Notes for Math 450
Lecture Notes 1

Renato Feres

1 Probability Models

We often have to deal with situations involving hidden or unpredictable factors. Sometimes the unpredictability is an essential element of the phenomenon under investigation. For example, physicists regard radioactive decay as a purely random phenomenon caused by quantum fluctuations of the state of radioactive atoms. More often, the unpredictability is due to our lack of information of all the factors affecting the system, on account of its complexity or instability, as in weather forecasting. Or because information is deliberately being kept hidden from us, for example, when playing a game of poker. Even relatively simple, deterministic systems can be very unstable and behave in a practically unpredictable way. (See the discussion of the forced pendulum in a later lecture.)

In such cases it may be fruitful to formulate a probabilistic model that can serve as the basis for forecasting or decision-making regarding the system. For example, although it is not practically possible to predict whether a person will enter or leave a public building at any given time, for most purposes we can treat the arrival times as random, with a probability distribution depending on a few parameters that can be estimated by statistical methods. Similarly, we cannot predict what the exchange rate between the dollar and the euro will be next week, but we may be able to probabilistically model its fluctuations well enough to estimate the likelihood that this rate will be within a given range of values.

Sometimes we may want to use a randomized procedure to find the solution to a problem or to make a decision. There are many problems in mathematics in which a random search in a large set of candidates for a solution can be much more efficient than a systematic search. (This relates to the topic of Random Algorithms and Monte Carlo Methods.) Even in ordinary life situations we may want to contrive random procedures for deciding on a course of action, either because randomness may be equated with fairness (e.g., when picking which soccer team will give the initial kick) or because the situation at hand is so complicated that any attempt to deliberate in more analytical ways could lead to not making any decision at all (e.g., whether to marry Camilla or Sophia.)

Throughout this course we will consider probabilistic models of a variety of phenomena in areas ranging from finance to molecular biology. We will learn some of the most useful theoretical tools for studying random processes, such as Markov chains and stochastic differential equations, and will implement them by computer simulations.

1.1 Interpretations of probability

The mathematical theory of probability is very well developed and its foundations, as laid by Kolmogorov and others by the first few decades of the 20th century, are widely accepted as definitive. The axioms of probability theory that will be reviewed later are sufficient to develop all the purely mathematical results that we will encounter in the course. This axiomatization is centered on the notion of a *probability space*, which captures abstractly the idea of an experiment with a random outcome. A probability space consists of a triple of entities $(S, \mathcal{F}, P)$, where S, called the *sample space*, is a set describing the possible elementary outcomes of the experiment, $\mathcal{F}$ is a collection of subsets of S called *events*, and P is the *probability measure*, which assigns a probability, $P(E)$, to each event E in $\mathcal{F}$. Proper definitions of these terms will be given later.

To compare a probability model to a concrete situation, however, we need some interpretation that relates the mathematical concept of the probability $P(E)$ of an event E to our physical experience. There is some disagreement about how probability should be interpreted, and this is a topic of much philosophical discussion. The most common interpretations are the classical, the frequentist, and the Bayesian. (I take this summary from [Wilk]):

Classical interpretation. The classical interpretation of probability is based on the assumption that the events of interest are generated by a set of equally likely elementary events. In other words, the sample space might be decomposable (or *partitioned*, to use the standard term) into sets that, by symmetry or some other consideration, are assumed to have equal likelihood. If this is the case, one can then assign probabilities for the other events by expressing them as unions of elementary events. Most of the basic theory of probability for finite sample spaces is based on this idea.

For example, throwing two dice is an experiment with sample space consisting of all pairs (i, j) in the product set

$$S = \{1, 2, \dots, 6\} \times \{1, 2, \dots, 6\}.$$

If the dice are not unbalanced, we have no reason to suppose that any pair is more or less likely to occur than any other. As the total probability is 1 and there are 36 elementary events, the probability of each of them is 1/36. For a composite event $E \subset S$ we then have $P(E) = n/36$, where n is the number of elementary events in E. For example, the probability that the total number of pips is 4 is the probability of drawing one of the following pairs: $(1, 3), (2, 2), (3, 1)$. Since there are 3 favorable outcomes in 36, the probability is 1/12.

Frequentist interpretation. The frequentist interpretation is the one adopted by traditional statistical methodology. It applies to experiments that can in principle be repeated an arbitrary number of times under essentially identical conditions. The probability of an event E is then defined as the long-run proportion that E occurs under many repetitions of the experiment. If a mathematical model of the experiment is available in which it makes sense to consider an infinite repetition and passage to a limit, then the frequentist definition of probability takes the following form:

$$P(E) = \lim_{n\to\infty} \frac{\#\{k \in \{1, 2, \ldots, n\} : X_k \in E\}}{n},$$

where X_n denotes the outcome of the nth repetition of the experiment.

It is not immediately clear that such a limit should even exist. (Consider, for example, the possible outcome of an infinite sequence of coin tosses in which 0 represents "Head" and 1 represents "Tail":

$$1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \cdots$$

where the single digit blocks have sizes $1, 2, 3, 6, 12, 24, \ldots$.) Under appropriate mathematical conditions, the limit exits with probability 1, by the *law of large numbers* (to be studied later), so exceptional sequences as the one just given constitute a set of zero probability. The real problem, however, is that, as the interpretation only applies to outcomes of repeatable experiments under well-controlled conditions, statements such as "the probability that it will rain tomorrow" are not meaningful under the frequentist interpretation.

