infty}^{\infty} \mathbb{P}(X = z - y)\mathbb{P}(Y = y)\end{aligned}$$

in the discrete case and the convolution, h, of the density of X, f, and the density of Y, g, is given by

$$h(z) = (f * g)(z) = \int_{-\infty}^{\infty} f(x)g(z - x)\mathrm{d}x = \int_{-\infty}^{\infty} f(z - y)g(y)\mathrm{d}y$$

in the continuous case. It is easy to see that the calculations can be pretty messy if lots of variables with different distributions are involved.

Sometimes, things are nice. For example, the sum of independent normal random variables is normally distributed and the sum of i.i.d. exponential random variables is distributed according to a special case of the gamma distribution (called the Erlang distribution). But most things are not so nice. For example, try to work out the distribution of n exponential random variables with parameters $\lambda_1, \ldots, \lambda_n$.

There are various tools in mathematics that help us deal with sums of independent random variables. For example, we have the Lindeberg central limit theorem and a version of the strong law of large numbers. However, these do not answer all the questions we might reasonably ask and there are lots of random variables that do not satisfy the technical conditions of these theorems. Simulation is a useful tool for solving problems in these settings.

We will consider a couple of examples that use normal random variables. Recall that if $Z \sim \mathsf{N}(0,1)$ then $X = \mu + \sigma Z \sim \mathsf{N}(\mu, \sigma^2)$. This means we can simulate a normal random variable with mean μ and variance σ^2 in Matlab using the command

```
X = mu + sqrt(sigma_sqr) * randn;
```

Another new concept in the examples is *relative error*.

Definition 1.5.1 (Relative Error)**.** The relative error of an estimator $\widehat{\ell}$ is defined by

$$\mathsf{RE} = \frac{\sqrt{\mathrm{Var}(\widehat{\ell})}}{\ell}.$$

Basically, the relative error tell us the size of our estimator's error as a percentage of the thing we are tying to estimate (i.e., if we have a relative error of 0.01, that means that the standard deviation of our estimator is about 1 percent of ℓ). The relative error is often a more meaningful measure of error in settings where the thing we are trying to estimate, ℓ, is small. In practice, the relative error needs to be estimated.

Example 1.5.2 (Sums of log-normal random variables)**.** Consider a portfolio consisting of n stocks (which, for some reason, are independent of one another). At the start of the year, the stocks all have value 1. The changes in value of these stocks over a year are given by the random variables $V_1, \ldots, V_n$, which are log-normal random variables, i.e., $V_1 = e^{Z_1}, ., V_n = e^{Z_n}$ with $Z_1 \sim \mathsf{N}(\mu_1, \sigma_1^2), \cdots, Z_1 \sim \mathsf{N}(\mu_n, \sigma_n^2)$. It is not so straightforward to calculate the distribution of $S_n = V_1 + \cdots + V_n$.

If $n = 5$, with $\boldsymbol{\mu} = (-0.1, 0.2, -0.3, 0.1, 0)$ and $\boldsymbol{\sigma^2} = (0.3, 0.3, 0.3, 0.2, 0.2)$, what is the probability that the portfolio is worth more than 20 at the

end of the year? It is straightforward to use Monte Carlo to get an estimate of $\ell = \mathbb{P}(S_n > 20)$. Here, we check that the mean and variance of our simulation output correspond to the theoretical mean and variance (it never hurts to check things seem to be working properly). The mean of a log-normal random variable is given by $\exp\{\mu + \sigma^2/2\}$ and the variance is given by $(\exp\{\sigma^2\} - 1)\exp\{2\mu + \sigma^2\}$. In addition, we estimate $\mathbb{E}\left[\max(V_1, \ldots, V_5) \,|\, S_5 > 20\right]$ (that is, the average value of the largest portfolio component when the portfolio has a value bigger than 20) and $\mathbb{E}\left[S_5 \,|\, S_5 > 20\right]$.

Listing 1.15: Matlab code

```
N = 5*10^7; S = zeros(N,1); threshold = 20;
V_max = zeros(N,1); V_mean = zeros(N,1);

mu = [-0.1 0.2 -0.3 0.1 0]; sigma_sqr = [.3 .3 .3 .2 .2];

for i = 1:N
    Z = mu + sqrt(sigma_sqr) .* randn(1,5);
    V = exp(Z);
    V_max(i) = max(V);
    S(i) = sum(V);
end

est_mean = mean(S)
actual_mean = sum(exp(mu + sigma_sqr/2))

est_var = var(S)
actual_var = sum((exp(sigma_sqr) - 1) .* exp(2 * mu + sigma_sqr))

ell_est = mean(S>threshold)
ell_RE = std(S>threshold) / (ell_est * sqrt(N))

[event_occurs_index dummy_var] = find(S > threshold);
avg_max_v = mean(V_max(event_occurs_index))
avg_S = mean(S(event_occurs_index))
```

Running this one time produced the following output

```
est_mean = 5.6576
actual_mean = 5.6576
est_var = 1.9500
actual_var = 1.9511
ell_est = 2.3800e-06
ell_RE = 0.0917
```

```
avg_max_v = 15.9229
avg_S = 21.5756
```

Notice that, on average, the rare event seems to be caused by a single portfolio component taking a very large value (rather than all the portfolio components taking larger than usual values). This is typical of a class of random variables called heavy tailed random variables, of which the log-normal distribution is an example.

Example 1.5.3 (A gambler's ruin problem)**.** Consider an incompetent businessman. His company starts off with €10000 but makes a loss, on average, each day. More precisely, the profit or loss on the ith day is given by $Y_i \sim \mathsf{N}(-20, 10000)$. If his company can get €11000 in the bank, he is able to sell his company to a competitor. If his company's bank account drops below €0 he goes bankrupt. What is the probability that he is able to sell the company?

We can formulate this as a problem about hitting times. Define $S_n = \sum_{i=1}^{n} Y_i$, as the company bank account (minus the initial €10000) on the nth day. Define the time at which he can sell by

$$\tau_S = \inf\{n \geq 1 : S_n \geq 1000\}$$

and the time at which he can go bankrupt by

$$\tau_B = \inf\{n \geq 1 : S_n \leq -10000\}.$$

We want to know $\ell = \mathbb{P}(\tau_S < \tau_B)$. This is easy to simulate, we just increase n by one until either $S_n \leq -10000$ or $S_n \geq 1000$. We might as well find out $\mathbb{E}[\tau_S \,|\, \tau_S < \tau_B]$ and $\mathbb{E}[\tau_B \,|\, \tau_B < \tau_S]$ while we are doing that.

Listing 1.16: Matlab code

```
N = 10^7; sold = zeros(N,1); days = zeros(N,1);
mu = -20; sigma_sqr = 10000; sigma = sqrt(sigma_sqr);
up = 1000; low = -10000;

for i = 1:N
    S = 0; n = 0;
    while S > low && S< up
        S = S + (mu + sigma * randn);
        n = n + 1;
    end
    sold(i) = S > up;
    days(i) = n;
```

```
end

ell_est = mean(sold)
est_RE = sqrt(sold) / (sqrt(N)*ell_est)

[event_occurs_index dummy_var] = find(sold == 1);
[event_does_not_occur_index dummy_var] = find(sold == 0);

avg_days_if_sold = mean(days(event_occurs_index))
avg_days_if_bankrupt = mean(days(event_does_not_occur_index))
```

Running this one time produced the following output

```
ell_est = 0.0145
ell_RE = 0.0026
avg_days_if_sold = 52.9701
avg_days_if_bankrupt = 501.5879
```

1.6 Importance Sampling

In examples 1.5.2 and 1.5.3, the probabilities we were interested in were quite small. Estimating such quantities is usually difficult. If you think about it, if something only happens on average once every 10^6 times, then you will need a pretty big sample size to get many occurrences of that event. We can be a bit more precise about this. Consider the relative error of the estimator $\widehat{\ell}$ for $\ell = \mathbb{P}(X > \gamma) = \mathbb{E}\mathbb{I}(X > \gamma)$. This is of the form

$$\mathsf{RE} = \frac{\sqrt{\mathbb{P}(X > \gamma)\left(1 - \mathbb{P}(X > \gamma)\right)}}{\mathbb{P}(X > \gamma)\sqrt{N}}.$$

So, for a fixed RE, we need

$$\sqrt{N} = \frac{\sqrt{\mathbb{P}(X > \gamma)\left(1 - \mathbb{P}(X > \gamma)\right)}}{\mathbb{P}(X > \gamma)\mathsf{RE}} \Rightarrow N = \frac{1 - \mathbb{P}(X > \gamma)}{\mathbb{P}(X > \gamma)\mathsf{RE}^2}.$$

So $N = O(1/\mathbb{P}(X > \gamma))$ which means N gets big very quickly as $\ell = \mathbb{P}(X > \gamma) \to 0$. This is a big problem in areas where events with small probabilities are important. There are lots of fields where such events are important: for example, physics, finance, telecommunication, nuclear engineering, chemistry and biology. One of the most effective methods of estimating these probabilities is called *importance sampling*.

In the case of sums of independent random variables, the basic idea is to change the distributions of the random variables so that the event we are interested in is more likely to occur. Of course, if we do this, we will have a biased estimator. So, we need a way to correct for this bias. It is easiest to describe these things using continuous random variables and densities but, as we will see in the examples, everything works for discrete random variables as well.

Consider a random variable X taking values in $\mathbb{R}$ with density f. Suppose we wish to estimate $\ell = \mathbb{E}S(X)$. Note that we can write

$$\mathbb{E}S(X) = \int_{-\infty}^{\infty} S(x)\, f(x)\, \mathrm{d}x.$$

Now, this suggests the natural estimator

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N} S(X^{(i)}),$$

where $X^{(1)}, \ldots, X^{(N)}$ are i.i.d. draws from the density f. Now, suppose the expectation of $S(X)$ is most influenced by a subset of values with low probability. For example, if $S(X) = \mathbb{I}(X > \gamma)$ and $\mathbb{P}(X > \gamma)$ is small, then this set of values would be $\{x \in \mathcal{X} : S(x) > \gamma\}$. We want to find a way to make this 'important' set of values happen more often. This is the idea of *importance sampling*. The idea is to sample $\{X^{(i)}\}_{i=1}^{N}$ according to another density, g, that ascribes much higher probability to the important set. Observe that, given a density g such that $g(x) = 0 \Rightarrow f(x)S(x) = 0$, and being explicit about the density used to calculate the expectation,

$$\begin{aligned}\mathbb{E}_f S(X) &= \int_{-\infty}^{\infty} S(x)\, f(x)\, \mathrm{d}x = \int_{-\infty}^{\infty} S(x)\frac{g(x)}{g(x)} f(x)\, \mathrm{d}x \\ &= \int_{-\infty}^{\infty} S(x)\frac{f(x)}{g(x)} g(x)\, \mathrm{d}x = \mathbb{E}_g \frac{f(X)}{g(X)} S(X).\end{aligned}$$

We call $f(x)/g(x)$ the *likelihood ratio*. This suggests, immediately, the importance sampling estimator

Definition 1.6.1 (The Importance Sampling Estimator)**.** The importance sampling estimator, $\widehat{\ell}_{\mathsf{IS}}$, of $\ell = \mathbb{E}S(X)$ is given by

$$\widehat{\ell}_{\mathsf{IS}} = \frac{1}{N}\sum_{i=1}^{N} \frac{f(X^{(i)})}{g(X^{(i)})} S(X^{(i)}),$$

where the $\{X^{(i)}\}_{i=1}^{N}$ are i.i.d. draws from the importance sampling density g.

Because we wish to use this estimator for variance reduction, it makes sense for us to calculate its variance.

Lemma 1.6.2. The variance of the importance sampling estimator, $\widehat{\ell}_{\mathsf{IS}}$, is given by

$$\mathrm{Var}(\widehat{\ell}_{\mathsf{IS}}) = \frac{1}{N}\left(\mathbb{E}_f\left[\frac{f(X)}{g(X)}S(X)^2\right] - \ell^2\right).$$

Proof. We have that

$$\begin{aligned}
\mathrm{Var}(\widehat{\ell}_{\mathsf{IS}}) &= \frac{1}{N}\mathrm{Var}\left(\frac{f(X)}{g(X)}S(X)\right) \\
&= \frac{1}{N}\left(\mathbb{E}_g\left[\frac{f(X)^2}{g(X)^2}S(X)^2\right] - \left(\mathbb{E}_g\frac{f(X)}{g(X)}S(X)\right)^2\right) \\
&= \frac{1}{N}\left(\left[\int_{-\infty}^{\infty}\frac{f(x)^2}{g(x)^2}S(x)^2\, g(x)\,\mathrm{d}x\right] - \ell^2\right) \\
&= \frac{1}{N}\left(\left[\int_{-\infty}^{\infty}\frac{f(x)}{g(x)}S(x)^2\, f(x)\,\mathrm{d}x\right] - \ell^2\right) \\
&= \frac{1}{N}\left(\mathbb{E}_f\left[\frac{f(X)}{g(X)}S(X)^2\right] - \ell^2\right)
\end{aligned}$$

□

Comparing $\mathrm{Var}(\widehat{\ell})$, the variance of the normal Monte Carlo estimator, to $\mathrm{Var}(\widehat{\ell}_{\mathsf{IS}})$, the variance of the importance sampling estimator, we see that

$$\mathrm{Var}(\widehat{\ell}_{\mathsf{IS}}) < \mathrm{Var}(\widehat{\ell}) \Leftrightarrow \mathbb{E}_f\left[\frac{f(X)}{g(X)}S(X)^2\right] < \mathbb{E}_f S(X)^2.$$

When we are estimating probabilities, $S(x)$ is an indicator function. For example, it could be $S(x) = \mathbb{I}(x > \gamma)$. Then,

$$\mathbb{E}S(X) = \mathbb{P}(X > \gamma) = \mathbb{E}\mathbb{I}(X > \gamma) = \mathbb{E}\mathbb{I}(X > \gamma)^2 = \mathbb{E}S(X)^2,$$

so the condition above reduces to requiring that $\mathbb{E}_f\left[\frac{f(X)}{g(X)}S(X)^2\right] < \ell$.

The above technology is easily combined to problems involving discrete random variables. Just replace integrals with sums and densities with probability mass functions.

Example 1.6.3 (Importance sampling with a normal random variable)**.** Consider the problem of estimating $\ell = \mathbb{P}(X > \gamma)$, where $X \sim \mathsf{N}(0, 1)$.

If γ is big, for example $\gamma = 5$, then ℓ is very small. The standard estimator of $\ell = \mathbb{P}(X > \gamma)$ is

$$\widehat{\ell} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left(X^{(i)} > \gamma\right),$$

where the $\{X^{(i)}\}_{i=1}^{N}$ are i.i.d. $\mathsf{N}(0,1)$ random variables. This is not a good estimator for large γ. We can code this as follows.

Listing 1.17: Matlab code

```
gamma = 5; N = 10^7;
X = randn(N,1);
ell_est = mean(X > gamma)
RE_est = std(X > gamma) / (sqrt(N) * ell_est)
```

For $\gamma = 5$ with a sample size of 10^7, an estimate of the probability is 2×10^{-7} and an estimate of the relative error is 0.7071. So, this problem is a good candidate for importance sampling. An obvious choice of an importance sampling density is a normal density with variance 1 but with mean γ. The likelihood ratio $f(x)/g(x)$ is given by

$$\frac{f(x)}{g(x)} = \frac{\left(\sqrt{2\pi}\right)^{-1} \exp\{-\frac{1}{2}x^2\}}{\left(\sqrt{2\pi}\right)^{-1} \exp\{-\frac{1}{2}(x-\gamma)^2\}} = \exp\left\{\frac{\gamma^2}{2} - x\gamma\right\}.$$

Thus, the estimator will be of the form

$$\widehat{\ell}_{\mathsf{IS}} = \frac{1}{N} \sum_{i=1}^{N} \exp\left\{\frac{\gamma^2}{2} - X^{(i)}\gamma\right\} \mathbb{I}\left(X^{(i)} > \gamma\right),$$

where the $\{X^{(i)}\}_{i=1}^{N}$ are i.i.d. $\mathsf{N}(\gamma,1)$ random variables. The code for this follows.

Listing 1.18: Matlab code

```
gamma = 5; N = 10^7;
X = gamma + randn(N,1);
values = exp(gamma^2 / 2 - X*gamma) .* (X > gamma);
ell_est = mean(values)
RE_est = std(values) / (sqrt(N) * ell_est)
```

For $\gamma = 5$ with a sample size of 10^7 an estimate of the probability is 2.87×10^{-7} and an estimate of the relative error is 7.53×10^{-4}. We can check the true value in this case using the Matlab command

```
1 - normcdf(5)
```

This gives a value of 2.87×10^{-7} which is more or less identical to the value returned by our estimator.

If we have a sum of n independent variables, $X_1, \ldots, X_n$, with densities $f_1, \ldots, f_n$, we can apply importance sampling using densities $g_1, \ldots, g_n$. We would then have a likelihood ratio of the form

$$\prod_{i=1}^{n} \frac{f_i(x)}{g_i(x)}.$$

Everything then proceeds as before.

Example 1.6.4 (A rare event for a random walk)**.** Given a random walk, $\{X_n\}_{n\geq 0}$, with $x_0 = 0$ and $p = 0.4$, what is $\ell = \mathbb{P}(X_{50} > 15)$? We can estimate this in Matlab using standard Monte Carlo.

Listing 1.19: Matlab code

```
N = 10^5; threshold = 15;
n = 50; p = 0.4; X_0 = 0;
X_50 = zeros(N,1);

for i = 1:N
    X = X_0;
    for j = 1:n
        Y = rand <= p;
        X = X + 2*Y - 1;
    end
    X_50(i) = X;
end
ell_est = mean(X_50 > threshold)
RE_est = std(X_50 > threshold) / (sqrt(N) * ell_est)
```

Running this program once, we get an estimated probability of 1.2×10^{-4} and an estimated relative error of 0.29. This is not so great, so we can try using importance sampling. A good first try might be to simulate a random walk, as before, but with another parameter, $q = 0.65$. If we write

$$X_n = \sum_{i=1}^{n} Z_i,$$

then, the original random walk is simulated by generating the $\{Z_i\}_{i\geq 1}$ according to the probability mass function $p\,\mathbb{I}(Z = 1) + (1-p)\,\mathbb{I}(Z = -1)$. Generating the new random walk means generating the $\{Z_i\}_{i\geq 1}$ according to

the probability mass function $q\,\mathbb{I}(Z=1)+(1-q)\,\mathbb{I}(Z=-1)$. This then gives a likelihood ratio of the form

$$\prod_{i=1}^{n}\frac{p\,\mathbb{I}(Z_i=1)+(1-p)\,\mathbb{I}(Z_i=-1)}{q\,\mathbb{I}(Z_i=1)+(1-q)\,\mathbb{I}(Z_i=-1)}=\prod_{i=1}^{n}\left[\frac{p}{q}\mathbb{I}(Z_i=1)+\frac{1-p}{1-q}\mathbb{I}(Z_i=-1)\right].$$

We can implement the estimator in Matlab as follows.

Listing 1.20: Matlab code

```
N = 10^5; threshold = 15;
n = 50; p = 0.4; X_0 = 0; q = 0.65;
X_50 = zeros(N,1); LRs = zeros(N,1);

for i = 1:N
    X = X_0; LR = 1;
    for j = 1:n
        Y = rand <= q;
        LR = LR * (p/q * (Y == 1) + (1-p) / (1 - q) * (Y == 0));
        X = X + 2*Y - 1;
    end
    LRs(i) = LR;
    X_50(i) = X;
end
ell_est = mean(LRs .* (X_50 > threshold))
RE_est = std(LRs .* (X_50 > threshold)) / (sqrt(N) * ell_est)
```

Running this program, we get an estimated probability of 1.81×10^{-4} and a relative error of 0.0059. We can check this makes sense by using the standard Monte Carlo estimator with a much bigger sample size. Using a sample size of $N=10^8$, we get an estimate of 1.81×10^{-4}, confirming the importance sampling gives the right result.

Rules of Thumb for Effective Importance Sampling

Recall the definition of a *moment generating function*.

Definition 1.6.5 (Moment Generating Function)**.** We define the moment generating function of a random variable X by

$$M(\theta)=\mathbb{E}e^{\theta X}.$$

For $\theta=0$, $M(\theta)=1$. However, for other values of θ, it may not be the case that $M(\theta)<\infty$. In order for $M(\theta)$ to be finite for some $\theta\neq0$, the probability of X taking very large (or small) values has to go to

zero exponentially fast. This leads to the definition of *light-tailed* random variables. Usually, people assume that light-tailed means right light-tailed.

Definition 1.6.6 (Light-tailed random variable)**.** We say a random variable X is (right) light-tailed if $M(\theta) < \infty$ for some $\theta > 0$. We say X is left light-tailed if $M(\theta) < \infty$ for some $\theta < 0$.

The rules of thumb, which only apply when dealing with light-tailed random variables, are as follows.

- For sums of i.i.d. random variables, e.g., $\mathbb{P}(X_1 + \cdots + X_n > \gamma)$ choose the importance sampling density, g, so that $\mathbb{E}_g X_1 = \gamma/n$.
- For stopping time problems, e.g., $\mathbb{P}(\tau_A < \tau_B)$ or $\mathbb{P}(\tau_A < \infty)$, where the process is drifting away from A, the set of interest, choose g so that the drift of the stochastic process is reversed. For example, if $A = \{10, 11, \ldots\}$ and $S_n = \sum_{i=1}^n Y_i$, with $Y_i \sim \mathsf{N}(-1, 1)$, then choose g so that $\mathbb{E}_g Z_1 = 1$.

1.6.1 Weighted Importance Sampling

If is often the case that we wish to estimate $\ell = \mathbb{E}_f S(\mathbf{X})$ without knowing everything about the density f. For example, a density can be written as

$$f(\mathbf{x}) = \frac{h(\mathbf{x})}{Z},$$

where Z is the normalizing constant (that is $Z = \int h(\mathbf{x})\,\mathrm{d}\mathbf{x}$). In many realistic applications, we do not know Z, even if we do know h and have a way of sampling from f (it is often possible to sample from a density without knowing its normalizing constant). An obvious problem, then, is how we should carry out importance sampling in such a setting.

Weighted importance sampling is one way of addressing this problem. The normal importance sampling estimator is of the form

$$\widehat{\ell}_{\mathsf{IS}} = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\mathbf{X}^{(i)})}{g(\mathbf{X}^{(i)})} S(\mathbf{X}^{(i)}),$$

where $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(N)}$ is an i.i.d. sample from g. This returns a sample mean that is weighted by the likelihood ratios

$$\frac{f(\mathbf{X}^{(1)})}{g(\mathbf{X}^{(1)})}, \ldots, \frac{f(\mathbf{X}^{(N)})}{g(\mathbf{X}^{(N)})}.$$

The idea of weighted importance sampling is to use another weight, which is of the form

$$W(\mathbf{x}) = \frac{h(\mathbf{x})}{g(\mathbf{x})}.$$

Instead of using the standard importance sampling estimator, we now have to use an estimator of the form

$$\widehat{\ell}_{\mathsf{WIS}} = \frac{\frac{1}{N}\sum_{i=1}^{N} W\left(\mathbf{X}^{(i)}\right) S\left(\mathbf{X}^{(i)}\right)}{\frac{1}{N}\sum_{i=1}^{N} W\left(\mathbf{X}^{(i)}\right)},$$

with $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(N)}$ an i.i.d. sample from g.

Now,

$$\begin{aligned}\mathbb{E}_g W(\mathbf{X}) S(\mathbf{X}) = \mathbb{E}_g \frac{h(\mathbf{X})}{g(\mathbf{X})} S(\mathbf{X}) &= \int \frac{h(\mathbf{x})}{g(\mathbf{x})} S(\mathbf{x}) g(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ &= Z \int \frac{h(\mathbf{x})}{Z} S(\mathbf{x})\, \mathrm{d}\mathbf{x} = Z\, \mathbb{E}_f S(\mathbf{X}).\end{aligned}$$

Likewise,

$$\mathbb{E}_g W(\mathbf{X}) = \mathbb{E}_g \frac{h(\mathbf{X})}{g(\mathbf{X})} = \int \frac{h(\mathbf{x})}{g(\mathbf{x})} g(\mathbf{x})\, \mathrm{d}\mathbf{x} = Z \int \frac{h(\mathbf{x})}{Z}\, \mathrm{d}\mathbf{x} = Z.$$

Thus, by the strong law of large numbers, we get in the limit that

$$\lim_{N\to\infty} \frac{\frac{1}{N}\sum_{i=1}^{N} W\left(\mathbf{X}^{(i)}\right) S\left(\mathbf{X}^{(i)}\right)}{\frac{1}{N}\sum_{i=1}^{N} W\left(\mathbf{X}^{(i)}\right)} \to \frac{Z\, \mathbb{E}_f S(\mathbf{X})}{Z} = \mathbb{E}_f S(\mathbf{X}).$$

Note that this estimator is biased. However, the bias is $O(1/n)$, so it is not too bad in practice (and we don't always have an alternative).

1.6.2 Sequential Importance Sampling

So far, when we have considered a problem like $\ell = \mathbb{P}(X_{10} > 5)$, we have considered a process that can be written as a sum of independent random variables. For example

$$X_n = \sum_{i=1}^{n} Y_i,$$

where $Y_1, \ldots, Y_n$ are i.i.d. draws from f. In this case, we can write

$$\widehat{\ell}_{\mathsf{IS}} = \frac{1}{N} \sum_{i=1}^{N} \left(\prod_{j=1}^{10} \frac{f(Y_j^{(i)})}{g(Y_j^{(i)})} \right) \mathbb{I}\left(\sum_{j=1}^{10} Y_j^{(i)} > 5 \right).$$

One advantage of this formulation is that if we can sample Y_1, calculate the likelihood ratio, then sample Y_2, update the likelihood ratio (by multiplying by $f(Y_2)/g(Y_2)$), and so on. In particular, if we wish to simulate a process until a stopping time, then we can simply stop when this stopping time is reached, without having to worry about how to calculate the joint density afterwards. When the $\{Y_j\}_{j=1}^n$ are dependent, things are a little more complicated. Continuing with our $\ell = \mathbb{P}(X_{10} > 5)$ example, in the case of dependent random variables, we would write

$$\widehat{\ell}_{\mathsf{IS}} = \frac{1}{N}\sum_{i=1}^{N} \frac{f(Y_1^{(i)},\ldots,Y_n^{(i)})}{g(Y_1^{(i)},\ldots,Y_n^{(i)})} \mathbb{I}\left(\sum_{j=1}^{10} Y_j^{(i)} > 5\right).$$

However, we can often write this in a more convenient form. Note that,

$$f(y_1,\ldots,y_n) = f(y_1)f(y_2\,|\,y_1)\cdots f(y_n\,|\,y_1,\ldots,y_{n-1}),$$

or, in Bayesian notation (which makes things a bit more compact),

$$f(y_{1:n}) = f(y_1)f(y_2\,|\,y_1)\cdots f(y_n\,|\,y_{1:n-1})$$

Likewise, we can write

$$g(y_{1:n}) = g(y_1)g(y_2\,|\,y_1)\cdots g(y_n\,|\,y_{1:n-1}).$$

If we know these conditional densities, then we can write the likelihood ratio in the form

$$W_n(y_{1:n}) = \frac{f(y_1)f(y_2\,|\,y_1)\cdots f(y_n\,|\,y_{1:n-1})}{g(y_1)g(y_2\,|\,y_1)\cdots g(y_n\,|\,y_{1:n-1})}.$$

If we write,

$$W_1(y_1) = \frac{f(y_1)}{g(y_1)},$$

then

$$W_2(y_{1:2}) = \frac{f(y_2\,|\,y1)}{g(y_2\,|\,y1)}\frac{f(y_1)}{g(y_1)} = \frac{f(y_2\,|\,y1)}{g(y_2\,|\,y1)}W_1(y_1),$$

and, more generally,

$$W_n(y_{1:n}) = \frac{f(y_n\,|\,y_{1:n-1})}{g(y_n\,|\,y_{1:n-1})}W_{n-1}(y_{1:n-1}).$$

In cases where the Markov property holds,

$$W_n(y_{1:n}) = \frac{f(y_n\,|\,y_{n-1})}{g(y_n\,|\,y_{n-1})}W_{n-1}(y_{1:n-1}).$$

Using this formulation, we can update until a stopping time, then stop updating. This formulation also allows for sophisticated methods, such as those were certain low probability paths (i.e., paths with very small weights) are randomly killed.

1.6.3 Self-Avoiding Random Walks

Consider a random walk on a 2d lattice. That is, a Markov chain, $\{X_n\}_{n\geq 0}$, on $\mathbb{Z}\times\mathbb{Z}$ with $X_0 = 0$ and transition probabilities given by

$$\mathbb{P}\left(X_n = (k,l) \,|\, X_{n-1} = (i,j)\right) = \begin{cases} 1/4 & , \text{ if } |k-i| + |l-j| = 1 \\ 0 & , \text{ otherwise} \end{cases}.$$

Self-avoiding random walks are simply random walks that do not hit themselves.

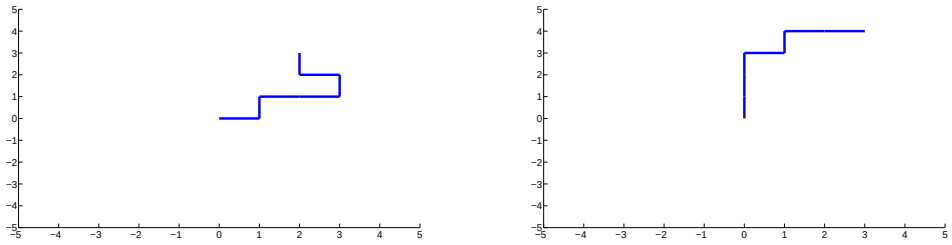

Figure 1.6.1: Two realizations of a self-avoiding random walk with 7 steps.

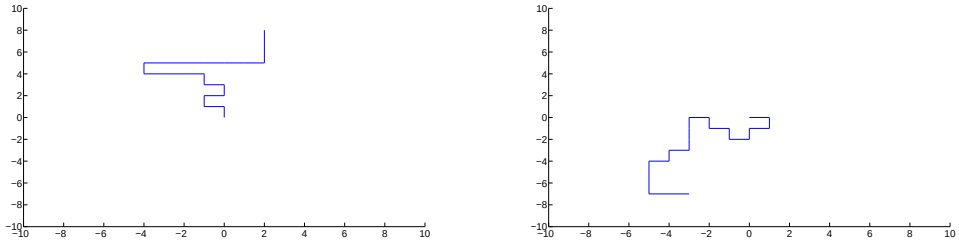

Figure 1.6.2: Two realizations of a self-avoiding random walk with 20 steps.

Self avoiding walks are useful as simple models of objects like polymers. They capture some fundamental behavior of strings of molecules that cannot be too close to one another, but otherwise have minimal interaction. They also appear in mathematical objects like random graphs and percolation clusters.

It is easy to generate a self-avoiding random walk of length n via Monte Carlo if n is small. We simply simulate random walks of length n until one of them is self-avoiding.

