



UNIVERSITY  
*of* ALASKA  

---

*Many Traditions One Alaska*

## Mathematical Modeling and Simulation with MATLAB

Item Type	Book
Authors	Buzby, Megan; Lee, Sheldon
Citation	Lee, S., Buzby, M. (2021). Mathematical Modeling and Simulation with MATLAB. ©2021 by Sheldon Lee and Megan Buzby. This book is Licensed under CC BY-SA 4.0
Download date	26/10/2022 08:27:57
Link to Item	<a href="http://hdl.handle.net/11122/12246">http://hdl.handle.net/11122/12246</a>

# Mathematical Modeling and Simulation with MATLAB

Sheldon Lee, Viterbo University

Megan Buzby, University of Alaska Southeast

2021

**Authors:** Sheldon Lee and Megan Buzby

**Copyright:** ©2021 by Sheldon Lee and Megan Buzby

**License:** This book is Licensed under CC BY-SA 4.0

**Website:** <http://hdl.handle.net/11122/12246>

# Contents

<b>Preface</b>	<b>5</b>
<b>1 Introduction</b>	<b>7</b>
1.1 What is Modeling? . . . . .	7
1.1.1 Types of Mathematical Models . . . . .	7
1.2 Modeling with Equations . . . . .	8
1.3 Modeling with Recurrence Relations . . . . .	10
1.4 Modeling with Differential Equations . . . . .	11
1.5 Stochastic Models . . . . .	12
1.6 Exercises . . . . .	12
<b>2 Programming in MATLAB</b>	<b>17</b>
2.1 Why is Programming Important? . . . . .	17
2.2 MATLAB Basics . . . . .	18
2.3 MATLAB Functions and Terminology . . . . .	20
2.4 Plotting Points and Curves . . . . .	21
2.4.1 Using ezplot() . . . . .	21
2.4.2 Using plot() . . . . .	21
2.5 The Symbolic Toolkit . . . . .	22
2.6 For Loops . . . . .	22
2.7 While Loops . . . . .	25
2.8 Conditional Statements . . . . .	28
2.9 Exercises . . . . .	31
<b>3 Iterative Methods</b>	<b>35</b>
3.1 Fixed Point Iteration . . . . .	35
3.2 MATLAB function files . . . . .	38
3.3 The Bisection Method . . . . .	39
3.4 Newton's Method . . . . .	41
3.5 Exercises . . . . .	44

<b>4</b>	<b>Matrices</b>	<b>46</b>
4.1	Matrix Review . . . . .	46
4.2	Eigenvectors and Eigenvalues . . . . .	54
4.3	Summary of MATLAB commands . . . . .	57
4.4	Exercises . . . . .	57
<b>5</b>	<b>Discrete Models</b>	<b>60</b>
5.1	Introduction . . . . .	60
5.2	Linear Dynamical Systems . . . . .	61
5.3	State, Age, and Stage Matrix Models . . . . .	63
5.4	Markov Chains . . . . .	65
5.5	Higher Order and Nonlinear Discrete Dynamical Systems . . . . .	71
5.6	Exercises . . . . .	74
<b>6</b>	<b>Continuous Models</b>	<b>81</b>
6.1	Modeling with Differential Equations . . . . .	81
6.1.1	Euler's Method . . . . .	84
6.1.2	Improved Euler's Method . . . . .	86
6.1.3	Qualitative Analysis of Differential Equations . . . . .	88
6.2	Systems of Differential Equations . . . . .	90
6.3	Qualitative Analysis of Systems of Differential Equations . . . . .	93
6.4	Linear systems . . . . .	94
6.5	Nonlinear Systems of Equations . . . . .	99
6.6	Exercises . . . . .	100
<b>7</b>	<b>Stochastic Modeling</b>	<b>113</b>
7.1	Discrete Random Variables . . . . .	114
7.2	Continuous Random Variables . . . . .	117
7.3	Properties of Random Variables . . . . .	119
7.4	Monte Carlo Integration . . . . .	122
7.5	The Binomial Distribution . . . . .	123
7.6	The Normal Distribution . . . . .	126
7.7	Waiting Times . . . . .	127
7.7.1	The Poisson Distribution . . . . .	127
7.7.2	The Exponential Distribution . . . . .	129
7.7.3	Sampling from the Exponential and Poisson Distributions . . . . .	131
7.8	Stochastic Processes . . . . .	134
7.8.1	Poisson Processes . . . . .	135
7.8.2	Birth Death Processes . . . . .	136
7.9	Exercises . . . . .	139
<b>A</b>	<b>In-class Activities and Exercises</b>	<b>145</b>

<i>CONTENTS</i>	4
<b>B Solutions to in-class Exercises</b>	<b>188</b>
<b>C Selected Solutions</b>	<b>193</b>
<b>D MATLAB Commands</b>	<b>198</b>
<b>E Debugging</b>	<b>202</b>
<b>F Completed Proofs</b>	<b>204</b>
<b>Index</b>	<b>209</b>

# About this text

This textbook attempts to provide you with an overview of the commonly used basic mathematical models, as well as a wide range of applications. It offers a perspective that brings you back to the modeling process and the assumptions that go into it.

The text is designed for the student new to both computer programming and mathematical modeling. It gives an overview how to use procedural programming in MATLAB in order to simulate and approximate solutions to mathematical models. This text assumes a basic knowledge of differential and integral calculus. It is helpful if you have taken courses in linear algebra, differential equations, and computer programming. It also assumes a background of descriptive statistics, specifically, you should have seen concepts such as mean, standard deviation, percentiles, histograms, and linear regression. If you have already taken a course in computer science or even know some MATLAB you should find Chapter 2 relatively simple but may still benefit from reading through it and working through some exercises. If you are not familiar with these areas then don't panic - this book attempts to give you a brief introduction to each and the tools you will need.

You will need access to MATLAB in order to successfully work through the text. Add-ons such as the Symbolic and Statistics and Machine learning toolkits are helpful but not required.

In this first chapter, we give a rough overview of the modeling process and discuss some models you are probably already familiar with. The problems require you to review some of the topics from calculus, and are also meant to get you thinking about solving problems in modeling.

Chapter 2 gives a brief overview of MATLAB commands - how to define variables, carry out basic mathematical operations, and make basic plots. The chapter later introduces procedural programming in MATLAB with several examples from various areas of mathematics. In Chapter 3, you will be able to put your programming knowledge to use by implementing iterative techniques for solving equations or finding extrema. Chapters 4, 5, 6, and 7 give some of the building blocks needed to study some of the most common mathematical models.

This text is meant to encourage an active learning approach. Although it contains discussions, theory, and a few examples it does require readers to work out problems on their own in order to fully grasp the concepts. The activities and exercises in Appendix A are designed for this purpose. Appendix A is essentially a workbook that can be printed to be worked out with pencil, as well as give practice with MATLAB. The items labeled as "Activities" are longer exercises that provide the reader with step-by-step guidance. These are meant to give readers practice as well as encourage them to discover important results. The items labeled "Exercises" are meant to give additional practice - these are similar to the exercises in the back of each chapter.

# Acknowledgements

Both authors were granted sabbatical leave to work on this project from their respective universities. We sincerely thank Viterbo University and the University of Alaska Southeast for providing these opportunities and for supporting this work in particular.



# Chapter 1

## Introduction

### 1.1 What is Modeling?

Mathematical modeling refers to the process of using mathematics to solve problems. In earlier math classes, you may have referred to this as “word problems”, or “application problems”. These problems can give insight as to how specific math concepts are utilized. A more challenging situation arises when a real-life problem must be solved without any or minimal mathematical context provided. This requires you to decide on a strategy for analyzing the problem, and putting into a mathematical context. This is all part of the mathematical modeling process.

Below is a list of objectives to consider during the math modeling process.

- Determine the objective of the problem.
- Determine which variables or parameters are important in the model, and which variables may be neglected. Determine which variables affect the system (independent variables), and which variables we want to know about (dependent variables).
- Determine the assumptions, simplifications, and approximations.
- Formulate the mathematical model. The formulation may include equations, diagrams, tables, graphs, or some other process.
- Use mathematical theorems and techniques to make conclusions and predictions.
- Test and refine the model.
- Using typical data, check the solution against common sense.

We should consider the balance between the simplicity or sophistication of a model and the objective of the problem. If a model is too simple, it may not contain enough information to accurately describe the phenomena. We may need to consider other variables, allow for more variability in the parameters, or use fewer assumptions. On the other hand, if a model is too complicated and has too many parameters, it may be impractical or not useful. We may end up needing to ignore or combine variables, use more assumptions, or restrict the class of problems that the model is designed to solve.

#### 1.1.1 Types of Mathematical Models

Mathematical models can generally be divided into two types: deterministic and stochastic models. In a **deterministic model**, all parameters are assumed to be known exactly, and there is no uncertainty about

the results once parameter values and initial conditions are known. Most of the models you have worked with so far have likely been deterministic. Unlike deterministic models, **stochastic models** take uncertainty into account. These models accommodate parameters whose values are not exactly known, and describe the likelihood of certain events occurring. Stochastic models tend to be more complex to analyze and often require computational algorithms to implement. However, the use of stochastic models is becoming more prevalent with the increase in computational power.

Related to stochastic models, **simulation models** are useful for studying phenomena with a large amount of detail that would otherwise be difficult to study using mathematical tools. These are implemented using computers to simulate the situation using a wide variety of parameter values and reviewing the resulting output with statistical analyses.

In this textbook, deterministic models are covered in Chapters 3, 4, 5, and 6, while stochastic and simulation models are covered in Chapter 7.

## 1.2 Modeling with Equations

In your earlier math courses you have been exposed to models of real-life phenomenon using equations. For example, Distance = Rate · Time is a simple model that is a basis for many word problems, not to mention practical for daily commutes. In this chapter we will look at a few models that can be described using equations. Refer back to each step in the modeling process as you consider each one.

---

### Example 1.2.1

Suppose we know the longer a bee spends at a flower eating nectar, the more slowly the nectar comes out. We want to determine an equation to model the rate at which the bee consumes the nectar, then use it to determine an optimal strategy the bee might use along its nectar gathering process among several flowers over a given feeding period. Whether or not it is accurate, a simplifying assumption for this problem is to assume all things to be equal in terms of the flowers the bee gathers nectar from. For example, the access to and amount of nectar in each flower is the same, the time to fly and the distance between the flowers is the same, etc. We might also assume a closed system over the time duration of this model, wherein the bee has no distractions or predation which might cause it to divert from its objective of collecting nectar.

We begin with a function  $F(t)$  giving the proportion of nectar that is eaten  $t$  seconds after the bee has arrived at a single flower. Based on the properties of the function and the observed behavior of the bee, we suppose a biologist estimates  $F(t) = \frac{t}{t+0.5}$ .

Notice that for this function,  $F(0) = 0$  as we would expect; no nectar would be consumed when a bee first arrives at a flower. Given that  $F'(t) > 0$  and  $F''(t) < 0$ , we can see  $F(t)$  is an increasing function but the rate of increase goes down as  $t$  gets large. This matches our assumption that as the bee spends more time at a particular flower, the nectar comes out more slowly. Also notice that  $F(t)$  is never equal to 1, however  $\lim_{t \rightarrow \infty} F(t) = 1$ . This says that if the bee spent eternity (or at least the lifetime of the flower) at a single flower, the bee would consume 100% of its nectar.

Since the rate of  $F(t)$  is decreasing, it may be advantageous for the bee to travel to the next flower after it has eaten the easy-to-reach nectar from each flower. On the other hand, the bee must expend time and energy to travel between flowers. We suppose  $\tau$  is the amount of time spent traveling between two flowers, and  $t$  is the amount of time spent at each flower. We want to find the value of  $t$  that maximizes the overall rate of nectar collection for a given feeding period.

The rate at which nectar is collected can be written as

$$\begin{aligned} r(t) &= \frac{\text{Total nectar consumed}}{\text{Total time spent}} \\ &= \frac{\text{Proportion of nectar consumed per flower} \cdot \text{Number of flowers visited}}{\text{Time spent at each flower} \cdot \text{Number of flowers visited}} \\ &= \frac{\text{Proportion of nectar consumed per flower}}{\text{Time spent at each flower}} = \frac{F(t)}{t + \tau}, \end{aligned}$$

where time spent at each flower includes the average travel time between flowers,  $\tau$ . Using the definition of  $F(t)$ , the rate of nectar collection simplifies to

$$r(t) = \frac{t}{(t + .5)(t + \tau)},$$

measured as a proportion of nectar per second.

Let's take a moment to clarify what we mean by *total nectar consumed*. As an example, suppose a bee hypothetically consumes 80% of the nectar in a flower as given by the proportion  $F(t)$ . If the bee then visits 10 flowers, it has consumed the equivalent of 100% of the nectar from  $0.80 \cdot 10 = 8$  flowers. In this way, *total nectar consumed* is really *total flower equivalents of nectar consumed* and the proportion  $F(t)$  is still the proportion of nectar from a given flower.

Before we continue, note that maximizing the rate of nectar *proportion* consumed must occur at the same time that the total *amount* of nectar consumed is maximized.

To find the optimal value of  $t$  that maximizes  $r(t) = \frac{t}{(t + .5)(t + \tau)}$ , we can use techniques from calculus since  $r$  is a differentiable function. Solving  $r'(t) = 0$  results in  $\frac{0.5\tau - t^2}{[(t + .5)(t + \tau)]^2} = 0$ , which gives  $t = \sqrt{.5\tau}$ . We also see that  $r'(t) > 0$  when  $t < \sqrt{.5\tau}$  and  $r'(t) < 0$  when  $t > \sqrt{.5\tau}$ . By the first derivative test, we know that  $r(t)$  has a maximum value when  $t = \sqrt{.5\tau}$ .

For example, if a bee takes  $\tau = 1$  second to travel between flowers, then it should spend  $t = \sqrt{0.5} \approx 0.71$  seconds at each flower in order to optimize the rate of collecting nectar. Flowers spaced further apart and requiring more travel time between them would increase the time a bee would spend at each flower to optimize its rate of nectar collection. See the graphs below.

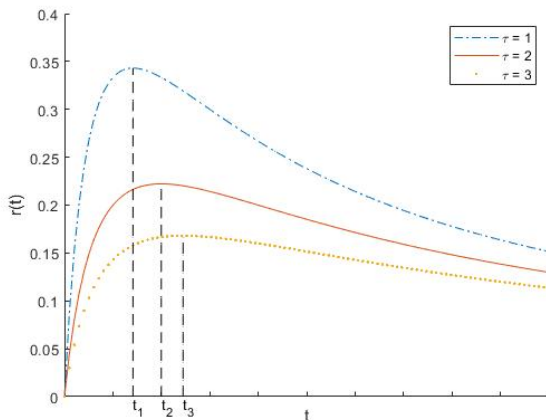


Figure 1.1: Plots of  $r(t)$  vs.  $t$  for  $\tau = 1, 2, 3$  with maximum rates at  $t_1 = \sqrt{0.5\tau_1}$ ,  $t_2 = \sqrt{0.5\tau_2}$ ,  $t_3 = \sqrt{0.5\tau_3}$ , respectively.

---

Refer to [Exercise A.0.1](#) for another example.

## 1.3 Modeling with Recurrence Relations

We often have an idea of how one quantity relates to other quantities at previous points in time. For example, the number of bobcats in a region for a given year may be 20% more than the number of bobcats in the previous year. The temperature of a metal plate may be  $1^\circ$  warmer than it was the minute before. Equations describing this relationship using discrete time increments are called recurrence relations.

To see this type of model in more detail, we suppose a patient is prescribed a particular medication. At the same time each day the patient's body absorbs half of the medication in his or her bloodstream. At the same time the medication is measured, the patient is also given a new dose to raise the medication concentration by 1 mg per liter. We let  $M_t$  be the concentration at day  $t$  and incorporate the above assumptions. The concentration the next day is then

$$M_{t+1} = 0.5M_t + 1. \quad (1.1)$$

The equation (1.1) is called a **recurrence relation**, and the sequence  $M_0, M_1, M_2, \dots$  generated by (1.1) is a **recursively defined sequence**. If we know the concentration at time  $t$ , the equation tells us the concentration at time  $t + 1$ . For example, if  $M_0 = 0$ , then

$$\begin{aligned} M_1 &= 0.5M_0 + 1 = 1, \\ M_2 &= 0.5M_1 + 1 = 0.5(1) + 1 = 1.5, \\ M_3 &= 0.5M_2 + 1 = 0.5(1.5) + 1 = 1.75, \end{aligned}$$

and so on. Sometimes (1.1) is referred to as a **difference equation**, but some authors prefer to use this term for sequences of the form  $x_{n+1} - x_n = f(n)$ . In both cases, the equation may also be referred to as a **discrete model** because the population (or any quantity of interest) is updated on regular time intervals.

### Example 1.3.1

A mathematical model to describe the amount of money in a savings account with fixed interest rate  $r$  is given by

$$P_{n+1} = P_n + rP_n. \quad (1.2)$$

Here  $P_n$  is the quantity of interest, a variable representing the amount in the account after the interest is compounded  $n$  times, and  $r$  is parameter giving the interest rate during the compounding period. For example, if the annual interest rate is 3.0% and the interest is compounded monthly (for a compounding period of 1 month), then the interest rate is  $r = 0.03/12 = 0.0025$ .

The equation (1.2) is a recurrence relation, however we can find a formula that gives  $P_n$  in terms of  $n, r$ , and  $P_0$ . First, write (1.2) as  $P_{n+1} = P_n(1 + r)$ . Next notice that

$$\begin{aligned} P_1 &= P_0(1 + r), \\ P_2 &= P_1(1 + r) = P_0(1 + r)(1 + r), \\ P_3 &= P_2(1 + r) = P_0(1 + r)(1 + r)(1 + r). \end{aligned}$$

Repeating this logic, we might conclude that

$$P_n = P_0(1 + r)^n. \quad (1.3)$$

Mathematical induction proves the relationship holds for all  $n \geq 0$ . Since this new formula (1.3) does not rely on knowing the previous values of  $P_n$  (other than  $P_0$ ) we say it is an **explicit formula**. The formula should be familiar to you if you have studied compound interest.

See **Activity A.0.2** for another example of a recursively defined sequence.

Now suppose we have two quantities that change with respect to time. If the two quantities depend on each other, this results in a system of recurrence relations. One example of this is the predator-prey model:

$$\begin{aligned}x_{n+1} &= -ay_n + bx_n \\ y_{n+1} &= cy_n + dx_n,\end{aligned}$$

where  $x_n$  is the size of the prey population and  $y_n$  is the size of the predator population at the  $n^{\text{th}}$  time step. This model has four parameters:  $a, b, c$  and  $d$ . The parameter  $a$  represents the rate at which the prey population is decreasing due to predation and  $b$  represents the natural growth rate of the prey population in the absence of predators. The parameter  $c$  represents the growth of the predator population due to predation and  $d$  represents the natural growth rate of the predator.

We will study recurrence relations in detail in Chapter 5.

## 1.4 Modeling with Differential Equations

Differential equations are similar to difference equations, except we update the model continuously in time. Since the derivative describes the rate of change of a particular quantity, resulting models often take on the form of differential equations.

For example, differential equations can be used to model disease epidemics. Consider the spread of a disease in a population of individuals. Let  $S(t)$  be the number of individuals at time  $t$  who are susceptible to the disease but are not infected, and let  $I(t)$  be the number of individuals who are infected with the disease at time  $t$  and can spread it to others. We can model the spread of the disease with the differential equations

$$\begin{aligned}\frac{dS}{dt} &= rI - cSI, \\ \frac{dI}{dt} &= cSI - rI.\end{aligned}\tag{1.4}$$

There are two parameters in this model:  $r$  and  $c$ , both positive values. The parameter  $r$  represents the rate at which infected people recover from the disease, and  $c$  is proportional to the probability that a susceptible individual becomes infected after interacting with an infected individual, also called the **law of mass action**. The *law of mass action* assumes all individuals mix randomly and interact with each other with equal likelihood. This can be a fairly accurate assumption when talking about the intermixing of liquid or air molecules. When modeling disease spread in people or animals, mass action is a good initial assumption to simplify the mathematics or when considering a particular disease in which an infected person does not change their behavior as a result of infection.

It is always a good idea to make sure the units are consistent when applying a model. If  $t$  is measured in days, notice that  $dS/dt$  and  $dI/dt$  are measured in persons per day. Thus, the units on the right-hand sides of (1.4) must also be persons per day. Thus, the units of  $r$  has to be  $\frac{1}{\text{day}}$ , and the units of  $c$  must be  $\frac{1}{\text{person}\cdot\text{day}}$ .

See [Exercise A.0.3](#) for an example of a continuous predator-prey interaction model.

Note that a differential equation model can often be converted into a difference equation model and vice-versa. Each of the two types has advantages and disadvantages. We will study differential equation models in detail in Chapter 6.

### Example 1.4.1

The equation

$$\frac{dQ}{dt} = kQ + b\tag{1.5}$$

is used to model exponential growth (if  $k > 0$ ) or exponential decay (if  $k < 0$ ) of the quantity  $Q$  over time. In a Calculus course, you have likely solved the simpler version in which  $b = 0$ ,

$$\frac{dy}{dt} = ky,\tag{1.6}$$

by using separation of variables to get the solution

$$y(t) = y(0)e^{kt}. \quad (1.7)$$

We can quickly verify that (1.7) does indeed solve (1.6). Differentiating (1.7) we get  $y'(t) = ky(0)e^{kt} = ky$ , which satisfies (1.6).

To solve (1.5), we first define a new variable  $y(t) = kQ(t) + b$ , then differentiate and substitute to give

$$\frac{dy}{dt} = k \frac{dQ}{dt} = k(kQ + b) = ky.$$

We already saw that the solution to  $\frac{dy}{dt} = ky$  is  $y(t) = y(0)e^{kt}$ . Since  $y(t) = kQ(t) + b$ , substituting gives

$$y(t) = kQ(t) + b = y(0)e^{kt} = (kQ(0) + b)e^{kt}.$$

Solving for  $Q(t)$  gives

$$Q(t) = \left(Q(0) + \frac{b}{k}\right)e^{kt} - \frac{b}{k}. \quad (1.8)$$

Notice that if  $t = 0$ , we get  $Q(t) = Q(0)$  as expected. Also, if  $k < 0$  then  $\lim_{t \rightarrow \infty} Q(t) = -b/k$ . For example if  $Q(t)$  represents the population at time  $t$  then the population will eventually approach  $-b/k$ , a positive value.

See [Exercise A.0.4](#) for an example of an exponential decay model.

## 1.5 Stochastic Models

Stochastic models are often used as an additional level of complexity to account for variability in parameter values. For example, the amount of money in a mutual fund account is governed by a variable interest rate which may be affected by various economic patterns or indicators. We may choose to model this with the equation

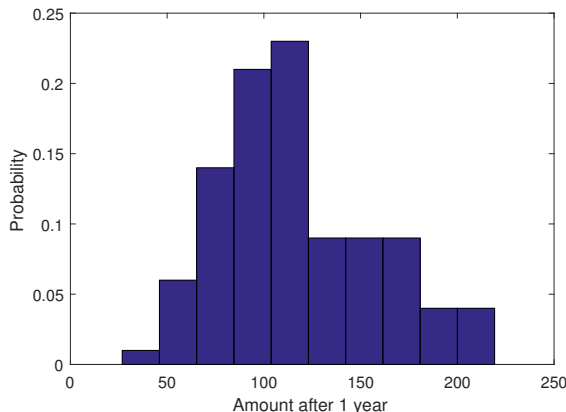
$$P_{n+1} = P_n(1 + r(n)). \quad (1.9)$$

As in Example 1.3.1,  $P_n$  is the amount in the account at month  $n$ , but now  $r(n)$  is a variable interest rate during month  $n$ . We assume the rate  $r(n)$  varies randomly but has some known probability distribution. A simulation model could be used to determine the long term behavior of the account.

For example, suppose we want to know how much money will be in the account after one year. A single simulation would involve using a random number generator to find the monthly rates  $r(n)$  and then apply (1.9) for 12 months. We would then repeat this simulation many times, each time getting potentially different values for  $r(n)$  and a different estimate of the amount of money in the account after one year. We may visualize the results using a histogram like the one shown in Figure 1.2. We will study models like this in Chapter 7.

## 1.6 Exercises

- Suppose the number of yeast cells in a laboratory culture is modeled by the function  $f(t) = \frac{a}{b + ce^{-kt}}$ , where  $t$  is measured in hours, and  $a, b, c$  and  $k$  are positive constants.
  - Find the rate at which the number of yeast cells are changing at time  $t$ , in terms of parameters  $a, b, c, k$ . Include your units.
  - For which values of  $t$  is  $f(t)$  increasing? Explain.

Figure 1.2: Distribution of  $P_{12}$  from simulation

- (c) When is the population of cells increasing at its fastest rate?
- (d) According to this model, what happens to the size of the yeast population in the long run?
2. A bird is searching through bushes in a field for insects, and the total weight of insects found after  $t$  minutes of searching a single bush is given by  $w(t) = \frac{3t}{2t+7}$  grams. It takes the bird one minute to move from one bush to another. How long should a bird search a bush before moving to the next bush in order to maximize its food intake?
3. Suppose a forager consumes a food resource at the rate  $r(x) = \frac{ax}{1+ahx}$ , where  $x$  is the density of the resource available. This is an example of a Holling Type II functional response model which includes a necessary feeding time that decreases the time a forager can look for additional food. Think of a bear preying on a moose. The bear must take time to eat the moose, time it cannot spend foraging for other items such as berries, barnacles, etc. Here  $h$  is called the handling time (the average time required to consume a resource), and  $a$  is the attack rate (the rate at which the forager encounters the resource per unit of food density).
- (a) What is  $\lim_{x \rightarrow \infty} r(x)$ ? Interpret this in context of the forager.
- (b) Suppose a forager decides to stop if the rate of consumption is half of the attack rate. Determine the density  $x$  at which this occurs.
- (c) Now consider the time window in which a bear begins feeding on barnacles on a beach. The time spent licking up the barnacles is minimal compared to the “attack” rate which includes time walking between rocks and crushing barnacles. What does the consumption rate reduce to when handling time is zero? This is referred to as a Holling Type I functional response.
4. (Adapted from [8]) We examine a circular growth of mold (called the ‘mother’) on a solution of tea and sugar and find that each day the increase in area of the colony is proportional to the circumference of the colony the previous day. Assume that the thickness of the mold does not significantly change, so that the amount of mold can be measured in terms of area.
- (a) Find a recurrence relation for  $A_t$ , the amount of mold at day  $t$ .
- (b) Suppose that at day 0, the area is  $4 \text{ mm}^2$ , and at day 1, the area is  $8 \text{ mm}^2$ . Use this to solve for the unknown constants in the model.
- (c) Find the area of the mold on day 8.
5. A recent college graduate has a loan of \$20,000 with an annual interest rate of 5.5% compounded monthly. Each month she pays \$300 towards the loan after the interest is calculated.

- (a) Find a recurrence relation for the amount of the loan  $P_t$  at month  $t$  from now.
- (b) Find the amount of the loan at months  $t = 1, t = 2$ , and  $t = 3$ .
6. A lake of area  $3 \text{ km}^2$  and average depth of 15 meters has a river flowing into it at a rate of  $8000 \text{ m}^3$  per day. Suppose that a nearby factory releases 120 kg of chemical waste into the lake each day.
- (a) Determine a mathematical model that gives the amount of waste (in kg) in the lake on day  $t$ .
- (b) Determine a mathematical model that gives the concentration of waste (in  $\text{kg}/\text{m}^3$ ) in the lake on day  $t$ .
7. The Gompertz population growth and mortality model is given by  $P(t) = Ke^{-ce^{-bt}}$ , where  $K, c$ , and  $b$  are positive constants, and  $P(t)$  is the population at time  $t$ .
- (a) Find  $\lim_{t \rightarrow \infty} P(t)$ .
- (b) Show that  $P(t)$  satisfies the differential equation  $\frac{dP}{dt} = \alpha P \ln(K/P)$ , and specify how  $\alpha$  relates to  $K, c$ , and  $b$ .
8. Newton's Law of Cooling is based on the assumption that the rate of change of an object's temperature is proportional to the difference between its own temperature and the surrounding temperature. If we let  $T(t)$  represent the temperature of an object as a function of time, then  $\frac{dT}{dt}$  represents the rate at which that temperature changes. Newton's law of cooling can be written as

$$\frac{dT}{dt} = \alpha(T(t) - T_s),$$

where  $T_s$  is the surrounding temperature and  $\alpha$  is the proportionality constant related to how well insulated the object is.

- (a) Solve for  $T(t)$  in terms of  $t$  and  $T_s$  using separation of variables.
- (b) Suppose a cup of coffee has a temperature of  $160^\circ \text{ F}$  and you set it in a room with a temperature of  $70^\circ \text{ F}$ . After 10 minutes the temperature is  $150^\circ \text{ F}$ . Use your results from part (a) to find an equation giving the heat  $T(t)$  at time  $t$ .
9. Suppose that leaves accumulate on the forest floor at a rate of  $3 \text{ g}/\text{cm}^2/\text{year}$ . The leaves then decompose at a rate of 50% per year.
- (a) Write a differential equation governing the density of leaf litter in grams per square centimeter of forest floor. Assume an initial density of  $L_0 \text{ g}/\text{cm}^2$ .
- (b) Solve the differential equation.
- (c) Find the eventual density of leaves - that is, after a long time what does the density of leaves approach?
10. A person deposits \$25,000 in a bank that pays 5% per year interest, compounded continuously. The person continuously withdraws from the account at the rate of \$750 per year.
- (a) Using the assumption of continuous rates, set up a differential equation to model  $V(t)$ , the value of the account at time  $t$  after the initial deposit.
- (b) Solve for  $V(t)$ .
11. Assume an initial nutrient amount of  $Q_0$  kilograms in a tank with  $V$  liters of water. Assume that water is being pumped into the tank at a rate of  $r \text{ L}/\text{min}$  and that this water contains the nutrient with a concentration of  $c \text{ kg}/\text{L}$ . The tank is well mixed and is drained at a rate of  $r \text{ L}/\text{min}$ .
- (a) Find a differential equation describing the amount of nutrient in the tank,  $Q$ , in kg at time  $t$ . Remember to verify that your units match on both sides of the equation.
- (b) Solve the differential equation in terms of the parameters given above.



- (c) Find the eventual concentration of nutrients in the tank if the pump and drain operate indefinitely.
12. Newton's Law of Gravitation says that the force between two objects with masses  $m_1$  and  $m_2$  separated by distance  $r$  (as measured from the centers of each mass) is

$$F = G \frac{m_1 m_2}{r^2},$$

where  $G \approx 6.674 \cdot 10^{-11} \text{Nm}^2/\text{kg}^2$ . (Note: if the object is 'close' to Earth's surface, we may assume that the gravitational force is constant and equal to its value at the surface. In this case, the formula  $F = mg$  holds where  $g = 9.8 \text{m/sec}^2$  and  $F$  measures the person's weight due to gravity.)

How high does a person need to travel above the earth's surface in order to cut their weight by half? (Assume the earth's radius is 3960 miles.)

13. (Adapted from [1]) Many bees collect both pollen and nectar. Suppose the proportion of nectar harvested during  $t$  seconds on a flower is  $F(t) = t/(1+t)$  and the proportion of pollen harvested during  $t$  seconds on a flower is  $G(t) = t/(2+t)$ . We assume the bee collects pollen and nectar simultaneously, and the travel time between flowers is 1 second.
- (a) Find the optimal time for the bee to stay on each flower in order to collect nectar at the maximum rate.
- (b) Find the optimal time for the bee to stay on each flower to collect pollen at the maximum rate.
- (c) Suppose a bee values pollen twice as much as nectar. Find a single function  $V(t)$  that gives the value of resources collected by time  $t$  on a given flower. What is the optimal time spent on each flower? You may use a computer solver to determine the answer.
14. In a fish farm, a population of fish is introduced into a pond and harvested regularly. A model for the rate of change of the fish population is given by

$$dP/dt = r_0(1 - P/L)P - \beta P,$$

where  $P(t)$  is the population of fish at time  $t$ ,  $r_0$  is the birth rate of the fish,  $L$  is the maximum population of fish the pond habitat will sustain, and  $\beta$  is the proportion of the fish population that is harvested per unit time. Answer each of the following questions to better understand the model building phase of the mathematical modeling process.

- (a) How does increasing  $r_0$  affect the rate of change of population?
- (b) How does increasing  $\beta$  affect the rate of change of population?
- (c) What happens to  $dP/dt$  if there are no fish in the pond?
- (d) What happens to  $dP/dt$  if the fish population is equal to  $L$ ?
- (e) If  $\beta = 0$ , then for what population is the population increasing the fastest?
- (f) Find the stable population level in terms of parameters  $r_0$ ,  $L$  and  $\beta$  - that is, for what value of  $P$  is the population no longer changing? (Hint: if the population is no longer changing, what is  $dP/dt$ ?)
15. Suppose a drug administered to a patient produces a concentration in the bloodstream given by  $c(t) = Ate^{-0.4t}$  milligrams per milliliters,  $t$  hours after  $A$  units have been injected. The maximum safe concentration is 0.8 mg/ml.
- (a) Determine when the maximum concentration occurs. Then find how many units should be injected to reach this maximum safe concentration.
- (b) After the maximum concentration is reached, an additional amount of the drug is administered after the concentration falls to 0.5 mg/ml. Determine an approximate time when this second injection should be given.

16. Suppose a person has been taking 50 mg doses of a painkiller every 6 hours for several days. For this problem, we assume that a consistent (equilibrium) amount of the drug now remains in the bloodstream after every dose. Let  $A$  be this unknown quantity of the drug immediately after taking a dose. We will use the drug clearance model,  $f(t) = Ae^{-kt}$  to model the decay of the drug remaining in the bloodstream as the body processes it. Let  $k = 0.14$  and define  $f(t)$  as the amount of the drug (in mg) in the blood  $t$  hours after taking a dose.
- (a) Obtain an expression in terms of  $A$  that gives the amount of drug present after 6 hours, immediately before the next dose.
  - (b) Use the result from part (a) to obtain an expression for the amount present immediately after the next dose. [Hint: don't forget that the amount of each dose is 50 mg.]
  - (c) Because the amount of drug in the body has reached an equilibrium amount after every dose, the amount of drug in the blood will be periodic as it repeats every 6 hours. Therefore, the amount present immediately after that next dose (from part (b)) must also be equal to  $A$ . Use this fact to calculate the value of  $A$  and to determine the minimum amount present during the 6-hour period.
  - (d) Sketch a plot showing the concentration of the drug versus time for a 24-hour period. Note that we are assuming the patient has been taking the drug for several days already.
  - (e) Now suppose the patient needs to keep the amount no lower than 45 mg. How often would the 50 mg dose have to be administered?
  - (f) Repeat part (c) to find  $A$  with doses of  $D$  mg of a drug with rate constant  $k$  taken every  $T$  hours. Again, assume the equilibrium amount  $A$  has been reached.

## Chapter 2

# Programming in MATLAB

### 2.1 Why is Programming Important?

As mentioned previously, many models that approach reality eventually need the aid of a computer to solve. In this chapter we develop some basic tools to program in MATLAB, which we will build upon in successive chapters as our mathematical techniques expand.

An **algorithm** is a set of instructions that is used to complete some task. In mathematics, this task could be something like performing an arithmetic operation, solving an equation, or simulating a process. At an early age you learned algorithms used to do addition, subtraction, multiplication, and division. Later on, you learned more complicated algorithms to do things such as find the least common divisor or find the extreme values of a function. You have probably used calculators or computers to do these operations as well. At some point somebody had to program the computer or calculator to perform these tasks by providing the algorithm in a way that the computer can understand. As we study more complicated tasks, it is important to learn how to program a computer to do the tasks that are too time consuming to complete by hand.

The rules of the algorithm are determined by applying mathematical rules or theorems in a certain order. Applying algorithms are important in learning and appreciating basic mathematics. While this foundational learning often requires slow and methodical exploration with pencil and paper, computers are also very useful tools in a mathematician's toolbox. In many cases we want to understand how to solve the problem, but we leave the actual process of solving it to a computer. It is therefore important to understand how these computers are able to solve these problems, and to learn how to write your own computer programs to solve problems. Moreover, mathematicians and engineers are always coming up with new algorithms to solve new problems, or to solve old problems in a more efficient manner.

As an example, suppose you are asked to solve  $x^7 + x = 7$ . This problem has no algebraic solution, at least not one that is easy to express. You might try graphing the left-hand side and seeing where it crosses the line  $y = 7$ . Because the left-hand side is an odd-degree polynomial with end behavior at plus and minus infinity, you know it must have at least one solution. Indeed, graphing does give you the decimal approximation for a single solution. Solving this problem involved technology, as well as a little bit of theory.

Often, mathematicians must determine a method for which a computer may be used to approximate the solution to a problem. For example, in chapter 3, we examine some simple methods to approximate solutions to equations of the form  $f(x) = 0$ .

MATLAB is a great first programming language for students who have studied mathematics at the university level, and it works particularly well for mathematical modeling. In fact MATLAB is probably the most common language among engineers, physicists, and mathematicians. It is beneficial to learn MATLAB since there is a good chance you will end up using it again if you pursue further studies or a career in engineering, physics, or mathematics. Moreover, you may find that once you have learned one programming language it is much easier to pick up other programming languages. GNU Octave is a free online program that has many similar features to MATLAB.

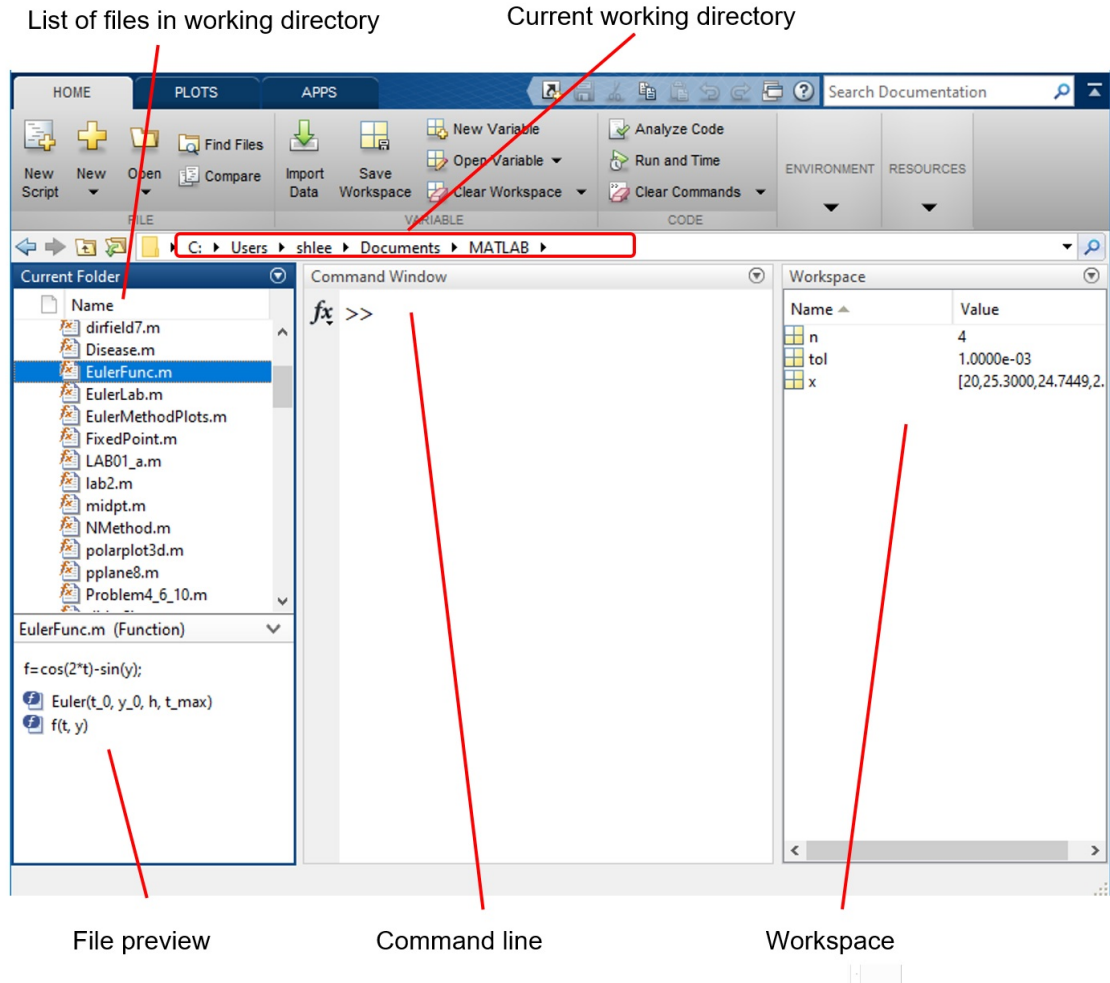


Figure 2.1: Typical MATLAB Window

## 2.2 MATLAB Basics

When you open the MATLAB program you will typically see several panels, as shown in Figure 2.1. In the Command Window, a user may enter any MATLAB command into the command line. To enter a command into the Command Window, type the appropriate command, then hit the enter key to execute it. There are two ways to get MATLAB to do something. You may enter commands directly into the Command Window, or you may enter a list of commands into a MATLAB M-file. We will be using the Command Window for now.

### Variables and Assignment Statements

You may define a variable by using an **assignment statement**. For example, the command

```
x=3
```

is a MATLAB assignment statement that stores the number 3 into the variable x for later use. MATLAB prints the value of x to the command window and in the Workspace. The Workspace (see Figure 2.1) displays all current variables shown in memory. You should see the variable x appear with a value of 3.

You may perform operations on numbers (use + for addition, - for subtraction, \* for multiplication, and / for division).

**You must use the asterisk symbol (\*) for multiplication!**

Even though it looks like `x` is simply a number, MATLAB views it as a matrix with one column and one row. A **matrix** is a rectangular array of numbers. A single number, or **scalar**, is stored as a  $1 \times 1$  matrix, and a **vector** is stored as a matrix with either a single row or a single column.

In this textbook, a **vector** refers to an ordered list of numbers, often in the form  $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$ , but to

save vertical space we will often use the notation  $\mathbf{v} = (v_1, v_2, \dots, v_n)^T$ , where  $T$  represents the **transpose operator**, which swaps the rows and columns of a matrix. A **row vector** is a matrix with 1 row and any number of columns - that is, it is a matrix of size  $1 \times n$  such as this  $1 \times 4$  vector: `[ 43 0 7 pi ]`. From the MATLAB command line, use `R = [43 0 7 pi]` to define a row vector. You may also use commas instead of spaces to separate elements such as `R = [43, 0, 7, pi]`. To display this row vector as a column vector, include an apostrophe after the row vector such as in this command `R'`. To reference specific elements in the vector, use the command `R(n)`, for example `R(3)` should return the 3rd element of the vector, the number 7.

Use the `size` command to determine the dimension of a vector or matrix. Since MATLAB considers `R` to be a  $1 \times 4$  matrix, entering `size(R)` returns “1 4”, meaning that `R` is a matrix with 1 row and 4 columns. We may also use the command `[x,y] = size(R)` to assign the number of rows of `R` as `x` and the number of columns of `R` as `y`.

A **column vector** is a matrix with 1 column and any number of rows - in other words, it is a matrix of size  $n \times 1$ , for example the  $3 \times 1$  vector  $\begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix}$ , which we may denote using  $(1, 4, 7)^T$ . To define a column vector, use semicolons in between the elements: `C = [1; 4; 7]`. By calling `size(C)`, we see that MATLAB is considering this to be a  $3 \times 1$  matrix. To reference specific values of the index, we generally use the same syntax we used for row vectors - for example `C(2)` should return the 2nd element of the vector, the number 4.

It often does not matter whether we use a column or row vector in MATLAB - however keep in mind that column vectors are sometimes easier to display since the command window scrolls vertically and not horizontally. The transpose operator may be used to display a row vector as a column vector or visa versa. Again, this is done in MATLAB with the single quote symbol, `'`. As described above, the transpose operator is used to swap the rows and columns of a matrix.

`R'`

To define an  $m \times n$  matrix  $A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \vdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$ , use

`A = [a11, a12, ..., a1n; a21, a22, ..., a2n; ... ; am1, am2, ..., amn]`

Again, you may use a space instead of a comma to separate columns, but a semicolon must be used to end a row.

MATLAB command	What it does
<code>A(r,c)</code>	Output $A_{rc}$ (row $r$ and column $c$ of $A$ )
<code>A(r,:)</code>	Output the row $r$ of $A$
<code>A(:,c)</code>	Output the column $c$ of $A$
<code>rref(A)</code>	Output the reduced row Echelon form of $A$
<code>size(A)</code>	Output the size of $A$ , in the form: [number of rows    number of columns]
<code>2*A</code>	Multiplies each element of $A$ by 2 to find $2A$ .
<code>A.*B</code>	Multiplies each element of $A$ by the corresponding element of $B$
<code>A'</code>	Output the transpose of $A$ (interchange rows with columns).

A **logical expression**, or **boolean expression** is a statement that is either true or false. In MATLAB (and most other computer languages), a value of true is coded as 1 and false is coded as 0.

See [Activity A.0.5](#) to get practice working with MATLAB variables and assignment statements.

## Sequences and Linspace

There are several ways to make basic sequences of numbers. Suppose we want a list of numbers from  $a$  to  $b$  in increments of size `inc` - we could use `a:inc:b`. For example, use `x = 1:0.25:3` to assign a list of numbers from 1 to 3, in increments of 0.25, to a variable called `x`. If we instead wanted a list of `n` equally spaced numbers from `a` to `b`, we could use `linspace(a, b, n)`. For example, try entering `linspace(1, 3, 8)`. Keep in mind that `linspace` generates a row vector - simply use the `'` symbol if you prefer a column vector.

## Rounding Numbers

To round numbers to the nearest whole number, use the `round` command. For example, `round(pi)` returns 3. If you want to round down to the next smallest whole number, use the `floor` command. For example, `floor(1.9999)` returns 1. To round up to the next largest whole number, using the `ceil` command (ceiling). For example, `ceil(pi)` returns 4.

You can change the default number of digits by using `format`. Use `help format` to view all of the various options. Try entering these lines:

```
format short
pi
format long
pi
```

If you call `format long` MATLAB will remember to use this format until you switch it with another `format` command.

## 2.3 MATLAB Functions and Terminology

Earlier in this chapter, we saw some examples of MATLAB's built-in **functions**. In MATLAB, we execute a function with a command like this:

```
functionName(list of arguments)
```

Programmers refer to executing a function as **calling** the function. The list of arguments is a list of inputs into the function. Sometimes these inputs are called parameters but in this text we will refer to them as **arguments**.

For example, we can call the `ceil()` function with `pi` as a single argument using `ceil(pi)`. The function **returns**, or outputs the number 4. Some functions may return multiple numbers or objects, although a function does not necessarily have to return anything once it completes the algorithm.

We can create our own functions that do whatever we program them to do, and we will get a chance to practice this in later chapters. The functions we have been using are said to be built-in because they come loaded in MATLAB. We can get help documentation for any built-in function by using `help` followed by the name of the function, for example:

```
help ceil
```

An example of a function with two arguments is `gcd`. This function takes two arguments and returns the greatest common divisor to the two numbers. For example,

```
gcd(16, 36)
```

returns the number 4 since 4 is the largest number that divides both 16 and 36.

There are other built-in MATLAB functions that return boolean values (true or false) - for example `isprime()` is a function that returns a 1 (true) if the number is prime and a 0 (false) otherwise. Try it:

```
isprime(3)
isprime(10)
```

## 2.4 Plotting Points and Curves

### 2.4.1 Using ezplot()

The easiest way to plot a curve in the  $xy$  plane is by using the `ezplot()` function. To plot a curve of the form  $y = f(x)$ , use `ezplot('f(x)')`. For example, try plotting  $y = x^2$  by using `ezplot('x^2')`.

If the curve is not an explicit function of  $x$ , convert it into the form  $f(x, y) = 0$ , then call `ezplot('f(x,y)')`. For example, to plot the circle  $x^2 + y^2 = 9$ , use `ezplot('x^2+y^2-9')`.

Recall from section 2.3 the `help` command displays documentation for any MATLAB command. For example, try entering `help ezplot` from the command line. Among other things, you will find commands needed to label the  $x$  and  $y$  axes.

See [Activity A.0.6](#) for practice using `ezplot`.

### 2.4.2 Using plot()

The alternative to using `ezplot` is the `plot()` function. Its basic form is `plot(x, y)`, where `x` and `y` are vectors of equal sizes. MATLAB will plot each ordered pair  $(x_i, y_i)$  and connect successive points with a straight line.

#### Example 2.4.1

To plot the function  $y = x^2$  on the interval  $[-2, 3]$ , we first define a vector for the  $x$ -values:

```
x = linspace(-2,3,100)
```

This command creates a vector `x` with 100 equally spaced values, from -2 to 3. In general, you want to use enough points so that the plot appears smooth. This will vary depending on the nature of the function that you are plotting. Next we can define the  $y$ -values so that  $y = x^2$ . However, the command `y = x^2` will result in an error. The reason is that MATLAB interprets `x` as a matrix and is attempting to apply matrix multiplication to calculate `xx`, which is not defined. Instead, we actually want to apply  $x_i^2$  for each element  $x_i$  in `x`. To tell MATLAB to apply the operation to each element, use the dot (`.`) before the operation, whether it be `+`, `-`, `^`, or `/`.

```
y = x.^2
```

We are ready to plot the function:

```
plot(x,y)
```

We can label the axes and give it a title using:

```
xlabel('x')
ylabel('y')
title('y=x^2')
```

---

See [Activity A.0.7](#) for additional practice using `plot`. See [Activity A.0.8](#) for practice with piecewise defined functions.

## 2.5 The Symbolic Toolkit

Your version of MATLAB may have one or more toolkits, each of which has additional functionality. One of these toolkits is the Symbolic Toolkit. This toolkit is useful for taking derivatives or integrals of functions, or simplifying algebraic expressions. However, we will not make use of the toolkit in this textbook. If you happen to have this toolkit installed and want to check out some of its capabilities, see [Activity A.0.9](#).

## 2.6 For Loops

Often, approximating the solution to mathematical problems is an iterative process. That is, a series of steps is repeated. The idea is that each step brings us closer to the true solution of the problem. We may instruct a computer to perform this iterative process by using different types of loops. The first of these is a `for` loop.

A **for loop** is used to implement an algorithm that contains some sort of iterative process or any process that is repeated a specified number of times. This process has a counting variable (sometimes called an **index**) that is increased by 1 at each step. During each step, a segment of code is run that often depend on this counting variable.

In MATLAB, the general form of the `for` loop looks like this:

```
For loop
for i=start:end
    statements
    :
end
```

where the index `i` iterates through the integers `start` to `end`. Note that the set of statements will be executed `end - start + 1` times, for the various values of `i`. As with other commands, you can use the help menu for more details, `help for`.



See [Activity A.0.10](#) for an example of using a `for` loop to generate values of a recursively defined sequence. You will also learn to set up a **script file** or **m-file** in this activity as well as the following example.

### Example 2.6.1

To compute a sum such as

$$\sum_{k=3}^{50} \frac{k^2}{e^k}, \quad (2.1)$$

we could use a `for` loop. For this example,  $k$  would be the index which would run from 3 to 50. Inside the `for` loop is a segment of code that adds the expression  $k^2/e^k$  to some variable that keeps track of the sum. The MATLAB code will look something like this:

```
s = 0;
for k = 3:50
    s = s + k^2/exp(k);
end
s
```

To make a MATLAB program that uses iteration, we will create a MATLAB M-file, instead of typing the commands into the command line. There are two types of MATLAB files - scripts and functions. In this section, we focus on making scripts. Referring back to Figure 2.1, notice the current working directory towards the top. This is the directory to where MATLAB files are saved. On the left is a list of files in the working directory. To change the working directory, you may click on the “Browse for folder” icon to the left of the file path.

On the line inside of the `for` loop for the code segment above, a semicolon is used in order to suppress output to the command window. If we were to remove the semicolons, you would see all the values of `s` output to the screen.

On a side note, you have to be somewhat careful about how you name your variables because things can go wrong if you use a name that is also the name of a MATLAB function. For example `sum` is a built-in function that takes the sum of a vector, and you may be tempted to use it in your code. If you use `sum` instead of `s`, then the `sum` function will essentially be disabled until the variable is removed from the workspace. By the way, you could actually use `sum` to find (2.1). You would use a couple lines of code like this:

```
x=3:50
sum(x.^2./exp(x))
```

This allows you to compute the sum without using a `for` loop. This does not mean that a `for` loop is not important, it just means that sometimes using a `for` loop can be avoided.

## Vectorizing code and preallocation

We could also use a `for` loop to generate values of a recursively defined sequence.

### Example 2.6.2

Consider the sequence  $x_{n+1} = 0.8x_n + 1, x_0 = 3$ . If we want to find  $x_1, x_2, \dots, x_{20}$  we could use the code segment

```
x=3;
for i=1:20
    x = .8*x+1
end
```

At each step in the loop, the old value of  $x$  is replaced by the next value,  $0.8x + 1$ . Another strategy would be to **vectorize** the code, meaning to keep track of all values of the sequence as a vector. The code would look like this:

```
x=3;
for i=1:20
    x(i+1) = .8*x(i)+1;
end
x'
```

All of the values are saved to the variable  $x$  instead of simply the most recent one. The last line displays all of the values of  $x_n$  in a single column. We can also make a plot with a single command:

```
plot(x)
```

This plots  $x$  against the index, which in this case runs 1 to 21. The indices are “off by one”, since we are actually plotting  $x_0$  to  $x_{20}$ . To remedy this, you could use `0:20` to generate the correct 21 indices, then make a plot using

```
plot(0:20, x)
```

Notice that on each iteration of the `for` loop, we append the next value of the sequence onto the vector  $x$ , increasing the size of  $x$  by 1. On each step of the loop, MATLAB is spending time assigning more memory to the variable  $x$  so that more information can be stored in it. This is not a problem for this example, but if we are performing millions of iterations, the time it takes to assign extra memory on each step can be significant. It is therefore generally preferable to set the size of the vector ahead of time. To do this, we will use the command

```
x = zeros(21, 1)
```

to define  $x$  as a vector of size 21. Keep in mind that the indices run  $0, 1, \dots, 20$  so there are actually 21 values to keep track of. In general, the command `zeros(n,m)` is used to create a matrix of size  $n \times m$  containing all zeros. This process of defining a matrix or vector this way is called **preallocation**, and can be used if one knows how large a vector or matrix will end up being. By preallocating a variable, MATLAB will reserve all of the computer memory space needed at the beginning, instead of squeezing larger and larger matrices into memory throughout the run of the program. Consequently, MATLAB spends less time allocating memory for the variable, and the code will run much more quickly. After preallocating  $x$ , the code will look like this:

```
x=zeros(21,1);
x(1)=3;
for i=1:20
    x(i+1) = .8*x(i)+1;
end
plot(0:20,x)
xlabel('n')
ylabel('x_n')
```

You would not notice a difference in speed between this code segment where  $x$  is preallocated and the previous code segment, since both versions of the code will seem to run instantaneously. However it could make a difference for code that requires more time to run.

See [Activity A.0.11](#) for another example of using a `for` loop to generate values of a recursively defined sequence.

An extension of the above example uses a nested `for` loop in order to define several recursive sequences for different growth rates, assigning each to rows of a matrix.