Subjective interpretation. The final common interpretation of probability is somewhat controversial, but it avoids the difficulties and limitations faced by the other two. It is based on the idea of *degree of belief* in the likelihood that the event will occur, subjectively estimated by an observer. It accepts that different people will assign different probabilities to the same event. One operational definition of subjective probability of an event E could be that $P(E)$ is the value (as a fraction of a dollar or any other measure of monetary value) that you would consider a *fair price* for a gamble that pays 1 dollar if E occurs, and nothing if E does not occur.

The subjective interpretation is sometimes known as the *degree of belief interpretation* or the *Bayesian interpretation*, after Thomas Bayes, the 18-th century Presbyterian minister who first proposed it. Despite its apparent vagueness, the Bayesian interpretation of probability underlies a powerful theory of statistical inference known as *Bayesian statistics*, which is widely used in science and engineering.

1.2 Sources of randomness

One useful way to specify a probability model, particularly when dealing with simulations, is to regard all the unpredictability of the system as being generated by a single, fixed "source of randomness," and define the experiment as a mathematical function of the state of this source. To make this idea more precise, consider a system that can be at a number of different states represented by elements of a state space S. (A more standard term is *sample space.*) Let X denote the state of the system at the time it is observed. For example, X might be the number of people in a public building at a given time, in which case S is the set $\{0, 1, 2, 3, \dots\}$. Or X might be the time between two blips of a Geiger counter, and S the set of all positive real numbers. Our fixed "source of randomness" is viewed as a separate system, independent of the one being investigated, whose state space we denote by Ω. The precise nature of Ω need not always be specified, but for the sake of fixing ideas let us picture it as an idealized fortune wheel, in which all angles between 0 and 2π are allowed outcomes. Writing the angle in the form $\theta = 2\pi x$, we can represent the state of our fortune wheel by a number x in the interval $\Omega = [0, 1]$. Spinning the wheel and waiting until it comes to a stop produces a number $x \in [0, 1]$ in such a way that all number between 0 and 1 are equally likely to occur. A probability model is now defined as a function $X : \Omega \to S$. As a simple example, consider the decision whether to marry Camilla or Sophia. This can be modeled by the following formal experiment: set $S = \{\text{Camilla}, \text{Sophia}\}$ and define the function $X : [0, 1] \to S$ by

$$X(x) = \begin{cases} \text{Camilla} & \text{if } x \leq 1/2 \\ \text{Sophia} & \text{if } x > 1/2. \end{cases}$$

In other words, we spin the wheel and wait until it settles down. If when it does the pointer is on the upper-half of the wheel, choose Camilla. Otherwise, choose Sophia. More generally, if we wish to model a random experiment with outcomes $s_1, s_2, \dots, s_k$, and probabilities $p_1, p_2, \dots, p_k$, we similarly divide the wheel into k sectors with the sector labeled by s_i comprising an angle $2\pi p_i$.

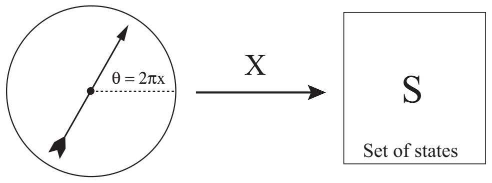

Figure 1: Schematic representation of a probability model with an idealized fortune wheel as our primary source of randomness. The number x lies in the interval $[0, 1]$. We can represent the position $e^{2\pi i x}$ of the arrow by the parameter x, keeping in mind that 0 and 1 describe the same position.

This idea is not limited to finite, or even countably infinite sets of possible outcomes. We will see later how to simulate in this way much more complicated random experiments. The function $X : \Omega \to S$ will be called at times a *probability model*, a *random experiment*, or, more often, a *random variable*.

The source of randomness Ω is more properly defined as a triple $(\Omega, \mathcal{F}, P)$, where Ω is the set of states of the source, $\mathcal{F}$ is a collection of subsets of Ω, called *events*, and P is a function on events defining a *probability measure*. For example, if $\Omega = [0, 1]$, then $\mathcal{F}$ consists of all subintervals $[a, b]$ and all subsets of $[0, 1]$ that can be obtained from intervals by taking unions, intersections, and complements a finite or countably infinite number of times. A standard choice of probability measure here is the one that assigns $[a, b]$ its length,

$$P([a, b]) = b - a.$$

The probability of other sets in $\mathcal{F}$ is then uniquely determined by the basic properties of P regarding the set operations. (This will be made a bit clearer later.) The probability space $(\Omega, \mathcal{F}, P)$ just defined is meant to represent abstractly the random experiment of picking a real number between 0 and 1 so that the probability that the number falls in $[a, b]$ is equal to $b - a$. We also call P, in this case, the *uniform probability measure* on $[0, 1]$.

The notion of a probability model as a mathematical function $X : \Omega \to S$ is very natural from the point of view of computer simulation. Most computers have a built-in function to produce pseudo-random numbers that approximates the idea of picking a uniformly distributed random number between 0 and 1. For example, in Matlab this is done by the function `rand`. We then use such a random number generator to simulate more complicated random situations.

It should be noted that it is often not necessary to refer to the space $(\Omega, \mathcal{F}, P)$ explicitly since the random variable X can be used to define a probability measure directly on S. In fact, the subset A of S can stand for the set $\{x \in \Omega : X(x) \in A\}$ (also written $\{X \in A\}$), and we can associate to A the probability

$$P_X(A) = P(X \in A).$$

For example, the event that the above decision experiment yields "Camilla," which is formally represented by the interval $[0, 1/2]$, has probability

$$P_X(\text{Camilla}) = P(X = \text{Camilla}) = P([0, 1/2]) = 1/2 - 0 = 1/2.$$

By this remark, the sets Ω and S are often not distinguished. We will often use the letters S or Ω in similar situations.