Listing 1.21: Matlab code

```
n = 7; i = 1;

while(i ~= n)
    X = 0; Y = 0;
    lattice = zeros(2*n + 1, 2*n+1);
    lattice(n+1, n+1) = 1;
    path = [0 0];
    for i = 1:n

        U = rand;
        if U < 1/2
            X = X + 2 * (U < 1/4) - 1;
        else
            Y = Y + 2 * (U < 3/4) - 1;
        end

        path_addition = [X Y];
        path = [path; path_addition];

        lattice_x = n + 1 + X;
        lattice_y = n + 1 + Y;

        if lattice(lattice_x, lattice_y) == 1
            i = 1; break;
        else
            lattice(lattice_x, lattice_y) = 1;
        end
    end
end

clf; hold on;
axis([-n n -n n]);
for j = 1:n
    line([path(j,1), path(j+1,1)], [path(j,2), path(j+1,2)]);
end
```

The problem is that for large n, it is very unlikely that a random walk will be a self-avoiding random walk. To put this in perspective, there are 4^n possible random walks of length n on the 2D square lattice. In general, the number of self-avoiding random walks for a given n is not known. However, for small n, these have been calculated.

- For $n = 5$, there are 284 self-avoiding walks. So, the probability that a

single random walk of length 5 will be self-avoiding is

$$\frac{284}{4^5} = \frac{284}{1024} \approx 0.2773.$$

This means we only need to generate roughly 4 walks in order to get a self-avoiding one.

- For $n = 10$, there are 441000 self-avoiding random walks. So, the probability is
$$\frac{44100}{4^{10}} = \frac{44100}{1048576} \approx 0.0421.$$
This means we need to generate about 24 walks in order to get a self-avoiding one.

- For $n = 20$, there are 897697164 self-avoiding random walks. The probability is
$$\frac{897697164}{4^{20}} \approx 8.16 \times 10^{-4}.$$
so we need to generate about 1125 walks in order to get a self-avoiding one.

Pretty clearly, the situation becomes unworkable by $n = 100$ or $n = 150$. Unfortunately, people are often interested in asymptotic results when considering objects like self-avoiding random walks. In order to get information about asymptotic behavior, we need to be able to generate statistics for random walks with large n values. An obvious modification to the standard algorithm would be to try to choose the next step of the random walk to avoid the places the random walk has already been. The simplest way to do this is to chose the next site of the random walk from the set of empty neighbors of the current site.

PICTURE HERE

This approach is straightforward to implement in Matlab.

Listing 1.22: Matlab code

```
n = 250; i = 1;
moves = [0 1; 0 -1; -1 0; 1 0];

while(i ~= n)
    X = 0; Y = 0;
    lattice = zeros(2*(n+1) + 1, 2*(n+1)+1);
    lattice(n+2, n+2) = 1;
```

```
    path = [0 0];
    for i = 1:n
        lattice_x = n + 2 + X; lattice_y = n + 2 + Y;

        up = lattice(lattice_x,lattice_y + 1);
        down = lattice(lattice_x,lattice_y - 1);
        left = lattice(lattice_x-1,lattice_y);
        right = lattice(lattice_x+1,lattice_y);
        neighbors = [1 1 1 1] - [up down left right];

        if sum(neighbors) == 0
            i = 1; break;
        end

        direction = ...
            min(find(rand<(cumsum(neighbors)/sum(neighbors))));
        X = X + moves(direction,1);
        Y = Y + moves(direction,2);

        lattice_x = n + 2 + X; lattice_y = n + 2 + Y;
        lattice(lattice_x,lattice_y) = 1;
        path_addition = [X Y];
        path = [path; path_addition];
    end
end

clf; hold on;
axis([-n n -n n]);
for j = 1:n
    line([path(j,1), path(j+1,1)], [path(j,2), path(j+1,2)]);
end
```

This approach does not solve all our problems (it is still possible to a path to die out early), however it significantly increases the length of the self-avoiding walks we are able to generate in a reasonable amount of time. Unfortunately, this approach does not generate self-avoiding walks of length n uniformly. Consider the two self-avoiding random walks of length 5 shown in figures 1.6.3 and 1.6.4. The first has probability $1/4 \times 1/3 \times 1/3 \times 1/3 \times 1/3$ and the second has probability $1/4 \times 1/3 \times 1/3 \times 1/3 \times 1/2$. Basically, the algorithm is biased towards more compact configurations. You can also see this in figure 1.6.5 and figure 1.6.5, which are less spread out than most self-avoiding walks. The obvious way to try to fix this is importance sampling.

The probability mass function for self-avoiding walks starting at $\mathbf{x}_0 =$

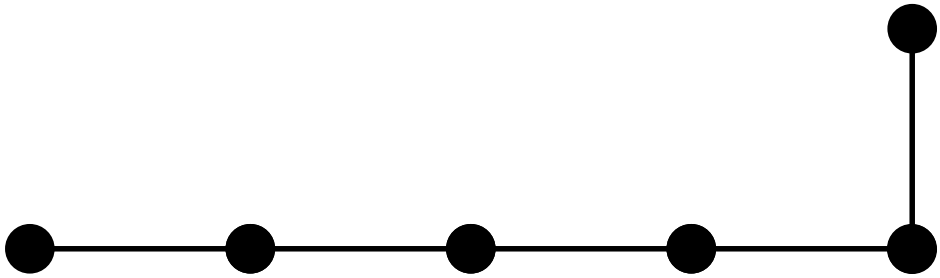

Figure 1.6.3: A path of a self-avoiding walk of length 5.

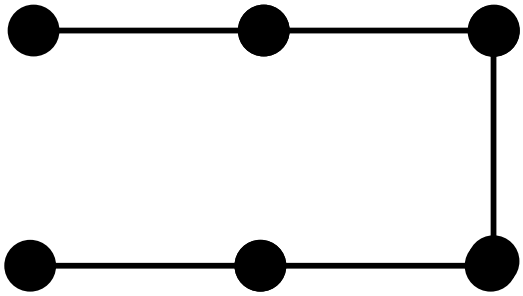

Figure 1.6.4: A path of a self-avoiding walk of length 5.

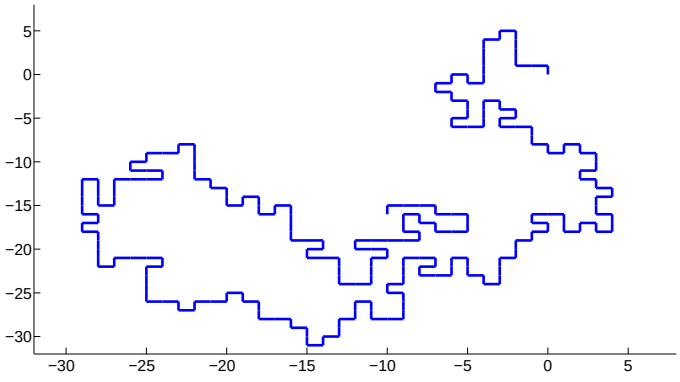

Figure 1.6.5: A realization of a self-avoiding random walk with 250 steps using the new technique.

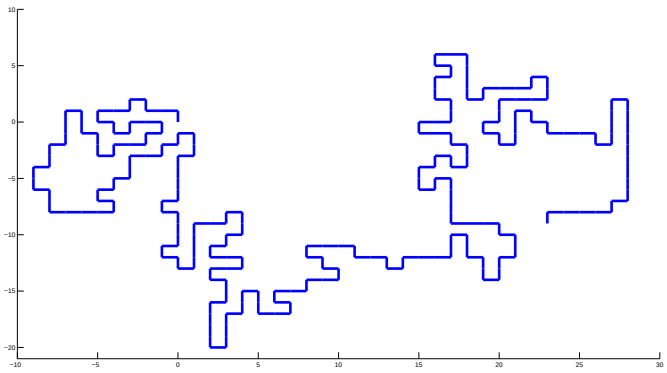

Figure 1.6.6: A realization of a self-avoiding random walk with 250 steps using the new technique.

(x_0, y_0), which we represent by $\mathbf{x}_1 = (x_1, y_1), \ldots, \mathbf{x}_n = (x_n, y_n)$, is given by

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \frac{\mathbb{I}\left((\mathbf{x}_1, \ldots, \mathbf{x}_n) \in E_n\right)}{Z_n},$$

where E_n is the set of self-avoiding random walks of length n. Unfortunately, as mentioned before, we do not know Z_n. However, we can use weighted importance sampling instead. To do this, we still need an expression for $q(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, the probability mass function based on the new method. We can get this expression using sequential importance sampling. Note that, at step $i-1$ of the random walk, we know all the information up to step $i-1$, so we can calculate $q(\mathbf{x}_i \,|\, \mathbf{x}_0, \ldots, \mathbf{x}_{i-1})$. Let d_{i-1} be the number of unoccupied neighbors of $\mathbf{x}_{i-1}$. This is a function of $\mathbf{x}_0, \ldots, \mathbf{x}_{i-1}$. Then,

$$q(\mathbf{x}_i \,|\, \mathbf{x}_0, \ldots, \mathbf{x}_{i-1}) = \begin{cases} 1/d_{i-1}, & \text{if } \mathbf{x}_i \text{ is an unoccupied neighbor of } \mathbf{x}_{i-1} \\ 0, & \text{otherwise} \end{cases}$$

Thus, a successful realization of a self-avoiding random walk under our algorithm, $\mathbf{x}_1, \ldots, \mathbf{x}_n$ will have a probability of

$$q(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \frac{\mathbb{I}\left((\mathbf{x}_1, \ldots, \mathbf{x}_n) \in E_n\right)}{d_0 \cdots d_{n-1}}$$

Note that $p(\mathbf{x}_1, \ldots, \mathbf{x}_n) \propto \mathbb{I}\left((\mathbf{x}_1, \ldots, \mathbf{x}_n) \in E_n\right)$, so we can use weights of the form

$$W(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \mathbb{I}\left((\mathbf{x}_1, \ldots, \mathbf{x}_n) \in E_n\right) d_0 \cdots d_{n-1},$$

in weighted importance sampling.

Estimating Mean Square Extension

One of the classical objects of interest for self-avoiding random walks is mean square extension. Given a self-avoiding random walk of length n, the mean square extension is defined as $\mathbb{E}\left\|\mathbf{X}_n - \mathbf{x}_0\right\|^2$. Starting at $\mathbf{x}_0 = (0,0)$, this is $\mathbb{E}\left\|\mathbf{X}_n\right\|^2 = \mathbb{E}\left\|X_n^2 + Y_n^2\right\|^2$. The standard Monte Carlo estimator of this would be

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{X}_n^{(i)}\right\|^2,$$

where $\mathbf{X}_n^{(1)},\ldots,\mathbf{X}_n^{(N)}$ are i.i.d. draws from $p(\mathbf{x}_1,\ldots,\mathbf{x}_n)$. The weighted importance sampling estimator, using the alternative approach, is

$$\widehat{\ell}_{\mathsf{IS}} = \frac{\frac{1}{N}\sum_{i=1}^{N} d_0^{(i)}\cdots d_{n-1}^{(i)}\left\|\mathbf{X}_n^{(i)}\right\|^2}{\frac{1}{N}\sum_{i=1}^{N} d_0^{(i)}\cdots d_{n-1}^{(i)}},$$

where $\mathbf{X}_n^{(1)},\ldots,\mathbf{X}_n^{(N)}$ are i.i.d. draws from $q(\mathbf{x}_1,\ldots,\mathbf{x}_n)$, and the values $d_1^{(i)},\ldots,d_n^{(i)}$ etc. are functions of the appropriate self-avoiding random walk.

An implementation of the standard Monte Carlo approach is

Listing 1.23: Matlab code

```
N = 10^5; n = 5;
square_extension = zeros(N,1);

for step_i = 1:N
    i = 1;
    while(i ~= n)
        X = 0; Y = 0;
        lattice = zeros(2*n + 1, 2*n+1);
        lattice(n+1, n+1) = 1;
        for i = 1:n
            U = rand;
            if U < 1/2
                X = X + 2 * (U < 1/4) - 1;
            else
                Y = Y + 2 * (U < 3/4) - 1;
            end

            lattice_x = n + 1 + X;
            lattice_y = n + 1 + Y;

            if lattice(lattice_x, lattice_y) == 1
```

```
                i = 1;
                break;
            else
                lattice(lattice_x, lattice_y) = 1;
            end
        end
    end

    square_extension(step_i) = X^2 + Y^2;
end

mean_square_extension = mean(square_extension)
```

An implementation of the importance sampling version is

Listing 1.24: Matlab code

```
N = 10^5; n = 150; square_extension = zeros(N,1);
moves = [0 1; 0 -1; -1 0; 1 0];

for step_i = 1:N
    i = 1;
    while(i ~= n)
        X = 0; Y = 0; weight = 1;
        lattice = zeros(2*(n+1) + 1, 2*(n+1)+1);
        lattice(n+2, n+2) = 1;
        for i = 1:n
            lattice_x = n + 2 + X; lattice_y = n + 2 + Y;
            up = lattice(lattice_x,lattice_y + 1);
            down = lattice(lattice_x,lattice_y - 1);
            left = lattice(lattice_x-1,lattice_y);
            right = lattice(lattice_x+1,lattice_y);
            neighbors = [1 1 1 1] - [up down left right];

            if sum(neighbors) == 0
                i = 1; break;
            end
            weight = weight * sum(neighbors);
            direction = ...
                min(find(rand<(cumsum(neighbors)/sum(neighbors))));
            X = X + moves(direction,1);
            Y = Y + moves(direction,2);
            lattice_x = n + 2 + X; lattice_y = n + 2 + Y;
            lattice(lattice_x,lattice_y) = 1;
        end
```

```
    end
    weights(step_i) = weight;
    square_extension(step_i) = X^2 + Y^2;
end
mean_square_extension = ...
    mean(weights'.*square_extension) / mean(weights)
```

Chapter 2

Poisson Processes and Continuous Time Markov Chains

2.1 Stochastic Processes in Continuous Time

In order to discuss stochastic processes in continuous time, we need to update a few of the definitions we have been working with.

Technically, a stochastic process lives on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Typical choices of Ω are $C[0,\infty)$, the space of all continuous functions on $[0,\infty)$, and $D[0,\infty)$, the space of right continuous left limit (càdlàg) functions. An element, ω, of Ω is then a function. We can then define a stochastic process by $X_t(\omega) = \omega(t)$. We often take $\mathcal{F}$ to be the smallest σ-algebra such that $\omega \to \omega(t)$ is measurable for each $t \geq 0$. We usually work with a set of probability measures $\{\mathbb{P}_x\}_{x\in\mathcal{X}}$, where $\mathbb{P}_x$ is the probability measure of a stochastic process starting at $X_0 = x$ (i.e., $\mathbb{P}_x(X_0 = x) = 1$). I will use $\mathbb{P}$ without a subscript to denote a stochastic process started at 0 (or sometimes something else if the context is explicit enough).

A continuous time, general state space stochastic process takes values in a state space $\mathcal{X}$ that is equipped with an appropriate σ-algebra, Σ. Often, $\mathcal{X}$ will be $\mathbb{R}$ and Σ will then be $\mathcal{B}(\mathbb{R})$, the Borel σ-algebra on $\mathbb{R}$.

The first thing we need is a way of rigorously talking about the history of a process. In the discrete time case, we could simply consider $X_0, \ldots, X_n$. Now, we need something a bit more sophisticated, called a *filtration*.

Definition 2.1.1 (Filtration)**.** A filtration $\{\mathcal{F}_t\}_{t\geq 0}$ is an increasing family of sub σ-algebras of $\mathcal{F}$ (that is, $\mathcal{F}_s \subseteq \mathcal{F}_t \subseteq \mathcal{F}$ for all $s \leq t$).

We will require/assume that a filtration is right continuous. That is,

$$\mathcal{F}_t = \bigcap_{s \geq t} \mathcal{F}_s \quad \forall t \geq 0.$$

We need the filtration to contain some information about $\{X_t\}_{t\geq 0}$ (otherwise, it is not very useful). We say $\{X_t\}_{t\geq 0}$ is adapted to a filtration, $\{\mathcal{F}_t\}_{t\geq 0}$ if the filtration contains sufficient information about $\{X_t\}_{t\geq 0}$. More formally,

Definition 2.1.2. A process, $\{X_t\}_{t\geq 0}$ is adapted to a filtration $\{\mathcal{F}_t\}_{t\geq 0}$, or $\mathcal{F}_t$-adapted, if X_t is $\mathcal{F}_t$ measurable for all $t \geq 0$.

We normally assume we are working the *natural filtration* of a process (augmented so that it is right-continuous).

Definition 2.1.3 (Natural Filtration)**.** The natural filtration of a process, $\{\mathcal{F}_t^X\}_{t\geq 0}$ is the filtration generated by the process itself. That is

$$\mathcal{F}_t^X = \sigma(X_s : 0 \leq s \leq t).$$

If we augment the natural filtration by adding certain sets of measure zero, we will usually (in situations we consider) get a right-continuous filtration. When this is possible, we assume that by natural filtration we mean the augmented natural filtration.

Equipped with the concept of a filtration, we can define stopping times in continuous time.

Definition 2.1.4 (Stopping Time : Continuous Time Version)**.** A random variable $\tau : \Omega \to [0, \infty]$ is said to be a stopping time with respect to the filtration $\{\mathcal{F}_t\}_{t\geq 0}$ if $\{\tau \leq t\} \in \mathcal{F}_t$ for all $t \geq 0$.

We can also define a continuous time general state space version of the Markov property.

Definition 2.1.5 (Markov Property: Continuous Time)**.** We say an $\mathcal{F}_t$-adapted stochastic process $\{X_t\}_{t\geq 0}$ taking values in $\mathcal{X}$ has the Markov property if

$$\mathbb{P}_x(X_{t+s} \in A \,|\, \mathcal{F}_s) = \mathbb{P}_x(X_{t+s} \in A \,|\, X_s) \quad \mathbb{P}_x \text{ almost surely},$$

for all $x \in \mathcal{X}$, $0 \leq s \leq t$ and $A \in \Sigma$.

Likewise, we can define a continuous time version of the strong Markov property.

Definition 2.1.6 (Strong Markov Property: Continuous Time)**.** We say an $\mathcal{F}_t$-adapted stochastic process $\{X_t\}_{t\geq 0}$ taking values in $\mathcal{X}$ has the strong Markov property if, given a stopping time τ and conditional on $\tau < \infty$,

$$\mathbb{P}_x(X_{t+\tau} \in A \,|\, \mathcal{F}_\tau) = \mathbb{P}_x(X_{t+\tau} \in A \,|\, X_\tau) \quad \mathbb{P}_x \text{ almost surely,}$$

for all $x \in \mathcal{X}$, $0 \leq s \leq t$ and $A \in \Sigma$.

2.2 The Poisson Process

The *Poisson process* is one of the fundamental stochastic processes in probability. We can use it to build many complex and interesting stochastic objects. It is a very simple model for processes such as the arrival of people in a queue, the number of cars arriving at a red traffic light, and the occurrence of insurance claims.

2.2.1 Point Processes on $[0, \infty)$

A Poisson process is a *point process* on $[0, \infty)$. We will not consider such point processes in their most general form, but rather a straightforward and easy to work with subset of point processes. We can use a number of equivalent definitions. The first of these is to see the point process in terms of a counting process. A counting process, $\{N_t\}_{t\geq 0}$ counts the number of events that have happened by time t.

Definition 2.2.1 (Point Process on $[0, \infty)$: Counting Process Version)**.** A point process on $[0, \infty)$ is a process $\{N_t\}_{t\in[0,\infty)}$ taking values in $\mathbb{N}$ that:

(i) Vanishes at 0. That is, $N_{0^-} = N_0 = 0$, where $N_{t^-} = \lim_{u\uparrow t} N_u$.

(ii) Is non-decreasing. That is, $N_0 \leq N_s \leq N_t$ for $0 \leq s \leq t$.

(iii) Is right continuous. That is, $N_t = N_{t^+}$, where $N_{t^+} = \lim_{u\downarrow t} N_u$.

(iv) Has unit jumps. That is, $N_t - N_{t^-} \in 0, 1$. Technically, a process where only one jump can occur at a given time t is called a *simple point process*.

(v) Has an infinite limit. That is, $\lim_{t\to\infty} N_t = \infty$.

Note, again, that some of these requirements can be relaxed.

Definition 2.2.2 (Point Process on $[0,\infty)$: Jump Instances Version)**.** A point process on $[0,\infty)$ is defined by a sequence $\{T_n\}_{n\geq 1}$ of random variables that are positive and increasing to infinity. That is,

$$0 < T_1 < T_2 < \cdots < \infty \quad \text{and} \quad \lim_{n\to\infty} T_n = \infty.$$

This defines the counting process

$$N_t = \sum_{n\geq 1} \mathbb{I}(T_n \leq t) = \sup\{n \geq 1 : T_n \leq t\}.$$

Definition 2.2.3 (Point Process on $[0,\infty)$: Inter-arrivals Version)**.** A point process on $[0,\infty)$ is defined by a sequence $\{S_n\}_{n\geq 1}$ of positive random variables such that $\sum_{n\geq 1} S_n = \infty$. These define a sequence of jump instances by $S_1 = T_1$ and $S_n = T_n - T_{n-1}$ for $n \geq 2$.

PICTURE HERE

These three definitions of point processes suggest three possible ways to simulate them.

(i) We can simulate the counting process $\{N_t\}_{t\geq 0}$ (or its increments).

(ii) We can simulate the jump times $\{T_n\}_{n\geq 1}$.

(iii) We can simulate the inter-arrival times $\{S_n\}_{n\geq 1}$.

The key properties of a Poisson process (aside from being a point process) are that it has *stationary increments* and *independent increments*.

Definition 2.2.4 (Stationary Increments)**.** A stochastic process $\{X_t\}_{t\geq 0}$ has stationary increments if the distribution of $X_{t+h} - X_t$ depends only on h for $h \geq 0$.

Definition 2.2.5 (Independent Increments)**.** A stochastic process $\{X_t\}_{t\geq 0}$ has independent increments if the random variables $\{X_{t_{i+1}} - X_{t_i}\}_{i=1}^n$ are independent whenever $0 \leq t_1 < t_2 < \cdots < t_n$ and $n \geq 1$.

A process that has stationary and independent increments is attractive from a simulation perspective because we can simulate it in 'parts' (the increments).

2.2.2 Poisson Process

Equipped with the necessary definitions, we can now define a Poisson process.

Definition 2.2.6 (Poisson Process on $[0,\infty)$)**.** A (homogenous) Poisson process on $[0,\infty)$ with parameter $\lambda > 0$ is a point process on $[0,\infty)$ with stationary and independent increments and $N_t \sim \mathsf{Poi}(\lambda t)$ for all $t \geq 0$. That is,

$$\mathbb{P}(N_t = k) = e^{-\lambda t}\frac{(\lambda t)^k}{k!}.$$

Actually, we can define a Poisson process in a number of different ways.

Theorem 2.2.7 (Poisson process)**.** The following definitions are equivalent definitions.

(i) A Poisson process is a point process, $\{N_t\}_{t\geq 0}$, with stationary and independent increments with $N_t \sim \mathsf{Poi}(\lambda t)$ for all $t \geq 0$.

(ii) A Poisson process is a point process, $\{N_t\}_{t\geq 0}$, with independent increments and the property that, as $h \downarrow 0$, uniformly in t

  (a) $\mathbb{P}(N_{t+h} - N_t = 0) = 1 - \lambda h + o(h)$.
  (b) $\mathbb{P}(N_{t+h} - N_t = 1) = \lambda h + o(h)$.
  (c) $\mathbb{P}(N_{t+h} - N_t > 1) = o(h)$.

(iii) A Poisson process is a point process defined by its inter-arrival times, $\{S_n\}_{n\geq 1}$, which are i.i.d. $\mathsf{Exp}(\lambda)$.

Before we prove anything, we need a few results.

Lemma 2.2.8 (Memoryless Property)**.** We say that exponential random variables have the *memoryless property*. That is, for $t, s \geq 0$, if $X \sim \mathsf{Exp}(\lambda)$, then

$$\mathbb{P}(X > t + s \,|\, X > s) = \mathbb{P}(X > t).$$

Proof. We can write $\mathbb{P}(X > t + s \,|\, X > s)$ as

$$\frac{\mathbb{P}(X > t + s, X > s)}{\mathbb{P}(X > s)}.$$

As $s < t$ then $\mathbb{P}(X > t + s, X > s) = \mathbb{P}(X > t)$, so

$$\frac{\mathbb{P}(X > t + s, X > s)}{\mathbb{P}(X > s)} = \frac{\mathbb{P}(X > t + s)}{\mathbb{P}(X > s)} = \frac{e^{-\lambda(t+s)}}{e^{-\lambda s}} = e^{-\lambda(t)} = \mathbb{P}(X > t).$$

□

Theorem 2.2.9 (Markov Property)**.** If $\{N_t\}_{t\geq 0}$ is a Poisson process with rate $\lambda > 0$, then, for any $s > 0$, $\{N_{t+s} - N_s\}_{t\geq 0}$ is also a Poisson process with rate λ. Furthermore, this process is independent of $\{N_r\}_{r\leq s}$.

Proof. First, note that the event $\{N_s = i\}$ can be written as

$$\{N_s = i\} = \{T_i \leq s\} \cap \{S_{i+1} > s - T_i\}$$

and, given this, for $r \leq s$,

$$N_r = \sum_{j=1}^{i} \mathbb{I}(S_j \leq r).$$

Define the process starting at time s by $\widetilde{N}_t = N_{t+s} - N_s$. Then, given $\{N_s = i\}$, $\widetilde{S}_1 = S_{i+1} - (s - T_i)$ and $\widetilde{S}_n = S_{i+n}$ for $n \geq 2$. Now, by the memoryless property, $\widetilde{S}_1$ is an $\mathsf{Exp}(\lambda)$ random variable. Thus the $\{\widetilde{S}_n\}_{n\geq 1}$ are i.i.d. $\mathsf{Exp}(\lambda)$ random variables independent of $S_1, \ldots, S_n$. Thus, conditional of $\{N_s = i\}$, $\{\widetilde{N}_t\}_{t\geq 0}$ is a Poisson process independent of $\{X_r\}_{r\leq s}$. □

Now, we can prove Theorem 2.2.7.

Proof. I give a number of proofs of equivalences here. The rest will be left for exercises or self-study.

Part 1. First, we will show that (i) implies (ii). Now, as the increments are stationary, we know $N_{t+h} - N_t$ has the same distribution as $N_h - N_0$. So,

$$\mathbb{P}(N_{t+h} - N_t = 0) = \mathbb{P}(N_h - N_0 = 0) = \mathbb{P}(N_h = 0) = e^{-\lambda h}.$$

Taking a Taylor expansion of the exponential function, we get $e^{-\lambda h} = 1 - \lambda h + o(h)$. Likewise,

$$\mathbb{P}(N_{t+h} - N_t = 1) = \mathbb{P}(N_h = 1) = \lambda h e^{-\lambda h} = \lambda h + o(h).$$

and $\mathbb{P}(N_{t+h} - N_t > 1) = o(h)$.

Part 2. We will show that (ii) implies (i). We do this by solving some differential equations. First, let us define $p_j(t) = \mathbb{P}(N_t = j)$. Then,

$$p_0(t+h) = \mathbb{P}(N_{t+h} = 0) = \mathbb{P}(N_{t+h} - N_t = 0)\mathbb{P}(N_t = 0) = (1 - \lambda h + o(h))p_0(t).$$

Rearranging, we have

$$\frac{p_0(t+h) - p_0(t)}{h} = -\lambda p_0(t) + \frac{o(h)}{h}.$$

Because this holds for all t, we also get

$$\frac{p_0(t) - p_0(t-h)}{h} = -\lambda p_0(t-h) + \frac{o(h)}{h}.$$

Letting $h \to 0$ shows that $p_0(t)$ has a derivative (as the limit exists). This gives us

$$p_0(t)' = -\lambda p_0(t) \Rightarrow p_0(t) = Ae^{-\lambda t}.$$

Now, because $N_0 = 0$, we know that $p_0(0) = 1$ so $A = 1$ (i.e., $p_0(t) = e^{-\lambda t}$). Doing the same for $p_j(t)$ we have

$$\begin{aligned} p_j(t+h) &= \sum_{i=0}^{j} \mathbb{P}(N_{t+h} - N_t = i)\mathbb{P}(N_t = j-i) \\ &= \mathbb{P}(N_{t+h} - N_t = 0)\mathbb{P}(N_t = j) + \mathbb{P}(N_{t+h} - N_t = 1)\mathbb{P}(N_t = j-1) \\ &\quad + \sum_{i=2}^{j} \mathbb{P}(N_{t+h} - N_t = i)\mathbb{P}(N_t = j-i) \\ &= (1 - \lambda h + o(h))p_j(t) + (\lambda h + o(h))p_{j-1}(t) + o(h). \end{aligned}$$

Rearranging, we have

$$\frac{p_j(t+h) - p_j(t)}{h} = -\lambda p_j(t) + \lambda p_{j-1}(t) + \frac{o(h)}{h}.$$

By a similar argument to the one above, we get

$$p_j(t)' = -\lambda p_j(t) + \lambda p_{j-1}(t).$$

Now, remember that the product rule tells us that $(fg)' = f'g + g'f$. If we use the integrating factor $e^{\lambda t}$, we get

$$\begin{aligned} &p_j(t)' = -\lambda p_j(t) + \lambda p_{j-1}(t) \Rightarrow e^{\lambda t}p_j(t) = -\lambda e^{\lambda t}p_j(t) + \lambda e^{\lambda t}p_{j-1}(t) \\ &\Rightarrow e^{\lambda t}p_j(t)' + \lambda e^{\lambda t}p_j(t) = \lambda e^{\lambda t}p_{j-1}(t) \Rightarrow \left(e^{\lambda t}p_j(t)\right)' = \lambda e^{\lambda t}p_{j-1}(t). \end{aligned}$$

We can solve this by induction. We start with $j = 1$. This gives

$$\left(e^{\lambda t}p_1(t)\right)' = \lambda e^{\lambda t}p_0(t) = \lambda e^{\lambda t}e^{-\lambda t} = \lambda.$$

Integrating, we get

$$e^{\lambda t}p_1(t) = \lambda t + A \Rightarrow p_1(t) = t\lambda e^{\lambda t} + Ae^{\lambda t}.$$

Now, $p_j(0) = 0$ for $j > 0$, so $A = 0$ and $p_1(t) = t\lambda e^{\lambda t}$. Repeating in this way, we get

$$p_j(t) = \mathbb{P}(N_t = j) = e^{-\lambda t}\frac{(\lambda t)^j}{j!}.$$

Part 3. We show that (iii) implies (ii). This is just like the proof that (i) implies (ii). Using the Markov property (which was based on the interarrivals definition) we observe that $N_{t+h} - N_t$ has the same distribution as N_h, so

$$\mathbb{P}(N_{t+h} - N_t = 0) = \mathbb{P}(N_h = 0) = \mathbb{P}(S_1 > h) = e^{-\lambda h} = 1 - \lambda h + o(h),$$

and

$$\begin{aligned}&\mathbb{P}(N_{t+h} - N_t = 1) = \mathbb{P}(N_h = 0) = \mathbb{P}(S_1 < h, S_1 + S_2 > h)\\ &= \int_0^h e^{-\lambda(h-u)}\lambda e^{-\lambda u}\,\mathrm{d}u = \int_0^h \lambda e^{-\lambda h}\,\mathrm{d}u = \lambda h e^{-\lambda h} = \lambda h + o(h).\end{aligned}$$

It is also straightforward to see that

$$\mathbb{P}(N_{t+h} - N_t > 1) = \mathbb{P}(S_1 < h, S_1 + S_2 < h) \leq \mathbb{P}(S_1 < h)\mathbb{P}(S_2 < h) = o(h).$$

□

2.2.3 Order Statistics and the Distribution of Arrival Times

Order Statistics

Consider identically distributed random variables $X_1, \ldots, X_n$ with distribution $F(x)$ (that is $\mathbb{P}(X < x) = F(x)$. The *order statistics* of these random variables are simply the random variables ordered from smallest to largest. We write these as $X_{(1)} \leq X_{(2)} \leq \ldots \leq X_{(n)}$. Two of the most important order statistics are the minimum, $X_{(1)}$, and the maximum, $X_{(n)}$.