### Example 2.6.3

Consider the sequence  $x_{n+1} = r \cdot x_n + 1$ ,  $x_0 = 3$  for  $n = 1, 2, \dots, 20$ , where  $r$  is one of four values 0.6, 0.7, 0.8, 0.9. Below is the code to assign the sequence corresponding to the  $i^{\text{th}}$  rate,  $r(i)$ , to the  $i^{\text{th}}$  row of the `x` matrix, where the initial value of the sequence is assigned to each row of the first column of `x`. Notice how each sequence can be plotted by referencing each row of the matrix `x(i,:)`.

```
x=zeros(4,21);
x(:,1)=3;
r = [0.6 0.7 0.8 0.9];
for i = 1:4
    for j = 1:20
        x(i,j+1) = r(i)*x(i,j) + 1;
    end
end
figure
hold on
plot(0:20,x(1,:), 0:20,x(2,:), 0:20,x(3,:), 0:20,x(4,:))
xlabel('n')
ylabel('x_n')
legend('r=0.6', 'r=0.7', 'r=0.8', 'r=0.9')
hold off
```

We can see from the graph that varying the parameter  $r$  gives a range of values for  $x_{20}$  between about 2.5 and 9. Using the command `x(:,end)` shows the precise values in the last column to range from 2.5 to 9.1490.

This programming technique will also come in useful for simulation modeling in Chapter 7.

## 2.7 While Loops

In the last section, we see that `for` loops are useful to run a segment of code a set number of times. The disadvantage of the `for` loop is that the programmer must know in advance the number of iterations that he or she wants to run. To iterate a segment of code in which the number of iterations is not known in advance, we may use a **while loop**. A `while` loop will iterate a particular section of code until some condition is satisfied. The general form of the `while` loop looks like this:

### While loop

```
while logical_expression
    statements
:
end
```

The *logical\_expression* is a statement that is true or false (see [Activity A.0.5](#)). The set of statements inside the loop are executed as long as *logical\_expression* is true. If *logical\_expression* is not true, then MATLAB will skip the set of statements and continue execution after the `end` statement.

See [Activity A.0.13](#) for a step by step example of how a `while` loop works. [Activity A.0.14](#) explains how to use a `while` loop to enforce a stopping condition when seeking convergence of an infinite sequence.

Note that two logical expressions may be combined to form a new logical expression, sometimes called a **compound logical expression**.

Compound Logical Expression	MATLAB notation	When is it true?
$p$ and $q$	<code>p &amp;&amp; q</code>	When both $p$ and $q$ are true
$p$ or $q$	<code>p    q</code>	When at least one of $p$ or $q$ is true

See [Activities A.0.15](#) for an example of using compound logical expressions.

### For Loop or While Loop?

It is always possible to convert from a `for` loop to a `while` loop: both of these two segments of code compute  $\sum_{n=1}^{100} \frac{1}{n^2}$ .

<pre>s = 0; for i = 1:100     s = s + 1/i^2; end s</pre>	<pre>s = 0; i=1; while i &lt;= 100     s = s + 1/i^2;     i = i + 1; end s</pre>
--	--

It is not always feasible to convert from a `while` loop to a `for` loop. `While` loops are useful when you want to run the program until some convergence criteria is satisfied. In the code segment below, terms are added until the  $i^{\text{th}}$  term is below some **tolerance** (in this case  $10^{-3}$ ). In general, we do not know how many iterations it will run until we actually run the program.

<pre>s=0; tol = 1e-3; i = 1; while 1/i^2 &gt; tol     s = s + 1/i^2;     i = i + 1; end i s</pre>
---

We could vectorize the code to store either each term or the partial sum at each step in the `while` loop.

<pre>partialSum = 0; tol=1e-3; i=1; while 1/i^2 &gt; tol     partialSum(i+1) = partialSum(i) + 1/i^2;     i = i + 1; end partialSum'</pre>
--

After running the code, we can find the partial sum  $\sum_{n=1}^N \frac{1}{n^2}$  for any  $N$  from 1 to 100. For example, `partialSum(50)` should return  $\sum_{n=1}^{50} \frac{1}{n^2}$ .

## Stopping a MATLAB Script

When MATLAB is done running a script, you should see `>>` appear at the next line in the Command window. If you are getting tired of waiting for this to happen, click in the command window and use “Ctrl+C” to stop running the script. If your program does not seem to be ending, it might mean that MATLAB is stuck in an infinite loop. This is very common for beginning programmers when using `while` loops. Often the problem is that the `while` loop is set up so that the stopping condition never holds. MATLAB will dutifully run through the loop forever in this case. Here is one example:

```
x=2
while x==2
    y = x + 1;
end
```

In this example, the `while` loop continues as long as `x` is 2. However the line inside the `while` loop modifies `y`, not `x`. Therefore `x` remains 2, and the line inside the `while` loop will run forever (or until the program crashes).

## Commenting Code

As your code becomes more and more complex, commenting becomes increasingly important. Even for simple code, it can be quite useful in recalling what your program is intended to do, particularly if you come back to look at your script file some time after you’ve written it.

---

### Example 2.7.1

Suppose we want to write a MATLAB script that implements the division algorithm. The division algorithm takes a nonnegative integer  $a$ , a positive integer  $d$ , and finds the unique integers  $q$  and  $r$  such that  $a = dq + r$  and  $0 \leq r < d$ . In other words, we want to divide  $a$  by  $d$ , and find the unique quotient  $q$  and remainder  $r$ . We sometimes indicate the remainder by using the notation  $r \equiv \text{mod}(a, d)$ .

We simply apply the division algorithm in order to find these integers. To find  $q$  and  $r$ , we repeatedly subtract  $d$  from  $a$  until the result is less than  $d$  yet nonnegative. In the sample code below, the numbers on the left column indicate the line number for reference and are not part of the code.

```
%This script computes the quotient and remainder when a is divided by d.
%The quotient (q) and remainder (r) are output
1  a = 20          %a is the initial number
2  d = 6          %d is the divisor
3  r = a;         %initilize r to a
4  q = 0;         %the quotient q is the number of times that d is subtracted from r
5  while r >= d   %loop until r is less than d
6      r = r - d; %subtract d from r
7      q = q + 1; %add 1 to q
8  end
9  q              %output q
10 r             %output r
```

Note that  $20 = 6 \cdot 3 + 2$ , so we should have a quotient of  $q = 3$  and a remainder of  $r = 2$ . The top of the code contains a comment describing what the code does, and what values are output. In lines 3 and 4, comments describe the roles of the variables `r` and `a`. It is a good idea to at least describe the purpose of any variables that are initialized by using comments, but comments may be used to explain any line of code.

---

## 2.8 Conditional Statements

A **conditional statement**, or an **if-then statement**, is used to run a code segment only if some condition is met. It takes on the form “if A then B”. In MATLAB, the basic **if-then statement** looks like this:

```
If-then statement
if logical_expression
    statements
    :
end
```

The set of statements are only run if *logical\_expression* is true.

Often, we want something different to happen when the logical expression is false. In this case, we may need an **if-else statement**. These take on the form

```
If-else statement
if logical_expression
    statements A
else
    statements B
end
```

If *logical\_expression* is true, then the set of statements A are executed. If *logical\_expression* is not true, then the set of statements B are executed.

For example, you could write an M-file with the following code

```
if 2==3
    disp('the statement is true')
else
    disp('the statement is false')
end
```

If you run it, you should see displayed

```
>> the statement is false
```

since it is not true that  $2 = 3$ .

Refer to [Exercise A.0.16](#) for a basic example of conditional statements.

To write a conditional statement with more than two possibilities, we may want to use a **multiple if-else statement**. There are many variations, one of which is shown below:

```
Multiple if-else statement
if logical_expression 1
    statements A
elseif logical_expression 2
    statements B
elseif logical_expression 3
    statements C
else
    statements D
end
```

For this example, if *logical\_expression 1* is true, then *statements A* are executed. Otherwise, if *logical\_expression 2* is true, *statements B* are executed. If *logical\_expression 1* and *logical\_expression 2* are false, and *logical\_expression 3* is true, then *statements C* are executed. If none of these logical expressions are true, then *statements D* are executed. Note that only one set of *statements A*, *B*, *C*, and *D* are allowed

to be executed. Also, the very last `else` statement is not required, you may not want to do anything if none of the logical expression are true. There are many possible variations of multiple if-else statements.

One option you may wish to employ is to exit the conditional statement, `for` loop, or `while` loop if a condition is not met. In such cases, you may use the `break` or `return` commands. See `help break` for more information. The code segment below takes each number from 1 to 10 and displays whether it is divisible by 2, 3, or 5. However if the number is divisible by more than one of these then it only displays that it is divisible by the smallest of them. For example if `i` is 6 then it will display that it is divisible by 2 - but the following `elseif` statements are not checked.

```
n = 10;
for i = 1:n
    if rem(i,2) == 0
        disp([num2str(i), ' is divisible by 2'])
    elseif rem(i,3) == 0
        disp([num2str(i), ' is divisible by 3'])
    elseif rem(i,5) == 0
        disp([num2str(i), ' is divisible by 5'])
    else
        break
    end
end
```

Notice the `disp` command has an argument that is written as entries of a row vector, separated by a comma. The first entry uses the command `num2str` to change the numerical value of `i` to a string character so it can be used in conjunction to the string statement `' is divisible by 2 '`, for example.

See [Exercise A.0.17](#) for practice with a combination of loops and conditional statements.

A **switch statement** is used when there are many conditions and we wish to make the code more simple to read. Suppose we have some variable and have a variety of code to run depending on its value. Each alternative for each particular value of the variable is called a **case**, and a different set of statements may be run for each alternative.

```
Switch statement
switch variable
    case case A
        statements A
    case case B
        statements B
    case {case C1, case C2, ...}
        statements C
    otherwise
        statements D
end
```

Notes:

- Any case may consist of multiple values of the variable, such as the one shown in the third case statement.
- If no case expression matches the value of the variable, control passes to the `otherwise` case, if it exists.

## Summary of MATLAB commands

MATLAB command	What it does									
<code>x = some number</code>	Assigns a number to variable <code>x</code>									
<code>x = start:increment:end</code>	Assigns a vector of numbers to variable <code>x</code> . The first value is <code>start</code> , and the subsequent values are obtained by taking increments of size <code>increment</code> , and the array ends when value <code>end</code> has been reached. Example: <code>x = 2:3:17</code> yields <code>x = [2, 5, 8, 11, 14, 17]</code>									
<code>A=[1 2 3; 4 5 6; 7 8 9]</code>	Assigns matrix <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td></tr></table> to variable <code>A</code>	1	2	3	4	5	6	7	8	9
1	2	3								
4	5	6								
7	8	9								
<code>R = [1 2 3]</code>	Assigns the row vector <code>[1 2 3]</code> to variable <code>R</code>									
<code>C = [1; 4; 7]</code>	Assigns the column vector <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1</td></tr><tr><td>4</td></tr><tr><td>7</td></tr></table> to variable <code>C</code>	1	4	7						
1										
4										
7										
<code>A(n,m)</code>	Output $a_{nm}$ (row $n$ and column $m$ of $A$ )									
<code>A(n,:)</code>	Output row $n$ of $A$									
<code>A(:,m)</code>	Output column $m$ of $A$									
<code>size(A)</code>	Output the size of $A$ in the form <code>[# columns # rows]</code>									
<code>A'</code>	Computes the transpose of $A$									
<code>x*y</code>	Compute product $xy$									
<code>x.*y</code>	Computes element by element product									
<code>sqrt(x)</code>	Compute $\sqrt{x}$									
<code>x^n</code>	Compute $x^n$									
<code>log(x), log10(x)</code>	Compute $\ln x$ , $\log_{10}(x)$ , respectively									
<code>abs(x)</code>	Compute absolute value $ x $ .									
<code>exp(x)</code>	Compute $e^x$									
<code>sin(x), cos(x), tan(x)</code>	Compute $\sin(x)$ , $\cos(x)$ , $\tan(x)$									
<code>asin(x), acos(x), atan(x)</code>	Compute $\arcsin(x)$ , $\arccos(x)$ , $\arctan(x)$									
<code>floor(x)</code>	Compute floor (greatest integer below $x$ )									
<code>ceil(x)</code>	Compute ceiling (smallest integer above $x$ )									
<code>round(x)</code>	Round $x$ to nearest integer									
<code>vpa(fraction)</code>	Converts fractions into decimal approximations									
<code>rref(A)</code>	Reduces $A$ to row reduced echelon form (performs Gaussian elimination)									
<code>inv(A)</code>	Computes the matrix inverse $A^{-1}$									
<code>figure(n)</code>	Opens up figure number $n$ (useful for working with multiple figures)									
<code>hold on</code>	After running this line, any subsequent plot command is added to the current figure (instead of creating a new figure from scratch).									
<code>;</code>	Add the semicolon at the end of a line to suppress output to the command line. (Useful for speeding up code.)									
<code>ezplot('cos(x)')</code>	plot $\cos(x)$ with default window									
<code>ezplot('cos(x)', [xmin,xmax])</code>	plot $\cos(x)$ on the interval $x_{\min} \leq x \leq x_{\max}$									
<code>ezplot('cos(x)', [xmin,xmax, ymin,ymax])</code>	plot $\cos(x)$ on the interval $x_{\min} \leq x \leq x_{\max}$ , $y_{\min} \leq y \leq y_{\max}$									
<code>plot(x,y)</code>	Plot vector $y$ against $x$									
<code>x &lt; y</code>	Test logical expression $x < y$ (returns 1 for true, 0 for false)									
<code>x &lt;= y</code>	Test logical expression $x \leq y$ (returns 1 for true, 0 for false)									
<code>x == y</code>	Test logical expression $x = y$ (returns 1 for true, 0 for false)									
<code>x ~= y</code>	Test logical expression $x \neq y$ (returns 1 for true, 0 for false)									
<code>format long</code>	Displays numbers with 15 decimal places									
<code>format short</code>	Displays numbers with 4 decimal places									



MATLAB command	What it does
for i=a:b <i>code segment</i> end	For loop executes the code segment that follows, as variable <i>i</i> increments from <i>a</i> to <i>b</i> .
while <i>logical_expression</i> <i>code segment</i> end	While loop executes the code segment that follows until <i>logical_expression</i> is false.
for i=a:b <i>code segment</i> end	For loop executes the code segment that follows, as variable <i>i</i> increments from <i>a</i> to <i>b</i> .
if <i>logical_expression</i> <i>code segment</i> end	If-then statement executes the code segment that follows, if <i>logical_expression</i> is true.
mod(q,d)	Returns the remainder if <i>q</i> is divided by <i>d</i>
zeros(n,m)	Make a $n \times m$ matrix of all zeros

## 2.9 Exercises

- Use `linspace` to generate a row vector of 120 equally spaced numbers in the interval (2, 5).
- Use `linspace` to generate a row vector of numbers in the interval  $[-1, 3]$  that are all separated by a 0.1. Then repeat the process using the command `a:inc:b`.
- Use `plot` to plot the function  $f(x) = 3 \cos(2x + 3)e^{-5x}$  in the interval  $[0, 3]$ . Include the  $x$ -axis in your plot, and label the axes.
- Repeat problem 3 but use `ezplot` instead of `plot`.
- Suppose a force of 100 Newtons is placed on an object and displaced by 3 meters. Use MATLAB to compute the work done on the object ( $W = Fd \cos \theta$ ) if the angle between the force and displacement vectors is  $\theta = [0, 15, 30, 45, 60, 75]$  degrees. Plot the work done versus the angle using a plot symbol of your choice.
- A powerline that freely hangs between two poles has the shape of a catenary curve. If  $x = 0$  at the vertex, suppose the curve of the powerline is given by the equation  $f(x) = 2(e^{x/4} + e^{-x/4})$ . Plot the curve over the interval  $[-4, 4]$  using `plot` or `ezplot`.
- Suppose that the rate of resource gain for a forager in a given area is given by the function  $g(t) = \frac{\ln(t+0.5)}{t}$ , where  $t$  is the amount of time spent foraging in this area. Plot  $g(t)$  in MATLAB and use it to estimate the value of  $t$  that maximizes  $g(t)$ . What does this represent in terms of the forager?
- Let  $f(x) = 3e^{-0.1x^2}$ ,  $g(x) = -2 \cos(x/2)$ . Use `plot` to make a plot of the functions  $f(x)$  and  $g(x)$  on the same set of axes for the values  $0 \leq x \leq 6$ . Also,
  - Label the axes  $x$  and  $y$ .
  - Set the range of  $y$ -values to  $-3 \leq y \leq 4$ .
  - Make  $f(x)$  and  $g(x)$  two different colors.
  - Include the  $x$ -axis in black.
  - Include a legend to indicate which function is which.
- Repeat problem 8 but use `ezplot` instead of `plot`.

10. Use `plot` to plot the piecewise defined function

$$f(x) = \begin{cases} 2x + 1, & 0 \leq x < 1, \\ 3/x, & 1 \leq x \leq 4 \end{cases}$$

on the interval  $0 \leq x \leq 4$ .

11. Use `plot` to plot the piecewise defined function

$$f(x) = \begin{cases} -x/2, & -1 \leq x < 0, \\ e^{3x}, & 0 \leq x \leq 1 \end{cases}$$

on the interval  $-1 \leq x \leq 1$ . Use open and closed circles on the graph to indicate a function.

12. Write a MATLAB file to approximate the integral  $\int_0^1 x^2 dx$  using the sum  $\sum_{i=1}^n x^2 \Delta x$  with  $\Delta x = \frac{1}{n}$ . Compare the error in your results for  $n = 10$ ,  $n = 100$ , and  $n = 1000$ .
13. Write a MATLAB file used to compute the summation  $\sum_{n=1}^{10} \left( \frac{1}{n} - \frac{1}{n+1} \right)$ .
14. Write a MATLAB file that computes  $n!$  for any integer  $n \geq 0$ . Use a `while` loop.
15. The Euclidean Algorithm is used to find the greatest common divisor of  $a$  and  $b$ . The greatest common divisor,  $d$ , has the property that  $d$  is a divisor of both  $a$  and  $b$ , and if  $c$  is another common divisor of  $a$  and  $b$ , it must follow that  $c \leq d$ . The pseudocode below is the Euclidean Algorithm, used to find the greatest common divisor of positive integers  $A$  and  $B$  with  $A > B > 0$ . Convert the code to working MATLAB code that takes arguments  $A$  and  $B$  and outputs their greatest common divisor.

```

a := A
b := B
r := B
while b ≠ 0
    r := a mod b
    a := b
    b := r
end while
gcd := a
output gcd

```

16. Write a MATLAB file that finds the first  $n$  Fibonacci numbers, for any  $n$ .
17. Write a program that outputs all divisors of a given positive integer.
18. Use a nested `for` loop to write a program to find the sum of two matrices  $A$  and  $B$ . Before adding, use a conditional statement to ensure the dimensions of the matrices are compatible and return an error statement if they are not.
19. Without using the `isprime()` function, write a MATLAB file used to determine if a number  $x$  is prime or not. To do this, divide  $x$  by all the integers below  $\sqrt{x}$ , and examine the remainder. The code should output text to the command window, saying whether or not the number is a prime.
20. Recall from Calculus II that the harmonic series  $\sum_{n=1}^{\infty} \frac{1}{n}$  diverges to infinity. Write MATLAB code to compute  $\sum_{n=1}^N \frac{1}{n}$  for  $N = 10, 100, 1000$ , and  $10000$ . Determine how large  $N$  must be in order to satisfy  $\sum_{n=1}^N \frac{1}{n} \geq 15$ .
21. It turns out that  $\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$ . Write MATLAB code to compute  $\sum_{n=1}^N \frac{1}{n^2}$  for  $N = 10, 100, 1000$ , and  $10000$ . Compare each of these to  $\frac{\pi^2}{6}$ . Determine how large  $N$  must be in order for the error to satisfy  $\left| \sum_{n=1}^N \frac{1}{n^2} - \sum_{n=1}^{\infty} \frac{1}{n^2} \right| < 10^{-6}$ .

22. Suppose the sequence  $\{a_i\}_{i=1}^{\infty}$  is defined as follows:

$$a_1 = 1, \quad a_2 = 3, \quad a_k = a_{k-2} + 2a_{k-1} \text{ for integers } k \geq 3.$$

Write a MATLAB file used to generate the sequence, and use it to find  $a_{20}$ .

23. See [6]. Consider the velocity of a rocket given by the piecewise function below. Develop an m-file using conditional statements to compute  $v$  as a function of  $t$ . Use this to generate a plot of  $v$  versus  $t$  on the interval  $-5 \leq t \leq 50$ .

$$v(t) = \begin{cases} 11t^2 - 5t, & 0 \leq t \leq 10 \\ 1100 - 5t, & 10 < t \leq 20 \\ 50t + 2(t - 20)^2, & 20 < t \leq 30 \\ 1520e^{-0.2(t-30)}, & t > 30 \\ 0, & \text{otherwise} \end{cases}$$

24. Consider the recursively defined sequence  $x_0 = 3, x_{n+1} = \sqrt{1 + x_n}$ . Write a `for` loop used to calculate  $x_n$  for  $n = 1, 2, \dots, 30$ . [Hint: to take the square root of a number `x` in MATLAB use `sqrt(x)`.] At the beginning of your script, enter the line `format long` to increase the number of decimal places that are displayed.

- Write out the exact number that you are computing on the first, second and third steps, then give  $x_{30}$ . Does the sequence seem to be converging to a number and if so, what number?
- Compute the solution to  $x = \sqrt{1 + x}$  by hand. Compare to your answer in (a).
- Modify your MATLAB code to generate values of the sequence  $x_n$  until the “error”  $|x_n - \phi|$  is less than  $10^{-4}$ , where  $\phi$  is the answer you found in (b).

25. Write a MATLAB file used to count the sizes of the gaps between the first 1000 prime numbers. For example, the first few prime numbers are 2, 3, 5, 7, 11, and so the first few gaps would be 1, 2, 2, 4. Then generate a histogram (using `hist()`) of these 1000 numbers and comment on the shape of the distribution. [Hint: start by making a vector of size 1000 called `gap`. Set up a `while` loop to stop after 1000 prime numbers are found. Inside the `while` loop use `isprime()` to test whether the number is prime. If it is, you want to increase the number of prime numbers found by 1. If it is not, you want to increase the value of `gap(i)` by 1, where `i` is the number of prime numbers found so far.]

26. In studying rocket propulsion, we have to keep in mind that the overall mass changes as a rocket burns fuel. Due to conservation of momentum, the overall momentum remains constant. This means that the acceleration of the rocket will not be constant.

The initial momentum of the system is  $p_i = mv$  where  $m$  is the mass of the rocket and fuel combined, and  $v$  is the velocity of the rocket. Assuming that the engines burn at a constant rate, then during a small time interval  $dt$  the mass decreases by  $dm_g$  with velocity  $-u$  (where we use the negative sign since it is moving at the opposite direction of the rocket). The exhaust gas has velocity  $v - u$  (with respect to earth). Let  $dv$  be the change in velocity during the time interval  $dt$ . The final momentum is

$$p_f = p_{\text{rocket}} + p_{\text{gas}} = (m - dm_g)(v + dv) + dm_g(v - u).$$

- (a) Set  $p_f = p_i$  to show that

$$m \cdot dv = dm_g \cdot dv + dm_g \cdot u. \quad (2.2)$$

- (b) Show that  $dv = \frac{dm_g u}{m - dm_g}$ .

- (c) Suppose that a spacecraft is moving in gravity-free space in a straight path. Thrusters are turned on, fuel is ejected at  $2.0 \times 10^2$  kg/s at a speed (relative to the rocket) of  $2.5 \times 10^2$  m/s. The initial mass of the spacecraft and unburned fuel is  $2.0 \times 10^4$  kg and the thrusters are on for 30 seconds. What is the thrust (force applied to rocket by ejected fuel)? What is the acceleration  $a(t)$ ? Write a MATLAB code to estimate the velocity throughout the 30 second period. Use  $\Delta t = 1$ .

- (d) In (2.2), we can probably ignore  $dm_g \cdot dv$  since this is a product of two small quantities. Show that this results in  $dv = u \frac{dm}{m}$ .
- (e) Integrating the equation you just found throughout a period of time,

$$\int_{t_0}^{t_f} dv = -u \int_{t_0}^{t_f} \frac{1}{m} dm.$$

This results in

$$dv = u \ln \left( \frac{m_i}{m} \right).$$

This is the rocket equation, first established by the Soviet physicist Konstantin Tsiolkovsky in 1897. Use the rocket equation to solve the problem posed in (c), and compare to your answer found in (d).

## Chapter 3

# Iterative Methods

The focus of this chapter is on solving equations of the form  $f(x) = 0$ . Such a solution is called a zero, or a root of the function  $f(x)$ . In earlier courses, you may have focused on solving this problem by hand for specific types of functions  $f$ , such as linear, polynomial, or rational functions. Some of these problems were designed so that finding an analytic solution is possible. However, many equations are not so easily solved by hand - for example,  $x^6 + x - 1 = 0$ , or  $\sqrt{x+1} = 2^x$ . A crude method of approximating solutions is to use technology to graph the equation in the form  $f(x) = 0$ , then zoom in to determine where the graph crosses the  $x$ -axis. How can we write a computer program to mimic this process in order to approximate the roots? In this section, we look at iterative techniques for approximating these roots. Knowing how well an algorithm will work, or even if it will work at all requires knowing some qualitative properties of  $f(x)$ . This means we need to introduce a little bit of theory in order to understand when the algorithm can be used.

### 3.1 Fixed Point Iteration

In [Activity A.0.10](#) you found that the sequence

$$M_{t+1} = 0.5M_t + 1, \quad M_0 = 1 \quad (3.1)$$

converged to 2, that is  $\lim_{t \rightarrow \infty} M_t = 2$ . It turns out that  $\lim_{t \rightarrow \infty} M_t = 2$  for any starting value  $M_0$ . You can verify this for yourself by running the code in [Activity A.0.10](#) and changing the initial value of `m`.

Now imagine that we don't know the limit, call it  $M^* = \lim_{t \rightarrow \infty} M_t$ . As long as a limit exists, we should have  $M^* \approx M_{t+1} \approx M_t$  for large values of  $t$ . Assuming that  $M^* = M_{t+1} = M_t$  and substituting this into (3.1), we get  $M^* = 0.5M^* + 1$ . Solving this in turn gives  $M^* = 2$ . With a fixed point iteration we are reversing this process in a sense. The goal is to solve an equation in the form  $f(x^*) = x^*$ . For this particular example it is easy to solve  $M^* = 0.5M^* + 1$  algebraically so there is no need to use any special algorithm to solve it. However, the fixed point iteration method can be used to solve an equation such as  $\ln x + 2 = x$ , which does not have a closed-form solution.

**Definition 3.1.1.** A value  $x^*$  is a **fixed point** of a function  $f(x)$  if  $f(x^*) = x^*$ .

The algorithm to find a fixed point of a recurrence relation  $f(x_n)$  is relatively simple. We compute values of the sequence  $x_{n+1} = f(x_n)$  until the values seem to be converging. To be precise, we will use a specific error tolerance to ensure the fixed point is as close as we'd like to be to the true fixed point. This will be programmed in a way similar to the tolerances we used in Chapter 2 to approximate the value of a summation.

To describe a generic algorithm that is used to compute a fixed point, computer programmers or mathematicians often use **pseudocode**. The pseudocode provides an outline of the computer code in a generic form,

and does not correspond to a specific programming language. The pseudocode for **fixed point iteration** is shown below.

#### Fixed Point Iteration to solve $f(x) = x$

```

choose initial guess  $x_0$ 
set  $n := 1$ 
loop until some convergence criterion is met, or until  $n$  is too large
     $x_{n+1} := f(x_n)$ 
     $n := n + 1$ 
end loop
output  $x_n$ 

```

It's worth taking a minute to consider how one might come up with an initial "guess". Different initial guesses may lead to different results, particularly if there are multiple roots to the equation  $f(x) - x = 0$ . It will sometimes take a bit of trial and error to determine an initial guess that will work. We may know something about the function or be able to use mathematical theory to help narrow down the interval in which a fixed point may occur. The intermediate value theorem, for example, guarantees a root in a closed interval provided the function is continuous and has values with opposite signs at each endpoint. Technology may also give us a rough approximation to the root that can be used as an initial guess. You may use these methods and others in choosing  $x_0$ .

#### Example 3.1.2

We wish to determine if the recurrence relation  $x_{n+1} = \ln(x_n) + 2$  converges. Use fixed point iteration to solve  $\ln(x) + 2 = x$ , accurate to within a tolerance of  $10^{-4}$ .

Using the pseudocode above, a basic MATLAB script to find the fixed point is given here.

```

1 x=1; %initialize x
2 tol=1e-4; %set tolerance
3 i=1; %set the index
4 x(i+1)=log(x(i))+2; %apply fixed point iteration
5 while abs(x(i+1)-x(i))>tol %until convergence is met
6     i = i + 1; %update index
7     x(i+1)=log(x(i))+2; %apply fixed point iteration
8 end
9 x' %output iterates

```

On line 1, we set our initial "guess" for the root to  $x_0 = 1$ . On lines 2 and 3 we set the tolerance to  $10^{-4}$ , and initialize `i` to keep track of the number of iterations. On line 4 we apply the first step of the fixed point algorithm. Without this, there is no way to evaluate the condition in line 5, since that requires knowing the *two* most recent values of `x`. In line 6 we update the index `i` and line 7 performs the fixed point iteration. Lines 6 and 7 are repeated until the difference between  $x(i)$  and  $x(i+1)$  is less than the tolerance of  $10^{-4}$ . Keep in mind that in this script, `i` is initially set to 1, so line 5 will initially compare `x(1)` and `x(2)` which is what we want. The first time through the loop, `i` will increase to 2, then on line 7 we update `x(3)`. Next we end up on line 5 again, but this time compare `x(3)` with `x(2)`. It is easy to be off by an index, so it is important to mentally work through your code step by step.

One common pitfall that beginning MATLAB programmers make is to initialize a vector `x` using `x(1)`. The problem is that if `x` is already in the workspace as a vector (this could be from a program you ran last week), then MATLAB will only update `x(1)` which is probably not desirable. This is why in line 1, we initialized `x` without using an index - this tells MATLAB to remove any previous values of `x` and start over by setting it to a single number.

**Avoid using  $x(1)$  to initialize a vector  $x$**

By running this code you should see that the iterates converge to 3.1462.

In the example above, the sequence  $x_{n+1} = \ln(x_n) + 2$  converges to 3.1462 for most initial guesses. However there are two solutions to  $x = \ln(x) + 2$ , the other one being close to 0.1586. Also if your initial guess is less than 0.1586, the iterates become complex numbers (the outputs are of the form  $a \pm bi$ ), since MATLAB is eventually taking the logarithm of negative numbers. So the fixed point algorithm does not find all of the fixed points, and in other cases it might not find any fixed points.

Theorems 3.1.4 and 3.1.5 give us some insight into when the fixed point iteration works. Specifically,  $f(x)$  needs to satisfy certain properties. See Appendix F for the proofs to these (and other) theorems.

**Definition 3.1.3.** We say that a function  $f$  is **smooth** on an interval  $I$  if  $f'$  exists on  $I$  and  $f'$  is continuous on  $I$ .

The following theorem tells us a condition that guarantees  $f$  has a fixed point.

**Theorem 3.1.4.** If  $f$  is continuous on  $[a, b]$ , and  $a \leq f(x) \leq b$  for all  $x \in [a, b]$ , then  $f$  has a fixed point in  $[a, b]$ .

Notice that both  $x$  and  $f(x)$  must be bounded by the same values  $a$  and  $b$ . This is not always easy to do, in practice.

The next theorem gives conditions for which a fixed point is unique, and for which the fixed point iteration method  $x_n = f(x_{n-1})$  is guaranteed to converge to this unique fixed point.

**Theorem 3.1.5.** If  $f$  is smooth on  $[a, b]$ ,  $a \leq f(x) \leq b$ , and  $|f'(x)| \leq k$  for some  $k < 1$  on  $(a, b)$ , then  $f$  has a unique fixed point  $x^* \in [a, b]$ . In addition, for any number  $x_0$  in  $[a, b]$ , the sequence defined by  $x_n = f(x_{n-1})$ ,  $n \geq 1$ , converges to this unique fixed point  $x^*$  in  $[a, b]$ . Moreover,  $|x_n - x^*| \leq k^n |b - a|$ .

### Example 3.1.6

Can we apply Theorems 3.1.4 and 3.1.5 to ensure that  $f(x) = \sqrt{1+x}$  has a fixed point on some interval  $[a, b]$ ? How many iterations are required to be within  $10^{-5}$  of the solution?

.....

Notice that  $f$  is continuous for all  $x \geq -1$ . The second condition of Theorem 3.1.4 holds for several values of  $a$  and  $b$ . For example, if  $0 \leq x \leq 10$  then certainly  $0 \leq \sqrt{1+x} \leq 10$ . Therefore,  $0 \leq f(x) \leq 10$  for all  $x \in [0, 10]$ , and Theorem 3.1.4 ensures that  $f$  must have a fixed point in  $[0, 10]$ .

Next,  $f'(x) = \frac{1}{2\sqrt{1+x}}$ . Notice that  $f'$  is a decreasing function on  $[0, 10]$  so therefore must have a maximum value at  $x = 0$ . Also,  $f'$  is continuous on  $[0, 1]$  so  $f$  is smooth on  $[0, 10]$ . Moreover,  $f'$  is nonnegative on  $[0, 10]$  so

$$|f'(x)| = f'(x) \leq f'(0) = \frac{1}{2} < 1.$$

By Theorem 3.1.5 (with  $k = \frac{1}{2}$ ),  $f$  has a unique fixed point in  $[0, 10]$  and the sequence  $x_n = \sqrt{1+x_{n-1}}$  should converge to this fixed point for any initial point  $x_0 \in [0, 10]$ .

If we want  $x_n$  to be within  $10^{-5}$  of the root, we need  $|x_n - x^*| \leq k^n |b - a| = 0.5^n \cdot 10 < 10^{-5}$ . Solving for  $n$ , we see that  $n \geq 20$ , meaning that we should use at least 20 iterations.

A common question for iterative methods is how to determine when we have reached a root. If we are not sure how many iterations we need, it makes sense to use a while loop with a **stopping condition** or **convergence**

**criteria.** The stopping condition used in the MATLAB script in Example 3.1.2 is  $|x_n - x_{n-1}| < \text{TOL}$ , where TOL is  $10^{-5}$ . That is, we stop until the iterates are changing by less than TOL. This usually means that  $x_n$  is within TOL units of  $x^*$  but this is not guaranteed.

Another type of stopping condition called the **absolute relative approximate error** takes into account the relative size of  $x_n$ . The stopping condition is

$$\left| \frac{x_{n+1} - x_n}{x_{n+1}} \right| < \text{TOL}.$$

Work through **Activity A.0.18** to see how the tolerance relates to the number of iterations.

Refer to **Exercise A.0.19** for an example of root finding with fixed point iteration.

## 3.2 MATLAB function files

Recall in Section 2.3 that a function carries out a set of instructions and may have inputs (called arguments) and it may return outputs (such as numbers, matrices, and so on).

So far we have been working with MATLAB scripts as well as MATLAB's built-in functions. In this section we will create our own MATLAB functions and learn why functions are so useful. One thing that working with functions does is to help modularize larger programs. Modularization refers to splitting code up into smaller segments, where each segment has a specific task. The resulting code may be easier to work with, especially if it is a large project and there are multiple programmers involved. Another benefit of using a function is that the MATLAB will execute code more quickly if it is in a function compared to a script.

When you create a MATLAB function file, you must begin the file with “function file\_name(arguments)” and end the file with “end”. The arguments are variables that are input into the function. To run the program in which there are arguments, you must “call” this program from the command line, indicating the values of the arguments to be used. You may write multiple functions within a single .m file, or use separate .m files for different functions.

Recall the code in Example 3.1.2 used to compute a fixed point. We could implement this code as a function instead of a script. In the code below, the MATLAB file must be saved as **FixedPoint.m**. Once you program this, you will be able to use it in future exercises and activities.



```

%FixedPoint takes one argument x0 as an initial guess, and approximates a fixed point
%of the function f, given below.
%Display values of x at each iteration and number of iterations to convergence.

1  function FixedPoint(x0)
2  tol=1e-4;           %tolerance
3  x(1)=x0;           %set first iterate x1 equal to the initial guess x0
4  x(2)=f(x(1));      %compute next x value
5  n=1;
6  while abs(x(n+1)-x(n)) > tol %loop until |x(n+1)-x(n)| is below tol
7      n=n+1;          %update n
8      x(n+1)=f(x(n)); %update x
9      if n > 100 || abs(x(n+1))>100 %if n or |x(n+1)| get too large, stop
10         disp('Fixed point iteration appears to be diverging')
11         break       %exit while loop
12     end
13 end
14 numIterations=n+1 %display the number of iterations
15 format long      %change setting to display more digits
16 x'               %display all values of x
17 end
18
19 function f=f(x)   %this is the function to find the fixed point of
20 f = cos(x);
21 end

```

Function files are allowed to have other functions inside the code, and may also call other function files that are contained in the current directory. This file contains another function `f` defined in lines 19 - 20, which takes in one argument, `x`, and outputs a number `f`. On line 4, this function `f` is called with argument `x(1)` and on line 8 it is called with argument `x(n)`. On line 20, the function `f` is defined to return the value of  $\cos(x)$ . One could easily change the function to find the fixed point of by changing line 20. Another new feature of this code is shown on lines 9 to 12. The `if` statement on line 9 is used to determine if  $n$  is too large (greater than say 100), or if  $|x_n|$  is too large (greater than 100). On line 10, we display a note to the command line, indicating that the iteration is not converging. On line 11, the `break` statement tells MATLAB to immediately exit the while loop. On line 14, we output the number of iterations to the command window and on line 16 we output the entire vector  $x$  to the command window (we use `x'` instead of `x` so that it is displayed as a column vector instead of a row vector) - this gives the value of the entire sequence  $x_n$  up until convergence is achieved.

If you haven't already, create this file and save it as `FixedPoint.m`. To run the code, you cannot simply press F5, since it is looking for a value of the input argument `x0`. You need to call the program from the command line (or from another function file). So from the command line, you need to enter `FixedPoint(x0)`, where  $x_0$  is your initial guess. For example,

```
FixedPoint(1)
```

should result in convergence to the fixed point 0.7394.

Refer to [Activity A.0.20](#) for practice using Theorem 3.1.4, Theorem 3.1.5, and the `FixedPoint.m` function.

### 3.3 The Bisection Method

The **bisection method** of approximating solutions to  $f(x) = 0$  is relatively intuitive once we recall the Intermediate Value Theorem (IVT). To begin, we take two points  $(a, f(a))$  and  $(b, f(b))$  on the graph that

are on opposite sides of the  $x$ -axis. As long as  $f(x)$  is continuous, the IVT states the root must be between  $a$  and  $b$ . On the first step we choose the midpoint of  $a$  and  $b$ , call this  $c$ . The root must either be between  $a$  and  $c$ , or between  $b$  and  $c$ . In Figure 3.1, we see that the root must be between  $a$  and  $c$ , since  $f(a)$  and  $f(c)$  have opposite signs. Now we repeat this process with points  $a$  and  $c$  and watch as the root estimates converge to a single value.

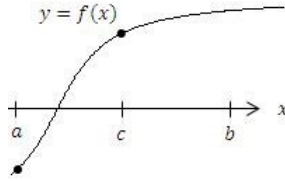


Figure 3.1: Since the function  $f(x)$  is a continuous function and  $f(a)$  and  $f(c)$  have opposite signs, the Intermediate Value Theorem states  $f$  must have a root in the interval  $[a, c]$ .

See [Exercise A.0.21](#) to practice working through the Bisection algorithm.

A natural way to implement this process is with a computer code. We must assume that  $f$  is some continuous function, and  $f(a)$  and  $f(b)$  have opposite signs. For practice, you may wish to add conditional statements which exit the program and show an error statement if these conditions are not met.

#### Bisection Method Algorithm to solve $f(x) = 0$

```

n := 1
Loop until some convergence criterion is met, or until n is too large
  xn := (a + b)/2
  If f(xn) and f(b) have opposite signs, then
    a := xn
  else
    b := xn
  end
  n := n + 1
end
output xn

```

As a reminder, even straight-forward code can easily include indices that are “off by one”. It may help to use the debugger, which is explained in Appendix E. If you get an error such as

Attempted to access x(3); index out of bounds because numel(x)=2

then it is likely that your indexing is off by one. The error tells you that it is trying to access `x(3)`, but `x` has less than 3 things in it. To check if two numbers  $x$  and  $y$  are of opposite signs, you could use the `sign` command, where `sign(x)` returns 1 if  $x > 0$ , and  $-1$  if  $x < 0$ . Or, simply use the fact that  $x$  and  $y$  have opposite signs if and only if  $xy < 0$ .

At each iteration in the process, the length of the interval containing the root is cut in half. This means that the error should not exceed  $\frac{b-a}{2^n}$  where  $a$  and  $b$  are the endpoints of the original interval and  $n$  is the number of iterations. The following theorem formalizes this fact.

**Theorem 3.3.1.** *Suppose that  $f$  is smooth on  $[a, b]$ , and that  $f(a)$  and  $f(b)$  have opposite signs. The Bisection method generates a sequence  $x_n$  approximating a zero  $x$  of  $f$  with  $|x_n - x| \leq (b - a)/2^n$ ,  $n \geq 1$ .*

See [Exercise A.0.22](#) for an example of using Theorem 3.3.1. So far we have seen two root-finding algorithms, and we will see two more before the end of the chapter. The next definition is used to compare the performance between the various algorithms.

**Definition 3.3.2.** Suppose that  $\{x_n\}_{n=0}^{\infty}$  is a sequence that converges to  $x$ , with  $x_n \neq x$  for all  $n$ . If positive constants  $\lambda$  and  $\alpha$  exist with

$$\lim_{n \rightarrow \infty} \frac{|x_{n+1} - x|}{|x_n - x|^\alpha} = \lambda,$$

then  $x_n$  converges to  $x$  of order  $\alpha$ , with asymptotic error constant  $\lambda$ . If  $\alpha = 1$ , then the sequence is said to be linearly convergent. If  $\alpha = 2$ , the sequence is quadratically convergent.

To demonstrate the different convergence speeds, suppose that  $x_n$  and  $y_n$  both converge to zero, and satisfy

$$\lim_{n \rightarrow \infty} \frac{|x_{n+1}|}{|x_n|} = \frac{1}{2} \quad \text{and} \quad \lim_{n \rightarrow \infty} \frac{|y_{n+1}|}{|y_n|^2} = \frac{1}{2}.$$

That is,  $x_n$  is linear and  $y_n$  is quadratic. For simplicity, assume that

$$\frac{|x_{n+1}|}{|x_n|} \approx \frac{1}{2} \quad \text{and} \quad \frac{|y_{n+1}|}{|y_n|^2} \approx \frac{1}{2}.$$

Then we can show that

$$|x_n - 0| \approx \left(\frac{1}{2}\right)^n |x_0| \quad \text{and} \quad |y_n - 0| \approx \left(\frac{1}{2}\right)^{2^n - 1} |y_0|^{2^n}.$$

It turns out that bisection and fixed-point iteration are linearly convergent. See [Exercise A.0.23](#) for practice showing linear convergence.

The next algorithm in this chapter is Newton's method, which turns out to be quadratically convergent.

## 3.4 Newton's Method

Newton's method is a method of solving  $f(x) = 0$  that works in the following way. Given an iterate  $x_n$ , we will find  $x_{n+1}$  by following the tangent line of  $f(x)$  at  $x_n$ , to where it crosses the  $x$ -axis. On the next step we find the tangent line to  $f$  at this new point  $x_{n+1}$  and repeat this process. Since we are following a tangent line at each step it is necessary that  $f$  is differentiable, at least near the desired root.

Figure 3.2 shows a function  $f$  along with its tangent line at the point where  $x = x_n$ . Since this tangent line has slope  $f'(x_n)$  and passes through  $(x_n, f(x_n))$ , its equation must be  $y = f(x_n) + f'(x_n)(x - x_n)$ . To obtain the next iterate  $x_{n+1}$ , we determine where the tangent line crosses the  $x$ -axis. We thus set  $y = 0$  and  $x = x_{n+1}$  to get  $0 = f(x_n) + f'(x_n)(x_{n+1} - x_n)$ . Solving for  $x_{n+1}$  gives

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \quad (3.2)$$

As with other iterative methods, we must choose an initial guess  $x_0$ .

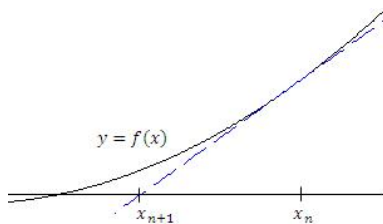


Figure 3.2: One iteration of Newton's method. Dashed line is the tangent line to  $f(x)$  at the point  $(x_n, f(x_n))$

**Newton's Method Algorithm to solve  $f(x) = 0$** 

```

choose initial guess  $x_0$ 
set  $n := 1$ 
loop until some convergence criterion is met, or until  $n$  is too large
     $x_{n+1} := x_n - f(x_n)/f'(x_n)$ 
     $n := n + 1$ 
end loop
output  $x_n$ 

```

See [Exercise A.0.24](#) for an example working through Newton's method by hand.

The code that follows applies Newton's Method for the function  $f(x) = \cos x$ .

```

%Apply Newton's method with initial guess x0
%Output the number of iterations and the iterates

1  function NewtonsMethod(x0)
2  tol=1e-4;
3  x(1)=x0;
4  x(2)=x(1) - f(x(1))/f_prime(x(1));
5  n=1;
6  while abs(x(n+1)-x(n)) > tol
7      n=n+1;
8      x(n+1)=x(n)-f(x(n))/f_prime(x(n));
9      if n > 100 || abs(x(n+1))>100
10         disp('Newtons method appears to be diverging')
11         break
12     end
13 end
14 num_iterations=n
15 x'
16 end          %end of Newtons Method function
17
18 function f=f(x) %function to iterate
19     f = cos(x);
20 end
21
22 function f_prime=f_prime(x)
23     f_prime = -sin(x);
24 end

```

The Newton's method code is almost identical to the fixed point code shown in Section 3.2, but here we have a new function called `f_prime` used to compute  $f'$ . The other difference is shown in lines 4 and 8, where Newton's method is applied instead of the fixed point iteration.

Note also that the number of iterations is given as `n` since the indices are shifted one due to  $x_0$  being defined in MATLAB as `x(1)`.

Newton's method works for a much wider variety of functions compared to fixed point iteration. This theorem gives a sufficient condition for Newton's method to converge.

**Theorem 3.4.1.** *Suppose  $f(x)$  has a root  $p$  in  $[a, b]$ ,  $f'(p) \neq 0$ , and  $f'$  is smooth on  $[a, b]$ . Then there is some closed interval  $I$  containing  $p$  such that Newton's method converges to  $p$  for any initial guess  $x_0 \in I$ .*

**Example 3.4.2**

The equation  $\ln x + 2 = x$  has two solutions. Use Newton's method to find them. What initial guesses will lead to finding each of the two solutions?

.....

First, we need to get the equation into the form  $f(x) = 0$ , so subtracting  $x$  gives  $f(x) = \ln x + 2 - x$ .

In Example 3.1.2 we found that the fixed point iteration only found one of the two solutions, 3.1462.

Notice that  $f'(x) = 1/x - 1$  is a smooth function for  $x > 0$ , so by Theorem 3.4.1 Newton's method should be able to produce both solutions (there cannot be negative solutions since the domain of  $f$  is  $(0, \infty)$ ). With some experimentation of initial conditions, you should see that Newton's method converges to 3.1462 if  $x_0 > 1$ , and it converges to 0.15859 if  $x_0$  is between 0 and  $b$ , where  $b \approx 0.36$ . If  $x_0$  is between 0.37 and 1, the iterates end up being complex numbers and when this happens Newton's method will not find a real root.

See [Exercise A.0.25](#) for an example in which Newton's method has a difficult time finding a root.

**Remark:** Note that we may also use Newton's method to approximate local maxima and minima of a function  $f(x)$ . Recall that we may seek such extrema by setting  $f'(x) = 0$ . In other words, we seek roots of  $f'(x)$ . Applying Newton's method to  $f'$ , we get  $x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$ .

**Secant Method**

The secant method is similar to Newton's method but does not require knowing  $f'$ . We approximate  $f'$  at step  $n$  by using

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \quad (3.3)$$

Using Newton's Method along with (3.3) as an approximation for  $f'$  is called the **secant method**.

To solve  $f(x) = 0$ , where  $f$  is some differentiable function we can use the following pseudocode,

**Secant Method Algorithm to solve  $f(x) = 0$** 

```

choose initial guesses  $x_0, x_1$ 
 $n := 1$ 
loop until some convergence criterion is met, or until  $n$  is too large
     $x_{n+1} := x_n - f(x_n) \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$ 
     $n := n + 1$ 
end loop
output  $x_n$ 

```

See [Exercise A.0.26](#) for an example of how to find the first few iterates using the secant method.

**Summary of MATLAB commands**

MATLAB syntax	What it does
<code>sign(x)</code>	-1 if $x < 0$ , 0 if $x = 0$ , and 1 if $x > 0$
<code>break</code>	Forces MATLAB to exit a while or for loop
<code>return</code>	Forces MATLAB to exit the entire script or function
<code>(CTRL+C)</code>	Exits execution of code

### 3.5 Exercises

- Can one use a fixed-point iteration to approximate a root of  $f(x) = \ln(x) + \sin x$ ? What happens if you try to do this using FixedPoint.m in MATLAB? Recall in MATLAB, we use the command `log(x)` for  $\ln x$ .
- (a) Show that  $g(x) = (3 + x - 2x^2)^{1/4}$  has a fixed point equal to the root of the function  $f(x) = x^4 + 2x^2 - x - 3$ .  
(b) Perform four iterations using  $g(x)$  to approximate a root of  $f(x)$ , with an initial guess  $x_0 = 1$ .
- Use fixed point iteration in MATLAB to approximate  $\sqrt[3]{25}$  that is accurate to within  $10^{-4}$ . Hint: Define a function with a root or fixed point at  $\sqrt[3]{25}$ .
- Determine a function  $g$  and an interval  $[a, b]$  on which the fixed point iteration will converge to a positive solution to the equation  $e^x - 3x^2 = 0$ .
- Write a MATLAB function with arguments  $x_0, x_1$  that approximates roots of  $f(x)$  using the secant method, where  $x_0$  and  $x_1$  are the initial guesses.
- Use both Newton's method and the Secant method to estimate solutions in the desired range, accurate to within  $10^{-5}$ .
  - $x^3 - 2x^2 - 5 = 0, 1 \leq x \leq 4$
  - $e^x + 2^{-x} + 2 \cos x - 6 = 0, 1 \leq x \leq 2$
  - $e^x - 3x^2 = 0, 0 \leq x \leq 1, \text{ and } 3 \leq x \leq 5$
- Consider the problem of estimating the point on the graph of  $y = x^2$  that is closest to  $(1, 0)$ . Write a function that expresses the square of the distance between  $(1, 0)$  and the point  $(x, y)$  on the curve as a function of  $x$ . Then use Newton's method to approximate the point.
- The equation  $x^2 - 10 \cos x = 0$  has two solutions. Use Newton's method with different initial guesses to approximate both of these solutions. What range of initial guesses can be used to find each root?
- Use Newton's method to find all solutions accurate to within  $10^{-4}$  for the equation  $x^3 + 3x^2 - 1 = 0$  in the interval  $[-3, 2]$ . Repeat with the secant method.
- Newton's method works particularly well for approximating square roots. To solve  $x = \sqrt{M}$ , we seek zeros of the function  $f(x) = x^2 - M$ .
  - Does  $f$  satisfy the conditions of Theorem 3.4.1 for any  $M$ ? Explain why or why not.
  - Write a MATLAB function with argument  $M$  that approximates the square root of  $M$  to the nearest six digits using Newton's method.
- Let  $f(x) = -x^3 - \cos x$ 
  - Using  $x_0 = -1$ , apply Newton's method to find  $x_1$ . Could  $x_0 = 0$  be used instead?
  - Using  $x_0 = -1, x_1 = 0$ , use the secant method to find  $x_2$ .
- Use Newton's method to approximate the minimum value of  $f(x) = x^2 + e^x$ , and where this minimum value occurs.
- Let  $f(x) = \cos(x) + 2^{-x}$ . Use the Intermediate Value Theorem to show that  $f$  has a root on the interval  $[1, 4]$ . Then, use two iterations of the bisection method to approximate the root. For each iteration, give the value of  $a, b, x_n, f(a), f(b)$ , and  $f(x_n)$ .
- Write a MATLAB function that takes in two arguments,  $a$  and  $b$ , and applies the Bisection method to approximate a root to a function  $f(x)$ .

15. Use your code for the Bisection method to find roots accurate to within  $10^{-5}$  for the following functions, on the corresponding intervals. Give the value of each  $a$  and  $b$  you used, and check that  $f(a)$  and  $f(b)$  have opposite signs.
- (a)  $f(x) = x - 2^{-x}, 0 \leq x \leq 1$
- (b)  $f(x) = e^x - x^2 + 3x - 2, 0 \leq x \leq 1$
- (c)  $f(x) = 2x \cos(2x) - (x + 1)^2, -3 \leq x \leq -2$  and  $-1 \leq x \leq 0$
16. Use Theorem 3.3.1 to find a bound for the number of iterations needed to achieve an approximation with accuracy  $10^{-3}$  to the solution of  $x^3 + x - 4 = 0$  lying in the interval  $[1, 4]$ . Find an approximation to the root with this degree of accuracy.
17. Let  $f(x) = (x - 1)^{10}$  (note that  $x = 1$  is a root). Also consider the sequence  $x_n = 1 + \frac{1}{n}$  for positive integers  $n$ , which converges to  $x = 1$ . Show that  $|f(x_n)| < 10^{-3}$  whenever  $n > 1$ , but that  $|1 - x_n| < 10^{-3}$  requires that  $n > 1000$ .
18. Show that the sequence  $x_n = \frac{1}{n^k}$  converges linearly to zero for any positive integer  $k$ . Given a pair of positive integers  $k$  and  $m$ , determine a number  $N$  for which  $\frac{1}{N^k} < 10^{-m}$ .
19. Show that the sequence  $x_n = 10^{-2^n}$  converges quadratically to zero.
20. Suppose that the rate of resource gain for a forager in a given area is given by the function  $g(t) = \frac{\ln(t+0.5)}{t}$ , where  $t$  is the amount of time spent foraging in this area. Use Newton's method to determine the value of  $t$  that maximizes  $g(t)$  and explain what this represents in terms of the forager.
21. Write a MATLAB function that implements the Euclidean algorithm. This function will take two positive integers,  $a$  and  $b$ , and output the greatest common divisor. (Do not use the built-in `gcd` function except to check your code.)
22. For an extra challenge, consider the Mandelbrot set, a set of points in the complex plane that satisfy a certain property. Consider the sequence  $z_{n+1} = z_n^2 + c, z_0 = 0$ . A complex number  $c$  belongs to the Mandelbrot set if the norm  $|z_n|$  does not diverge to infinity. Write a MATLAB program that plots a grid of points in the Mandelbrot set in the region  $[-2, 2] \times [-2, 2]$  of the complex plane. (Note that the complex number  $z = a + bi$  is plotted with the point  $(a, b)$ , where the real component is plotted on the  $x$ -axis and the complex coefficient is plotted on the  $y$ -axis.) Use at least 100 gridpoints for both  $x$  and  $y$  (so that you are testing a total of  $100 \cdot 100$  points). [Tips: Use `norm(z)` to compute the absolute value of a complex number `z`. In MATLAB, the letter `i` is reserved to be  $i = \sqrt{-1}$ , so you may want to avoid using `i` as a variable in your code. You can make a nicer picture if you color code according to how many iterations it takes to diverge to some arbitrarily high number.]