A point that may not be obvious right now, but will become natural as we consider more examples, is that our idealized fortune wheel (or, more prosaically, a source of uniformly distributed numbers between 0 and 1) is all that we need in order to produce even the most complicated probability distributions we will encounter. The precise meaning and justification of this claim requires some knowledge of measure theory and is outside the scope of our discussion. I only mention a few remarks to dramatize the point.

First, I note that a same source Ω can take many different guises without essentially changing its nature as a probability space. More precisely, two probability spaces $(\Omega_1, \mathcal{F}_1, P_1)$ and $(\Omega_2, \mathcal{F}_2, P_2)$ are said to be *equivalent* if, after possibly discarding sets of zero probability in Ω_1 and Ω_2, we can find a bijection $X : \Omega_1 \to \Omega_2$ such that (i) X establishes a bijection between elements of $\mathcal{F}_1$ and $\mathcal{F}_2$ (note that, if A is a set in $\mathcal{F}_2$, then $\{\omega \in \Omega_1 : X(\omega) \in A\}$ belongs to $\mathcal{F}_2$, and vice-versa using X^{-1}), and (ii) if A_1 and A_2 are elements of $\mathcal{F}_1$ and $\mathcal{F}_2$ that correspond to each other under X, then $P_1(A_1) = P_2(A_2)$. If the two probability spaces are related in this way by such a function X, then they should be regarded as indistinguishable sources of randomness and each can be viewed as a random experiment whose source is the other, via the functions X and X^{-1}.

Now, it should not be obvious if you have not thought about this before, but it is not a terribly difficult fact to establish that there exists an equivalence of probability spaces

$$F : [0, 1] \to [0, 1]^K,$$

where K is any positive integer. (In fact, K can be taken to be ∞, $[0, 1]^\infty$ being the set of all sequences $(x_1, x_2, x_3, \dots)$ of real numbers in $[0, 1]$.) This can be interpreted as follows: the experiment of picking K uniformly distributed, independent random numbers between 0 and 1 can be modeled by a formal experiment $X : [0, 1] \to S$, where $S = [0, 1]^K$. Here is a hint of how this function can be obtained: expand a real number in base 2, say, and rearrange the digits so as to produce K other numbers, also in base 2. If you do it right, sets of numbers that are bijectively related by this map will have the same probability.

The infinite nature of the interval $[0, 1]$ makes it an idealized mathematical notion that is only approximated in computer simulations. We will have more to say about algorithms for generating (pseudo-) random numbers. For now, we will simply trust that computer operations such as `rand`, in Matlab, are good enough approximations in practice of what our idealized fortune wheel does in theory.

1.3 Stochastic processes

We have so far considered experiments that are performed once. Much of our course will be dedicated to the study of *random (or stochastic) processes*. A stochastic process is a family, X_t, of random variables indexed by a time parameter t. It describes the time evolution of a random quantity under study. For example, X_t, $t = 0, 1, 2, \dots$, could represent the daily average of the exchange rate between dollars and euros. The parameter t can be discrete as in this example, or continuous, as in the number, N_t, of occupants of a public building at any time t during a day. It may involve a discrete set of states as in these examples, or a continuous set of states as in Brownian motion and other diffusion processes that will be studied later.

An example of a stochastic process is a random walk. Consider the motion in the plane specified by the following probabilistic rule. Starting at the origin, $X_0 = (0, 0)$, choose at random (with equal probabilities) an angle θ in the set $\{2\pi k/u : k = 1, 2, \dots, u\}$ then jump to the point $X_1 = X_0 + r(\cos(\theta), \sin(\theta))$, where r is a fixed length covered by a single jump. By the same procedure, find X_{n+1} from X_n, for $n = 1, 2, \dots$. This describes a path on the plane that can be interpreted as a particle moving uniformly for a unit time, then hitting an obstacle that causes it to change direction randomly. After the collision, the particle resumes its uniform motion until the next collision one unity of time later. More realistic models of random motion involve random collision times, which will be considered later. You will play with this example further in the exercises below. The following figure shows a typical sample path for the random walk.

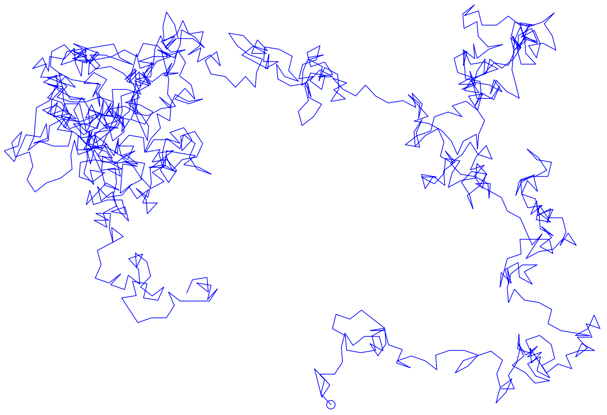

Figure 2: Random walk on the plane. The random angles are multiples of $2\pi/19$ and there are 1000 steps. The initial position is marked with a circle.

1.4 Markov Chains

An important class of stochastic processes are Markov chains. A large part of our study will be dedicated to Markov chains in discrete and continuous time.

A discrete time Markov chain is a system defined by the following properties.