Lemma 2.2.10. The distribution of the minimum is given by

$$\mathbb{P}\left(X_{(1)} \leq x\right) = 1 - (1 - F(x))^n.$$

Proof. We have

$$\begin{aligned}&\mathbb{P}\left(X_{(1)} \leq x\right) = 1 - \mathbb{P}\left(X_{(1)} > x\right) = 1 - \mathbb{P}(X_1 > x, \ldots, X_n > x)\\ &= 1 - \mathbb{P}(X_1 > x)\cdots\mathbb{P}(X_n > x) = 1 - (\mathbb{P}(X > x))^n = 1 - (1 - F(x))^n.\end{aligned}$$

□

Lemma 2.2.11. The distribution of the maximum is given by

$$\mathbb{P}\left(X_{(n)} \leq x\right) = F(x)^n$$

Proof. We have

$$\mathbb{P}\left(X_{(n)} \leq x\right) = \mathbb{P}(X_1 \leq x, \ldots, X_n \leq x) = \mathbb{P}(X \leq x)^n = F(x)^n.$$

□

Lemma 2.2.12. The density of the order statistics of n random variables $U_1, \ldots, U_n$, with distribution $\mathcal{U}(a, b)$, is given by

$$f(u_1, \ldots, u_n) = \frac{n!}{(b-a)^n}\mathbb{I}(a \leq u_1 \leq \cdots \leq u_n \leq b)$$

Proof. It is easiest to begin with the distribution and then take derivatives to get the density. We wish to calculate $\mathbb{P}\left(U_{(1)} \leq u_1, \ldots, U_{(n)} \leq u_n\right)$. Note that there are $n!$ orderings of the uniform random variables, each of which is equally likely. So,

$$\begin{aligned}\mathbb{P}\left(U_{(1)} \leq u_1, \ldots, U_{(n)} \leq u_n\right) &= n!\,\mathbb{P}(U < u_1)\cdots\mathbb{P}(U < u_n) \\ &= n!\frac{u_1 - a}{b-a}\cdots\frac{u_n - a}{b-a}.\end{aligned}$$

Taking derivatives, we get

$$\begin{aligned}f(u_1, \ldots, u_n) &= \frac{\partial}{\partial u_1}\cdots\frac{\partial}{\partial u_n}\mathbb{P}\left(U_{(1)} \leq u_1, \ldots, U_{(n)} \leq u_n\right) \\ &= \frac{n!}{(b-a)^n}\mathbb{I}(a \leq u_1 \leq \cdots \leq u_n \leq b).\end{aligned}$$

□

Distribution of Arrival Times

As it turns out, given that we know how many arrivals a Poisson process has had in an interval $[0, t]$ (that is, we know N_t), the arrival times will be uniformly distributed in the interval. This implies that the points of a Poisson process have very little structure to them (in some sense, it is a process that puts points as arbitrarily as possible on a line).

Theorem 2.2.13. Let $\{N_t\}_{t\geq 0}$ be a Poisson process. Then, conditional on $\{N_t = n\}$, $T_1, \ldots, T_n$ have the joint density function

$$f(t_1, \ldots, t_n) = \frac{n!}{t^n}\mathbb{I}(0 \leq t_1 \leq \cdots \leq t_n \leq t).$$

This is the density of the order statistics of i.i.d. uniform random variables on the interval $[0, t]$. This means the arrival times are distributed uniformly on this interval.

Proof. Consider the event $\{T_1 = t_1, \ldots, T_n = t_n, N_t = n\}$. Because there is a bijection between arrival times and inter-arrival times, this should have the same probability density as the event

$$\{S_1 = t_1, S_2 = t_2 - t_1, \ldots, S_n = t_n - t_{n-1}, S_{n+1} > t - t_n\}.$$

Because this is the joint density of i.i.d. exponential random variables, we can write this explicitly as

$$\lambda e^{-\lambda u_1} \lambda e^{-\lambda(u_2 - u_1)} \cdots \lambda e^{-\lambda(u_n - u_{n-1})} e^{-\lambda(t-u_n)} = \lambda^n e^{-\lambda t}.$$

We then get the conditional density we wish by dividing by the probability of the event $\{N_t = n\}$,

$$\begin{aligned} f(t_1, \ldots, t_n) &= \frac{\lambda^n e^{-\lambda t}}{(\lambda t)^n e^{-\lambda t}/n!} \mathbb{I}(0 \le t_1 \le \cdots \le t_n \le t) \\ &= \frac{n!}{t^n} \mathbb{I}(0 \le t_1 \le \cdots \le t_n \le t). \end{aligned}$$

□

2.2.4 Simulating Poisson Processes

As already mentioned, there are at least three ways of simulating a Poisson process. These follow directly from the different definitions we have used.

Using the Infinitesimal Definition to Simulate Approximately

The infinitesimal definition gives us a way to simulate the counting process $\{N_t\}_{t \ge 0}$ directly. This simulation is approximate but becomes increasingly good as $h \downarrow 0$. The idea is to slice the interval $[0, t]$ up into little pieces of length (sometimes called mesh size) h. In each one of these slices, we increase $\{N_t\}_{t \ge 0}$ with probability λh.

Listing 2.1: Matlab code

```
lambda = 4; t = 1; h = 0.0001;
mesh = 0:h:t; N = zeros(1, length(mesh));
S = []; jump_indices = [];

N(1) = 0;

for i = 2:length(mesh)
    if rand < lambda * h
```

```
        jump_indices = [jump_indices i];
        N(i) = N(i-1) + 1;
    else
        N(i) = N(i-1);
    end
end

if isempty(jump_indices)==0
    Ts = (jump_indices - 1)*h;
    S(1) = Ts(1);
    if length(jump_indices) > 1
        for i = 2:length(jump_indices)
            S(i) = Ts(i) - Ts(i-1);
        end
    end
end
```

Simulating the Arrival Times

The idea of this approach is to simulate the arrival times directly. Given an interval $[0, t]$, we know that we have a $\mathsf{Poi}(\lambda t)$ random number of arrivals. These are then distributed uniformly in $[0, t]$.

Listing 2.2: Matlab code

```
t = 5; lambda = 2;

T = [];
n = poissrnd(lambda * t);

if n~=0
    T = sort(t * rand(n,1));
    S = zeros(n,1);
    S(1) = T(1);
    if n > 1
        for i = 2:n
            S(i) = T(i) - T(i-1);
        end
    end
end
```

Simulating the Inter-Arrival Times

The idea here is to simulate the inter-arrival times, which are i.i.d. $\mathsf{Exp}(\lambda)$ random variables. Note that, if $U \sim \mathcal{U}(0,1)$, then $-\log(U)/\lambda$ is $\mathsf{Exp}(\lambda)$.

Listing 2.3: Matlab code

```
lambda = 4; h = 0.0001; t = 1;

s = 0;

S = []; Ts = [];

while s <= t
    inter_time = - log(rand) / lambda;;
    s = s + inter_time;
    if s > t
        break;
    else
        Ts = [Ts s];
        S = [S inter_time];
    end
end
```

2.2.5 Inhomogenous Poisson Processes

For many of the processes that Poisson processes are used to model, such as queues and traffic, the assumption that events occur at a constant rate (i.e., λ is constant) is very unrealistic. If you think about traffic (either on the internet or on a road) it tends to be heavier in some time periods and lighter in others. Inhomogenous Poisson processes modify the definition of a Poisson process so that it can incorporate time-dependent arrivals. Inhomogenous Poisson processes can be defined in a number of ways. Note, however, that it is no longer straightforward to use a definition based on inter-arrival times.

Definition 2.2.14 (Inhomogenous Poisson Process)**.** A point process $\{N_t\}_{t\geq 0}$ is said to be an inhomogenous Poisson process with intensity function $\lambda(t) \geq 0 \, \forall t \geq 0$.

(i) $N_0 = 0$.

(ii) For each $t \geq 0$, N_t has a Poisson distribution with paramater $\Lambda = \int_0^t \lambda(s)\,\mathrm{d}s$.

(iii) For each $0 \leq t_1 < t_2 < \cdots < t_m$, the random variables $N_{t_1}, \ldots, N_{t_m} - N_{t_{m-1}}$ are independent (that is, $\{N_t\}_{t\geq 0}$ has independent increments).

Definition 2.2.15 (Inhomogenous Poisson Process (Infinitesimal Definition))**.** A point process $\{N_t\}_{t\geq 0}$ is said to be an inhomogenous Poisson process with intensity function $\lambda(t) \geq 0 \forall t \geq 0$ if, as $h \downarrow 0$,

(i) $\{N_t\}_{t\geq 0}$ has independent increments.

(ii) $\mathbb{P}(N_{t+h} - N_t = 0) = 1 - \lambda(t)h + o(h)$.

(iii) $\mathbb{P}(N_{t+h} - N_t = 1) = \lambda(t)h + o(h)$.

(iv) $\mathbb{P}(N_{t+h} - N_t > 1) = o(h)$.

2.2.6 Simulating an Inhomogenous Poisson Process

There are at least two ways to simulate an inhomogenous Poisson process.

Acceptance-Rejection

One way to simulate an inhomogenous Poisson process on an interval $[0, t]$ is to simulate a homogenous Poisson process with parameter

$$\lambda^* = \max\{\lambda(s) : 0 \leq s \leq t\}$$

then 'thin' this process by only accepting arrivals with a certain probability. If an arrival / jump occurs at time T_i it should only be accepted with probability $\lambda(T_1)/\lambda^*$. It is not hard to check that this method works using the infinitesimal definition.

Listing 2.4: Matlab code

```
t = 10; lambda_star = 1;

T = [];
n = poissrnd(lambda_star * t);

if n~=0
    point_count = 0;
    for i = 1:n
        T_temp = t * rand;
        if rand < sin(T_temp)^2 / lambda_star
            point_count = point_count + 1;
            T(point_count) = T_temp;
```

```
        end
    end
    if point_count ~= 0
        T = sort(T);
        S = zeros(point_count,1);
        S(1) = T(1);
        if point_count > 1
            for i = 2:point_count
                S(i) = T(i) - T(i-1);
            end
        end
    end
end
```

Infinitesimal Approach (Approximate)

As in the homogenous Poisson process case, we can simulate an inhomogenous Poisson process approximately using its infinitesimal definition. This approximation becomes better as $h \downarrow 0$.

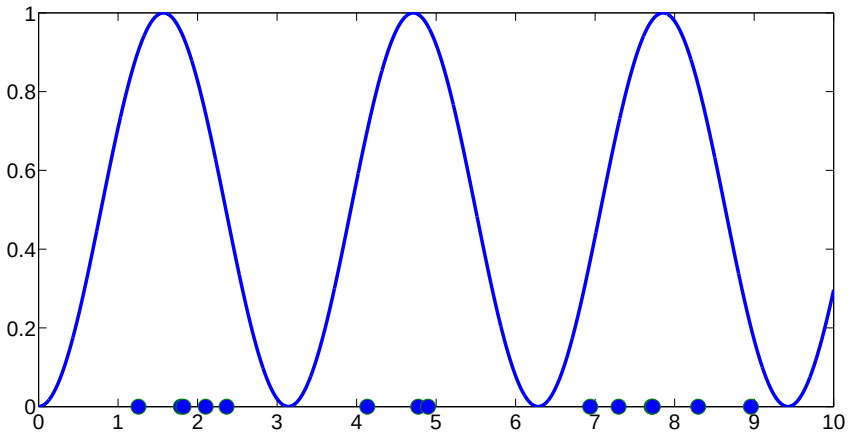

Figure 2.2.1: A realization of a inhomogenous Poisson process with the intensity function $\lambda(t) = 3\sin^2(t)$ plotted.

Listing 2.5: Matlab code

```
lambda = 1; t = 10; h = 0.0001;
mesh = 0:h:t; N = zeros(1, length(mesh));
S = []; jump_indices = [];
```

```
4
5  N(1) = 0;
6
7  for i = 2:length(mesh)
8      if rand < 3*sin(h*(i-1))^2 * h
9          jump_indices = [jump_indices i];
10         N(i) = N(i-1) + 1;
11     else
12         N(i) = N(i-1);
13     end
14 end
15
16 if isempty(jump_indices)==0
17     Ts = (jump_indices - 1)*h;
18     S(1) = Ts(1);
19     if length(jump_indices) > 1
20         for i = 2:length(jump_indices)
21             S(i) = Ts(i) - Ts(i-1);
22         end
23     end
24 end
```

2.2.7 Compound Poisson Processes

A very useful extension of a Poisson process is what is called a *compound Poisson process*. A compound Poisson process replaces the unit jumps of a homogenous Poisson process with random jump sizes.

Definition 2.2.16 (Compound Poisson Process)**.** Given a homogenous Poisson process, $\{N_t\}_{t\geq 0}$ and a jump distribution G, we say

$$X_t = \sum_{i=1}^{N_t} J_i,$$

is a compound Poisson process, where the jumps $\{J_n\}_{n\geq 0}$ are i.i.d. draws from the distribution G.

Example 2.2.17. A street musician plays the accordion in the main street of Ulm for three hours. He hopes to earn enough for a beer, which costs €3.50. Throughout the three hours, people give him coins at random. There does not seem to be any pattern to when people give him money, so a Poisson

process is a good model. The amount of money each person gives is random, with distribution

$$\begin{aligned}\mathbb{P}(€0.05) &= 2/5\\ \mathbb{P}(€0.10) &= 2/5\\ \mathbb{P}(€0.20) &= 1/5.\end{aligned}$$

On average, 5 people per hour give the street musician money. This implies that the Poisson process has intensity $\lambda = 5$. What is the probability the musician gets his beer? That is, what is $\ell = \mathbb{P}(X_3 \geq 3.50)$. We can estimate this easily using Monte Carlo.

Listing 2.6: Matlab code

```
t = 3; lambda = 5; N = 10^6;
beer = zeros(N,1); beer_price = 350;

for i = 1:N

    n = poissrnd(lambda * t);

    if n~=0
        coins = zeros(n,1);
        for j = 1:n
            U = rand;
            coins(j) = (U <= 2/5)*5 + ...
                (U > 2/5 && U <= 4/5)*10 + (U > 4/5)*20;
        end
    end

    beer(i) = (sum(coins) >= beer_price);
end

ell_hat = mean(beer)
re_hat = std(beer) / (ell_hat * sqrt(N))
```

2.3 Continuous Time Markov Chains

Continuous time Markov chains (CTMCS) — or, at least, the ones we consider — behave much like discrete time Markov chains, with the key difference that the the jumps between states take place at random times rather than at fixed steps. Continuous time Markov chains are trickier to

work with than discrete time Markov chains, because there are a number of technical issues and strange / pathological things that can happen. This behavior is mainly related to situations where an infinite number of jumps can happen in a finite time. In this course, we will not consider chains where such things happen. This is because these chains can not really be simulated and considering a smaller subset of continuous time Markov chains will be sufficient for modeling most real world phenomena we might be interested in.

The chains we consider can be described in two different but equivalent ways: *transition functions* and *infinitesimal genertors*. Warning: these descriptions are not always equivalent or even valid when working with more general classes of CTMCs.

2.3.1 Transition Function

Transition functions are the continuous time equivalent of the transition matrix, P, that we have already encountered when discussing discrete time Markov chains. That is, they are functions that allow us to calculate probabilities of the form $\mathbb{P}_i(X_t = j)$. More formally, a transition function is defined as follows.

Definition 2.3.1 (Transition function)**.** A transition function $p_t(i, j)$ with $i, j \in \mathcal{X}$ and $t \geq 0$ is a real-valued function satisfying the following properties

(i) $p_t(i, j) \geq 0$ for all $i, j \in \mathcal{X}$ and $t \geq 0$.

(ii) $\sum_{j \in \mathcal{X}} p_t(i, j) = 1$ for all $i \in \mathcal{X}$ and $t \geq 0$.

(iii) $\lim_{t \downarrow 0} p_t(i, i) = p_0(i, i) = 1$ for all $i \in \mathcal{X}$.

(iv) The Chapman-Kolmogorov equations:

$$p_{s+t}(i, j) = \sum_{k \in \mathcal{X}} p_s(i, k) p_t(k, j)$$

for all $i, j \in \mathcal{X}$ and $t \geq 0$.

We can think of transitions functions as a family of matrices, $\{P(t)\}_{t \geq 0}$, indexed by t. For a fixed value of t, $(P(t))_{i,j} = p_t(i, j)$. We can write the properties out again in terms of matrices:

Definition 2.3.2 (Transition function: Matrix Version)**.** A transition function is a family of matrices, $\{P(t)\}_{t \geq 0}$ with the following properties.

(i) They are non-negative, real-valued and $\sum_{j\in\mathcal{X}} P_{i,j} = 1$ for all $i \in \mathcal{X}$ and $t \geq 0$. Matrices that are non-negative with unit row sums are called *stochastic matrices*.

(ii) They satisfy the Chapman-Kolmogorov equations: $P(s+t) = P(s)P(t)$.

2.3.2 Infinitesimal Generator

Definition 2.3.3. The infinitesimal generator (or Q-matrix) is a real valued matrix satisfying the following properties

(i) $q_{i,j} \geq 0$ for all $i \neq j$.

(ii) $\sum_{j\in\mathcal{X}} q_{i,j} = 0$.

Thus, we require that $q_{i,i} = -\sum_{j\neq i} q_{i,j}$. Because it plays an important role in our treatment of CTMCs, we denote $q_i = -q_{i,i}$.

INF DEF. COMPETING POISSON PROCESS VERSION.

EXAMPLE.

2.3.3 Continuous Time Markov Chains

FORMALLY. OUR VERSION.

2.3.4 The Jump Chain and Holding Times

If we only consider the sequence of states that a CTMC visits (by looking at the value the CTMC takes immediately after each jump), we get a discrete time Markov chain $\{Y_n\}_{n\geq 0}$ called the *jump chain* of the CTMC. We call the time spent in each state a *holding time*. Associate with a path of the jump chain $Y_0, \ldots, Y_n$ is a sequence of holding times $S_1, \ldots, S_{n+1}$, where S_1 is the time spent in Y_0 before jumping to Y_1, S_2 is the time spent in Y_1 before jumping to Y_2, and so on. The definitions of the jump chain and holding times are made clear in figure (REF).

PICTURE HERE

The jump chain $\{Y_n\}_{n\geq 0}$ is a discrete time Markov chain, so it can be described by a transition matrix J (we do not use P in order to avoid confusion with the transition function P). As it turns out, J can be written in terms of the Q-matrix.

Lemma 2.3.4. Given a CTMC with infinitesimal generator Q, the jump matrix J is defined by

$$J_{i,i} = 0$$

for all $i \in \mathcal{X}$ and

$$J_{i,j} = \frac{q_{i,j}}{q_i}$$

for all $i, j \in \mathcal{X}$ such that $j \neq i$.

The holding times are also defined by the Q-matrix.

Lemma 2.3.5. Given $Y_0, Y_1, \ldots, Y_n$, the holding times $S_1, \ldots, S_{n+1}$ are exponential random variables with parameters $q_{Y_0}, q_{Y_1}, \ldots, q_{Y_n}$.

Example 2.3.6. Consider the following CTMC.

PICTURE HERE

The Q matrix of this chain is given by

$$Q = \begin{bmatrix} -\lambda & \lambda & 0 \\ \mu & -(\lambda+\mu) & \mu \\ 0 & \mu & -\mu \end{bmatrix}.$$

Thus, the transition matrix of the jump chain is

$$J = \begin{bmatrix} 0 & 1 & 0 \\ \frac{\mu}{\lambda+\mu} & 0 & \frac{\lambda}{\lambda+\mu} \\ 0 & 1 & 0 \end{bmatrix},$$

and the amount of time spent in state 1 is $\mathsf{Exp}(\lambda)$, the amount of time spent in state 2 is $\mathsf{Exp}(\lambda+\mu)$ and the amount of time spent in state 3 is $\mathsf{Exp}(\mu)$.

2.3.5 Examples of Continuous Time Markov Chains

Poisson Process

We are already quite familiar with one CTMC: the Poisson process. This process has exponential holding times (always with parameter λ) and a jump chain that always increases by one. That is, the jump chain is of the form

$$J = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 1 & 0 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}.$$

This gives a Q-matrix of the form

$$Q = \begin{bmatrix} -\lambda & \lambda & 0 & 0 & 0 & \cdots \\ 0 & -\lambda & \lambda & 0 & 0 & \cdots \\ 0 & 0 & -\lambda & \lambda & 0 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}.$$

Birth-Death Process

A Poisson process is sometimes called a pure birth process, as it models a population that is constantly growing. More generally, birth-death processes are simple models of populations where births and deaths happen at random. A simple example is the following

PICTURE HERE

The Q-matrix for this chain is of the form

$$Q = \begin{bmatrix} -\lambda & \lambda & 0 & 0 & 0 & \cdots \\ \mu & -(\mu+\lambda) & \lambda & 0 & 0 & \cdots \\ 0 & \mu & -(\mu+\lambda) & \lambda & 0 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}.$$

2.3.6 Simulating Continuous Time Markov Chains

If we think about CTMCs in terms of jump chains and holding times, then the generic method for simulating them becomes obvious. We simply wait an exponential rate of time in each state, determined by the Q-matrix, the jump to the next state according to the J matrix. The jump chain itself is simulated just as in the section on discrete time Markov chains. As with discrete time Markov chains, it is important to consider whether your chain is finite or infinite, as this usually changes the choice of how to encode J. If the chain is infinite, then it obviously is not possible to write out the J matrix.

Example 2.3.7 (A finite state space CTMC)**.** Consider the following CTMC

PICTURE HERE,

with initial distribution $\boldsymbol{\lambda} = (1/2, 1/2, 0, 0)$. This chain has Q-matrix

$$Q = \begin{bmatrix} -9 & 2 & 3 & 4 \\ 1 & -4 & 3 & 0 \\ 1 & 2 & -7 & 4 \\ 1 & 0 & 3 & -4 \end{bmatrix}.$$

and jump matrix

$$J = \begin{bmatrix} 0 & 2/9 & 3/9 & 4/9 \\ 1/4 & 0 & 3/4 & 0 \\ 1/7 & 2/7 & 0 & 4/7 \\ 1/4 & 0 & 3/4 & 0 \end{bmatrix}.$$

The Matlab code for simulating this is as follows.

Listing 2.7: Matlab code

```
T = 10; t = 0;

Q = [-9 2 3 4; 1 -4 3 0; 1 2 -7 4; 1 0 3 -4];
J = [0 2/9 3/9 4/9; 1/4 0 3/4 0; 1/7 2/7 0 4/7; 1/4 0 3/4 0];

U = rand; X = (U <= 1/2) + 2*(U > 1/2);
jump_times = [0]; jump_chain = [X];

while t <= T
    t = t + log(rand) / Q(X,X);
    if t > T
        break;
    else
        jump_times = [jump_times t];
        X = min(find(rand<cumsum(J(X,:))));
        jump_chain = [jump_chain X];
    end
end
```

Example 2.3.8 (A birth-death process)**.** Consider the birth-death process pictured in figure (REF) such that $\mathbb{P}(X_0 = 0) = 1$.

PICTURE HERE

We can implement this in Matlab by encoding the transition rule itself.

Listing 2.8: Matlab code

```
T = 100000; t = 0;
lambda = 2; mu = 0.25;

X = 0; jump_times = [0]; jump_chain = [X];

while t <= T
    t = t - log(rand) / (lambda + X*mu);
    if t > T
```

```
        break;
    else
        jump_times = [jump_times t];

        if rand < lambda / (lambda + X*mu)
            X = X + 1;
        else
            X = X - 1;
        end
        jump_chain = [jump_chain X];
    end
end
```

2.3.7 The Relationship Between P and Q in the Finite Case

In the finite case, we can write P directly in terms of Q. Remember that P must satisfy $P(s+t) = P(t)P(s)$. This implies that P should be some kind of exponential like function. In fact, we can write

$$P(t) = e^{tQ}$$

where e^{tQ} is the *matrix exponential*. It might at first be unclear how to define a matrix exponential. However, if we remember the Taylor series definition of the exponential, then we realise we can write something equivalent using matrices. That is

$$P(t) = \sum_{k=0}^{\infty} \frac{(tQ)^k}{k!}.$$

Warning! It is often very computationally challenging to compute a matrix exponential. The answer the computer gives you might not always be correct. In Matlab you need to use 'expm' rather than 'exp' if you wish to use the matrix exponential.

Example 2.3.9. Consider the CTMC given earlier with Q-matrix

$$Q = \begin{bmatrix} -9 & 2 & 3 & 4 \\ 1 & -4 & 3 & 0 \\ 1 & 2 & -7 & 4 \\ 1 & 0 & 3 & -4 \end{bmatrix}$$

and initial distribution $\boldsymbol{\lambda} = (1/2, 1/2, 0, 0)$. We can calculate

We can calculate e^{tQ} easily using Matlab. For $t = 0.01, 0.1, 1, 10, 100$ we have

Listing 2.9: Matlab code

```
[1/2 1/2 0 0] * expm(.01 * Q) = 0.4619 0.4901 0.0285 0.0194
[1/2 1/2 0 0] * expm(.1 * Q) = 0.2472 0.4112 0.1896 0.1520
[1/2 1/2 0 0] * expm(1 * Q) = 0.1000 0.2061 0.3000 0.3939
[1/2 1/2 0 0] * expm(10 * Q) = 0.1000 0.2000 0.3000 0.4000
[1/2 1/2 0 0] * expm(100 * Q) = 0.1000 0.2000 0.3000 0.4000.
```

Looking at the example, we see that the distribution of the chain seems to converge to a stationary distribution.

2.3.8 Irreducibility, Recurrence and Positive Recurrence

As in the case of discrete time Markov chains, we want to establish conditions under which a CTMC has a unique stationary distribution. As it transpires, these conditions are basically the same as those for discrete time Markov chains. This is because it is the structure of the jump chain that determines whether a stationary distribution exists or not.

We say a state i leads to a state j (written $i \to j$) if

$$\mathbb{P}_i(X_t = j \text{ for some } t \geq 0) > 0.$$

We say i and j communicate (or $i \leftrightarrow j$) if $i \to j$ and $j \to i$. The following theorem shows that it is sufficient to look at the jump chain in order to determine communicating classes.

Theorem 2.3.10. Consider a CTMC $\{X_t\}_{t\geq 0}$ with Q-matrix Q and jump chain $\{Y_n\}_{n\geq 0}$. For distinct states i and j the following are equivalent

(i) $i \to j$.

(ii) $i \to j$ for the jump chain.

(iii) $q_{i,k_1} q_{k_1,k_2} \cdots q_{k_n,j} > 0$ for some $n > 0$ and states $k_1, \ldots, k_n$.

(iv) $p_t(i, j) > 0$ for all $t > 0$.

(v) $p_t(i, j) > 0$ for some $t > 0$.

Proof. See [3]. □

As in the discrete time case, a CTMC is irreducible if $i \leftrightarrow j$ for all $i, j \in \mathcal{X}$.

We say a state i is *recurrent* if

$$\mathbb{P}_i(\{t \geq 0 : X_t = i\} \text{ is unbounded}) = 1.$$

We say a state j is *transient* if

$$\mathbb{P}_i(\{t \geq 0 : X_t = i\} \text{ is unbounded}) = 0.$$

Again, it is sufficient to look at the jump chain to establish that these properties hold.

Theorem 2.3.11. Consider a CTMC $\{X_t\}_{t\geq 0}$ with jump chain $\{Y_n\}_{n\geq 0}$.

(i) If i is recurrent for $\{Y_n\}_{n\geq 0}$ it is recurrent for $\{X_t\}_{t\geq 0}$.

(ii) If i is transient for $\{Y_n\}_{n\geq 0}$ it is recurrent for $\{X_t\}_{t\geq 0}$.

Proof. See [3]. □

A state i is positive recurrent if it is positive recurrent for the jump chain.

2.3.9 Invariant Measures and Stationary Distribution

Definition 2.3.12 (Invariant Measure)**.** We say a measure (remember this is a vector with non-negative elements) $\boldsymbol{\mu}$ is invariant for a Q-matrix Q if

$$\boldsymbol{\mu} Q = \mathbf{0}.$$

Invariant measures of CTMCs are closely related to invariant measures for the associated jump chain.

Theorem 2.3.13. Given a CTMC $\{X_t\}_{t\geq 0}$ with Q-matrix Q and jump matrix J, the following are equivalent

(i) $\boldsymbol{\mu}$ is invariant for Q.

(ii) $\boldsymbol{\nu} J = \boldsymbol{\nu}$, where $\nu_i = \mu_i q_i$ for all $i \in \mathcal{X}$.

Note that $1/q_i$ is the expected holding time in state i, so $\boldsymbol{\mu}$ has elements $\mu_i = \nu_i/q_i$ and is thus a invariant measure for the jump chain reweighted by the expected time spent in each state. Note that neither $\boldsymbol{\mu}$ or $\boldsymbol{\nu}$ is necessarily a probability distribution. Furthermore, even if $\boldsymbol{\mu}$ is a probability distribution, the $\boldsymbol{\nu}$ defined in respect to it will probably not be a probability

distribution (until it is normalized). This is also the case if $\boldsymbol{\nu}$ is a probability distribution.

We are now able to give conditions for the existence of a unique stationary measure for a CTMC $\{X_t\}_{t\geq 0}$. Unsurprisingly, these are the same conditions as for discrete time Markov chains.

Theorem 2.3.14. Let Q be an irreducible Q matrix. Then, the following are equivalent.

(i) Every state is positive recurrent.

(ii) Some state, i, is positive recurrent.

(iii) Q has an invariant distribution $\boldsymbol{\rho}$.

Example 2.3.15. Given that a stationary distribution exists, we can find it by solving the linear system $\boldsymbol{\rho}Q = \mathbf{0}$. Consider the example from earlier with Q matrix

$$Q = \begin{bmatrix} -9 & 2 & 3 & 4 \\ 1 & -4 & 3 & 0 \\ 1 & 2 & -7 & 4 \\ 1 & 0 & 3 & -4 \end{bmatrix}.$$

Solving this system, we find $\boldsymbol{\rho} = (1/10, 2/10, 3/10, 4/10)$.

MENTION LIMITING DISTRIBUTION

2.3.10 Reversibility and Detailed Balance

As is the case with discrete time Markov chains, reversible chains are particularly nice to work with. This largely due to detailed balance equations holding.

Theorem 2.3.16. Given a CTMC $\{X_t\}_{t\geq 0}$ with irreducible Q-matrix Q and invariant distribution $\boldsymbol{\rho}$ (which also serves as the initial distribution of $\{X_t\}_{t\geq 0}$). The process $\{\widehat{X}_t\}_{t\geq 0}$ defined by $\widehat{X}_t = X_{T-t}$ is Markov $(\boldsymbol{\rho}, \widehat{Q})$. Additionally, $\widehat{Q}$ is irreducible and has invariant distribution $\boldsymbol{\rho}$.

Proof. See [3]. □

We say a CTMC is *reversible* if $Q = \widehat{Q}$.