# Chapter 4

## Matrices

Systems of linear equations arise in all areas of mathematical modeling. Matrices give a structured way to solve and analyze these linear systems. In this chapter, we offer a brief overview of matrices, how to use MATLAB to work with them, and also give some specific matrix models. If you have taken a course in linear or matrix algebra, much of the mathematical content will be review.

### 4.1 Matrix Review

An  $m \times n$  **matrix** is a rectangular array of numbers with  $m$  rows and  $n$  columns, which may be written in the form

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \vdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix},$$

where each  $a_{ij} \in \mathbb{R}$ . We say that  $A \in \mathbb{R}^{m \times n}$ . Note that the notation  $a_{ij}$  refers to the element in row  $i$ , column  $j$  of matrix  $A$ .

Suppose we want to work with the  $2 \times 3$  matrix  $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ . Recall from Chapter 2, we enter the matrix by using the command:

```
A=[1 2 3; 4 5 6]
```

The command `A(n,m)` returns  $a_{nm}$  (the entry in the  $n^{\text{th}}$  row and  $m^{\text{th}}$  column). Entering `A(n,:)` returns the  $n^{\text{th}}$  row of  $A$ , and entering `A(:,m)` returns the  $m^{\text{th}}$  column of  $A$ .

The **transpose** of a matrix is the new matrix obtained from swapping its rows with its columns. We denote the transpose of a matrix  $A$  by  $A^T$ . In MATLAB, we use the command `A'` to return the transpose of the matrix  $A$ . Using matrix  $A$  above,  $A'$  is a  $3 \times 2$  matrix equivalent to

```
A' = [1 4; 2 5; 3 6]
```

A **vector** is a matrix with only one column, or one row. In this textbook, we will refer to vectors as matrices with a single column; for example  $\mathbf{v} = \begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix}$ . In order to save space on paper, we may also use the notation  $\mathbf{v} = (1, 4, 7)^T$ . To assign this column vector to a variable in MATLAB, use the command



$$\mathbf{v} = [1; 4; 7]$$

Note that since semicolons are used to separate rows, the command exactly matches that of matrices. To return the second element of the vector, either use the command `v(2,1)`, or simply `v(2)` for short. To enter the row vector  $\mathbf{w} = [2 \ 5 \ 8]$ , use the command

$$\mathbf{w} = [2 \ 5 \ 8]$$

To return the second element of the row vector, either use the command `v(1,2)`, or simply `v(2)`. Vectors are often denoted using either boldface lowercase letters ( $\mathbf{v}$ ), or a letter with an arrow above it ( $\vec{v}$ ). If  $\mathbf{v}$  is an element with  $n$  elements, we may write  $\mathbf{v} \in \mathbb{R}^n$ .

The **zero vector** is the vector consisting of all zeros, and we denote it by  $\mathbf{0} = (0, 0, \dots, 0)^T$ .

The **length** or **magnitude** of a vector  $\mathbf{u} = (u_1, u_2, \dots, u_n)^T$  is defined as the real number  $|\mathbf{u}| = \sqrt{u_1^2 + u_2^2 + \dots + u_n^2}$ .

We may add or subtract matrices and vectors if their sizes are equal. In this case, we add or subtract the corresponding entries. We may also multiply matrices by numbers, which are often called **scalars** in the context of matrices. This operation is called **scalar multiplication**. To do this, we multiply the scalar by every entry in the matrix or vector. See [Exercise A.0.27](#) for an example.

## Linear Equations

Suppose we want to solve the following system of three equations and three variables:

$$\begin{array}{rclcl} -3x_1 & - & x_2 & + & 2x_3 & = & -3 & (i) \\ 5x_1 & + & 4x_2 & + & 3x_3 & = & -2 & (ii) \\ x_1 & + & x_2 & + & x_3 & = & -1 & (iii) \end{array}$$

Using the elimination method, we can

- Multiply an equation by a constant (usually to match up one of the coefficients of the system of equations).
- Add two equations together (usually to eliminate one of the variables).

We will use a process called **Gaussian elimination** to solve the system by using matrices. Each step in the process makes use of the operations listed above, resulting in a new system with the same solution set as the original system. Additional properties for solving a matrix equation will be given later in this section. For now we consider the **augmented matrix** of the system. It consists of the coefficient matrix with the constants from the right sides of the equation on the rightmost column.

$$\left[ \begin{array}{ccc|c} -3 & -1 & 2 & -3 \\ 5 & 4 & 3 & -2 \\ 1 & 1 & 1 & -1 \end{array} \right] \quad (4.1)$$

Keep in mind that each row in an augmented matrix corresponds to an equation involving the variables  $x_1, x_2, x_3$ . There are three elementary row operations we can do which do not change the solution set of the system:

1. (Scaling) Multiply all entries in a row by a nonzero constant. We may do this since we may multiply both sides of an equation by any nonzero number.
2. (Interchange) Interchange two rows. We may do this since the order in which the equations appear makes no difference.
3. (Replacement) Replace one row by the sum of itself and a multiple of another row. We may do this since we may multiply both sides of an equation by a number and we can also add equations together.

Our goal is to use the row operations above until we can easily find solutions to  $x_1, x_2$ , and  $x_3$ . The first objective is to obtain a **triangular system**, which has zeros in the lower left triangle:

$$\left[ \begin{array}{ccc|c} b_{11} & b_{12} & b_{13} & d_1 \\ 0 & b_{22} & b_{23} & d_2 \\ 0 & 0 & b_{33} & d_3 \end{array} \right].$$

This matrix is said to be in **row echelon form**.

For our example, we will start at the leftmost column. First make sure that the top-left element is nonzero, which it is (otherwise swap two rows so that it is not). We need zeros underneath the -3 in the top left entry, and this may be done by using a replacement operation. However, in order to avoid fractions, we could swap rows 1 and 3, resulting in a 1 in the top left position. Starting with (4.1),

$$\left[ \begin{array}{ccc|c} -3 & -1 & 2 & -3 \\ 5 & 4 & 3 & -2 \\ 1 & 1 & 1 & -1 \end{array} \right] \xRightarrow{R_1 \leftrightarrow R_3} \left[ \begin{array}{ccc|c} 1 & 1 & 1 & -1 \\ 5 & 4 & 3 & -2 \\ -3 & -1 & 2 & -3 \end{array} \right].$$

In order to get zeros below the top left entry, we multiply the top row by -5 and add it to row 2, replacing row 2 with the result; then multiply the top row by 3 and add it to row 3, replacing row 3 with the result. Notice row 1 does not change.

$$\left[ \begin{array}{ccc|c} 1 & 1 & 1 & -1 \\ 5 & 4 & 3 & -2 \\ -3 & -1 & 2 & -3 \end{array} \right] \xRightarrow{\substack{-5R_1+R_2 \rightarrow R_2 \\ 3R_1+R_3 \rightarrow R_3}} \left[ \begin{array}{ccc|c} 1 & 1 & 1 & -1 \\ 0 & -1 & -2 & 3 \\ 0 & 2 & 5 & -6 \end{array} \right]$$

We now focus on the leading nonzero entry of the second row, and attempt to get zeros below this element.

$$\left[ \begin{array}{ccc|c} 1 & 1 & 1 & -1 \\ 0 & -1 & -2 & 3 \\ 0 & 2 & 5 & -6 \end{array} \right] \xRightarrow{2R_2 + R_3 \rightarrow R_3} \left[ \begin{array}{ccc|c} 1 & 1 & 1 & -1 \\ 0 & -1 & -2 & 3 \\ 0 & 0 & 1 & 0 \end{array} \right] \quad (4.2)$$

The matrix is now in row echelon form. We could either reduce the matrix further, or determine  $x_1, x_2, x_3$  by **back solving**. To solve by back solving, recall that each row of the matrix corresponds with an equation,

$$\begin{array}{rclcl} x_1 & + & x_2 & + & x_3 & = & -1 \\ & & -x_2 & - & 2x_3 & = & 3 \\ & & & & x_3 & = & 0 \end{array} .$$

Since  $x_3 = 0$ , substituting this into the second equation gives  $-x_2 - 2 \cdot 0 = 3$ , so  $x_2 = -3$ . Finally, substituting  $x_2 = -3$  and  $x_3 = 0$  into the top equation gives  $x_1 + (-3) + 0 = -1$ , and  $x_1 = 2$ . Thus we obtain the solution set  $x_1 = 2, x_2 = -3, x_3 = 0$ .

As an alternative to back solving, we could have instead reduced the matrix further, to get zeros above the leading nonzero entries of each row as well as zeros below. This gives a matrix in the form:

$$\left[ \begin{array}{ccc|c} c_{11} & 0 & 0 & e_1 \\ 0 & c_{22} & 0 & e_2 \\ 0 & 0 & c_{33} & e_3 \end{array} \right].$$

If these leading entries  $c_{11}, c_{22}, c_{33}$  are all equal to 1, it is said that the matrix is in **reduced row echelon form** (rref). At this point, the solution to the system is simply the elements of the fourth column.

To get the matrix into this form, starting from (4.2) we could multiply the second row by -1 in order to make all the leading row entries equal to 1.

$$\left[ \begin{array}{ccc|c} 1 & 1 & 1 & -1 \\ 0 & -1 & -2 & 3 \\ 0 & 0 & 1 & 0 \end{array} \right] \xRightarrow{-R_2 \rightarrow R_2} \left[ \begin{array}{ccc|c} 1 & 1 & 1 & -1 \\ 0 & 1 & 2 & -3 \\ 0 & 0 & 1 & 0 \end{array} \right].$$

Next we start with the rightmost column to get zeros above the 1 in entry  $b_{33}$ ,

$$\left[ \begin{array}{ccc|c} 1 & 1 & 1 & -1 \\ 0 & 1 & 2 & -3 \\ 0 & 0 & 1 & 0 \end{array} \right] \xrightarrow{\substack{-R_3+R_1 \rightarrow R_1 \\ -2R_3+R_2 \rightarrow R_2}} \left[ \begin{array}{ccc|c} 1 & 1 & 0 & -1 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 0 \end{array} \right].$$

To finish, we get zeros above the leading 1 in the second row,

$$\left[ \begin{array}{ccc|c} 1 & 1 & 0 & -1 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 0 \end{array} \right] \xrightarrow{-R_2 + R_1 \rightarrow R_1} \left[ \begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 0 \end{array} \right].$$

Again, remember that each row of the matrix corresponds with an equation,

$$\begin{array}{rcl} x_1 & & = 2 \\ & x_2 & = -3 \\ & & x_3 = 0 \end{array}$$

We see the solution set appears as the elements of the rightmost column,  $x_1 = 2, x_2 = -3, x_3 = 0$ , just as we found before with back solving.

## Summary of Echelon Forms

A rectangular matrix is in **row echelon form** (ref) if it has the following three properties:

1. All nonzero rows are above any rows of all zeros.
2. Each leading entry of a row is in a column to the right of the leading entry of the row above it.
3. All entries in a column below a leading entry are zero. (In other words, the nonzeros fill in an upper-right triangular region of the matrix.)

If a matrix in row echelon form satisfies the following additional conditions, then it is in **reduced row echelon form** (rref):

4. The leading entry in each nonzero row is 1.
5. Each leading 1 is the only nonzero entry in its column.

It can be shown that each matrix is row equivalent to one and only one reduced echelon matrix.

To reduce the matrix to reduced row echelon form in MATLAB, use `rref()`. For example, to row reduce (4.1):

```
A = [-3 1 2 -3; 5 4 3 -2; 1 1 1 -1]
rref(A)
```

---

### Example 4.1.1

Solve the following systems of equations

(a)

$$\begin{array}{rcl} x_1 - 2x_2 & = & 3 \\ -2x_1 + 4x_2 & = & 7 \end{array}$$

(b)

$$\begin{array}{rcl} x_1 - 2x_2 + 2x_3 & = & -6 \\ -x_1 + 3x_2 + 4x_3 & = & 3 \end{array}$$

For (a), first note the augmented form of the system  $\left[ \begin{array}{cc|c} 1 & -2 & 3 \\ -2 & 4 & 7 \end{array} \right]$ . To solve, we could use the MATLAB command `rref([1 -2 3; -2 4 7])`. This gives the row reduced echelon form of the matrix,  $\left[ \begin{array}{cc|c} 1 & -2 & 0 \\ 0 & 0 & 1 \end{array} \right]$ . This results in the system of equations

$$\begin{array}{rcl} x_1 & - & 2x_2 = 0 \\ 0x_1 & + & 0x_2 = 1 \end{array}.$$

The problem is that the bottom equation states that  $0 = 1$ , a contradiction. In other words, if the original set of equations is true, then  $0 = 1$ , which is impossible. So the original set of equations cannot be true for any  $x_1, x_2$ . In fact, if one plots these two equations in the  $x_1 - x_2$  plane, we see that the two lines are parallel and hence never intersect. We say there is no solution to the system.

In part (b), the system is an example of an **under-determined** system, since there are fewer equations than variables. (An **over-determined** system has more equations than variables). Beginning with the augmented matrix  $\left[ \begin{array}{ccc|c} 1 & -2 & 2 & -6 \\ -1 & 3 & 4 & 3 \end{array} \right]$ , the row reduced echelon form of the system is

$$\left[ \begin{array}{ccc|c} 1 & 0 & 14 & -12 \\ 0 & 1 & 6 & 3 \end{array} \right],$$

resulting in the system of equations

$$\begin{array}{rcl} x_1 + 14x_3 & = & -12 \\ x_2 + 6x_3 & = & 3. \end{array}$$

There are infinitely many solutions, and we can describe them by finding  $x_1$  and  $x_2$  in terms of  $x_3$ , giving

$$\begin{array}{l} x_1 = -14x_3 - 12 \\ x_2 = -6x_3 + 3 \\ x_3 \in \mathbb{R} \end{array}$$

We say that  $x_3$  is a **free variable**, and we may set it to any number we like, as long as  $x_1$  and  $x_2$  satisfy the equations above. For example, if we choose  $x_3 = 1$ , we get  $x_1 = -14 \cdot 1 - 12 = -26$ , and  $x_2 = -6 \cdot 1 + 3 = -3$ . So  $(-26, -3, 1)$  is one of the infinitely many solutions.

See [Exercise A.0.28](#) for additional examples of solving systems using MATLAB.

## Matrix-Vector Multiplication

Let  $A$  be an  $m \times n$  matrix, and let  $\mathbf{x} \in \mathbb{R}^n$  be a column vector of length  $n$ . The product of  $A$  and  $\mathbf{x}$  is the linear combination of the columns of  $A$  using the corresponding entries in  $\mathbf{x}$  as weights, as shown below:

$$\mathbf{Ax} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \vdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}$$

Note that the number of columns of  $A$  must equal the number of entries in  $\mathbf{x}$ . The result,  $\mathbf{Ax}$ , is a column vector in  $\mathbb{R}^m$ .

See [Exercises A.0.29](#) and [A.0.30](#) for examples.

Here are some useful rules about matrix-vector multiplication. Let  $A \in \mathbb{R}^{m \times n}$ ,  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ ,  $\alpha \in \mathbb{R}$ , and  $\mathbf{0} = (0, 0, \dots, 0)^T \in \mathbb{R}^n$ . Then,

- $A(\mathbf{u} + \mathbf{v}) = A\mathbf{u} + A\mathbf{v}$
- $A(\alpha\mathbf{u}) = \alpha(A\mathbf{u})$
- $A \cdot \mathbf{0} = \mathbf{0}$

Multiplication of two matrices  $A$  and  $B$  uses successive matrix-vector multiplication of the first matrix  $A$  by the columns of the second matrix  $B = [\mathbf{b}_1 \ \mathbf{b}_2 \ \dots \ \mathbf{b}_r]$ . Each column of the product  $AB$  will be the product of matrix  $A$  by the corresponding column vectors in  $B$ ,  $AB = [A\mathbf{b}_1 \ A\mathbf{b}_2 \ \dots \ A\mathbf{b}_r]$ . Notice each column in the product will then have the same number of rows as  $A$ . This implies the condition for matrix multiplication such that the number of columns of  $A$  must match the number of rows of  $B$ . In other words, for  $A \in \mathbb{R}^{m \times n}$ ,  $B \in \mathbb{R}^{n \times r}$  we have the resulting product  $AB \in \mathbb{R}^{m \times r}$ . Because of this condition, note that in general  $AB$  may exist while  $BA$  may not. If both products do exist, it is not clear that the two will be equivalent. See the notes on Special Matrices below.

---

**Example 4.1.2**

Consider the  $2 \times 3$  matrix  $A = \begin{bmatrix} 1 & -2 & 1 \\ 3 & 0 & 2 \end{bmatrix}$  multiplied by the  $3 \times 4$  matrix  $B = \begin{bmatrix} 2 & 3 & 1 & -5 \\ -1 & 1 & -2 & 3 \\ -4 & 2 & -1 & 1 \end{bmatrix}$ .

The first column of the product  $AB$  is given by  $A\mathbf{b}_1 = A \begin{bmatrix} 2 \\ -1 \\ -4 \end{bmatrix} = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$ .

The second column of the product is  $A\mathbf{b}_2 = A \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 13 \end{bmatrix}$ .

Repeating the process for the remaining columns of  $B$  gives the  $2 \times 4$  matrix  $AB = \begin{bmatrix} 0 & 3 & 4 & -10 \\ -2 & 13 & 1 & -13 \end{bmatrix}$ .

Note that  $BA$  does not exist since  $B \in \mathbb{R}^{3 \times 4}$  and  $A \in \mathbb{R}^{2 \times 3}$  do not have matching column-row dimensions. Matrix  $B$  has 4 columns while matrix  $A$  has 2 rows.

---

## Special Matrices

- A matrix is a **square matrix** if the number of rows equals the number of columns. (If  $A \in \mathbb{R}^{n \times n}$ , then  $A$  is square.)
- A square matrix is a **diagonal matrix** if all elements are zero except possibly those on the main diagonal.

Example:  $A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -3 \end{bmatrix}$ .

- The **identity matrix** of size  $n$  is the  $n \times n$  diagonal matrix with all entries as 1, denoted  $I_n$ .

If  $A$  is a square matrix, then we use the notation  $A^n = A \cdot A \dots A$  (i.e.  $A^2 = A \cdot A$ ,  $A^3 = A \cdot A \cdot A$ , etc.) See [Exercise A.0.31](#) for an example of how to square matrices in MATLAB.

## Matrix Inverses

**Definition 4.1.3.** Let  $A$  be a  $n \times n$  matrix, and let  $I_n$  be the  $n \times n$  identity matrix. Suppose there exists a  $n \times n$  matrix  $C$  such that  $AC = I_n = CA$ . Then  $A$  is **invertible**, or **nonsingular**, and  $C$  is the inverse of  $A$ . (We also say that  $A$  is the inverse of  $C$ ). A matrix that is not invertible is called **singular**.

To denote the inverse of a matrix  $A$ , we use the notation  $A^{-1}$ , so that  $AA^{-1} = I_n$ , and  $A^{-1}A = I_n$ . Knowing the inverse of a matrix is useful for solving systems of equations where the number of variables is equal to the number of equations. In particular, suppose  $A$  is an invertible  $n \times n$  matrix and  $\mathbf{b}$  is a vector in  $\mathbb{R}^n$ . In order to solve the matrix equation  $A\mathbf{x} = \mathbf{b}$ , we can multiply both sides on the left by  $A^{-1}$  to get  $A^{-1}A\mathbf{x} = A^{-1}\mathbf{b}$ . This simplifies to  $I_n\mathbf{x} = A^{-1}\mathbf{b}$ , or simply to  $\mathbf{x} = A^{-1}\mathbf{b}$ . Recall that matrix multiplication is not commutative in general. This is why we must be consistent about which side to multiply by  $A^{-1}$ .

There are several methods to compute an inverse. You may recall one method from linear algebra given by the following theorem.

**Theorem 4.1.4.** *If  $A$  is invertible, and we reduce the augmented matrix  $[A|I]$  to reduced row echelon form of  $A$ , the result will be the augmented matrix  $[I|A^{-1}]$ .*

In the  $2 \times 2$  case, you can show the inverse of  $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$  is given by  $A^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$ .

Alternatively, we can use the MATLAB command `inv(A)` to compute the inverse of a matrix  $A$ .

The determinant of a square matrix describes properties about the linear system whose coefficients are determined by the matrix. In particular, the determinant can tell us whether the matrix is invertible. The rules for computing determinants are typically covered in a course in linear algebra, however the  $2 \times 2$  case is simple enough that we provide the definition here.

**Definition 4.1.5.** *The **determinant** of a  $2 \times 2$  matrix is defined as*

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc.$$

We will use MATLAB to compute determinants, saving the definition for a linear algebra course. In MATLAB, we use the command `det(A)` to compute the determinant of a matrix  $A$ .

We will soon be using matrices to model linear systems and other scenarios. When using a linear system to model a situation, we may first want to know whether there is a solution, and if so, whether it is unique. For a square system (meaning that the number of equations equals the number of variables), it turns out that there are several ways to check whether a system has a solution. The following theorem gives several ways to check this. If any one of the five following conditions in the theorem are true, then the other four statements must hold true as well.

**Theorem 4.1.6.** *Let  $A$  be a  $n \times n$  matrix. Then the following statements are equivalent.*

1.  $A$  is invertible
2.  $\det(A) \neq 0$
3.  $A\mathbf{x} = \mathbf{0}$  has only the trivial solution  $\mathbf{x} = \mathbf{0}$ .
4.  $A\mathbf{x} = \mathbf{b}$  has a unique solution for all  $\mathbf{b} \in \mathbb{R}^n$
5.  $A$  can be row reduced to the identity matrix

We may also characterize the non-invertible matrices using the following theorem.

**Theorem 4.1.7.** *Let  $A$  be a  $n \times n$  matrix. Then the following statements are equivalent.*

1.  $A$  is not invertible
2.  $\det(A) = 0$
3.  $A\mathbf{x} = \mathbf{0}$  has some nontrivial solution  $\mathbf{x} \neq \mathbf{0}$ .

4.  $A\mathbf{x} = \mathbf{b}$  either has no solution or infinitely many solutions, depending on the choice of  $\mathbf{b}$ .
5.  $A$  does not row reduce to the identity matrix (it row reduces to a matrix with all zeros in the bottom row).

Refer to **Exercise A.0.32** for an example. Notice in the  $2 \times 2$  case, the coefficient of  $A^{-1}$  is  $\frac{1}{\det(A)}$ . If  $\det(A) = 0$ , the inverse of  $A$  is undefined and according to the above theorems, the equation  $A\mathbf{x} = \mathbf{b}$  has no solution, or infinitely many.

**Example 4.1.8**

Solve the matrix equation  $A\mathbf{x} = \mathbf{b}$ , where

$$A = \begin{bmatrix} 1 & 2 & -3 & 1 \\ 3 & 0 & 1 & 4 \\ -1 & 7 & 4 & 2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}.$$

Note that  $A^{-1}$  cannot exist, since  $A$  is not a square matrix. Alternatively, we will row reduce the augmented matrix  $[A|\mathbf{b}]$ . Using MATLAB,

$$\left[ \begin{array}{cccc|c} 1 & 2 & -3 & 1 & 1 \\ 3 & 0 & 1 & 4 & 0 \\ -1 & 7 & 4 & 2 & -2 \end{array} \right] \xrightarrow{\text{rref}} \left[ \begin{array}{cccc|c} 1 & 0 & 0 & 1.2396 & 0.1146 \\ 0 & 1 & 0 & 0.3021 & -0.0729 \\ 0 & 0 & 1 & 0.2813 & -0.3438 \end{array} \right].$$

This gives us

$$\begin{aligned} x_1 + 1.2396x_4 &= 0.1146 \\ x_2 + 0.3021x_4 &= -0.0729 \\ x_3 + 0.2813x_4 &= -0.3438 \end{aligned}$$

We see that  $x_4$  appears in multiple equations, so we choose it as the free variable. We write  $x_1, x_2, x_3$  in terms of  $x_4$ , giving

$$\begin{aligned} x_1 &= -1.2396x_4 + 0.1146 \\ x_2 &= -0.3021x_4 - 0.0729 \\ x_3 &= -0.2813x_4 - 0.3438 \\ x_4 &\in \mathbb{R} \end{aligned}$$

We may alternatively express the solution in vector form,

$$\begin{aligned} \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} &= \begin{bmatrix} -1.2396x_4 + 0.1146 \\ -0.3021x_4 - 0.0729 \\ -0.2813x_4 - 0.3438 \\ x_4 \end{bmatrix} = \begin{bmatrix} -1.2396x_4 \\ -0.3021x_4 \\ -0.2813x_4 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.1146 \\ -0.0729 \\ -0.3438 \\ 0 \end{bmatrix} \\ &= x_4 \begin{bmatrix} -1.2396 \\ -0.3021 \\ -0.2813 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.1146 \\ -0.0729 \\ -0.3438 \\ 0 \end{bmatrix}. \end{aligned}$$

**Example 4.1.9**

Let  $C = \begin{bmatrix} 1 & 3 & 4 \\ -4 & 2 & -6 \\ -3 & -2 & -7 \end{bmatrix}$ , and  $\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$ . Does  $C\mathbf{x} = \mathbf{d}$  have a solution for all possible  $d_1, d_2, d_3$ ?

We see that using the MATLAB command `inv([1 3 4; -4 2 -6; -3 -2 -7])` results in:

Warning: Matrix is singular to working precision.

ans =

```
Inf    Inf    Inf
Inf    Inf    Inf
Inf    Inf    Inf
```

This means the algorithm that MATLAB used to find the inverse failed, most likely due to the fact that the matrix is not invertible. If we row reduce the matrix, we see that

$$\begin{bmatrix} 1 & 3 & 4 \\ -4 & 2 & -6 \\ -3 & -2 & -7 \end{bmatrix} \xrightarrow{\text{rref}} \begin{bmatrix} 1 & 0 & 1.8571 \\ 0 & 1 & 0.7143 \\ 0 & 0 & 0 \end{bmatrix}.$$

It turns out that for most choices of  $\mathbf{d}$ , there is no solution to  $C\mathbf{x} = \mathbf{d}$ . For example, if  $\mathbf{d} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ , we

have

$$\left[ \begin{array}{ccc|c} 1 & 3 & 4 & 1 \\ -4 & 2 & -6 & 1 \\ -3 & -2 & -7 & 1 \end{array} \right] \xrightarrow{\text{rref}} \left[ \begin{array}{ccc|c} 1 & 0 & 1.8571 & 0 \\ 0 & 1 & 0.7143 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right].$$

The bottom row provides a contradiction, indicating there is no solution.

## 4.2 Eigenvectors and Eigenvalues

In this section we will learn about eigenvectors and eigenvalues, and several of their properties. If we think of a matrix  $A$  acting on a vector  $\mathbf{v}$  by the operation  $A\mathbf{v}$ , the eigenvalues and eigenvectors say something about how  $A\mathbf{v}$  relates to  $\mathbf{v}$  in a geometric context. A course in linear algebra would discuss these connections further, so we will focus on the algebraic definitions in this discussion.

**Definition 4.2.1.** An *eigenvector* for the square matrix  $A$  is a nonzero vector  $\mathbf{v}$  such that  $A\mathbf{v} = \lambda\mathbf{v}$  for some  $\lambda \in \mathbb{R}$ . The number  $\lambda$  is called the *eigenvalue* associated with the eigenvector  $\mathbf{v}$ .

### Example 4.2.2

It turns out that  $A = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix}$  has two eigenvectors:  $\mathbf{v}_1 = (-1, 2)^T$  and  $\mathbf{v}_2 = (1, 1)^T$ . Find the corresponding eigenvalues  $\lambda_1$  and  $\lambda_2$ .

If  $\mathbf{v}_1$  is indeed an eigenvector, there should be some  $\lambda_1$  such that  $A\mathbf{v}_1 = \lambda_1\mathbf{v}_1$ . That is,

$$\begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \lambda_1 \begin{bmatrix} -1 \\ 2 \end{bmatrix}.$$

Simplifying the left hand side, we get

$$\begin{bmatrix} 1 \\ -2 \end{bmatrix} = \lambda_1 \begin{bmatrix} -1 \\ 2 \end{bmatrix},$$

which means that  $\lambda_1$  must be -1. To find  $\lambda_2$ , we notice that

$$\begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} = 2 \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

so  $\lambda_2 = 2$ .



**Proposition 4.2.3.** *If  $\mathbf{v}$  is an eigenvector associated with the eigenvalue  $\lambda$ , then  $\alpha\mathbf{v}$  is also an eigenvector associated with  $\lambda$ , for any  $\alpha \in \mathbb{R}$ .*

The proposition above says that since eigenvectors can be of any length, they may be characterized only by their direction. For example,  $(0.5, -1)^T$  is the same eigenvector as  $(-1, 2)^T$  in the previous example.

**Example 4.2.4**

Suppose  $A$  is a diagonal matrix of the form  $A = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix}$ . Determine the eigenvalues and eigenvectors.

.....

We would like to find  $\lambda$  and  $\mathbf{v} = (x_1, x_2, x_3)^T$  such that  $A\mathbf{v} = \lambda\mathbf{v}$ . Notice that

$$A\mathbf{v} = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} d_1x_1 \\ d_2x_2 \\ d_3x_3 \end{bmatrix}, \text{ and } \lambda\mathbf{v} = \begin{bmatrix} \lambda x_1 \\ \lambda x_2 \\ \lambda x_3 \end{bmatrix}.$$

We could first equate the top element of each vector, so that  $d_1x_1 = \lambda x_1$ , suggesting that  $\lambda = d_1$ . In order for the other two elements to be equal, we simply set  $x_2 = 0, x_3 = 0$ . In mind of Proposition 4.2.3, we could arbitrarily set  $x_1 = 1$ , giving the eigenvalue  $\lambda_1 = d_1$  with corresponding eigenvector  $\mathbf{v}_1 = (1, 0, 0)^T$ . Applying a similar process with the other elements of the vector, we would find that  $\lambda_2 = d_2, \mathbf{v}_2 = (0, 1, 0)^T$  and  $\lambda_3 = d_3, \mathbf{v}_3 = (0, 0, 1)^T$ . To summarize, the eigenvalues of a diagonal matrix are simply the entries on the diagonal, and the eigenvectors are the corresponding columns of the identity matrix.

**Remark:** Solving  $A\mathbf{v} = \lambda\mathbf{v}$  amounts to solving  $A\mathbf{v} - \lambda\mathbf{v} = \mathbf{0}$ , or  $A\mathbf{v} - I\lambda\mathbf{v} = \mathbf{0}$  where  $I$  is the identity matrix of appropriate size. Then we may factor out  $\mathbf{v}$  so that  $(A - \lambda I)\mathbf{v} = \mathbf{0}$ . The solution must be nonzero, since eigenvectors may not be zero. To find the eigenvalues, we may use Theorem 4.1.7,  $(3 \Rightarrow 2)$ , and simply seek solutions of the equation  $\det(A - \lambda I) = 0$ . To find the associated eigenvectors, we plug in each eigenvalue found into the equation  $(A - \lambda I)\mathbf{v} = \mathbf{0}$  and solve for nonzero solutions by row reducing the augmented matrix  $[A - \lambda I | \mathbf{0}]$ .

**Definition 4.2.5.** *We say that  $\det(A - \lambda I) = 0$  is the **characteristic equation** of the square matrix  $A$ .*

**Example 4.2.6**

Find the eigenvalues of  $A = \begin{bmatrix} 3/4 & 1/2 \\ 3/4 & 2 \end{bmatrix}$  by solving the characteristic equation, then find eigenvalues by row reduction.

.....

We want to solve the characteristic equation  $\det(A - \lambda I) = 0$ . First, simplifying  $A - \lambda I$  gives

$$A - \lambda I = \begin{bmatrix} 3/4 & 1/2 \\ 3/4 & 2 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 3/4 - \lambda & 1/2 \\ 3/4 & 2 - \lambda \end{bmatrix}.$$

Using Definition 4.1.5,

$$\det(A - \lambda I) = (3/4 - \lambda)(2 - \lambda) - \frac{1}{2} \cdot \frac{3}{4} = \lambda^2 - \frac{11}{4}\lambda + \frac{9}{8} = (\lambda - 1/2)(\lambda - 9/4).$$

Setting  $A - \lambda I = \mathbf{0}$  gives us the eigenvalues  $\lambda_1 = 1/2, \lambda_2 = 9/4$ . To find the eigenvector associated with  $\lambda_1 = 1/2$ , recall Remark 4.2 which says to row reduce  $[A - \frac{1}{2}I | \mathbf{0}]$ . Simplifying  $A - \frac{1}{2}I$ ,

$$\begin{aligned} A - \frac{1}{2}I &= \begin{bmatrix} 3/4 & 1/2 \\ 3/4 & 2 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 3/4 & 1/2 \\ 3/4 & 2 \end{bmatrix} - \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \\ &= \begin{bmatrix} 3/4 - 1/2 & 1/2 \\ 3/4 & 2 - 1/2 \end{bmatrix} = \begin{bmatrix} 1/4 & 1/2 \\ 3/4 & 3/2 \end{bmatrix}. \end{aligned}$$

Next, row reducing  $[A - \frac{1}{2}I | \mathbf{0}]$  gives

$$\left[ \begin{array}{cc|c} 1/4 & 1/2 & 0 \\ 3/4 & 3/2 & 0 \end{array} \right] \xrightarrow{\text{rref}} \left[ \begin{array}{cc|c} 1 & 2 & 0 \\ 0 & 0 & 0 \end{array} \right].$$

Notice that we get a row of zeros - this should always happen when you are following this procedure. If we let  $\mathbf{v}_1 = (x_1, x_2)^T$ , then the top row tells us that  $1 \cdot x_1 + 2 \cdot x_2 = 0$ , or  $x_1 = -2x_2$ . We may actually set  $x_2$  to any real number (see Proposition 4.2.3), so we arbitrarily set it equal to 1. With  $x_2 = 1$ , then  $x_1 = -2 \cdot 1 = -2$ , giving us the eigenvector  $\mathbf{v}_1 = (-2, 1)^T$ . We repeat this same process to find the other eigenvector  $\mathbf{v}_2$ ,

$$A - \frac{9}{4}I = \begin{bmatrix} 3/4 & 1/2 \\ 3/4 & 2 \end{bmatrix} - \frac{9}{4} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 3/4 - 9/4 & 1/2 \\ 3/4 & 2 - 9/4 \end{bmatrix} = \begin{bmatrix} -3/2 & 1/2 \\ 3/4 & -1/4 \end{bmatrix}.$$

Row reducing as before gives

$$\left[ A - \frac{9}{4}I | \mathbf{0} \right] = \left[ \begin{array}{cc|c} -3/2 & 1/2 & 0 \\ 3/4 & -1/4 & 0 \end{array} \right] \xrightarrow{\text{rref}} \left[ \begin{array}{cc|c} 1 & -1/3 & 0 \\ 0 & 0 & 0 \end{array} \right].$$

Thus, we see that if  $\mathbf{v}_2 = (y_1, y_2)^T$ , then  $y_1 - \frac{1}{3}y_2 = 0$ , or  $y_1 = \frac{1}{3}y_2$ . To avoid fractions, we could set  $y_1 = 1$ , making  $y_2 = 3$ . This gives us the eigenvector  $\mathbf{v}_2 = (1, 3)^T$ .

See [Activity A.0.34](#) for another example.

## Using MATLAB's eig command

For a square matrix, you may use the `eig(A)` command to compute the eigenvalues. If you also want the eigenvectors, you must use the command `[x y] = eig(A)`. This command gives you the eigenvalues and eigenvectors in a rather cryptic form. For example, for a  $2 \times 2$  matrix, the command `[x y] = eig(A)` gives the eigenvectors and eigenvalues of  $A$  in the following format:

$$\begin{aligned} x &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \\ y &= \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}. \end{aligned}$$

Here,  $\lambda_1$  and  $\lambda_2$  are eigenvalues. The eigenvector associated with  $\lambda_1$  is  $\mathbf{v}_1 = \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix}$ , and the eigenvector associated with  $\lambda_2$  is  $\mathbf{v}_2 = \begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix}$ . Again, recall that  $\alpha \mathbf{v}_1$  is also an eigenvector corresponding to  $\lambda_1$  for any  $\alpha$ . By default, MATLAB chooses  $\mathbf{v}_1$  in order to satisfy  $|\mathbf{v}_1|^2 = a_{11}^2 + a_{21}^2 = 1$ . Similarly,  $|\mathbf{v}_2|^2 = a_{12}^2 + a_{22}^2 = 1$ . See [Exercises A.0.33](#) and [A.0.35](#) for examples.

### 4.3 Summary of MATLAB commands

MATLAB command	What it does
$A'$	Computes the transpose of $A$
$A.^n$	Computes the product $A \cdot A \dots A$ (with $n$ factors of $A$ )
$A.^n$	Raises each element of $A$ to the power $n$
$\text{rref}(A)$	Reduces $A$ to row reduced echelon form (does Gaussian elimination)
$\text{inv}(A)$	Computes $A^{-1}$
$A*B$	Computes the matrix product $AB$
$A \setminus b$	Solves the equation $Ax = b$ for $x$ by using Gaussian elimination
$\det(A)$	Computes the determinant of $A$
$\text{eye}(n)$	Produces the $n \times n$ identity matrix, $I_n$ .
$\text{eig}(A)$	Computes the eigenvalues of $A$
$[x \ y] = \text{eig}(A)$	Computes the eigenvectors and eigenvalues of $A$

### 4.4 Exercises

1. Determine if the following systems have solutions. If so, give the solution set. Is the solution unique?

$$(a) \begin{cases} x_1 & & + x_3 & = & 1 \\ x_1 & + & 2x_2 & + & 6x_3 & = & 7 \\ -3x_1 & + & x_2 & - & 4x_3 & = & 0 \end{cases}$$

$$(b) \begin{cases} & x_2 & - & 4x_3 & = & 8 \\ 2x_1 & - & 3x_2 & + & 2x_3 & = & 1 \\ 5x_1 & - & 8x_2 & + & 7x_3 & = & 1 \end{cases}$$

$$(c) \begin{cases} x_1 & - & 2x_2 & + & 8x_3 & = & -5 \\ 2x_1 & - & 3x_2 & + & 14x_3 & = & -7 \\ -x_1 & + & 3x_2 & - & 10x_3 & = & 8 \end{cases}$$

2. Let  $A \in \mathbb{R}^{n \times n}$ ,  $\mathbf{x}_0 \in \mathbb{R}^n$ . Define the sequence  $\mathbf{x}_{n+1} = A\mathbf{x}_n$ ,  $n \geq 1$ .

- (a) Show that  $\mathbf{x}_3 = A^3\mathbf{x}_0$ .  
 (b) Use your answer from (a) to guess an expression for  $\mathbf{x}_n$  that depends only on  $A$ ,  $n$ , and  $\mathbf{x}_0$ .

3. Let  $A = \begin{bmatrix} 0.60 & 0.29 & 0.16 \\ 0.26 & 0.37 & 0.27 \\ 0.14 & 0.34 & 0.57 \end{bmatrix}$ . This matrix is called a stochastic matrix, or probability matrix, since each column sums to one.

- (a) Consider the sequence  $\mathbf{x}_{n+1} = A\mathbf{x}_n$ ,  $\mathbf{x}_0 = \begin{bmatrix} 0.12 \\ 0.32 \\ 0.56 \end{bmatrix}$ . Compute  $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ .

- (b) Use MATLAB to estimate  $\lim_{n \rightarrow \infty} \mathbf{x}_n$ , with each entry accurate to four decimal places.

4. Let  $A = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 4 & -3 & 8 \end{bmatrix}$ ,  $\mathbf{b} = \begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix}$ .

- (a) Compute  $A^{-1}$  by using the MATLAB command `inv(A)`.  
 (b) Use part (a) to solve  $A\mathbf{x} = \mathbf{b}$ .  
 (c) Check your answer in part (b) by computing  $A\mathbf{x}$  and verifying that it is approximately equal to  $\mathbf{b}$ .

5. Consider the equation  $A\mathbf{x} = \mathbf{b}$ , where

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}.$$

- (a) Show that the equation  $A\mathbf{x} = \mathbf{b}$  has infinitely many solutions. Describe the set of possible solutions.
- (b) Try to compute  $A^{-1}$  by row reducing the  $3 \times 6$  matrix  $\left[ \begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 4 & 5 & 6 & 0 & 1 & 0 \\ 7 & 8 & 9 & 0 & 0 & 1 \end{array} \right]$ . Why does Theorem 4.1.4 not seem to apply?
- (c) Use MATLAB's command `inv(A)`. What does it tell you?
- (d) Use MATLAB's commands `inv(A)*b` and `A\b` to attempt to solve  $A\mathbf{x} = \mathbf{b}$ . In each case, use your answer in (a) to determine if these are indeed solutions.

6. By hand, find the characteristic polynomial, eigenvalues, and corresponding eigenvectors for the matrix  $A = \begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix}$ .

7. Use MATLAB to find eigenvalues and the corresponding eigenvectors for the matrix

$$A = \begin{bmatrix} 3 & 0 & 0 \\ 4 & 2 & 1.5 \\ -5 & 0 & .5 \end{bmatrix}. \text{ Give integer value entries for the eigenvectors, if possible.}$$

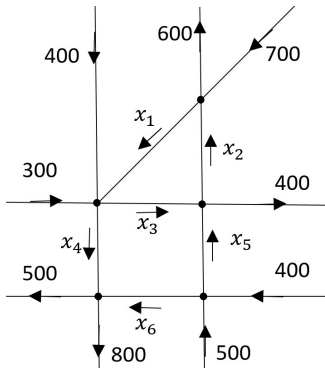
8. Prove Proposition 4.2.3.

9. (Leontief's "Input-Output" Model) Suppose an economy has three sectors, Coal (C), Steel (S), and Electric (E). Each sector requires the resources of the others to produce a given amount. Suppose that:
- To make \$1 of coal, it takes no coal, but \$.03 of steel and \$.01 of electricity.
  - To make \$1 of steel, it takes \$.15 of coal, \$.04 of steel, and \$.08 of electricity.
  - To make \$1 of electricity, it takes \$.41 of coal, \$.18 of steel, and \$.05 of electricity.

In addition to the coal, steel, and electricity it takes each sector to operate, suppose we want to produce an excess of \$1 billion coal, \$2 billion steel, and \$3 billion electricity. Determine how much each industry should produce to accomplish this. Let  $c$ ,  $s$ , and  $e$  be the output of coal, steel, and electricity in billions of dollars.

- (a) With the assumptions above, show that making an excess of \$1 billion in coal requires that  $1 + 0.15s + 0.41e = c$
- (b) Find two similar equations by using the given requirements to make an excess of \$2 billion steel and \$3 billion electricity.
- (c) Form a matrix equation and solve it using MATLAB.

10. The figure shows the traffic flow (cars per hour) on several one-way streets in a city center during the most busy times on a typical weekday. For example, the diagonal road in the upper-right corner has an average of 400 cars per hour traveling southwest.



Each street can handle a maximum of 1000 cars per hour without causing congestion.

- We assume that the amount of traffic entering an intersection is equal to the traffic exiting the intersection - for example from the top-most intersection we have  $700 + x_2 = x_1 + 600$ . Use this logic to write a system of equations involving  $x_1, x_2, \dots, x_6$ .
  - Determine two combinations of values of  $x_1, x_2, \dots, x_6$  that ensure that traffic will flow smoothly.
  - The city is considering doing some maintenance on  $x_4$  which will slow (but not stop) the rate of traffic on that street. What is the minimum flow rate of  $x_4$  required in order to avoid congestion?
11. (Adapted from [4]). Consider working through this problem as an in-class activity or in pairs for homework. Suppose that in a biological system there are  $n$  species of animals and  $m$  sources of food. Let  $x_j$  represent the population of species  $j$ ,  $b_i$  be daily quantity of food  $i$ , and  $a_{ij}$  be the amount of food  $i$  consumed per day on average by species  $j$ . Let  $\mathbf{x} = (x_1, \dots, x_n)^T$ ,  $\mathbf{b} = (b_1, \dots, b_m)^T$  and define the matrix  $n \times m$  matrix  $A$  having entries  $a_{ij}$ .
- Interpret the vector  $A\mathbf{x}$  in the context of this biological system.
  - Consider the matrix equation  $A\mathbf{x} = \mathbf{b}$ . Discuss what this represents in the context of this biological system.
  - Let  $A = \begin{bmatrix} 1 & 2 & 0 & 3 \\ 1 & 0 & 2 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix}$ ,  $\mathbf{x} = (1000, 500, 350, 400)^T$ ,  $\mathbf{b} = (3500, 2700, 900)^T$ . Is there sufficient food of each type to satisfy the average daily consumption? [Note: since entries in  $\mathbf{b}$  are on the order of thousands, MATLAB will likely use scientific notation when giving results of matrix calculations. Use `format short g` to display results without scientific notation.]
  - Assuming that the number of species 2, 3, and 4 remains constant as given in (c), what is the maximum population of species 1 that could be added to the system so that the food supply still meets the needs of all of the species?
  - Suppose species 1 becomes extinct. If species 3 and 4 remain constant as given in (c), what is the maximum population of species 2 that could be added to the system so that the food supply still meets the needs of all of the species?
  - From a modeling process point of view, what assumptions are implied with the given matrix model?
  - What potential errors and/or ethical considerations might you report, based on this model and your results? For example, for what reasons might reality differ from the results of your analysis? In what situations might these results be reliable for management decisions? When might they be unreliable?

## Chapter 5

# Discrete Models

### 5.1 Introduction

In Calculus, you studied the relationship between continuous variables and how one changes with respect to another. For example, derivatives and differential equations may be used to describe the continuous dynamical systems of disease or predator-prey models with respect to time. An example of a continuous model that we have already studied is the growth formula  $y = a \cdot e^{kt}$  where a population  $y$  varies with time  $t$ . In this model, the variable  $y$  changes as time increases, even if  $t$  increases by a tiny amount. In section 1.3 you saw a couple examples of discrete models, where the system is updated only at **discrete** points in time. In practice, when you use a numerical method such as Euler's method to solve a continuous model you are essentially approximating it with a discrete model - since you are only concerned with what happens at each time step.

In discrete models, we keep track of what happens at regular intervals of time, such as every month or every year. Often this makes a lot of sense for modeling population dynamics - for example, the crocus flower blooms in early spring, moose drop antlers in the fall, and bears have cubs in the spring. If we are to keep track of the total numbers of individuals with this type of seasonal behavior, there may be no added benefit to considering time as a continuous variable.

A basic component of discrete models is a recursive relation. For example, let  $x_n$  be the quantity of some population at time step  $n$ . A **recursive relation**, or **difference equation**, is an equation of the form

$$x_n = f(x_{n-1}, x_{n-2}, \dots, x_1, x_0).$$

That is, the population at time  $n$ ,  $x_n$  is a function of the population at all previous time steps. A simple equation of the form  $x_n = f(x_{n-1})$  is considered to be a **first order** recurrence relation, as it only depends on the value on the previous time step. A sequence of the form  $x_n = f(x_{n-1}, x_{n-2})$  is considered a **second order** recurrence relation.

If a model uses one or more recurrence relations to describe some phenomena, it is said to be a **discrete dynamical system**.

To begin, we examine some basic population growth models with discrete time. First suppose that the rate of growth of a population for the next time interval is proportional to the current population, a first order relationship. Letting  $P_t$  be the population at the time step  $t$ , we should see that  $P_{t+1} - P_t = rP_t$  where  $r$  is some fixed growth constant. The left-hand side  $P_{t+1} - P_t$  represents the increase in growth over the time step  $t$  and we see that it is proportional to the current population  $P_t$ , with  $r$  being the constant of proportionality. Solving for  $P_{t+1}$  gives the model

$$P_{t+1} = P_t + rP_t = (1 + r)P_t. \quad (5.1)$$

This is sometimes called an exponential growth model, or a **Malthusian model**, named after Thomas Malthus (1766 - 1834). See [Activity A.0.36](#) for an example of using this to model the U.S. population.

Notice the conditions on  $r$  which result in exponential decay of the population ( $r < 0$  or  $1 + r < 1$ ) and the conditions on  $r$  which give exponential growth ( $r > 0$  or  $1 + r > 1$ ).

Often it is the case that if the population increases beyond a certain point then the experiment is over and the model falls apart - for example the amount of algae in a lake eventually reaches some maximum capacity. In this case, the rate of growth slows as the population approaches this limiting capacity. The **logistic growth model** describes this process. Let  $K$  be the maximum capacity of the population that the environment will support (this is sometimes called the **carrying capacity** of the population). We assume that the population grows exponentially when the population  $P$  is relatively small compared to  $K$ , and growth slows as the population increases towards the carrying capacity. The discrete logistic model is

$$P_{t+1} = P_t + rP_t \left(1 - \frac{P_t}{K}\right), \quad (5.2)$$

where  $K$  is the carrying capacity and  $P_t$  is the population at time step  $t$ . Notice that when  $P_t$  is small compared to  $K$ ,  $P_t/K$  will be close to zero, which means that  $P_{t+1} \approx P_t + rP_t$ . This means that when the population is small the logistic model is close to the exponential growth model, (5.1). On the other hand, when  $P_t$  gets close to  $K$ , we have  $P_t/K \approx 1$  and so  $P_{t+1} \approx P_t + rP_t(1 - 1) = P_t$ . In other words, growth is relatively constant as the population approaches the carrying capacity.

Writing the logistic growth equation (5.2) another way gives

$$P_{t+1} = (1 + r)P_t - \frac{r}{K}P_t^2.$$

While still first order, in this form it is more obvious logistic growth is a nonlinear recurrence relation because of the quadratic term  $-\frac{r}{K}P_t^2$ .

## 5.2 Linear Dynamical Systems

Next we will look at a system of more than one linear recurrence relations representing inter-related quantities or populations. Suppose we have two populations  $x_t$  and  $y_t$  at time step  $t$  modeled with

$$\begin{aligned} x_{t+1} &= a_{11}x_t + a_{12}y_t \\ y_{t+1} &= a_{21}x_t + a_{22}y_t, \end{aligned} \quad (5.3)$$

where  $a_{11}, a_{12}, a_{21}, a_{22}$  are all constants. Notice that the right-hand sides of (5.3) are linear with respect to  $x_t$  and  $y_t$ . Since this is a linear model, we may re-write the system as an equivalent matrix equation,

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix},$$

or simply

$$\mathbf{x}_{t+1} = A\mathbf{x}_t,$$

where  $\mathbf{x}_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix}$  and  $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ .

Given a matrix model such as this, it is worth exploring the eigenvalues and eigenvectors of matrix  $A$ . What are the meanings of these mathematical constructs in the context of our populations? It turns out that the eigenvalues and eigenvectors of  $A$  determine the long term behavior of the system and each population.

To begin, we need the following result, which is typically proven in a course in linear algebra (see [14] for example).

**Proposition 5.2.1.** *If  $A$  is an  $n \times n$  matrix with distinct eigenvalues  $\lambda_1, \lambda_2, \dots, \lambda_n$ , and associated eigenvectors  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ , then for any  $\mathbf{x} \in \mathbb{R}^n$ , we may write  $\mathbf{x} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \dots + c_n\mathbf{v}_n$  for some constants  $c_1, c_2, \dots, c_n \in \mathbb{R}$ .*

With this result, we may also prove the following proposition (see Exercise 15), used to determine the long term behavior of the sequence  $\mathbf{x}_{k+1} = A\mathbf{x}_k$ .

**Proposition 5.2.2.** *Let  $A$  be a  $n \times n$  matrix with distinct eigenvalues  $\lambda_1, \lambda_2, \dots, \lambda_n$ , and associated eigenvectors  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ . Then the sequence defined by  $\mathbf{x}_{k+1} = A\mathbf{x}_k, k \geq 0$  satisfies*

$$\mathbf{x}_k = c_1 \lambda_1^k \mathbf{v}_1 + c_2 \lambda_2^k \mathbf{v}_2 + \dots + c_n \lambda_n^k \mathbf{v}_n,$$

where  $c_1, c_2, \dots, c_n$  are constants that may be determined from  $\mathbf{x}_0$ .

See **Activity A.0.37** for an example of using Proposition 5.2.2 to analyze a predator prey model.

### Example 5.2.3

In Activity A.0.37 we modeled the owl and wood rat populations using

$$\begin{aligned} L_{k+1} &= 0.5L_k + 0.4R_k \\ R_{k+1} &= -0.104L_k + 1.1R_k \end{aligned} \quad ,$$

where  $L_k$  is the number of owls in the region studied, and  $R_k$  is the number of thousands of rats in the area at week  $k$ .

Before we proceed with graphical representations of these two populations, it's helpful to take a minute and reality check our model. For inter-dependent populations, first consider the case where each population is extinct - how does this impact the other population?

If  $R_k = 0$  and there are no rats, does the owl population persist? In this case, our model reduces to  $L_{k+1} = 0.5L_k$  so the owl population will decline eventually to zero. This is reasonable if we have assumed rats are the main food source for these owls.

Alternatively, if  $L_k = 0$  and there are no owls, the recurrence relation for rats reduces to  $R_{k+1} = 1.1R_k$ . With a positive growth rate, this gives an exponentially growing rat population. If we have assumed an abundant food supply for the rats with owls being their main predator, this type of growth makes sense if the owls are extinct. Notice that a more complicated model of the food web the rats and owls are a part of might include additional predators and additional prey or food sources. However, this simplified model is a great place to start given the significant relationship between these two species.

Now suppose we want to plot the two populations by week as well as the trajectories of both species over a 50 week period of time. The following code generates the iterates  $L_k$  and  $R_k$  for  $k = 0, 1, \dots, 50$ .

```

%DDS_OwlRat.m
1  n = 50;                                %number of time steps
2  L = zeros(n+1,1);                       %number of owls
3  R = zeros(n+1,1);                       %number of rats (thousands)
4  L(1) = 30;                               %set initial number of owls
5  R(1) = 25                                %set initial number of rats
6  for i=1:n
7      L(i+1) = 0.5*L(i) + 0.4*R(i);
8      R(i+1) = -0.104*L(i) + 1.1*R(i);
9  end
10 week = 1:n+1;
11 plot(week, L, week, R)
12 legend('Owls','Rats (thousands)')
13 xlabel('Week')
14 ylabel('Population')
15
16 figure                                  %new figure for trajectory plot
17 scatter(L, R)                            %Make trajectory plot
18 xlabel('Owls')
19 ylabel('Rats (thousands)')

```



In lines 2 and 3 we preallocate vectors to keep track of all values of  $L_k$  and  $R_k$  (see Section 2.6 for a refresher). Keep in mind that  $k$  runs 0 to  $n$ , but in MATLAB the index must start at 1 - so our indices will run 1 to  $n + 1$ . The initial populations  $L_0 = 30, R_0 = 25$  are set on lines 4 and 5. In line 10, we create a vector of weeks, containing the integers 1 to  $n + 1$  so that we may plot populations against the week. On line 11 we plot both populations using a single command, as we did in [Activity A.0.8](#). In lines 16 - 19 we open a new figure window and make a **trajectory plot**, showing how the populations depend on each other.

---

We can tie these graphical results into the long-term behavior we saw in [Activity A.0.37](#) with the dominant eigenvalue and eigenvector. In particular, for a matrix model of the form  $\mathbf{x}_{k+1} = A\mathbf{x}_k$ , if the matrix  $A$  has an eigenvalue  $\lambda_d > 1$  such that  $|\lambda_d| \geq |\lambda_i|$  for all  $i = 1, 2, \dots, n$ , then  $\lim_{k \rightarrow \infty} \mathbf{x}_k = \lambda_d^k c_d \mathbf{v}_d$ . In this case, we call  $\lambda_d$  the **dominant eigenvalue** and  $\mathbf{v}_d$  the **dominant eigenvector**. The value  $|\lambda_d|$  is called the **spectral radius** of the matrix  $A$ . The dominant eigenvalue determines the overall growth of the system (2% in the owl and rat example), and the dominant eigenvector determines the eventual ratio of the two species.