1. The system can be at any of a finite or countably infinite number of states, comprising set of states $S = \{s_1, s_2, \dots\}$.

2. Starting in some initial state at time $t = 0$, the system changes its state randomly at times $t = 1, 2, \dots$. Representing the state at time k by the random variable X_k, the evolution of the system is described by the chain of random variables
$$X_0 \to X_1 \to X_2 \to \cdots$$

3. At time $t = 0$, the system occupies the state s_k with initial probability
$$\pi_0(s_k) = P(X_0 = s_k),\ k = 1, 2, \dots$$

4. Suppose that at any time n the system has the following history:

$$X_i = x_i, \; i = 0, 1, 2, \ldots, n$$

where $x_n = s_i$ for some i. Then the probability that the system goes into the state $x_{n+1} = s_j \in S$ at the next time step only depends on the state at the present time n and not on the system's past. The transition probability from state s_i to s_j is written

$$p_{ij}(n) = P(X_{n+1} = s_j | X_n = s_i).$$

In other words, given the present state of the system, $X_n = s_i$, the future of the system is independent of the past. The numbers $p_{ij}(n)$ are called the *transition probabilities* of the system. If the Markov property holds and the transition probabilities do not depend on n, we say that the process is a *(time) homogeneous Markov chain.*

A useful way to represent the transition probabilities for a Markov chain is by a graph, such as in figure 3

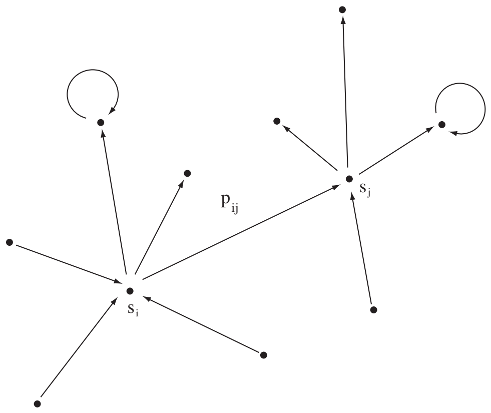

Figure 3: Markov chains correspond to random walks on graphs. The numbers p_{ij} are the probabilities that the state will change along the direction specified by the corresponding arrow.

For example, consider the following simple model of random web surfing. Let $S = \{s_1, s_2, \ldots, s_n\}$ denote the set of all web pages on the internet. A page, $s_i \in S$, is linked to a number d_i of other pages. A random surfing corresponds to a chain, $X_0, X_1, X_2, \ldots$ such that

1. at time $t = 0$, $X_0 =$ `http://www.myhomepage.edu` with probability 1; and

2. if at time $t = k$ the surfer is at the web page s_i, then at time $t = k + 1$ he/she will have jumped to one of the d_i pages linked from s_i with equal

probability, $1/d_i$. If a page s_i has no links from it we define $p_{ij} = 0$ for all $j \neq i$, and $p_{ii} = 1$. This is an example of a *random walk on a graph.* (More realistic models of web browsing have been proposed and studied.)

Here is a crude Markov chain model of weather forecasting. The town of Markoville has only two possible weather conditions:

$$s_1 = \text{"fair" and } s_2 = \text{"rainy."}$$

Empirical observation has shown that the best predictor of Markoville's weather tomorrow is today's weather, with the following transition probabilities:

		tomorrow	
		rain	fair
today	rain	0.75	0.25
	fair	0.25	0.75

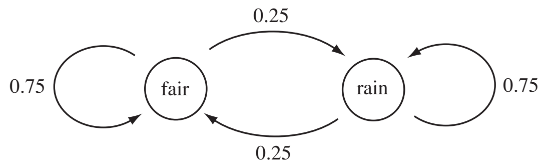

Figure 4: Two representations of the transition probabilities of a Markov chain: as a transition probability matrix and as a graph whose vertices are the possible states and the edges are the available transitions between states.

The information required to specify a Markov chain model is an initial probability distribution of states, π, and a transition probability matrix P. In the weather example, this matrix is given by

$$P = \begin{pmatrix} 0.75 & 0.25 \\ 0.25 & 0.75 \end{pmatrix}.$$

We will consider many more examples later. Markov chains become especially powerful probabilistic models after we introduce the idea of a random time step between state transitions. This will be studied later in the course.

1.5 Stochastic Differential Equations

Stochastic differential equations generalize ordinary differential equations by adding a noisy driving term. For example, a first order differential equation of the classical type has the form $y' = f(t, y)$. In a typical stochastic ODEs, one wishes to include an additional term

$$y' = f(t, y) + g(t, y)\text{noise}(t).$$

It takes a certain amount of work to make proper sense of the random driving term added to the classical equation and to figure out how to interpret the solution as a stochastic process. Roughly speaking, this can be understood as follows. Recall that the classical equation $y' = f(t, y)$ can be solved by taking limits of approximate solutions constructed using Euler's scheme: given an approximation y_n of $y(t_n)$, obtain $y_{n+1} = y_n + f(t_n, y_n)h$, where h is a small time step. For the stochastic equation we add a random term to $y(t_n)$:

$$y_{n+1} = y_n + f(t_n, y_n)h + g(t_n, y_n)N_n$$

where N_n is a (normally distributed) random variable with expected value 0 and variance proportional to h. Thus the process consists of a slow acting term, f, called the *drift*, and a random term that adds relatively big jumps (proportional to $\sqrt{h}$, which is the order of magnitude of the standard deviation of the noise term) at a random direction. We will spend a fair amount of time later making sense of these ideas.