Definition 2.3.17 (Detailed Balance Equations)**.** A Q-matrix, Q, and a measure $\boldsymbol{\mu}$ are in *detailed balance* if

$$\mu_i q_{i,j} = \mu_j q_{i,j} \text{ for all } i, j \in \mathcal{X}.$$

The following theorem is more or less identical to the discrete version.

Theorem 2.3.18. If a Q-matrix, Q, and a measure $\boldsymbol{\mu}$ are in detailed balance then $\boldsymbol{\mu}$ is invariant for Q.

Proof. For a given $i \in \mathcal{X}$, we have

$$(\boldsymbol{\mu}Q)_i = \sum_{j \in \mathcal{X}} \mu_j q_{j,i} = \sum_{j \in \mathcal{X}} \mu_i q_{i,j} = 0.$$

□

The only CTMCs that satisfy detailed balance are reversible ones.

Theorem 2.3.19. Let Q be an irreducible Q-matrix and $\boldsymbol{\rho}$ a distribution. Suppose $\{X_t\}_{t \geq 0}$ is Markov $(\boldsymbol{\rho}, Q)$. Then, the following are equivalent

(i) $\{X_t\}_{t \geq 0}$ is reversible.

(ii) Q and $\boldsymbol{\rho}$ are in detailed balance.

Proof. See [3]. □

Chapter 3

Gaussian Processes and Stochastic Differential Equations

3.1 Gaussian Processes

Gaussian processes are a reasonably large class of processes that have many applications, including in finance, time-series analysis and machine learning. Certain classes of Gaussian processes can also be thought of as spatial processes. We will use these as a vehicle to start considering more general spatial objects.

Definition 3.1.1 (Gaussian Process)**.** A stochastic process $\{X_t\}_{t\geq 0}$ is Gaussian if, for any choice of times $t_1, \ldots, t_n$, the random vector $(X_{t_1}, \ldots, X_{t_n})$ has a *multivariate normal distribution.*

3.1.1 The Multivariate Normal Distribution

Because Gaussian processes are defined in terms of the multivariate normal distribution, we will need to have a pretty good understanding of this distribution and its properties.

Definition 3.1.2 (Multivariate Normal Distribution)**.** A vector $\mathbf{X} = (X_1, \ldots, X_n)$ is said to be multivariate normal (multivariate Gaussian) if all linear combinations of $\mathbf{X}$, i.e. all random variables of the form

$$\sum_{k=1}^{n} \alpha_k X_k$$

have univariate normal distributions.

This is quite a strong definition. Importantly, it implies that, even if all of its components are normally distributed, a random vector is not necessarily multivariate normal.

Example 3.1.3 (A random vector with normal marginals that is not multivariate normal)**.** Let $X_1 \sim \mathsf{N}(0,1)$ and

$$X_2 = \begin{cases} X_1 & \text{if } |X_1| \leq 1 \\ -X_1 & \text{if } |X_1| > 1 \end{cases}.$$

Note that $X_2 \sim \mathsf{N}(0,1)$. However, $X_1 + X_2$ is not normally distributed, because $|X_1 + X_2| \leq 2$, which implies $X_1 + X_2$ is bounded and, hence, cannot be normally distributed.

Linear transformations of multivariate normal random vectors are, again, multivariate normal.

Lemma 3.1.4. Suppose $\mathbf{X} = (X_1, \ldots, X_n)$ is multivariate normal and A is an $m \times n$ real-valued matrix. Then, $\mathbf{Y} = A\mathbf{X}$ is also multivariate normal.

Proof. Any linear combination of $Y_1, \ldots, Y_m$ is a linear combination of linear combinations of $X_1, \ldots, X_n$ and, thus, univariate normal. □

Theorem 3.1.5. A multivariate normal random vector $\mathbf{X} = (X_1, \ldots, X_n)$ is completely described by a mean vector $\boldsymbol{\mu} = \mathbb{E}\mathbf{X}$ and a covariance matrix $\Sigma = \mathrm{Var}(\mathbf{X})$.

Proof. The distribution of $\mathbf{X}$ is described by its characteristic function which is

$$\mathbb{E}\exp\left\{i\boldsymbol{\theta}^\intercal \mathbf{X}\right\} = \mathbb{E}\exp\left\{i\sum_{i=1}^n \theta_i X_i\right\}.$$

Now, we know $\sum_{i=1}^n \theta_i X_i$ is a univariate normal random variable (because $\mathbf{X}$ is multivariate normal). Let $m = \mathbb{E}\sum_{i=1}^n \theta_i X_i$ and $\sigma^2 = \mathrm{Var}\left(\sum_{i=1}^n \theta_i X_i\right)$. Then,

$$\mathbb{E}\left\{i\sum_{i=1}^n \theta_i X_i\right\} = \exp\left\{im - \frac{1}{2}\sigma^2\right\}.$$

Now

$$m = \mathbb{E}\sum_{i=1}^n \theta_i X_i = \sum_{i=1}^n \theta_i \mu_i$$

and

$$\sigma^2 = \mathrm{Var}\left(\sum_{i=1}^n \theta_i X_i\right) = \sum_{i=1}^n \sum_{j=1}^n \theta_i \theta_j \mathrm{Cov}(X_i, X_j) = \sum_{j=1}^n \sum_{j=1}^n \theta_i \theta_j \Sigma_{i,j}.$$

So, everything is specified by $\boldsymbol{\mu}$ and Σ. □

INDEPENDENCE

There is nothing too difficult about dealing with the mean vector $\boldsymbol{\mu}$. However, the covariance matrix makes things pretty difficult, especially when we wish to simulate a high dimensional random vector. In order to simulate Gaussian processes effectively, we need to exploit as many properties of covariance matrices as possible.

Symmetric Positive Definite and Semi-Positive Definite Matrices

Covariance matrices are members of a family of matrices called *symmetric positive definite matrices*.

Definition 3.1.6 (Positive Definite Matrices (Real-Valued))**.** An $n \times n$ real-valued matrix, A, is positive definite if and only if

$$\boldsymbol{x}^\intercal A \boldsymbol{x} > 0 \text{ for all } \boldsymbol{x} \neq \mathbf{0}.$$

If A is also symmetric, then A is called symmetric positive definite (SPD). Some important properties of SPD matrices are

(i) $\mathsf{rank}(A) = n$.

(ii) $|A| > 0$.

(iii) $A_{i,i} > 0$.

(iv) A^{-1} is SPD.

Lemma 3.1.7 (Necessary and sufficient conditions for an SPD)**.** The following are necessary and sufficient conditions for an $n \times n$ matrix A to be SPD

(i) All the eigenvalues $\lambda_1, \ldots, \lambda_n$ of A are strictly positive.

(ii) There exists a unique matrix C such that $A = CC^\intercal$, where C is a real-valued lower-triangular matrix with positive diagonal entries. This is called the *Cholesky decomposition* of A.

Definition 3.1.8 (Positive Semi-definite Matrices (Real-Valued))**.** An $n \times n$ real-valued matrix, A, is positive semi-definite if and only if

$$\boldsymbol{x}^\intercal A \boldsymbol{x} \geq 0 \text{ for all } \boldsymbol{x} \neq \mathbf{0}.$$

If A is also symmetric, then A is called symmetric positive semi-definite (SPSD). Note that if A is an SPSD then there is a real-valued decomposition

$$A = LL^\intercal,$$

though it is not necessarily unique and L may have zeroes on the diagonals.

Lemma 3.1.9. Covariance matrices are SPSD.

Proof. Given an $n \times 1$ real-valued vector $\mathbf{x}$ and a random vector $\mathbf{Y}$ with covariance matrix Σ, we have

$$\mathrm{Var}\left(\mathbf{x}^\intercal \mathbf{Y}\right) = \mathbf{x}^\intercal \mathrm{Var}\left(\mathbf{Y}\right)\mathbf{x}.$$

Now this must be non-negative (as it is a variance). That is, it must be the case that

$$\mathbf{x}^\intercal \mathrm{Var}\left(\mathbf{Y}\right)\mathbf{x} \geq 0,$$

so Σ is positive semi-definite. Symmetry comes from the fact that $\mathrm{Cov}(X, Y) = \mathrm{Cov}(Y, X)$. □

Lemma 3.1.10. SPSD matrices are covariance matrices.

Proof. Let A be an SPSD matrix and $\mathbf{Z}$ be a vector of random variables with $\mathrm{Var}(\mathbf{Z}) = I$. Now, as A is SPSD, $A = LL^\intercal$. So,

$$\mathrm{Var}(L\mathbf{Z}) = L\mathrm{Var}(\mathbf{Z})L^\intercal = LIL^\intercal = A,$$

so A is the covariance matrix of $L\mathbf{Z}$. □

COVARIANCE POS DEF ...

Densities of Multivariate Normals

If $\mathbf{X} = (X_1, \ldots, X_n)$ is $\mathsf{N}(\boldsymbol{\mu}, \Sigma)$ and Σ is positive definite, then $\mathbf{X}$ has the density

$$f(\mathbf{x}) = f(x_1, \ldots, x_n) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\intercal \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right\}.$$

Simulating Multivariate Normals

One possible way to simulate a multivariate normal random vector is using the Cholesky decomposition.

Lemma 3.1.11. If $\mathbf{Z} \sim \mathsf{N}(\mathbf{0}, I)$, $\Sigma = AA^\intercal$ is a covariance matrix, and $\mathbf{X} = \boldsymbol{\mu} + A\mathbf{Z}$, then $\mathbf{X} \sim \mathsf{N}(\boldsymbol{\mu}, \Sigma)$.

Proof. We know from lemma 3.1.4 that $A\mathbf{Z}$ is multivariate normal and so is $\boldsymbol{\mu} + A\mathbf{Z}$. Because a multivariate normal random vector is completely described by its mean vector and covariance matrix, we simple need to find $\mathbb{E}\mathbf{X}$ and $\mathrm{Var}(\mathbf{X})$. Now,

$$\mathbb{E}\mathbf{X} = \mathbb{E}\boldsymbol{\mu} + \mathbb{E}A\mathbf{Z} = \boldsymbol{\mu} + A\mathbf{0} = \boldsymbol{\mu}$$

and

$$\mathrm{Var}(\mathbf{X}) = \mathrm{Var}(\boldsymbol{\mu} + A\mathbf{Z}) = \mathrm{Var}(A\mathbf{Z}) = A\mathrm{Var}(\mathbf{Z})A^\intercal = AIA^\intercal = AA^\intercal = \Sigma.$$

□

Thus, we can simulate a random vector $\mathbf{X} \sim \mathsf{N}(\boldsymbol{\mu}, \Sigma)$ by

(i) Finding A such that $\Sigma = AA^\intercal$.

(ii) Generating $\mathbf{Z} \sim \mathsf{N}(\mathbf{0}, I)$.

(iii) Returning $\mathbf{X} = \boldsymbol{\mu} + A\mathbf{Z}$.

Matlab makes things a bit confusing because its function 'chol' produces a decomposition of the form $B^\intercal B$. That is, $A = B^\intercal$. It is important to be careful and think about whether you want to generate a row or column vector when generating multivariate normals.

The following code produces column vectors.

Listing 3.1: Matlab code

```
mu = [1 2 3]';
Sigma = [3 1 2; 1 2 1; 2 1 5];
A = chol(Sigma);
X = mu + A' * randn(3,1);
```

The following code produces row vectors.

Listing 3.2: Matlab code

```
mu = [1 2 3];
Sigma = [3 1 2; 1 2 1; 2 1 5];
A = chol(Sigma);
X = mu + randn(1,3) * A;
```

The complexity of the Cholesky decomposition of an arbitrary real-valued matrix is $O(n^3)$ floating point operations.

3.1.2 Simulating a Gaussian Processes Version 1

Because Gaussian processes are defined to be processes where, for any choice of times $t_1, \ldots, t_n$, the random vector $(X_{t_1}, \ldots, X_{t_n})$ has a multivariate normal density and a multivariate normal density is completely describe by a mean vector $\boldsymbol{\mu}$ and a covariance matrix Σ, the probability distribution of Gaussian process is completely described if we have a way to construct the mean vector and covariance matrix for an arbitrary choice of $t_1, \ldots, t_n$.

We can do this using an *expectation function*

$$\mu(t) = \mathbb{E}X_t$$

and a covariance function

$$r(s,t) = \mathrm{Cov}(X_s, X_t).$$

Using these, we can simulate the values of a Gaussian process at fixed times $t_1, \ldots, t_n$ by calculating $\boldsymbol{\mu}$ where $\mu_i = \mu(t_1)$ and Σ, where $\Sigma_{i,j} = r(t_i, t_j)$ and then simulating a multivariate normal vector.

In the following examples, we simulate the Gaussian processes at evenly spaced times $t_1 = 1/h, t_2 = 2/h, \ldots, t_n = 1$.

Example 3.1.12 (Brownian Motion)**.** *Brownian motion* is a very important stochastic process which is, in some sense, the continuous time continuous state space analogue of a simple random walk. It has an expectation function $\mu(t) = 0$ and a covariance function $r(s,t) = \min(s,t)$.

Listing 3.3: Matlab code

```
%use mesh size h
m = 1000; h = 1/m; t = h : h : 1;
n = length(t);

%Make the mean vector
mu = zeros(1,n);
for i = 1:n
    mu(i) = 0;
end

%Make the covariance matrix
Sigma = zeros(n,n);
```

```
for i = 1:n
    for j = 1:n
        Sigma(i,j) = min(t(i),t(j));
    end
end

%Generate the multivariate normal vector
A = chol(Sigma);
X = mu + randn(1,n) * A;

%Plot
plot(t,X);
```

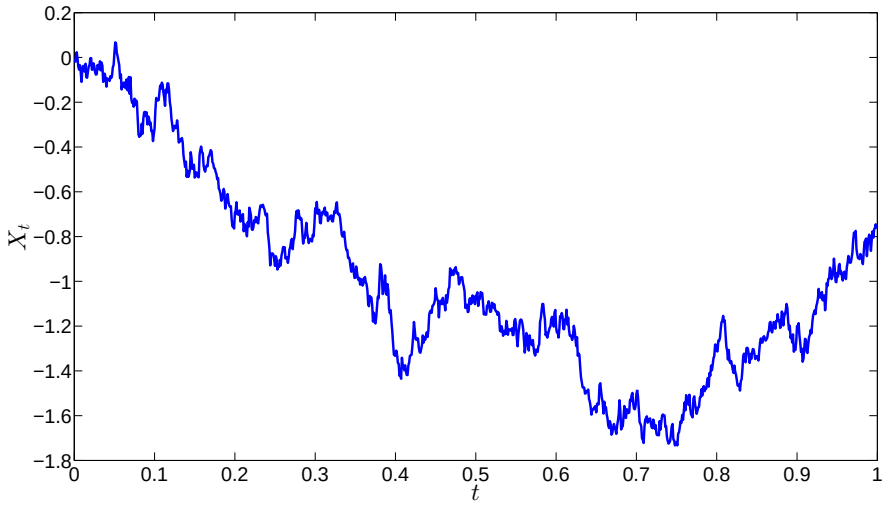

Figure 3.1.1: Plot of Brownain motion on $[0, 1]$.

Example 3.1.13 (Ornstein-Uhlenbeck Process)**.** Another very important Gaussian process is the Ornstein-Uhlenbeck process. This has expectation function $\mu(t) = 0$ and covariance function $r(s, t) = e^{-\alpha|s-t|/2}$.

Listing 3.4: Matlab code

```
%use mesh size h
m = 1000; h = 1/m; t = h : h : 1;
n = length(t);
%paramter of OU process
alpha = 10;

```

```
%Make the mean vector
mu = zeros(1,n);
for i = 1:n
mu(i) = 0;
end

%Make the covariance matrix
Sigma = zeros(n,n);
for i = 1:n
for j = 1:n
Sigma(i,j) = exp(-alpha * abs(t(i) - t(j)) / 2);
end
end

%Generate the multivariate normal vector
A = chol(Sigma);
X = mu + randn(1,n) * A;

%Plot
plot(t,X);
```

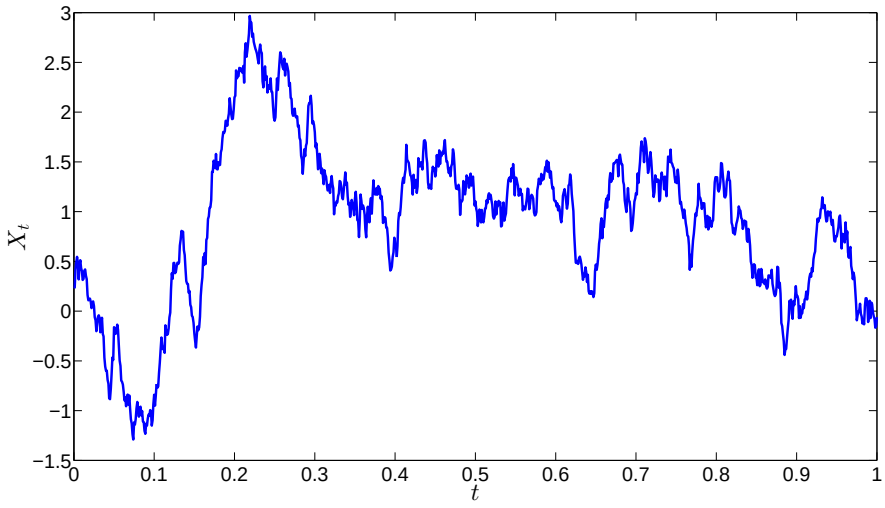

Figure 3.1.2: Plot of an Ornstein-Uhlenbeck process on $[0, 1]$ with $\alpha = 10$.

Example 3.1.14 (Fractional Brownian Motion)**.** Fractional Brownian motion (fBm) is a generalisation of Brownian Motion. It has expectation function $\mu(t) = 0$ and covariance function $\text{Cov}(s, t) = 1/2(t^{2H} + s^{2H} - |t - s|^{2H})$,

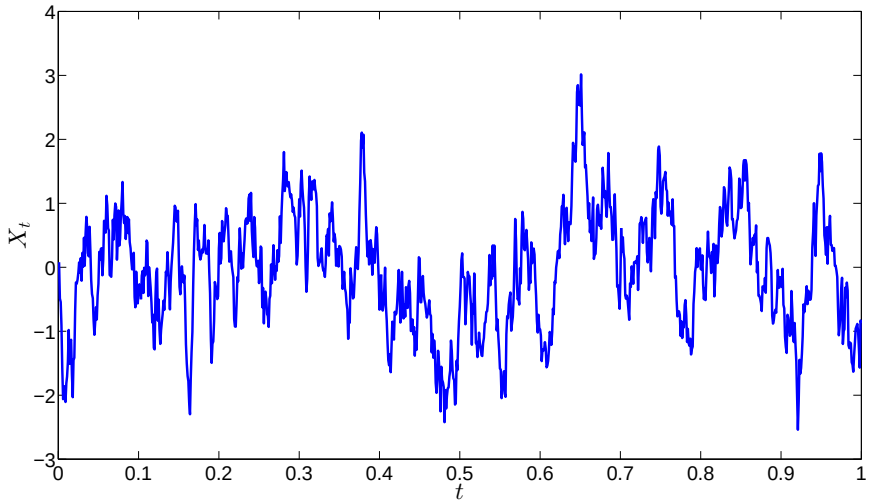

Figure 3.1.3: Plot of an Ornstein-Uhlenbeck process on $[0, 1]$ with $\alpha = 100$.

where $H \in (0, 1)$ is called the Hurst parameter. When $H = 1/2$, fBm reduces to standard Brownian motion. Brownian motion has independent increments. In contrast, for $H > 1/2$ fBm has positively correlated increments and for $H < 1/2$ fBm has negatively correlated increments.

Listing 3.5: Matlab code

```
%Hurst parameter
H = .9;

%use mesh size h
m = 1000; h = 1/m; t = h : h : 1;
n = length(t);

%Make the mean vector
mu = zeros(1,n);
for i = 1:n
    mu(i) = 0;
end

%Make the covariance matrix
Sigma = zeros(n,n);
for i = 1:n
    for j = 1:n
        Sigma(i,j) = 1/2 * (t(i)^(2*H) + t(j)^(2*H)...
            - (abs(t(i) - t(j)))^(2 * H));
```

```
    end
end

%Generate the multivariate normal vector
A = chol(Sigma);
X = mu + randn(1,n) * A;

%Plot
plot(t,X);
```

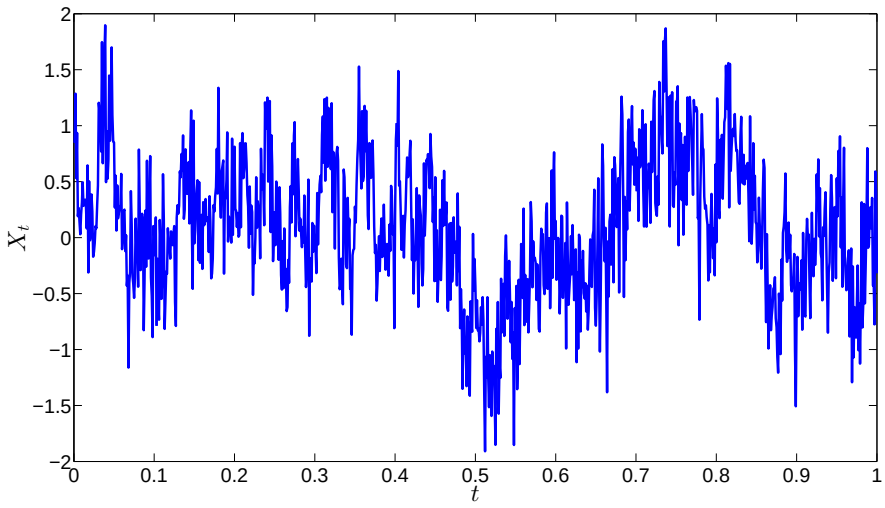

Figure 3.1.4: Plot of fractional Brownian motion on $[0, 1]$ with $H = 0.1$.

3.1.3 Stationary and Weak Stationary Gaussian Processes

Gaussian processes (and stochastic processes in general) are easier to work with if they are *stationary stochastic processes*.

Definition 3.1.15 (Stationary Stochastic Process)**.** A stochastic process $\{X_t\}_{t\geq 0}$ is said to be stationary if the random vectors $(X_{t_1}, X_{t_2}, \ldots, X_{t_n})$ and $(X_{t_1+s}, X_{t_2+s}, \ldots, X_{t_n+s})$ have the same distribution for all choices of s, n and $t_1, t_2, \ldots, t_n$.

An example of such a process would be an irreducible positive recurrent continuous time Markov chain started from its stationary distribution.

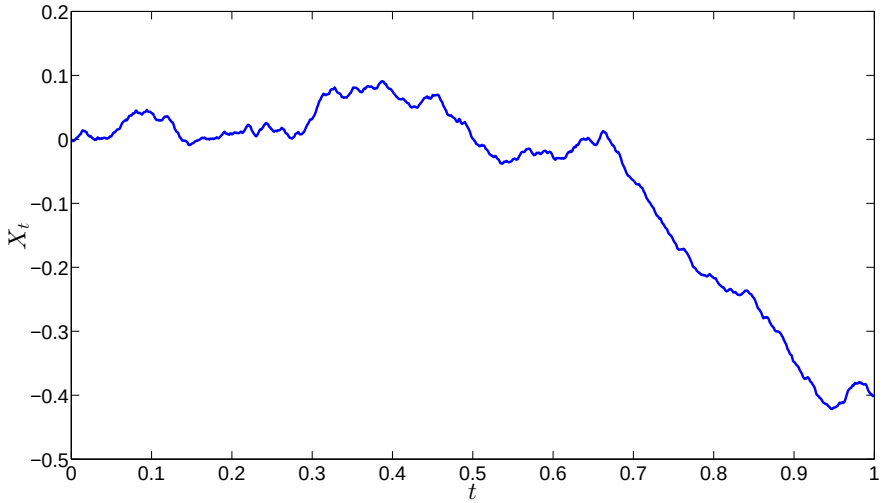

Figure 3.1.5: Plot of fractional Brownian motion on $[0, 1]$ with $H = 0.9$.

The conditions for a stochastic process to be stationary are quite strict. Often, a slightly weaker version of stationarity, called *weak stationarity* is required instead.

Definition 3.1.16 (Weak Stationary Stochastic Process)**.** A stochastic process $\{X_t\}_{t\geq 0}$ is said to be weak stationary (sometimes *wide sense stationary* or *second order stationary*) if $\mathbb{E}X_t = c$ for all $t \geq 0$ and $\text{Cov}(X_t, X_{t+s})$ does not depends on t.

An example of a weak stationary process is the Ornstein-Uhlenbeck process, as $\mathbb{E}X_t = \mu(t) = 0$ and $\text{Cov}(X_t, X_{t+s}) = r(t, t+s) = e^{-\alpha|t-(t+s)|/2} = e^{-\alpha s/2}$.

Lemma 3.1.17. Gaussian processes that are weak stationary are stationary.

Proof. We know from theorem 3.1.5 that Gaussian distributions are entirely determined by their mean vector and covariance matrix. Since the mean and covariance of a weakly stationary process do not change when the times are all shifted by s, a weakly stationary Gaussian process is stationary. □

An Ornstein-Uhlenbeck process is a weak stationary Gaussian process, so it is a stationary stochastic process.

The covariance matrix of a stationary Gaussian process evaluated at equidistant times (e.g. $t_0 = 0, t_1 = 1/h, t_2 = 2/h, \ldots, t_n = 1$) has a par-

ticularly nice form:

$$\Sigma = \begin{bmatrix} \sigma_0 & \sigma_1 & \sigma_2 & \cdots & \sigma_n \\ \sigma_1 & \sigma_0 & \sigma_1 & \cdots & \sigma_{n-1} \\ \sigma_2 & \sigma_1 & \sigma_0 & \ddots & \sigma_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_n & \sigma_{n-1} & \sigma_{n-2} & \cdots & \sigma_0 \end{bmatrix}.$$

Technically, it is a *symmetric Toeplitz matrix*.

Definition 3.1.18 (Toeplitz Matrix)**.** A Toeplitz matrix, A, is a matrix where each diagonal takes a single, constant value. That is, $a_{i,j} = a_{i+1,j+1}$ for all $i, j \in \{1, \ldots, n-1\}$.

We can embed this matrix in a $2n \times 2n$ *circulant matrix*

$$\Sigma = \left[\begin{array}{ccccc|ccccc} \sigma_0 & \sigma_1 & \sigma_2 & \cdots & \sigma_n & \sigma_{n-1} & \sigma_{n-2} & \cdots & \sigma_2 & \sigma_1 \\ \sigma_1 & \sigma_0 & \sigma_1 & \cdots & \sigma_{n-1} & \sigma_n & \sigma_{n-1} & \cdots & \sigma_3 & \sigma_2 \\ \sigma_2 & \sigma_1 & \sigma_0 & \ddots & \sigma_{n-2} & \sigma_{n-1} & \sigma_n & \cdots & \sigma_4 & \sigma_3 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \sigma_n & \sigma_{n-1} & \sigma_{n-2} & \cdots & \sigma_0 & \sigma_1 & \sigma_2 & \cdots & \sigma_n & \sigma_{n-1} \\ \hline \sigma_{n-1} & \sigma_n & \sigma_{n-1} & \cdots & \sigma_1 & \sigma_0 & \sigma_1 & \cdots & \sigma_{n-3} & \sigma_{n-2} \\ \sigma_{n-2} & \sigma_{n-1} & \sigma_n & \cdots & \sigma_2 & \sigma_1 & \sigma_0 & \cdots & \sigma_{n-4} & \sigma_{n-3} \\ \sigma_{n-3} & \sigma_{n-2} & \sigma_{n-1} & \cdots & \sigma_3 & \sigma_2 & \sigma_1 & \cdots & \sigma_{n-5} & \sigma_{n-4} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \sigma_1 & \sigma_2 & \sigma_3 & \cdots & \sigma_{n-1} & \sigma_{n-2} & \sigma_{n-3} & \cdots & \sigma_1 & \sigma_0 \end{array}\right].$$

If $\sigma_0 \geq \sigma_1 \geq \cdots \geq \sigma_n \geq 0$ and $2\sigma_k \leq \sigma_{k-1} + \sigma_{k+1}$ for $k = 1, \ldots, n-1$, then C is also a covariance matrix.

Definition 3.1.19 (Circulant Matrix)**.** A circulant matrix is a matrix of the form

$$B = \begin{bmatrix} b_0 & b_{n-1} & \cdots & b_2 & b_1 \\ b_1 & b_0 & \cdots & b_3 & b_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ b_{n-1} & b_{n-2} & \cdots & b_1 & b_0 \end{bmatrix}.$$

It is fully specified by $\mathbf{b}$, its first column. The last row of a circulant matrix is the first column in reverse order. The remaining rows are cyclic permutations of this row.

This might not seem very useful, but the strength of this representation comes from its link to the *discrete Fourier transform* and, through it, to that wonder of numerical analysis: the *Fast Fourier Transform* (FFT).

Definition 3.1.20 (Discrete Fourier Transform)**.** Given a complex valued vector $\mathbf{x} = (x_0, \ldots, x_n)^\intercal$, we define its discrete Fourier transform by $\widetilde{\mathbf{x}} = (\widetilde{x}_0, \ldots, \widetilde{x}_{n-1})^\intercal$, where

$$\widetilde{x}_j = \sum_{k=0}^{n-1} e^{-(2\pi i/n)jk} x_k = \sum_{k=0}^{n-1} \omega^{jk} x_k$$

for $j = 0, \ldots, n-1$, where $\omega = e^{-(2\pi i/n)}$.

Note that the discrete Fourier transform is equivalent to calculating

$$\widetilde{\mathbf{x}} = F\mathbf{x},$$

where

$$F = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \ddots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)^2} \end{bmatrix}.$$

As it turns out, $F^{-1} = \bar{F}/n$. Normally solving the system $\widetilde{\mathbf{x}} = F\mathbf{x}$ would require $O(n^2)$ steps, but the FFT reduces this to $O(n \log n)$ steps.

The link to circulant matrices is as follows. Let $\mathbf{b} = (b_0, b_1, \ldots, b_{n-1})^\intercal$ be a complex-valued vector. The circulant matrix corresponding to $\mathbf{b}$ is

$$B = \begin{bmatrix} b_0 & b_{n-1} & \cdots & b_2 & b_1 \\ b_1 & b_0 & \cdots & b_3 & b_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ b_{n-1} & b_{n-2} & \cdots & b_1 & b_0 \end{bmatrix}.$$

The eigenvalues of B, $\lambda_0, \ldots, \lambda_{n-1}$ are given by

$$\lambda_j = \mathbf{b}^\intercal \bar{\mathbf{f}}_j,$$

for $j = 0, \ldots, n-1$, with associated eigenvectors $\mathbf{f}_0, \ldots, \mathbf{f}_{n-1}$, where $\mathbf{f}_j$ is the jth column of F.

If B is diagonalizable, then it can be written as $B = PDP^{-1}$, where the columns of P are the right eigenvectors of B and $D = \mathsf{diag}(\lambda_0, \ldots, \lambda_{n-1})$. Thus, if we can calculate

$$\boldsymbol{\lambda} = (\lambda_0, \ldots, \lambda_{n-1}) = \bar{F}\mathbf{b}$$

and

$$E = F\sqrt{\mathsf{diag}(\boldsymbol{\lambda}/n)},$$

then

$$E\bar{E}^{\intercal} = F\mathsf{diag}(\boldsymbol{\lambda})\bar{F}/n = F\mathsf{diag}(\boldsymbol{\lambda})F^{-1} = B.$$

In other words E is the complex square-root of the matrix B.