For  $|\lambda_d| < 1$  and  $|\lambda_d| \geq |\lambda_i|$  for all  $i = 1, 2, \dots, n$ , the same limit applies but  $\lambda_d^k$  then decays to zero. If  $-1 < \lambda_d < 0$ ,  $\lambda_d^k$  oscillates between positive and negative values as it decays to zero in **damped oscillations**. For  $\lambda_d < -1$ ,  $\lambda_d^k$  oscillates but grows in **undamped oscillations**.

For complex eigenvalues,  $\lambda_d^k$  will oscillate and grow if the modulus  $|\lambda_d| > 1$  or decay if  $|\lambda_d| < 1$ . This gives a cyclical relationship between quantities or species in the long term.

---

**Remark:** Remember that given any eigenvector  $\mathbf{v}_i$ , any scalar multiple is also an eigenvector for the corresponding eigenvalue  $\lambda_i$ . To ensure the dominant eigenvector represents a ratio of these populations, we need its entries to be positive and sum to 1. If MATLAB gives a different result, dividing the dominant eigenvector by the sum of its entries should do the trick.

---

There are three possibilities for the nature of the solutions to  $\mathbf{x}_{k+1} = A\mathbf{x}_k, k \geq 0$  as they relate to the fixed point at the origin. (Note that  $\mathbf{x}^* = \mathbf{0}$  is always a fixed point or equilibrium of the system  $\mathbf{x}_{k+1} = A\mathbf{x}_k$ .) The nature of the solution and this equilibrium depends on the eigenvalues. If all eigenvalues have absolute value larger than 1, then the trajectories tend away from the origin, and the origin is said to be a **repeller** of the dynamical system. We also say the origin is an **unstable equilibrium**. If all eigenvalues have absolute value smaller than 1, the trajectories tend towards  $\mathbf{0}$  and the origin is said to be a **stable equilibrium**. We also say the origin is an **attractor** of the dynamical system. If some of the eigenvalues have absolute value more than 1, while other have absolute value less than one, then we say the origin is a **saddle point**. In this case, the origin attracts solutions from some directions and repels solutions from other directions. In 2- and 3-dimensions, a **trajectory** refers to a parametric plot of points from the sequence  $\{\mathbf{x}_k\}$ . In [Activity A.0.37](#), the number of owls are plotted against the number of rats for each time step  $0, 1, 2, \dots, k$ .

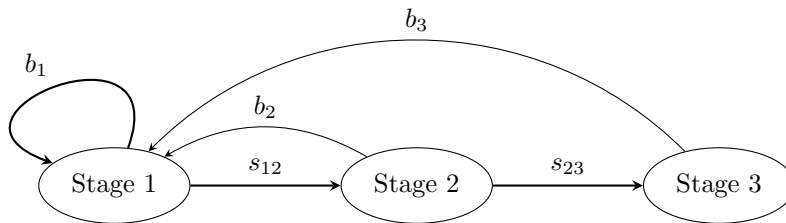
See [Exercise A.0.38](#) for an example of classifying the origin as an attractor, repeller, or saddle point.

## 5.3 State, Age, and Stage Matrix Models

In addition to modeling interdependent populations as a matrix model, we can use this same structure to describe the dynamics of a single population or group of objects that can be separated into distinct states, ages or stages. Transition rates between individuals or quantities from one state to another (or back to the same state) are given according to the discrete time length; e.g. year, month, day, etc. This type of model is useful when the average behavior of all individuals is not descriptive enough. We may wish to divide a population into ages by each year if the data is available or into groups using a collection of years for each class if we wish to reduce the number of classes to a more manageable amount.

For example, consider a population of humans divided into three groups of 15 years each: 0 – 14 years (Stage 1), 15 – 29 years (Stage 2), and 30 – 44 years (Stage 3). In this model, a time step of 15 years is logical. As with many population models, we will only keep track of the females in this population. If we assume the population is 50% female and 50% male, we need only double the female population in each class to determine the total. Also note that no females older than 44 are captured in any stage. The assumption here is that females of this age do not contribute significantly to the population. Consider the validity of this assumption. Is this valid in modern first-world countries? Perhaps the reproductive age is higher. What about remote populations or those in third-world countries? Perhaps it is lower. To include a more accurate description for the given population, age classes could be designed to accommodate 5-year increments, perhaps to age 54 or age 39.

The **state diagram** below shows each state in a circle along with arrows for transitions between states and the values (or symbols) for each of these rates. Birth rates,  $b_i$ , are given to show the contribution of new individuals to the first age class per individual in state  $i$ . These values can be less than or greater than one. Survival rates  $s_{ij}$  give the proportion of individuals from state  $i$  that survive to state  $j$ , so they must be less than or equal to one. In age models, survival often progresses from state  $i$  to state  $i + 1$  at each time step. In stage models such as plant growth, a first-year seed may stay dormant and require another year to germinate or the seed may develop into a flowering plant. Here the stages are numbered to keep track of them, but transition rates may not represent a sequential progression along a linear path of development. In this case, weights  $w_{ij}$  or transition rates are more appropriate descriptors than survival rates. Note that no individuals are assumed to stay within a particular class and death is implied by transition rates less than one, it is not explicitly modeled.



We can now describe the total number of individuals at time  $t$  in each of the three classes as a **state vector**,

$$\mathbf{x}_t = \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix},$$

where

$$\begin{aligned} x_{t+1} &= b_1 x_t + b_2 y_t + b_3 z_t \\ y_{t+1} &= s_{12} x_t \\ z_{t+1} &= s_{23} y_t \end{aligned}$$

This gives rise to the matrix equation  $\mathbf{x}_{t+1} = A\mathbf{x}_t$ , where  $A$  is the **projection matrix**

$$A = \begin{bmatrix} b_1 & b_2 & b_3 \\ s_{12} & 0 & 0 \\ 0 & s_{23} & 0 \end{bmatrix}.$$

Notice the structure of this projection matrix, including birth rates in the first row - all contributions to the first age class, and the survival rates along the sub-diagonal - individuals in each class must progress to the next age class or die. This feature of the model makes sense because each of the age classes encompass the same number of years as the length of the time step,  $t$ . A matrix with this structure is called a **Leslie matrix**. These models are used in population ecology to project species extinction and survival. Note that if the time step and size of age classes differ or the age classes themselves are of differing lengths, a variation

on the Leslie matrix may include values on the main diagonal to account for the rates at which individuals remain in a given age class.

As with the owl and rat example, we can determine the long term average growth rate of humans and the distribution of age classes using the dominant eigenvalue and eigenvector, respectively, of the projection matrix  $A$ .

Suppose the following rates are provided:

$$b_1 = 0.4271 \quad b_2 = 0.8498 \quad b_3 = 0.1273 \quad s_{12} = 0.9924 \quad s_{23} = 0.9826.$$

Using MATLAB, we determine the dominant eigenvalue and corresponding eigenvector for  $A$  to be

$$\lambda_d = 1.209 \quad \mathbf{v}_d = \begin{bmatrix} 0.402 \\ 0.330 \\ 0.268 \end{bmatrix}.$$

From  $\lambda_d$  we see an overall growth rate of 20.9% every 15 years. The population distribution for each of the three age classes is given by  $\mathbf{v}_d$ . At first glance, the percentage of individuals under age 15 seems quite large at 40.2%. Before we get too far, remember this number is a percentage of *the population being modeled* - in this case, females age 0–44. Women over age 45 have not been included. Knowing this, what assumptions in your model gave rise to this large percentage of young people? Are those assumptions valid? In the United States? Elsewhere?

See [Activity A.0.41](#) and chapter exercise for more examples.

**Remark:** Notation for matrix models can become a bit cumbersome and run into multiple meanings. If we are careful and use the context of the problem, we can usually discern the appropriate meaning. For

example, the state vector  $\mathbf{x}_t = \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix}$  includes values for three quantities with the subscript  $t$  used to

indicate the time. However, several matrix models may include more than three quantities. Rather than use every letter of the alphabet, we may wish to use subscripts of  $\mathbf{x}$  for the  $i^{\text{th}}$  quantity and parentheses around  $t$  to indicate the time. For example, we may describe a state vector with  $n$  quantities at time  $t$  as

$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}$ . In this case, we must use the context to recognize  $x_1$  as the first quantity of the state

vector, rather than the state vector at time step  $t = 1$ .

## 5.4 Markov Chains

As a special case, we next consider a situation in which something may be in one of several states, and we know the probability that the system moves from each state to the next. We assume the probability of moving from state to state never changes, regardless of the time step or size of a given state. This assumption is called the Markov property. A **Markov chain** gives a simple way to describe this situation. For example, people often move between the city and its suburbs in a metropolitan area. We might describe people being in one of two states (city and suburbs) and if we know the probabilities of people moving between the city and its suburbs each year then we can model this situation with a Markov chain.

With the Markov property, these probabilities are constant - even if the population in the suburbs are high or the population in the cities are low, etc. In the year 2020, this assumption was not valid as a high proportion

of people moved out of cities and into the suburbs with forced social distancing and lifestyle changes due to the novel coronavirus.

Instead of a state vector with the total number of individuals in a given state, a Markov chain uses a probability vector. A **probability vector** is a vector with non-negative entries adding to 1. Each entry represents the probability of the system being in that particular state. An example of a probability vector is  $x = (0.6, 0.4)^T$ . This means that there are two states, with probabilities 0.6 and 0.4. Notice that  $0.6 + 0.4 = 1$ .

**Definition 5.4.1.** A *left stochastic matrix*, or *transition matrix*,  $M$  is a square matrix whose columns are probability vectors (i.e. the columns add to one).

If  $\mathbf{x}_0$  is a probability vector, then the sequence  $\mathbf{x}_{n+1} = M\mathbf{x}_n, n \geq 1$ , creates a **Markov chain**  $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots$ . Sometimes  $\mathbf{x}_n$  is called a **state vector** where context is used to distinguish its entries as probabilities rather than total numbers of individuals. The following proposition implies that each iterate  $\mathbf{x}_n$  in a Markov chain is a probability vector.

**Proposition 5.4.2.** If  $\mathbf{x}$  is a probability vector, and  $M$  is a transition matrix, then  $M\mathbf{x}$  is a probability vector.

In simple terms, we may use a Markov chain model any time an object may take on a number of states, and we know the probability that the system moves from each state to the next. We also assume that the Markov property holds, which says that the probability of moving from state  $j$  to state  $i$  does not change throughout time or depend on how state  $j$  was arrived at.

Notice that a Markov chain is a specific case of a linear dynamical system of the form  $\mathbf{x}_{n+1} = M\mathbf{x}_n$ , where the columns of  $M$  sum to 1 and the  $a_{ij}$  entry gives the probability an individual in state  $j$  moves to state  $i$ .

**Remark:** A right stochastic matrix  $M$  is a square matrix in which the rows add to one (instead of the columns). In this case, a probability vector  $\mathbf{x}$  may be defined to be a row vector adding to one. Instead of writing  $M\mathbf{x}$  we would write  $\mathbf{x}M$ . If you use external sources to study Markov Chains, you may find that other authors use right stochastic matrices instead of left stochastic matrices. It turns out that  $(M\mathbf{x})^T = \mathbf{x}^T M^T$ , so it makes no difference which convention is used.

**Example 5.4.3**

Suppose that 5% of people living in the suburbs move to the city each year, while 3% of city dwellers move to the suburbs each year. What will happen in the long term? If we assume this percentage remains constant over time, we may model this situation using a Markov chain. At year  $n + 1$  the proportion of people living in the suburbs  $s_n$  and the city  $c_n$  is

$$s_{n+1} = 0.95s_n + 0.03c_n, \quad c_{n+1} = .05s_n + 0.97c_n.$$

The transition matrix is then

		From:
		Suburbs   City
To:	Suburbs	$\begin{bmatrix} 0.95 & 0.03 \\ 0.05 & 0.97 \end{bmatrix}$
	City	

Suppose in 2019, the population of the city is 600,000, and the population of the suburbs is 400,000.

- (a) Find the distribution of the population in 2020 and 2021.
- (b) Use MATLAB to find the distribution of the population in 2030.
- (c) Do the two populations approach some equilibrium? If so, what is it?

.....

Set  $\mathbf{x}_0 = \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix}$ . Notice that  $\mathbf{x}_0$  is a probability vector since it sums to 1. To answer (a),  $\mathbf{x}_1 = \begin{bmatrix} 0.95 & 0.03 \\ 0.05 & 0.97 \end{bmatrix} \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0.398 \\ 0.602 \end{bmatrix}$ , and  $\mathbf{x}_1 = \begin{bmatrix} 0.95 & 0.03 \\ 0.05 & 0.97 \end{bmatrix} \begin{bmatrix} 0.398 \\ 0.602 \end{bmatrix} = \begin{bmatrix} 0.396 \\ 0.604 \end{bmatrix}$

To answer (b), we need to find  $\mathbf{x}_8$ . We could use the Command Window:

```
A=[.95 .03; .05 .97]
A* [.4;.6]
A*ans
A*ans
⋮
```

We would repeat `A*ans` until we reach the 8th iterate, or we could use a for loop. The model predicts that 38.8% of the population will be in the city and 61.2% will be in the suburbs by 2021.

To answer (c), we want to determine  $\lim_{n \rightarrow \infty} \mathbf{x}_n$ , and estimate this by taking  $n$  large. You should see that the population is settling to 37.5% in the city and 62.5% in the suburbs.

It turns out that we could have actually defined  $\mathbf{x}_n$  as state vectors with the populations at year  $n$  instead of the proportions of the population. We could have set  $\mathbf{x}_0 = \begin{bmatrix} 600000 \\ 400000 \end{bmatrix}$ . While the model would not give probability vectors, it would give the population distributions assuming that the total population did not change. This is equivalent to making the assumption that no immigration or emigration occurred and the number of births and deaths were equal during the given time frame.

In a Markov Chain model  $\mathbf{x}_{n+1} = M\mathbf{x}_n$ , we often want to determine a **limiting distribution**  $\mathbf{x}^*$  such that  $\lim_{n \rightarrow \infty} \mathbf{x}_n = \mathbf{x}^*$ . It turns out that a stochastic matrix always has a dominant eigenvalue equal to 1. Since 1 is an eigenvalue, we have  $M\mathbf{v}_d = \mathbf{v}_d$ , where  $\mathbf{v}_d$  is the dominant eigenvector. If we think of  $M$  as a function defined by  $f(\mathbf{v}) = M\mathbf{v}$ , we see that  $\mathbf{v}_d$  is a fixed point of this function - that is, the Markov chain never changes if we start from  $\mathbf{v}_d$ . Consequently, such a vector  $\mathbf{v}_d$  is sometimes called a steady state, or **stationary distribution**.

Let  $\mathbf{v}_i$  denote the eigenvector corresponding to the eigenvalue  $\lambda_i$  and let  $\mathbf{v}_1$  be the dominant eigenvector (so that  $\lambda_1 = 1$ ). By Proposition 5.2.2, for any probability vector  $\mathbf{x}_k$  we have

$$\mathbf{x}_k = c_1 \lambda_1^k \mathbf{v}_1 + c_2 \lambda_2^k \mathbf{v}_2 + \dots + c_n \lambda_n^k \mathbf{v}_n,$$

where  $c_1, c_2, \dots, c_n$  are constants. Since 1 is the dominant eigenvalue, the other eigenvalues satisfy  $\lambda_i < 1$  by definition. Therefore,  $\lim_{k \rightarrow \infty} \lambda_i^k = 0$  for all  $i > 1$ . Hence,

$$\lim_{k \rightarrow \infty} \mathbf{x}_k = c_1 \mathbf{v}_1.$$

Since  $\mathbf{x}_k$  is always a probability vector, it must be the case that  $c_1 = 1$ . Hence,  $\mathbf{x}_k \rightarrow \mathbf{v}_1$  as  $k \rightarrow \infty$  for any starting value  $\mathbf{x}_0$ . This means that the limiting distribution  $\mathbf{x}^*$  is simply the dominant eigenvector of  $M$  and satisfies  $M\mathbf{x}^* = \mathbf{x}^*$ .

Alternatively as in [20], we can define the **limiting matrix**  $L$  as

$$L = \lim_{k \rightarrow \infty} M^k, \tag{5.4}$$

where  $M$  and  $L$  are left stochastic matrices. Notice

$$\mathbf{x}^* = \lim_{k \rightarrow \infty} \mathbf{x}_k = \lim_{k \rightarrow \infty} M^k \mathbf{x}_0 = L\mathbf{x}_0.$$

For  $\mathbf{x}_0 = \mathbf{x}^*$ , we have  $\mathbf{x}^* = L\mathbf{x}^*$ , verifying  $L$  also has a dominant eigenvalue equal to 1 since  $\mathbf{x}^* = \mathbf{v}_1$ . Using the properties of stochastic matrices, we can show that the columns of  $L$  are in fact repeated instances of

the dominant eigenvector  $\mathbf{v}_1$ . Approximating  $L$  with  $M^n$  for large  $n$  gives another method of determining the dominant eigenvector for a Markov chain model.

**Example 5.4.4**

For the Markov chain

$$\mathbf{x}_{n+1} = \begin{bmatrix} 0.95 & 0.03 \\ 0.05 & 0.97 \end{bmatrix} \mathbf{x}_n,$$

find  $\mathbf{x}^*$  by finding the eigenvalues and eigenvectors of  $M$ .

.....  
Using MATLAB to find the eigenvalues and eigenvectors,

```
A=[.95 .03; .05 .97]
[x y] = eig(A)
```

gives us eigenvalues of 0.92 and 1, and the eigenvector corresponding to the eigenvalue of 1 is  $(-0.5145, -0.8575)^T$ . Recall that MATLAB outputs eigenvectors with norm equal to 1, so we would need to convert it to a stochastic vector:

```
evector=x(:, 2)
evector = evector/sum(evector)
```

Approximating the limiting matrix  $L$  with  $M^{100}$  in MATLAB gives

$$L = \begin{bmatrix} 0.375 & 0.375 \\ 0.625 & 0.625 \end{bmatrix}$$

and the dominant eigenvector  $\begin{bmatrix} 0.375 \\ 0.625 \end{bmatrix}$  as we saw above.

See [Exercise A.0.42](#) for another example of a Markov chain model.

## Absorbing Markov Chains

An absorbing Markov chain has one or more states in which the probability of remaining in the state is 1. An absorbing state  $i$  has the probability vector of the form  $e_i$  (i.e. all zeros except for a 1 in location  $i$ ). It is often convenient to arrange the matrix so that the absorbing states correspond with the lowermost rows. For example, the transition matrix

$$M = \begin{bmatrix} 0.1 & 0.3 & 0 & 0 & 0 \\ 0.1 & 0.1 & 0.4 & 0 & 0 \\ 0.2 & 0.4 & 0.3 & 0 & 0 \\ 0.3 & 0.2 & 0.2 & 1 & 0 \\ 0.3 & 0 & 0.1 & 0 & 1 \end{bmatrix}$$

has two absorbing states, four and five. In general, if a Markov chain has  $a$  absorbing states and  $b$  non-absorbing states, the transition matrix takes on the block form

$$T = \begin{bmatrix} A & 0 \\ B & I \end{bmatrix}, \quad (5.5)$$

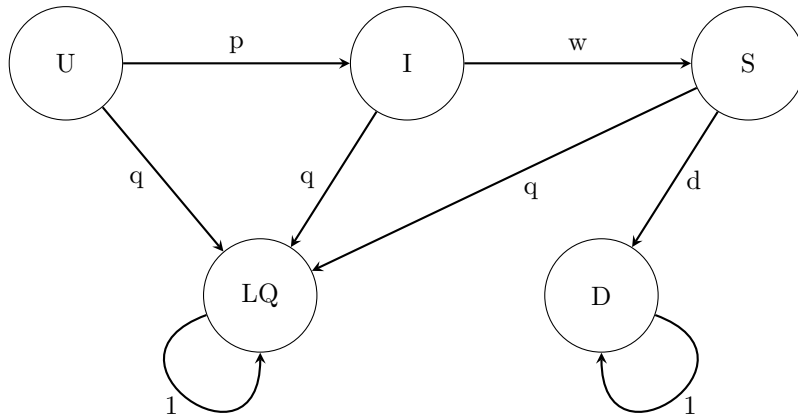
where  $A$  is a  $b \times b$  matrix,  $0$  is the  $b \times a$  matrix of zeros,  $I$  is the  $a \times a$  identity matrix, and  $B$  is an  $a \times b$  matrix. We may need to re-order the states so that the absorbing states are on the bottom row.

**Example 5.4.5**

The public U.S. Centers for Disease Control and Prevention (CDC) model for HIV and for hepatitis B describes the process of how a disease is detected. The Markov chain has three transient states: uninfected (U), infected without symptoms (I), and infected with symptoms (S). There are two absorbing states: the person is lost or quits treatment (LQ), and detection of the disease (D). For each week, assume the following transition probabilities.

- $p$  is the probability of an uninfected person becoming infected.
- $w$  is the probability that an infected person begins having symptoms.
- $d$  is the probability that a person with symptoms is correctly diagnosed.
- $q$  is the probability that a person is lost or quits treatment.

Note the state diagram below and the probabilities for each absorbing state. These must be equal to 1 by definition.



Forming the transition matrix with transient states followed by absorbing states gives

$$\begin{matrix} & & \begin{matrix} \text{U} & \text{I} & \text{S} & \text{LQ} & \text{D} \end{matrix} \\ \begin{matrix} \text{U} \\ \text{I} \\ \text{S} \\ \text{LQ} \\ \text{D} \end{matrix} & \left[ \begin{array}{ccccc} 1-p-q & 0 & 0 & 0 & 0 \\ p & 1-w-q & 0 & 0 & 0 \\ 0 & w & 1-q-d & 0 & 0 \\ q & q & q & 1 & 0 \\ 0 & 0 & d & 0 & 1 \end{array} \right] & .
 \end{matrix}$$

Notice that the columns sum to one, so this is indeed a left-transition matrix. There are a few interesting questions we can ask:

- What is the probability of detecting the disease versus the person being lost or quitting treatment?
- What is the average time to diagnosis?

.....

We could answer the first question by running the system for a long time and examining the probabilities.

DDS\_Disease.m

```

p=.2; %probability uninfected person becomes infected
w=.7; %probability infected person begins showing symptoms
d=.5; %probability person with symptoms is correctly diagnosed
q=.1; %probability a person is lost or quits treatment

T=[1-p-q, 0, 0, 0, 0; %probability transition matrix
   p, 1-w-q, 0, 0, 0;
   0, w, 1-q-d, 0, 0;
   q, q, q, 1, 0;
   0, 0, d, 0, 1];
x=[1;0;0;0;0] %initial conditions with all uninfected
n=100 %number of weeks
for i=1:n
    xold=x; %allocate previous week's distribution
    x=T*x; %update current week's distribution
end
x %final distribution of individuals

```

After running the for loop 100 times, `x` should tell us the probability of a person being lost or quitting is 0.5139 and the probability of being diagnosed is 0.4861. This code segment does not however tell us what the average time to diagnosis is. It is possible to modify the code to approximate the average time but it might be more useful to learn a little bit of theory about absorbing Markov chains.

First we get the transition matrix into block form as in (5.5). For an absorbing state Markov model with transition matrix  $T$ ,  $t$  transient states and  $r$  absorbing states, the sub-matrix  $A$  is a  $t \times t$  matrix formed from the transient states,  $B$  is a  $r \times t$  matrix,  $0$  is a  $t \times r$  matrix of all zeros, and  $I$  is the  $r \times r$  identity matrix formed from the absorbing states, provided the columns and rows of  $T$  are rearranged so the absorbing states are at the end.

We are in luck, because our matrix already is in this block form with 3 transient states (U, I, S) and 2 absorbing states (LQ, D). This gives  $A$  as a  $3 \times 3$  matrix,  $I$  is a  $2 \times 2$  identity matrix,  $0$  is  $3 \times 2$ , and  $B$  is  $2 \times 3$ . We can add these lines of code to our MATLAB script to extract  $A$  and  $B$ :

```

A=T(1:3,1:3) %transient sub matrix
B=T(4:5, 1:3) %lower left block matrix

```

Now let  $F = \sum_{k=0}^{\infty} A^k = (I - A)^{-1}$ , where  $F$  is sometimes called the **fundamental matrix** of the Markov chain. The  $(i, j)$  entry of  $F$  gives the expected number of steps an individual spends in state  $j$ , given that it began in state  $i$ . To find the total number of time steps an individual in state  $i$  is expected to spend in transient states before being absorbed, we sum each entry in the  $i^{\text{th}}$  row of the fundamental matrix. We could also compute this by multiplying  $FC$  where  $C$  is a column matrix of ones.

```

F=inv(eye(3)-A) %fundamental matrix
F*[1;1;1] %number of steps to reach absorbing state

```

We see that 3.33 is the expected number of weeks in which a person without the disease is eventually either lost/quit or diagnosed, while a person that initially has symptoms will take an average of almost 4.1 weeks before they are lost/quit or diagnosed. A more interesting question might be “if a person is diagnosed with the disease what is the average time to diagnosis after the initial infection?” .

It turns out that the probability of being absorbed in the absorbing state  $i$  when starting from transient state  $j$  is given as the  $(i, j)$ -entry of the matrix product  $BF$ . *Notice the switch in the roles of  $i$  and  $j$  for these probabilities.* By computing this you should see that the probabilities for an individual beginning in the first transient state (uninfected) are again 0.5139 to reach the first absorbing state (lost or quit treatment) and 0.4861 to reach the second absorbing state (diagnosed), exactly as we found earlier. The third column of  $BF$  gives the probability of 0.167 an individual exhibiting symptoms will eventually



be lost or quit treatment and a probability of 0.833 an individual with symptoms will eventually be diagnosed. See [20] for more on this topic.

We can also obtain these absorbing state probabilities from the limiting matrix  $L$ , described in (5.4). For absorbing state Markov models, the limiting matrix will take the block form below.

$$L = \lim_{k \rightarrow \infty} M^k = \lim_{k \rightarrow \infty} T^k = \begin{bmatrix} 0_{t \times t} & 0_{t \times r} \\ B_{r \times t}(I_{t \times t} - A_{t \times t})^{-1} & I_{r \times r} \end{bmatrix}.$$

Notice the lower left block gives the matrix product  $BF$ .

See [20] for more on this topic.

## 5.5 Higher Order and Nonlinear Discrete Dynamical Systems

In this section we will look at more general forms of discrete dynamical systems. These may involve nonlinear terms or higher order recurrence relations involving values at several previous times instead of just the most recent value.

We start with a classic example of a second order recurrence relation.

### Example 5.5.1

The flowering plant commonly called a sneezewort, has a particular growth pattern. Initially the plant grows for two months before it branches, then it continues to branch each following month. Each new shoot also has to grow two months before it branches, then branches each following month. Let  $x_n$  be the number of branches at the beginning of the  $n^{\text{th}}$  month and find a recurrence relation.

.....  
 Starting with a tree with no branches at month 0 give us  $x_0 = 1$ . Since it takes two months to form a branch, we also have  $x_1 = 1$ . At the beginning of month 2, there is a new branch so  $x_2 = 2$ . At month 3, the main trunk forms a new branch but the new branch from the previous month is not yet ready to branch, so  $x_3 = 3$ . To get a recurrence relation, notice that

$$\text{Total number of branches} = \text{Number of branches at previous month} + \text{Number of new branches.}$$

The number of new branches is going to be the number of branches two months ago, since those are the only ones mature enough to form a new branch. This gives us

$$x_n = x_{n-1} + x_{n-2}, \quad x_0 = 1, x_1 = 1.$$

Notice that this gives a famous sequence you have seen before - what is it?

Also recognize this is a second order non-linear nature of this relation since the  $n^{\text{th}}$  value depends on the *two* previous values and not just  $x_{n-1}$ . Both initial conditions of  $x_0$  and  $x_1$  are needed to find  $x_n$  in this case, for any  $n \geq 2$ .

Recall a *linear* recurrence relation has the form  $x_n = f(x_{n-1})$ , where  $f$  is some function of  $x_{n-1}$ . Suppose we want to know if the sequence  $x_n$  converges to some finite value. From Theorem 3.1.5, we know that if  $f$  is a differentiable function with  $|f'(x)| < 1$ , then the sequence is guaranteed to converge to some  $x^*$ , which also satisfies  $x^* = f(x^*)$ . From Definition 3.1.1  $x^*$  is a fixed point or equilibrium of the dynamical system.

We defined stability of equilibria for linear systems in Section 5.2 but need to use a more general definition for the case in which the system is nonlinear. We say that an equilibrium  $x^*$  is **stable** if the following condition holds: For all  $x_0$  sufficiently close to  $x^*$ ,  $x_n$  approaches  $x^*$  as  $n \rightarrow \infty$ . More formally, we say that

the equilibrium is stable if there is a number  $\epsilon > 0$  such that for all  $x_0 \in (x^* - \epsilon, x^* + \epsilon)$ , we have  $x_n \rightarrow x^*$ . An equilibrium  $x^*$  is **unstable** if this condition holds: There exist  $x_0$  arbitrarily close to  $x^*$  such that  $x_n$  does not approach  $x^*$  as  $n \rightarrow \infty$ . More formally, an equilibrium is unstable if for all  $\epsilon > 0$ , there is some  $x_0 \in (x^* - \epsilon, x^* + \epsilon)$  such that  $x_n$  does not converge to  $x^*$ .

As a first exploration into finding and classifying equilibria, we will often use the fixed point definition and an analysis of trajectories. See [Exercise A.0.39](#) for an example.

The following theorem is based on Theorem 3.1.5 and provides conditions on which an equilibrium (once it is known) is stable or unstable according to the epsilon definitions above. We will use the result without proof.

**Theorem 5.5.2.** *Suppose  $x_{n+1} = f(x_n)$  is a dynamical system,  $x^*$  is an equilibrium of the system, and  $f(x)$  is a function that is differentiable on some open interval containing  $x^*$ . Then  $x^*$  is a stable equilibrium if  $|f'(x^*)| < 1$  and it is an unstable equilibrium if  $|f'(x^*)| > 1$ .*

See [Exercise A.0.40](#) for an example of using Theorem 5.5.2 to classify equilibria.

### Example 5.5.3

Recall [Activity A.0.10](#), modeling the concentration of medication in a patient's bloodstream. The concentration of medicine  $M_t$  at the beginning of day  $t$  (immediately after a new dosage has been given) is modeled by Equation 3.1, restated here

$$M_{t+1} = 0.5M_t + 1.$$

In this case, we have  $M_{t+1} = f(M_t)$ , where  $f(x) = 0.5x + 1$ . To find the equilibrium we would solve  $M = 0.5M + 1$  for  $M$ , resulting in  $M = 2$  mg per liter. As you saw in [Activity A.0.10](#), one can verify that  $M_t$  does indeed get closer and closer to 2 as  $t$  increases.

To examine stability, notice that  $f'(x) = 0.5$ , and so  $|f'(2)| = 0.5 < 1$ . By Theorem 5.5.2, we know that the equilibrium is stable. This agrees with what we found previously; that by experimenting with different initial conditions  $M_0$ , the sequence  $M_t$  always converges to 2.

## A System of Two Difference Equations

For our last example, we take a closer look at nonlinear systems with more than one dependent variable. A system with two dependent variables looks like this:

$$x_{n+1} = f(x_n, y_n), \quad y_{n+1} = g(x_n, y_n). \quad (5.6)$$

Earlier in this chapter we looked at this scenario in the case where  $f(x, y)$  and  $g(x, y)$  were both linear functions of  $x$  and  $y$ . If either  $f$  or  $g$  are nonlinear then we cannot model this system using matrices as we did previously. However we can still study what happens in the long term.

We define an equilibrium in almost the same way we did in Section 5.5. An equilibrium is a point  $(x^*, y^*)$  such that both  $x^* = f(x^*, y^*)$  and  $y^* = g(x^*, y^*)$ . Determining stability analytically is more complicated and requires a bit of linear algebra.

### Example 5.5.4

Consider the predator-prey model

$$x_{t+1} = 2(1 - x_t)x_t - \beta x_t y_t$$

$$y_{t+1} = 0.8y_t + 3\beta x_t y_t.$$

Here  $x_t$  and  $y_t$  represent population densities of the prey and predator populations, respectively. The parameter  $\beta$  represents the conversion rate at which an encounter between the two species results in a decrease in prey and an increase in predators. Suppose that the initial populations are  $x_0 = 0.6$ ,  $y_0 = 0.1$ , and assume that  $\beta = 0.5$ . Plot the solution curves of the prey and predator population versus time, as well as a trajectory plot of the two populations together. Can the two populations coexist?

.....

Refer to Example 5.2.3 to see how to generate the sequences  $x_t, y_t$  in MATLAB as well as make the plots shown in Figure 5.1. It turns out that as  $t \rightarrow \infty$ ,  $x_t$  is approaching 0.13 and  $y_t$  is approaching 1.47. We

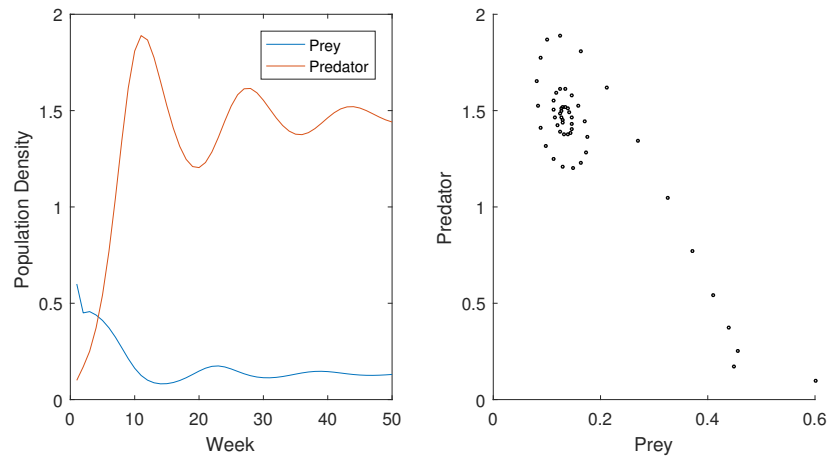


Figure 5.1: Left: Plot of prey and predator populations against time, Right: Trajectory plot

could find the equilibria algebraically by solving the system

$$x = 2(1 - x)x - 0.5xy, \quad y = 0.8y + 3(0.5)xy.$$

Moving terms to one side and factoring results in

$$0 = x(1 - 2x - 0.5y), \quad 0 = y(-0.2 + 3(0.5)x).$$

If  $x = 0$ , then  $y = 0$  so  $(0, 0)$  is one equilibrium value.

On the other hand, if we take the second equation and assume  $-0.2 + 3(0.5)x = 0$ , we get  $x = 4/30$ . Substituting this into the other equation tells us that  $y = 22/15$ , which should agree with the results you obtained from running the MATLAB code. Thus, the two populations can coexist!

---

## 5.6 Exercises

1. Suppose a non-profit organization starts an endowment fund with \$200 million. Each year they plan to spend 30% of all funds above \$80 million on operations, invest the remainder of all funds at 8% interest, and collect \$40 million in donations.
  - (a) Devise a model for the amount in the endowment fund  $E_t$  at the beginning of each year  $t$ .
  - (b) Find the equilibrium and determine its stability for this model. Discuss whether this spending strategy is sustainable.
3. A recent college graduate has a loan of \$20,000 with a monthly interest rate of 0.4%. Each month she pays \$300 towards the loan.
  - (a) Find a difference equation model for the amount of the loan  $P_t$  at month  $t$  from now. If you'd like a challenge, find a closed form equation with  $P_t = f(t)$  for any month  $t$ .
  - (b) Determine how long it will take for the loan to be paid off.
  - (c) Suppose she cannot afford \$300 per month towards the loan payment. What does the monthly payment need to be in order to pay off the loan by 10 years?

4. Consider a breathing process where a lung has a volume of  $V$  liters when full. With each breath, some fraction  $q$  of the air is exhaled and replaced with ambient air. Suppose the lung contains some chemical compound with concentration  $c_t$  at breath  $t$ . The ambient air has a concentration of  $\gamma$  of this compound. We want to derive a recurrence relation involving  $c_t$ . Let  $a_t$  be the amount of compound in the lungs at time  $t$ , so that  $a_t = c_t V$ .
  - (a) Use the fact that
 
$$a_{t+1} = a_t - \text{Amount of compound exiting lung} + \text{Amount of compound entering lung}$$
 to find a difference equation for  $a_t$ .

- (b) Use your answer for (a) along with the fact that  $a_t = c_t V$  to devise a difference equation for the concentration  $c_t$ .
  - (c) Suppose that the lung capacity is 3 liters, 0.6 liters is exhaled at each breath, and that the ambient concentration of the compound is 5.0 millimoles per liter. Find  $q$ ,  $\gamma$ , and give the difference equation for  $c_t$ .
  - (d) Find the equilibrium concentration, and determine whether it is stable or unstable.
  - (e) With the above parameter values, how many breaths does it take for the compound concentration in the lungs to be within 0.1 millimoles per liter of equilibrium? Make an assumption about the average length of a breath to determine how long this is in minutes.
5. Consider the basic discrete-time logistic growth model  $x_{t+1} = rx_t(1 - x_t)$  with growth rate  $r$ , assuming  $x_t$  is a density per unit area at time  $t$ .
  - (a) Show that the equilibria are  $x = 0$  and  $x = (r - 1)/r$ .
  - (b) Show that  $x = 0$  is stable when  $r < 1$ .
  - (c) Show that  $x = (r - 1)/r$  is stable when  $1 < r < 3$ .
  - (d) Let  $x_0 = 0.3$ , and pick some  $r$  in the interval  $(0, 1)$ . Use technology to make a plot of  $x_t$  for integer values  $0 \leq t \leq 50$ .
  - (e) Let  $x_0 = 0.3$ , and pick some  $r$  in the interval  $(1, 2)$ . Use technology to make a plot of  $x_t$  for integer values  $0 \leq t \leq 50$ .
  - (f) Suppose  $x_0 = 0.3$  and  $r = 4$ . What is  $x_{50}$ ?
  - (g) Suppose  $x_0 = 0.30001$  and  $r = 4$ . What is  $x_{50}$ ?

- (h) If  $r = 4$ , explain what potential problems may arise with this model.
6. Ricker's equation is a common population model used to model fish populations. The discrete version takes on the form:

$$N_{t+1} = N_t \exp(r(1 - N_t/k)),$$

where  $N_t$  is the population density at generation  $t$ , and  $r$  and  $k$  are positive constants.

- (a) Find two equilibrium values. Determine the stability of each (the answer may depend on parameters  $r$  and  $k$ ).
- (b) Let  $N_0 = 100, k = 200$  and run simulations of  $N_t$  for various values of  $r$ . Summarize the effect that  $r$  has on the population. Discuss whether the population is sustained or dies out, and whether it oscillates.
7. Consider the sequence

$$x, \sqrt{1+x}, \sqrt{1+\sqrt{1+x}}, \sqrt{1+\sqrt{1+\sqrt{1+x}}}, \dots$$

- (a) Write this sequence as a recurrence relation of the form  $x_{n+1} = f(x_n)$ .
- (b) Find the exact equilibrium for the recurrence relation, and determine whether or not it is stable.
8. Adapted from [9]. Suppose insurgent forces have a strong foothold in a city named Urbana, a large metropolis in the country of Ibebest. Intelligence estimates that the insurgents currently have a force of about 1,000 fighters. The local police force has approximately 1,300 officers, many of whom have no formal training for this type of situation. Based on data collected over the previous year, approximately 8% of insurgents switch sides and join the police each week, whereas about 11% of police switch sides and join the insurgents. Intelligence also estimates around 120 new insurgents arrive from a neighboring country of Ubeworst each week. Recruiting efforts in Ibebest yield about 85 new police recruits each week as well. In armed conflict with insurgent forces, the local police are able to capture or kill approximately 10% of the insurgent force each week on average, while losing about 3% of the police forces.
- Determine a dynamical system describing both the insurgent and police forces for any week in the future.
  - Determine an equilibrium state, if one exists.
  - Plot the two police forces over the next year.
  - Adjust the values in your model to determine those which will enable a reversal of the outcome predicted by the current parameters.

9. Each matrix  $A$  defines a dynamical system  $\mathbf{x}_{t+1} = A\mathbf{x}_t$ .

a) Use the eigenvalues of  $A$  to classify the origin as an attractor, repeller, or saddle point of the dynamical system.

b) Use the initial value  $\mathbf{x}_0 = (1, 5)^T$  and make a trajectory plot for the  $A_{2 \times 2}$  case. Experiment with changing the number of iterations until you can clearly see the pattern of the trajectory and verify that it behaves in the expected manner. Set the appropriate x- and y-limits on the plot so that the trajectory pattern can be clearly seen. See Chapter 2 for a reminder on MATLAB commands or use the `help` menu.

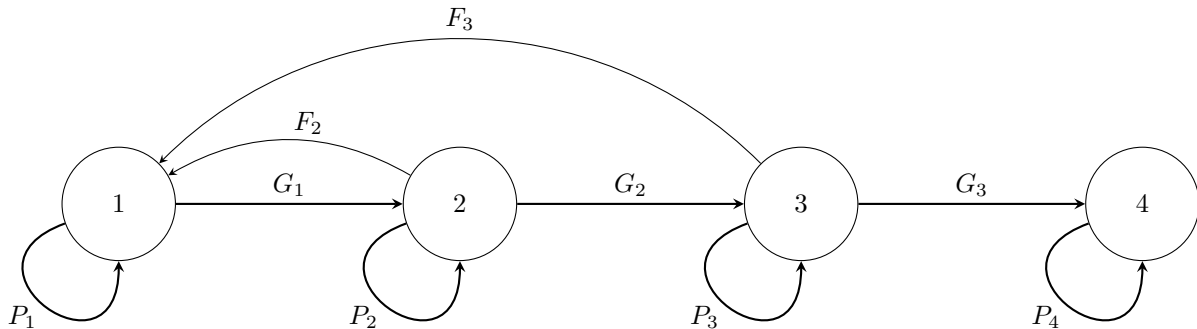
i  $A = \begin{bmatrix} 1.7 & -0.3 \\ -1.2 & 0.8 \end{bmatrix}$

ii  $A = \begin{bmatrix} 0.4 & 0.5 \\ -0.4 & 1.3 \end{bmatrix}$

iii  $A = \begin{bmatrix} 0.8 & 0.3 \\ -0.4 & 1.5 \end{bmatrix}$

$$\text{iv } A = \begin{bmatrix} 0.6 & 1.4 \\ -0.2 & 0.8 \end{bmatrix}$$

10. (Adapted from [3]) Below is the stage diagram for a population (called a pod) of killer whales where only the females are modeled. Assume the females are categorized into four stages: yearlings (1), juveniles (2), mature adults (3), and post-reproductive adults (4). Yearlings are whales up to 1 year old. The age ranges of juveniles, mature adults, and post-reproductive adults varied depending on signs of reproductive ability. Appropriately, we will take a time step of 1 year for this model.



- (a) Suppose the values for each  $G_i, P_i, F_i$  are given below. Determine the projection matrix  $A$  in the matrix model  $\mathbf{x}_{t+1} = A\mathbf{x}_t$ , where  $\mathbf{x}_t$  gives the population in each of the four killer whale stages at year  $t$ .

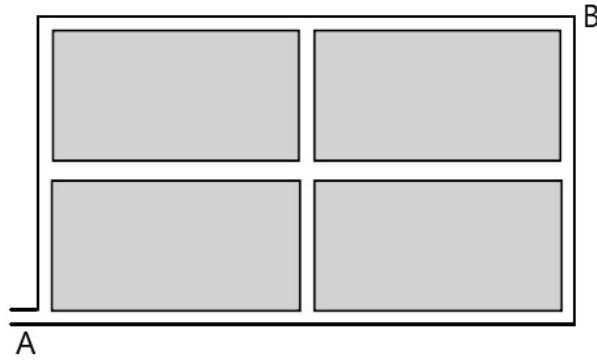
$$\begin{aligned} G_1 &= 0.9775, & G_2 &= 0.0736, & G_3 &= 0.0452, \\ P_1 &= 0, & P_2 &= 0.9111, & P_3 &= 0.9534, & P_4 &= 0.9804, \\ F_2 &= 0.0043, & F_3 &= 0.1132 \end{aligned}$$

- (b) What can you say about the number of years included in each stage?  
 (c) What proportion of post-reproductive females die each year?  
 (d) Will this pod of killer whales grow and thrive over time?  
 (e) Scientists believe the post-reproductive females are vital to the survival and health of each killer whale pod, particularly influencing the younger generation. Among other things, post-reproductive females provide extra fish to the young whales and carry the knowledge for locations of food sources and potential environmental threats. At equilibrium, what percentage of these “grandma” whales persist in the population?
11. In this exercise you will construct a Leslie matrix model for a northern American red squirrel population. with the following assumptions:
- Only female squirrels are modeled; assume “squirrel” means “female squirrel”.
  - Birth rates are presented (at half the total) assuming a 50-50 ratio of female to male squirrels.
  - Squirrels 0 - 1 years old do not produce offspring and 22% survive their first year.
  - Squirrels 1 - 2 years old produce 1.6 offspring per year and 60% survive their second year.
  - Squirrels 2 - 3 years old produce 2.1 offspring per year.
  - For this model we assume squirrels do not live more than 3 years.

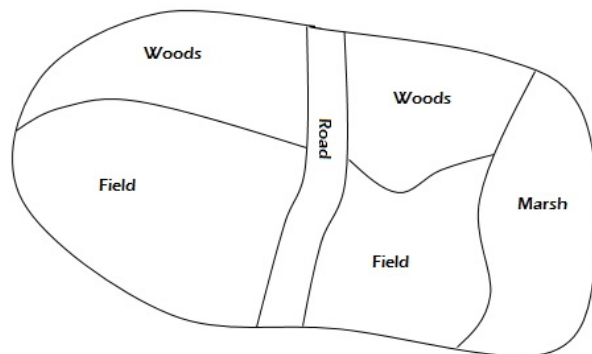
This divides the squirrel population up into three life “stages”: call them juvenile, adult, and mature.

- (a) Draw a state diagram to describe the flow of squirrels between stages.  
 (b) Suppose there are currently 80 juvenile squirrels, 70 adult squirrels, and 50 mature squirrels. Calculate the number of squirrels in each age category that there will be next year.

- (c) Let  $j_t, a_t,$  and  $m_t$  denote the number of juvenile, adult, and mature squirrels at year  $t$ . Find a system of three difference equations for  $j_t, a_t,$  and  $m_t$ .
- (d) Now define a matrix  $A$ , and a vector  $\mathbf{x}_t$  (where  $t$  is the number of years), so that the model is given in the form of a dynamical system  $\mathbf{x}_{t+1} = A\mathbf{x}_t$ . Use this to compute the number of squirrels in each age category there will be after 10 years.
- (e) Compute the spectral radius, and describe the long term behavior of the system. Determine whether the squirrel populations survives, and if so, find the growth rate. Also, determine what percent of the population will be juvenile, adult, and mature squirrels.
- (f) Due to a change in environment, suppose the survival rate from year 1 to 2 drops from 22% to 15% and the survival rate from year 2 to 3 drops from 60% to 50%. Discuss the long term impact this has on the squirrel population.
- (g) Generalize the process you did in part (b). Suppose at time  $k = 0$ , there are  $j_0$  juvenile squirrels,  $a_0$  adult squirrels, and  $m_0$  mature squirrels. In each category there is a specific fertility rate, call them  $f_1, f_2, f_3$ . There are also survival rates: let  $s_1$  be the survival rate from juvenile to adult and  $s_2$  be the survival rate from adult to mature. Use this to construct the general matrix  $A$  associated with the dynamical system  $\mathbf{x}_{t+1} = A\mathbf{x}_t$ .
- (h) Now suppose we change our assumptions to allow squirrels to live up to 8 years. The first 3 years will be the same as given originally. For future years, assume the same birth rates as mature squirrels (2.1 offspring per year) and a 10% decrease in survival rate each following year (50%, 40%, etc.). What effect does this have on the population?
12. In this exercise you will construct another Leslie matrix model. See Exercise 11 and 10 Consider a species of fish that only reproduces during its third year and then dies. An initial population of 1000 newborn fishes is introduced into a fishery. During the first year, 30% survived. Of the ones that survived the second year, 60% survive to their reproduction age. Assume that fish produce 6 offspring during this year.
- (a) Let  $n_t, j_t,$  and  $m_t$  denote the number of newborn, juvenile, and mature fish at year  $t$ . Find a system of three difference equations for  $j_t, a_t,$  and  $m_t$ .
- (b) Define a matrix  $A$ , and a vector  $\mathbf{x}_t$  (where  $t$  is the number of years), so that the model is given in the form of a dynamical system  $\mathbf{x}_{t+1} = A\mathbf{x}_t$ .
- (c) Compute the spectral radius, and describe the long term behavior of the system. Determine whether the fish populations survives, and if so, find the growth rate. Also, determine what percent of the population will be newborn, juvenile, and mature.
13. Let  $P$  be a transition matrix for a Markov chain. What can we say about the entries of  $P^n$  in terms of the limiting distribution  $x^*$ ?
14. Pick an arbitrary probability vector  $\mathbf{x}_0$ . Find the transition vectors  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$  for the Markov chain with the transition matrix  $P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ . Do the same for the transition matrix  $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ . Interpret what happens in each of these processes.
15. Prove Proposition 5.2.2 for the cases  $k = 1, k = 2, k = 3$ . Hint: Apply Proposition 5.2.1 to  $\mathbf{x}_0$ .
16. See [17]. A whimsical cyclist cycles through the neighborhood shown in the figure below. He begins at location  $A$  and traverses the streets at random, staying in the neighborhood the entire time. During each time interval he either rests at an intersection or pedals exactly one block. Suppose that at each intersection the cyclist is three times as likely to pedal as to rest. If he pedals, he is equally likely to take any street open to him. Label each intersection and form a Markov chain model for this situation.



17. [17]. In the setting described in Exercise 16, suppose the cyclist never rests, that at any intersection he is equally likely to take any street available to him, and that once he reaches location  $B$  he stays there. Form a new Markov chain model with these assumptions.
18. An old model suggests that a man's profession is classified as professional, skilled laborer, or unskilled laborer. Assume that, of the sons of professional men, 80 percent are professional, 10 percent are skilled laborers, and 10 percent are unskilled laborers. In the case of sons of skilled laborers, 60 percent are skilled laborers, 20 percent are professional, and 20 percent are unskilled. Finally, in the case of unskilled laborers, 50 percent of the sons are unskilled laborers, and 25 percent each are in the other two categories. Assume that every man has at least one son, and form a Markov chain by following the profession of a randomly chosen son of a given family through several generations.
- Set up the matrix of transition probabilities.
  - Find the probability that a randomly chosen grandson of an unskilled laborer is a professional man.
  - What assumptions are being made in this model? Does the model assume the Markov property holds?
  - From this model, what distribution of professions can we expect in the male population over many generations? Is this likely to occur in reality? Why or why not?
19. (Adapted from [17]) A deer's range is shown in the figure below. The location of the deer is noted every hour, and every time it either stays in its current area or moves to an adjacent area. For the purposes of this model, adjacent areas must share an edge or share a length of road. *e.g. The woods on the left of the road are adjacent to the woods on the right of the road, but not adjacent to the field on the right of the road.* Suppose the deer is twice as likely to remain where it is as to move. If the deer does move, each option that does not involve crossing the road is equally likely. In addition, if the deer moves, each option that does not require it to cross the road is three times as likely to be selected as an option that involves crossing the road.



- (a) Find an appropriate transition matrix  $P$ .



- (b) Find the limiting matrix of  $P$  and interpret it.
20. At a certain university, suppose each student moves on to the next year of classes with probability  $p$ , stays in the same class with probability  $r$ , and leaves the university for reasons other than graduating with probability  $q$ . Assume there are 4 classes, and after moving on from class 4, the student graduates and becomes an alumni of the university.
- Sketch a state diagram and determine the transition matrix. What are the absorbing states?
  - If  $p = .7, q = .2, r = .1$ , find out the average amount of time a student spends in college. Discuss your result.
  - Determine the probability a student currently in their third year of classes will graduate from the university.
21. Consider a simplified model for the damage induced by repeated loads placed on a composite material (e.g. fiberglass, metal composites, plastic composites, etc.), once per day. Between loads placed on the material, assume a technician will repair any fixable elements of the material so that the material is in one of three states: *intact*, *damaged and fixable*, and *damaged beyond repair*. With each additional load, suppose the laminate that is intact remains intact with probability 69.9%, but will be damaged beyond repair with probability 0.1%. A laminate that is damaged and fixable will be repaired by the technician so it becomes intact with probability 49.8%, but will become damaged beyond repair with probability 0.2%. Of course a laminate that is damaged beyond repair will remain so with probability 1.
- Draw a state diagram for this scenario assuming a step of one day. Label the states as transient and absorbing.
  - Formulate the transition matrix so it is in the appropriate block form.
  - Compute the fundamental matrix and interpret the results. How long with the composite last? How much of that time will it be intact, and how much of that time will it be damaged (fixable or beyond repair)?
22. The Nicholson-Bailey model [21] for a host-parasite interaction is given by

$$\begin{aligned} N_{t+1} &= \lambda N_t e^{-aP_t}, \\ P_{t+1} &= cN_t(1 - e^{-aP_t}), \end{aligned}$$

where  $N_t$  is the host population and  $P_t$  is the parasite population at generation  $t$ ,  $a$  is the searching efficiency of the parasites,  $\lambda$  is the host reproductive rate, and  $c$  is the average number of viable eggs laid by a parasite per host.

- Suppose that  $N_0 = 15, P_0 = 8, a = 0.02, c = 1$ . Make plots of  $N_t$  and  $P_t$  against time.
- Plot the trajectory of  $P_t$  vs.  $N_t$  with the conditions above.
- Find the equilibrium or equilibria.
- Now plot the trajectory of  $P_t$  vs.  $N_t$  again, but with an initial condition very near the nontrivial equilibrium you found in part (c). What can you conjecture about the stability of this equilibrium? Can the host and parasite coexist?

23. (Adapted from [1]) Suppose there are two types of bacteria: an original type (sometimes called the wild type) with population  $b_t$  at time  $t$ , and a mutant type with population  $m_t$ . Suppose the original type has a per capita production of 1.5 and the mutant type has a per capita production of 2.0. This leads to the dynamical system

$$b_{t+1} = 1.5b_t, m_{t+1} = 2.0m_t.$$

Define a new variable  $p_t = \frac{m_t}{m_t + b_t}$ .

- What does  $p_t$  represent?
  - If  $m_t = 2.0 \times 10^5$  and  $b_t = 3.0 \times 10^6$ , what is  $p_t$ ?
  - If  $m_t = 2.0 \times 10^5$  and  $b_t = 3.0 \times 10^6$ , what fraction does the original type make up of the total population?
  - Find a difference equation for  $p_t$  that is not expressed in terms of  $m_t$  or  $b_t$ . Hint: first find  $p_{t+1}$ , then use the fact that the fraction of original type and the fraction of the mutant type must sum to one.
  - Find the equilibrium for  $m_t$  and determine its stability. What eventually happens to the bacteria?
24. Consider the symbiotic model below.

$$\begin{aligned} x_{t+1} &= x_t * \exp\left(\frac{2 + 1.4 * y_t}{1 + y_t} - x_t\right) \\ y_{t+1} &= y_t * \exp\left(\frac{1 + 1.2 * x_t}{1 + x_t} - y_t\right) \end{aligned}$$

Assume  $x_t$  and  $y_t$  are given in population densities per unit area.