2 Exercises and Computer Experiments

In this section we give a quick overview of some of the basic Matlab functions dealing with probability and random numbers and use them to perform some simple numerical experiments.

2.1 Uniform random numbers

The command, `rand`, is based on a popular deterministic algorithm called *multiplicative congruential method.* It uses the equation

$$u_{i+1} = Ku_i (\text{mod } M), \quad i = 1, 2, 3, \ldots$$

where K and M are usually integers and mod M is the operation that gives the remainder of division by M. (The remainder is a number between 0 and M.) In Matlab this operation is written `rem(K*u, M)`. Therefore, $r_i = u_i/M$ is a number in $[0, 1)$. To apply this method, one chooses suitable constants K and M and an initial value u_1, called the *seed.* Matlab seems to use $K = 7^5$ and $M = 2^{31} - 1$. These are, in any event, appropriate values for the constants.

```
r = rand               %returns a single random number uniformly
                       %distributed over (0,1). Its value changes
                       %each time the command is invoked.

r = rand(n)            %returns an n-by-n matrix of independent
                       %uniformly distributed random entries.

r = rand(m,n)          %returns an m-by-n matrix of independent
                       %uniformly distributed random entries.

r = rand([m n])        %same
```

```
r = rand(m,n,p,...)   %generates a random array of dimensions
                      %m-by-n-by-p-...

r = rand([m n p...])  %same

rand('seed', 57)      %sets the 'seed' of the pseudo-random
                      %number generator to 57. Once the seed
                      %is set to a given value, the algorithm
                      %always produces the same sequence of
                      %random numbers. This is useful if we
                      %need to use the same random numbers
                      %more than once, or to produce identical
                      %runs of the same simulation.

s = rand('state')     %returns a 35-element vector containing the
                      %current state of the uniform generator.

rand('state',0)       %Resets the generator to its initial state.

rand('state',j)       %For integer j, resets the generator
                      %to its j-th state.

rand('state',sum(100*clock))
                      %Resets it to a different state each time.
```

We can use a histogram to visualize a distribution of numbers. This is done by the Matlab command `hist`.

```
n = hist(Y)             %bins the elements in vector Y into
                        %10 equally spaced containers and returns
                        %the number of elements in each container
                        %as a row vector. If Y is an m-by-p matrix,
                        %hist treats the columns of Y as vectors
                        %and returns a 10-by-p matrix n. Each
                        %column of n contains the results for the
                        %corresponding column of Y.

n = hist(Y,x)           %where x is a vector, returns the distribution
                        %of Y among length(x) bins with centers
                        %specified by x. For example, if x is a
                        %5-element vector, hist distributes the
                        %elements of Y into five bins centered on
                        %the x-axis at the elements in x. Note:
                        %use histc if it is more natural to specify
                        %bin edges instead of centers.

n = hist(Y,nbins)       %where nbins is a scalar, uses
                        %nbins number of bins.

[n,xout] = hist(...)    %returns vectors n and xout containing
                        %the frequency counts and the bin locations.
                        %You can use bar(xout,n) to plot the histogram.

hist(...)               %without output arguments produces a histogram
                        %plot of the output described above. hist
                        %distributes the bins along the x-axis between
                        %the minimum and maximum values of Y.
```

For example, `Y=rand(10000,1); hist(Y,20)` produces the plot of figure 5.

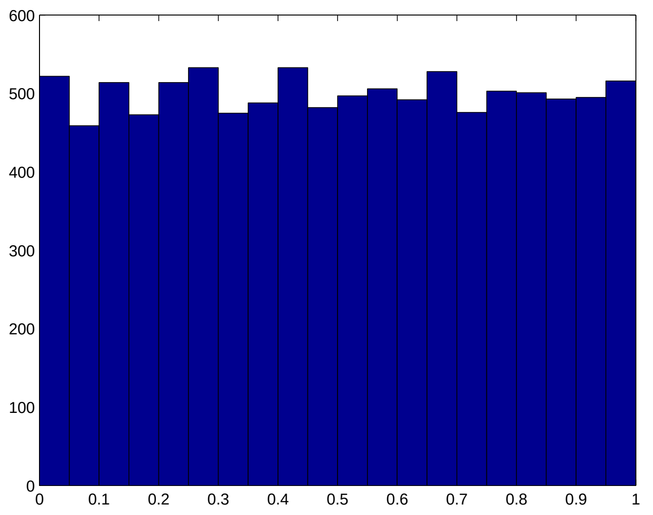

Figure 5: Histogram plot of a set of 10000 uniformly distributed random numbers in $(0, 1)$ divided into 20 bins.

Exercise 2.1 The experiment of tossing a coin once can be simulated by

```
if rand < 0.5
    a=0;
else
    a=1;
    end
```

Modify this program to simulate the following experiment: toss 10 coins each time and count how many come up tail. Repeat the experiment 1000 times and plot a histogram of the outcome. (The outcome is a vector `Y` of size 1000 whose entries are integers between 1 and 10. Use `hist(Y)` to produce a histogram plot with 10 bins.)

Exercise 2.2 Repeat exercise 2.1, but instead of Matlab's `rand`, use the multiplicative congruential algorithm

$$u_{i+1} = Ku_i (\text{mod } M), \quad i = 1, 2, 3, \ldots$$

described at the beginning of the section as the source of pseudo-random numbers.

Exercise 2.3 Plot a histogram with 50 bins of the determinants of 5000 matrices of size 3-by-3 whose entries are independent uniformly distributed over $(0, 1)$ random numbers. Do the same for the trace instead of determinant.