Let B be the covariance matrix of a zero-mean Gaussian process evaluated at equidistant steps. Now, if we write $E = E_1 + iE_2$ then

$$B = E\bar{E}^{\intercal} = (E_1 + iE_2)\,(E_1^{\intercal} - iE_2^{\intercal}) = E_1E_1^{\intercal} + E_2E_2^{\intercal} + i\,(E_2E_1^{\intercal} - E_1E_2^{\intercal})$$

As B is real-valued, $B = E_1E_1^{\intercal} + E_2E_2^{\intercal}$.

Let $\mathbf{Z}_1$ and $\mathbf{Z}_2$ be $\mathsf{N}(\mathbf{0}, I)$ and define

$$\mathbf{X} = E\mathbf{Z} = (E_1 + iE_2)(\mathbf{Z}_1 + i\mathbf{Z}_2) = (E_1\mathbf{Z}_1 - E_2\mathbf{Z}_2) + i(E_2\mathbf{Z}_1 + E_1\mathbf{Z}_2).$$

Let

$$\mathbf{X}_1 = \mathcal{R}(\mathbf{X}) = E_1\mathbf{Z}_1 - E_2\mathbf{Z}_2$$

and

$$\mathbf{X}_2 = \mathcal{I}(\mathbf{X}) = E_2\mathbf{Z}_1 + E_1\mathbf{Z}_2.$$

Then, clearly, $\mathbb{E}\mathbf{X}_1 = \mathbf{0}$ and

$$\mathrm{Var}(\mathbf{X}_1) = \mathrm{Var}(E_1\mathbf{Z}_1 - E_2\mathbf{Z}_2) = E_1E_1^{\intercal} + E_2E_2^{\intercal} = B.$$

The calculation for $\mathbf{X}_2$ is more or less identical. Note $\mathbf{X}_1$ and $\mathbf{X}_2$ are not independent of one another.

This gives the following algorithm for generating a stationary Gaussian process at equidistant intervals. Using the FFT, the algorithm is $O(n \log n)$.

Algorithm 3.1.1 (Generating a Stationary Gaussian Process)**.**

(i) Set $\mathbf{c} = (\sigma_0, \sigma_1, \ldots, \sigma_{n-1}, \sigma_n, \sigma_{n-1}, \ldots, \sigma_1)$.

(ii) Compute $\boldsymbol{\lambda} = F\mathbf{c}$ using the FFT.

(iii) Generate $\mathbf{Z} = \mathbf{Y}_1 + \mathbf{Y}_2$ where $\mathbf{Y}_1, \mathbf{Y}_2 \sim \mathsf{N}(\mathbf{0}, I)$.

(iv) Compute $\boldsymbol{\eta} = \sqrt{\mathsf{diag}(\boldsymbol{\lambda}/n)}\mathbf{Z}$.

(v) Compute $\mathbf{V} = F\boldsymbol{\eta}$ and let A be the first $n+1$ elements of $\mathbf{V}$.

(vi) Output $\mathbf{X} = \mathcal{R}(A)$.

Example 3.1.21. Consider a stationary, zero-mean, Gaussian process on an equally-spaced mesh of $n+1 = 10^4+1$ points on $[a,b] = [0,5]$ with $\sigma_k = \exp\{-(b-a)k/n\}$ for $k = 0, 1, \ldots, n$.

Listing 3.6: Matlab code

```
n=10^4; a=0; b=5;
t=linspace(a,b,n+1); sigma=exp(-(t-t(1)));
c=[sigma sigma((end-1):-1:2)]';
lambda=fft(c); %eigenvalues
Z=randn(2*n,1)+sqrt(-1).*randn(2*n,1); %complex normal vectors
eta=sqrt(lambda./(2*n)).*Z;
V=fft(eta);
A=V(1:(n+1));
X=real(A);
plot(t,X)
```

3.1.4 Finite Dimensional Distributions

A very nice property of Gaussian processes is that we know their *finite dimensional distributions*.

Definition 3.1.22 (Finite Dimensional Distributions)**.** The finite dimensional distributions of a stochastic process $\{X_t\}_{t\geq 0}$ are the distributions of all vectors of the form $(X_{t_1}, \ldots, X_{t_n})$ with $n > 0$ and $0 \leq t_1 \leq \cdots \leq t_n$.

The finite dimensional distributions tell us a lot about the behaviour of a stochastic process. It is worth noting, however, that they do not fully specify stochastic processes. For example, Brownian motion is almost surely continuous but this property does not follow simply from specifying the finite dimensional distributions.

We can simulate the finite dimensional skeletons of a Gaussian process exactly. That is, we can generate $X_{t_1}, \ldots, X_{t_n}$ for any choice of n and $t_1, \ldots, t_n$. This is not always true for other stochastic processes. However, we do encounter a new form of error, *discretization error*.

Definition 3.1.23 (Discretization Error)**.** Discretization error is the error that arises from replacing a continuous object with a discrete object.

For example, if we wish to calculate the variance of the proportion of time that a Gaussian process spends above 0, or the expectation of the first time a process hits a set, A, then we will encounter an error in considering the process only at a fixed number of points.

Discretization error needs to be considered when we decide on a sampling budget. Consider, for example, an evenly spaced mesh of points $t_1 = 1/m, t_2 = 2/m, \ldots t_n = 1$. As m gets bigger, the mesh gets finer and the discretization error gets smaller. We still have statistical error, however, so we need to make sure we generate a large enough sample of realizations of the stochastic process (determined by the sample size N). The total work done by our simulation is then given by

$$\mathsf{work} = \text{number of samples} \times \text{work to make one sample} = Nf(m).$$

Usually, f grows at least linearly in m. In the case of Cholesky decomposition, for example, it is $O(m^3)$. For a fixed level of work, we need to decide on how much effort to allocate to reducing discretization error (how large m should be) and how much effort to allocate to reducing statistical error (how large N should be). Finding the optimal tradeoff can be difficult. We will consider such tradeoffs in a number of situations.

3.1.5 Marginal and Conditional Multivariate Normal Distributions

The multivariate normal distribution has many attractive properties. In particular, its marginal distributions are also multivariate normal. In addition, if we condition on part of the a multivariate normal vector, the remaining values are also multivariate normal.

To see this, we need to write normal random vectors in the appropriate form. Let $\mathbf{X} \sim \mathsf{N}(\boldsymbol{\mu}, \Sigma)$. We can decompose $\mathbf{X}$ into two parts, $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)^\intercal$. We can then write $\boldsymbol{\mu}$ as $(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)^\intercal$ and Σ as

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}.$$

Theorem 3.1.24 (Multivariate Normal Marginal Distributions)**.** Given $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2) \sim \mathsf{N}(\boldsymbol{\mu}, \Sigma)$,

$$\mathbf{X}_1 \sim \mathsf{N}(\boldsymbol{\mu}_1, \Sigma_{11}),$$

and

$$\mathbf{X}_2 \sim \mathsf{N}(\boldsymbol{\mu}_2, \Sigma_{22}).$$

Theorem 3.1.25 (Multivariate Normal Conditional Distributions)**.** Given $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2) \sim \mathsf{N}(\boldsymbol{\mu}, \Sigma)$, $\mathbf{X}_2$ conditional on $\mathbf{X}_1$ is multivariate normal with

$$\mathbb{E}[\mathbf{X}_2 \,|\, \mathbf{X}_1] = \boldsymbol{\mu}_2 + \Sigma_{21}\Sigma_{11}^{-1}(\mathbf{X}_1 - \boldsymbol{\mu}_1)$$

and

$$\mathrm{Var}(\mathbf{X}_2 \,|\, \mathbf{X}_1) = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}.$$

3.1.6 Interpolating Gaussian Processes

Because we know the finite dimensional distributions of Gaussian processes and know a lot about working with the multivariate normal distribution, we are able to interpolate between already simulated points of a Gaussian process. We can do this by simply drawing the new points conditional on the values that we have already generate. For example, if we have generate values for the process at 0.5 and 1, then we can generate values at the points 0.25 and 0.75 conditional on $X_{0.5}$ and X_1.

There are a number of reasons why interpolation might be useful. One is that it might not make sense to simulate the whole stochastic process on a fine mesh (which is expensive), but rather to simulate a coarse path of the stochastic process first and then focus our efforts on a particular region of this path. For example, if we are trying to estimate the first time a stochastic process hits a particular value, then we might want to focus our simulation efforts on the part of the stochastic process that is closest to this value. We should be careful, however, as simulating in this way could introduce a bias.

Example 3.1.26 (Iteratively Updating Brownian Motion)**.** Consider an example where we update Brownian motion in an iterative fashion. Remember, Brownian motion has mean function $\mu(t) = 0$ and covariance function $r(s,t) = \min(s,t)$. We interpolate between points to simulate a process on an increasingly fine mesh.

Listing 3.7: Matlab code

```
num_levels = 10;

%Make the first two points (at 0.5 and 1)
t = [.5 1]; n = length(t);
Sigma = zeros(n,n);
for i = 1:n
    for j = 1:n
        Sigma(i,j) = min(t(i),t(j));
    end
end
X = chol(Sigma)' * randn(2,1);

plot([0; t'],[0; X]);
axis([0 1 -2.5 2.5]);

%Interpolate
for level = 2:num_levels
    %Make the additional mesh points
```

```
    t_new = 1/2^level : 2/2^level : (2^level-1)/(2^level);
    n_new = length(t_new);

    %Record the time points for the whole process
    t_temp = [t t_new];
    n_temp = length(t_temp);

    %Make a covariance matrix for the whole thing
    Sigma_temp = zeros(n_temp,n_temp);
    for i = 1:n_temp
        for j = 1:n_temp
            Sigma_temp(i,j) = min(t_temp(i),t_temp(j));
        end
    end

    %Make the separate Sigma components
    Sigma_11 = Sigma;
    Sigma_21 = Sigma_temp(n+1:n_temp, 1:n);
    Sigma_12 = Sigma_temp(1:n, n+1:n_temp);
    Sigma_22 = Sigma_temp(n+1:n_temp, n+1:n_temp);

    temp_mean = Sigma_21 * inv(Sigma_11) * X;
    Sigma_new = Sigma_22 - Sigma_21 * inv(Sigma_11) * Sigma_12;
    X_new = temp_mean + chol(Sigma_new)' * randn(n_new,1);
    X = [X; X_new];
    t = t_temp;
    n = n_temp;
    Sigma = Sigma_temp;
    [dummy index] = sort(t);
    another_dummy = waitforbuttonpress;

    plot([0; t(index)'],[0; X(index)]);
    axis([0 1 -2.5 2.5]);
end
```

3.1.7 Markovian Gaussian Processes

If a Gaussian process, $\{X_t\}_{t\geq 0}$, is Markovian, we can exploit this structure to simulate the process much more efficiently. Because $\{X_t\}_{t\geq 0}$ is Markovian, we can use we only need to know the value of X_{t_i} in order to generate $X_{t_{i+1}}$. Define

$$\sigma_{i,i+1} = \mathrm{Cov}(X_{t_i}, X_{t_{i+1}})$$

and

$$\mu_i = \mathbb{E}X_{t_i}.$$

By theorem 3.1.24, we know that $(X_{t_i}, X_{t_{i+1}})^\intercal$ has a multivariate normal distribution. In particular,

$$\begin{pmatrix} X_{t_i} \\ X_{t_{i+1}} \end{pmatrix} \sim \mathsf{N}\left(\begin{pmatrix} \mu_i \\ \mu_{i+1} \end{pmatrix}, \begin{pmatrix} \sigma_{i,i} & \sigma_{i,i+1} \\ \sigma_{i,i+1} & \sigma_{i+1,i+1} \end{pmatrix} \right).$$

Using theorem 3.1.25, we have

$$X_{t_{i+1}} \,|\, X_{t_i} = x_i \sim \mathsf{N}\left(\mu_i + \frac{\sigma_{i,i+1}}{\sigma i, i}(x_i - \mu_i)\,,\, \sigma_{i+1,i+1} - \frac{\sigma_{i,i+1}^2}{\sigma_{i,i}} \right).$$

Algorithm 3.1.2 (Generating a Markovian Gaussian Process)**.**

(i) Draw $Z \sim \mathsf{N}(0,1)$. Set $X_{t_1} = \mu_1 + \sqrt{\sigma_{i,i}}Z$.

(ii) For $i = 1, \ldots, m-1$ draw $Z \sim \mathsf{N}(0,1)$ and set

$$X_{t_{i+1}} = \mu_i + \frac{\sigma_{i,i+1}}{\sigma i, i}(X_{t_i} - \mu_i) + \left(\sqrt{\sigma_{i+1,i+1} - \frac{\sigma_{i,i+1}^2}{\sigma_{i,i}}} \right) Z.$$

Example 3.1.27 (Brownian Motion)**.** Consider Brownian motion, which is a Markov process. Now,

$$\mu_i = \mathbb{E}X_{t_i} = 0$$

and

$$\sigma_{i,i+1} = \min(t_i, t_{i+1}) = t_i.$$

So, our updating formula is

$$X_{t_{i+1}} = X_{t_i} + \left(\sqrt{t_{i+1} - t_i} \right) Z.$$

Listing 3.8: Matlab code

```
m = 10^4; X = zeros(m,1);
h = 1/m;
X(1) = sqrt(h)*randn;

for i = 1:m-1
    X(i+1) = X(i) + sqrt(h) * randn;
end

plot(h:h:1,X);
```

Example 3.1.28 (Ornstein-Uhlenbeck Process)**.** The Ornstein-Uhlenbeck process has

$$\mu_i = \mathbb{E}X_{t_i} = 0$$

with

$$\sigma_{i,i+1} = \exp\{\alpha|t_i - t_{i+1}|/2\}$$

and $\sigma_{i,i} = 1$. So, our updating formula is

$$X_{t_{i+1}} = \exp\{-\alpha|t_i - t_{i+1}|/2\}X_{t_i} + \left(\sqrt{1 - \exp\{\alpha|t_i - t_{i+1}|\}}\right) Z.$$

Listing 3.9: Matlab code

```
alpha = 50; m = 10^4;

X = zeros(m,1); h = 1/m;
X(1) = randn;

for i = 1:m-1
    X(i+1) = exp(-alpha * h / 2)*X(i)...
        + sqrt(1 - exp(-alpha * h))*randn;
end

plot(h:h:1,X);
```

3.2 Brownian Motion

One of the most fundamental stochastic processes. It is a Gaussian process, a Markov process, a Lévy process, a Martingale and a process that is closely linked to the study of harmonic functions (do not worry if you do not know all these terms). It can be used as a building block when considering many more complicated processes. For this reason, we will consider it in much more depth than any other Gaussian process.

Definition 3.2.1 (Brownian Motion)**.** A stochastic process $\{W_t\}_{t\geq 0}$ is called Brownian motion (a Wiener process) if:

(i) It has independent increments.

(ii) It has stationary increments.

(iii) $W_t \sim \mathsf{N}(0, t)$ for all $t \geq 0$.

(iv) It has almost surely continuous sample paths. That is,

$$\mathbb{P}(\{\omega : X(t,\omega) \text{ is continuous in } t\}) = 1.$$

This definition implies that the increments of Brownian motion are normally distributed. Specifically, $W_{t+s} - W_t \sim \mathsf{N}(0,s)$. This implies the following simualtion scheme.

Listing 3.10: Matlab code

```
m = 10^3; h = 1/m;
X = cumsum(sqrt(h)*randn(m,1));
plot(h:h:1,X);
```

The first three parts of the definition of Brownian motion are equivalent to saying Brownian motion is a Gaussian process with $\mathrm{Cov}(X_t, X_s) = \min(t,s)$ and $\mathbb{E}X_t = 0$. However, the almost sure continuity of the paths of Brownian motion does not follow from this.

Theorem 3.2.2. The following two statements are equivalent for a stochastic process $\{X_t\}_{t\geq 0}$.

(i) $\{X_t\}_{t\geq 0}$ has stationary independent increments and $X_t \sim \mathsf{N}(0,t)$.

(ii) $\{X_t\}_{t\geq 0}$ is a Gaussian process with $\mu(t) = \mathbb{E}X_t = 0$ and $r(s,t) = \mathrm{Cov}(X_t, X_s) = \min(t,s)$.

Proof.

Part 1. First, we show (i) implies (ii). In order to show this, we need to show that $(X_{t_1}, \ldots, X_{t_n})$ is multivariate normal for all choices of $0 \leq t_1 \leq \cdots \leq t_n$ and $n \geq 1$. In order to $X_{t_1}, \ldots, X_{t_n}$ to be multivariate normal, we need

$\sum_{k=1}^{n} \alpha_k X_{t_k}$ to be univariate normal (for all choices of n etc.). Now,

$$\begin{aligned}
\sum_{k=1}^{n} \alpha_k X_{t_k} &= \alpha_1 X_{t_1} + \sum_{k=2}^{n} \alpha_k X_{t_k} \\
&= \alpha_1 X_{t_1} + \sum_{k=2}^{n} \alpha_k \left(\sum_{j=1}^{k} X_{t_j} - \sum_{j=1}^{k-1} X_{t_j} \right) \\
&= \alpha_1 X_{t_1} + \sum_{k=2}^{n} \alpha_k \left(\sum_{j=1}^{k} X_{t_j} - \sum_{j=2}^{k} X_{t_{j-1}} \right) \\
&= \sum_{k=1}^{n} \alpha_k X_{t_1} + \sum_{k=2}^{n} \sum_{j=2}^{k} \alpha_k \left(X_{t_j} - X_{t_{j-1}} \right) \\
&= \left(\sum_{k=1}^{n} \alpha_k \right) X_{t_1} + \sum_{j=2}^{n} \left(\sum_{k=j}^{n} \alpha_k \right) \left(X_{t_j} - X_{t_{j-1}} \right) \\
&= \beta_1 X_{t_1} + \sum_{j=2}^{n} \beta_j \left(X_{t_j} - X_{t_{j-1}} \right).
\end{aligned}$$

Because X_{t_1} and $X_{t_2} - X_{t_1}, X_{t_3} - X_{t_2}, \ldots$ are independent random variables, it follows that the final equation results in a univariate normal random variable (sums of independent normals are normal). Now, we just need to check the expectation and covariance. It is easy to see that $\mathbb{E}X_t = 0$. In order to calculate the covariance, assume that $s < t$. We have that,

$$\mathrm{Cov}(X_s, X_t) = \mathbb{E}X_s X_t - \mathbb{E}X_s \mathbb{E}X_t = \mathbb{E}X_s X_t,$$

and

$$\mathbb{E}X_s X_t = \mathbb{E}X_s(X_t - X_s + X_s) = \mathbb{E}X_s^2 + \mathbb{E}X_s(X_t - X_s)$$

and, as the independent increments property implies $\mathbb{E}X_s(X_t - X_s) = 0$,

$$\mathrm{Cov}(X_s, X_t) = \mathbb{E}X_s^2 = s.$$

Repeating this with $t \geq s$, we get $\mathrm{Cov}(X_s, X_t) = \min(s, t)$.

Part 2. We show that (ii) implies (i). It follows immediately from the definition in (ii) that $X_t \sim \mathsf{N}(0, t)$. It is also immediate that $X_{t+s} - X_t$ is univariate normal (as this is a linear combination of multivariate normal random variables). We can thus show this increment is stationary by showing that mean is constant and the variance only depends on s. Now, $\mathbb{E}X_{t+s} - X_t = 0$ and

$$\mathrm{Var}(X_{t+s} - X_s) = \mathrm{Var}(X_t) + \mathrm{Var}(X_s) - 2\mathrm{Cov}(X_s, X_t) = t + s - 2t = s.$$

Thus, the increments are stationary. Because the increments are multivariate normal, we can show the increments of the process are independent if we can show the covariance between increments is 0. So, take $u < v \leq s < t$. Then,

$$\begin{aligned}
&\mathrm{Cov}(X_v - X_u, X_t - X_s) \\
&= \mathrm{Cov}(X_v, X_t) - \mathrm{Cov}(X_v, X_s) - \mathrm{Cov}(X_u, X_t) + \mathrm{Cov}(X_u, X_s) \\
&= \min(v,t) - \min(v,s) - \min(u,t) + \min(u,s) \\
&= v - v - u + u = 0.
\end{aligned}$$

So, non-overlapping increments are independent.

□

3.2.1 Existence

Because it is a nice proof, we will show that Brownian motion exists. We will do this constructively. Along the way, you might get some ideas about certain ways you might simulate Brownian motion. Before we get to the result itself, we need a couple of results. The first is a result that is often useful.

Theorem 3.2.3. Let $Z \sim \mathsf{N}(0,1)$. Then, as $x \to \infty$,

$$\mathbb{P}(Z > x) \sim \frac{1}{x}\frac{e^{-x^2/2}}{\sqrt{2\pi}},$$

and

$$\frac{x}{1+x^2}\frac{e^{x^2/2}}{\sqrt{2\pi}} < \mathbb{P}(Z > x) < \frac{1}{x}\frac{e^{-x^2/2}}{\sqrt{2\pi}}.$$

The next result is a little more technical and uninteresting (but cleans up the coming proof).

Lemma 3.2.4. Suppose we have two random variables X_s and X_t, with $t > s$, defined on the same probability space such that $X_t - X_s \sim \mathsf{N}(0, t-s)$. Then, there exists a random variable $X_{(t+s)/2}$ defined on the same space such that

$$X_{(t+s)/2} - X_s \stackrel{D}{=} X_t - X_{(t+s)/2},$$

with $X_{(t+s)/2} - X_s$ and $X_t - X_{(t+s)/2}$ independent and identically distributed with $\mathsf{N}(0, (t-s)/2)$ distributions. Also,

$$\left| X_{(t+s)/2} - \frac{X_t + X_s}{2} \right| = \frac{1}{2}|V|,$$

where $V \sim \mathsf{N}(0, t-s)$.

Proof. Let $U = X_t - X_s$. Take $V \sim \mathsf{N}(0, t-s)$ independent of U. Define $X_{(t+s)/2}$ by

$$X_t - X_{(t+s)/2} = \frac{U+V}{2},$$

and

$$X_{(t+s)/2} - X_s = \frac{U-V}{2}.$$

That is, we set

$$X_{(t+s)/2} = \frac{X_t}{2} + \frac{X_s}{2} - \frac{V}{2}.$$

Both $X_t - X_{(t+s)/2}$ and $X_{(t+s)/2} - X_s$ are clearly normal with mean 0. Now,

$$\mathrm{Var}\left(\frac{U+V}{2}\right) = \frac{\mathrm{Var}(U)}{4} + \frac{\mathrm{Var}(V)}{4} = (t-s)/2,$$

and

$$\mathrm{Var}\left(\frac{U-V}{2}\right) = \frac{\mathrm{Var}(U)}{4} + \frac{\mathrm{Var}(V)}{4} = (t-s)/2.$$

As they are normally distributed, we can show that $X_t - X_{(t+s)/2}$ and $X_{(t+s)/2} - X_s$ are independent by showing that their covariance is 0. Now,

$$\begin{aligned}
&\mathrm{Cov}\left(X_t - X_{(t+s)/2}, X_{(t+s)/2} - X_s\right) \\
&= \mathbb{E}\left(X_t - X_{(t+s)/2}\right)\left(X_{(t+s)/2} - X_s\right) - \mathbb{E}\left(X_t - X_{(t+s)/2}\right)\mathbb{E}\left(X_{(t+s)/2} - X_s\right) \\
&= \mathbb{E}\left(X_t - X_{(t+s)/2}\right)\left(X_{(t+s)/2} - X_s\right) - 0 = \mathbb{E}\left(\frac{U+V}{2}\right)\left(\frac{U-V}{2}\right) \\
&= \frac{1}{4}\mathbb{E}\left[U^2 - UV + UV - V^2\right] = \frac{1}{4}\mathbb{E}\left[U^2 - V^2\right] = 0.
\end{aligned}$$

For the final part, note that

$$\left|X_{(t+s)/2} - \frac{X_t + X_s}{2}\right| = \left|\frac{X_t}{2} + \frac{X_s}{2} - \frac{V}{2} - \frac{X_t}{2} - \frac{X_s}{2}\right| = \left|\frac{V}{2}\right|.$$

□

Remember that the difficult thing to show for Brownian motion is that it is almost surely continuous. To show this, we are going to consider the space $C[0,1]$, which is the space of continuous functions on $[0,1]$ with the norm $\|f\| = \max_{0\le t\le 1} |f(t)|$. Because we want to say something about convergence, we need to recall a couple of definitions.

Definition 3.2.5 (Cauchy Sequence)**.** We say a sequence (x_n) is a normed space is a *Cauchy sequence* if $\|x_n - x_m\| \to 0$ as $n, m \to \infty$. That is, for all $\epsilon > 0$, there is an integer $N(\epsilon)$ such that $\|x_n - x_m\| < \epsilon$ for all $m, n > N(\epsilon)$.

Definition 3.2.6 (Banach Space)**.** We say a normed linear vector space X is complete if every Cauchy sequence in X has a limit in X.

Lemma 3.2.7. The space $C[0,1]$ is a Banach space.

Proof. See almost any textbook on analysis. □

We now prove the main theorem.

Theorem 3.2.8. Brownian motion exists and may be constructed from a sequence of i.i.d. $\mathsf{N}(0,1)$ random variables.

Proof. We construct Brownian motion by constructing a Cauchy sequence on $C[0,1]$ that converges to Brownian motion. We will only construct it on $[0,1]$, but it is not too much work to extend this to $[0,\infty)$. We begin with a sequence of i.i.d. $\mathsf{N}(0,1)$ random variables $Z_1, Z_2, \ldots$, which we scale to get the set of independent normal random variables

$$\{V_{k/2^n} : k = 1, 2, \ldots, 2^n, n \geq 1\},$$

where $V_{i/2^{n+1}} \sim \mathsf{N}(0, 1/2^n)$.

Now, define $X_0 = 0$, $X_1 = V_1$ and use $V_{1/2}$ to construct $X_{1/2}$ using lemma 3.2.4. Then, $X_{1/2} - X_0$ and $X_1 - X_{1/2}$ are i.i.d. $\mathsf{N}(0, 1/2)$ random variables. If we continue, in an iterative fashion, using the supply of Vs and lemma 3.2.4, we can construct

$$\{X_{k/2^n}, 0 \leq k \leq 2^n, n \geq 1\},$$

such that

$$X_{\frac{2k+1}{2^{n+1}}} - X_{\frac{k}{2^n}} \quad \text{and} \quad X_{\frac{k+1}{2^n}} - X_{\frac{2k+1}{2^{n+1}}}$$

are independent and both have $\mathsf{N}(0, 1/2^{n+1})$ distributions. Now, for each $n \geq 1$, define $\{W_t^{(n)}\}_{t\in[0,1]}$ by

$$W_t^{(n)}(\omega) = \begin{cases} X_t(\omega) & \text{for } t \in \{\frac{k}{2^n}, 0 \leq k \leq 2^n\} \\ \text{linear in each interval } \left[\frac{k}{2^n}, \frac{k+1}{2^n}\right] & \text{for } t \notin \{\frac{k}{2^n}, 0 \leq k \leq 2^n\} \end{cases},$$

By construction, $W^{(n)}$ is continuous. If we can show that the $\{W^{(n)}\}_{n\geq 1}$ form a Cauchy sequence, then we know that their limit is also continuous (as $C[0,1]$ is a Banach space). In order to do this, define

$$\triangle^{(n)}(\omega) = \max_{0\leq t\leq 1} |W_t^{(n+1)}(\omega) - W_t^{(n)}(\omega)|$$
$$\max_{0\leq k\leq 2^{n-1}} \max_{\frac{k}{2^n}\leq t\leq \frac{k+1}{2^n}} |W_t^{(n+1)}(\omega) - W_t^{(n)}(\omega)|.$$

Now,

$$\begin{aligned}\max_{\frac{k}{2^n}\le t\le\frac{k+1}{2^n}} |W_t^{(n+1)}(\omega) - W_t^{(n)}(\omega)| &= \left| \frac{W_{k/2}^{(n)} + W_{(k+1)/2}^{(n)}}{2} - W_{\frac{2k+1}{2^{n+1}}}^{(n+1)} \right| \\ &= \left| \frac{X_{k/2} + X_{(k+1)/2}}{2} - X_{\frac{2k+1}{2^{n+1}}} \right|\end{aligned}$$

and, by lemma 3.2.4, we know that

$$\left| \frac{X_{k/2} + X_{(k+1)/2}}{2} - X_{\frac{2k+1}{2^{n+1}}} \right| = \frac{1}{2} \left| V_{\frac{2k+1}{2^{n+1}}} \right|,$$

where $V_{\frac{2k+1}{2^{n+1}}} \sim \mathsf{N}(0, 1/2^n)$. Thus,

$$\triangle^{(n)}(\omega) = \frac{1}{2} \max_{0\le k\le 2^{n+1}} \left| V_{\frac{2k+1}{2^{n+1}}} \right|.$$

If we can show that $\triangle^{(n)}(\omega) \to 0$ as $n \to 0$, then we have demonstrated that the $\{W^{(n)}\}_{n\ge 1}$ form a Cauchy sequence. We will show that this is true almost surely. To do this, we use the bound on normal tail probabilities given in theorem 3.2.3. Let $Z \sim \mathsf{N}(0,1)$ and observe

$$\begin{aligned}&\mathbb{P}\left(\triangle^{(n)} > \frac{x/2}{\sqrt{2^n}}\right) = \mathbb{P}\left(\frac{1}{2} \max_{0\le k\le 2^n/1} \left| V_{\frac{2k+1}{2^{n+1}}} \right| > \frac{1}{2}\frac{x}{\sqrt{2^n}}\right) \\ &\le 2^n \mathbb{P}\left(\left|\frac{1}{\sqrt{2^n}} Z\right| > \frac{x}{\sqrt{2^n}}\right) \le 2^n 2 \mathbb{P}\left(\frac{1}{\sqrt{2^n}} Z > \frac{x}{\sqrt{2^n}}\right) \\ &= 2^{n+1}\mathbb{P}\left(Z >\right) \le \frac{2^{n+1}}{\sqrt{2\pi}} \frac{e^{-x^2/2}}{x}.\end{aligned}$$

Thus, for $x = 2\sqrt{n}$, we have

$$\mathbb{P}\left(\triangle^{(n)} > \frac{\sqrt{n}}{\sqrt{2n}}\right) \le \frac{2^{n+1}}{\sqrt{2\pi}} \frac{e^{-2n}}{2\sqrt{n}} \le c\left(\frac{2}{e^2}\right)^n,$$

for some $c > 0$. Thus,

$$\sum_{n=1}^{\infty} \mathbb{P}\left(\triangle^{(n)} > \frac{\sqrt{n}}{\sqrt{2^n}}\right) \le \sum_{n+1}^{\infty} c\left(\frac{2}{e^2}\right)^n < \infty.$$

So, by the Borel-Cantelli lemma,

$$\mathbb{P}\left(\triangle^{(n)} > \frac{\sqrt{n}}{\sqrt{2^n}} \text{ infinitely often}\right) = 0.$$

This implies that, for all sufficiently large n, $\triangle^{(n)} \leq \sqrt{n/2^n}$. As a result, with probability one, $\sum_{n=1}^{\infty} \triangle^{(n)} < \infty$. Now, for $m > n$,

$$\sup_{0 \leq t \leq 1} \left| W_t^{(n)} - W_t^{(m)} \right| \leq \triangle^{(n)} + \cdots + \triangle^{(m-1)} \to 0$$

as $m, n \to \infty$. Thus, the $\{W^{(n)}\}_{n \geq 1}$ form a Cauchy sequence in $C[0,1]$ almost surely, and thus their limit is continuous. Thus, we define Brownian motion by

$$W = \begin{cases} \lim_{n \to \infty} B^{(n)} & \text{if } \lim_{n \to \infty} W^{(n)} \text{ exists} \\ 0 & \text{otherwise (with probability 0)} \end{cases}.$$

Of course, it remains to check that the resulting process has the desired properties. We will leave this as an exercise for the reader. If you are interested, take a look at [4]. □

3.2.2 Some useful results

It was already mentioned that Brownian motion is Markov in the section on Gaussian Markovian processes. Brownian motion is also strong Markov.