- Using  $x_0 = 1, y_0 = 0.5$ , find  $x_t$  for integer values of  $0 \leq t \leq 20$ . Plot  $x_t$  and  $y_t$  against time, and also plot the trajectories.
- What do the populations  $x_t$  and  $y_t$  approach as  $t$  gets large?
- What are the maximum values of the two populations on the given time interval, and at what times do these occur?
- In this context, a symbiotic relationship is one which is beneficial to both species. Does this appear to be the case here? Explain.

## Chapter 6

# Continuous Models

In the last chapter, phenomena was modeled at discrete time steps. Continuous models on the other hand keep track of what is happening on a continuous time scale. For population dynamics that change every hour for many years, we may approximate this with continuous time rather than taking thousands of discrete time steps. For another example, consider a chemical reaction that changes slowly at first then abruptly some time later. The resulting model often takes on the form of differential equations, as opposed to difference equations. As many of these differential equations do not have closed form solutions, it is necessary to understand numerical methods as well.

A preliminary introduction of modeling with differential equations was given in Section 1.4 along with exercises in 1.6. You may wish to refer back to these for a review before proceeding with the topics and exercises in this chapter.

### 6.1 Modeling with Differential Equations

Some of the common continuous population models you may have seen in calculus include the exponential model and logistic model. We will revisit these, and also study a variety of other models.

Recall that in the discrete Malthusian population model (5.1), the rate of change of growth was proportional to the current population. Describing this phenomenon with a differential equation gives us the continuous Malthusian model, also known as the exponential growth model. Letting  $N(t)$  be the population at time  $t$  we have

$$\frac{dN}{dt} = rN(t), \quad N(0) = N_0 \quad (6.1)$$

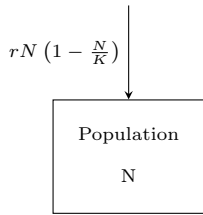
where  $r$  is some growth constant. We say that (6.1) is a continuous version of (5.1). This is a simpler version of the model we solved in Example 1.4.1. See also [Exercise A.0.43](#) and [A.0.44](#).

The continuous version the logistic model (5.2) is

$$\frac{dN}{dt} = rN \left( 1 - \frac{N}{K} \right), \quad N(0) = N_0. \quad (6.2)$$

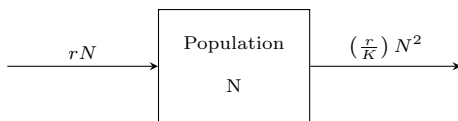
Here,  $K$  is the carrying capacity, and  $N(t)$  is the population at time  $t$ . Refer to [Exercise A.0.45](#) to discover a couple important properties of the logistic model. This shows us that when the population is small compared to the carrying capacity, the model is very close to the basic Malthusian model of exponential growth. For small population values, population growth is unrestricted by the carrying capacity. On the other hand when the population approaches the carrying capacity, the ratio  $N/K$  will approach 1, and we will have  $N'(t) \approx 0$ . This means that the growth rate will eventually slow to zero to account for the limited population that the environment is able to support.

As with the discrete case, a diagram can be helpful in developing our model, particularly in terms of inputs and outputs to our quantity of interest. In a continuous model, the rate of change of our quantity of interest is equivalent to the sum of the positive rates causing an increase and the negative rates causing a decrease in the quantity of interest. The **compartment diagram** below shows the logistic model.



This gives rise to the differential equation 6.2. Notice that proportional rates in this type of diagram must show a product with the population  $N$ . Different from the discrete stage diagrams with survival rates for example, there is no implied product when an arrow stems from a particular quantity. This can be especially helpful when a rate is dependent on multiple variable quantities.

With a single positive rate in, it is easy to assume  $rN(1 - \frac{N}{K}) > 0$ . However, if a situation arises where  $N > K$ , then the rate is negative. To show this more explicitly in the diagram, we can separate the rate into its positive and negative components. In particular, the term due to natural birth  $rN$  is always positive and causes an increase in the population, while the competition term  $(\frac{r}{K})N^2$  is always negative and causes a decrease in the population.



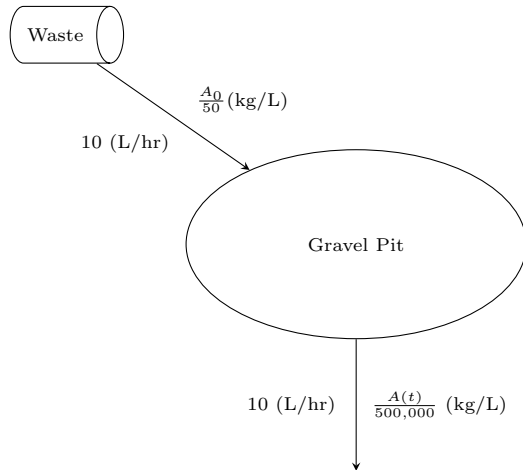
The differential equation in this case could be written as  $\frac{dN}{dt} = rN - (\frac{r}{K})N^2$ .

**Example 6.1.1**

Suppose a rusting 50 gallon drum of hazardous waste is beginning to leak near a gravel pit that people have been using to swim. The container of waste contains  $A_0$  kilograms of dissolved contaminants. The gravel pit is filled with a volume of approximately 500,000 gallons of water. The waste water flows slowly into the gravel pit at 10 gallons per hour with a concentration of  $C(t)$  kilograms of contaminant per gallon of water. For simplicity we assume the contaminants are mixed uniformly throughout the water in the pit and the water mixture flows out at a rate of 10 gallons per hour. How many grams of contaminant are there in the gravel pit at time  $t$ ?

.....  
 As the volume and time are changing continuously, we consider the rate of change of our quantity of interest,  $A$  grams of contaminant, with respect to time  $t$  in hours. Notice the units of rates in and rates out must also be in grams per hour. Use this as a key to check your work.

Before jumping to an equation, begin with your own sketch or a compartment diagram of the situation.



We have the concentration and flow rate of waste flowing into the gravel pit, which gives an increase to the contaminant at  $(A_0 \text{ kg}/50 \text{ L}) \cdot 10 \text{ (L/hr)} = \frac{A_0}{5} \text{ (kg/hr)}$ .

For the concentration of waste flowing out of the gravel pit, note that the rate of fluid into the gravel pit is equivalent to the rate of fluid leaving the pit. This gives no change to the total volume of water in the pit. Thus the rate of waste leaving the pit per hour is given by the unknown concentration  $A(t) \text{ kg}/500,000 \text{ gal}$  multiplied by the flow rate  $10 \text{ (gal/hr)}$ .

The differential equation is then

$$\frac{dA}{dt} = \frac{A_0}{5} - \frac{A(t)}{50,000}.$$

Notice, this equation can be solved by a method of integrating factors which we show here as a reminder.

$$\frac{dA}{dt} = \frac{A_0}{5} - \frac{A}{50,000}$$

Adding  $A/50,000$  to both sides and multiplying by the integrating factor  $e^{(1/50,000)t}$  gives

$$\left( \frac{A}{50,000} + \frac{dA}{dt} \right) e^{(1/50,000)t} = \frac{A_0}{5} e^{(1/50,000)t}$$

which you can verify is equivalent to

$$\frac{d}{dt} \left( A e^{(1/50,000)t} \right) = \frac{A_0}{5} e^{(1/50,000)t}.$$

Integrating both sides with respect to  $t$  gives

$$A e^{(1/50,000)t} = 10,000 A_0 e^{(1/50,000)t} + C$$

for some constant  $C$ . Solving for  $A$  then gives

$$A = 10,000 A_0 + C e^{-(1/50,000)t}.$$

The initial amount of waste in the gravel pit must be 0, which gives  $C = -10,000 A_0$  and the final solution

$$A(t) = 10,000 A_0 \left( 1 - e^{-(1/50,000)t} \right).$$

Given this solution, what can you say about the eventual concentration of waste in the gravel pit? At what point does this model stop being valid?

Another extension of this model may be to revisit the assumption for uniform mixing of the incoming contaminants with the water in the gravel pit. This assumption may not be realistic in this case since the waste comes into the gravel pit at a slow rate and the water in a pit generally has minimal turbulence for mixing. First, imagine a fluid mixing problem where this assumption would be more appropriate, such as a lake with frequent boat traffic or a water purification system that continually churns the water. Second, consider the changes you might make if the volume of water (and contaminant) flowing in is different than the volume of water (and contaminant) flowing out. How would the total volume change with time? In what way does this impact the concentrations flowing in and out?

See **Exercise A.0.46** for an example on drug administration.

### 6.1.1 Euler's Method

A differential equation of the following form, along with the initial condition given below

$$\begin{aligned}\frac{dx}{dt} &= f(t, x), & t_0 \leq t \leq t_f \\ x(t_0) &= x_0\end{aligned}$$

is called an **initial value problem**. For some common forms of  $f$ , we may be able to solve for  $x(t)$  analytically, using separation of variables or integration by parts, for example. For other forms of  $f$ , we might not be able to solve for  $x(t)$  analytically and instead must utilize a numerical approximation technique. The simplest such approximation technique is **Euler's Method**. Recall that according to the definition of the derivative,

$$x'(t_0) \approx \frac{x(t_0 + h) - x(t_0)}{h}$$

when  $h$  is small. If we solve this for  $x(t_0 + h)$ , we get the linear approximation of  $x$  at  $t_0$

$$\begin{aligned}x(t_0 + h) &\approx x(t_0) + x'(t_0)h \\ &= x(t_0) + f(t_0, x(t_0))h.\end{aligned}$$

We first divide the time interval  $t_0 \leq t \leq t_f$  into equal sub-intervals of size  $h$ . Given an initial point  $(t_0, x(t_0))$ , we then use the derivation above to approximate the solution at  $t_0 + h$ . To approximate the solution at  $t_0 + 2h$ , we use the linear approximation at  $x(t_0 + h)$ , given by

$$\begin{aligned}x(t_0 + 2h) &\approx x(t_0 + h) + x'(t_0 + h)h \\ &= x(t_0 + h) + f(t_0 + h, x(t_0 + h))h,\end{aligned}$$

and so on. This gives approximate solutions at each point on the interval  $t_0, t_0 + h, t_0 + 2h, \dots, t_f$ .

This process is called Euler's Method. Generally speaking, we have that

$$\text{New } x = \text{Old } x + \Delta x = \text{Old } x + \text{Slope at old point} \cdot h$$

The value  $h$  is sometimes called the **step size** and can also be denoted using  $\Delta t$ . We will see that smaller step sizes give more accurate results.

**Euler's Method Algorithm to approximate the solution to**  
 $\frac{dx}{dt} = f(t, x), x(t_0) = x_0$  **on the interval**  $t_0 \leq t \leq t_f$

Initialize the step size  $h$

$$n := (t_f - t_0)/h$$

Initialize  $x$  to be a vector of size  $n + 1$

Initialize  $t$  to be a vector of  $n + 1$  equally spaced numbers from  $t_0$  to  $t_f$

For  $i = 1 : n$

$$x_{i+1} := x_i + f(t_i, x_i)h$$

End For

Return  $x_{n+1}$ , the approximation of  $x(t_f)$ .

Instead of using a for loop, you could instead use a while loop and continue until  $t$  reaches the final time  $t_f$ . We used the relation  $n = \frac{t_f - t_0}{h}$ , but in some cases you have to round the quantity to the nearest integer in the case where the length of the time interval is not divisible by  $h$ .

### Example 6.1.2

Approximate the solution to the initial value problem

$$\frac{dx}{dt} = x(1 - x), \quad x(0) = 0.1 \quad (6.3)$$

on the interval  $[0, 8]$  using Euler's method. Notice that this equation is simply the logistic growth model with carrying capacity  $M = 1$ . To estimate the solution of  $x(t)$ , we will apply Euler's method using the function  $f(t, x) = x(1 - x)$  with  $x(0) = 0.1$ . Using a step size of  $h = 0.4$ , we get the estimate

$$\tilde{x}(0 + h) = x(0) + hf(0, x(0)) = 0.1 + 0.4(0.1(1 - 0.1)) = 0.1360.$$

We now have  $\tilde{x}(0.4) = 0.1360$ . The next iterate is

$$\tilde{x}(0.8) = \tilde{x}(0.4) + hf(0.4, x(0.4)) = 0.1360 + 0.4(0.1360(1 - 0.1360)) = 0.1830.$$

Continuing in this fashion until  $t = 8$ , we arrive at  $\tilde{x}(8) = 0.9988$ . The values of  $\tilde{x}$  are plotted at each step in Figure 6.1, along with the values of the analytic solution of (6.3).

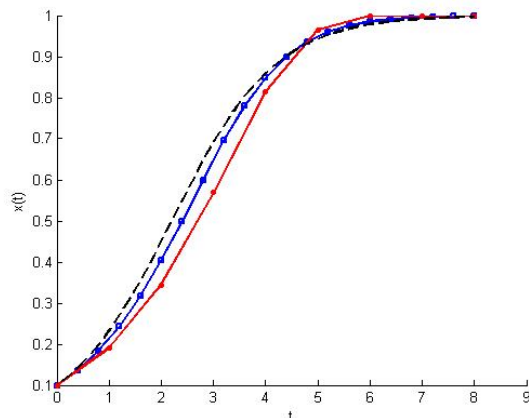


Figure 6.1: Result of using Euler's method to solve  $dx/dt = x(1 - x), x(0) = 0.1$ . Dashed line is analytic solution, solid lines are approximations using Euler's method with  $h = 0.1$  (middle, in blue) and  $h = 0.4$  (bottom, in red).

The approximations from Euler's method are a good estimate of the solution in this case. The solution of (6.3) has the property that  $x \rightarrow 1$  as  $t \rightarrow \infty$ , which is the case for the approximate values of  $x$  generated by Euler's method. There are other cases for which Euler's method does a poor job approximating the solution.

The initial value problem in [Activity A.0.47](#) is  $\frac{dx}{dt} = t(x - 1), x(0) = 2$ . The solution is concave up when  $x > 0$ , so Euler's method consistently underestimates the solution as shown in Figure 6.2. The error in this case increases exponentially with  $t$ .

An upper bound of the error at each step is given by Theorem 6.1.3.

**Theorem 6.1.3.** *Let  $x$  be the solution to the initial value problem  $x'(t) = f(t, x), x(t_0) = x_0$ . Suppose that there is a constant  $M$  such that  $|x''(t)| \leq M$  for all  $t \in [t_0, t_n]$ . In addition, suppose that there is a constant  $L$  such that  $|f(t, x) - f(t, y)| \leq L|x - y|$  for all  $x, y$ . Then the error at time  $t_n$  satisfies*

$$|x(t_n) - x_n| \leq \frac{hM}{2L} \left( e^{L(t-t_0)} - 1 \right).$$

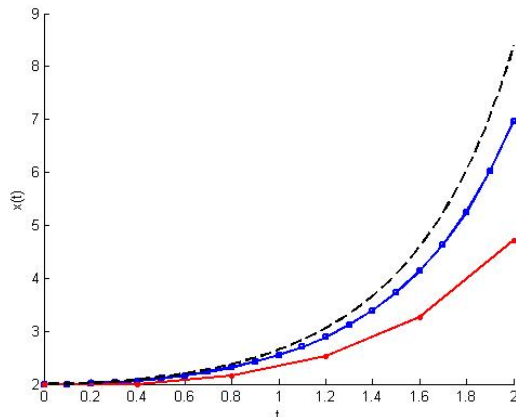


Figure 6.2: Result of using Euler's method to solve  $\frac{dx}{dt} = t(x-1)$ ,  $x(0) = 2$ . Dashed line is analytic solution, solid lines are approximations using Euler's method with  $h = 0.1$  (middle, in blue) and  $h = 0.4$  (bottom, in red).

This theorem says that if  $f$  is reasonably well behaved, the error can potentially increase exponentially as  $t - t_0$  increases for a fixed value of  $h$ . It also says that the maximum error decreases proportionally to  $h$ .

A method has **global truncation error of order  $h^k$**  if the error bound satisfies

$$\text{Error} = Ch^k \quad (6.4)$$

where  $C$  is a constant that does not depend on  $h$ . For short we say that the method has global truncation error  $O(h^k)$ . For example, the global truncation error of Euler's method is  $O(h)$ .

Taking the logarithm of both sides of (6.4), we get  $\log(\text{Error}) = \log(C) + k \log(h)$ . This suggests that if we plot the log of the errors against the log of  $h$ , the slope  $k$  is the approximate order of the method.

**Remark:** The global truncation error of a method refers to the maximum error over multiple time steps. The local truncation error refers to the error in a single step,  $t_i$  to  $t_i + h$ .

We would like a better way to reduce the error, without making  $h$  impossibly small. It turns out that there are many variations on Euler's method that have improved error bounds. Below, we discuss the Improved Euler's method, which falls into a larger class of methods known as Runge-Kutta methods. These methods are used to reduce the accumulation of errors and increase the stability of the sequence. See Exercise 5 for an example of what can go wrong with Euler's method.

### 6.1.2 Improved Euler's Method

There are several algorithms related to Euler's method that do a nice job approximating a solution, and avoid the error spiraling out of control as in the case of Activity A.0.47. The simplest such algorithm is called the Improved Euler's Method. Recall that for Euler's Method we used the approximation

$$x'(t) \approx \frac{x(t+h) - x(t)}{h}.$$

However, we could have instead used

$$x'(t+h) \approx \frac{x(t+h) - x(t)}{h}.$$



It seems that the expression  $\frac{x(t+h)-x(t)}{h}$  should be close to the average of  $x'(t)$  and  $x'(t+h)$ . That is,

$$\frac{x(t+h) - x(t)}{h} \approx \frac{x'(t) + x'(t+h)}{2}.$$

Solving for  $x(t+h)$  gives

$$x(t+h) \approx x(t) + \frac{h}{2}(x'(t) + x'(t+h)).$$

Substituting the differential equation  $x'(t) = f(t, x(t))$  into the expression above, we obtain

$$x(t+h) \approx x(t) + \frac{h}{2}(f(t, x(t)) + f(t+h, x(t+h))).$$

The problem is that  $x(t+h)$  appears on the right hand side of the above expression. To resolve this, we use the regular Euler method where we see  $x(t+h)$  on the right hand side, by setting  $x(t+h) := x(t) + hf(t, x)$ . Putting these together, we get the Improved Euler's Method:

$$x(t+h) \approx x(t) + \frac{h}{2}(f(t, x(t)) + f(t+h, x(t) + hf(t, x))). \quad (6.5)$$

#### Improved Euler's Method Algorithm

```

Initialize the step size  $h$ 
 $n := (t_f - t_0)/h$ 
Initialize  $x$  to be a vector of size  $n + 1$ 
Initialize  $t$  to be a vector of  $n + 1$  equally spaced times from  $t_0$  to  $t_f$ 
For  $i = 1 : n$ 
     $k_1 := f(t_i, x_i)$ 
     $k_2 := f(t_{i+1}, x_i + k_1 h)$ 
     $x_{i+1} = x_i + \frac{h}{2}(k_1 + k_2)$ 
End For
Return  $x_{n+1}$ , the approximation of  $x(t_f)$ .

```

#### Example 6.1.4

Consider the initial value problem  $x' = t(x - 1)$ ,  $x(0) = 2$ , and suppose we want to find the solution at  $t = 1$ . This particular problem can be solved using separation of variables, and it turns out that  $x(1) = e^{1/2} + 1 \approx 2.6487$ . We may therefore compare the errors for Euler's method and the Improved Euler's method for various step sizes  $h$ .

$h$	Euler method error	Improved Euler method error
0.1	0.1016	0.00084
0.01	0.0109	$7.03 \times 10^{-6}$
0.001	0.0011	$6.89 \times 10^{-8}$
0.0001	0.00011	$6.871 \times 10^{-10}$

A plot will verify that the error in Euler's method decreases linearly with  $h$ , which is suggested by the error bound formula in Theorem 6.1.3. On the other hand, the error in the improved Euler's method decreases at a faster rate.

It turns out that the maximum error obtained from the improved Euler's method decreases at a rate proportional to  $h^2$ . In other words, the improved Euler's method has a global truncation error that is  $O(h^2)$ . Another method known as the **Runge-Kutta method**, has a global truncation error  $O(h^4)$ . See Exercise 16 to learn more about the Runge Kutta method.

### 6.1.3 Qualitative Analysis of Differential Equations

In this section we examine qualities of a system of differential equations, including equilibria and stability. We want to understand how the initial value as well as model parameters affect the long term behavior of the solution just as we did with our discrete models.

Consider a differential equation of the form  $\frac{dN}{dt} = f(N)$ . We call this an **autonomous** differential equation as  $f$  depends only on  $N$  and not explicitly on  $t$ . Many laws of physics result in autonomous systems, since the physical laws are independent of time. We say that the system is in **equilibrium** if  $\frac{dN}{dt} = 0$ , meaning that no growth is occurring as was the case for discrete dynamical systems. An equilibrium point  $N^*$  is said to be **stable** if the limit  $N(t) \rightarrow N^*$  as  $t \rightarrow \infty$ . This indicates that the growth levels off over a period of time toward this value of  $N^*$ , and the graph of  $N(t)$  vs.  $t$  will have a horizontal asymptote at  $N^*$ . An equilibrium point  $N^*$  is **unstable** if we start with some  $N(t_0)$  that is arbitrarily close to  $N^*$  but as time moves forward, the solution  $N$  does not return back to  $N^*$ .

For an autonomous equation, we can visualize the stability of the system by using a **phase line diagram**. A phase line diagram is the graph of  $\frac{dN}{dt}$  as a function of  $N$ , indicating the stability of each equilibrium. The phase line diagram starts with a plot of  $f(N)$  against  $N$ . The equilibria will be solutions to  $f(N) = 0$ , so these occur at the horizontal intercepts. Arrows on the horizontal axis represent whether  $N$  increases or decreases as we move forward in time. Arrows pointing right indicate that at these values,  $N$  will increase in time. Arrows pointing to the left indicate that for these values,  $N$  will decrease with time. Note that  $N$  increases if the rate of change  $\frac{dN}{dt}$  is positive, and decreases otherwise. Hence we point the arrows on the horizontal axis to the right where  $\frac{dN}{dt} > 0$ , and to the left where where  $\frac{dN}{dt} < 0$ . On the phase line, a stable equilibrium is indicated by the arrows facing each other, whereas an unstable equilibrium is indicated by the arrows pointing away from each other. We use a solid dot to represent a stable equilibrium and an open dot to represent an unstable equilibrium.

Refer to [Exercise A.0.48](#) for a phase line example of the logistic equation.

#### Example 6.1.5

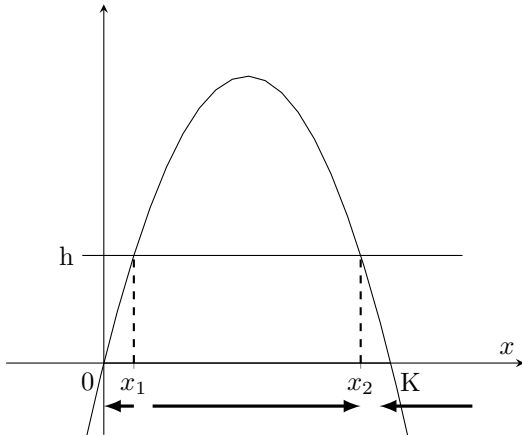
Suppose we want to consider the effects of harvesting on a population of fish  $x(t)$  at any time  $t \geq 0$ , which exhibits logistic growth with carrying capacity  $K$ . Remember the logistic model shows exponential growth for small populations and no growth once the population reaches carrying capacity. For harvesting rate  $H(x)$  (fish per year), use a phase line to determine if the harvesting effort is sustainable over time for  $H(x) = h$ .

.....  
 First consider the differential equation for constant harvesting

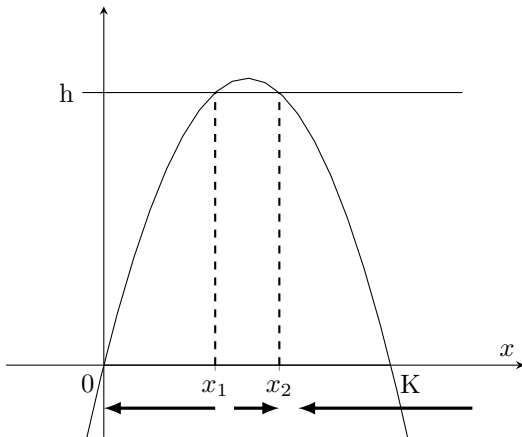
$$\frac{dx}{dt} = rx \left(1 - \frac{x}{K}\right) - h$$

and plot the phase line with positive component  $G(x) = rx \left(1 - \frac{x}{K}\right)$  and the negative component  $H(x) = h$  versus the population  $x$ . The intersection of these two curves gives the equilibrium point where  $\frac{dx}{dt} = 0$ . For  $G > H$ ,  $\frac{dx}{dt} > 0$  and for  $G < H$ ,  $\frac{dx}{dt} < 0$ . This technique also gives us the benefit of using our phase line from the logistic model in [Exercise A.0.48](#) and simply adding the harvesting component. We draw the phase line arrows along the horizontal  $x$  axis. See [20].

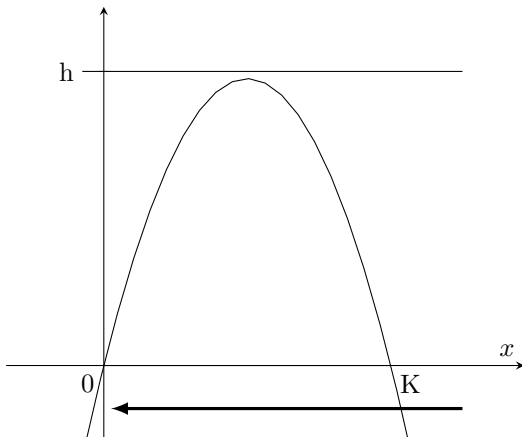
If we begin with a low harvesting rate as in the figure below, we find two points of intersection at  $x_1$  and  $x_2$ . These points move closer together as the harvest rate increases. For  $x = 0$ , we have  $\frac{dx}{dt} = -h < 0$ . However, no harvesting would occur if there are no fish so  $h$  is effectively zero here and  $x^* = 0$  is a stable equilibrium. The fixed point  $x_1$  is unstable resulting in a declining population for low numbers of fish. Harvesting in a large population gives a stable equilibrium  $x_2$  at a point lower than the natural carrying capacity though still sustainable.



As the harvesting rate increases, the two fixed points move closer together decreasing the margin of error between unstable and stable equilibria. At this point, it is prudent to ask what the accuracy is on the growth rate  $r$  and carrying capacity  $K$ . Of course this is unknowable in practice but can be estimated through experimental results (annual surveys, isolated testing, etc.). In addition, even a regulated harvest rate may not reflect the actual harvest in the case of under- or over-fishing. Rather than selecting the highest precise harvest rate assuming perfect accuracy, for most populations harvest regulations should likely be set at a more conservative level.



In the last case, selecting a harvest rate that does not intersect the logistic growth curve,  $\frac{dx}{dt} < 0$  for all  $x$  and the population must decline to zero. Consistent over-fishing will most certainly cause the population to crash.



After considering the example above, we may realize that the practice of harvesting is much easier with large populations when there are more fish to be found. With less fish, more time and resources are needed to find the same numbers of fish and may not be practically possible. To account for this in the model, we may adjust with a linear harvest rate  $H(x) = hx$  which is proportional to the population. In [Activity A.0.49](#) you will analyze this new model with phase lines.

## 6.2 Systems of Differential Equations

In this section, we consider the system of differential equations

$$\begin{aligned}\frac{dx}{dt} &= ax(1 - x/K) - bxy, \\ \frac{dy}{dt} &= cxy - dy,\end{aligned}$$

where  $x(t)$  is the prey population at time  $t$  and  $y(t)$  is the predator population at time  $t$ . As before, the constant  $K$  is the carrying capacity of the prey population. This model describes a predator-prey model with competition within the prey population. This *intra*-specific competition can be seen more explicitly by expanding the first term in the prey equation noting the term  $-\frac{a}{K}x^2$ , which includes the product of a species with itself,  $x^2$ . *Inter*-specific competition is noted by the product of different species as given in the terms  $bxy$  and  $cxy$ .

As a quick reality check, notice that if  $x = K$ , the equation for prey becomes  $x' = -bxy$  and the population of  $x$  begins to decline from predation as we would expect. In the predator equation, if there are no prey  $x = 0$  and  $y' = -dy$ . This results in the decline of predators, as it should if we assume this particular prey to be the only food source.

Notice this system of differential equations has two dependent variables  $x$  and  $y$  that both depend on implicitly on time  $t$ . The rate of change of  $x$  and  $y$  each depend on  $x$  and  $y$  at each point of time. For a system such as this, we typically cannot find analytic solutions for  $x$  and  $y$  in terms of  $t$  except in a few special cases. In this text, we focus on approximating the solution by using a computational algorithm, just as we did in the case of a single equation. We later examine the equilibria and stability of the system.

While most systems such as this cannot be solved explicitly, the process below gives a method for reducing its complexity in such a way that the qualitative analysis is simplified.

### Nondimensionalization

In the predator-prey example above, notice there were 5 parameters:  $a, b, c, d$ , and  $K$ . It turns out that by making some clever substitutions we can reduce the number of parameters to 2. This process is called **nondimensionalization**.

We start by introducing new variables  $X = x/K$ ,  $Y = (b/a)y$ , and  $T = at$ . The choice of these substitutions is indeed ‘clever’ and relies on some intuition and practice. At this point, we would like to write a new system of differential equations entirely in terms of  $X, Y$ , and  $T$ . We use the chain rule extensively.

The chain rule says that  $\frac{dX}{dT} = \frac{dx}{dt} \frac{dX}{dx} \frac{dt}{dT}$ , and we also notice that  $dX/dx = 1/K$  and  $dt/dT = 1/a$ . The first equation then becomes

$$\frac{dX}{dT} = \frac{dx}{dt} \frac{dX}{dx} \frac{dt}{dT} = (ax(1 - x/K) - bxy) \cdot \frac{1}{K} \cdot \frac{1}{a}.$$

Using the facts that  $x = XK$  and  $y = (a/b)Y$  we have

$$\frac{dX}{dT} = \frac{aXK(1 - X) - bXK(a/b)Y}{Ka},$$

which simplifies to

$$\frac{dX}{dT} = X(1 - X) - XY.$$

Similarly, the second equation will be expressed in terms of the new variables as

$$\frac{dY}{dT} = \frac{cK}{a}XY - \frac{d}{a}Y. \quad (6.6)$$

Notice that the parameter  $b$  does not appear in either of these equations, so we have reduced the number of parameters from 5 to 4. However, notice that we only have two terms on the right hand side of (6.6), which implies that we need at most two parameters for this equation. We thus make additional substitutions  $s_1 = cK/a$  and  $s_2 = d/a$ . The system of two equations now has only two parameters,  $s_1$  and  $s_2$  and can be written as

$$\begin{aligned} \frac{dX}{dT} &= X(1 - X) - XY, \\ \frac{dY}{dT} &= s_1XY - s_2Y. \end{aligned}$$

The process of nondimensionalization reduces the number of parameters from 5 to 2. Precisely because the new system of equations is simply a scaled version of the original system, the solutions of the nondimensionalized system will look and behave the same as the solutions to the original system. For example, we set  $X = x/K$  so that  $X$  represents the proportion of the carrying capacity taken up by the prey. This means that if the prey population is less than the carrying capacity, we have  $0 \leq X \leq 1$  instead of  $0 \leq x \leq K$ . See Figure 6.3 for a visual comparison. There are often several ways to nondimensionalize a system of equations. The general idea is to define variables and parameters to make the system as simple as possible.

Now that we have a simplified system of equations, we focus on its numerical solutions and qualitative analysis.

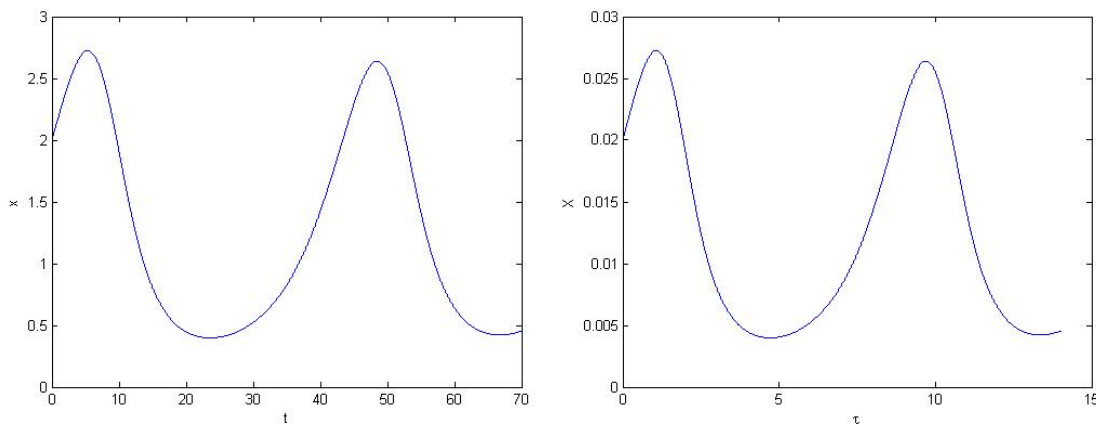


Figure 6.3: Left: Original  $x(t)$ , Right: Nondimensionalized  $X(T)$

## Euler's Method for Systems

Suppose we have a system of  $n$  differential equations with  $n$  dependent variables, each of which depend on time  $t$ .

$$\begin{aligned} x'_1(t) &= f_1(t, x_1, x_2, \dots, x_n), & x_1(t_0) &= \alpha_1 \\ x'_2(t) &= f_2(t, x_1, x_2, \dots, x_n), & x_2(t_0) &= \alpha_2 \\ &\vdots & &\vdots \\ x'_n(t) &= f_n(t, x_1, x_2, \dots, x_n), & x_n(t_0) &= \alpha_n \end{aligned} \quad (6.7)$$

To apply Euler's method for the system, we essentially iterate Euler's method for each variable  $x_1, x_2, \dots, x_n$  on each step.

We can simplify the notation considerably by expressing (6.7) in vector form:

$$\mathbf{x}'(t) = \mathbf{f}(t, \mathbf{x}), \quad \mathbf{x}(t_0) = \mathbf{x}_0,$$

where

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}, \mathbf{x}'(t) = \begin{bmatrix} x_1'(t) \\ \vdots \\ x_n'(t) \end{bmatrix}, \mathbf{f} = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}, \text{ and } \mathbf{x}_0 = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}.$$

We are now denoting  $\mathbf{x}'(t)$  to be the vector of functions containing the derivatives of each of the components of  $\mathbf{x}$ . Euler's method for systems can be expressed as a sequence of vectors,

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\mathbf{f}(t_n, \mathbf{x}_n)$$

with  $\mathbf{x}_0$  as given above. We can also devise an improved Euler's method for systems analogous to (6.5), with the equations

$$\begin{aligned} \mathbf{u}_{n+1} &= \mathbf{x}_n + h\mathbf{f}(t_n, \mathbf{x}_n) \\ \mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{h}{2} (\mathbf{f}(t_n, \mathbf{x}_n) + \mathbf{f}(t_{n+1}, \mathbf{u}_{n+1})), \end{aligned}$$

with  $\mathbf{x}_0$  as given above.

Below are a couple points to keep in mind as you program systems in MATLAB. You may wish to refer back to the MATLAB code from Example 5.2.3 and the pseudocode of Euler's Method Algorithm given earlier in this chapter. The Improved Euler's Method Algorithm will have similar modifications but are not shown here.

In the discrete system of owls and rats given in Example 5.2.3, the MATLAB code updates each population in a new line. You could do this for each of the  $n$  equations in the matrix system above, but it will be much cleaner to treat the matrix equation as a single unit. For  $n$  differential equations in the system and  $k$  time steps, we update a matrix of size  $n \times (k + 1)$  with the approximate solutions to  $\mathbf{x}$ . Initial conditions can be given by the MATLAB code

$$\mathbf{x}(:,1) = [\alpha_1; \alpha_2; \cdots; \alpha_n]$$

which fills in the first column of  $\mathbf{x}$ . In the `for` loop of Euler's method, we update the  $(i + 1)^{\text{st}}$  column of  $\mathbf{x}$  with the  $i^{\text{th}}$  column and ensure the function  $f$  also takes in the  $i^{\text{th}}$  entry of the time vector and the  $i^{\text{th}}$  column of  $\mathbf{x}$ .

$$\mathbf{x}(:,i+1) = \mathbf{x}(:,i) + \mathbf{f}(\mathbf{t}(i), \mathbf{x}(:,i)) h$$

Note that initializing the step size  $h$ , determining the number of steps  $k = (t_f - t_0)/h$ , and defining the time vector of size  $k + 1$  will be equivalent to the single dimension Euler's method.

The default of plotting from a matrix in MATLAB is to plot by row. Thus the following command will plot each row of  $\mathbf{x}$ , first to last, against the time vector  $\mathbf{t}$ .

$$\text{plot}(\mathbf{t}, \mathbf{x})$$

Using a `legend` will help identify the appropriate curve with its corresponding differential equation. To plot trajectories in a 2-dimensional system, use the first and second rows of the matrix  $\mathbf{x}$ .

Refer to [Activity A.0.50](#) for an example of applying Euler's method for systems to solve a predator-prey model.

## 6.3 Qualitative Analysis of Systems of Differential Equations

In Section 6.1.3 we studied qualitative analysis for a single autonomous differential equation. Here we study qualitative properties of a system of autonomous differential equations.

A system of autonomous differential equations takes on the form  $\mathbf{x}'(t) = f(\mathbf{x})$  or

$$\begin{aligned} x_1'(t) &= f_1(x_1, x_2, \dots, x_n), & x_1(t_0) &= \alpha_1 \\ x_2'(t) &= f_2(x_1, x_2, \dots, x_n), & x_2(t_0) &= \alpha_2 \\ &\vdots & &\vdots \\ x_n'(t) &= f_n(x_1, x_2, \dots, x_n), & x_n(t_0) &= \alpha_n. \end{aligned} \tag{6.8}$$

We say that an **equilibrium point**, or **critical point**, of a linear system is a point  $(x_1^*, x_2^*, \dots, x_n^*)$  in which  $f_i(x_1^*, x_2^*, \dots, x_n^*) = 0$  for each  $i = 1, 2, \dots, n$ . We may interpret this by saying that at equilibrium, the values of  $x_1, x_2, \dots, x_n$  are not changing, since their derivatives are all zero. A critical point  $(x_1^*, x_2^*, \dots, x_n^*)$  is **stable** if each  $x_i$  approaches  $x_i^*$  as  $t \rightarrow \infty$ , and the critical point is **unstable** otherwise. It turns out that if the solution  $\mathbf{y} = (x_1, x_2, \dots, x_n)^T$  approaches a finite limit  $\mathbf{y}^* = (x_1^*, x_2^*, \dots, x_n^*)^T$  as  $n \rightarrow \infty$ , then  $\mathbf{y}^*$  is a stable equilibrium of the solution.

We can analyze a system of two autonomous differential equations by plotting a **phase plane diagram**, the two-dimensional analogue of the phase line plot discussed in Section 6.1.3. The basic two equation system is given by

$$\begin{aligned} x'(t) &= f_1(x, y), & x(t_0) &= x_0 \\ y'(t) &= f_2(x, y), & y(t_0) &= y_0. \end{aligned}$$

A phase plane diagram is a two-dimensional picture of how  $x$  and  $y$  change with time with respect to different sectors of the graph. It usually only shows the first quadrant (unless  $x$  and  $y$  are allowed to be negative), and includes nullclines, equilibria, and several arrows that show the direction of growth.

A **nullcline** of the system is a curve on the phase plane for which either  $x' = 0$  or  $y' = 0$ . For a point to be an equilibrium, we need both  $x'$  and  $y'$  equal to zero, so these occur where an  $x' = 0$  nullcline crosses a  $y' = 0$  nullcline. We may also indicate the direction of growth across the nullclines with arrows. For a nullcline of the form  $x' = 0$ , we know that there is no growth in the  $x$  direction, however  $y$  could be increasing or decreasing. We indicate this with vertical arrows. Similarly, we indicate growth across nullclines of the form  $y' = 0$  with horizontal arrows. A phase plane diagram could also include arrows to indicate the direction of growth between the nullclines. One of the goals of the phase plane diagram is to give a sense of how  $x$  and  $y$  evolve with time.

### Example 6.3.1

Consider the system

$$\begin{aligned} \frac{dx}{dt} &= x(y - 1) \\ \frac{dy}{dt} &= y(2 - x^2 - y). \end{aligned}$$

Assuming that  $x$  and  $y$  represents densities or quantities, we will only be concerned with regions for which  $x > 0$  and  $y > 0$ .

**Find the direction of growth across the  $x' = 0$  nullclines:** To find the  $x' = 0$  nullclines, we set  $x' = 0$  to get  $0 = x(y - 1)$ . This gives two  $x' = 0$  nullclines:  $x = 0$  and  $y = 1$ . These are plotted along the  $xy$  plane. We know that  $x$  is constant along these lines (since  $x' = 0$ ), but we want to know where  $y$  is increasing or decreasing across each of these. Along the  $x = 0$  nullcline,  $y' = y(2 - 0^2 - y) = y(2 - y)$  which is positive when  $0 < y < 2$  and negative if  $y > 2$  (or if  $y < 0$  but we are only concerned with  $x, y > 0$ ). We indicate this on the graph with upward arrows on the segment between 0 and 2, and downward arrows

where  $y > 2$ , see Figure 6.4. Keep in mind that the arrows only go vertically and not horizontally since  $x$  is constant along a  $x' = 0$  nullcline. Along the  $y = 1$  nullcline,  $y' = 1(2 - x^2 - 1) = 1 - x^2 = (1 - x)(1 + x)$ , which is positive for  $-1 < x < 1$  and negative if  $x > 1$  or  $x < -1$ . Again we use vertical arrows to indicate the direction of growth since this was a  $x' = 0$  nullcline.

**Find the direction of growth across the  $y' = 0$  nullclines:** To find the  $y' = 0$  nullclines, solve  $0 = y(2 - x^2 - y)$ , which gives us  $y = 0$  and  $y = 2 - x^2$ . These are plotted along the  $xy$  plane - but consider using colored or dashed lines to distinguish the  $x' = 0$  nullclines from the  $y' = 0$  nullclines. There is no growth in the  $y$  direction across these  $y' = 0$  lines so we have only horizontal directional changes given by horizontal arrows. Thus we want to know where  $x$  is increasing or decreasing along each of these. Starting with  $y = 0$ , notice that  $x' = x(-1) = -x$ , which is negative when  $x > 0$ . Along  $y = 2 - x^2$ , we have  $x' = x(2 - x^2 - 1) = x(1 - x^2) = x(1 - x)(1 + x)$ , which is positive when  $0 < x < 1$  and negative when  $x > 1$ .

**Find the equilibria** The equilibria are points  $(x, y)$  that satisfy both  $x' = 0$  and  $y' = 0$ . These are represented in the plot by points in which an  $x' = 0$  nullcline intersects a  $y' = 0$  nullcline. These are points  $(0, 0)$ ,  $(0, 2)$ , and  $(1, 1)$ . Notice that  $(0, 1)$  is not an equilibrium because  $y' \neq 0$  there, and  $(\sqrt{2}, 0)$  is not an equilibrium since  $x' \neq 0$ .

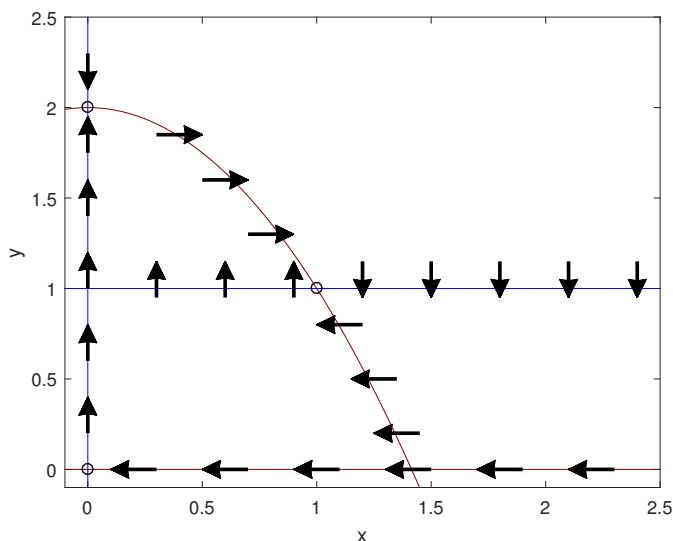


Figure 6.4: Phase plane diagram for Example 6.3.1. Blue lines indicate  $x' = 0$  nullclines, red lines are  $y' = 0$  nullclines. Equilibria occur where a red line crosses a blue line:  $(0,0)$ ,  $(0,2)$ , and  $(1, 1)$

---

Refer to [Activity A.0.51](#) for another example of a making and applying a phase plane diagram.

## 6.4 Linear systems

In this section, we consider the special case where each differential equation in the system is autonomous and linear with respect to the dependent variables. A **linear system** of  $n$  first order differential equations



with  $n$  dependent variables takes on the form

$$\begin{aligned} x'_1(t) &= a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n + b_1, & x_1(0) &= \alpha_1 \\ x'_2(t) &= a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n + b_2, & x_2(0) &= \alpha_2 \\ &\vdots & &\vdots \\ x'_n(t) &= a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n + b_n, & x_n(0) &= \alpha_n, \end{aligned}$$

where  $a_{ij}, b_i \in \mathbb{R}$  for all  $i, j \in \{1, 2, \dots, n\}$ . We may express the system in the vector matrix form

$$\mathbf{x}'(t) = A\mathbf{x}(t) + \mathbf{b}, \quad (6.9)$$

where

$$\mathbf{x}'(t) = \begin{bmatrix} x'_1(t) \\ \vdots \\ x'_n(t) \end{bmatrix}, \quad A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \vdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}, \quad \mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}.$$

Notice that many of our previous examples have taken the form  $\mathbf{x}'(t) = A\mathbf{x}$ , which is also a linear system but with  $\mathbf{b} = \mathbf{0}$ . All the theory in this section applies, but is made simpler with this caveat. See Example 6.4.4 for an example with  $\mathbf{b} \neq \mathbf{0}$ .

#### Example 6.4.1

Consider the “decoupled” system

$$x'(t) = 3x(t) \quad (6.10)$$

$$y'(t) = -5y(t). \quad (6.11)$$

We say that the system is decoupled since the equation involving  $x$  does not depend on  $y$ , and the equation involving  $y$  does not depend on  $x$ . We can therefore solve each one independently to get  $x(t) = c_1e^{3t}$ ,  $y(t) = c_2e^{-5t}$ , where  $c_1, c_2$  are constants that depend on initial conditions. Notice that if we write the system in matrix notation, we get

$$\frac{d}{dt} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & -5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

The solution may be written using vector notation:

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = c_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} e^{3t} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} e^{-5t}.$$

Since the system is decoupled, its corresponding matrix  $A = \begin{bmatrix} 3 & 0 \\ 0 & -5 \end{bmatrix}$  is diagonal. Recalling Example 4.2.4, we see that the eigenvalues are the entries on the diagonal, 3 and  $-5$ . It turns out that we can use the eigenvalues and eigenvectors to determine the nature of the solutions to *any* linear system of differential equations.

In general, for arbitrary linear differential equations with  $n$  equations and  $n$  variables, we will seek solutions of the form  $\mathbf{v}e^{\lambda t}$  where  $\lambda$  is an eigenvalue of the system and  $\mathbf{v}$  is the corresponding eigenvector. This textbook gives a brief overview of the theory behind solving these linear systems - for proofs or additional detail refer to [28].

The following theorem gives the solution to a linear system in the case that there are distinct real eigenvalues.

**Theorem 6.4.2.** If the matrix  $A$  has  $n$  real, distinct eigenvalues  $\lambda_1, \lambda_2, \dots, \lambda_n$  and associated eigenvectors  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ , then the general solution to  $\mathbf{x}'(t) = A\mathbf{x}(t) + \mathbf{b}$  is

$$\mathbf{x}(t) = \mathbf{x}^* + c_1 e^{\lambda_1 t} \mathbf{v}_1 + \dots + c_n e^{\lambda_n t} \mathbf{v}_n,$$

where  $c_1, c_2, \dots, c_n \in \mathbb{R}$  and  $\mathbf{x}^*$  is the equilibrium and solution to  $A\mathbf{x}^* + \mathbf{b} = \mathbf{0}$ .

### Example 6.4.3

Find the solution to the initial value problem

$$\mathbf{x}'(t) = \begin{bmatrix} 6 & 4 & 4 \\ -7 & -2 & -1 \\ 7 & 4 & 3 \end{bmatrix} \mathbf{x}(t), \quad \mathbf{x}(0) = \begin{bmatrix} 3 \\ -6 \\ 4 \end{bmatrix}.$$

.....  
We find the general solution as in Theorem 6.4.2 then apply the initial conditions to determine  $c_1, c_2, c_3$ .

First, find the eigenvectors and eigenvalues of the matrix:

$$A = \begin{bmatrix} 6 & 4 & 4 \\ -7 & -2 & -1 \\ 7 & 4 & 3 \end{bmatrix}$$

$$[x \ y] = \text{eig}(A)$$

There are 3 eigenvalues:  $\lambda_1 = 6, \lambda_2 = -1, \lambda_3 = 2$  and the eigenvectors are  $\mathbf{v}_1 = (1, -1, 1)^T, \mathbf{v}_2 = (0, -1, 1)^T, \mathbf{v}_3 = (-1, 2, -1)^T$ . Since the eigenvalues are real and distinct we may use Theorem 6.4.2 to obtain the general solution

$$\mathbf{x}(t) = \mathbf{x}^* + c_1 e^{6t} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + c_2 e^{-t} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} + c_3 e^{2t} \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix},$$

where  $\mathbf{x}^*$  solves  $A\mathbf{x}^* = \mathbf{0}$ . We see that  $\mathbf{x}^* = \mathbf{0}$  (either by calculating its value or using theory from linear algebra which says that if the eigenvalues of  $A$  are unique then  $A\mathbf{x} = \mathbf{0}$  has only the trivial solution  $\mathbf{x} = \mathbf{0}$ ). Next, we enforce the initial condition to get

$$\mathbf{x}(0) = \begin{bmatrix} 3 \\ -6 \\ 4 \end{bmatrix} = c_1 \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + c_2 \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} + c_3 \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ -1 & -1 & 2 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}.$$

Solving the system:

$$\text{rref}([1 \ 0 \ -1 \ 3; -1 \ -1 \ 2 \ -6; 1 \ 1 \ -1 \ 4])$$

we see that  $c_1 = 1, c_2 = 1, c_3 = -2$ . This gives us the solution

$$\mathbf{x}(t) = e^{6t} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + e^{-t} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} - 2e^{2t} \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}.$$

Letting  $\mathbf{x} = (x_1, x_2, x_3)^T$  we can also write the solution in terms of its individual components,

$$x_1(t) = e^{6t} + 2e^{2t}, \quad x_2(t) = -e^{6t} - e^{-t} - 4e^{2t}, \quad x_3(t) = e^{6t} + e^{-t} + 2e^{2t}.$$

We now consider the same example, but with  $\mathbf{b} \neq \mathbf{0}$ .

**Example 6.4.4**

Find the solution to the initial value problem

$$\mathbf{x}'(t) = \begin{bmatrix} 6 & 4 & 4 \\ -7 & -2 & -1 \\ 7 & 4 & 3 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \quad \mathbf{x}(0) = \begin{bmatrix} 3 \\ -6 \\ 4 \end{bmatrix}.$$

.....  
As before, we find the general solution then apply the initial conditions.

From the example above, the coefficient matrix  $A$  has eigenvalues  $\lambda_1 = 6, \lambda_2 = -1, \lambda_3 = 2$  and eigenvectors  $\mathbf{v}_1 = (1, -1, 1)^T, \mathbf{v}_2 = (0, -1, 1)^T, \mathbf{v}_3 = (-1, 2, -1)^T$ . Since the eigenvalues are real and distinct, Theorem 6.4.2 applies and gives the general solution

$$\mathbf{x}(t) = \mathbf{x}^* + c_1 e^{6t} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + c_2 e^{-t} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} + c_3 e^{2t} \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix},$$

where  $\mathbf{x}^*$  solves the equation  $A\mathbf{x}^* + \mathbf{b} = \mathbf{0}$ . Since  $\mathbf{b} \neq \mathbf{0}$  as in the previous example,  $\mathbf{x}^* \neq \mathbf{0}$ . Solving

$$A\mathbf{x}^* = -\mathbf{b} \text{ or } \mathbf{x}^* = \begin{bmatrix} 6 & 4 & 4 \\ -7 & -2 & -1 \\ 7 & 4 & 3 \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ -2 \\ -3 \end{bmatrix} \text{ gives the solution } \mathbf{x}^* = \begin{bmatrix} 1.5 \\ -6 \\ 3.5 \end{bmatrix}.$$

Adding this result to the general solution gives

$$\mathbf{x}(t) = \begin{bmatrix} 1.5 \\ -6 \\ 3.5 \end{bmatrix} + c_1 e^{6t} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + c_2 e^{-t} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} + c_3 e^{2t} \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}.$$

Imposing the initial condition, we have

$$\mathbf{x}(0) = \begin{bmatrix} 3 \\ -6 \\ 4 \end{bmatrix} = \begin{bmatrix} 1.5 \\ -6 \\ 3.5 \end{bmatrix} + c_1 \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + c_2 \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} + c_3 \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}.$$

Subtracting the constant vector  $\mathbf{x}^*$  from  $\mathbf{x}_0$  gives the system

$$\begin{bmatrix} 1 & 0 & -1 \\ -1 & -1 & 2 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 0 \\ 0.5 \end{bmatrix},$$

with solution  $c_1 = 2, c_2 = -1, c_3 = 0.5$ . Finally, we have our solution to the linear system

$$\mathbf{x}(t) = \begin{bmatrix} 1.5 \\ -6 \\ 3.5 \end{bmatrix} + 2e^{6t} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} - e^{-t} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} + \frac{1}{2}e^{2t} \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}.$$

Writing  $\mathbf{x} = (x_1, x_2, x_3)^T$  in its individual components gives

$$x_1(t) = 1.5 + 2e^{6t} - \frac{1}{2}e^{2t}, \quad x_2(t) = -6 - 2e^{6t} + e^{-t} + e^{2t}, \quad x_3(t) = 3.5 + 2e^{6t} - e^{-t} - \frac{1}{2}e^{2t}.$$

Note that this solution differs from the previous example of  $\mathbf{x}^* = A\mathbf{x}$  in the values of  $c_1, c_2, c_3$  and the equilibrium vector  $\mathbf{x}^*$ . The eigenvalue and eigenvector components are the same.

Recall that when a real-valued matrix has complex eigenvalues, they occur in complex conjugate pairs (i.e. have the form  $a \pm bi$ ). In this case, we may apply the following theorem.

**Theorem 6.4.5.** *If the square matrix  $A$  has complex eigenvalues  $r = \alpha \pm i\beta$  with corresponding eigenvectors  $\mathbf{v} = \mathbf{a} \pm i\mathbf{b}$ , then one real solution to  $\mathbf{x}'(t) = A\mathbf{x}$  is*

$$\mathbf{x}(t) = c_1(e^{\alpha t} \cos(\beta t)\mathbf{a} - e^{\alpha t} \sin(\beta t)\mathbf{b}) + c_2(e^{\alpha t} \sin(\beta t)\mathbf{a} + e^{\alpha t} \cos(\beta t)\mathbf{b}), \quad (6.12)$$

where  $c_1, c_2 \in \mathbb{R}$ .