2.2 Random integers

The Matlab functions `ceil`, `floor`, `round`, and `fix`, combined with `rand`, are useful for generating random integers.

```
ceil(x)     %Smallest integer greater than or equal to x

floor(x)    %Largest integer less than or equal to x

round(x)    %x rounded to the nearest integer

fix(x)      %rounds x towards 0 to the nearest integer. If
            %x is positive, fix(x) equals floor(x) if x is
            %positive and ceil(x) if x is negative
```

These functions also act on vectors and arrays. For example, to get a 3-by-3 matrix A with independent random entries drawn from $\{1, 2, 3, \ldots, k\}$ with equal probabilities, we can use `x=rand ; A=ceil(k*x)`. If we want to draw from $\{0, 1, 2, \ldots, k-1\}$ we can use `floor` instead of `ceil`.

Exercise 2.4 Explain why the following statements are true:

1. The Matlab command `a=floor(2*rand)` picks a number from $\{0, 1\}$ with equal probability 1/2.
2. The Matlab command `a=round(2*rand)` picks a number from $\{0, 1, 2\}$ with different probabilities. What are those probabilities?

The Matlab relational operators $<$, $<=$, $>$, $>=$, $==$, $\sim=$ are often needed to indicate whether a number belongs to a set or not. Relational operators perform element-by-element comparisons between two arrays. They return a logical array of the same size, with elements set to 1 (true) where the relation is true, and 0 (false) where it is not. For example, if we define `x=[1 3 2 5 1]` and `y=[0 3 1 6 2]`, then `(x>=y)` returns the array `[1 1 1 0 0]`. If all the entries in an array `X` are being compared to the same number `a`, then we can apply the relational operators to `a` directly. For example, set `rand('seed',121)`. Then `(2*rand(1,8) <= 1)` produces the sequence 1 0 1 0 1 0 1 1.

We can combine the relational operators with the logical operations `&` (logical AND), | (logical OR), $\sim$ (logical complement, NOT) and `xor` (exclusive OR) to represent sets. For example, the command `(x>=2 & x<5)` returns 1 if a number x belongs to the interval $[2, 5)$ and 0 if not. It represents the *indicator function* of the interval.

Consider the experiment of drawing (sampling) numbers from $\{1, 2, \ldots, k\}$ with probabilities p_1, p_2, $\ldots$, p_k. The following script defines a function

`x=samplefromp(p,n)`

of $p = [p_1, p_2, \ldots, p_k]$ (where $p_1 + p_2 + \cdots + p_k = 1$) and positive integer n that returns a row vector `x` of length n of numbers drawn from $\{1, 2, \ldots, k\}$ with the probabilities specified by p.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function x=samplefromp(p,n)
%Inputs - p is the probability vector of length k
%       - n is the number of random
%         integers from 1,2, ...,k returned
%Output - a row vector of length n with entries
%         from the set {1, 2, ..., k} with
%         probabilities specified by p.
k=size(p,2);
u=rand(1,n);
x=zeros(1,n);
for i=1:k
    x=x+i*(sum(p(1:i))-p(i) <=u & u<sum(p(1:i)));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Exercise 2.5 Perform the following simulated experiment. Roll 3 dice and add the number of pips on them. Do this 1000 times and plot a histogram with 18 bins showing the frequencies of numbers from 1 to 18.

Suppose that x is a row or column vector regarded as a data set with n entries. Recall that the mean of x is

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

There are two textbook definitions of standard deviation of x. They are:

$$s_1 = \frac{1}{n-1}\left(\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^{1/2} \quad \text{and} \quad s_2 = \frac{1}{n}\left(\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^{1/2}.$$

The first will be our preferred definition for data sampled from a population or simulation. We will justify this choice later. We now summarize some of the Matlab functions for basic data analysis.

```
mean(x)       %gives the arithmetic mean, or average, of x.

median(x)     %gives the middle value or the arithmetic mean
              %of the two middle values of the data.

std(x)        %gives the standard deviation of x according
              %to the first definition, s1, above.

std(x,1)      %gives the standard deviation of x according
              %to the second definition.

max(x)        %finds the largest value in the data set x.

min(x)        %finds the smallest value in the data set x.

sum(x)        %computes the sum of the data.

prod(x)       %computes the product of all data values.
```

These functions also apply to matrices. If A is an m-by-n matrix, the operations act on the columns of A, returning a row vector. For example `mean(A)` returns a row vector of size n whose jth entry is the means of the jth column of A. The following are also useful:

```
cumsum(x)     %computes the cummulative sum. For example,
              %if x=[1 2 3 4], cumsum(x)=[1 3 6 10].

cumprod(x)    %computes the cummulative product. For example,
              %if x=[1 2 3 4], cumprod(x)=[1 2 3 24].
```

```
sort(x)       %sorts data in ascending order.
              %An optional argument, [y,k]=sort(x), gives the
              %indices of the sorted values. For example,
              %if x=[23 4 7 3 5 12 3], then [y,k]=sort(x) returns
              %y=[3 3 4 5 7 12 23] and k=[4 7 2 5 3 6 1].
```