Theorem 3.2.9. Brownian motion is Markov and strong Markov.

Proof. See [2]. □

Historically, Brownian motion was first conceived of as the limit of a scaled random walk. The following theorem is a very famous result, sometimes called a *functional central limit theorem*, that justifies this interpretation.

Theorem 3.2.10 (Donsker's Theorem)**.** Let $\{X_i\}_{i \geq 0}$ be i.i.d. random variables with $\mathbb{E}X_1 = 0$ and $\mathbb{E}X_1^2 = 1$. Define the partials sums of these random variables by

$$S_n = X_1 + \cdots X_n.$$

Let

$$Z_n(t) = \frac{S_{[nt]}}{\sqrt{n}} \quad , \quad 0 \leq t \leq 1.$$

Then, $Z_n \stackrel{D}{\Rightarrow} W$, where W is a Brownian motion on $[0,1]$.

Proof. See [2]. □

3.2.3 Integration With Respect to Brownian Motion

Famously, Brownian motion has rougher paths than most of the deterministic functions we are used to. However, it turns out that there is still lots of order in the randomness.

Theorem 3.2.11. Brownian motion is almost surely not differentiable in the sense that, for every $0 \leq t \leq 1$,

$$\mathbb{P}\left(W(\omega) \text{ is not differentiable at } t\right) = 1.$$

Proof. See [2]. □

In order to understand stochastic differential equations, we will need to briefly discuss integration with respect to Brownian motion. The following concepts help clarify the difficulties in defining stochastic integrals.

Definition 3.2.12 (Bounded Variation)**.** A right continuous function has *bounded variation* on the interval $[0, t]$ if

$$V_f(t) = \sup \sum_{j=1}^{k} |f(t_j) - f(t_{j-1})| < \infty,$$

where the supremum is taken over all $k \in \mathbb{N}$ and partitions

$$0 = t_0 \leq t_1 \leq \cdots \leq t_{k-1} \leq t_k = t.$$

3.3 Stochastic Differential Equations

BASIC INTUITION. We will consider a special class of stochastic differential equations (SDEs) called *Itô diffusions*.

Definition 3.3.1 (Itô diffusion)**.** An Itô diffusion is a process, $\{X_t\}_{0 \leq t \leq T}$ satisfying

$$X_t = X_0 + \int_0^t a(X_u, u)\,\mathrm{d}u + \int_0^t b(X_u, u)\,\mathrm{d}W_u,$$

where $\{W_t\}_{0 \leq T}$ is a Bronwian motion. This equation is often written in shorthand as

$$\mathrm{d}X_t = a(X_t, t)\,\mathrm{d}t + b(X_t, t)\,\mathrm{d}W_t.$$

Sometimes $a(\cdot, \cdot)$ is called the drift coefficient and $b(\cdot, \cdot)$ is called the diffusion coefficient.

It is possible to have more general stochastic differential equations, where a and b are functions of the whole path up until time t. In addition, stochastic differential equations can be defined for more general classes of stochastic processes than Brownian motion.

In settings where a and b are linear, it is possible to find "closed form" solutions of SDEs. However, in general, SDEs need to be solved numerically.

3.3.1 Itô's Lemma

Itô's lemma is one of the most celebrated tools in modern mathematics. Basically, it allows us to write out the SDE for a function of another SDE. Sometimes, people say that it is the stochastic equivalent of the chain rule (though this is a little confusing, as we are not really differentiating at any point). It holds for more general processes than just Itô diffusions.

Definition 3.3.2 (Itô Process)**.** An Itô process is a stochastic process $\{X_t\}_{0\leq t\leq T}$ satisfying

$$X_t = X_0 + \int_0^t \mu_s \,\mathrm{d}s + \int_0^t \sigma_s \mathrm{d}W_s,$$

where $\{\mu_t\}_{t\geq 0}$ and $\{\sigma_t\}_{t\geq 0}$ are (possibly stochastic) adapted processes such that $\int_0^T |\mu_s| \,\mathrm{d}s < \infty$ almost surely and $\int_0^T |\sigma_s| \,\mathrm{d}s < \infty$ almost surely. In shorthand, this is written as

$$\mathrm{d}X_t = \mu_t \,\mathrm{d}t + \sigma_t \,\mathrm{d}W_t.$$

Note that $\{\mu_t\}_{t=0}^T$ and $\{\sigma_t\}_{t=0}^T$ can be functionals of the whole path up until time t (i.e., $\{X_s\}_{s=0}^t$). However, we will usually consider processes of the form $\mu_t = a(X_t, t)$ and $\sigma_t = b(X_t, t)$.

Lemma 3.3.3 (Itô's Lemma)**.** Let $\{X_t\}_{t=0}^T$ be an Itô process given by $\mathrm{d}X_t = \mu_t \,\mathrm{d}t + \sigma_t \,\mathrm{d}W_t$ and let $g(t,x) \in C^2([0,\infty) \times \mathbb{R})$. Then, $\{Y_t\}_{t=0}^T$ defined by $Y_t = g(t, X_t)$ is again an Itô process and

$$\mathrm{d}Y_t = \frac{\partial g}{\partial t}(t, X_t) \,\mathrm{d}t + \frac{\partial g}{\partial x}(t, X_t) \,\mathrm{d}X_t + \frac{1}{2}\frac{\partial^2 g}{\partial x^2}(t, X_t)(\mathrm{d}X_t)^2,$$

where $(\mathrm{d}X_t)^2$ is computed according to the rules $\mathrm{d}t \,\mathrm{d}t = 0$, $\mathrm{d}t \,\mathrm{d}W_t = \mathrm{d}W_t \,\mathrm{d}t = 0$ and $\mathrm{d}W_t \,\mathrm{d}W_t = \mathrm{d}t$.

3.3.2 Numerical Solutions of SDEs

A numerical solution to an SDE is an approximate solution of the SDE given a realization of a Brownian path, $\{W_t\}_{t=0}^T$. Most such numerical

schemes are not random, in the sense that, given a realization of a Brownian motion and a set of parameters, there is no randomness involved. We will consider the two most widely used schemes, the *Euler method* and the *Milstein method*.

The Euler Method

Consider the ordinary differential equation (ODE)

$$f' = a(f(t), t). \tag{3.1}$$

This is the same as considering the stochastic differential equation

$$\mathrm{d}X_t = a(X_t, t)\mathrm{d}t,$$

(i.e., the SDE with no stochastic coefficient). If a is a sufficiently complicated function, this can be difficult to solve. The Euler method was first used for deterministic differential equations like this one. We can derive it in two ways, both of which give us intuition about how to come up with such schemes. In the first, we can use the Taylor approximation of f around t_0 to get

$$\begin{aligned} f(t_0 + h) &= f(t_0) + f'(t_0)h + \frac{1}{2}f''(t_0)h^2 + O(h^3) \\ \Rightarrow f(t_0 + h) &\approx f(t_0) + f'(t_0)h = f(t_0) + a(f(t), t)h, \end{aligned}$$

for small h. This means that, if we know $f(t_0)$, we know (approximately) $f(t_0 + h)$. Thus, if we are able to start with something we know (usually $f(0)$), and a partition

$$0 = t_0 < t_1 < \cdots < t_k = t, \quad \text{where } t_j - t_{j-1} = h,\ 1 \le j \le k,$$

we can successively approximate $f(t)$ using the recursion

$$\widehat{f}(t_j) = \widehat{f}(t_{j-1}) + a(\widehat{f}(t_{j-1}), t_{j-1})h.$$

Alternatively, we can integrate both sides of equation (3.1) to get

$$\int_0^{t+h} f'(u)\,\mathrm{d}u = \int_0^{t+h} a(f(t), t)\,\mathrm{d}u = f(t) + \int_t^{t+h} a(f(u), u)\,\mathrm{d}u,$$

which we then approximate by

$$f(t) + \int_t^{t+h} f(u)\,\mathrm{d}u \approx f(t) + a(f(t), t)\int_t^{t+h} \mathrm{d}u = f(t) + a(f(t), t)\,h.$$

This leads to the same scheme.

The Euler approach for SDEs is more or less identical. Remember that, by definition,

$$\mathrm{d}X_t = a(X_t, t)\,\mathrm{d}t + b(X_t, t)\,\mathrm{d}W_t,$$

is

$$X_t = X_0 + \int_0^t a(X_u, u)\,\mathrm{d}u + \int_0^t b(X_u, u)\,\mathrm{d}W_u.$$

Thus, we can write

$$\begin{aligned} X_{t+h} &= X_t + \int_t^{t+h} a(X_u, u)\,\mathrm{d}u + \int_t^{t+h} b(X_u, u)\,\mathrm{d}W_u \\ &\approx X_t + a(X_t, t)\int_t^{t+h} \mathrm{d}u + b(X_t, t)\int_t^{t+h} \mathrm{d}W_u \\ &= X_t + a(X_t, t)\,h + b(X_t, t)\,(W_{t+h} - W_t). \end{aligned}$$

As we know that $W_{t+h} - W_t \sim \mathsf{N}(0, h)$, this leads to the approximating process defined recursively on the partition

$$0 = t_0 < t_1 < \cdots < t_k = t, \quad \text{where } t_j - t_{j-1} = h,\ 1 \le j \le k,$$

by $\widehat{X}_{t_0} = X_0$ and

$$\widehat{X}_{t_{j+1}} = \widehat{X}_{t_j} + a(\widehat{X}_{t_j}, t_j)\,h + b(\widehat{X}_{t_j}, t_j)\sqrt{h}Z_j,$$

where the $\{Z_j\}_{j=0}^{k-1}$ are i.i.d. $\mathsf{N}(0, 1)$.

Example 3.3.4. Consider the SDE for geometric Brownian motion. This is given by

$$\mathrm{d}X_t = \mu X_t\,\mathrm{d}t + \sigma X_t\,\mathrm{d}W_t.$$

We can simulate this using the Euler scheme as follows.

Listing 3.11: Matlab code

```
m = 10^3; h = 1/m;
mu = 0.1; sigma = 0.2; X_0 = 1;
X = zeros(1,m+1);
X(1) = X_0;
for i = 2:m+1
  X(i) = X(i-1) + mu*X(i-1)*h + sigma*X(i-1)*sqrt(h)*randn;
end
plot(0:h:1,X)
```

Milstein's Method

The idea of Milstein's method is to improve on the approximation

$$\int_t^{t+h} b(X_u, u)\,\mathrm{d}W_u \approx b(X_t, t)\int_t^{t+h} \mathrm{d}W_u.$$

To do this, we use Itô's lemma to get an expression for $b(X_t, t)$ that allows us to get a better approximation. The SDE we wish to solve is

$$\mathrm{d}X_t = a(X_t, t)\,\mathrm{d}t + b(X_t, t)\,\mathrm{d}W_t.$$

We set $g(X_t, t) = b(X_t, t)$. Let $b_t(x,t) = \frac{\partial}{\partial t}b(x,t)$, $b_x(x,t) = \frac{\partial}{\partial x}b(X_t,t)$ and $b_{xx}(x,t) = \frac{\partial^2}{\partial x^2}b(X_t,t)$ and use Itô's lemma to get the expression

$$\begin{aligned}\mathrm{d}b(X_t,t) &= b_t(X_t,t)\,\mathrm{d}t + b_x(X_t,t)\,\mathrm{d}X_t + \frac{1}{2}b_{xx}(X_t,t)(\mathrm{d}X_t)^2\\ &= b_t(X_t,t)\,\mathrm{d}t + b_x(X_t,t)\,[a(X_t,t)\,\mathrm{d}t + b(X_t,t)\,\mathrm{d}W_t]\\ &\quad + \frac{1}{2}b_{xx}(X_t,t)\,b(X_t,t)^2\,\mathrm{d}t.\end{aligned}$$

Writing this in integral form with $s > t$, we have

$$\begin{aligned}b(X_s,s) =& b(X_t,t) + \int_t^s b_t(X_u,u)\mathrm{d}u + \int_t^s b_x(X_u,u)\,a(X_u,u)\,\mathrm{d}u\\ &+ \int_t^s b_x(X_u,u)\,b(X_u,u)\,\mathrm{d}W_u + \frac{1}{2}\int_t^s b_{xx}(X_u,u)\,b(X_u,u)^2\,\mathrm{d}u.\end{aligned}$$

Thus,

$$\begin{aligned}\int_t^{t+h} b(X_s,s)\,\mathrm{d}W_s = \int_t^{t+h}\Bigg[&b(X_t,t) + \int_t^s b_t(X_u,u)\,\mathrm{d}u + \int_t^s b_x(X_u,u)\,a(X_u,u)\,\mathrm{d}u\\ &+ \int_t^s b_x(X_u,u)\,b(X_u,u)\,\mathrm{d}W_u + \frac{1}{2}\int_t^s b_{xx}(X_u,u)\,b(X_u,u)^2\,\mathrm{d}u\Bigg]\mathrm{d}W_s\end{aligned}$$

Disregarding all terms involving $\mathrm{d}u\,\mathrm{d}W_s$, we get that

$$\begin{aligned}\int_t^{t+h} b(X_s,s)\,\mathrm{d}W_s &\approx \int_t^{t+h} b(X_t,t)\,\mathrm{d}W_s + \int_t^{t+h}\int_t^s b_x(X_u,u)\,b(X_u,u)\,\mathrm{d}W_u\,\mathrm{d}W_s\\ &\approx b(X_t,t)\int_t^{t+h}\mathrm{d}W_s + b_x(X_t,t)\,b(X_t,t)\int_t^{t+h}\int_t^s \mathrm{d}W_u\,\mathrm{d}W_s\\ &\approx b(X_t,t)(W_t - W_s) + b_x(X_t,t)\,b(X_t,t)\frac{1}{2}\left[(W_{t+h} - W_t)^2 - h\right]\end{aligned}$$

This leads to the approximating process defined recursively on the partition

$$0 = t_0 < t_1 < \cdots < t_k = t, \quad \text{where } t_j - t_{j-1} = h,\ 1 \le j \le k,$$

by $\widehat{X}_{t_0} = X_0$ and

$$\widehat{X}_{t_{j+1}} = \widehat{X}_{t_j} + a(\widehat{X}_{t_j}, t_j)\, h + b(\widehat{X}_{t_j}, t_j)\sqrt{h} Z_j + b_x(\widehat{X}_{t_j}, t_j) b(\widehat{X}_{t_j}, t_j) \frac{h}{2}(Z_j^2 - 1),$$

where the $\{Z_j\}_{j=0}^{k-1}$ are i.i.d. $\mathsf{N}(0,1)$.

Example 3.3.5. Consider again the SDE for geometric Brownian motion. That is,

$$\mathrm{d}X_t = \mu X_t \,\mathrm{d}t + \sigma X_t \,\mathrm{d}W_t.$$

In order to simulate this using the Milstein scheme, we need to find $b_x(x,t)$. Now, $b(x,t) = \sigma x$, so $b_x(x,t) = \sigma$.

Listing 3.12: Matlab code

```
m = 10^3; h = 1/m;
mu = 0.1; sigma = 0.2; X_0 = 1;
X = zeros(1,m+1);
X(1) = X_0;
for i = 2:m+1
    Z = randn;
    X(i) = X(i-1) + mu*X(i-1)*h + sigma*X(i-1)*sqrt(h)*Z...
        + sigma^2 * X(i-1) * h/2 * (Z^2 - 1);
end
plot(0:h:1,X)
```

3.3.3 Multidimensional SDEs

Itô diffusions can also be defined in higher dimensions.

Definition 3.3.6 (Itô Diffusion (Multidimensional Version))**.** A multidimensional Itô diffusion process is the solution of

$$\mathrm{d}\mathbf{X}_t = a(\mathbf{X}_t, t)\,\mathrm{d}t + B(\mathbf{X}_t, t)\,\mathrm{d}\mathbf{W}_t,$$

where $\mathbf{X}_t \in \mathbb{R}^m$, $a(\mathbf{x}, t) : [0,T] \times \mathbb{R}^m \to \mathbb{R}^m$, $B(\mathbf{x}, t) : [0,T] \times \mathbb{R}^{m+n} \to \mathbb{R}^m$, and $\{\mathbf{W}_t\}_{t \ge 0}$ is an n-dimensional Brownian motion.

The Euler scheme is easily applied in multiple dimensions.

Example 3.3.7 (Simplified Duffing-Van der Pol Oscillator)**.** Consider the simplified Duffing-Van der Pol oscillator. This is given by the multidimensional SDE

$$\begin{aligned} \mathrm{d}X_t &= Y_t \mathrm{d}t \\ \mathrm{d}Y_t &= \left(X_t(\alpha - X_t^2) - Y_t\right) \mathrm{d}t + \sigma X_t \mathrm{d}W_t. \end{aligned}$$

This is easily simulated in Matlab, using an obvious extension of the Euler scheme.

Listing 3.13: Matlab code

```
alpha = 1; sigma = 0.5;
m = 10^3; h = 1/m; T = 10^3;

X = zeros(T*m+1,1); Y = zeros(T*m+1,1);

X(1) = -2; Y(1) = 0;

for k = 1:T*m
    X(k+1) = X(k) + Y(k) * h;
    Y(k+1) = Y(k) + (X(k) * (alpha - X(k)^2) - Y(k))*h...
        + sigma*X(k)*sqrt(h)*randn;
end

figure(1),plot(h:100*h:T,X(1:100:T*m))
figure(2),plot(h:100*h:T,Y(1:100:T*m))
figure(3),plot(X,Y)
```

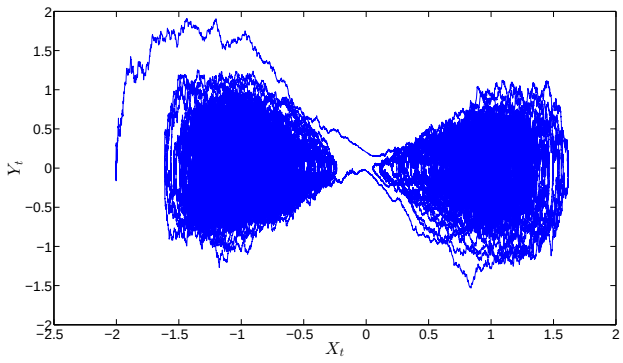

Figure 3.3.1: Plot of $\{X_t\}_{t\geq 0}$ against $\{Y_t\}_{t\geq 0}$ for the simplified Duffing-Van der Pol oscillator.

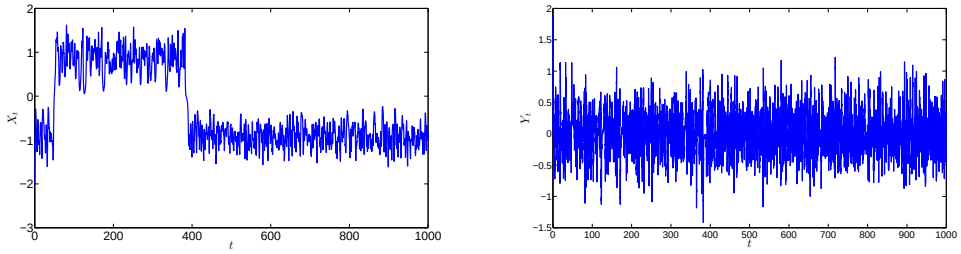

Figure 3.3.2: Plots of $\{X_t\}_{t\geq 0}$ and $\{Y_t\}_{t\geq 0}$ for the simplified Duffing-Van der Pol oscillator.

3.4 Existence and Uniqueness Result

Whenever you solve a differential equation numerically, you should always check that a solution exists first. Otherwise, you will just end up with a bunch of garbage. There are a number of different existence and uniqueness solutions for SDEs. The following is not the most general. However, it has conditions that are straightforward to check. There are a number of SDEs not covered by the following theorem. However, if you come across on of these, you should be able to find the relevant theorem somewhere.

Theorem 3.4.1 (Existence and Uniqueness of Solutions to SDEs)**.** Let $T > 0$ and let , $a(\mathbf{x}, t) : [0, T] \times \mathbb{R}^m \to \mathbb{R}^m$ and $B(\mathbf{x}, t) : [0, T] \times \mathbb{R}^{m+n} \to \mathbb{R}^m$ be measurable functions satisfying

(i) $|a(\mathbf{x}, t)| + |B(\mathbf{x}, t) - B(\mathbf{y}, t)| \leq C(1 + |\mathbf{x}|)$ for all $\mathbf{x} \in \mathbb{R}^m$ and $t \in [0, T]$, where $|B| = \sqrt{\sum |B_{ij}|}$ and C is a constant.

(ii) $|a(\mathbf{x}, t) - a(\mathbf{y}, t)| + |B(\mathbf{x}, t) - B(\mathbf{y}, t)| \leq D|\mathbf{x} - \mathbf{y}|$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ and $t \in [0, T]$ where D is a constant.

Furthermore, let $\mathbf{Z}$ be a random variable which is independent of the σ-algebra $\mathcal{F}_\infty^{(n)}$ generated by $\{\mathbf{W}_s\}_{s\leq t}$ and such that $\mathbb{E}|\mathbf{Z}|^2 \leq \infty$. Then, the stochastic differential equation

$$\mathrm{d}\mathbf{X}_t = a(\mathbf{X}_t, t)\mathrm{d}t + B(\mathbf{X}_t, t)\mathrm{d}W_t, \quad 0 \leq t \leq T, \, \mathbf{X}_0 = \mathbf{Z},$$

has a unique t-continuous solution $\{\mathbf{X}_t(\omega)\}_{t\in[0,T]}$ that is adapted to the filtration $\mathcal{F}_t^{\mathbf{Z}}$ generated by $\mathbf{Z}$ and $\{\mathbf{W}_s\}_{s\leq t}$ and

$$\mathbb{E}\int_0^T |X_t|^2 \, \mathrm{d}t < \infty.$$

3.5 SDEs and PDEs

There

3.6 Error Analysis for Numerical Solutions of SDEs

Given a realization of a Brownian motion, a numerical solution gives an approximate solution, $\{\widehat{\mathbf{X}}_t\}_{t\geq 0}$ to a specified SDE, $\{\mathbf{X}_t\}_{t\geq 0}$. This solution will not be completely correct. However, as the step size h becomes smaller, it should become better. In order to emphasize that a numerical solution depends on its step size, we will write $\{\widehat{\mathbf{X}}_{t,h}\}_{t\geq 0}$ in this section. There are a number of different ways to measure the error of an approximate solution of an SDE. For example, we could measure the average difference between the true solution and numerical solution. Alternatively, we could measure the difference in distribution between the true solution and the numerical solutions. These two approaches are embodied in the notions of *weak convergence* and *strong convergence* of numerical solutions of SDEs.

Let C_P^r be the space of r times continuously differentiable functions with polynomial growth and polynomial growth of derivatives of order up to and including r.

Definition 3.6.1 (Weak Convergence)**.** We say that a numerical solution, $\{\widehat{\mathbf{X}}_{t,h}\}_{t\geq 0}$, converges weakly with order $\beta > 0$ to the exact solution, $\{\mathbf{X}_t\}_{t\geq 0}$, at time T and as $h \downarrow 0$ if, for each $g \in C_P^{2(\beta+1)}$, there exists $C > 0$ independent of h and $0 < h_0 < \infty$ such that

$$\left\| \mathbb{E}g(\mathbf{X}_T) - \mathbb{E}g(\widehat{\mathbf{X}}_{T,h}) \right\| \leq C\, h^\beta, \quad \forall h \in (0, h_0).$$

Definition 3.6.2 (Strong Convergence)**.** We say that a numerical solution, $\{\widehat{\mathbf{X}}_{t,h}\}_{t\geq 0}$, converges strongly with order $\gamma > 0$ to the exact solution, $\{\mathbf{X}_t\}_{t\geq 0}$, at time T and as $h \downarrow 0$ if, for each $g \in C_P^{2(\beta+1)}$, there exists $C > 0$ independent of h and $0 < h_0 < \infty$ such that

$$\mathbb{E}\left\| \mathbf{X}_T - \widehat{\mathbf{X}}_{T,h} \right\| \leq C\, h^\gamma, \quad \forall h \in (0, h_0).$$

Under some technical conditions (see, e.g., ...) the Euler scheme has weak order convergence rate $\beta = 1$ and strong order convergence rate $\gamma = 0.5$. Under some different technical conditions (see, e.g., ...) the Milstein scheme

has weak order convergence rate $\beta = 1$ and strong order convergence rate $\gamma = 1$.

In general, in Monte Carlo we are interested in estimating expectations. This means that weak order convergence is usually the relevant convergence concept. It is important to bear this in mind. Although the Milstein scheme appears better, it involves calculating derivatives (sometimes these can be done by hand in advance, but sometimes they need to be done numerically). This imposes an additional cost which may not be worth it if strong order convergence is not important.

Example 3.6.3. Consider the SDE for geometric Brownian motion

$$\mathrm{d}X_t = \mu\, X_t\, \mathrm{d}t + \sigma\, X_t\, \mathrm{d}W_t.$$

This has the exact solution

$$X_t = X_0 \exp\left\{\left(\mu - \frac{\sigma^2}{2}\right) t + \sigma W_t\right\}.$$

For a given realization of $\{W_t\}_{t\in[0,1]}$, we can compare the numerical solutions with the exact solutions. The following code generates a realization of Brownian motion (at a fixed number of points) and then plots the Euler and Milstein approximations against the exact solution for a number of mesh sizes.

Listing 3.14: Matlab code

```
L = 4; m_max = 2^L;
mu = 0.1; sigma = 0.2; X_0 = 1;

Z = randn(m_max,1);

for l = 1:L
    m = 2^l; h = 1/m;
    Z_coarse = zeros(m,1);
    X_eul = zeros(1,m+1); X_eul(1) = X_0;
    X_mil = zeros(1,m+1); X_mil(1) = X_0;
    X = zeros(1,m+1); X = X_0;

    for k = 1:m
        Z_coarse(k) = 1/sqrt(2^(L-l))...
            *sum(Z((k-1)*2^(L - l)+1:k*2^(L - l)));
    end
    W_coarse = sqrt(h) * cumsum(Z_coarse);
    for i = 2:m+1
```

```
        X_eul(i) = X_eul(i-1) + mu*X_eul(i-1)*h ...
            + sigma*X_eul(i-1)*sqrt(h)*Z_coarse(i-1);
        X_mil(i) = X_mil(i-1) + mu*X_mil(i-1)*h ...
            + sigma*X_mil(i-1)*sqrt(h)*Z_coarse(i-1)...
            + sigma^2 * X_mil(i-1) * h/2 * (Z_coarse(i-1)^2 - 1);
        X(i) = X_0...
            *exp((mu - sigma^2 / 2)*((i-1)*h) + sigma*W_coarse(i-1));
    end
    figure(l); hold on;
    plot(0:h:1,X,'black')
    plot(0:h:1,X_eul,'red')
    plot(0:h:1,X_mil,'blue')
    legend('Exact','Euler Approximation','Milstein Approximation')
end
```

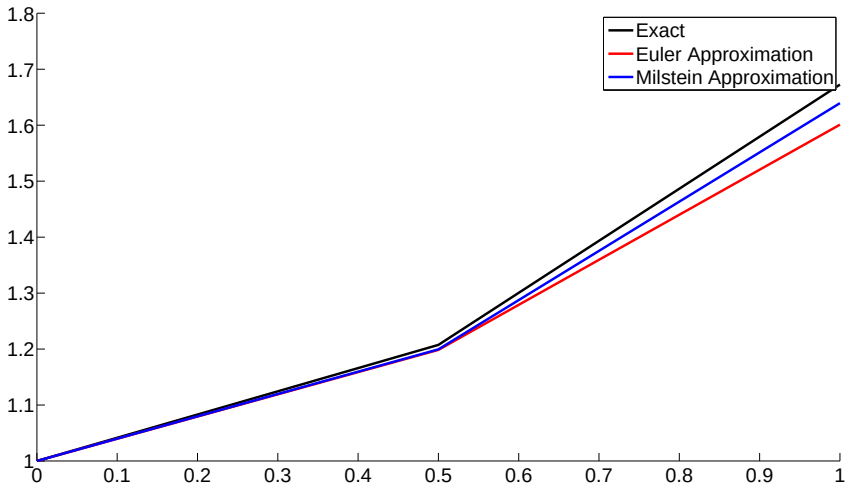

Figure 3.6.1: Plots of the exact solution, Euler approximation and Milstein approximation of geometric Brownian motion for a fixed realization of $\{W_t\}_{t\in[0,1]}$ with mesh size $h = 1/2$

As we have often seen, the common object of interest in Monte Carlo is estimating $\ell\mathbb{E}f(\mathbf{X})$, where $\mathbf{X}$ is a random vector. In the case of SDEs, $\mathbf{X}$ is often $\mathbf{X}_T$, the value of the solution of an SDE at time T. There are other, more complicated, things that we might wish to estimate, such as functionals of paths of SDEs. However, I will not focus too much on these.