If  $A$  is a  $2 \times 2$  matrix with complex eigenvalues, then (6.12) describes all of the solutions to  $\mathbf{x}'(t) = A\mathbf{x}$  and to solve  $\mathbf{x}'(t) = A\mathbf{x}(t) + \mathbf{b}$  we add  $\mathbf{x}^*$  to the solution given in (6.12) where  $\mathbf{x}^*$  solves  $A\mathbf{x}^* + \mathbf{b} = \mathbf{0}$  as was the case in Theorem 6.4.2. You may refer to just about any textbook in differential equations for an explanation of how to piece together solutions for the case in which there are a mix of real and complex eigenvalues, or repeat eigenvalues.

#### Example 6.4.6

Find a general solution of the system

$$\mathbf{x}'(t) = \begin{bmatrix} 3.4 & -9.2 & -14.2 \\ 0.6 & -2.8 & -3.4 \\ 0.6 & -0.8 & -1.8 \end{bmatrix} \mathbf{x}(t).$$

The eigenvalues are  $\lambda_1 = -0.8$ ,  $\lambda_2 = -0.2 + 0.6i$ ,  $\lambda_3 = -0.2 - 0.6i$ , and the eigenvectors are  $\mathbf{v}_1 = (1, 2, -1)^T$ ,  $\mathbf{v}_2 = (5, -1, 2)^T + i(2, 2, -1)^T$ ,  $\mathbf{v}_3 = (5, -1, 2)^T - i(2, 2, -1)^T$ . From Theorem 6.4.2, the general solution is

$$\mathbf{x}(t) = c_1 e^{-0.8t} \mathbf{v}_1 + c_2 e^{(-0.2+0.6i)t} \mathbf{v}_2 + c_3 e^{(-0.2-0.6i)t} \mathbf{v}_3.$$

Using Theorem 6.4.5 with  $\mathbf{a} = (5, -1, 2)^T$  and  $\mathbf{b} = (2, 2, -1)^T$ , the second and third terms can then be written in terms of real-valued sine and cosine functions, giving

$$\mathbf{x}(t) = c_1 e^{-0.8t} \mathbf{v}_1 + c_2 e^{-0.2t} (\cos(0.6t)\mathbf{a} - \sin(0.6t)\mathbf{b}) + c_3 e^{-0.2t} (\sin(0.6t)\mathbf{a} + \cos(0.6t)\mathbf{b}).$$

Given a general solution for linear systems that includes an equilibrium  $\mathbf{x}^*$ , we now consider the conditions of stability. Recall that when we studied discrete dynamical systems of the form  $\mathbf{x}'(t) = A\mathbf{x}$  in Chapter 4, we needed all the eigenvalues to satisfy  $|\lambda| < 1$  in order for the trajectories to approach a stable equilibrium at  $(0, 0)$ . For linear systems of differential equations, we have different criteria for stability.

First imagine that all of the eigenvalues are distinct and nonnegative. From Theorem 6.4.2, the solution takes on the form  $\mathbf{x}(t) = \mathbf{x}^* + c_1 e^{\lambda_1 t} \mathbf{v}_1 + \dots + c_n e^{\lambda_n t} \mathbf{v}_n$ . If all of the eigenvalues are negative then certainly  $\lim_{t \rightarrow \infty} \mathbf{x}(t) = \mathbf{x}^*$ . On the other hand, if even one of the eigenvalues is positive, then the solution will tend towards  $\pm\infty$ .

Now suppose that there are complex eigenvalues which would lead us to take a look at Theorem 6.4.5. If the real part of the eigenvalue,  $\alpha$ , is negative, then the solutions must tend towards zero. If the real part is positive, then the solutions do not converge. We have not discussed what happens in the case that there are a mix of complex and real eigenvalues, or the case that there are repeated eigenvalues, but it turns out that we can always look at the real part of the eigenvalues to determine stability. The following theorem states that we need all of the eigenvalues to have a negative real part in order to ensure stability of the equilibrium solution.

**Theorem 6.4.7.** *The equilibrium solution  $\mathbf{x} = \mathbf{x}^*$  for the equation  $\mathbf{x}'(t) = A\mathbf{x}(t) + \mathbf{b}$ , where  $\det(A) \neq 0$  and  $A\mathbf{x}^* + \mathbf{b} = \mathbf{0}$ , is asymptotically stable if and only if all of the real eigenvalues of  $A$  are negative and all of the complex eigenvalues of  $A$  have a negative real part.*

See [Exercise A.0.52](#) for an example of applying Theorem 6.4.2 and Theorem 6.4.7.

## 6.5 Nonlinear Systems of Equations

For nonlinear systems, we would like to use the techniques from Section 6.4 to determine the stability of equilibria. First, suppose we have a system of two autonomous differential equations

$$\begin{aligned}\frac{dx}{dt} &= f(x, y) \\ \frac{dy}{dt} &= g(x, y),\end{aligned}$$

where  $f$  and  $g$  are not necessarily linear functions of  $x$  and  $y$ , such as in the competition terms of the predator-prey equations.

Let  $(x^*, y^*)$  denote an equilibrium of the system, meaning that  $f(x^*, y^*) = 0$  and  $g(x^*, y^*) = 0$ . Recall from multivariable calculus the formula for the linear approximation of  $f(x, y)$  about  $(x^*, y^*)$ :

$$f(x, y) \approx f(x^*, y^*) + \frac{\partial f}{\partial x}(x^*, y^*)(x - x^*) + \frac{\partial f}{\partial y}(x^*, y^*)(y - y^*).$$

Similarly,

$$g(x, y) \approx g(x^*, y^*) + \frac{\partial g}{\partial x}(x^*, y^*)(x - x^*) + \frac{\partial g}{\partial y}(x^*, y^*)(y - y^*).$$

Since  $f(x^*, y^*) = g(x^*, y^*) = 0$ , it follows that

$$\begin{aligned}\frac{dx}{dt} = f(x, y) &\approx \frac{\partial f}{\partial x}(x^*, y^*)(x - x^*) + \frac{\partial f}{\partial y}(x^*, y^*)(y - y^*), \\ \frac{dy}{dt} = g(x, y) &\approx \frac{\partial g}{\partial x}(x^*, y^*)(x - x^*) + \frac{\partial g}{\partial y}(x^*, y^*)(y - y^*).\end{aligned}$$

This can be written in matrix form,

$$\begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix} \approx \begin{bmatrix} \frac{\partial f}{\partial x}(x^*, y^*) & \frac{\partial f}{\partial y}(x^*, y^*) \\ \frac{\partial g}{\partial x}(x^*, y^*) & \frac{\partial g}{\partial y}(x^*, y^*) \end{bmatrix} \begin{bmatrix} x - x^* \\ y - y^* \end{bmatrix}$$

or in the more compact form,

$$\frac{d\mathbf{z}}{dt} \approx J(\mathbf{z}^*)(\mathbf{z} - \mathbf{z}^*),$$

where  $\mathbf{z}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$ ,  $\mathbf{z}^* = \begin{bmatrix} x^* \\ y^* \end{bmatrix}$ , and  $J(x, y) = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix}$  is the **Jacobian matrix** corresponding to the linear system. We can also expand the equation to obtain

$$\frac{d\mathbf{z}}{dt} \approx J(\mathbf{z}^*)\mathbf{z} - J(\mathbf{z}^*)\mathbf{z}^*,$$

and we may think of  $J(\mathbf{z}^*)\mathbf{z}^*$  as a vector of constants. Thus, the system approximately takes on the form

$$\frac{d\mathbf{z}}{dt} = A\mathbf{z} + \mathbf{b}.$$

The system is approximately linear near an equilibrium and we may therefore use Theorem 6.4.7, where  $J(\mathbf{z}^*)$  is used in place of the matrix  $A$ . This technique can be used for autonomous systems with more than two variables as well.

**Theorem 6.5.1.** *Let  $(x^*, y^*)$  be an equilibrium point for the system*

$$\frac{dx}{dt} = f(x, y), \quad \frac{dy}{dt} = g(x, y).$$

*Then the equilibrium is stable if and only if all real eigenvalues of  $J(x^*, y^*)$  are negative and all complex eigenvalues of  $J(x^*, y^*)$  have a negative real part.*

**Example 6.5.2**

Recall the system from Example 6.3.1,

$$\frac{dx}{dt} = f(x, y) = x(y - 1) \quad \frac{dy}{dt} = g(x, y) = y(2 - x^2 - y).$$

We found three equilibria points in this example,  $(0, 0)$ ,  $(0, 2)$ , and  $(1, 1)$ . Because the system is nonlinear we need Theorem 6.5.1 to determine the stability of each. The Jacobian is given here

$$J(x, y) = \begin{bmatrix} \frac{df}{dx} & \frac{df}{dy} \\ \frac{dg}{dx} & \frac{dg}{dy} \end{bmatrix} = \begin{bmatrix} y - 1 & x \\ -4xy & 2 - 2x^2 - 2y \end{bmatrix}.$$

We will need to evaluate the Jacobian at each equilibrium and then find the eigenvalues of the resulting matrix  $J(\mathbf{z}^*)$ . We use MATLAB to facilitate this process.

```
x=0;
y=0;
J=[y-1 x; -4*x*y 2-2*x^2-2*y]
eig(J)

x=0;
y=2;
J=[y-1 x; -4*x*y 2-2*x^2-2*y]
eig(J)

x=1;
y=1;
J=[y-1 x; -4*x*y 2-2*x^2-2*y]
eig(J)
```

We see that the eigenvalues at  $(0,0)$  are  $-1$  and  $2$ , eigenvalues at  $(0,2)$  are  $-2$  and  $1$ , and eigenvalues at  $(1,1)$  are  $-1 \pm 1.7321i$ . By Theorem 6.5.1,  $(1,1)$  is a stable equilibrium and  $(0,0)$  and  $(0,2)$  are unstable.

We could verify this by running Euler's method on the linearized system and observing the approximate solutions near each equilibria.

To see that  $(0,0)$  is unstable, we could use an initial point  $(x_0, y_0)$  that is very close to  $(0,0)$ . For example if we use  $x(0) = 0.1$  and  $y(0) = 0.1$ , you should see that the solution tends towards the stable equilibrium of  $(1,1)$ . The same holds for starting values near  $(0,2)$ . It seems that solutions tend towards  $(1,1)$  for any initial condition. There are exceptions - for example if  $x = 0$  then the phase plane diagram (see Figure 6.4) suggests that points will tend towards  $(0,2)$ . Try this with  $x(0) = 0, y(0) = 0.5$  or with  $x(0) = 0, y(0) = 3$ . If this system represented two interacting species, the line where  $x = 0$  would represent the absence of one of species  $x$ , and  $2$  would be the carrying capacity of species  $y$ .

See **Activity A.0.53** for practice using Theorem 6.5.1 to analyze the stability of equilibria.

## 6.6 Exercises

1. In this exercise we will vary the Malthusian model (6.1) to try and fit the population data in Table A.0.36 for all years 1790 - 2010. We consider a variable growth rate and suppose  $r$  depends linearly with time to give the model

$$N'(t) = (at + b)N(t), \quad N(0) = N_0,$$

where  $r = at + b$  and  $a \neq 0, b \neq 0$ .

- (a) Solve the differential equation for  $N(t)$ . (Hint: use separation of variables).
- (b) From your answer in part (a), take the  $\ln$  of both sides. You should get an equation for  $\ln(N)$  that is quadratic with respect to time  $t$  including coefficients involving parameters  $a$  and  $b$ . We will use the population data given in Table A.0.36 to determine the values of these parameters. First update your Excel file from [Activity A.0.36](#) to include a column for  $\ln(N)$ , then plot  $\ln(N)$  against the year  $t$  from 1790 - 2010. You should observe a relationship that looks quadratic and not linear. Use the built-in trend-line tool to determine a quadratic regression line and estimate the parameters  $a$  and  $b$  that best fit the population data. Use these values to update your solution for  $N(t)$  from part (a). Finally, plot the population data and the predictions from  $N(t)$  on the same set of axes for years 1790 - 2010. Describe how the data fits the model.
2. Derive a solution to the logistic model (6.2).
3. Use the pseudocode in Section 6.1.1 to write a MATLAB code segment to implement Euler's method to solve an initial value problem of the form  $x'(t) = f(t, x)$  with  $x(t_0) = x_0$  and plot its solution against time. Your code should be in the form of a function with parameters  $h, t_0, t_f, x(t_0)$ , where  $h$  is the step size,  $t_0$  is the initial time value,  $t_f$  is the final time, and  $x(t_0)$  is the initial value of  $x$  at time  $t_0$ . Your code should include a function that computes  $f$ . Unless otherwise specified, you may use this code for future exercises.
4. Use the pseudocode in Section 6.1.2 to write a MATLAB code segment to implement the improved Euler's method to solve an initial value problem of the form  $x'(t) = f(t, x)$  with  $x(t_0) = x_0$  and plot its solution against time. Your code should be in the form of a function with parameters  $h, t_0, t_f, x(t_0)$ , where  $h$  is the step size,  $t_0$  is the initial time value,  $t_f$  is the final time, and  $x(t_0)$  is the initial value of  $x$  at time  $t_0$ . Your code should include a function that computes  $f$ . Unless otherwise specified, you may use this code for future exercises.
5. Consider the initial value problem  $y'(t) = -2.3y$ ,  $y(0) = 1$ .
- (a) Approximate the solution on the interval  $0 \leq t \leq 4$  using Euler's method with a step size of  $h = 1$ . Describe what is happening - how does the result differ from the true solution?
- (b) Approximate the solution on the interval  $0 \leq t \leq 4$  using the improved Euler method. Does this fix the problem described in part (a)?
- (c) Do your results change for parts (a) and/or (b) with a smaller step size  $h$ ?  
*For additional practice, you may wish to perform Euler's method and improved Euler's method by hand for  $h = 1$  as well as with MATLAB.*
6. Consider the initial value problem  $dy/dt = t^2y + y \cos(t)$ ,  $y(1) = 2$ .

- (a) Solve the initial value problem by hand, then find a decimal approximation for  $y(3)$  [Hint: you should get  $y(3) \approx 5764.43758$ ].
- (b) Use your MATLAB codes to complete the table, using at least 4 significant digits.

$h$	Actual $y(3)$	Euler method approx. $y(3)$	Error Euler's method	Improved Euler's method approx. $y(3)$	Error improved Euler's method
0.5					
0.01					
0.002					
0.0001					
0.00001					

7. Consider the initial value problem  $x'(t) = 2t(x - t) + x/t$ ,  $x(1) = e + 1$ .
- (a) Verify that  $x(t) = te^{t^2} + t$  solves the initial value problem.

- (b) Use Euler's method and the improved Euler's method with  $h = 0.1$  to approximate the solution on the interval  $[1, 2]$ . Compute the two errors when  $t = 2$ . Plot the two approximate solutions, along with the exact solution on the same set of axes.
- (c) Use Euler's method and the improved Euler's method with  $h = 0.01$  to approximate the solution on the interval  $[1, 2]$ . Compute the two errors when  $t = 2$ . Plot the two approximate solutions, along with the exact solution on the same set of axes.
8. Consider the initial value problem  $dy/dx = \sin(y^2) + x$ ,  $y(0) = 1$ .
- (a) Use time step  $h = 0.5$  to approximate  $y(2)$  using both Euler's method and improved Euler's method.
- (b) Repeat part (a) with  $h = 0.01$ .
- (c) Plot the solution on the interval  $[0, 2]$  you get from using the improved Euler's method with  $h = 0.01$ .
9. Consider the initial value problem  $y'(t) = 2ty^2$ ,  $y(2) = 1$ .
- (a) Solve the initial value problem using separation of variables.
- (b) Use time step  $h = 0.1$  to approximate  $y(3)$  using Euler's method.
- (c) Compare the values obtained from Euler's method to the analytic solution when  $t = 2.5, 3$ . What appears to go wrong with Euler's method, and why? Explain.
10. The solution to the initial value problem  $y' = 2/x^4 - y^2$ ,  $y(1) = -0.414$  crosses the  $x$ -axis at some point in the interval  $[1, 2]$ . By experimenting with your MATLAB solver (Euler's method or improved Euler's method), determine this point to three decimal places. Note: you could automate the process of searching for an  $x$ -intercept by using a `while` loop instead of a `for` loop, or by putting an `if` statement inside the `for` loop. You want to stop searching when  $y_{i+1}$  and  $y_i$  have different signs, which means that their product will be negative. This should remind you of the bisection method from Section 3.3.
11. Use Euler's method to estimate the root in  $[0, 1.4]$  for the solution of  $y' = (x + y + 2)^2$ ,  $y(0) = -2$ .
12. Use Euler's method to approximate the maximum value of  $y$  over  $[0, 2]$ , where  $y$  solves  $y' = \sin(x + y)$ ,  $y(0) = 2$ . Give your estimate of the maximum value as well as where this maximum occurs.
13. The solution to  $y' + y/x = x^3y^2$ ,  $y(1) = 3$  has a vertical asymptote at some point in the interval  $[1, 2]$ . By experimenting with your Euler method code or improved Euler method code, determine this point to six decimal places. Hint: you could automate the process of searching for a vertical asymptote by adding a conditional statement to check whether the  $|y_i|$  exceeds some arbitrarily large number.
14. Recall from calculus: If a function  $f$  is continuous on  $[a, b]$ , then it has an antiderivative  $F$  such that  $F'(x) = f(x)$  for all  $x \in [a, b]$ . By the fundamental theorem of calculus,  $F(x) = \int_a^x f(t) dt$ . So to solve the definite integral  $\int_a^x f(t) dt$  we can instead solve the initial value problem  $F'(x) = f(x)$ ,  $F(a) = 0$ .
- (a) Apply Euler's method (with step size  $h$ ) to this initial value problem to derive the approximation formula
- $$\int_a^b f(x) dx \approx \sum_{i=0}^{n-1} f(a + ih)h,$$
- where  $n$  is the number of steps (so that  $h = (b - a)/n$  and  $x_i = a + ih$ ). This is the left-hand sum rule that you may have learned about in calculus.
- (b) Apply the technique above with  $h = 0.1$  to approximate the value of  $\int_{-2}^2 \frac{1}{\sqrt{2\pi}} \exp(-x^2/2) dx$ . Include a plot of  $y(x) = \int_{-2}^x \frac{1}{\sqrt{2\pi}} \exp(-t^2/2) dt$  on the interval  $-2 \leq x \leq 2$ .



- (c) You may recall from statistics that the integral  $\int_{-2}^2 \frac{1}{\sqrt{2\pi}} \exp(-x^2/2) dx$  computes the probability  $P(a < X < b)$  for a normal random variable  $X \sim N(\mu = 0, \sigma = 1)$  with mean  $\mu$  and standard deviation  $\sigma$ . The Empirical Rule states that the probability a normally distributed random variable lies within 2 standard deviations of the mean is approximately 0.95. Does this match with your results from part (b)? You may wish to experiment with different values for  $n$ .
- (d) In general, the density function for a normally distributed random variable  $X$  with mean  $\mu$  and standard deviation  $\sigma$  is

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

Use the technique above, but with Improved Euler's method to devise a MATLAB function with four arguments:  $\mu$ ,  $\sigma$ ,  $a$ , and  $b$ , to compute  $P(a < X < b)$  for any normal random variable  $X$  with mean  $\mu$  and standard deviation  $\sigma$ .

15. Suppose you are given that a family of methods to solve  $x' = f(x, t)$ ,  $x(0) = x_0$  with global truncation errors  $O(h^2)$  takes on the following form:

$$\begin{aligned} k_{1i} &:= f(t_i, x_i) \\ k_{2i} &:= f(t_i + h/(2\rho), x_i + h/(2\rho)k_{1i}) \\ x_{i+1} &:= x_i + h[(1-\rho)k_{1i} + \rho k_{2i}], \end{aligned}$$

where  $1/2 \leq \rho \leq 1$ .

- (a) If  $\rho = 1/2$ , show that this is simply the Improved Euler's method.
- (b) If  $\rho = 3/4$ , we get Heun's method. Write a MATLAB file to implement Heun's method, and test it on the initial value problem  $x' = t^2x + x \cos(t)$ ,  $x(1) = 2$  to estimate  $x(3)$ . The analytic solution can be found to be  $x(3) \approx 5764.43758$ .
- (c) Repeat part (b) with at least six values of  $h$ . Make a plot of  $\log(\text{Error})$  and  $\log(h)$  and find the slope of the regression line. What does this slope represent?
16. The basic Runge-Kutta method has a global truncation error  $O(h^4)$ . This algorithm is the industry standard for solving an initial value problem. The steps are to iterate:

$$\begin{aligned} k_{1i} &= f(t_i, x_i) \\ k_{2i} &= f(t_i + h/2, x_i + (h/2)k_{1i}) \\ k_{3i} &= f(t_i + h/2, x_i + (h/2)k_{2i}) \\ k_{4i} &= f(t_i + h, x_i + hk_{3i}) \\ x_{i+1} &= x_i + h/6(k_{1i} + 2k_{2i} + 2k_{3i} + k_{4i}) \end{aligned}$$

- (a) Write a MATLAB file to implement the Runge-Kutta method. Test your example on the initial value problem  $x' = t^2x + x \cos(t)$ ,  $x(1) = 2$  to estimate  $x(3)$ . The analytic solution can be found to be  $x(3) \approx 5764.43758$ .
- (b) Repeat part (a) with at least six values of  $h$ . Make a plot of  $\log(\text{Error})$  and  $\log(h)$  and find the slope of the regression line. What does this slope represent?
17. Nitrogen dioxide is a brown toxic gas easily formed from the reaction of nitric oxide (found naturally as well as through engine combustion and fossil fuels) with oxygen in the air. The reaction between nitric oxide (not to be confused with nitrous oxide) and oxygen to form nitrogen dioxide is given by the balanced chemical equation  $2\text{NO} + \text{O}_2 = 2\text{NO}_2$ . The rate at which  $\text{NO}_2$  is formed is given by

$$\frac{dx}{dt} = k(\alpha - x)^2 \left(\beta - \frac{x}{2}\right),$$

where  $x(t)$  denotes the concentration of  $\text{NO}_2$  at time  $t$ ,  $k$  is a rate constant,  $\alpha$  is the initial concentration of  $\text{NO}$ , and  $\beta$  is the initial concentration of  $\text{O}_2$ . At  $25^\circ \text{C}$ ,  $k \approx 7.13 \times 10^3 \frac{\text{L}^2}{\text{mol}^2\text{s}}$ . Assume that

$\alpha = 0.0010$  mol/L,  $\beta = 0.0041$  mol/L, and  $x(0) = 0$  mol/L. Verify the units match up in the differential equation.

Choose an appropriate value for  $h$  and use Euler's method or improved Euler's method to approximate  $x(10)$  with six decimal places. Compare your result to the EPA 1-hour standard of  $2.174 \times 10^{-6}$  mol/L.

18. Numerical solvers may be used for second order equations by turning them into an equivalent system of first order equations. For example, to solve the second order initial value problem

$$3x'' + 2x' - 5x = \sin(2t), \quad x(0) = 2, \quad x'(0) = 3 \quad (6.13)$$

we could define  $y(t) = x'(t)$ . Then  $y' = x''$ , and (6.13) becomes  $3y' + 2y - 5x = \sin(2t)$ . To apply Euler's method, we write the system as

$$\begin{aligned} y' &= \frac{1}{3}(-2y + 5x + \sin(2t)), & y(0) &= 3 \\ x' &= y, & x(0) &= 2. \end{aligned}$$

Use this technique to determine an appropriate system and plot the solution of the new initial value problem

$$0.5x'' + x' + 7x = \sin(t), \quad x(0) = -3, \quad x'(0) = 2$$

on the interval  $0 \leq t \leq 10$ .

19. Draw a phase line diagram for the differential equation  $x'(t) = 2 \sin(\pi x)$ . Find the equilibria and classify them as stable or unstable.
20. Consider population growth given by the differential equation  $x'(t) = rx \left(1 - \frac{x}{K}\right) (x - x_c)$ , where  $r > 0$  is the growth rate,  $K > 0$  is the carrying capacity, and  $x_c$  is some threshold value in the population with  $0 < x_c < K$ .
- Draw a phase line diagram for the differential equation. Find the equilibria and classify them as stable or unstable.
  - A population described by this type of differential equation exhibits the Allee effect. Mathematically speaking, how does population growth differ in this model as compared to logistic growth? What might be some practical reasons for this new behavior to occur in a given population?
21. A reaction-diffusion equation describes how a chemical concentration,  $C$ , changes in response to chemical reactions as well as the movement or spread of particles into the air or container. A simple model takes on the form

$$\frac{dC}{dt} = a(T - C) + f(C),$$

where  $a$  and  $T$  are positive constants. The first term describes diffusion at a rate proportional to a modified concentration, and  $f(C)$  is the rate at which the chemical is created or destroyed. For this exercise, suppose  $a = 1.0/\text{min}$ ,  $T = 4.0$  mol/L, and  $f(C) = b \frac{C}{2+C}$  with  $b = 1.0$  mol/L/min.

- To understand the reaction and diffusion, consider each term in isolation. Describe how the reaction rate  $f(C)$  depends on the concentration. Describe how the rate of change due to diffusion changes with respect to the concentration  $T$ .
  - Determine the equilibrium concentration.
  - Draw a phase line diagram for this model and determine the stability of the equilibrium concentration.
22. A general version of Newton's Law of Cooling states that the temperature inside a building is given by the solution to the differential equation

$$dT/dt = -k(T(t) - M(t)) + H(t) + U(t), \quad (6.14)$$

where  $T(t)$  is the (interior) temperature of the building at time  $t$ ,  $M(t)$  is the surrounding (outside) temperature,  $H(t)$  is the heat produced by people, lights, machines, and  $U(t)$  is the heating or cooling

provided by a furnace or air conditioner, and  $k$  is the temperature decay constant. The value  $1/k$  is called the **time constant** for the building. A typical time constant is 2 to 5 hours but it can be shorter if there are open windows or a fan.

- (a) If  $H(t) = h$ ,  $M(t) = m$  and  $U(t) = u$  are all positive constants with  $k > 0$  known and  $T(0) = T_0$ , give the general solution for  $T(t)$ .
  - (b) Suppose the conditions of part (a) hold in a building with no heating or cooling (so that  $H(t) = U(t) = 0$ ) and determine the equilibrium temperature inside the building.
  - (c) Use your results from parts (a) and (b) with a time constant of 5 hours to estimate the lowest and highest temperatures inside the building if the outside temperature consistently fluctuates between  $16^\circ\text{C}$  at 2:00 a.m. and  $32^\circ\text{C}$  at 2:00 p.m. What is a reasonable assumption to make about the value of  $T_0$  in determining each estimate?
23. Refer to Newton's Law of Cooling, (6.14), described in problem 22. Suppose that a warehouse without heating or cooling has a time constant of 4 hours. The outside temperature is  $60^\circ\text{F}$  at 7:00 a.m. and increases linearly with time up to  $90^\circ\text{F}$  at 2:00 p.m.
- (a) Set up a differential equation to model the temperature inside the warehouse between 7:00 a.m. and 2:00 p.m.
  - (b) Assume that the internal warehouse temperature is  $72^\circ\text{F}$  at 7:00 a.m. Plot the temperature during the seven hour period. [Note: it is possible to solve this analytically using theory from differential equations. Otherwise you may use a numerical solver and MATLAB to generate a plot.]
24. Suppose the population of fish in a pond is modeled with the logistic equation,  $dP/dt = 0.001P(500 - P)$ , where  $P(t)$  is the number of fish in the pond at year  $t$ .
- (a) What is the carrying capacity?
  - (b) Draw the phase line diagram for this differential equation. Classify each equilibrium as stable or unstable.
  - (c) Use Euler's method with  $h = 0.1$  to estimate the solution on the interval  $0 \leq t \leq 20$ . Solve with the initial conditions  $P(0) = 10, P(0) = 100, P(0) = 300, P(0) = 700$ . Plot the four solution curves on a single set of axes.
  - (d) Suppose that 200 fish per year are harvested in this pond. Write a new differential equation to take this assumption into account.
25. In a fish farm, a population of fish is introduced into a pond and harvested regularly. A model for the rate of change of the fish population is given by

$$\frac{dP}{dt} = r_0 \left(1 - \frac{P}{L}\right) P - h,$$

where  $P(t)$  is the population of fish at time  $t$ ,  $r_0$  is the birth rate of the fish,  $L$  is the maximum population of fish in the pond, and  $h$  is the number of the fish population that is harvested per unit time. Suppose the pond can sustain 10,000 fish, the birth rate is 90% and 1500 fish are harvested every year.

- (a) Find the equilibria, draw the phase line diagram for  $0 \leq P \leq 10000$ , and classify each equilibria as stable or unstable.
- (b) If there are initially 2000 fish in the pond, estimate how long it will take for the fish to die out.
- (c) Is it possible to guarantee survival of the fish population by changing the initial fish population? If so, what is the minimum number of fish initially needed to ensure that the fish population will not die out to zero?

- (d) Suppose there are initially 5000 fish in the pond. What levels of harvesting are sustainable? i.e. Find the maximum number of fish that may be harvested each year in order for the population to not die off.
26. Let  $P(t)$  be the population of thick-billed parrots in a particular region, at time  $t$ . The number of parrots is modeled by  $dP/dt = P(0.04 - 10^{-8}(P - 2200)^2)$ . This model is another demonstration of the Alee effect, which says that a larger population density may some times benefit all members of a population.
- (a) Estimate the equilibria, using a numerical solver.
- (b) Plot the phase line diagram for the differential equation. Classify each equilibrium as stable or unstable.
- (c) Estimate the solutions on the interval  $0 \leq t \leq 100$ , by using Euler's method with  $h = 1$  with the initial conditions  $P(0) = 500, 1000, 3000, 5000$ . Plot these four solution curves on the same set of axes.
- (d) Estimate the solution on the interval  $[0, 400]$  using initial conditions  $P(0) = 50$  and  $P(0) = 100$ . Plot these solutions on the same set of axes.
27. A nitric acid solution flows at a constant rate of 6 L/min into a large tank that initially held 200 L of a 0.5% nitric acid solution. The solution inside the tank is kept well stirred and flows out of the tank at a rate of 8 L/min. If the solution entering the tank is 20% nitric acid, determine the volume of nitric acid in the tank after  $t$  minutes. When will the percentage of nitric acid in the tank reach 10%? When will the tank be empty?
28. Suppose that a new industry starts up river from a lake at  $t = 0$  days, and this industry starts dumping a toxic pollutant,  $P(t)$ , into the river at a rate of 7g/day, which flows directly into the lake with no loss of pollutant along the bottom or banks of the river. The flow of the river is 1000 m<sup>3</sup>/day, which goes into the lake that maintains a constant volume of 400,000 m<sup>3</sup>. The lake loses 50 m<sup>3</sup>/day of (pure) water to evaporation, while the remainder of the water exits at a rate of 950 m<sup>3</sup>/day through a river. We assume that all quantities are well-mixed and that there is no time delays for the pollutant reaching the lake from the river.
- (a) Write a differential equation that describes the amount of pollutant,  $P(t)$ , in the lake.
- (b) Suppose that a concentration of 2 mg/m<sup>3</sup> is toxic to the fish population. Find how long until this level is reached.
- (c) If unchecked by regulations, find the eventual concentration of pollution in the lake.
- (d) Now suppose the lake is at this limiting level, and the industry is shut down at time  $t = 0$  days. Write a new differential equation describing this situation and solve. Find how long it takes for the lake to return to a level that allows fish to survive.
29. The half-life of morphine in the body is 2 hours, meaning that every two hours the amount of morphine is reduced by half. Suppose that morphine is administered to a patient intravenously at a rate of 1.8 mg per hour. Write a differential equation for the quantity,  $Q$ , of morphine in the blood after  $t$  hours. Determine the long-term behavior of  $Q$ .
30. In this exercise, we will model the arterial pressure,  $P_a(t)$  during a single beat of the heart. Define variables:
- $V$  is the stroke volume, the amount of blood pumped by the heart during one beat (in liters/beat).
  - $Q$  is the cardiac output, representing the amount of blood pumped by the heart (in liters/min)
  - $T$  is the duration of a heart beat (in min).
  - $P_{\text{sys}} = P_a(0)$  is the systolic (maximum) pressure.
  - $P_{\text{dia}} = P_a(T)$  is the diastolic (minimum) pressure.

- $P_v(t)$  is the venous pressure
- $Q_s(t)$  is the systemic blood flow
- $R_s$  is the systemic resistance
- $V_a(t)$  is the arterial volume
- $C_a$  is the compliance, or stretchability of a vessel

Assume that blood flow  $Q_s(t)$  satisfies  $Q_s(t) = \frac{1}{R_s}(P_a(t) - P_v(t))$ . Also assume that arterial volume satisfies  $V_a(t) = C_a P_a(t)$ .

- (a) Write an expression for  $Q$  in terms of  $V$  and  $T$ .
  - (b) Assume that the rate of change of  $V_a(t)$  satisfies  $V_a'(t) = -Q_s(t)$ . Use this to write a differential equation expressing the rate of change of  $P_a(t)$  in terms of  $C_a$ ,  $R_s$ , and  $P_a(t)$ .
  - (c) Use the initial conditions to get an explicit solution to this problem.
  - (d) An athlete has a pulse of 60 bpm, a blood pressure of 120/75, and a measured cardiac output of 6 liters per minute.
    - i. Use your work in part (a) to find  $C_a$ .
    - ii. Give the solution  $P_a(t)$  in terms of  $t$  and  $R_s$ .
    - iii. Use the condition  $P_a(T) = P_{\text{dia}}$  to find  $R_s$ .
    - iv. Plot the corresponding arterial pressure for a full heart cycle,  $0 \leq t \leq T$ .
31. Consider the initial value problem

$$\begin{aligned}x' &= 3x - 2y, & x(0) &= 3 \\y' &= 5x - 4y, & y(0) &= 6\end{aligned}$$

Find the Euler iterative formulas, and approximate  $x(0.2)$ ,  $y(0.2)$  using  $h = 0.1$ .

32. Consider the system

$$\begin{aligned}x'(t) &= -.75x - y + 4.5, \\y'(t) &= 4.0625x - 0.25y - 7.375.\end{aligned}$$

- (a) Use analytic techniques to find the equilibrium point, and determine if it is stable or unstable.
  - (b) Using initial conditions  $x(0) = 4$ ,  $y(0) = 4$ , use Euler's method to approximate the solution on the interval  $[0, 10]$ , using a step size of  $h = 0.05$ . Plot trajectories in the  $xy$  plane.
33. Suppose an object with temperature  $H$  is placed inside a room with temperature  $A$ . If the room is large compared to the object, the object will have little affect on  $A$ . However, for an object that is relatively large, a more general version of Newton's law of cooling says that

$$\begin{aligned}\frac{dH}{dt} &= \alpha(A - H), \\ \frac{dA}{dt} &= \beta(H - A).\end{aligned}$$

For the following parts, assume that  $\alpha = 0.3$ ,  $\beta = 0.2$ ,  $H(0) = 60$ , and  $A(0) = 20$ .

- (a) Find the equilibria of the model.
- (b) Plot a phase plane diagram, indicating the equilibria, nullclines, and direction of growth across the nullclines.
- (c) Use Euler's method to approximate  $H$  and  $A$  for  $0 \leq t \leq 10$ . Include a plot of  $x$  and  $y$  against  $t$ . Also include a phase-plane trajectory.
- (d) Describe what  $\alpha$  and  $\beta$  represent for this model. Explain your reasoning.

34. Consider the competition equations

$$\begin{aligned}\frac{dx}{dt} &= a \left( 1 - \frac{x+y}{K_x} \right) x, \\ \frac{dy}{dt} &= b \left( 1 - \frac{x+y}{K_y} \right) y.\end{aligned}$$

Here,  $x$  and  $y$  are populations that are in competition with each other. For the following parts, assume that  $x(0) = 750$ ,  $y(0) = 500$ ,  $a = b = 2$ ,  $K_x = 800$ , and  $K_y = 600$ .

- Find the equilibria of the model.
  - Plot a phase plane diagram, indicating the equilibria, nullclines, and direction of growth across the nullclines.
  - Use Euler's method to approximate  $x$  and  $y$  for  $0 \leq t \leq 10$ . Include a plot of  $x$  and  $y$  against  $t$ . Also include a phase-plane trajectory.
  - Describe what  $a$  and  $b$  represent for this model. Explain your reasoning.
  - Describe what  $K_x$  and  $K_y$  represent for this model. Explain your reasoning.
35. A system of differential equations may be used to model the growth of a tumor in an organism. Let  $N$  be the total number of cells in a tumor,  $P$  be the population of those cells that proliferate by splitting (exponential growth) and a population of cells  $Q$  that remain quiescent (dormant or inactive). The proliferating cells can make a transition to the quiescent state with rate  $r(N)$ , which typically increases with the overall size of the tumor. Defining  $r(N) = 1 + \ln(N)$  gives:

$$\begin{aligned}\frac{dP}{dt} &= cP - b(1 + \ln N)P, \\ \frac{dQ}{dt} &= b(1 + \ln N)P, \\ \frac{dN}{dt} &= \frac{d(P+Q)}{dt} = cP.\end{aligned}$$

- Use a numerical technique to solve for  $N$ ,  $Q$  and  $P$  over the time interval ?.
  - Compute the Jacobian matrix for the system of differential equations.
  - At each equilibrium, find the eigenvalues and use them to conclude whether the equilibrium is stable or unstable.
36. The system

$$\begin{aligned}x' &= x - x^2 - xy \\ y' &= 4y - 2xy - 7y^2\end{aligned}$$

represents two species  $x$  and  $y$  that are competing amongst themselves and between species for resources.

- Plot a phase plane diagram, indicating the equilibria, nullclines, and direction of growth across the nullclines.
- Use Euler's method with  $h = 0.1$ ,  $x(0) = 0.1$ ,  $y(0) = 0.1$  to approximate the solution on the interval  $[0, 20]$ . Show plots of  $x$  and  $y$ , and show a plot of the trajectories.
- Use Euler's method with  $h = 0.1$ ,  $x(0) = 2$ ,  $y(0) = 1$  to approximate the solution on the interval  $[0, 20]$ . Show plots of  $x$  and  $y$ , and show a plot of the trajectories.
- Compute the Jacobian matrix for the system of differential equations.
- At each equilibrium, find the eigenvalues of the Jacobian and use them to conclude whether the equilibrium is stable or unstable. Will the two populations be able to co-exist?

37. Consider the competition model:

$$\begin{aligned} dx/dt &= x(1 - x - y) \\ dy/dt &= y(6 - 2x - 3y) \end{aligned}$$

- Find all equilibria for the system.
  - Compute the Jacobian matrix for the system of differential equations.
  - At each equilibrium, find the eigenvalues of the Jacobian and use them to conclude whether the equilibrium is stable or unstable.
  - Reflecting on this model as it represents populations of two competing species, what is the most likely outcome given some initial number of individuals for  $x$  and  $y$ ?
38. In the 1930s, the Soviet ecologist Georgii Gause performed a series of experiments on competition among two yeasts with populations  $P_1$  and  $P_2$ . Independently, each yeast satisfies the logistic equations:  $P_1'(t) = 0.2P_1(\frac{13-P_1}{13})$ ,  $P_2'(t) = 0.06P_2(\frac{6-P_2}{6})$ . When grown together, he found that the interactions among the two yeasts could be modeled by:

$$\begin{aligned} \frac{dP_1}{dt} &= 0.02P_1 \left( \frac{13 - (P_1 + 3P_2)}{13} \right) \\ \frac{dP_2}{dt} &= 0.06P_2 \left( \frac{6 - (P_2 + 0.4P_1)}{6} \right) \end{aligned}$$

- Plot a phase plane diagram for positive populations. Indicate the equilibria, nullclines, and direction of growth across the nullclines.
  - Gause's experiments supported what would be called the "competitive exclusion principle". Based on your results from part (a), use your own words to explain what the "competitive exclusion principle" shows. How does this compare to the population growth of the yeasts when grown independently of each other?
39. Consider the initial value problem

$$\mathbf{y}'(t) = \begin{bmatrix} 6 & -3 & -8 \\ 2 & 1 & -2 \\ 3 & -3 & -5 \end{bmatrix} \mathbf{y}(t), \quad \mathbf{y}(0) = \begin{bmatrix} 0 \\ -1 \\ -1 \end{bmatrix}.$$

Use either Theorem 6.4.2 or 6.4.5 to find the exact solution.

40. Lead is a heavy metal that is toxic when ingested or inhaled. The compartmental model below describes the transport of lead in the blood and bones of a vertebrate body. Let  $X(T)$  and  $Z(T)$  be the amount of lead (in micrograms) in the blood and bones, respectively, at day  $T$ .

$$\frac{dX}{dT} = R - (k_3 + r)X + k_4Z \tag{6.15}$$

$$\frac{dZ}{dT} = k_3X - k_4Z \tag{6.16}$$

The parameter  $R$  gives the amount of lead entering the blood per day in micrograms.

- Determine the equilibrium amounts of lead in terms of the parameters  $R, r, k_3, k_4$ .
- Determine the non-dimensionalized system by using the substitutions

$$T = \frac{1}{k_3}t, X = \frac{R}{r}x, Z = \frac{R}{r}z.$$

Show that the dimensionless two-component lead transport model is

$$\begin{aligned} x'(t) &= q - (1 + q)x + \varepsilon z, \\ z'(t) &= x - \varepsilon z. \end{aligned} \tag{6.17}$$

- (c) Determine the equilibria for the non-dimensionalized system in terms of  $q$  and  $\varepsilon$ . Do these values match your results from part (a)?
- (d) Given  $R = 40$  and parameter values from [23] of  $r = 0.0277$ ,  $k_3 = 0.0039$ ,  $k_4 = 0.000035$ , determine the numerical values of the equilibria and their stability.
- (e) Use Euler's method with  $h = 1$  to plot solutions for  $X$  and  $Z$ . Try several time intervals to determine the approximate time it takes to reach equilibrium levels of lead in the blood and bones.
- (f) Use Euler's method to plot solutions for  $x$  and  $z$  for the equivalent value of  $t$  you found in part (e).
41. In this problem, we consider the short term dynamics of the lead-transport model in Exercise 40.
- (a) Write the system (6.16) in the form  $\mathbf{x}'(t) = A\mathbf{x}(t) + \mathbf{b}$  and solve using analytical techniques with parameter values  $R = 40$ ,  $r = 0.0277$ ,  $k_3 = 0.0039$ ,  $k_4 = 0.000035$ .
- (b) Assume a healthy individual (with no lead present in their body initially) takes in 40 micrograms of lead per day for two years, then stops ingesting any lead. What does the model predict after two years of ingesting lead at this rate? What does the model predict after four years with no additional lead ingested? Plot your solution for both the initial 2-year period of ingesting lead and the 4-year period that follows.

42. (Adapted from [15]) Suppose the biomass  $x(t)$  of a plant changes according to the differential equation

$$\frac{dx}{dt} = rx - \frac{qx}{a+x},$$

where  $a, q, r > 0$ . The first term represents plant growth and the second represents loss due to herbivory. Non-dimensionalize the equation by introducing new variables  $X = xa^{-1}$ ,  $T = rt$ ,  $Q = q(ra)^{-1}$ .

- (a) Give a differential equation for  $X(T)$ . (Your result should have no lower-case symbols.)
- (b) Note that  $X$  and  $T$  represent the plant biomass and time. If we want to study the effect of growth and herbivory on plant biomass, why might we prefer to study  $X(T)$  rather than  $x(t)$ ?
43. The Lorenz oscillator is defined by

$$\frac{dx}{dt} = \sigma(y - x), \quad \frac{dy}{dt} = x(\rho - z) - y, \quad \frac{dz}{dt} = xy - \beta z. \quad (6.18)$$

The constant  $\sigma$  is called the Prandtl number and  $\rho$  is called the Rayleigh number.

- (a) Modify your Euler's or Improved Euler's method code to work for 3 dimensions. Use the parameters  $\sigma = 10$ ,  $\rho = 28$ ,  $\beta = 8/3$ . Use initial conditions  $x(0) = 2$ ,  $y(0) = 5$ ,  $z(0) = 20$  and a step size of  $h = 0.1$  to find the solution to the Lorenz oscillator on the interval  $[0, 10]$ . To plot the trajectory in  $\mathbb{R}^3$ , use the command `plot3(x,y,z)`.
- (b) Find all equilibria for the system.
- (c) Use Euler's method or Improved Euler's method to investigate the stability of the equilibria. Use the parameters given in part (a), and use starting conditions very close to each of the equilibria and report on the long term behavior of the system.
- (d) Use the Jacobian to determine the stability of each of the equilibrium values.
44. Assume there is a disease in a finite population of size  $N$ . Further, assume the disease spreads throughout the population, and once an individual is infected, the individual recovers and are then immune to any further infection. Members of the population fall into three classes:

- $S(t)$  = the number of susceptible individuals (the number of those who have not been infected).
- $I(t)$  = the number of infected individuals.



- $R(t)$  = the number of individuals who have recovered from the infection.

The SIR epidemic model assumes that on average, an infected (and infectious) individual encounters  $a$  people per unit time. The susceptible population diminishes as they are infected (i.e. as they encounter infectious people).

$$\frac{dS}{dt} = -aSI.$$

Note that quarantining would decrease the constant  $a$ , which is needed to control the spread of the disease. Also, the infected population increases with rate  $a$ , and decreases as people recover with the recovery rate  $k$ .

$$\frac{dI}{dt} = aSI - kI.$$

Finally, the recovered population increases with the recovery rate  $k$ .

$$\frac{dR}{dt} = kI.$$

- Show that  $I(t)$  does not increase if  $S(0)$  is less than or equal to  $k/a$ .
  - Find and interpret the equilibria.
  - Set parameters  $a = 0.003, k = 0.5, I(0) = 1, S(0) = 700, R(0) = 0$ . Use Euler's method to plot  $S, R, I$  on the same set of axes. Estimate the time at which the number of infected people is at a maximum, and the time at which the number of infected people decreases to zero. Make a plot of the trajectories  $(S, I)$ .
45. Consider two languages  $x$  and  $y$  that are spoken by populations  $X(t)$  and  $Y(t)$  at time  $t$ . Assume that there is no shift from language  $x$  to  $y$ , as  $x$  is a more modern language. We want to know whether language  $y$  will continue to persist. A model developed in [22] is

$$\begin{aligned}\frac{dX}{dt} &= cXY + \alpha_x X \left(1 - \frac{X}{S_x}\right) \\ \frac{dY}{dt} &= -cXY + \alpha_y Y \left(1 - \frac{Y}{S_y}\right),\end{aligned}$$

where  $S_x$  and  $S_y$  are carrying capacities of the populations in the absence of competition,  $\alpha_x$  and  $\alpha_y$  are constants that describe the net growth rate of each population, and  $c$  is the rate of conversion from language  $y$  to  $x$  [22].

- Show that  $(0, 0), (0, S_y)$  and  $(S_x, 0)$  are all equilibria. Explain what each of these equilibria represent in context of the competing languages.
  - Show that if  $S_x < \frac{\alpha_y}{c}$  then there is another equilibrium point  $(x_e, y_e)$ , then find this equilibrium.
  - Show that the equilibrium  $(x_e, y_e)$  is stable.
  - Make a plot showing  $x_e$  and  $y_e$  both in terms of  $c$ , for  $c \in [0, 7 \cdot 10^{-4}]$ .
46. In a particular forest, the populations of owls and squirrels are modeled by

$$\begin{aligned}\frac{dx}{dt} &= 1.2x - py, \\ \frac{dy}{dt} &= 0.4x + 0.3y,\end{aligned}$$

where  $x$  is the squirrel population density and  $y$  is the owl population density.

- Show that if the predation parameter  $p$  is 0.32, both populations grow. Estimate the long-term growth rate and the eventual ratio of owls to squirrels.
- Determine the values of  $p$  that result in both species dying off.

47. As strange as it may sound, there was in fact a cylindrical tank of molasses that exploded in Boston, MA at 12:30pm on January 15, 1919. The cause of the disaster is believed to be a result of structural defects in the tank and unseasonably warm temperatures. Using information you find online at

`<https://en.wikipedia.org/wiki/Great\_Molasses\_Flood>`,

develop a more realistic model for the amount of molasses in the tank at any time  $t$  after the explosion. What assumptions will you make based on this description? Adjust the parameter values of the tank size and capacity, as well as the concentration of the molasses based on the given information. Be consistent with your units!

For a more in depth problem, consider modeling the pressure it would take for the tank to explode, the velocity of the molasses at any time  $t$  after the explosion. If the information is available on the amount of damage and location of nearby buildings, develop a model of fluid flow that would indicate the location(s) on the tank that gave way. Imagine an investigation of the incident that must occur after the fact, to determine the cause and potential fault of the explosion. Do your results match the pictures and descriptions given from historical records?

48. See [29] Derive a differential equation modeling the spread of flu among individuals in a fixed population  $N$ . That is, express the *rate of increase* of the number of infected individuals in terms of the number of infected individuals,  $I(t)$ .

$$dI/dt = \dots$$

You can assume (a) the disease is nonfatal, and so  $N$  is constant; (b) the number of susceptible individuals  $S(t)$  is the same as the number of uninfected individuals at time  $t$ ; and (c) the rate of increase of infected individuals is proportional to the number of infected times the number of susceptible individuals at any given time. Obtain a single equation by eliminating  $S(t)$  and solve for  $I(t)$  at any time  $t$ .

## Chapter 7

# Stochastic Modeling

Much real-life phenomena have uncertain factors. A stochastic model is one that takes into account the effects of such randomness. For example, predicting the weather or who will win a football game both take into account uncertainty and both include uncertainty in their predictions. First, we will define random variables and discuss how to generate random numbers in MATLAB.

A **random variable**  $X$  is a variable that takes on an unknown value that depends on chance. The set of all possible values that  $X$  can take on is called the **sample space**. The random variable is defined by a **probability distribution** which describes the probabilities of the possible outcomes given by the sample space. A random variable can be either discrete or continuous, depending on the nature of the sample space. A discrete random variable has a sample space that is countable or finite, and a continuous random variable has an uncountably infinite sample space.

To generate a random number in MATLAB, use `rand`. This generates a random number in the interval  $[0, 1]$ . You can also use `rand(n)` to generate a vector of  $n$  random numbers or `rand(n, m)` to generate a  $n \times m$  matrix of random numbers. To generate random numbers in intervals other than  $[0, 1]$ , simply scale by multiplication and/or addition. For example, to generate a random number in the interval  $[a, b]$  you would use `(b-a)*rand+a`. We can think of this process as sampling from a random variable with the uniform distribution. A **uniform random variable** is a simple example of a continuous random variable, where the variable takes on values across an interval with equal probabilities. We denote  $X \sim U(a, b)$  to be the uniform random variable on the interval  $[a, b]$ . To generate a uniformly distributed random value in the interval  $[8, 10]$ , use `2*rand+8`.

In practice, we often want to choose the value of a random variable with a finite number of choices,  $n$ , given by a discrete random variable  $X$  (see below). We can use the uniform distribution on  $[0, 1]$  and the `rand` function to do so. With this distribution, all values on the interval  $[0, 1]$  are equally likely to be chosen so dividing the interval into  $n$  equal subintervals of length  $1/n$  gives a one-to-one correspondence between the  $n$  subintervals and the  $n$  discrete values of  $X$ . For example, let the sample space of  $X$  be  $S = \{0, 1, 2\}$ . To randomly choose one of the three values for  $X$ , first let  $u \in [0, 1]$  be a random value from the uniform distribution on the interval  $[0, 1]$ . Splitting the interval  $[0, 1]$  into three equal subintervals  $[0, 1/3)$ ,  $[1/3, 2/3)$ ,  $[2/3, 1]$ , we note which interval  $u$  is contained in and assign the value of  $x \in S$ . For  $u \in [0, 1/3)$ , let  $x = 0$ ; for  $u \in [1/3, 2/3)$ , let  $x = 1$ ; and for  $u \in [2/3, 1]$ , let  $x = 2$ . See Example 7.1.1 for an equivalent method and how to program it.

For an example of a stochastic difference equation model similar to the ones discussed in Chapter 5, see [Activity A.0.54](#) and note the definitions for mean and standard deviation below.

## 7.1 Discrete Random Variables

If the sample space  $S$  of a random variable  $X$  is a countable set (one that is either finite or has infinitely many values resulting from a counting process), then we say that  $X$  is a **discrete random variable**. Examples of countable sets include the set of integers, the set of whole numbers, and any finite set. The associated probability distribution is given by the **probability mass function** (pmf),  $\{P(k)\}_{k \in S}$ , which gives the probabilities of each outcome. We denote the probability that  $X$  is equal to a particular outcome,  $k$ , by  $P(X = k)$ . The probabilities of all outcomes in the sample space must add to 1, that is,  $\sum_{k \in S} P(X = k) = 1$ .

The **mean**, or **expected value** of a discrete random variable  $X$  is the average of all the outcomes of  $X$ , weighted by the probabilities of each outcome occurring, specifically

$$E(X) = \sum_{k \in S} k \cdot P(X = k). \quad (7.1)$$

The **variance** of  $X$  measures how spread out, or varied the outcomes are, and is defined as

$$\text{Var}(X) = \sum_{k \in S} (k - E(X))^2 P(X = k). \quad (7.2)$$

The **standard deviation** of  $X$  is defined as  $\text{SD}(X) = \sqrt{\text{Var}(X)}$ .

Note that standard deviation is most often reported along with the mean (rather than the variance) since the standard deviation has the same units as the data.

See [Exercise A.0.55](#)

### Example 7.1.1

Consider the following game. You roll a fair six-sided die. If an even number turns up, you lose an amount equal to the number. If an odd number turns up, you win an amount equal to the number.

Let  $X$  be the amount won or lost on each play. Then  $X$  is a discrete random variable with sample space  $S = \{-2, -4, -6, 1, 3, 5\}$ . Since the sample space is finite,  $X$  is a discrete random variable. The probability mass function is simply the set of probabilities of each event in the sample space. Assuming the die is fair, the probabilities for each of these events are  $1/6$ . That is,

$$P(X = -2) = P(X = -4) = P(X = -6) = P(X = 1) = P(X = 3) = P(X = 5) = \frac{1}{6}.$$

The set of probabilities above make up the probability mass function for  $X$ . Using (7.1), the expected value is

$$E(X) = \sum_{k \in S} k \cdot P(X = k) = -2 \cdot \frac{1}{6} - 4 \cdot \frac{1}{6} - 6 \cdot \frac{1}{6} + 1 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} = -\frac{1}{2}.$$

Using (7.2), the variance is

$$\begin{aligned} \text{Var} &= \sum_{k \in S} (k - E(X))^2 P(X = k) = \left(-2 + \frac{1}{2}\right)^2 \cdot \frac{1}{6} + \left(-4 + \frac{1}{2}\right)^2 \cdot \frac{1}{6} \\ &\quad + \left(-6 + \frac{1}{2}\right)^2 \cdot \frac{1}{6} + \left(1 + \frac{1}{2}\right)^2 \cdot \frac{1}{6} + \left(3 + \frac{1}{2}\right)^2 \cdot \frac{1}{6} + \left(5 + \frac{1}{2}\right)^2 \cdot \frac{1}{6} = \frac{179}{12} \approx 14.9167 \end{aligned}$$

Next, suppose we want to perform a simulation of 100 games, comparing the resulting winnings (or loss) with the expected value we found. We could use the code segment:

```

%Stochastic_DieGame.m
1  numGames=100;           %number of times to play the game
2  amount=zeros(numGames,1); %vector of outcomes of each game
3  for i=1:numGames
4      k=ceil(6*rand);      %pick a random integer 1 through 6
5      if mod(k,2)==0      %if k is even
6          amount(i) = -k; %lose k
7      else                %if k is odd
8          amount(i) = k;  %win k
9      end
10 end
11 amount                 %output the values of a
12 sumAmt = sum(amount)   %output the total winnings
13 meanAmt = mean(amount) %output the average winnings per game, compare to -.5
14 varAmt = var(amount)   %output variance, compare to 14.9167

```

In line 2, we defined `amount` to be a vector of size `numGames`, to keep track of all of the outcomes. On line 4, `6*rand` generates a random variable in  $[0, 6]$  and `ceil` rounds the result up to the next integer. This results in an integer from the set  $\{1, 2, 3, 4, 5, 6\}$ .