2.3 Random walks

The following Matlab script can be used to generate a sample random walk like the one of figure 2.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
K=1000;
r=1;
u=19;
X=[0 0];
for n=1:K
    s=ceil(u*rand);
    Z=X(n,:)+r*[cos(2*pi*s/u) sin(2*pi*s/u)];
    X=[X;Z];
end
plot(X(:,1), X(:,2))
hold on
plot(0,0,'o')
hold off
axis equal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Exercise 2.6 Modify the above program to find the moment of time when a particle undergoing random walk starting from $(0, 0)$ will first be at a distance at least a from the origin. For each $a = 1, 1.5, 2, 2.5, 3, \ldots, 10$ repeat the experiment 100 times and find the mean time $T(a)$. (This took about half a minute to run on my laptop.) Plot the mean time T as a function of a. On the same coordinate system, plot the quadratic best fit of T. (Matlab's graph window has commands for curve fitting.) You will note that the best quadratic fit looks like a fairly good fit. This is an indication that the random walk first reaches a distance a at time proportional to a^2. We will provide a mathematical justification later.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r=1;
u=19;
t=[];
k=20;
for a=1:0.5:k
    S=[];
    for j=1:100
        Z=[0 0];
        for n=1:1000000000000
            s=ceil(u*rand);
            Z=Z+r*[cos(2*pi*s/u) sin(2*pi*s/u)];
            if Z(1)^2+Z(2)^2>=a^2
                S=[S n];
                break
            end
        end
    end
    t=[t sum(S)/100];
end
a=1:0.5:k;
plot(a,t)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Exercise 2.7 Let A_1, A_2, A_3 denote three points on the coordinate plane. For concreteness, we take $A_1 = (0,0)$, $A_2 = (2,0)$, and $A_3 = (1,\sqrt{3})$. Choose an initial point, $X_0 \in \mathbb{R}^2$ and define a random point X_n, for each positive integer, by the following procedure. Suppose X_{n-1} is the position at present. To determine the position X_n, choose one of the three points at random with equal probabilities, say $B_n \in \{A_1, A_2, A_3\}$, then set X_n as the middle point between B_n and X_{n-1}. Plot the first one thousand points $X_1, \ldots, X_{1000}$. (This time, do not connect the points with lines as in the first random walk.)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%A random walk on the plane.
%
%Starting from a point X1, define
%a sequence of random points X(1), X(2), ... by
%the recursive relation: X(n) = (X(n-1) + A)/2,
%where A is a random point chosen from among three
%points A1, A2, A3 with equal probabilities.
%N is the number of iterations.
A1=[0 0];
A2=[2 0];
A3=[1 sqrt(3)];
X1=[1 1];
N=1000;
X=X1;
for i=1:N-1
```

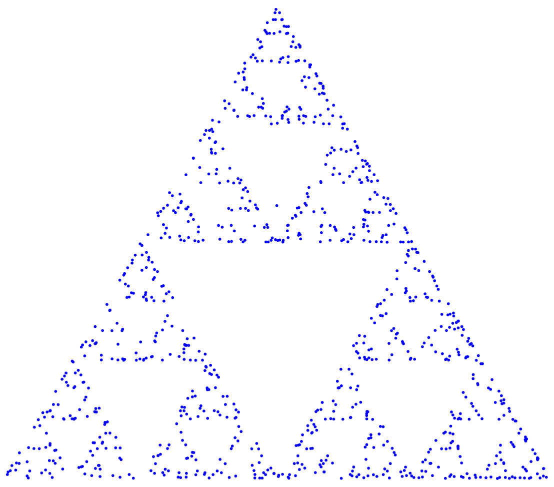

Figure 6: The first one thousand points of a sample trajectory of a random walk starting at $(1, 1)$ such that each X_n is the middle point between X_{n-1} and one of A_1, A_2, A_3 chosen at random with equal probabilities.

```
    x=rand;
    A=A1*(x<=1/3)+A2*(x>1/3 & x<=2/3)+A3*(x>2/3);
    B=(X(i,:)+A)/2;
    X=[X;B];
end
plot(X(:,1),X(:,2),'.')
axis equal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

2.4 Simulating a discrete time Markov chain

Consider a Markov chain with finite state space $S = \{s_1, \ldots, s_n\}$, initial probability vector $\pi_0 = (p_1, p_2, \ldots, p_n)$, and transition probabilities matrix $P = (p_{ij})$. We wish to do a computer simulation of the process $X_0, X_1, X_2, \ldots$. Each run of the program should produce a sequence

$$x_0 \to x_1 \to x_2 \to \ldots$$

in such a way that x_0 is chosen from S according to the probability vector π_0 and each new x_{n+1} is chosen according to the transition probability matrix and the just obtained value of x_n.

We can use our program `samplefromp`. Denote by r_i the ith row of P. Then r_i is the probability vector that gives the transition probabilities from state i to each of the other states. To simulate a sample trajectory of the Markov chain, first set the probability vector to $p = \pi_0$, and draw a state `i=samplefromp(p,1)`. Then, at each future step, set $p = r_i$ where i is the last state drawn, and apply again `i=samplefromp(p,1)`. The following Matlab script implements this procedure.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function X=samplefrommarkov(p,P,n)
%Inputs  - p probability distribution of initial state
%        - P transition probability matrix
%        - n number of iterates.
%Output  - X sample chain of length n.
%
%Note: need function samplefromp.m
q=p;
i=samplefromp(q,1);
X=[i];
for j=1:n-1
    q=P(i,:);
    i=samplefromp(q,1);
    X=[X i];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

As an example we implement this method for the Markov chain with initial probability vector

$$\pi_0 = (0.2, 0.5, 0.3)$$

and transition probability matrix

$$P = \begin{pmatrix} 0.8 & 0.1 & 0.1 \\ 0.3 & 0.4 & 0.3 \\ 0.4 & 0.1 & 0.5 \end{pmatrix}.$$