If we wish to estimate $\ell = \mathbb{E}f(\mathbf{X}_T)$, we generally use the standard Monte

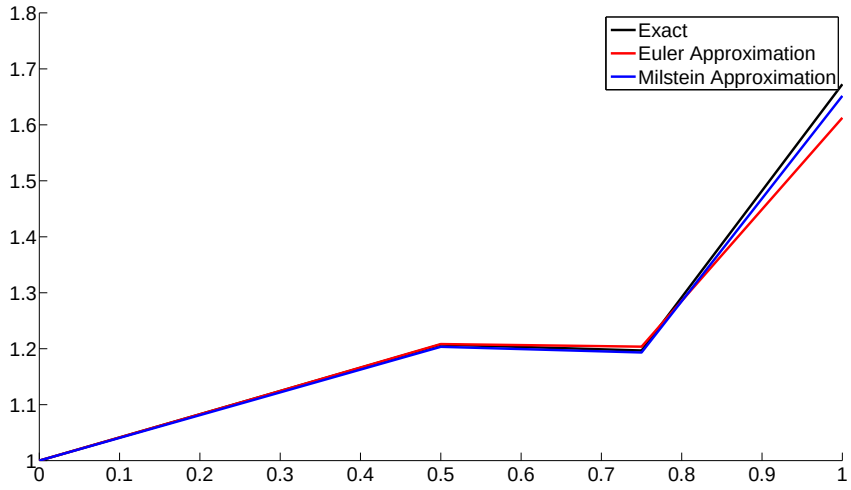

Figure 3.6.2: Plots of the exact solution, Euler approximation and Milstein approximation of geometric Brownian motion for a fixed realization of $\{W_t\}_{t\in[0,1]}$ with mesh size $h = 1/4$

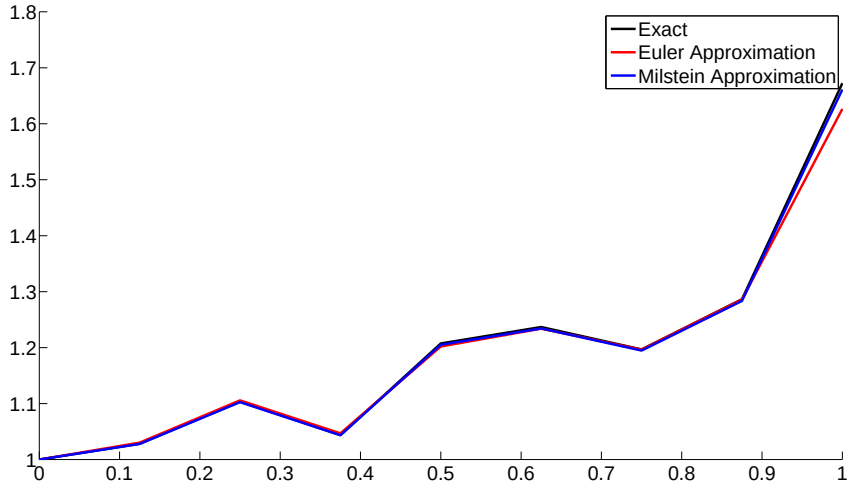

Figure 3.6.3: Plots of the exact solution, Euler approximation and Milstein approximation of geometric Brownian motion for a fixed realization of $\{W_t\}_{t\in[0,1]}$ with mesh size $h = 1/8$

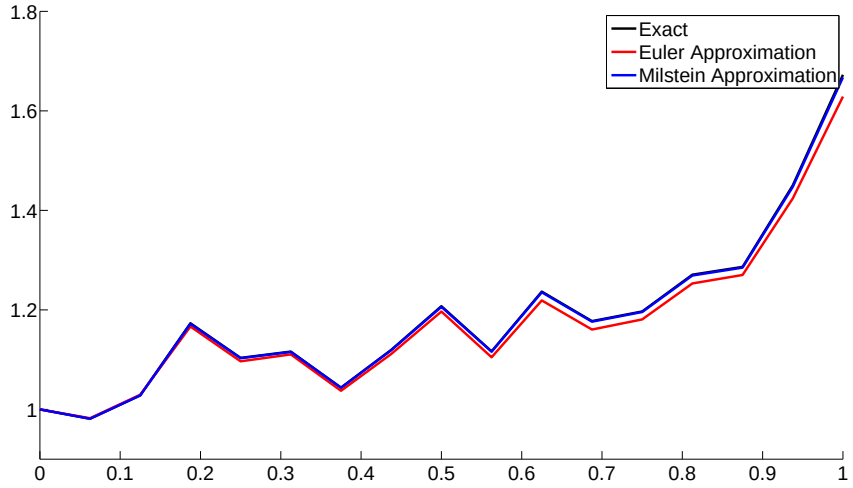

Figure 3.6.4: Plots of the exact solution, Euler approximation and Milstein approximation of geometric Brownian motion for a fixed realization of $\{W_t\}_{t\in[0,1]}$ with mesh size $h = 1/16$

Carlo estimator

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N} f\left(\mathbf{X}_{T,h}^{(i)}\right),$$

where $\mathbf{X}_{T,h}^{(1)}, \ldots, \mathbf{X}_{T,h}^{(N)}$ are iid copies of the numerical solution of with mesh size h. In general, this estimator is biased. This is because both the Euler and Milstein schemes are biased. This means that we should measure its error using MSE. Remember that the MSE of an estimator is of the form

$$\mathsf{MSE}(\widehat{\ell}) = \mathrm{Var}(\widehat{\ell}) + \mathsf{Bias}(\widehat{\ell})^2.$$

The two parts of the MSE correspond to different types of error: *statistical error* and *approximation error*. The statistical error, measured by the variance, comes from the fact that there are an uncountably infinite number of paths of Brownian motion and we are only sampling a small number. The approximation error, measured by the bias, comes from the fact that given a realization of a Brownian path, we still only get an approximate solution to our SDE.

If we think a little harder, we can write the MSE as

$$\mathsf{MSE}(\widehat{\ell}) = \mathrm{Var}(\widehat{\mathbf{X}}_{T,h}) + \left\|\mathbb{E}f(\mathbf{X}_T) - \mathbb{E}f(\widehat{\mathbf{X}}_{T,h})\right\|^2.$$

Note that the rate at which the second term on the right goes to zero is measured by the order of weak convergence. When the order of weak convergence is 1 and for large enough N and small enough h we can approximate the MSE by

$$\mathsf{MSE}(\widehat{\ell}) \approx \frac{C_1}{N} + C_2 h^2,$$

where $C_1, C_2 > 0$ are constants.

Remember that we defined the work done by an estimator as

$$\mathsf{work} = \text{number of samples} \times \text{work to make one sample.}$$

In our case, assuming work increases linearly in sample size and inversely in step size, we can write

$$\mathsf{work}(\widehat{\ell}) \propto N/h.$$

3.7 Multilevel Monte Carlo

Suppose we want the *root mean square error* (RMSE) of our estimator, that is $\mathsf{RMSE}(\widehat{\ell}) = \sqrt{\mathsf{MSE}}$, to be of order ϵ. Then, we need $\mathsf{MSE} = O(\epsilon^2)$. This is true if $\mathrm{Var}(\widehat{\ell}) = O(\epsilon^2)$ and $\mathsf{Bias}(\widehat{\ell}) = O(\epsilon^2)$. If we are using the Euler scheme, this means that we need $N = O(\epsilon^2)$ and $h = O(\epsilon^{-1})$. This, in turn, implies that the work our estimator need to do is $O(\epsilon^3)$. Many variance reduction schemes aim to reduce the cost (measured by work) of the estimator while retaining its accuracy.

The key insight of multilevel Monte Carlo, which was introduced in [1], is that, in some sense, most of the information about where a SDE is likely to be at time T can be obtained by looking at the results of very coarse approximations (i.e. numerical solutions with large values of h). Thus, a good idea is to try to spend most of the computational effort on estimating values using these coarse approximations.

3.7.1 The Multilevel Estimator

Consider a sequence of increasingly smaller step sizes, $\{h(l)\}_{l=0}^{L}$, with $h(l) = M^{-l}T$. A common choice of M is 2. In this case, if $T = 1$, $h(0) = 1$, $h(1) = 1/2$, $h(2) = 1/4$, $h(3) = 1/8$, etc.

Let $P = f(X_T)$. Then, clearly, we wish to estimate $\ell = \mathbb{E}P$. Likewise, define $\widehat{P}_l = f(\widehat{X}_{T,h(l)})$. Then, a good approximation of $\mathbb{E}P$ is $\mathbb{E}\widehat{P}_L$, where L is large (as $L \to \infty$ this expectation should converge to the correct value).

Now, observe that we can write $\mathbb{E}P_L$ as

$$\begin{aligned}\mathbb{E}\widehat{P}_L &= \mathbb{E}\widehat{P}_0 + \sum_{l=1}^{L}\left(\mathbb{E}\widehat{P}_l - \mathbb{E}\widehat{P}_{l-1}\right)\\ &= \mathbb{E}\widehat{P}_0 + \sum_{l=1}^{L}\mathbb{E}(\widehat{P}_l - \widehat{P}_{l-1})\end{aligned}$$

We can estimate these using independent estimators for each summand, which are defined by

$$\mathbb{E}\widehat{P}_0 \approx Y_0 = \frac{1}{N_0}\sum_{i=1}^{N_0} f\left(\widehat{X}^{(i)}_{T,h(0)}\right)$$

and, for $l = 1, \ldots, L$,

$$\mathbb{E}(\widehat{P}_l - \widehat{P}_{l-1}) \approx Y_l = \frac{1}{N_l}\sum_{i=1}^{N_l}\left[f\left(\widehat{X}^{(i)}_{T,h(l)}\right) - f\left(\widehat{X}^{(i)}_{T,h(l-1)}\right)\right],$$

where, very importantly, $\widehat{X}^{(i)}_{T,h(l)}$ and $\widehat{X}^{(i)}_{T,h(l-1)}$ are generated using the same realization of Brownian motion. Then, $\mathbb{E}\widehat{P}_L$ can be estimated by $Y(L)$, where

$$Y(L) = Y_0 + \sum_{l=1}^{L} Y_l.$$

3.7.2 Variance, Work and Optimal Sample Sizes

The variance of the estimator is given by

$$\mathrm{Var}(Y(L)) = \mathrm{Var}\left(Y_0 + \sum_{l=1}^{L} Y_l\right) = \sum_{l=0}^{L}\frac{V_l}{N_l},$$

where $V_0 = \mathrm{Var}(f(\widehat{X}_{T,h(0)}))$ and, for $l = 1, \ldots, L$,

$$V_l = \mathrm{Var}\left(f(\widehat{X}_{T,h(l)}) - f(\widehat{X}_{T,h(l-1)})\right).$$

We can write work done by this estimator in terms of the various sample sizes and step sizes as

$$\mathsf{work}(Y(L)) \propto \sum_{l=0}^{L}\frac{N_l}{h_l}.$$

Taking a fixed work budget, proportional to W, and assuming that we do not need integer sample sizes (rounding afterwards should more or less preserve optimality), we can find the optimal sample sizes using standard calculus. We solve

$$\nabla_N \left(\sum_{l=0}^{L} \frac{V_l}{N_l} + \lambda \left(\sum_{l=0}^{L} \frac{N_l}{h_l} - W \right) \right) = \mathbf{0}$$

to get

$$N_l = \sqrt{\frac{V_l h_l}{\lambda}} \propto \sqrt{V_l h_l}.$$

3.7.3 A Rough Sketch of Why Multilevel Monte Carlo Works

If we use the Euler scheme, we have that, as $h \to 0$,

$$\mathbb{E}\left[\widehat{P}_l - P\right] = O(h_l),$$

as this is measured using weak convergence. Likewise, we have

$$\mathbb{E}\left\|X_T - \widehat{X}_{T,h(l)}\right\| = O(h^{1/2}) \to \mathbb{E}\left\|X_T - \widehat{X}_{T,h(l)}\right\|^2 = O(h)$$

by strong convergence. We will assume f is Lipschitz, which means that there exists $C > 0$ such that

$$\|f(\mathbf{x}) - f(\mathbf{y})\| \leq C \|\mathbf{x} - \mathbf{y}\|,$$

for all $\mathbf{x}$ and $\mathbf{y}$ in the domain of f. This is a very strong assumption, which is relaxed in the general theorem. However, it makes the calculations relatively straightforward. Now, observe that

$$\begin{aligned}\mathrm{Var}\left(\widehat{P}_l - P\right) &= \mathbb{E}\left(\widehat{P}_l - P\right)^2 - \left(\mathbb{E}\widehat{P}_l - P\right)^2 \leq \mathbb{E}\left(\widehat{P}_l - P\right)^2 \\ &= \mathbb{E}\left\|f(\widehat{X}_{T,h(l)}) - f(X_T)\right\|^2 \leq C\left\|\widehat{X}_{T,h(l)} - X_T\right\|^2 = O(h_l).\end{aligned}$$

We can obtain an upper bound on the variance of $\widehat{P}_l - \widehat{P}_{l-1}$ by writing

$$\widehat{P}_l - \widehat{P}_{l-1} = \left(\widehat{P}_l - P\right) + \left(P - \widehat{P}_{l-1}\right)$$

and observing that

$$\mathrm{Var}\left(\widehat{P}_l - \widehat{P}_{l-1}\right) \leq \left(\sqrt{\mathrm{Var}\left(\widehat{P}_l - P\right)} + \sqrt{\mathrm{Var}\left(P - \widehat{P}_{l-1}\right)}\right)^2 = O(h_l)$$

Now, choosing $N_l = O(\epsilon^{-2} L h_l)$ we get

$$\mathrm{Var}(Y(L)) = \sum_{l=0}^{L} \frac{V_l}{N_l} = \sum_{l=1}^{L} \frac{O(h_l)}{O(\epsilon^{-2} L h_l)} = O(\epsilon^{-2}).$$

If we then choose

$$L = \frac{\log \epsilon^{-1}}{\log M} + O(1)$$

as $\epsilon \to 0$, then $h_L = O(\epsilon)$. This implies that we have $\mathsf{MSE} = O(\epsilon^2)$ with

$$\mathsf{work} \propto \sum_{l=0}^{L} \frac{N_l}{h_l} = \sum_{l=0}^{L} \frac{O(\epsilon^{-2} L h_l)}{h_l} = O(\epsilon^{-2} L^2) = O(\epsilon^{-2} (\log \epsilon)^2),$$

which is better than standard Monte Carlo.

3.7.4 The Key Theorem

The following theorem, first established in [1], gives quite general results under which the multilevel approach works well. For more details, extensions, and other applications, take a look at Mike Giles' website (go to `http://people.maths.ox.ac.uk/~gilesm/`).

Theorem 3.7.1. Let P denote a functional of the solution of

$$\mathrm{d}X_t = a(X_t, t)\,\mathrm{d}t + b(X_t, t)\,\mathrm{d}W_t,$$

for a given Brownian path $\{W_t\}_{t \geq 0}$ and let $\widehat{P}_l$ denote the corresponding approximation using a numerical discretization with timestep $h(l) = M^{-l} T$. If there exist independent estimators $\widehat{Y}_l$ based on N_l Monte Carlo samples and positive constants $\alpha \geq 1/2, \beta, c_1, c_2, c_3$ such that

(i) $\mathbb{E}[\widehat{P}_l - P] \leq c_1 h_l^{\alpha}$,

(ii) $\mathbb{E}\widehat{Y}_l = \begin{cases} \mathbb{E}\widehat{P}_0, & l = 0, \\ \mathbb{E}[\widehat{P}_l - \widehat{P}_{l-1}], & l > 0, \end{cases}$

(iii) $\mathrm{Var}(\widehat{Y}_l) \leq c_2 N_l^{-1} h_l^{\beta}$,

(iv) C_l, the computational complexity of $\widehat{Y}_l$ is bounded by

$$C_l \leq c_3 N_l h_l^{-1},$$

then there exists a positive constant c_4 such that for any $\epsilon > e^{-1}$, there are values L and N_l for which the multilevel estimator

$$\widehat{Y} = \sum_{l=0}^{L} \widehat{Y}_l$$

has an MSE with bound

$$\mathsf{MSE} < \epsilon^2,$$

with computational complexity C with bound

$$C \leq \begin{cases} c_4\epsilon^{-2}, & \beta > 1, \\ c_4\epsilon^{-2}(\log\epsilon)^2, & \beta = 1, \\ c_4\epsilon^{-2-(1-\beta)/\alpha}, & 0 < \beta < 1. \end{cases}$$

Proof. See [1] □

3.7.5 Implementation

Listing 3.15: Matlab code

```
mu = 0.1; sigma = 0.05;
X_0 = 1; L = 6; Y = zeros(L,1);

N_0 = 10^(L+1); N = zeros(L,1);

for l = 1:L
    N(l) = 10^(L+1-l);
end

m = 1; h = 1;
vals = zeros(N_0,1);

for i = 1:N_0
    X = X_0;
    for k = 1:m
        X = X + mu*X*h + sigma*X*sqrt(h)*randn;
    end
    vals(i) = X;
end

Y_0 = mean(vals);

```

```
for l = 1:L
    m_old = 2^(l-1); h_old = 1/m_old;
    m_new = 2^l; h_new = 1/m_new;
    vals = zeros(N(l),1);

    for i = 1:N(l)
        X_old = X_0; X_new = X_0;
        for k = 1:m_old
            Z_1 = sqrt(h_new)*randn; Z_2 = sqrt(h_new)*randn;
            Z = Z_1 + Z_2;
            X_old = X_old + mu*X_old*h_old + sigma*X_old*Z;
            X_new = X_new + mu*X_new*h_new + sigma*X_new*Z_1;
            X_new = X_new + mu*X_new*h_new + sigma*X_new*Z_2;
        end
        vals(i) = X_new - X_old;
    end

    Y(l) = mean(vals);
end

Y_0 + sum(Y)
```

Chapter 4

Spatial Processes

4.1 Random Fields

At the beginning of these notes we defined a stochastic process as a set of random variables $\{X_i\}_{i\in I}$ taking values in a state space $\mathcal{X}$ with index set $I \subset \mathbb{R}$. It seems natural to ask what would happen if we took I to be something "larger" than a subset of $\mathbb{R}$ — for example, a subset of $\mathbb{R}^N$. Such generalizations are called *random fields*. Intuitively, we can think of them as stochastic processes evolving over space instead of time. Sometimes, they can be thought of as spatial processes that are also evolving over time (for example, if $I = [a, b]^3 \times [0, T]$).

Definition 4.1.1 (Random Field)**.** Let $I \subset \mathbb{R}^N$ and $\{\mathbf{X}_i\}_{i\in I}$ be a set of d-dimensional random vectors indexed by I. We say $\{X_i\}_{i\in I}$ is a (N, d) random field.

Note that stochastic processes are random fields.

4.1.1 Gaussian Random Fields

Some of the simplest random fields to work with are *Gaussian random fields*. These are a natural extension of our definition of Gaussian processes.

Definition 4.1.2. A random field $\{X_i\}_{i\in I}$ is a Gaussian random field if $(X_{i_1}, \ldots, X_{i_n})$ has a multivariate normal distribution for all $1 \leq n < \infty$ and $(i_1, \ldots, i_n) \in I^n$.

As with Gaussian processes, we can describe the finite dimensional distributions of Gaussian random fields using a mean function $\mu(i) = \mathbb{E}X_i$ and a covariance function $r(i, j) = \mathrm{Cov}(X_i, X_j)$. If we wish to consider the Gaussian random field at a finite number of points, we can treat its values as a

random vector. We can use the mean function and covariance function to construct the mean vector, $\boldsymbol{\mu}$, and covariance matrix, Σ, of these points.

Using these functions, and treating a finite number of points of the random field as a vector, we can generate a realization of the field

Example 4.1.3 (Gaussian White Noise)**.** A trivial example of a Gaussian random field is *Gaussian white noise*. On $I = \{1, \ldots, m\} \times \{1, \ldots, m\}$, this is the process, $\{X_i\}_{i \in I}$ with mean function

$$\mu(i) = \mathbb{E}X_i = 0,$$

and covariance function

$$r(i,j) = r((x^i, y^i), (x^j, y^j)) = \mathrm{Cov}(X_{(x^i,y^i)}, X_{(x^j,y^j)}) = \begin{cases} 1 & \text{if } (x^i, y^i) = (x^j, y^j) \\ 0 & \text{otherwise} \end{cases}.$$

We can simulate this in Matlab by simple generating a matrix of standard

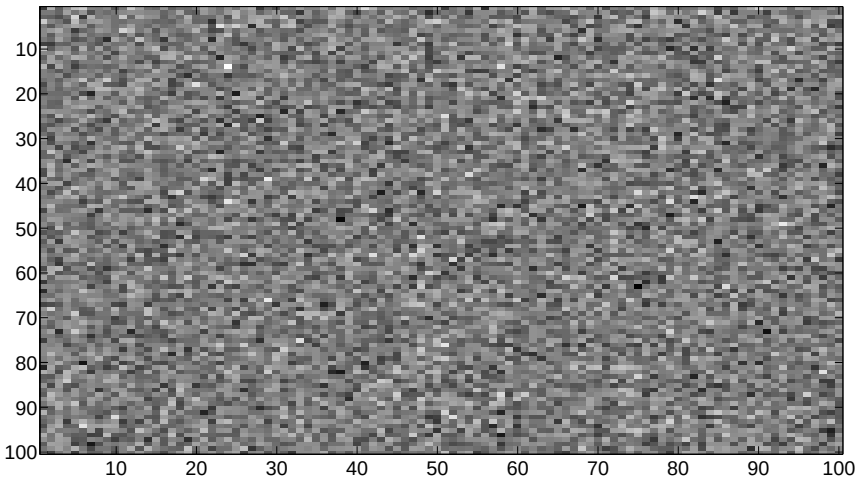

Figure 4.1.1: A realization of Gaussian white noise

normal random variables.

Listing 4.1: Matlab code

```
m = 100;
Z = randn(m,m);
imagesc(Z);
colormap(gray);
```

Example 4.1.4 (Brownian Sheet)**.** The *Brownian sheet*, $\{W_{(x,y)}\}_{(x,y)\in[0,1]^2}$ is a natural random field extension of Brownian motion on $[0,1]$. It has mean function

$$\mu(i) = \mathbb{E}X_i = 0,$$

and covariance function

$$r(i,j) = r((x^i,y^i),(x^j,y^j)) = \mathrm{Cov}(W_{(x^i,y^i)}, W_{(x^j,y^j)}) = \min(x^i,x^j)\min(y^i,y^j).$$

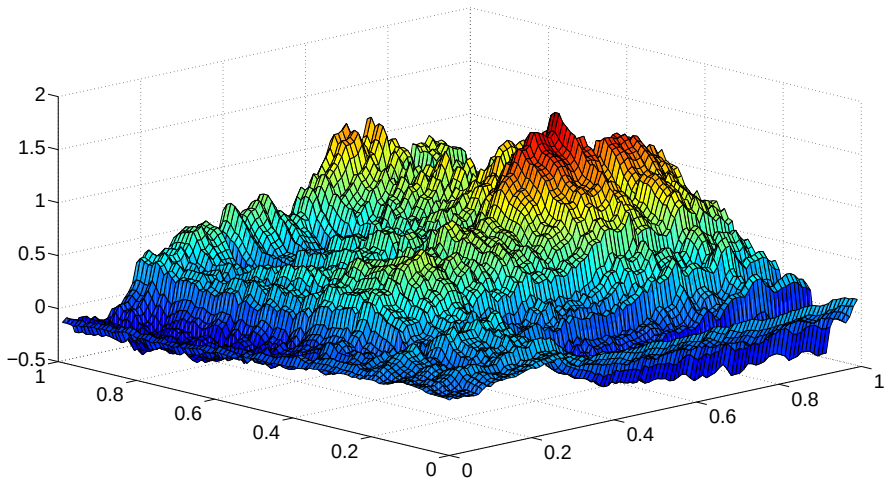

Figure 4.1.2: A realization of a Brownian sheet

We can easily simulate this in Matlab.

Listing 4.2: Matlab code

```
alpha = 1.5; x = .01:.01:1; y = .01:.01:1;

n1 = length(x); n2 = length(y);

Sigma = zeros(n1*n2,n1*n2);

for i = 1:n2
    for j = 1:n1
        row_state = (i-1)*n2 + j;
        for k = 1:n2
            for l = 1:n1
                column_state = (k-1)*n2 + l;
                Sigma(row_state,column_state) = ...
```

```
                    min(x(j),x(l))*min(y(i),y(k));
            end
        end
    end
end

A = chol(Sigma);
Y = randn(1,n1*n2) * A;

X = zeros(n1,n2);

for i = 1:n2
    for j = 1:n1
        X(i,j) = Y((i-1)*n2 + j);
    end
end

surf(x,y,X)
```

Random fields are, in general, difficult to work with. For that reason, it is nice to identify classes of random fields that are more tractable.

Definition 4.1.5 (Stationarity)**.** We say a random field, $\{X_i\}_{i\in I}$, is stationary if $(X_{i_1}, \ldots, X_{i_n})$ has the same distribtion as $(X_{i_1+s}, \ldots, X_{i_n+s})$ for all $n \geq 1$, $(i_1, \ldots, i_n) \in I^n$ and $s \in I$.

Weak stationary implies that the first two moments of the distributions do not change.

Definition 4.1.6 (Wide sense (weak) stationarity)**.** A random field, $\{X_i\}_{i\in I}$ is wide-sense stationary if $\mu(i) = c$ for all $i \in I$ and $r(i, j) = r(|i - j|)$. That is, the covariance function is simply a function of the displacement vector between the points i and j.

Analogously to the case of Gaussian processes, Gaussian fields are stationary if they are weak stationary. When considering random fields, a different type of invariance property is also useful. This is called *isotropy*. Basically, this means the distribution of the object does not change when the object is rotated. We will only consider a very special case of isotropy.

Definition 4.1.7 (Stationary wide sense isotropy)**.** A stationary random field is wide sense isotropic if $r(i, j) = r(\|i - j\|)$. That is, the covariance between points i and j only depends on the distance between them.

Note that non-stationary random fields can be isotropic.

4.1.2 Markov Random Fields

When considering stochastic processes, the Markov property simplified things considerably. Markov random fields are the random field analogues of Markov stochastic processes. To define a meaningful version of the Markov property, we exploit the idea of conditional independence.

Definition 4.1.8 (Conditional Independence for Events)**.** We say two events A and B are said to be *conditionally independent* given C if

$$\mathbb{P}(A \cap B \,|\, C) = \mathbb{P}(A \,|\, C)\mathbb{P}(B \,|\, C).$$

Clearly, a Markov process is conditional independent of its past given its present.

Definition 4.1.9 (Conditional Independence for Random Variables)**.** We can easily extend this definition to random vectors $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ with joint density $\pi(\mathbf{x}, \mathbf{y}, \mathbf{z})$. We say $\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent given $\mathbf{Z}$ (denoted $\mathbf{x} \perp \mathbf{y} \,|\, \mathbf{z}$) if

$$\pi(\mathbf{x}, \mathbf{y} \,|\, \mathbf{z}) = \pi(\mathbf{x} \,|\, \mathbf{z})\pi(\mathbf{y} \,|\, \mathbf{z}).$$

There is a nice criterion for determining conditional independence given a joint density.

Theorem 4.1.10 (Factorization Theorem)**.** Given random vectors $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ with joint density $\pi(\mathbf{x}, \mathbf{y}, \mathbf{z})$, we say $\mathbf{x} \perp \mathbf{y} \,|\, \mathbf{z}$ if and only if

$$\pi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = f(\mathbf{x}, \mathbf{z})g(\mathbf{y}, \mathbf{z})$$

for some functions f and g and all $\mathbf{z}$ with $\pi(\mathbf{z}) > 0$.

Example 4.1.11. Take

$$\pi(x, y, z) \propto \exp\{x + xz + yz\}$$

on some bounded region. We can write

$$\pi(x, y, z) \propto \exp\{x + xz\}\exp\{yz\} = f(x, z)g(y, z)$$

so $x \perp y \,|\, z$.

Example 4.1.12. Take

$$\pi(x, y, z) \propto \exp\{xyz\}$$

on some bounded region. We cannot factorize this into a function of x and z and a function of y and z, so x and y are not conditionally independent given z here.

We can represent conditional dependence relations using graphs called *graphical models*.

Example 4.1.13.

PICTURE HERE

This example encodes the dependency structure

$$\pi(a, b, c, d) = \pi(a)\pi(b)\pi(c \,|\, b, d)\pi(d \,|\, a, b, c).$$

Markov random fields depict a specific dependency structure using undirected graphs $G = (V, E)$. Alternatively, you can think of these as graphs with arrows on both ends of the edges.

PICTURE HERE

In Markov random fields, random variables are conditionally independent of the rest of the graph given their immediate neighbors.

Definition 4.1.14 (Neighbors of a Vertex)**.** The neighbors of the vertex i are the members of the set

$$\mathcal{N}_i = \{j \in V : (i, j) \in E\}.$$

For example, in the graph above, $\mathcal{N}_1 = \{2, 3\}$, $\mathcal{N}_2 = \{1, 3, 4\}$, $\mathcal{N}_3 = \{1, 2, 4\}$ and $\mathcal{N}_4 = \{2, 3\}$.

We can now define a Markov random field. It will be convenient to use to following notation. For $C \subset V$, let $\mathbf{X}_C = \{\mathbf{X}_i : i \in C\}$ and $\mathbf{X}_{-C} = \{\mathbf{X}_i : i \in V \setminus C\}$.

Definition 4.1.15 (Markov Random Field)**.** We say a sequence of random variables, $\mathbf{X} = \{X_i\}_{i \in V}$, indexed by the vertices of $G = (V, E)$, is a Markov random field if $X_i \,|\, \mathbf{X}_{-\{i\}}$ has the same distribution as $X_i \,|\, \mathbf{X}_{\mathcal{N}_i}$ for all $i \in V$.

4.1.3 Gaussian Random Markov Fields

We will consider a special class of Markov random fields, called *Gaussian random Markov fields*.

Definition 4.1.16 (Gaussian Random Markov Field)**.** A Gaussian random Markov field (GRMF) is a Markov random field which is also a Gaussian field.

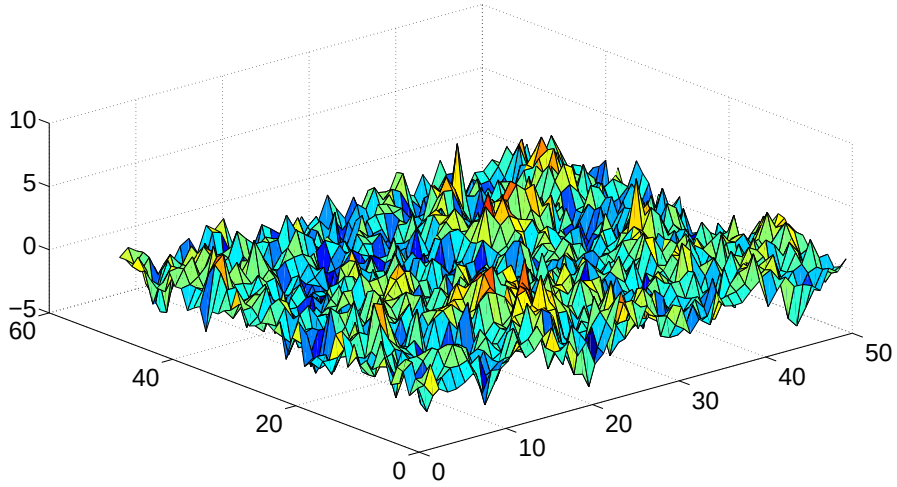

Figure 4.1.3: A realization of a Gaussian Markov random field on a 50×50 lattice

We will assume that the GRMFs we consider have densities. That is, the covariance matrix of a finite number of points, Σ, will always be positive definite. Recall that the pdf of a normal vector $\mathbf{X}$ is given by

$$f(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = (2\pi)^{-n/2} |\Sigma|^{-1/2} \exp\left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\intercal \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}.$$

In general, the covariance matrix is a natural object to consider when talking about Gaussian processes and fields. However, in the case of Gaussian random fields, it turns out that the *precision matrix* is a more natural object to work with.

Definition 4.1.17 (Precision Matrix)**.** The precision matrix, Q, of a covariance matrix, Σ, is its inverse. That is,

$$Q = \Sigma^{-1}.$$

We can rewrite the pdf of a normal vector in terms of its precision matrix as

$$f(\mathbf{x}; \boldsymbol{\mu}, Q) = (2\pi)^{-n/2} |Q|^{1/2} \exp\left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\intercal Q (\mathbf{x} - \boldsymbol{\mu}) \right\}.$$

The following result reveals the reason why precision matrices are so appealing when working with GRMFs.

Theorem 4.1.18. Let $\mathbf{X}$ be normally distributed with mean $\boldsymbol{\mu}$ and precision matrix Q. Then, for $i \neq j$,

$$X_i \perp X_j \,|\, \mathbf{X}_{-\{i,j\}} \iff Q_{i,j} = 0.$$

Proof. Assume without loss of generality that $\boldsymbol{\mu} = \mathbf{0}$. We have

$$\begin{aligned}\pi(\mathbf{x}) \propto \exp\left\{-\frac{1}{2}\mathbf{x}^\intercal Q \mathbf{x}\right\} &= \exp\left\{-\frac{1}{2}\sum_{l\in V}\sum_{k\in V} x_k Q_{k,l} x_l\right\} \\ &= \exp\left\{-\frac{1}{2}x_i x_j (Q_{i,j} + Q_{j,i}) - \frac{1}{2}\sum_{\{k,l\}\neq\{i,j\}} x_k Q_{k,l} x_l\right\}\end{aligned}$$

If you consider examples 4.1.12 and 4.1.13, then it is clear that this pdf can only be factorized in the necessary way if $Q_{i,j} = Q_{j,i} = 0$. □

Theorem 4.1.19. Let $\mathbf{X}$ be a GRMF with respect to $G = (V, E)$ with mean $\boldsymbol{\mu}$ and precision matrix Q. Then,

(i) $\mathbb{E}\left[X_i \,|\, \mathbf{X}_{-i}\right] = \mu_i - \frac{1}{Q_{i,i}} \sum_{j\in\mathcal{N}_i} Q_{i,j}(X_j - \mu_j)$.