When you run this code several times, the winnings vary considerably, although the average is approximately  $-0.50$ . By setting `numGames` to a larger value, such as 10000, the mean and variance of the winnings become closer to the theoretical values of  $-0.50$  and  $14.9167$ .

Simulating a random phenomena in this way is an example of a **Monte Carlo method** because it involves sampling from a probability distribution. In a broader sense, Monte Carlo methods are algorithms that use random number generation to simulate possible outcomes of a stochastic model.

We can visualize the results of this simulation by making a bar graph that shows the distribution of the outcomes, as shown in Figure 7.1. The  $x$  axis represents the values that  $X$  may take on, and the height of the bar represents the number or percentage of times that  $X$  takes on the value corresponding to the  $x$ -axis. The distribution in this case is roughly uniform, since all of the six outcomes have equal probabilities.

There are several ways to display the data with a bar graph. If you have the Statistics and Machine Learning Toolbox, then you can use the `tabulate()` function to create a frequency table, then use the `bar()` function on the result. If not, you can use either the `hist()` or `histcounts()` functions.

Using `histcounts()` for simulated winnings `amount` we need to define a variable `x` that specifies the endpoints of the bars.

```

x = [-6, -4, -2, 1, 3, 5];
y = histcounts(amount, x);
bar(x, y)
xlabel('Amount Won')
ylabel('Frequency')

```

---

Refer to [Activity A.0.56](#) for an additional example.

---

### Example 7.1.2

A person with type O-positive blood can receive blood only from other type O donors, either positive or negative. About 44% of the U.S. population has type O blood so it is in high demand for medical procedures. Suppose that at a blood drive, we want to know the number of potential donors we need in order to get three units of type O blood. Specifically, we are curious if 10 donors is enough - how likely

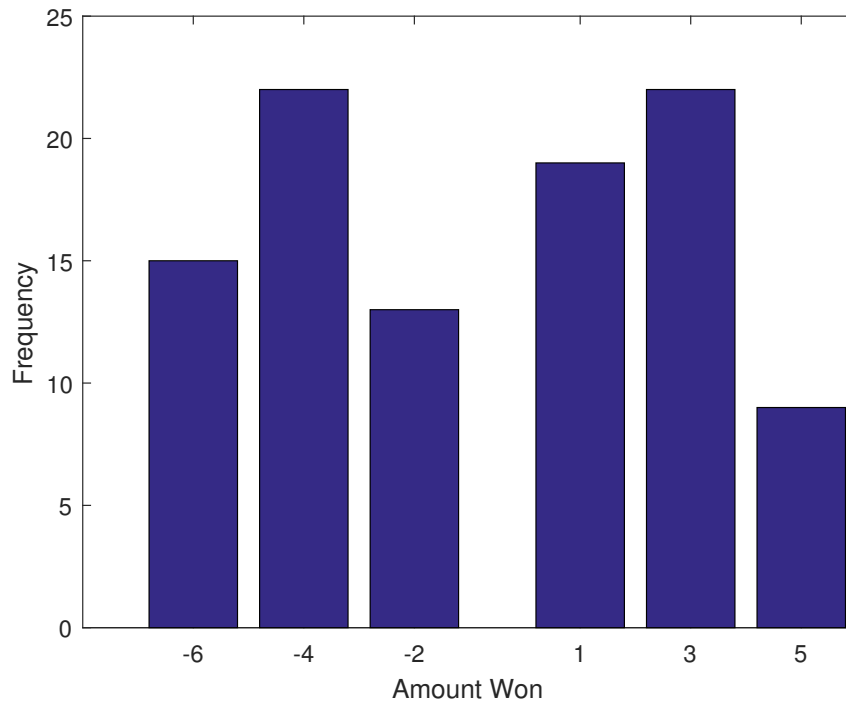


Figure 7.1: Bar graph showing the outcomes of 100 games

is it that we will need more than 10 donors? We will write a computer simulation to approximate the distribution of the number of donors needed, as well as to answer this question.

First, we want to simulate randomly drawing donors until three type O donors are found. A `while` loop is a natural tool for this task.

```

numO = 0; % number of type O donors
numDonors = 0; % number of donors
while numO < 3 % continue while there are fewer than 3 type O donors
    if rand < 0.44 % if a person is of type O
        numO = numO + 1; % increase numO by 1
    end
    numDonors = numDonors + 1; % increase the number of donors by 1
end
numDonors

```

This code segment simulates the process only one time. In order to find the distribution of donors we need to put it inside of a simulation loop.

```

ST_type0Donors.m
1 numSims = 1000;
2 numO = zeros(numSims, 1);           %number of type O donors
3 numDonors = zeros(numSims,1);      %number of donors
4 for k = 1:numSims                   %for each simulation
5     while numO(k) < 3               %continue while fewer than 3 type O donors
6         if rand < 0.44               %if a person is of type O
7             numO(k) = numO(k) + 1;   %increase numO by 1
8         end
9         numDonors(k) = numDonors(k) + 1; %increase #donors by 1
10    end
11 end
12 hist(numDonors)                    %plot distribution of numDonors
13 xlabel('Number of donors')
14 ylabel('Frequency')
15
16 meanDonor = mean(numDonors)         %calculate mean number of donors
17 stdDonor = std(numDonors)           %calculate standard deviation
18 mean10Donor = mean(numDonors > 10) %proportion w/ numDonors more than 10

```

On line 3 we vectorized the `numDonors` variable in order to keep track of the number of donors for each of the `numSims` simulations. The `numO` variable is also vectorized on line 2, although this is not required - you could certainly use a scalar variable for `numO` and reset it to 0 at the beginning of each simulation loop. A single simulation is performed in lines 5 to 10 - on these lines we draw donors until finding 3 type O donors, then keep track of the number of donors we drew in `numDonors(k)`.

On line 12 we make a histogram to visualize the number of donors. There are many ways to create a histogram with MATLAB as noted in the previous example. If `z` is a vector of numbers, then the command `hist(z)` will generate a histogram with 10 equally spaced bins. If you want to specify that there are `n` bins, use the command `hist(z, n)`.

We could opt to use `histcounts()` or customize the bin widths but a basic histogram is enough to get a rough sense of what the distribution looks like. You should that the distribution is skewed to the right.

The mean and standard deviation is output on lines 16 and 17, respectively. To estimate the probability that more than 10 donors were needed, we can use the logical expression `numDonors > 10`. This command will return 1000 values of true or false - true for the simulations that required more than 10 donors. Recall that MATLAB codes true values as a 1 and false values as a 0. So we really need to just add up the number of 1's and divide by 1000 simulations. We could equivalently just take the mean of this result: `mean(numDonors>10)`.

---

**Remark:** It is possible to use the negative binomial distribution to give analytic answers to Exercise 7.1.2. It turns out that the mean is 6.818, the standard deviation is 2.946, and the probability of needing more than 10 donors is 0.111.

## 7.2 Continuous Random Variables

If the sample space of a random variable  $X$  is uncountable, then we say that  $X$  is a **continuous random variable**. An uncountable set means that the set contains infinitely many values, and those values may be associated with measurements on a continuous scale without gaps or interruptions. Uncountable sets may

include intervals on the real line, and the real number line itself. This is often applied to variables such as time, length, weight, temperature, etc. in a stochastic model.

Since  $X$  can take on an uncountable number of values, we are only concerned with the probabilities that  $X$  takes on a certain range of values, rather than the individual values themselves. In fact, since there are so many values that  $X$  can take on, it turns out that  $P(X = k) = 0$  for any  $k$  in the sample space. To find the probability that  $X$  is in the set  $E = [a, b]$ , it does not make any sense to write  $\sum_E P(X = x_i)$  since we cannot sum over an uncountable set. However, we could split the interval  $[a, b]$  up into  $n$  subintervals of length  $\Delta x_i$ , with endpoints  $\{x_0, x_1, \dots, x_n\}$ . Then, we only need the probability that  $X$  is in each subinterval, and can use this to estimate the probability. This would give

$$P(a \leq X \leq b) \approx \sum_{i=1}^n P(X \in [x_{i-1}, x_i]).$$

We take the limit as  $n \rightarrow \infty$ , or equivalently as  $\Delta x \rightarrow 0$ , and define a function  $f$  to satisfy

$$P(a \leq X \leq b) = \lim_{\Delta x \rightarrow 0} \sum_{i=1}^n P(X \in [x_{i-1}, x_i]) = \int_a^b f(x) dx.$$

We call  $f(x)$  the **probability density function** of  $X$ . The probability density function  $f(x)$  must satisfy

$$f(x) \geq 0 \text{ for all } x \in \mathbb{R}, \text{ and } \int_{-\infty}^{\infty} f(x) dx = 1.$$

The expected value and variance of  $X$  are defined in a similar manner as in the discrete case, but we now have integrals instead of an infinite sums;

$$E(X) = \int_{-\infty}^{\infty} x f(x) dx, \quad \text{Var}(X) = \int_{-\infty}^{\infty} (x - E(X))^2 f(x) dx.$$

The **cumulative distribution function** (cdf) is used to find the probability that the random variable is below a certain value, and is defined as

$$P(X \leq x) = F(x) = \int_{-\infty}^x f(t) dt. \quad (7.3)$$

Note that by the Fundamental Theorem of Calculus,  $\frac{d}{dx} F(x) = f(x)$ . It follows that  $F$  is an increasing function and  $\lim_{x \rightarrow \infty} F(x) = 1$ .

### Example 7.2.1

Recall that a uniform random variable  $X \in U(a, b)$  takes on values across the interval  $[a, b]$  with equal probabilities. This means that for any two intervals in  $[a, b]$  of equal length, the probabilities that  $X$  is in one of these intervals is equal to the probability that it is in the other interval. More formally, for any  $a_1, a_2, b_1, b_2 \in [a, b]$  with  $a_1 < b_1, a_2 < b_2$ , and  $b_1 - a_1 = b_2 - a_2$  we have

$$P(X \in [a_1, b_1]) = P(X \in [a_2, b_2]).$$

The probability density function of  $X$  is constant and must integrate to 1. Therefore, it must have a height of  $1/(b - a)$ . In other words, the density function is

$$f(x) = \begin{cases} \frac{1}{b-a}, & x \in [a, b], \\ 0, & x \notin [a, b]. \end{cases}$$

The expected value is

$$E(X) = \int_a^b \frac{x}{b-a} dx = \frac{1}{b-a} \cdot \frac{1}{2}(b^2 - a^2) = \frac{1}{2} \cdot \frac{(b-a)(b+a)}{b-a} = \frac{a+b}{2},$$



which is simply the midpoint of the interval  $[a, b]$ . The variance is left as an exercise (8).

For example, if  $X \sim U(0, 10)$ , its probability density function is  $f(x) = \frac{1}{10}$  and its expected value is 5. To find the probability that  $X$  is between 2 and 5 we would take

$$P(2 \leq X \leq 5) = \int_2^5 \frac{1}{10} dx = \frac{3}{10}.$$

## 7.3 Properties of Random Variables

So far we have seen that if  $X$  is a discrete random variable, the expected value is  $E(X) = \sum_{k \in S} kP(X = k)$ , where  $S$  is the sample space of  $X$ . For a continuous random variable  $X$ , we have

$$E(X) = \int_S xf(x) dx,$$

where  $f$  is the density function of  $X$  and  $S$  is the set of points in  $\mathbb{R}$  for which  $f(x) > 0$ . The variance can be defined as

$$\text{Var}(X) = E[(E(X) - X)^2],$$

where this definition holds for both continuous and discrete random variables.

The following propositions gives some basic properties that hold for both discrete and continuous random variables.

**Proposition 7.3.1.** *Let  $X$  and  $Y$  be random variables, and let  $a$  be a real number. Then,*

- (a)  $E(a) = a$
- (b)  $E(aX) = aE(X)$
- (c)  $E(X + Y) = E(X) + E(Y)$
- (d)  $\text{Var}(aX) = a^2 \text{Var}(X)$
- (e)  $\text{Var}(X) = E(X^2) - E(X)^2$
- (f)  $\text{Var}(X + a) = \text{Var}(X)$

Next we need to define what it means for two random variables to have some sort of relationship to each other. We define the **conditional probability** that  $X$  takes on a specific value  $x$ , given that  $Y$  is known to take on a certain value  $y$ . We denote this probability by  $P(X = x|Y = y)$ .

The condition that  $Y = y$  may simply change the sample space for  $X$ . For example, suppose we are to draw 2 cards from a standard 52-card deck without replacement. The probability of selecting a *King* on the second draw, given that we know a *King* was drawn on the first draw is  $P(K \text{ 2nd} | K \text{ 1st}) = \frac{3}{51}$ . If instead we know a *King* was NOT drawn on the first draw, we have  $P(K \text{ 2nd} | \bar{K} \text{ 1st}) = \frac{4}{51}$ .

The basic **multiplicative law of probability** states that

$$P(X = x \text{ and } Y = y) = P(X = x)P(Y = y|X = x). \quad (7.4)$$

**Definition 7.3.2.** *Let  $X$  and  $Y$  be discrete random variables with sample spaces  $S(X)$  and  $S(Y)$  respectively. We say that  $X$  and  $Y$  are **independent** if  $P(X = x|Y = y) = P(X = x)$  for all  $x \in S(X), y \in S(Y)$ .*

In other words,  $X$  and  $Y$  are independent if the value of  $Y$  has no effect on  $X$ . The definition is symmetric, as it is possible to prove that if  $P(X = x|Y = y) = P(X = x)$  then  $P(Y = y|X = x) = P(Y = y)$  as well.

From (7.4), we see that  $X$  and  $Y$  are independent if and only if  $P(X = x \text{ and } Y = y) = P(X = x) \cdot P(Y = y)$  for all  $x \in S(X), y \in S(Y)$ . It also turns out that if  $X$  and  $Y$  are independent then their variances are additive, as stated in the following proposition.

**Proposition 7.3.3.** *Let  $X$  and  $Y$  be independent random variables. Then*

$$\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y).$$

The Central Limit Theorem is often considered to be the most important Theorem in statistics. It says that if events are repeated many times, then the average result falls into a predictable pattern.

**Theorem 7.3.4.** *(The Central Limit Theorem) Let  $\{X_i\}_{i=1}^\infty$  be a sequence of independent random variables, each having the same distribution with  $E(X_i) = \mu$  and  $\text{Var}(X_i) = \sigma^2$ . Let  $\bar{x}_n = \frac{1}{n} \sum_{i=1}^n X_i$  be the sample mean for any combination of  $n$  of the  $X_i$  random variables. Then*

- (a)  $E(\bar{x}_n) = \mu$
- (b)  $\text{Var}(\bar{x}_n) = \sigma^2/n$
- (c) *The distribution of  $\bar{x}_n$  approaches a normal distribution as  $n \rightarrow \infty$ .*

The fact that the distribution of sample means  $\bar{x}_n$  approaches a normal distribution is difficult to prove, but it can be verified using simulation. We will discuss the normal distribution in more detail later on in this chapter.

**Example 7.3.5**

Consider the experiment of flipping a coin  $n$  times. Let  $X_i$  be a random variable giving the number of heads from flipping a coin one time (either 0 or 1, each with probability  $1/2$ ).

- (a) Compute the expected value and variance of  $X_i$ .
- (b) Let  $\bar{x}_n = \frac{1}{n} \sum_{i=1}^n X_i$  be the mean number of heads appearing after flipping a coin  $n$  times. Compute the expected value and variance of  $\bar{x}_n$ .
- (c) Use simulation with 1000 trials to verify your answers to (a) and (b). Also use your code to estimate the probability of tossing more than 60% heads in 40 flips.
- (d) Use your simulation to determine a 95% confidence interval for the proportion of heads that should appear after 160 flips.

.....

- (a) The expected value is

$$E(X_i) = 0 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2} = 0.5,$$

and the variance is

$$\text{Var}(X_i) = (0 - 0.5)^2 \left(\frac{1}{2}\right) + (1 - 0.5)^2 \left(\frac{1}{2}\right) = \frac{1}{4}.$$

- (b) By Theorem 7.3.4,  $E(\bar{x}_n) = E(X_i) = p = \frac{1}{2}$  and  $\text{Var}(\bar{x}_n) = \text{Var}(X_i)/n = \frac{1}{4n}$ .

(c) Use the code segment

```

%coinFlips.m
1  numSims = 1000;           %Initialize number of simulations
2  heads = zeros(numSims, 1); %Initialize vector of number of heads
3  numFlips = 40;           %Initialize n
4  for k=1:numSims          %For all simulations
5      for i=1:numFlips      %For n flips
6          if rand < .5      %If a head was flipped
7              heads(k) = heads(k)+1; %Add 1 head to kth simulation
8          end
9      end
10 end
11 propHeads = heads/numFlips; %Convert vector from total heads to proportion
12 hist(propHeads)           %Histogram of proportions (sample means)
13 ExBar = mean(propHeads)   %Mean of sample means
14 VarxBar = var(propHeads)  %Variance of sample means
15 sigma2 = 1/4/numFlips    %Analytic variance
16 prop60 = mean(propHeads > .6) %Proportion of simulations w/ more than 60% heads

```

We want to verify that  $E(\bar{x}_n) = 0.5$  and  $\text{Var}(\bar{x}_n) = \frac{1}{4n}$ . Line 13 displays the estimated mean of the sample means (proportion of heads from 40 trials), and line 14 displays the estimated variance of the sample means. Line 15 displays the theoretical variance to be compared with the estimated variance. By running this code we should see that these values are close. Also, line 16 displays the proportions of simulations with more than 60% heads - this is used to estimate  $P(\bar{x}_n > 0.6)$ . If  $n = 40$ , you should see that the probability is roughly 0.077.

One way to shorten this code is to have MATLAB generate all of the `numFlips` random numbers in a single line. Then it is possible to set `heads(k)` to the number of random numbers that exceed 0.5. To do this, replace lines 5 through 9 with the following code:

```

r = rand(numFlips,1);
heads(k) = sum(r < .5);

```

It is actually possible to generate all of the `numSims*numFlips` random numbers in a single line which results in an even more efficient code.

(d) Suppose we want to make a 95% confidence interval for the proportion of heads that should appear after 160 flips. You may have learned how to use the normal distribution to make such an interval in a statistics class. In order to do that we would need to justify using the normal distribution, compute the mean and standard deviation of the proportion of heads, and use technology to compute probabilities (such as a z-table, TI-calculator, or computer software). However it is relatively simple to use your simulation to generate a good approximation of the confidence interval. You sort the values of the `propHeads` vector and determine the values in the 5<sup>th</sup> and 95<sup>th</sup> percentiles.

```

sortedHeads = sort(propHeads)
sortedHeads(round(.05*numSims))
sortedHeads(round(.95*numSims))

```

In the first line, we use the `sort` function to sort the values in increasing order. Then we need to look at the index of the 5th percentile - to do so we can multiply the size of the vector by .05. In case this is not a whole number, use `round` to round to the nearest whole number. The last line computes the upper end of the confidence interval. You should see a confidence interval (0.4375, 0.56875) for  $n = 160$ .

Refer to [Activity A.0.56 and A.0.57](#) for more practice with the Central Limit Theorem.

## 7.4 Monte Carlo Integration

It turns out that we may use Monte Carlo simulation in order to approximate the value of integrals. This method works well for approximating areas that cannot be found using calculus, and is especially powerful for approximating integrals in higher dimensions.

---

### Example 7.4.1

Write a Monte Carlo simulation to approximate the area of the shape  $S$  bounded by  $y = x^2$ ,  $y = 0$ ,  $x = 0$ ,  $x = 1$ .

It is true that for this case we can find the exact value using calculus:  $\int_0^1 x^2 dx = 1/3$ . Using Monte Carlo integration, we may approximate this true solution using random numbers. One way to do this is to repeatedly pick random points from a rectangle of known area, and determine which fractions of points fall into  $S$ . Notice that  $S$  lies entirely in the unit square  $D = [0, 1] \times [0, 1]$ . We could pick random points in  $D$  by taking two uniform random numbers,  $x$  and  $y$ . We then keep track of how many times that  $x^2 > y$ , as well as the fraction of random points that lie in  $S$ . The code could look something like this:

```
sum:=0
For i = 1 to n
  Choose a random number x ∈ [0, 1]
  Choose a random number y ∈ [0, 1]
  If x2 > y
    sum := sum + 1
  End
End
Output sum/n
```

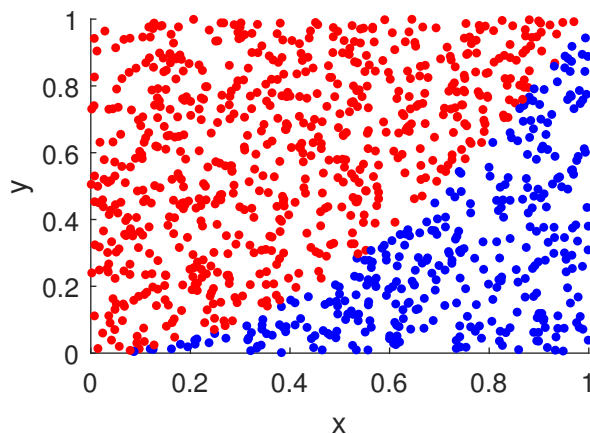


Figure 7.2: Monte Carlo Integration with 1000 sample points to approximate the area of  $S$ . Blue dots are random points in  $S$ , red dots are random points not in  $S$ .

---

There is another way to frame the Monte Carlo integration technique. Suppose we want to approximate  $\int_a^b f(x) dx$ . Instead of taking pairs of points in a rectangle about the region, we could instead take random

numbers  $x_i$  in the interval  $[a, b]$ . If we repeat this experiment, it is reasonable that the average values of  $f(x_i)$  should approach the average value of  $f$ ,  $\frac{1}{b-a} \int_a^b f(x) dx$ . In fact, you may recall from calculus that  $\sum_{i=1}^n f(x_i) \Delta x$  may be used to approximate  $\int_a^b f(x) dx$  as this is simply the right-hand approximation of the integral. Taking  $\Delta x$  to be  $\frac{b-a}{n}$  gives us

$$\frac{1}{n} \sum_{i=1}^n f(x_i) \approx \frac{1}{b-a} \int_a^b f(x) dx.$$

Under the proper conditions it is possible to prove that the approximation approaches the analytical value of the integral as  $n$  approaches  $\infty$ .

This suggests the following Monte Carlo integration algorithm to approximate  $\int_a^b f(x) dx$ .

#### Monte Carlo Algorithm to Estimate $\int_a^b f(x) dx$

```

sum := 0
For  $i = 1$  to  $n$ 
    Choose a random number  $x_i$  in  $[a, b]$ 
    sum := sum +  $f(x_i)$ 
End
Output  $A_n = \text{sum} \cdot \frac{b-a}{n}$ 

```

Recall that to get a random number  $y \in [a, b]$ , we could use `rand` to get  $x \in [0, 1]$ , and then set  $y = (b-a)x + a$ .

See [Activity A.0.58](#) for another example of approximating an integral using random numbers.

As with any numerical integration scheme, we do not expect an exact answer. The error decreases as the number of random samples increases. In the Monte Carlo method of integration, the integral is approximated by computing the average  $A_n = \frac{1}{n} \sum_{i=1}^n f(x_i)$ . Using Theorem 7.3.4, the variance of  $A_n$  approaches  $\sigma^2/n$ . Although we may not know  $\sigma$ , we know that increasing  $n$  results in decreasing the variance of  $A_n$  at a rate proportional to  $1/n$ . The standard deviation is then decreasing at a rate proportional to  $1/\sqrt{n}$ .

## 7.5 The Binomial Distribution

The binomial distribution is used to model situations in which there are several independent trials, each having two possibilities - success and failure. In this context, independence means that the result of a trial has no effect on any other trials. Therefore the probabilities of success and failure are fixed and do not change from trial to trial. These independent trials are called **Bernoulli trials**.

Examples of Bernoulli trials:

- If you flip a fair coin, there is always a 0.5 probability that it turns up heads.
- You are taking a multiple choice test that you are completely unprepared for and each question has 4 possible answers. If you randomly guess, there is a 0.25 probability of a correct answer for each question.
- A basketball player has a 70% free throw record. Each free throw is independent of the previous free throws (so the probability of success is always 0.7).

To formalize this, let  $X_i, i = 1, 2, \dots, n$  be discrete random variables that each satisfy the following. The sample space is  $\{0, 1\}$  to indicate a failure or success, respectively, and the probability mass function is

$$P(X_i = 1) = p, \quad P(X_i = 0) = 1 - p, \text{ where } 0 \leq p \leq 1. \quad (7.5)$$

Each random variable  $X_i$  models an event with two outcomes, 0 and 1. The probability that the outcome is 1 is  $p$ , and the probability that it is 0 is  $1 - p$ .

Using (7.1), we may compute the expected value of  $X_i$ ,

$$E(X_i) = 0 \cdot P(X = 0) + 1 \cdot P(X = 1) = 0 \cdot (1 - p) + 1 \cdot p = p. \quad (7.6)$$

Using (7.2) to compute the variance of  $X_i$ , we get

$$\text{Var}(X_i) = (0 - p)^2(1 - p) + (1 - p)^2 \cdot p.$$

After simplifying and canceling terms, we get

$$\text{Var}(X_i) = p(1 - p). \quad (7.7)$$

To obtain the Binomial distribution, we repeat the trials  $n$  times and consider the total number of successes,

$$S_n = X_1 + X_2 + \dots + X_n.$$

We say the random variable  $S_n$  has the **Binomial distribution** with parameters  $n, p$ . We denote this Binomial random variable by  $S_n \sim B(n, p)$ . To summarize: the Binomial distribution is used to model the number of successes out of the  $n$  independent trials.

Using Proposition 7.3.1 along with (7.6), we may compute the expected value of  $S_n$ ,

$$E(S_n) = E\left(\sum_{i=1}^n X_i\right) = \sum_{i=1}^n E(X_i) = \sum_{i=1}^n p = np. \quad (7.8)$$

Finally, using Proposition 7.3.1 with (7.7) we compute the variance to be

$$\text{Var}(S_n) = \text{Var}\left(\sum_{i=1}^n X_i\right) = \sum_{i=1}^n \text{Var}(X_i) = \sum_{i=1}^n p(1 - p) = np(1 - p). \quad (7.9)$$

The probability mass function of  $S_n$  may be justified in the following way. Since these events are independent, the probability that the first  $k$  trials are successes and the remaining  $n - k$  trials are failures is  $p^k(1 - p)^{n-k}$ . We need to multiply this probability by the number of ways that the  $k$  successes may occur from the set of  $n$  trials,  ${}_n C_k$ . Hence,

$$P(S_n = k) = {}_n C_k \cdot p^k(1 - p)^{n-k}, \text{ where } {}_n C_k = \frac{n!}{(n - k)! \cdot k!}. \quad (7.10)$$

To compute a factorial  $k!$  in MATLAB, use `factorial(k)` and to compute  ${}_n C_k$  use `nchoosek(n,k)`. For example to find  $5!$  simply take `factorial(5)` and to find  ${}_5 C_3$  use `nchoosek(5,3)`.

It is possible to use MATLAB to compute binomial probabilities if the Statistics and Machine Learning Toolbox is installed. Otherwise, you could use Excel, a graphing calculator with statistical functions, or statistical software such as R. Another option is to write your own MATLAB function to compute probabilities. The code below shows a function used to either create  $P(S_n = k)$  or  $P(S_n \leq k)$  depending on the argument `cumulative`.

```

%binomprob.m
%n=number of trials
%p=probability of success
%k=number of successes
%cumulative: 0 to find P(X=k), 1 to find P(X<=k)
function prob=binomprob(n,p,k,cumulative)
if cumulative==0
    prob=nchoosek(n, k)*p^k*(1-p)^(n-k);
else
    prob=0;
    for i=1:k+1
        prob=prob+nchoosek(n,i-1)*p^(i-1)*(1-p)^(n-i+1);
    end
end
end
    
```

To compute  $P(S_n \leq k)$  we use the fact that  $P(S_n \leq k) = \sum_{i=0}^k P(S_n = i)$ . We have to be aware that we cannot use 0 as an index for the for loop - this is why the loop runs from 1 to k+1 instead of from 0 to k, and we see i-1 in the formula instead of i.

For example, if  $n = 10$  and  $p = 0.3$  then to compute  $P(S_{10} = 4)$  you would use

```
binomprob(10, .3, 4, 0)
```

to get 0.2001 and to compute  $P(S_{10} \leq 4)$  you would use

```
binomprob(10, .3, 4, 1)
```

to get 0.8497.

**Example 7.5.1**

Consider the experiment of flipping a fair coin 40 times as we did in Example 7.3.5.

- (a) What is the distribution of the number of heads?
- (b) What are the mean and variance of the number of heads?
- (c) What is the probability of flipping over 60% heads if  $n = 40$ ?

.....

- (a) This is an example of a Binomial experiment: there are exactly two outcomes (heads and tails), each with a fixed probability 1/2. The experiment is repeated  $n = 40$  times, so the number of heads has the Binomial distribution. Letting  $X$  be the number of heads, we may write  $X \sim B(40, 0.5)$ .
- (b) Using (7.8), the mean is  $E(X) = np = 40(0.5) = 20$  and using (7.9) the variance is  $\text{Var}(X) = np(1-p) = 40(0.5)(0.5) = 10$ .
- (c) We want to find the probability of flipping over  $0.6(40)=24$  heads. Using the `binomprob` function,  $P(X > 24) = 1 - P(X \leq 24) = 1 - \text{binomprob}(40, 0.5, 24, 1) \approx 0.0769$ .

**Example 7.5.2**

Suppose you make 70% of your free throws. Describe the distribution of the number of shots made if you make 5 attempts. Find the probability of making at least 4 shots.

.....

This is a Binomial experiment with 5 trials, where the probability of success is 0.70. Hence, the number of shots made,  $X$ , has the Binomial distribution  $B(5, 0.70)$ . The probability of making at least 4 shots may be computed using  $P(X \geq 4) = P(X = 4) + P(X = 5) = {}_5C_4 \cdot 0.70^4 \cdot 0.30^1 + {}_5C_5 \cdot 0.70^5 \cdot 0.30^0 \approx 0.5282$ .

See [Exercise A.0.59](#) for additional practice.

## 7.6 The Normal Distribution

The **Normal** (or **Gaussian**) **distribution** is one of the most common continuous distributions. Its probability density function is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \quad (7.11)$$

The constants  $\mu$  and  $\sigma$  are the mean and standard deviation of the random variable. We use the notation  $X \sim N(\mu, \sigma)$  to denote that  $X$  is a random variable that is normally distributed with mean  $\mu$  and standard deviation  $\sigma$ . Due to the shape of the density function, the distribution is informally referred to as the bell-shaped curve.

It turns out that using (7.11) to compute probabilities is rather troublesome since  $P(X \in [a, b]) = \int_a^b e^{kx^2} dx$  cannot be solved analytically in general. In the past, statisticians would convert variables from the  $N(\mu, \sigma)$  distribution to the  $N(0, 1)$  distribution. To do this we introduce the random variable

$$Z = \frac{X - \mu}{\sigma}. \quad (7.12)$$

It then holds that if  $X \sim N(\mu, \sigma)$ , then  $Z \sim N(0, 1)$ . The process of converting from  $X$  to  $Z$  is called standardizing. We can think of  $Z$  as the number of standard deviations that  $X$  is above  $\mu$ . Statisticians could then use a  $z$ -table, which gives values of  $P(X \leq a) = \int_{-\infty}^a \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx$  for several different  $a$ . We can bypass the step of standardizing since we may use computers to estimate the values of the integrals.

### Example 7.6.1

Use (7.12) to show that  $Z$  has mean 0 and standard deviation 1.

.....  
Using Proposition 7.3.1(a),(b), and (c),

$$E(Z) = E\left(\frac{X - \mu}{\sigma}\right) = \frac{1}{\sigma}E(X - \mu) = \frac{1}{\sigma}(E(X) - E(\mu)) = \frac{1}{\sigma}(E(X) - \mu).$$

Since  $E(X) = \mu$ , we see that  $E(Z) = 0$ . Next, using Proposition 7.3.1(d) followed by 7.3.1(f),

$$\text{Var}(Z) = \text{Var}\left(\frac{X - \mu}{\sigma}\right) = \frac{1}{\sigma^2}\text{Var}(X - \mu) = \frac{1}{\sigma^2}\text{Var}(X).$$

Since  $\text{Var}(X) = \sigma^2$ , we see that  $\text{Var}(Z) = 1$ .

**Remark:** To take a random sample from the  $N(0, 1)$  distribution in MATLAB, use the command `randn`. Solving (7.12) for  $X$ , we see that  $X = \mu + Z\sigma$ . This means that to generate a random number from the  $N(\mu, \sigma)$  distribution, we may use the MATLAB command  `$\mu + \text{randn} * \sigma$` .



Refer to [Activities and Exercises A.0.60 - A.0.62](#) for simulations generated with a random variable that is normally distributed.

Computing probabilities for a normal random variable in MATLAB requires the Statistics and Machine Learning Toolbox. If you do not have this toolbox, you could use technology such as Excel or a TI-84 calculator. It is possible to write your own MATLAB function to compute a normal probability as well, see problem 14 in chapter 6.

To find:	In Excel	On a TI-84
$P(X \leq b)$	=NORM.DIST( $b, \mu, \sigma$ , TRUE)	normalcdf(-99999, $b, \mu, \sigma$ )
$P(X \geq b)$	=1-NORM.DIST( $b, \mu, \sigma$ , TRUE)	normalcdf( $b, 99999, \mu, \sigma$ )
$k$ such that $P(X \leq k) = p$	=NORM.INV( $p, \mu, \sigma$ )	invnorm( $p, \mu, \sigma$ )

## 7.7 Waiting Times

In this section, we will consider functions and stochastic processes to model waiting times and equipment failure times. For example, suppose we are interested in the number of incoming telephone calls to a police station in a large city. With a few assumptions, the time between two successive calls can be modeled using the exponential distribution. Also, the number of calls received in a particular unit of time can be modeled with the Poisson distribution. We will study both of these distributions in this section.

### 7.7.1 The Poisson Distribution

Suppose that we want to understand how many events happen in a fixed period of time. For example, suppose we are interested in the number of incoming telephone calls to a police station in a large city. Let  $\lambda$  be the mean number of calls per minute (assume this number is a fixed constant). We must assume that the number of calls received on any two time intervals are independent.

For example, the events  $E =$  “there are  $j$  calls between 5:00 and 5:15 P.M.” and  $F =$  “there are  $k$  calls between 6:00 and 6:15 P.M.” on the same day are independent.

So if there is an unusually large volume of calls received during one time period, it does not mean that there will be a smaller volume of calls during the next time period.

Suppose that there are an average of  $\lambda$  calls per minute. We will see that the number of calls received in a fixed time interval has the Poisson distribution. To begin to understand this process, we break up the time interval of length  $t$  into  $n$  subintervals of equal length. The length of each subinterval is then  $t/n$ . If the subintervals are very short, we may assume that there is either 0 or 1 occurrence. So during a time interval of length  $t$ , there are two possibilities:

- 1 occurrence with probability  $p = \frac{\lambda t}{n}$
- 0 occurrence with probability  $1 - p = 1 - \frac{\lambda t}{n}$

There are two outcomes: either a call occurs during the time interval or a call does not occur. We ignore the possibility that there is more than 1 occurrence because the time interval is small. Assuming that there are only two outcomes, we may model this with the Binomial distribution  $B(n, p)$ . Let  $X$  be a random variable counting the number of occurrences in given time interval of length  $t$ . For simplicity, assume that  $t = 1$ , so that  $p = \lambda/n$ . Using 7.10, we have

$$P(X = k) = {}_n C_k \cdot p^k (1 - p)^{n-k}.$$

We can show (see problem 14), that

$$\frac{P(X = k)}{P(X = k - 1)} = \frac{\lambda - (k - 1)p}{k(1 - p)}. \quad (7.13)$$

We now apply (7.13) repeatedly to arrive at a formula for  $P(X = k)$ . Starting with

$$P(X = 0) = (1 - p)^n,$$

(7.13) and  $k = 1$  implies that

$$P(X = 1) = P(X = 0) \left( \frac{\lambda}{1 - p} \right) = \lambda(1 - p)^{n-1}.$$

Similarly,

$$\begin{aligned} P(X = 2) &= P(X = 1) \left( \frac{\lambda - p}{2(1 - p)} \right) = \lambda(1 - p)^{n-1} \left( \frac{\lambda - p}{2(1 - p)} \right) = \frac{\lambda}{2}(\lambda - p)(1 - p)^{n-2}, \\ P(X = 3) &= P(X = 2) \left( \frac{\lambda - 2p}{3(1 - p)} \right) = \frac{\lambda}{2}(\lambda - p)(1 - p)^{n-2} \left( \frac{\lambda - 2p}{3(1 - p)} \right) = \frac{\lambda}{3!}(\lambda - p)(\lambda - 2p)(1 - p)^{n-3}. \end{aligned}$$

Using the fact from calculus that  $\lim_{n \rightarrow \infty} (1 - \frac{\lambda}{n})^n = e^{-\lambda}$  and  $(\lambda - \frac{\lambda}{n}) \rightarrow \lambda$  as  $n \rightarrow \infty$ , it is possible to show that

$$\lim_{n \rightarrow \infty} P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

for  $k = 0, 1, 2, 3$  (see Problem 15). Moreover, it is possible to use mathematical induction to show that this holds for all nonnegative integers  $k$ .

**Definition 7.7.1.** If  $X$  is a discrete random variable with probability mass function  $P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}$ ,  $k \geq 0$ , then  $X$  has the **Poisson distribution** with mean  $\lambda$ .

The probability  $P(X = k)$  may be thought of as the probability of  $k$  successes per unit time. The expected value is  $E(X) = \lambda$ . Sometimes  $\lambda$  is called the **rate parameter** of  $X$ . Depending on the context, we could say that  $k$  is the number of arrivals, occurrences, events, or emissions during a time period.

### Example 7.7.2

Recall the problem introduced at the beginning of this section. Suppose that telephone calls come into a police station at an average rate of  $\lambda = 4$  calls per minute.

- Find the probability that there are exactly 3 calls during one minute.
- Find the probability that there are less than 30 calls occurring in a 8-minute time interval.

To answer (a), we apply Definition 7.7.1 to get  $P(X = 3) = \frac{4^3}{3!} e^{-4} \approx 0.195$ .

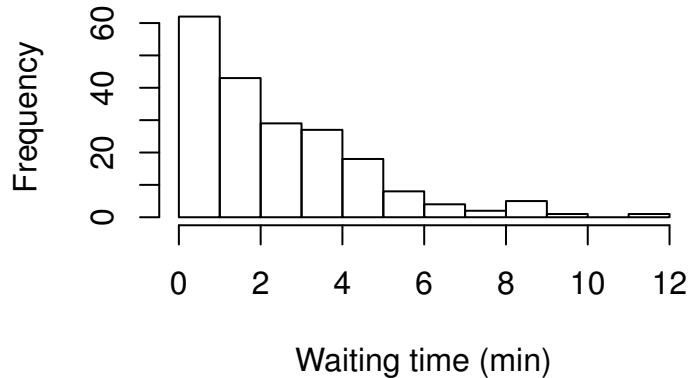
To answer (b), we need to find the average number of calls per 8 minute period, which is  $\lambda = 4 \cdot 8 = 32$ . We want to find  $P(X < 30) = \sum_{k=0}^{29} P(X = k) = \sum_{k=0}^{29} \frac{32^k}{k!} e^{-32}$ . Again, the Statistics and Machine Learning toolbox could be used to calculate this sum, but it is easy enough to find in MATLAB using

```
k=0:29
sum(32.^k./factorial(k).*exp(-32))
```

You should see that the probability is about 0.338.

### 7.7.2 The Exponential Distribution

As mentioned at the opening of this section, the probability density functions for waiting times and equipment failure times are often exponential decaying functions. Consider time (in minutes) spent waiting in line at a bank. Suppose one studies a sample of customers' waiting times over several days and generates the following histogram:



We see that the probability of spending  $t$  minutes waiting can be modeled by exponential decay.

For the situation in which events occur randomly and independently of previous events, the number of events per unit time follow the Poisson distribution, and the time between events is a continuous random variable that follows the exponential distribution.

Examples of phenomenon that are modeled with the exponential distribution include the following:

- The lifetime of some machine or device (light bulbs, batteries for example)
- The time until a radioactive particle decays
- The time between telephone calls at a call center
- The time it takes for a bank teller to service a customer
- Distance between mutations on a DNA strand

In general, we say that  $T$  is an **exponential random variable** with parameter  $\lambda$  if  $T$  has the density function

$$f(t) = \begin{cases} \lambda e^{-\lambda t} & t \geq 0 \\ 0 & t < 0. \end{cases}$$

It is easy to show that  $\int_0^{\infty} f(t) dt = 1$ ,  $E(T) = 1/\lambda$ , and  $\text{Var}(T) = 1/\lambda^2$ . From (7.3), the cumulative distribution function is

$$F(t) = P(T \leq t) = \int_0^t f(s) ds = \int_0^t \lambda e^{-\lambda s} ds = \frac{\lambda}{-\lambda} e^{-\lambda s} \Big|_0^t = -(e^{-\lambda t} - e^{-\lambda \cdot 0}) = 1 - e^{-\lambda t}. \quad (7.14)$$

Notice this also implies that

$$P(T > t) = 1 - P(T \leq t) = 1 - (1 - e^{-\lambda t}) = e^{-\lambda t}. \quad (7.15)$$

**Remark:** Some texts define the exponential distribution with the parameter  $\theta = 1/\lambda$ . Here,  $\theta$  is called the scale parameter, and represents the average time between events. The density function is then  $f(t) = \frac{1}{\theta}e^{-t/\theta}$  for  $t \geq 0$ , the expected value is  $E(T) = \theta$ , and the variance is  $\text{Var}(T) = \theta^2$ .

Let  $T$  be the time between events. To see why the times between events are exponentially distributed, suppose that the number of events per 1 unit of time,  $X$ , is Poisson with mean  $\lambda$ . From Definition 7.7.1,  $P(X = 0) = e^{-\lambda}$ . Notice that the two probabilities are equal,  $P(X = 0) = P(T > 1)$ , and matches the meaning of the variables since zero events happening during the one unit of time is the same as waiting more than one unit of time for an event to occur.

Now suppose we want to find the probability of waiting more than  $t$  units of time for an event. Then the number of events in the interval  $[0, t]$  is Poisson with mean  $\lambda t$ , so  $P(T > t) = e^{-\lambda t}$ . Notice that this agrees with (7.15), where the cumulative distribution function is  $P(T \leq t) = 1 - e^{-\lambda t}$ . It follows that the density function is  $f(t) = \frac{dF}{dt} = \frac{d}{dt}(1 - e^{-\lambda t}) = \lambda e^{-\lambda t}$ .

This shows that if the number of events  $X$  per unit time is Poisson with rate  $\lambda$ , then the time  $T$  between events must be exponential, also with rate  $\lambda$ . When establishing the value of parameter  $\lambda$ , note that  $E(T) = \frac{1}{\lambda}$  has units of *time until the next event* and  $E(X) = \lambda$  has units of *number of events per unit time*.

**Example 7.7.3**

Assume that the time spent waiting in line at a bank is exponentially distributed with a mean of 2.5 minutes.

- (a) Find the density function.
- (b) Find the probability of waiting less than 1 minute.
- (c) Find the probability of waiting more than 5 minutes.
- (d) Find the probability of waiting between 2 and 3 minutes.
- (e) Assuming there is only one line, find the probability that more than 5 people will be serviced during a 10-minute period.

.....

- (a) Let  $T$  be the time spent waiting in line. Since  $E(T) = 1/\lambda$ , it must be the case that  $\lambda = 1/2.5 = 0.4$ . The density function is therefore  $f(t) = 0.4e^{-0.4t}$  for  $t \geq 0$ . From (7.14), the cumulative distribution function is then  $F(t) = 1 - e^{-0.4t}$ .
- (b) We take  $P(T \leq 1) = F(1) = 1 - e^{-0.4} \approx 0.330$ .
- (c) We use (7.15) to see that  $P(T > 5) = e^{-0.4 \cdot 5} \approx 0.135$ .
- (d) Note, we could compute the integral  $\int_2^3 0.4e^{-0.4t} dt$  but we would be doing the same calculations as we already did. It is easier to simply compute the difference  $P(2 \leq T \leq 3) = P(T \leq 3) - P(T \leq 2)$  with

$$P(T \leq 3) - P(T \leq 2) = F(3) - F(2) = (1 - e^{-0.4 \cdot 3}) - (1 - e^{-0.4 \cdot 2}) = e^{-0.4 \cdot 2} - e^{-0.4 \cdot 3} \approx 0.148.$$

- (e) This question asks about the number of events, instead of the time between events. Here we use the Poisson distribution. Letting  $X$  be the number of customers serviced during a 10-minute period,  $X$  is Poisson with an average of  $\lambda = 0.4 \cdot 10 = 4$  customers per (every) 10-minute period. Notice that  $P(X > 5) = 1 - P(X \leq 5) = 1 - \sum_{k=0}^5 \frac{4^k}{k!} e^{-4}$ . Referring to Example 7.7.2, we can use the MATLAB code:

```
k=0:5
1-sum(4.^k./factorial(k).*exp(-4))
```

This gives a probability close to 0.215. It is easy to make minor mistakes with these calculations, so check your answer against your intuition - if an average of 4 people are serviced during a 10-minute period, it is reasonable that there is a 0.215 probability of more than 5 people being serviced during that time.

One important property of the exponential distribution is that knowledge of what has occurred in the past has no effect on future probabilities - this is sometimes called the “Markov property” or the “memoryless property”. This property can be stated as

$$P(T > t + h | T > t) = P(T > h). \tag{7.16}$$

This means that if you have already waited  $t$  minutes for an event, the probability of waiting an additional  $h$  minutes is the same as waiting  $h$  minutes immediately after an event. For example, suppose a device has a lifetime that is exponentially distributed. This Markov property says that an old device is not any more likely to break down at any time than a brand new device. In other words, the part stays as good as new until it suddenly breaks. For example, if the part has already lasted ten years, then the probability that it lasts another seven years is  $P(X > 17 | X > 10) = P(X > 7)$ . Even if this is not realistic, the exponential distribution might still do a reasonable job in modeling this type of situation.

**Example 7.7.4**

Suppose that telephone calls come into a police station at a rate of 4 calls per minute.

- (a) If a call has just come in, find the probability that it will take longer than 20 seconds for the next call to come in.
- (b) If there has not been any calls during the past 2 minutes, find the probability that there will still not be a call during the next 20 seconds.

.....

We will use the exponential distribution with rate  $\lambda = 4$ .

To answer (a), we use (7.15) to get  $P(X > 1/3) = e^{-4 \cdot 1/3} \approx 0.264$ .

For part (b), the Markov property says that  $P(X > 2 + 1/3 | X > 2) = P(X > 1/3) \approx 0.264$ .

Refer to **Activity A.0.63** for additional practice using the Poisson and exponential distributions.

**7.7.3 Sampling from the Exponential and Poisson Distributions**

We would like some way to generate random numbers from the exponential or Poisson distributions. The Statistics and Machine Learning toolkit has built-in functions to do this, as does statistical software such as R. If we want to use MATLAB and do not have access to the toolkit, it is easy enough to write our own code to do this.

It turns out that if we are able compute the inverse cumulative distribution function,  $F^{-1}(x)$ , then we may easily sample from the distribution by composing the uniform distribution with  $F^{-1}$ . This is stated in the following proposition.

**Proposition 7.7.5.** *If  $F(y)$  is a given cumulative distribution function that is strictly increasing when  $0 < F(y) < 1$  and if  $U$  is a random variable with uniform distribution on  $[0, 1]$ , then  $Y = F^{-1}(U)$  has the cumulative distribution function  $F(y)$ .*

Recall that the exponential distribution has the probability density function  $f(x) = \lambda e^{-\lambda x}$ , and the cumulative distribution function  $F(x) = 1 - e^{-\lambda x}$ . To help understand how Proposition 7.7.5 works, imagine a point  $(x, y)$  on the graph of  $F(x) = P(X \leq x) = y$ , shown in Figure 7.3. For this point  $(x, y)$ , we have  $x = F^{-1}(y)$ . To find  $F^{-1}$ , we set  $F(x) = y$  and solve for  $x$ ,

$$y = 1 - e^{-\lambda x} \implies e^{-\lambda x} = 1 - y \implies -\lambda x = \ln(1 - y) \implies x = -\frac{1}{\lambda} \ln(1 - y).$$

To find a random value  $x$  from this distribution, we simply take a random number  $y \in [0, 1]$ , then the value  $x = F^{-1}(y)$  will be an appropriate value from the distribution and whose cumulative probability  $P(X \leq x) = y$ .

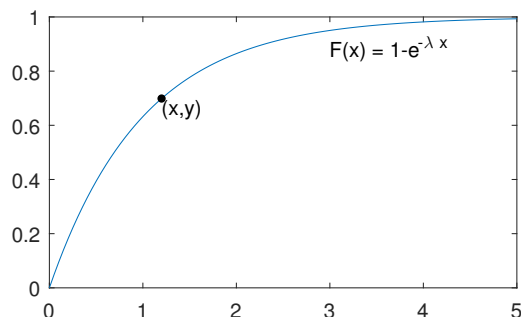


Figure 7.3: Cumulative distribution function,  $F(x) = 1 - e^{-\lambda x}$ ,  $\lambda = 1$

For example, suppose that  $\lambda = 1$  and we randomly select 0.7 from the set  $[0, 1]$ . Then we want to find the  $x$  such that  $P(X \leq x) = 0.7$ . From above,  $x = -\frac{1}{\lambda} \ln(1 - 0.7) \approx 1.20$ . Hence,  $P(X \leq 1.20) \approx 0.7$ , as desired.

To summarize, if  $Y \sim U(0, 1)$ , then

$$X = -\frac{1}{\lambda} \ln(1 - Y) \tag{7.17}$$

is exponentially distributed with probability density function  $f(x) = \lambda e^{-\lambda x}$ .

We may apply this trick to simulate the selection of any continuous random variable in which the cumulative distribution function  $F$  is invertible.

### Example 7.7.6

Suppose the time,  $T$ , that a light bulb lasts before burning out is exponentially distributed with a mean of 100 hours. Write a MATLAB code to simulate randomly selecting light bulbs and seeing how long they will last. In this case,  $\lambda = 1/100$  and the probability density function is  $f(t) = \lambda e^{-0.01t}$  for  $t \geq 0$ .

To generate 200 light bulb durations from the exponential distribution and create a histogram, we could use the MATLAB code

```
n = 200; %number of simulated values
duration = zeros(n,1); %initialize vector of random values
lambda = .01; %Events (number of lightbulbs) per hour
for i=1:n
    duration(i) = -log(1-rand)/lambda %draw random value with inverse cdf
end
hist(duration)
```

Previously, we have been using frequency histograms to represent data. Suppose now we want to make a **probability histogram**, meaning that the height of the bar corresponds with the proportion of elements lying in the interval. We first use `[x y]=hist(z,n)` to create a histogram object. The variables `x` and `y`

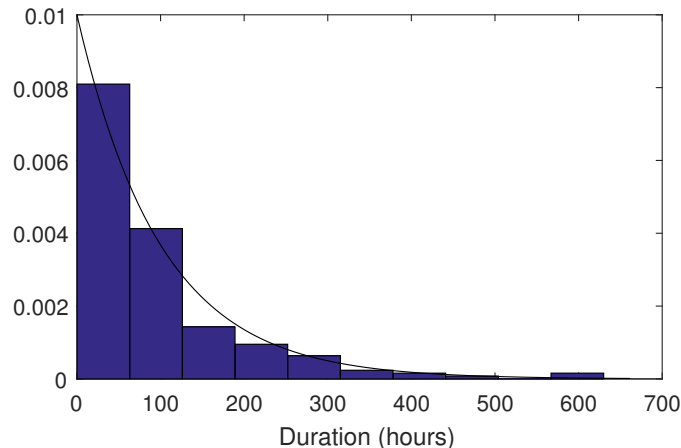


Figure 7.4: Density histogram of light bulb longevity with density function

will be  $n \times 1$  vectors; with `x` containing the bin counts and `y` containing the midpoints of the bins. To convert the counts to proportions, we divide by the the sum of all counts, by using `x/sum(x)`. Finally, we use `bar(y,x/sum(x),1)` to create a bar graph where `y` gives the centers of each bar, and `x/sum(x)` give the height of the bars. Setting the third parameter to 1 ensures that the bars will be adjacent, instead of having a gap. For our example, we should add the following lines to the end of our code to generate a probability histogram with 10 bins.

```
[x y]=hist(duration, 10);
bar(y,x/sum(x),1);
```

To compare a histogram to a probability density function, we need to scale the bin heights so that the area of the rectangles sum to 1. The result is called a **density histogram**. To do this in MATLAB, we simply need to divide by the bin width. We could use the line `binwidth = y(2)-y(1)` to compute the bin width, then plot the histogram using `bar(y,x/sum(x)/binwidth, 1)`.

We may want to plot the density function atop the probability histogram to make sure that the code is doing what it is supposed to be doing.

For our example we would add the following code to the end of the for loop - this will generate a plot like the one shown in Figure 7.4.

```
[x y] = hist(duration, 10);
binwidth=y(2)-y(1);
bar(y, x/sum(x)/binwidth,1)
hold on
T = linspace(0, max(y), 100);
plot(T, lambda*exp(-T*lambda),'k')
```

We may want to use our simulation to estimate a probability. For example if we want to know what percentage of bulbs lasted 100 hours or more, we could add the line

```
mean(duration > 100)
```

This should be close to the theoretical value obtained from the density function,  $P(T > 100) = e^{-0.01 \cdot 100} \approx 0.368$ .

Next, suppose we want to take a random sample from the Poisson distribution. We want to find the number of events that occur in some unit of time, assuming there an average of  $\lambda$  events during this unit of time. We will simply use (7.17) to generate a random waiting time from the exponential distribution, then repeat until the randomly generated times add up to one unit. The Poisson random variable is then the number of events that occurred during this unit of time.

#### To generate a Poisson( $\lambda$ ) random variable

```
numEvents := 0, time := 0
while time < 1
  Choose a random number  $y \in [0, 1]$ 
  time := time - (1/ $\lambda$ ) · ln(1 -  $y$ )
  numEvents := numEvents + 1
end
return num_Events - 1
```

#### Example 7.7.7

Continuing from Example 7.7.6, suppose a light bulb has a mean longevity of 100 hours, and we want to know how many light bulbs we will go through during a 1000 hour period of time (assume that as they die out they are instantaneously replaced). The number of bulbs used,  $X$  is Poisson with rate  $\lambda = 1000/100 = 10$  (on average we expect to use 10 bulbs during the 1000 hours). To get a sense of the distribution, we would take the code above and put it inside a simulation loop. We could use the following MATLAB code to simulate this 200 times:

```
n=200; %number of simulations
numEvents = zeros(n,1); %initialize vector of lightbulbs
lambda = 10; %lightbulbs per 1000 hr time unit
for i=1:n
  time = 0; %initialize wait time
  while time < 1 %wait time is less than one 1000-hr time unit
    randExp = - (1/lambda)*log(1-rand); %draw random value for next event
    time = time + randExp; %update total wait time
    numEvents(i) = numEvents(i) + 1; %update number of events
  end
  numEvents(i)=numEvents(i) - 1; %correct for last completion of while loop
end
hist(numEvents)
```

The resulting histogram should be relatively symmetric and centered at the mean of 10.