Set

```
p=[0.2 0.5 0.3]; P=[0.8 0.1 0.1;0.3 0.4 0.3;0.4 0.1 0.5]; n=5000;
```

then

```
X=samplefrommarkov(p,P,n);
```

We can get a sense of how the frequencies of the various states stabilize by calculating the frequencies at each moment of time:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
j=1;    %choose a state for which to obtain running frequency
n=5000; %this is the value used in X=samplefrommarkov(p,P,n)
f=[];
for i=1:n
    f=[f sum(X(1:i)==j)/i];
end
plot(1:n,f)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

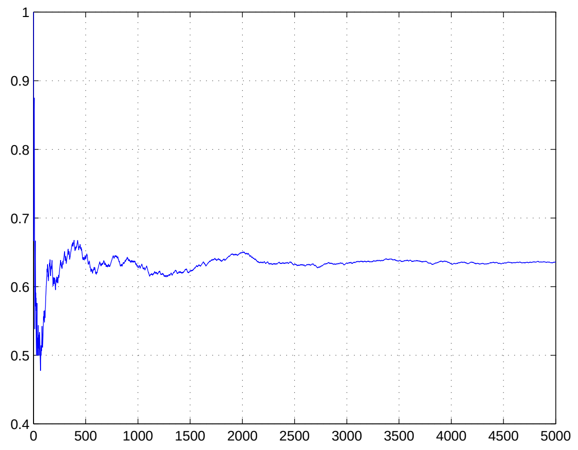

Figure 7: Frequency of occurrence of state 1 as function of time. The frequency at $n = 5000$ observed for this sample path is 0.6358. The exact value can be found by calculating the eigenvector of P for eigenvalue 1 (normalized, so that the entries add up to 1). Then the entries of the eigenvector are the relative frequencies of the Markov chain states. In this case, we obtain 0.643 for state 1.

Exercise 2.8 Simulate the weather forecasting Markov chain of section 1.4 and find the long term frequencies of fair and rainy weather. Compare the frequencies you obtain with the eigenvalues of the transition probability matrix.

2.5 Random permutations and shuffling

Suppose m books are piled up on your desk. The books are labeled by the integers $1, 2, \ldots, m$. The order of the books from the top of the pile down to the bottom is described by a permutation $s = (i_1, i_2, \ldots, i_m)$, where i_1 is the number of the book on top of the pile, i_2 the second from the top, and so forth. Therefore, the set of possible states the book pile can be in is described by the set of all permutations of m elements. This is a set (in fact, *group* in the algebraic sense) of $m!$ elements. To each book is associated a probability $p_i, i = 1, \ldots, m$ of being picked at any particular moment. After being picked, the book is returned to the top of the pile.

Let us look at the special case of three books: $m = 3$. We enumerate the states as follows: $s_1 = (1, 2, 3)$, $s_2 = (1, 3, 2)$, $s_3 = (2, 1, 3)$, $s_4 = (2, 3, 1)$, $s_5 = (3, 1, 2)$, $s_6 = (3, 2, 1)$. Having ordered the states of the pile, the transition probability matrix can now be written as follows:

$$P = \begin{pmatrix} p_1 & 0 & p_2 & 0 & p_3 & 0 \\ 0 & p_1 & p_2 & 0 & p_3 & 0 \\ p_1 & 0 & p_2 & 0 & 0 & p_3 \\ p_1 & 0 & 0 & p_2 & 0 & p_3 \\ 0 & p_1 & 0 & p_2 & p_3 & 0 \\ 0 & p_1 & 0 & p_2 & 0 & p_3 \end{pmatrix}.$$

The transition probabilities can also be represented in graphical form as shown in Figure 8.

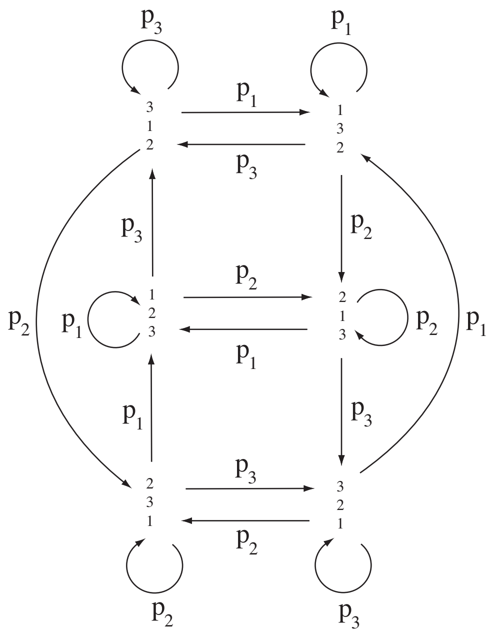

Figure 8: Transition probabilities for the “pile of books” example.

Exercise 2.9 Suppose that the books have probabilities of being picked equal to $p_1 = 0.5$, $p_2 = 0.3$, $p_3 = 0.2$. Obtain a sample of the book-shuffling Markov chain of length 5000 and use it to estimate the frequencies of each being on top to the pile.

Exercise 2.10 Find by hand the eigenvalues of the general three-book shuffling transition matrix P. How are the eigenvalues of P related to p_1, p_2, p_3?

References

[Wilk] Darren J. Wilkinson. *Stochastic Modelling for Systems Biology*, Chapman and Hall/CRC, 2006.

[Prat] Rudra Pratap. *Getting Started with Matlab - a Quick Introduction for Scientists and Engineers*, Oxford University Press, 2002.