(ii) $\mathrm{Prec}(X_i \,|\, \mathbf{X}_{-\{i\}}) = Q_i i$.

(iii) $\mathrm{Corr}(X_i, X_j \,|\, \mathbf{X}_{i,j}) = \frac{-Q_{i,j}}{\sqrt{Q_{i,i}Q_{j,j}}}$.

Theorem 4.1.20. Let $\mathbf{X}$ be a GRMF with respect to $G = (V, E)$. Then, the following are equivalent:

(i) The *pairwise Markov property*

$$X_i \perp X_j \,|\, \mathbf{X}_{-\{i,j\}} \quad (i,j) \notin E, i \neq j.$$

(ii) The *local Markov property*

$$X_i \perp \mathbf{X}_{-\{i\}\cup\mathcal{N}_i} \,|\, \mathbf{X}_{\mathcal{N}_i} \quad \forall i \in V.$$

(iii) The *global Markov property*

$$\mathbf{X}_A \perp \mathbf{X}_B \,|\, \mathbf{X}_C$$

for all disjoint A, B and C, where C separates A and B.

GRMFs are very appealing from a computational perspective as the precision matrix Q is sparse (that is, it is mostly zeros). In general, matrix operations are much faster for sparse matrices. In addition, far less memory is required in order to store sparse matrices.

Remember that the inverse of a SPD matrix is also SPD. This implies that Q has a Cholesky decomposition. We can use this to generate normal random vectors with the desired distributions.

Theorem 4.1.21. Given a SPD covariance matrix Σ and a vector $\boldsymbol{\mu}$, the random vector

$$\mathbf{X} = \boldsymbol{\mu} + (C^\intercal)^{-1}\,\mathbf{Z} \sim \mathsf{N}(\boldsymbol{\mu}, \Sigma),$$

where C is such that $\Sigma^{-1} = Q = CC^\intercal$ and $\mathbf{Z} \sim \mathsf{N}(\mathbf{0}, I)$.

Proof. We know $\mathbf{X}$ is multivariate normal, so we just need to check that it has the correct mean and variance. Now,

$$\mathbb{E}\left[\boldsymbol{\mu} + (C^\intercal)^{-1}\,\mathbf{Z}\right] = \boldsymbol{\mu} + (C^\intercal)^{-1}\,\mathbb{E}\mathbf{Z} = \boldsymbol{\mu},$$

and

$$\mathrm{Var}\left(\boldsymbol{\mu} + (C^\intercal)^{-1}\,\mathbf{Z}\right) = (C^\intercal)^{-1}\,\mathrm{Var}(\mathbf{Z})\,(C)^{-1} = (CC^\intercal)^{-1} = Q^{-1} = \Sigma.$$

□

Example 4.1.22 (A Gaussian random Markov field on a Lattice)**.** We simulate a zero mean GRMF on the 200×200 square lattice with $Q_{i,i} = 1$ and

$$Q_{i,j} = \begin{cases} -0.25 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}.$$

Listing 4.3: Matlab code

```
m = 200; d1 = 1; d2 = -0.25;
nels = m*(5*m-4);

a = zeros(1, nels); b = zeros(1,nels); q = zeros(1,nels);
%compute the links and weights for the precision matrix
k=0;
for i=1:m
    for j=1:m
        A = findneigh(i,j,m);
        number_neighbours = size(A,1);
        for h=1:number_neighbours
            a(k+h)= ij2k(i,j,m);
```

```
            b(k+h)= ij2k(A(h,1),A(h,2),m);
            if h==1
                q(k+h) = d1;
            else
                q(k+h) = d2;
            end
        end
        k = k+number_neighbours;
    end
end
%construct the precision matrix
Q = sparse(a,b,q,m^2,m^2);
%calculate the Cholesky matrix
C = chol(Q,'lower');
Z = randn(m^2,1);
% generate the Gaussian process
x = C\Z;
colormap gray, brighten(-0.2)
imagesc(reshape(x,m,m)) % plot the result
```

This uses the following functions.

Listing 4.4: Matlab code

```
function A = findneigh(i,j,m)
% find neighbors of the (i,j)-th site of an m by m square lattice
if i==1
    if j==1
        A = [1,1;1,2;2,1];
    elseif j==m
        A = [1,m;1,m-1;2,m];
    else
        A = [1,j;1,j-1;1,j+1;2,j];
    end
elseif i==m
    if j==1
        A = [m,1;m,2;m-1,1];
    elseif j==m
        A = [m,m;m,m-1;m-1,m];
    else
        A = [m,j;m,j-1;m,j+1;m-1,j];
    end
else
    if j==1
        A = [i,1;i,2;i-1,1;i+1,1];
```

```
    elseif j==m
        A = [i,m;i,m-1;i+1,m;i-1,m];
    else
        A = [i,j;i,j-1;i,j+1;i+1,j;i-1,j];
    end
end
end
```

Listing 4.5: Matlab code

```
function k = ij2k(i,j,m)
k = (i-1)*m + j;
```

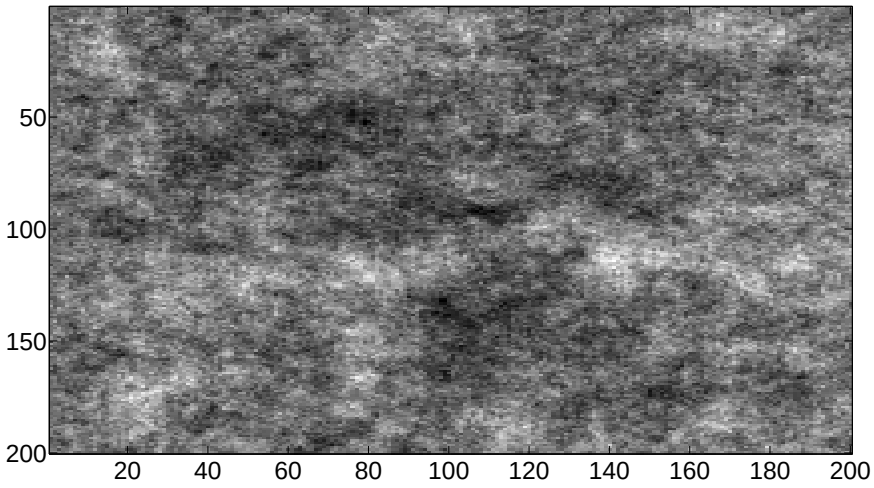

Figure 4.1.4: A realization of the random field

4.2 Spatial Poisson Processes

We want to extend the notion of a Poisson process, which we have already discussed in 1 dimension, to higher dimensions. Remember, we defined this process in three equivalent ways: the arrival times, the inter-arrival times, and the counting process $\{N_t\}_{t\geq 0}$. It no longer makes sense to think about inter-arrival times, but if we think of the arrival times as positions in space (on the 1D line) then it is clear that we can possibly extend this definition to space. As it turns out, however, the counting function definition is in some sense the nicest method of extending the Poisson process to space.

Think of the function $N([a,b]) = N_b - N_a$. This measures the number of points in the bounded set $[a,b]$. Such a measure is called a *counting measure*. If we know the locations of the points, it is easy to construct this function. Conversely, if we know the function we can find the location of the points.

In the following, we will use ν_d to denote d-dimensional Lebesgue measure.

4.2.1 Binomial Process

Consider a bounded window $W \subset \mathbb{R}^d$ with a single point placed uniformly in it. The location of this point, $\mathbf{x}$, has the probability density

$$f(\mathbf{x}) = \begin{cases} \frac{1}{\nu_d(W)} & \text{if } \mathbf{x} \in W \\ 0 & \text{otherwise} \end{cases}.$$

If we consider a bounded set $B \subset \mathbb{R}^d$, then, for $\mathbf{X}$, the point we randomly threw into W, we have

$$\mathbb{P}(\mathbf{X} \in B) = \int_B f(\mathbf{x})\,\mathrm{d}\mathbf{x} = \frac{\nu_d(B \cap W)}{\nu_d(W)}.$$

If we throw n points into W, then we have a *binomial process*.

Definition 4.2.1. A *binomial point process of n points* is a process of n independent points, $\mathbf{X}_1, \ldots, \mathbf{X}_n$, uniformly distributed in the bounded set W.

If we consider $N(B)$, the number of points in some bounded set B, then this can be written as

$$N(B) = \sum_{i=1}^{n} \mathbb{I}(\mathbf{X}_i \in B).$$

It is straightforward to see that $N(B)$ has a binomial distribution. Specifically, $N(B) \sim \mathsf{Bin}(n,p)$, where $p = \mathbb{P}(\mathbf{X} \in B)$, defined above.

It is straightforward to generate a binomial process. The only difficulty lies in generating points uniformly in the set W. This is primarily difficult in high dimensions but, in practice, may not be such an issue as the dimension of a point process is usually quite low.

Here is an example in two dimensions.

Listing 4.6: Matlab code

```
n = 20; dims = [0 10; 0 10];
```

```
X = zeros(n,2);

X(:,1) = (dims(1,2) - dims(1,1)) * rand(n,1);
X(:,2) = (dims(2,2) - dims(2,1)) * rand(n,1);

S = 200 * ones(n,1);
scatter(X(:,1),X(:,2),S,'fill')
```

Here is an example in three dimensions.

Listing 4.7: Matlab code

```
n = 20; dims = [0 10; 0 10; 0 10];

X = zeros(n,3);

X(:,1) = (dims(1,2) - dims(1,1)) * rand(n,1);
X(:,2) = (dims(2,2) - dims(2,1)) * rand(n,1);
X(:,3) = (dims(3,2) - dims(3,1)) * rand(n,1);

S = 200 * ones(n,1);
scatter3(X(:,1),X(:,2),X(:,3),S,'fill')
```

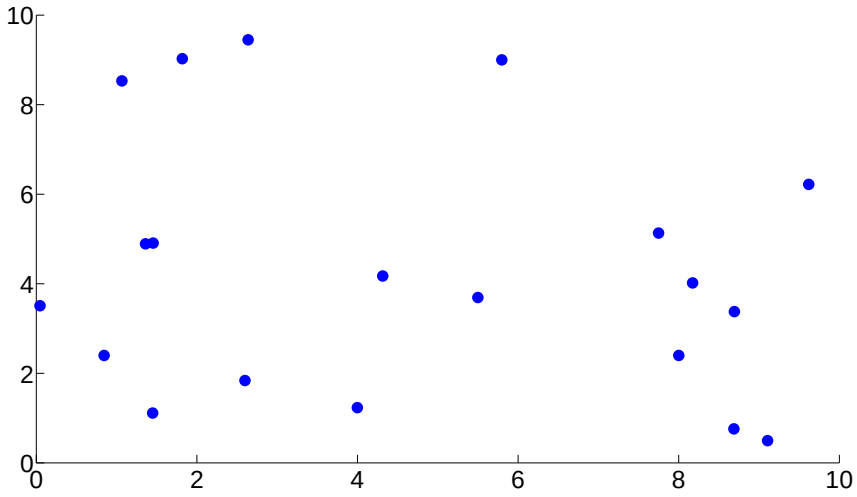

Figure 4.2.1: A realization of a binomial process on $[0, 10] \times [0, 10]$ with $n = 20$ points.

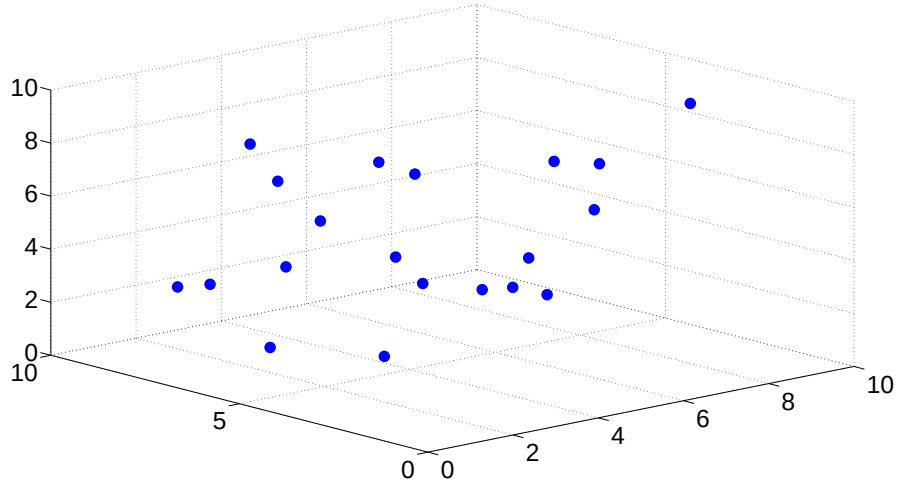

Figure 4.2.2: A realization of a binomial process on $[0, 10] \times [0, 10] \times [0, 10]$ with $n = 20$ points.

4.2.2 Spatial Point Processes

As stated above, we define spatial point processes in terms of counting measures, specifically *random counting measures*.

Definition 4.2.2 (Locally Finite Counting Measure)**.** Let $\mathbb{N}$ be the family of *locally finite counting measures* $\varphi : \mathcal{B}(\mathbb{R}^d) \to \{0, 1, \ldots\} \cup \{\infty\}$. That is,

$$\varphi\left(\bigcup_{n=1}^{\infty} B_n\right) = \sum_{n=1}^{\infty} \varphi(B_n)$$

for pairwise disjoint $B_1, B_2, \ldots \in \mathcal{B}(\mathbb{R}^d)$, and $\varphi(B) < \infty$ for all bounded $B \in \mathcal{B}(\mathbb{R}^d)$.

Let $\mathcal{N}$ be the smallest σ-algebra on $\mathbb{N}$ such that $\varphi \to \varphi(B)$ is $(\mathcal{N}, \mathcal{B}(\mathbb{R}^d))$-measurable for all bounded $B \in \mathcal{B}(\mathbb{R}^d)$.

Definition 4.2.3 (Random Counting Measure)**.** A *random counting measure* $N : \Omega \to \mathbb{N}$ is a measurable mapping of the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ into $(\mathbb{N}, \mathcal{N})$. Intuitively, it is a random element of $\mathbb{N}$.

Poisson Process

Definition 4.2.4 (Poisson Random Measure)**.** Let $\mathcal{B}_0(\mathbb{R}^d)$ be the collection of bounded Borel sets in $\mathbb{R}^d$ and $\mu : \mathcal{B}(\mathbb{R}^d) \to [0, \infty]$ a locally finite measure

(i.e., $\mu(B) < \infty$ for all $B \in \mathcal{B}_0(\mathbb{R}^d)$). We say $\{N_B\}_{B\in\mathcal{B}(\mathbb{R}^d)}$ is a *Poisson random measure* with intensity measure μ if

(i) $N_{B_1}, N_{B_2}, \ldots$ are independent random variables for pairwise disjoint $B_1, B_2 \ldots \in \mathcal{B}_0(\mathbb{R}^d)$.

(ii) $N_B \sim \mathsf{Poi}(\mu(B))$ for all $B \in \mathcal{B}_0(\mathbb{R}^d)$.

If μ is proportion to ν_d (i.e., $\mu(B) = \lambda\nu_d(B)$ for all $B \in \mathcal{B}(\mathbb{R}^d)$), then $\{N_B\}_{B\in\mathcal{B}(\mathbb{R}^d)}$ is said to be a homogenous Poisson counting measure with intensity λ.

Lemma 4.2.5. Consider a Poisson point process in $\mathbb{R}^d$ with constant intensity $\lambda > 0$. Let $W \subset \mathbb{R}^d$ be a region with $0 < \nu_d(W) < \infty$. Given $N(W) = n$, the conditional distribution of $N(B)$ for $B \subseteq W$ is binomial, with

$$\mathbb{P}(N(B) = k \,|\, N(W) = n) = \binom{n}{k} p^k (1-p)^{n-k},$$

where $p = \nu_d(B)/\nu_d(W)$. Furthermore, the conditional joint distribution of $N(B_1), \ldots, N(B_n)$ for any $B_1, \ldots, B_n \subset W$ is the same as the joint distribution of these random variables for a binomial process.

Proof. Let $0 \le k \le n$. Then

$$\begin{aligned}
\mathbb{P}(N(B) = k \,|\, N(W) = n) &= \frac{\mathbb{P}(N(B) = k, N(W) = n)}{\mathbb{P}(N(W) = n)} \\
&= \frac{\mathbb{P}(N(B) = k, N(W \setminus B) = n - k)}{\mathbb{P}(N(W) = n)} \\
&= \frac{\mathbb{P}(N(B) = k)\mathbb{P}(N(W \setminus B) = n - k)}{\mathbb{P}(N(W) = n)} \\
&= \frac{e^{-\lambda\nu_d(B)} \frac{(\lambda\nu_d(B))^k}{k!} e^{-\lambda\nu_d(W\setminus B)} \frac{(\lambda\nu_d(W\setminus B))^{n-k}}{(n-k)!}}{e^{-\lambda\nu_d(W)} \frac{(\lambda\nu_d(W))^n}{n!}} \\
&= \binom{n}{k} \left(\frac{\nu_d(B)}{\nu_d(B)}\right)^k \left(1 - \frac{\nu_d(B)}{\nu_d(B)}\right)^{n-k}.
\end{aligned}$$

□

Lemma 4.2.5 suggests that we can simulate a homogenous Poisson process on W be generating $N(W) \sim \mathsf{Poi}(\lambda\nu_d(W))$, then simulating a binomial process with $N(W)$ points.

Example 4.2.6. Consider a homogenous Poisson process on $W = \{(x, y) : x^2 + y^2 \leq 1\}$ with $\lambda = 100$. We can simulate this as follows.

Listing 4.8: Matlab code

```
lambda = 100; N = poissrnd(lambda * pi);
X = zeros(N,2); count = 1;

while count <= N
    x = 2*rand - 1; y = 2 * rand - 1;
    if (x^2 + y^2) <= 1
        X(count,1) = x;
        X(count,2) = y;
        count = count + 1;
    end
end

S = 200 * ones(N,1);
scatter(X(:,1),X(:,2),S,'fill')
```

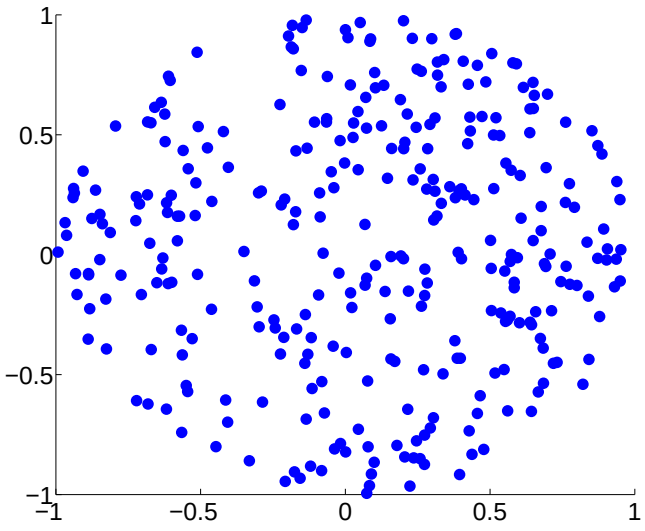

Figure 4.2.3: A realization of a Poisson process with constant intensity $\lambda = 100$ on $W = \{(x, y) : x^2 + y^2 \leq 1\}$.

Inhomogeneous Poisson Processes

Inhomogeneous Poisson processes (i.e. processes where it is not true that $\mu(B) = \lambda\nu_d(B)$ for all $B \in \mathcal{B}(\mathbb{R}^d)$) are at little more challenging to simulate

and work with than homogeneous Poisson processes. We will restrict our attention to processes with measures that are absolutely continuous with respect to d-dimensional Lebesgue measure. For such processes, there exists a Borel measurable function $\lambda : \mathbb{R}^d \to [0, \infty)$ such that

$$\mu(B) = \int_B \lambda(\mathbf{x})\, \mathrm{d}\mathbf{x}$$

for all $B \in \mathcal{B}(\mathbb{R}^d)$. This function is called the intensity function of the process $\{N_B\}_{B \in \mathcal{B}(\mathbb{R}^d)}$.

Theorem 4.2.7. Let $\lambda_1, \lambda_2 : \mathbb{R}^d \to [0, \infty)$ be two Borel measurable and locally integrable functions such that

$$\lambda_1(x) > \lambda_2(x)$$

for all $x \in \mathbb{R}^d$. Let $\{S_n\}$ be a measurable indexing of the atoms of a Poisson process with intensity function λ_1 and let $\{U_n\}$ be a sequence of i.i.d. uniform random variables. Then, the random counting measure $\{\widetilde{N}_B\}_{B \in \mathcal{B}(\mathbb{R}^d)}$ with

$$\widetilde{N}_B = \#\{n : S_n \in B, U_n \leq \lambda_2(S_n)/\lambda_1(S_n)\}$$

for all $B \in \mathcal{B}(\mathbb{R}^d)$, is a Poisson process with intensity function λ_2.

Example 4.2.8 (Inhomogeneous Poisson Process)**.** Consider an inhomogeneous Poisson process with intensity $\mu(x, y) = 20\left(\sin^2(x) + \cos^2(y)\right)$ on $W = [0, 5] \times [0, 5]$. We have $\max_{(x,y)\in W} 20\left(\sin^2(x) + \cos^2(y)\right) = 40$, so $\lambda^* = 40$.

Listing 4.9: Matlab code

```
lambda_star = 40;
N_star = poissrnd(lambda_star * (5 * 5));
X = [];

for i = 1:N_star
    x = 5 * rand; y = 5 * rand;
    if  rand < (20*(sin(x)^2 + cos(y)^2) / lambda_star)
        X = [X; x y];
    end
end

x = 0 : .1 : 5; y = 0 : .1 : 5;
n = length(x); Z = zeros(n,n);

```

```
for i = 1:n
    for j = 1:n
        Z(j,i) = 20*(sin(x(i))^2 + cos(y(j))^2);
    end
end

surf(x,y,Z);
hold on
colormap hot
S = 100 * ones(length(X),1);
scatter(X(:,1),X(:,2),S,'fill')
```

Cox Process

A Cox process is a natural extension of a Poisson process where the deterministic instensity measure is replaced by a random measure.

Definition 4.2.9 (Cox Process)**.** Let $\{\Lambda_B\}_{B\in\mathcal{B}(\mathbb{R}^d)}$ be a random measure that is locally finite with probability 1. The family of random counting measures $\{N_B\}_{B\in\mathcal{B}(\mathbb{R}^d)}$ is called a Cox process with random intensity Λ if

$$\mathbb{P}\left(\bigcap_{i=1}^{n}\{N_{B_i}=k_i\}\right)=\mathbb{E}\left[\prod_{i=1}^{n}\frac{\Lambda_{B_i}^{k_i}}{k_i!}\exp\{-\Lambda_{B_i}\}\right].$$

Example 4.2.10 (Cox Process)**.** Consider a Cox process with

Listing 4.10: Matlab code

```
x = .01:.01:1; y = .01:.01:1;
n1 = length(x); n2 = length(y);
Sigma = zeros(n1*n2,n1*n2);

for i = 1:n2
    for j = 1:n1
        row_state = (i-1)*n2 + j;
        for k = 1:n2
            for l = 1:n1
                column_state = (k-1)*n2 + l;
                Sigma(row_state,column_state) = ...
                    min(x(j),x(l))*min(y(i),y(k));
            end
        end
```

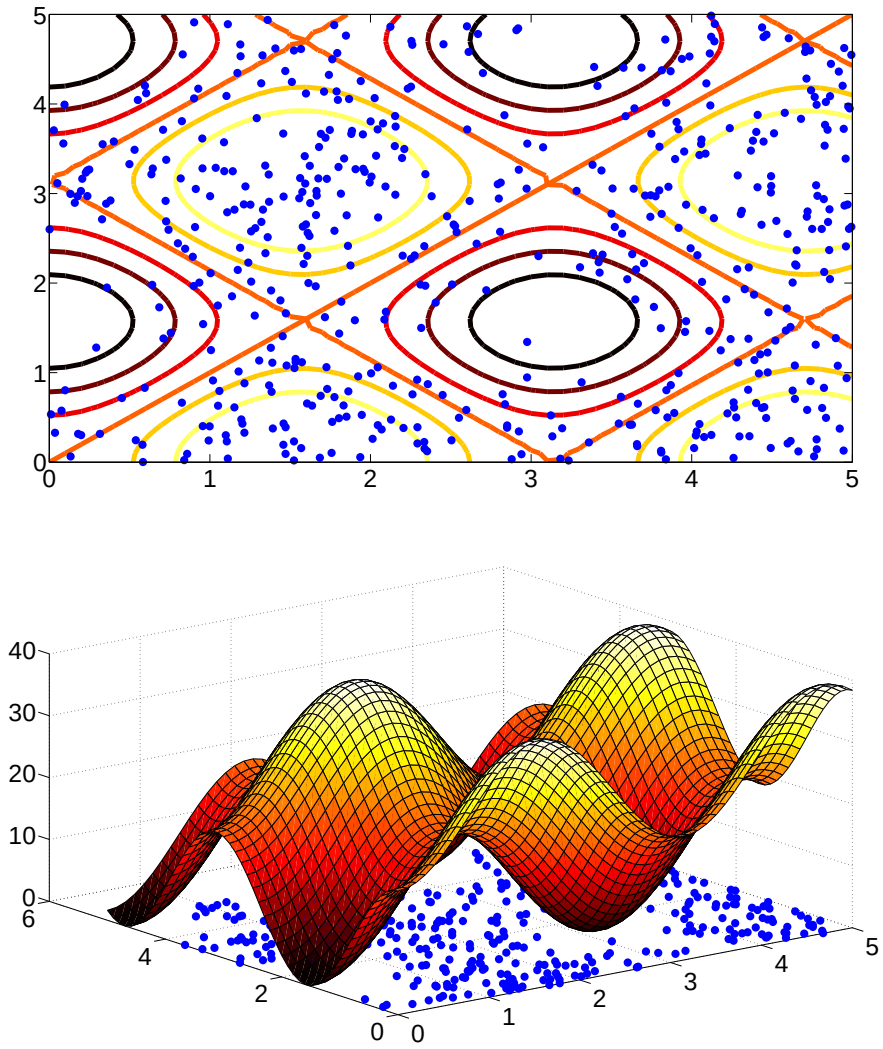

Figure 4.2.4: Realizations of an inhomogenous Poisson process with intensity $\mu(x, y) = 20\left(\sin^2(x) + \cos^2(y)\right)$ on $W = [0, 5] \times [0, 5]$.

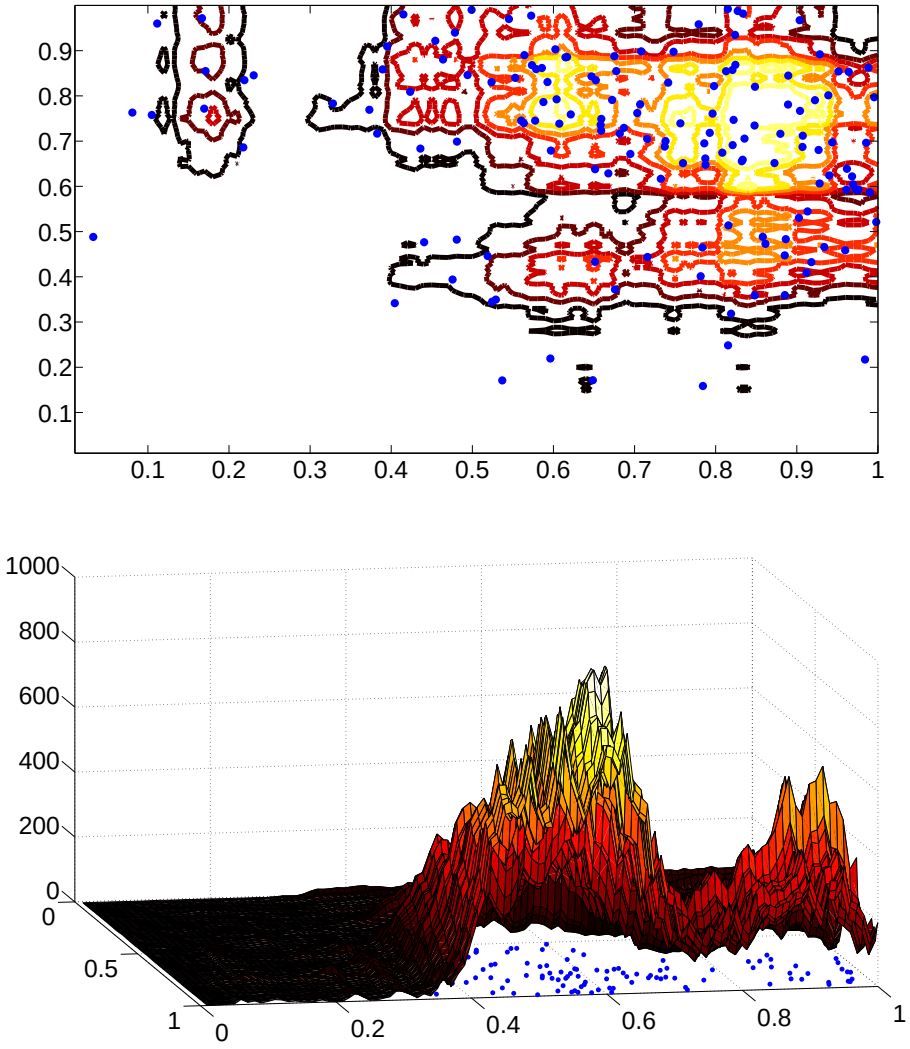

Figure 4.2.5: Realizations of an inhomogenous Poisson process with intensity $\mu(x, y) = 20\left(\sin^2(x) + \cos^2(y)\right)$ on $W = [0, 5] \times [0, 5]$.

```
    end
end

A = chol(Sigma); Y = randn(1,n1*n2) * A;
Lambda = reshape(300*Y.^2,n1,n2);
lambda_star = max(max(Lambda));
N = poissrnd(lambda_star)
X = [];

for i = 1:N
    x_temp = rand;
    y_temp = rand;
    ix = ceil(100 * x_temp);
    iy = ceil(100 * y_temp);

    if rand < (Lambda(iy,ix) / lambda_star)
        X = [X; x_temp y_temp];
    end
end

contour(x,y,Lambda)
hold on
S = 100 * ones(length(X),1);
scatter(X(:,1),X(:,2),S,'fill')
colormap hot
```

Bibliography

[1] M. B. Giles. Multilevel monte carlo path simulation. *Operations Research*, 56(3):607–617, 2008.

[2] T. M. Liggett. *Continuous Time Markov Processes: An Introduction*. American Maths Society, Providence, 2010.

[3] J. Norris. *Markov Chains*. Cambridge University Press, Cambridge, 1997.

[4] S. I. Resnick. *Adventures in Stochastic Processes*. Birkhäuser, Boston, 1992.