Refer to [Activity A.0.64](#) to practice sampling from the Poisson and exponential distributions.

## 7.8 Stochastic Processes

In section 5.4 we studied Markov Chains, where a system has finitely many states which change at discrete time intervals. If state changes may occur at continuous time intervals, we model this by a **stochastic process**.

**Definition 7.8.1.** A stochastic process is a collection  $X(t), t \in T \subset \mathbb{R}$ , where  $X(t)$  is a random variable for each  $t \in T$ . Usually, we think of  $t$  as representing time, and  $T = [0, \infty)$ . The state space of  $X(t)$  is the set of all values that  $X(t)$  may take on.



We will typically take the state space to be the set of nonnegative integers. Some models use a continuous state space, but we will use a discrete state space in this book.

### 7.8.1 Poisson Processes

One of the simplest types of stochastic processes is a Poisson process. If the number of events per unit time are Poisson with rate  $\lambda$ , then the corresponding Poisson process is the cumulative number of events that have occurred, as a function of time. As mentioned in the previous sections, this must mean that the time between events is exponentially distributed. Before defining this process, we include some notation we will use for rare events.

**Definition 7.8.2.** We say that the function  $f$  is *little-oh* of  $g(h)$  and write  $f(h) = o(g(h))$  if  $\lim_{h \rightarrow 0} \frac{f(h)}{g(h)} = 0$ .

We will use the special case where  $g(h) = h$  with notation  $o(h)$  to indicate the function  $f(h)$  such that  $\lim_{h \rightarrow 0} \frac{f(h)}{h} = 0$ . See [Exercise A.0.66](#) for an example of classifying functions as  $o(h)$  or  $o(h^2)$ .

Note that  $o(h)$  is essentially negligible when  $h$  is small. When simulating a birth process we generally use a relatively small  $h$  and then assume that  $o(h)$  is zero. In Definition 7.8.3, notice that the probability of more than one birth in a time interval of size  $h$  is  $o(h)$ . In other words,  $h$  is small enough that the probability of more than one birth occurring in a given time interval is negligible.

**Definition 7.8.3.** A *Poisson process* with rate  $\lambda$  is a stochastic process  $N(t)$ ,  $t \geq 0$ , taking values in  $\{0, 1, 2, \dots\}$  such that

- $N(t)$  is a nondecreasing function of  $t$
- $P(N(t+h) = k+1 \mid N(t) = k) = \lambda h + o(h)$ ,
- $P(N(t+h) = k \mid N(t) = k) = 1 - \lambda h + o(h)$ ,
- $P(N(t+h) > k+1 \mid N(t) = k) = o(h)$ .

Notice that the Markov property holds here, meaning that the probabilities do not change with time. The probability  $P(N(t+s) = k)$  depends only on the time duration  $s$ . Another way to say this is that

$$P(N(t+s) = k \mid N(t) = j) = P(N(s) = k \mid N(0) = j) \text{ for any } s > 0.$$

To simulate a Poisson process we could use the same code we did to sample from the Poisson distribution. We would just need to keep track of the values of  $t$  on which an event occurred.

Here, we use another strategy and discretize time using Definition 7.8.3 with a small step size  $h > 0$ . After each time step, the probability of increasing  $N(t)$  is  $\lambda h$ . Note that  $h$  must be small such that the probability  $\lambda h$  is in the interval  $[0, 1]$ .

#### Simulating a Poisson Process with rate $\lambda$

```

Initialize the current population  $N(t_0)$ 
Loop
  Choose a uniform random number  $R$  on  $[0, 1]$ 
  If  $R < \lambda h$ 
     $N(t+h) := N(t) + 1$ 
  Else
     $N(t+h) := N(t)$ 
   $t := t + h$ 
End Loop

```

Since  $\lambda$  is the average number of events per unit time, it is reasonable that  $E[N(t)] = \lambda t$ . That is, for a given value of  $t$  the expected number of events occurring in the interval  $[0, t]$  is  $\lambda t$ .

Recall the problem introduced in Section 7.7.1, where telephone calls come into a police station at a rate of  $\lambda = 4$  calls per minute. Let  $X(t)$  be the number of calls received up to time  $t$ . Then  $X(t)$  is a Poisson process with  $\lambda = 4$ . To simulate calls coming in during a ten-minute period we could use the following MATLAB code.

```

figure
hold on
lambda = 4;           %set rate
h = .01;             %set step size
tFinal = 10;        %set final time
steps = tFinal/h;   %set number of steps
numSims=20;
for k=1:numSims
    X=zeros(steps,1); %initialize X
    t=linspace(0, tFinal, steps+1); %initialize vector of times
    for i=1:steps
        if rand <= lambda*h %if an event occurs
            X(i+1)=X(i)+1;
        else
            X(i+1)=X(i);
        end
    end
    plot(t,X) %plot solution
end
T=linspace(0, tFinal, 100); %set a single vector of time values
plot(T, T*lambda, 'k', 'LineWidth',2) %plot expected value
xlabel('t')
ylabel('X(t)')
hold off

```

The last two lines are used to plot the expected value,  $E[X(t)] = \lambda t$ . The `LineWidth` parameter is used to make a thicker line. We should see that the values tend to be centered at this expected value.

Refer to [Activity A.0.65](#) for additional practice simulating a Poisson process.

## 7.8.2 Birth Death Processes

A **linear birth process**  $X(t)$  is a stochastic process in which the probability of an event is proportional to the current value of  $X(t)$ . Notice that this is very similar to a Poisson process, but with a birth process the rate increases as  $X$  increases. Although we use the term “birth”, we could use this to model any event.

**Definition 7.8.4.** We say that  $X(t)$  is a *linear birth process* if

- $P(X(t+h) = n+1 \mid X(t) = n) = n\beta h + o(h)$
- $P(X(t+h) = n \mid X(t) = n) = 1 - n\beta h + o(h)$
- $P(X(t+h) > n+1 \mid X(t) = n) = o(h),$

where  $\beta$  is the average birth rate.

By the same reason that the duration between events for a Poisson process is exponential, the duration between births for a birth process is also exponential. This model also has the Markov property, as the

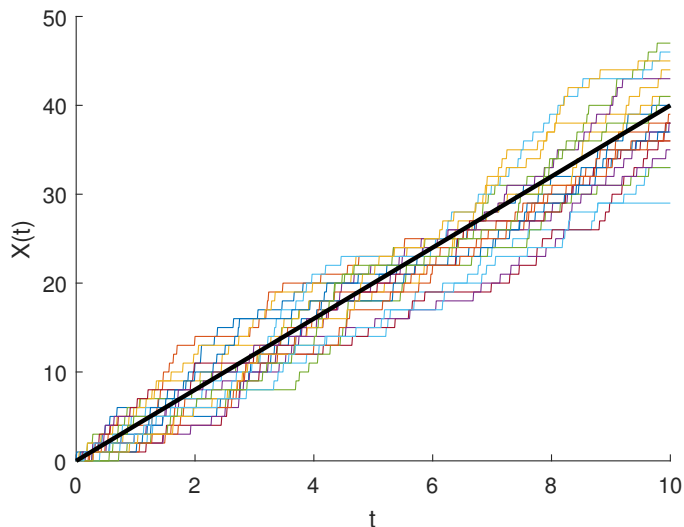


Figure 7.5: Poisson process  $X(t)$  for 20 simulations, with  $\lambda = 4, h = 0.01, X(0) = 0$ . Expected value  $E(X(t)) = \lambda t$  in black.

probability that  $X$  changes by a certain value depends only on the amount of time elapsed as well as the current value. A model having the Markov property is analogous to a differential equation model being autonomous, meaning that the rate of change does not depend on time. In practice, not all populations exhibit this property. For example, if we use  $X(t)$  to model a marmot population, then the birth rate during the spring is going to be different than the birth rate in autumn. In this case, the birth rate is dependent on time and cannot be modeled exactly with a linear birth process. However, a linear birth process might still be an approximate model for a marmot population. As is usually the case, it is a good idea to start with simplifying assumptions and add complexity if needed.

The code for simulating a birth process is almost identical to the code for simulating a Poisson process - the only difference is that the rate is proportional to  $N(t)$ .

#### Simulating a Birth Process with birth rate $\beta$

```

Initialize the current population  $X(t_0)$ 
Loop
  Choose a uniform random number  $R$  on  $[0, 1]$ 
  If  $R < X(t)\beta h$ 
     $X(t+h) := X(t) + 1$ 
  Else
     $X(t+h) := X(t)$ .
   $t := t + h$ 
End Loop

```

It turns out that the expected value of  $X(t)$  is  $E(X(t)) = X(0)e^{\beta t}$ , suggesting that this model is a probabilistic version of the differential equation model  $\frac{dx}{dt} = \beta x$ . Figure 7.6 shows 20 simulations of a birth process  $X(t)$ , with the expected value included in the plot.

If we want to take into account the possibility of a death in a population as well as a birth, we need to make a slight modification to the birth process model. Our stochastic process will now have three possibilities,

- $P(X(t+h) = n+1 \mid X(t) = n) = n\beta h + o(h)$
- $P(X(t+h) = n-1 \mid X(t) = n) = n\mu h + o(h)$
- $P(X(t+h) = n \mid X(t) = n) = o(h),$

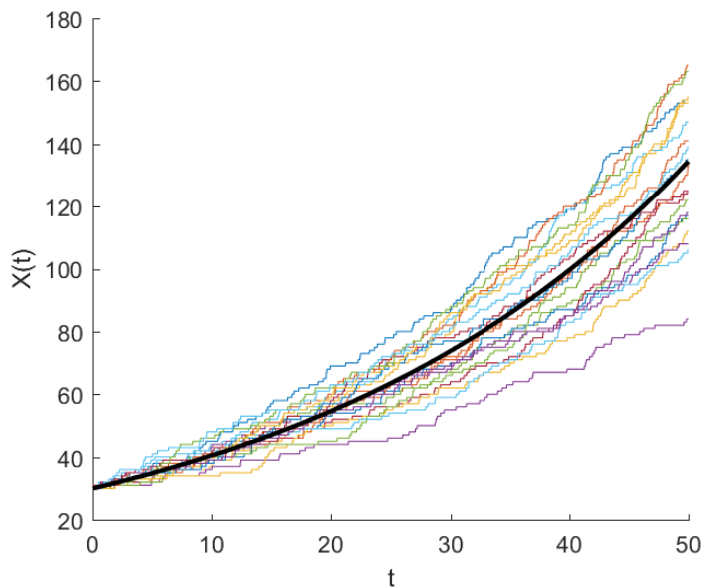


Figure 7.6: Birth process  $X(t)$  for 20 simulations, with  $\beta = 0.03, h = 0.01, X(0) = 30$ . Expected value  $E(X(t)) = 10e^{\beta t}$  in black.

where  $\beta$  is the growth rate and  $\mu$  is the death rate. This model is a **linear birth-death process** and it may be simulated in the same way as our linear birth process. Probabilities are defined and used to divide the interval  $[0, 1]$  into three regions; a random number is chosen and the population updated depending on where the number falls in the interval. It can be shown that this probability model has an expected value that solves  $x'(t) = \beta x - \mu x$  so that  $E(X(t)) = X(0)e^{(\beta - \mu)t}$ .

We can define a more general birth-death process where the birth and death rates may be any functions of the population at each time step. One more time, consider the probabilities any of the three possible events occurs to a single individual in a population of size  $X(t) = n$ , defined in the following way:

- $P(X(t+h) = n+1 \mid X(t) = n) = ng_1(n)h + o(h)$
- $P(X(t+h) = n-1 \mid X(t) = n) = ng_2(n)h + o(h)$
- $P(X(t+h) = n \mid X(t) = n) = 1 - n(g_1(n) + g_2(n))h + o(h),$

where the probability of more than one birth or death in the interval  $(t, t+h]$  is once again assumed to be zero. As you might guess, the expected value of this general birth-death process will be given by the solution of the differential equation

$$\begin{cases} x' = xg(x) = x(g_1(x) - g_2(x)) \\ x(0) = n_0 \end{cases}$$

It turns out, the opposite direction also holds. In other words, given any differential equation of the form  $\frac{dx}{dt} = x(g_1(x) - g_2(x))$ , where  $g_1(x) \geq 0$  and  $g_2(x) \geq 0$ , a probability model can be defined with the probabilities given above. For differential equations of other forms, this connection does not necessarily hold because of the properties of expected value.

Consider the differential equation for logistic growth of a population  $x(t)$  at any time  $t \geq 0$ .

$$\begin{cases} x' = x \left(1 - \frac{x}{K}\right) = x - \frac{x^2}{K} \\ x(0) = x_0, \end{cases}$$

where  $K$  is the carrying capacity for the number (or density) of individuals a particular environment or habitat can support. We'd like to build a probability model that has an expected value equivalent to the solution of this differential equation.

To find the birth and death rates, first note that the quantity  $(1 - \frac{x}{K})$  is positive for  $x < K$  and negative for  $x > K$ , which means the resulting probability, would result in a birth sometimes and a death other times. This gives the probabilities:

- $P(X(t+h) = n+1 | X(t) = n) = nh + o(h)$
- $P(X(t+h) = n-1 | X(t) = n) = \frac{n^2}{K}h + o(h)$
- $P(X(t+h) = n | X(t) = n) = 1 - n(1 + \frac{n}{K})h + o(h)$ .

## Summary of MATLAB commands

MATLAB syntax	What it does
<code>rand</code>	Generate a random number in $[0, 1]$
<code>rand(n,m)</code>	Generate a $n \times m$ matrix of random numbers in $[0, 1]$
<code>sort(x)</code>	Sort the vector $\mathbf{x}$ in increasing order
<code>randn</code>	Generate a normal random variable with mean 0 and standard deviation 1
<code>hist(x)</code>	Generate a histogram of the values in $\mathbf{x}$ with 10 equally spaced bins
<code>hist(x, n)</code>	Generate a histogram of the values in $\mathbf{x}$ with $n$ equally spaced bins
<code>bar(y, x/sum(x), 1)</code>	Generate a probability histogram with bin centers at $\mathbf{y}$ , heights of $\mathbf{x}/\text{sum}(\mathbf{x})$
<code>factorial(k)</code>	Compute $k!$
<code>nchoosek(n,k)</code>	Compute ${}_nC_k$

## 7.9 Exercises

1. Consider the stock of a particular company with initial value  $P_0 = 1$ . Suppose that for each week  $k$  the rate that the stock increases or decreases,  $r_k$ , has the probability mass function  $P(r_k = 0) = 0.3, P(r_k = -0.2) = 0.3, P(r_k = 0.2) = 0.2, P(r_k = 0.4) = 0.2$ . Therefore the value at week  $k+1$  is  $P_{k+1} = P_k + r_k P_k = P_k(1 + r_k)$ .
  - (a) Write a MATLAB program to simulate the stock's value over a one year period. Make a plot of  $P_k$  versus the week  $k$ .
  - (b) Put a simulation loop around your code to simulate several one-year periods. Use at least 1000 simulations, and generate a histogram showing the distribution of the value of the stock at the end of a year.
  - (c) Use your simulation to estimate the mean and standard deviation of the stock's value after one year.
  - (d) Use your simulation to estimate the probability that the stock went down during the one-year period (i.e.  $P_{52} < P_0$ ).
2. Write a program to carry out the following experiment. A coin is tossed 100 times and the number of heads that turn up is recorded. This experiment is then repeated 1000 times. Have your program plot a bar graph for the proportion of the 1000 experiments in which the number of heads is  $n$ , for each  $n$  in the interval  $[35, 65]$ .
3. Consider the random variable  $X$  with sample space  $S = 1, 2, 3$ , and probabilities

$$P(X = 1) = 1/2, P(X = 2) = 1/3, P(X = 3) = 1/6.$$

- (a) Compute the expected value and variance of  $X$ .
- (b) Consider this betting game. If  $X = 1$ , you lose \$2. If  $X = 2$ , you win \$1. If  $X = 3$ , you win \$3. Let  $Y$  be a new random variable indicating the amount of money won or lost. Find the expected value and variance of  $Y$ .
- (c) Write a MATLAB program to simulate playing this game 10, 100, and 1,000,000 times. In each case, find how much money was lost or won, and compare this to the expected value found in part (b).
4. Consider rolling a fair, six-sided die.
- (a) Find the expected value and variance for the number of dots on the die that occur from one roll.
- (b) Find the expected value and variance for the average number of dots that occur from 24 rolls.
- (c) Use the Central Limit Theorem to describe the approximate distribution for the average of dots occurring from 24 rolls.
- (d) Use part (c) along with a numerical integration to estimate the probability that the average number is between 3.0 and 4.0.
- (e) Write a MATLAB file that rolls a die 24 times, and outputs the average number of dots that occur. Run this program 1000 times and generate a histogram for the “average number of dots”. Plot the normal approximation density function on the same axes.
5. Consider the unit disk  $D = \{(x, y) | x^2 + y^2 \leq 1\}$ , and the unit square  $S = [0, 1] \times [0, 1]$ . Write a program that picks a random coordinate in  $S$ , and determines if this point is in  $D$  or not. If you repeat this experiment  $n$  times, let  $X$  be the number of times that the point was in  $D$ . Plot a figure showing the ratio  $X/n$  on the  $y$ -axis, and the number of trials,  $n$ , on the  $x$ -axis. Use many different values of  $n$ , up to at least  $n = 10,000$ . What do you expect this ratio to approach as  $n$  gets large?
6. (SOA): A company prices its hurricane insurance for a particular region in the United States using the following assumptions, based on average annual data. (Notice how these assumptions could seem more reasonable one year, but not the next.)
- In any calendar year, there can be at most one hurricane.
  - In any calendar year, the probability of a hurricane is 0.05.
  - The number of hurricanes in any calendar year is independent of the number of hurricanes in any other calendar year.
- (a) Using the company’s assumptions, calculate the probability that there are fewer than 3 hurricanes in a 20-year period.
- (b) Suppose the first assumption is changed to include at most two hurricanes in any calendar year. Can you still use the same model to compute the probability in part (a)? Why or why not?
7. Let  $X$  be a uniform random number on the interval  $[2, 10]$ .
- (a) Find the density function  $f(x)$ .
- (b) Find the probability of an event  $E$  for this experiment, where  $E$  is a subinterval  $[a, b]$  of  $[2, 10]$ .
- (c) Write a program to choose a random number  $X$  in the interval  $[2, 10]$ , 1000 times and record what fraction of the outcomes satisfy  $X^2 - 12X + 35 > 0$ . Compare this to the theoretical value of  $P(X^2 - 12X + 35 > 0)$ .
8. Let  $X$  be a uniform random number on  $[a, b]$ . Find  $\text{Var}(X)$ .
9. Scores on the Critical Reading portion of the SAT (Scholastic Aptitude Test) exam are normally distributed with mean 501 and standard deviation 112.
- (a) Find the estimated percentile for a student who scores 650 on Critical Reading.

- (b) Find the approximate score for a student who is at the 60th percentile for Critical Reading.
10. Heights of 10-year-old boys follow an approximate normal distribution with mean  $\mu = 55.5$  inches and standard deviation  $\sigma = 2.7$  inches.
- (a) What proportion of 10-year-old boys are between 4 ft 6 in and 5 ft tall?
- (b) If a 10-year-old boy is in the 90th percentile in height, about how tall is this boy?
11. A surveying instrument makes an error of  $-2, -1, 0, 1,$  or  $2$  feet with equal probabilities when measuring the height of a 200-foot tower.
- (a) Find the expected value and the variance for the height obtained using this instrument once.
- (b) Estimate the probability that in 18 independent measurements of this tower, the average of the measurements is between 199 and 201, inclusive.
12. The density function for an exponential random variable is  $f(x) = \lambda e^{-\lambda x}$  for  $x \geq 0$ .
- (a) Show that  $\int_0^{\infty} f(x) dx = 1$ .
- (b) Show that  $E(X) = 1/\lambda$ .
13. Suppose you are watching a radioactive source that emits particles at a rate described by the exponential density  $f(t) = e^{-\lambda t}$ , where  $\lambda = 1$ . Find the probability that a particle (not necessarily the first) will appear
- a) within the next 3 seconds.
- b) between 3 and 4 seconds from now.
- c) after 4 seconds from now.
14. Prove equation (7.13).
15. In Section 7.7.1, we saw that  $P(X = 0) = (1 - p)^n$ ,  $P(X = 1) = \lambda(1 - p)^{n-1}$ ,  $P(X = 2) = \frac{\lambda}{2}(\lambda - p)(1 - p)^{n-2}$ , and  $P(X = 3) = \frac{\lambda}{3!}(\lambda - p)(\lambda - 2p)(1 - p)^{n-3}$ . Use the facts that  $p = \lambda/n$  and  $\lim_{n \rightarrow \infty} (1 - \frac{\lambda}{n})^n = e^{-\lambda}$  to show that

$$\lim_{n \rightarrow \infty} P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

for  $k = 0, 1, 2, 3$

16. Customers arrive at a store according to a Poisson process at a rate of  $\lambda = 20$  customers per hour. Suppose the store opens at 9 a.m.
- (a) Find the probability that exactly two customers have arrived by 9:10.
- (b) Find the probability that at least 12 customers have arrived by 9:30.
17. Suppose that a certain type of wire has on average 1.2 flaws per meter. Assume the number of flaws in a given stretch of wire follows a Poisson distribution.
- (a) Find the probability that there are no flaws in a wire that is 3 meters long.
- (b) Find the probability that there are fewer than seven flaws in a wire that is 10 meters long.
18. A machine has five independently working parts and needs at least four of them to function. Each of these five parts has a longevity that is exponentially distributed with a mean of 6 months. Analyze the overall longevity of this machine. Include a histogram showing the distribution of its longevity.

19. The number of cars driving south on highway 10 has a Poisson distribution with 7 cars per minute. The cars drive east on county road B at a rate of 3 cars per minute. At one point, county road B intersects with highway 10 and ends. Assume that 10% of cars traveling south on highway 10 turn west onto B and the other 90% continue south. Also assume that 60% of cars traveling east on B turn south onto highway 10 and the others turn north. What can we say about the distribution of the number of cars going south on highway 10 to the south of this intersection? Answer this question by using a simulation or by the use of an analytic proof.
20. Suppose that customers arrive at a store according to a Poisson process at a rate of  $\lambda = 20$  customers per hour. We want to model the number of customers that arrive throughout a 9 hour period of time.
- Write a program to simulate a Poisson process that gives  $X(t)$  for all  $t \in [0, 9]$ .
  - Suppose that the rates change throughout the day. The rates are about 10 customers per hour at the time of opening at 9:00 p.m. and at the time of closing at 5:00 p.m. The peak rate is 30 customers per hour and this occurs at 1:00 p.m. Make a reasonable model giving the rate as a function of time. Then write a program to simulate a Poisson process that gives  $X(t)$  throughout the nine hours that the store is open. Make a plot showing 100 simulations. Also include a histogram showing the distribution of total customers that arrive throughout the day.
21. Usually, customers arrive at a store according to a Poisson process at a rate of  $\lambda = 20$  customers per hour. The time that a customer spends shopping is exponentially distributed with a mean of 10 minutes. Assume that if there are more than 30 customers in the store at any given point, then the store gets crowded and the rate that customers arrive slows to  $\lambda = 10$  customers per hour. Write a code to simulate the number of customers that are in the store throughout a 9 hour period of time. Include a histogram showing the distribution of total customers that arrive throughout the day.
22. Write a program to simulate logistic growth using a stochastic process model. Assume that the model is defined by

- $P(X(t+h) = X(t) + 1) = X(t)\beta h + o(h)$
- $P(X(t+h) = X(t) - 1) = \frac{X(t)^2\beta h}{K} h + o(h)$
- $P(X(t+h) = X(t)) = 1 - X(t)\beta h - \frac{X(t)^2\beta}{K} h + o(h)$ .

Assume that the carrying capacity is  $K = 100$ , the initial population is  $X(0) = 10$ , and  $\beta = 0.05$ . Make a plot showing 100 simulations on the interval  $[0, 120]$ .

23. Consider the stochastic process

- $P(X(t+h) = X(t) + 1) = X(t)\beta h + \nu + o(h)$ .
- $P(X(t+h) = X(t) - 1) = X(t)\mu h + o(h)$ .
- $P(X(t+h) = X(t)) = 1 - X(t)\beta h - \nu - X(t)\mu h + o(h)$ .

This model represents a birth death process with birth rate  $\beta$ , death rate  $\mu$  and immigration rate  $\nu$ . Write a program to simulate this model.

- Using  $X(0) = 30$ ,  $\beta = 0.02$ ,  $\mu = 0.05$  and  $\nu = 0.04$  make a plot showing 100 simulations on the interval  $[0, 20]$ .
- Using  $X(0) = 30$ ,  $\beta = 0.02$ ,  $\mu = 0.05$  and  $\nu = 0.04$ , the expected value of the population should converge. Describe what happens to  $X(t)$  in the long term. Include a 95% confidence interval for the population.
- Suppose the initial population is  $X(0) = 30$ , the birth rate is 0, the death rate is  $\mu = 0.2$ , and the immigration rate is  $\nu = 0.005$ . Estimate the probability of the population dying to extinction within  $0 \leq t \leq 40$ .



- (d) Suppose the initial population is  $X(0) = 30$ , the birth rate is 0 and the death rate is  $\mu = 0.1$ . Suppose that for the population to be sustainable, we need  $X(t) \geq 50$ . Estimate the minimum rate of immigration needed to sustain the population.

24. Suppose a large cable with lots of fibers is designed to suspend a load of 10000 kg for many years. As individual fibers break due to fatigue, they increase the load on the other fibers. Let  $X(t)$  be the number of unbroken fibers at time  $t$ . Then

$$P(X(t+h) = k-1 | X(t) = k) = \frac{A}{X(t)^B} h + o(h),$$

where  $A$  and  $B$  are constants.

Compare  $X(0) = 1000$  fibers with  $X(0) = 1200$  fibers. Run 1000 simulations each and find the average time it takes for all of the fibers to break.

25. Referring to Activity A.0.50, consider the predator-prey model

$$\begin{aligned} H'(t) &= 3H(t) - H(t)F(t) - H(t)^2 \\ F'(t) &= -0.5F(t) + 1H(t)F(t), \end{aligned}$$

Devise a stochastic process model in which  $H(t)$  and  $F(t)$  are both stochastic processes that each depend on  $H$  and  $F$ . Write code to simulate this model for  $0 \leq t \leq 20$ , using  $H(0) = 2$  and  $F(0) = 1$  as initial conditions. Include plots of  $H$  and  $F$  as functions of time.

26. Referring to Exercise 44 in Chapter 6, consider the spread of an infectious disease in a finite population. Let  $S(t)$  be the number of susceptible individuals (the number of those who have not been infected),  $I(t)$  be the number of infected individuals, and  $R(t)$  be the number of individuals who have recovered from the infection. Assume that the rate that people are infected is proportional to  $aS(t)I(t)$ , and the rate that infected people recover is  $kI(t)$ .

- Devise a stochastic process model in which  $S(t)$ ,  $I(t)$ , and  $R(t)$  are stochastic processes depend on each other.
- Set parameters  $a = 0.003$ ,  $k = 0.5$ ,  $I(0) = 1$ ,  $S(0) = 700$ ,  $R(0) = 0$ . Write code to simulate this model, and include plots of  $S$  and  $I$ .
- Using the parameters above, estimate the probability that over 100 people will end up with the disease at some point.
- Assume that  $k = 0.5$ ,  $I(0) = 1$ ,  $S(0) = 700$ , and  $R(0) = 0$ . The parameter  $a$  is related to the probability of a susceptible individual encountering an infected individual. If infected people make an effort to avoid interacting with other people, the value of  $a$  will decrease. Experiment with your code to find the value of  $a$  to ensure that the probability of the disease reaching over 200 people is under 0.8.

27. For the birth process defined in Definition 7.8.4, prove that  $E(X(t)) = x_0 e^{\beta t}$ , where  $x_0 = X(0)$ . Hint: Use induction on  $k$ .

28. Suppose a University currently has an endowment of \$150 million. Each year they spend 20% of all funds above \$60 million on operations. They invest the remainder of the funds into an account which has an annual interest rate that is normally distributed with mean 5% and standard deviation 2%. They also collect donations that are normally distributed with mean \$30 million and standard deviation \$5 million.

- Devise a model for the amount of the endowment at each year.
- Use simulation to determine the amount of the endowment in 10 years. Summarize your result by giving the mean and standard deviation of this amount, and display the distribution with a histogram.

- (c) Use simulation to analyze what happens in the long term with this spending strategy. Write up a summary of your findings.
29. The “Matching Problem” is a classic probability problem and can be stated as follows: Suppose that  $n$  married couples are at a party and that the men and women are randomly paired for a dance. A match occurs if a married couple happens to be paired together. Write a program to approximate the probability that no matches occur. Find the probabilities for the cases where  $n = 10, 50, 100$ . What happens to the probability as  $n$  gets large?
30. A basketball player makes his first free throw with probability .6. After this, he makes a free throw with probability .7 if he made the preceding one, and with probability .4 if he missed the preceding one. Write a program to simulate this process and compute the number of free throws made in a game with eight free-throw attempts. Run the simulation for 1000 games, and show a histogram showing the number of successful free throws made for each game.
31. (NBA draft) (Adapted from [25]) The National Basketball Association (NBA) draft lottery involves the 11 teams that had the worst won-lost records during the year. A total of 66 balls are placed in an urn. Each of these balls is inscribed with the name of a team. Eleven have the name of the team with the worst record, 10 have the name of the team with the second-worst record, 9 have the name of the team with the third-worst record, and so on (with 1 ball having the name of the team with the 11th-worst record). A ball is then chosen at random, and the team whose name is on the ball is given first pick in the draft of players about to enter the league. Another ball is then chosen, and if it “belongs” to a team different from the one that received the first draft pick, then the team to which it belongs receives the second draft pick. (if the ball belongs to the team receiving the first pick, then it is discarded and another one is chosen; this continues until the ball of another team is chosen.) Finally, another ball is chosen, and the team named on the ball receives the third draft pick (provided it is different from the previous two teams). The remaining draft picks 4 through 11 are then awarded to the remaining 8 teams, in reverse order of their won-lost records. For instance, if the team with the worst record did not receive any of the 3 lottery picks, then that team would receive the fourth draft pick. Let  $X$  denote the draft pick of the team with the worst record. Write a program to approximate the probability mass function of  $X$ .

## Appendix A

# In-class Activities and Exercises

### Chapter 1 Introduction

#### Exercise A.0.1

Consider an animal foraging for food and suppose that the net resource gained (energy consumed less energy expended) is given by the function  $f(t) = \sqrt{t} - 1$ . The resource gained per time spent in a given area is given by

$$g(t) = \frac{\sqrt{t} - 1}{t}, \quad t > 0.$$

(a) Show that  $g(t) \rightarrow 0$  as  $t \rightarrow \infty$ , and interpret this in context of the forager.

(b) Using calculus, find the value of  $t$  that maximizes  $g(t)$ . Explain what this means in context.

**Activity A.0.2**

Consider the sequence  $x_0, x_1, x_2, x_3, \dots$  given by

$$k, \sqrt{1+k}, \sqrt{1+\sqrt{1+k}}, \sqrt{1+\sqrt{1+\sqrt{1+k}}}, \dots$$

respectively, for some value of  $k > -1$ .

- (a) Write this as a recursively defined sequence. That is, find a way to write  $x_{n+1}$  in terms of  $x_n$ .
- (b) Pick a random value for  $k$  (it cannot be less than -1), and use a calculator or computer to compute the first ten or so terms. What does  $x_n$  seem to be converging to?
- (c) Let  $L = \lim_{n \rightarrow \infty} x_n$  be the limit of the sequence. Since we are looking for the value at which the sequence is no longer changing, this suggests that  $L = x_{n+1} = x_n$ . Use this relationship between  $x_{n+1}$  and  $x_n$  to solve for the limit  $L$ .

**Exercise A.0.3**

To describe the population dynamics of a predator and prey in some region, let  $x(t)$  and  $y(t)$  be the densities of the prey and predator population at time  $t$ , respectively. Note that densities are often measured in number of individuals per unit area, whose value depends on the species and habitat. A basic predator-prey model is

$$\begin{aligned}x'(t) &= ax(t) - bx(t)y(t) \\y'(t) &= cx(t)y(t) - dy(t)\end{aligned}$$

Here,  $a$ ,  $b$ ,  $c$ , and  $d$  are positive model parameters greater than zero.

- (a) What do these parameters represent? What assumptions are being made with this model?
- (b) Suppose the units on  $x$  and  $y$  are in number of species per acre, and time  $t$  is measured in years. What are the units of  $a$ ,  $b$ ,  $c$ , and  $d$ ?

**Exercise A.0.4**

The amount of radioactive substance at time  $t$  is described by

$$\frac{dy}{dt} = -ky. \quad (\text{A.1})$$

- (a) Solve (A.1) for  $y(t)$ .
- (b) The half-life of this particular substance is 3200 years, meaning that after 3200 years, half of the original amount remains. Use this to find the parameter  $k$ .
- (c) Suppose after 1000 years, there is 15 grams of the substance. Use this to find the remaining parameter from your solution in part (a).
- (d) How long will it take for the amount of the substance to be reduced to one gram?
- (e) Letting  $t_h$  be the half-life of the substance and  $y_0$  the initial amount at time  $t = 0$ , give a general formula that expresses  $y$  in terms of  $y_0$  and  $t_h$ .

## Chapter 2 MATLAB Programming

### Activity A.0.5

Enter the statements

```
x = 3
y = 5
```

Try adding the numbers using the command

```
x+y
```

To multiply the numbers together, use the command

```
x*y
```

Notice that MATLAB does not accept juxtaposition of variables for multiplication. If you try to enter `xy`, MATLAB is looking for a variable named “xy”.

**You must use the asterisk symbol (\*) for multiplication!**

You may change the value of  $x$  at any time. In MATLAB, and most other programming languages, the equals sign is the **assignment operator**. It computes the value on the right and stores it in the variable on the left. Try using the command

```
x = x + y
```

This last statement is an example of an **assignment statement**. It changes the value of  $x$  from 3 to 8. Mathematically speaking the equation above is not true, since  $x$  cannot be equal to  $x + y$  unless  $y$  is zero. Sometimes, when we write pseudocode, we clarify that we are making an assignment statement by using the notation:  $x := x + y$ . To work with the most current value given in the command line, use `ans`. For example, try using the command `ans^2`.

A vector is a matrix with exactly one column or row. To make a column vector  $(1, 4, 7)^T$  and save it as a variable called  $C$ , use `C = [1; 4; 7]`.

Scalar multiplication is done using `*`, for example to find  $5C$  use

```
5*C
```

We can subtract or add two vectors that are the same size. For example,

```
D=[2; -1; 5]
C+D
```

Or we can combine scalar multiplication with addition or subtraction - for example:

```
2*C - 7*D
```

To make a row vector  $[1 \ 2 \ 3]$  and save it to a variable called  $R$ , use `R = [1 2 3]`. We can take a row vector times a column vector to get a scalar - this is simply the dot product or scalar product that you may

have learned about in Physics. That is,  $RC = [1 \ 2 \ 3] \cdot \begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix} = 1 \cdot 1 + 2 \cdot 4 + 3 \cdot 7 = 30$ . Try it:

```
R*C
```

If two vectors (or matrices) are of the same size, we can use the `.*` command to multiply 'element by element'. For example, since the transpose of  $R$  is a column vector of the same size as column vector  $C$ , we could define  $P = R'$  to be a  $3 \times 1$  column vector. Recall that the transpose operator is indicated by the single quote symbol, `'`, in MATLAB. Thus, the command

```
P.*C
```

gives  $1 \cdot 1 + 2 \cdot 4 + 3 \cdot 7 = 30$ , the same result as if we multiplied `R*C`. Placing a `'` before other operations (`*`, `/`, `^`, etc.) will perform those operations 'element by element', provided the two vectors or matrices are of the same dimension. Addition and subtraction are already performed 'element by element' so they do not need the `'` command, only the same dimensions. Try it:

```
P./C
P.^C
P + C
P - C
```

Note that performing any operations on vectors or matrices with mismatched dimensions will give an error message. For example `R + C` gives an error, whereas `R'+ C` does not.

Next, let's practice working with matrices. Enter the matrix  $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ , using the command:

```
A = [1 2 3; 4 5 6]
```

Since this matrix has 2 rows and 3 columns, we say it is a  $2 \times 3$  matrix. We can use parentheses `()` to access individual elements, rows, or columns of this matrix.

Experiment with the following commands.

MATLAB command	What it does
<code>A(2,1)</code>	Output $A_{21}$ (row 2 and column 1 of $A$ )
<code>A(1,:)</code>	Output the first row of $A$
<code>A(:,1)</code>	Output the first column of $A$
<code>rref(A)</code>	Output the reduced row Echelon form of $A$
<code>size(A)</code>	Output the size of $A$ , in the form: [number of columns    number of rows]
<code>2*A</code>	Multiplies each element of $A$ by 2 to find $2A$
<code>A.*B</code>	Multiplies each element of $A$ by the corresponding element of $B$
<code>A'</code>	Output the transpose of $A$ (interchange rows with columns)

A **logical expression**, or **boolean expression** is a statement that is either true or false. In MATLAB (and most other computer languages), a value of true is coded as 1 and false is coded as 0.

For example, we know that 2 is less than 3, so the statement " $2 < 3$ " is a true statement. Let's see what MATLAB says:

```
2 < 3
```

It should return 1 for true. Let's try a false statement:

```
7 > 10
```

We can use `"=="` to test whether two quantities are equal or not:

```
2 == 3
2 == 10/5
```

Recall that a single `"="` will assign the quantity on the right-hand side to the variable on the left-hand side.

If we wanted to instead whether two quantities are NOT equal, we would use `~=` instead, for example:



```
5 ~= 8
8 ~= 8
```

We see that MATLAB returns TRUE if the quantities are not equal.

We can apply a logical expression to an entire vector or matrix. For example, let's first define the vector  $v$  with integers from 1 to 12. With the command below, the missing increment `inc` has a default of 1.

```
v = 1:12
```

The `isprime()` function returns TRUE if a number is a prime number and FALSE otherwise. Try calling it with `v`.

```
isprime(v)
```

Since true entries are coded as 1 and false entries are coded as 0, we can treat them as numbers - for example we could compute the total number or the proportion of prime numbers in `v`:

```
sum(isprime(v))
mean(isprime(v))
```

### Activity A.0.6

Plot  $\sin(x)$  on the interval  $-3 \leq x \leq 3$  by using the command `h = ezplot('sin(x)',[-3 3])`. You will see a plot of  $\sin(x)$  appear. We have also assigned the plot of  $\sin(x)$  to the **handle**  $h$ . We can use this handle to manipulate the plot later on. For example, make the plot red by using the command `set(h,'Color','r')`. You can instead use the letters y, m, c, g, b, w, k to set the color to yellow, magenta, cyan, green, blue, white, or black, respectively. Instead of using one of the predefined colors, you may use `[r g b]` where r, g, and b are numbers between 0 and 1 that say the amount of red, green, and blue to use to create the color, with 1 being the brightest and 0 being the dimmest.

Let's say we want to add another plot to the curve. We first have to use:

```
hold on
```

to tell MATLAB that we would like to work with the current plot figure instead of opening a new figure. All of your plot commands apply to the current figure until you either close the figure, open a new figure, or use the `hold off` command.

To close the most recent figure opened, use the command:

```
close
```

To close every figure that's open, use the command `close all`.

For practice with the plotting, first plot the function  $y = x/2$  by using `h2 = ezplot('x/2')`.

You may notice that the x and y limits have expanded to  $\pm 6$ . To remedy this, redraw the plot while specifying the x-limits:

```
h2 = ezplot('x/2',[-3,3])
```

Another option would be to use the `axis()` function. This requires you to pass in both the x and y limits, for example:

```
axis([-3, 3, -1.5, 1.5])
```

sets the plot window to  $-3 \leq x \leq 3, -1.5 \leq y \leq 1.5$ . We can set the color of the  $y = x/2$  line to red by using:

```
set(h2,'Color','r')
```

### Activity A.0.7

Plotting points is relatively straightforward - to plot the point (2,3) for example, simply use

```
plot(2, 3)
```

This is difficult to see, because MATLAB plots the point using a single pixel. Alternatively, plot a circle by using

```
plot(2, 3, 'o')
```

There are many such markers that may be used, for example you could have used 'x', '+', '\*', 's', or 'd'. Experiment with these different markers. To see a summary use

```
help plot
```

For plotting curves, the plot function works differently than ezplot because it accepts vectors as parameters instead of functions.

To plot a sequence, simply pass in the values of the sequence as the parameter. For example, to plot the values of the sequence {1, -3, 2, 5}, against their index (1, 2, 3, 4) in this case, use:

```
plot([1, -3, 2, 5])
```

By default this gives a plot of the points connected by straight lines. To instead use a marker (such as a circle) for example, use

```
plot([1, -3, 2, 5], 'o')
```

You can also set the color in the same line as the plot command, try it here:

```
plot([1, -3, 2, 5], 'o', 'col','m')
```

To plot a function of the form  $y = f(x)$ , we need to express  $y$  and  $x$  as vectors of equal sizes. For example, if  $\mathbf{x}$  and  $\mathbf{y}$  are both vectors (of the same size), you can use the command `plot(x, y)` to plot the set of ordered pairs  $(x(i), y(i))$ . MATLAB will plot these points, and connect them with straight lines by default.

To plot a function  $y = f(x)$ , the `linspace()` function will be helpful. To plot  $y = \sin(x)$  on the interval  $-3 \leq x \leq 3$  with  $n = 10$  gridpoints, use:

```
x = linspace(-3, 3, 10)
y = sin(x)
plot(x, y)
```

Notice that the plot looks a little rough - this is because we are only specifying the function for 10 points on the interval  $[-3, 3]$ . Notice that  $\mathbf{x}$  and  $\mathbf{y}$  both contain 10 numbers. So MATLAB is actually making a scatter plot of these 10 ordered pairs and connecting them with straight lines.

Let's try again but this time use 100 points instead of 10. Also, we defined the variable  $\mathbf{y}$  above but this was not necessary - we could have simply used `sin(x)` as the second argument in the plot command. Let's try this:

```
x = linspace(-3, 3, 100)
plot(x, sin(x))
```

We can label the axes and give it a title using

```
xlabel('x')
ylabel('y')
title('y=x^2')
```

We may use `axis()` to specify the x and y limits. For example, if you want to display a little extra space above the extreme values of this function, use:

```
axis([-3, 3, -2, 2])
```

The default graphic in MATLAB gives the values of each axis on the outer edge of the figure box, where the origin is not always in the lower left corner. In these cases, it is sometimes helpful to see the x and y axes on the plot. While there are a few ways to do this, one way to get the x-axis is to use:

```
hold on
plot([-3, 3], [0, 0], 'k')
```

As before, we use `hold on` to ensure that this line is plotted atop the current figure window. This plot command essentially connects the points (-3, 0) and (3, 0) with a straight line in black. Another way to do this is to use your x values:

```
plot(x, 0*x, 'k')
```

By the way, the y-coordinate is `0*x` because we want a vector of zeros, of the same size as x. If you simply put `plot(x, 0, 'k')`, MATLAB thinks you are plotting an entire vector against the scalar, 0, and does not produce the desired result. We can do a similar trick to get the y-axis:

```
plot([0, 0], [-1.5, 1.5], 'k')
```

In addition to changing the color of the line, we can change the line style to dotted, dash-dot, or dashed. For example, to plot the function  $y = x/2$  on top of our current plot, use:

```
plot(x, x/2, '-')
```

Finally, we can combine using different colors, line types, and point markers, we simply specify all of these options in a single character string. For example, suppose we want to add a plot of cosine with 20 points marked with an astericks, and the points connected with dashed lines.

```
x = linspace(-3, 3, 20)
plot(x, cos(x), '*-')
```

In addition, we can specify the color - this goes into the same parameter as the marker and line option - in any order!

```
plot(x, cos(x), 'm*-')
```

Besides having MATLAB mark each point with a symbol, you can also change the width and style of the line. To give an example:

```
plot(x, cos(x), '-sb', 'MarkerSize', 4, 'LineWidth', 2)
```

In the command above, `'-sb'` says to use a blue square markers and connect them with solid lines. The squares are set to have a diameter of 4 pixels, and the line width is set to 2 pixels.

Use `help plot` to review the different options for colors, line types, and marker types.

- (a) Plot the function  $f(x) = 2/x$  on the interval  $[0, 2]$ . Again, we want to use `linspace()` to set up a grid of  $x$  values on the interval  $[0, 2]$ :

```
x = linspace(0, 2, 100)
```

Your first thought may be to use the line:

```
plot(x, 2/x)
```

However this results in the error that the matrix dimensions must agree. The problem is that `x` is a vector of size 100, while 2 is a scalar. We actually want to apply  $2/x_i$  for each element  $x_i$  in `x`. To tell MATLAB to apply the operation to each element, use the dot (`.`) before the operation, whether it be `+`, `-`, `^`, or `/`. Try it:

```
plot(x, 2./x)
```

- (b) Plot the function  $g(x) = \frac{x^2-x}{8-4x}$  on the interval  $-1 \leq x < 3$ .

Here, we need to tell MATLAB that the squaring and division applies on each element of the vector:

```
x=linspace(-1, 3, 100)
y=(x.^2 - x)./(8-4*x)
plot(x,y)
```

And now for a few bookkeeping items:

Notice that the function  $y$  is not defined for  $x = 2$ . We sort of lucked out because our vector `x` happened to not include the number 2 exactly. If we would have picked a different vector size when we called `linspace`, (say 101 instead of 100), then the corresponding  $y$  would be assigned a value of `Inf`, and simply would not be included in the plot.

In addition, we got away with using `4*x` instead of `4.*x` although either of these would have worked. The reason is that MATLAB interprets `3*x` as scalar multiplication. If in doubt, simply include the dot.

You may have noticed that when you run these lines, MATLAB outputs all 100 values of  $x$  and  $y$  onto the command window. Any time you use an assignment statement, the new value is output to the command window. Outputting numbers to the command window is sometimes undesirable because of the visual clutter and it can also slow down your program. To avoid values being output to the command window, simply end the assignment statement with a semicolon. For example:

```
x=linspace(-1, 3, 100);
y=(x.^2-x)./(8-4*x);
plot(x,y)
```

**Activity A.0.8**

Suppose we want to plot the following piecewise defined function on  $0 \leq x \leq 2$ .

$$f(x) = \begin{cases} x^2, & 0 \leq x < 1 \\ 2 - x & 1 \leq x < 2. \end{cases}$$

A plot can be made using either `ezplot()` or `plot()`. To use `ezplot()`, we need to plot each part of the graph separately, and specify the range of  $x$ -values for each of these. To ensure that the two pieces are plotted on the same set of axes, we need to call `ezplot` after the first plot is called. At the end we can adjust the viewing window with the `axis` function.

MATLAB Commands	What it does
<code>ezplot('x^2', [0, 1])</code>	plot $y = x^2$ on $[0, 1]$
<code>hold on</code>	continue with the current figure
<code>ezplot('2-x', [1, 2])</code>	plot $y = 2 - x$ on $[1, 2]$
<code>axis([0, 2, 0, 1])</code>	adjust viewing window

Using `plot()` requires a little bit more coding, but it will be useful later when we do not have an explicit formula for the function that we want to plot. Here is one way to make this plot:

MATLAB Commands	What it does
<code>x1 = linspace(0, 1, 100);</code>	make a grid of $x$ -values on $[0, 1]$
<code>y1 = x1.^2;</code>	set $y = x^2$ on $[0, 1]$
<code>x2 = linspace(1, 2, 100);</code>	make a grid of $x$ -values on $[1, 2]$
<code>y2 = 2-x2;</code>	set $y = 2 - x$ on $[1, 2]$
<code>plot(x1, y1)</code>	plot $y = x^2$
<code>hold on</code>	continue with the current figure
<code>plot(x2, y2)</code>	plot $y = 2 - x$
<code>axis([0, 2, 0, 1])</code>	adjust viewing window

It turns out that we can combine the two plot calls into a single plot call. This has the advantage that the viewing window will be automatically set appropriately. Also, it turns out that it will be much easier to add a legend, shown below for the sake of example.

```
x1 = linspace(0, 1, 100);
y1 = x1.^2;
x2 = linspace(1, 2, 100);
y2 = 2-x2;
plot(x1, y1, x2, y2)
legend('y=x^2', 'y=2-x')
```

Note that the value of  $x = 1$  was used in the domains of both functions in order to make the graph. In cases where a discontinuity occurs for other piecewise-defined functions, you may wish to add an open and/or closed circle at the ends of each function piece. Experiment with adding each line of code to your graphs above.

```
plot(1, 1, '.', 'markersize', 20)
plot(1, 1, 'o', 'markersize', 5)
```

**Activity A.0.9**

Define  $x$  as a symbolic variable, using

```
syms x
```

Next, define a function:

```
f=sin(x^2)
```

Try experimenting with the following commands:

MATLAB command	What it does
<code>solve(f)</code>	Approximates the zeros of the function $f(x)$
<code>diff(f,x)</code>	Compute $f'$
<code>diff(f,x,2)</code>	Compute $f''$
<code>int(f,x,0,pi)</code>	Integrate $f$ from 0 to $\pi$
<code>vpa(s)</code>	Expresses the number $s$ as a decimal
<code>subs(f,x,pi/2)</code>	Evaluate $f(\pi/2)$
<code>syms x y</code>	Define two symbolic variables, $x$ and $y$
<code>ezsurf(x^2 + y^2, [-2 2 -3 3])</code>	Plot the 3d surface $z = x^2 + y^2$ on the interval $[-2, 2] \times [-3, 3]$

If MATLAB cannot compute a definite integral analytically, it will output:

**Warning: Explicit integral could not be found.**

You can use the `vpa` command to find a decimal approximation.

Next, try to define the function  $g(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2}$ . The list below shows how to enter functions such as  $e^x$  and  $\sqrt{x}$  in MATLAB.

Function	MATLAB syntax
$\sqrt{x}$	<code>sqrt(x)</code>
$e^x$	<code>exp(x)</code>
$\ln(x)$	<code>log(x)</code>
$\log_{10}(x)$	<code>log10(x)</code>
$ x $	<code>abs(x)</code>
$a^x$	<code>a^x</code>
$\sin(x), \cos(x), \tan(x)$	<code>sin(x), cos(x), tan(x)</code>
$\sin^{-1}(x), \cos^{-1}(x), \tan^{-1}(x)$	<code>asin(x), acos(x), atan(x)</code>

Compute the following:

(a)  $g(1)$

(b)  $g'(x)$

(c)  $g'(1)$

(d)  $\int_{-1}^1 g(x) dx$

Notice that when attempting to compute the definite integral, MATLAB gives an expression involving `erf`, the error function. This function is defined by  $\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ . This integral cannot be computed analytically, so use the `vpa` command to convert the answer to a decimal approximation. You can either use `vpa(ans)` after using the `int` command, or simply use `vpa(int(g, x, -1, 1))`.

**Activity A.0.10**

In this activity, we will re-visit the difference equation (1.1),  $M_{t+1} = 0.5M_t + 1$ . Suppose we have an initial quantity  $M_0 = 1$ . In the MATLAB command window, we could keep track of the quantity  $M_t$  using a variable  $m$ . First, set it to the initial value of 1:

```
m = 1
```

To compute the next quantity and update  $m$ , we would enter:

```
m = 0.5*m + 1
```

The quantity on the right,  $0.5m + 1$ , is computed and the result stored back in the variable  $m$ , overriding the previous value of  $m$ . The workspace should show that  $m$  is now equal to 1.5. Now, repeatedly execute the statement by using the up-arrow key, followed by the enter key. You should see that  $m$  is approaching 2.

Next we will automate the process of computing terms in the sequence  $m_{t+1} = 0.5m_t + 1$ , as an alternative to repeatedly hitting up-arrow and enter in the command line. We can do this by creating a `for` loop inside of a script file.

Open a new MATLAB script by using the New Script button in the MATLAB Home tab, then save the file using the File menu or (Ctrl+S).

Next enter the following code into the file:

```
1 m = 1;           %initialize m
2 N = 10;         %initialize number of iterations
3 for i=1:N
4     m = 0.5*m + 1 %update and display m
5 end
```

In the code segment above, **comments** are the text segments to the right of the `%` symbols. Generally speaking, comments are used to explain a line or a segment of code. MATLAB will ignore anything to the right of the `%` symbol and jump to the following line. In the code editor window, comments show up in green.

The lines inside the `for` loop should be automatically indented to make it easier to see which lines are part of the loop. In this case, only line 4 is run for each value of  $i$  from 1 to  $N$ .

Notice the semicolons at the end of lines 1 and 2. In MATLAB, any time you make an assignment statement, the new value is output to the command window, unless a semicolon appears at the end of the statement. Notice that lines 1 and 2 have semicolons at the end, so these numbers are not output to the command window. Line 4 does not have a semicolon - this means that all of the values of  $m$  will be output to the window at each iteration in the `for` loop. This can be a helpful tool in debugging (finding errors in) your code. Each time a value from your script file is output to your command window, you have an opportunity to see if it matches what you expect. If it does not, you know where to look to find errors in your code or in your understanding of the problem.

Save any new changes you've made to the MATLAB file before running the code. You can run all lines in the script by clicking the green Run arrow or pressing F5. To run a single line or part of a script, highlight the code segment and press F9.

Run the code several times, experimenting with different values of  $N$  and  $m$ . For example, if  $m_0 = 3$ , does this affect what the sequence converges to?

**Activity A.0.11**

Consider the recursively defined sequence

$$x_0 = 2, x_1 = 3, \quad x_{n+1} = 3x_n - 2x_{n-1} \text{ for } n \geq 1. \quad (\text{A.2})$$

The next couple values of the sequence are,

$$x_2 = 3x_1 - 2x_0 = 3(3) - 2(2) = 5, \quad x_3 = 3x_2 - 2x_1 = 3(5) - 2(3) = 9.$$

We will use a `for` loop to generate values up to  $x_{20}$  of this sequence.

Open a blank MATLAB script and enter the following code:

```

1  xold = 2;
2  xnew = 3;
3  N = 20;
4  for n = 2:N
5      xtemp = xnew;
6      xnew = 3*xnew - 2*xold
7      xold = xtemp;
8  end

```

Since the sequence references the previous two values, we need at least two variables to keep track of during the process - the most current value `xnew` and the previous value `xold`. There are 3 assignment statements inside the `for` loop. Think of `xnew` as  $x_{n-1}$  and `xold` as  $x_{n-2}$  at this point. On line 6 we apply A.2 to set `xnew` equal to  $x_n$ . However we will need the previous value of `xnew` representing  $x_{n-1}$ , since this will become  $x_{n-2}$  in the next iteration of the `for` loop. This is why we need to save it to a temporary variable in line 5 and then assign it to `xold` on line 7. There are certainly other strategies that would work and the code above is just one example of how to generate the sequence.

Save the m-file (Ctrl+S). To run the code, press F5. Run the code several times, experimenting with different values of  $N$ . Notice that there are semicolons at the end of lines 5 and 7, but not line 6 - this way the new values `xnew` are displayed to the command window at each iteration.

Another strategy would be to vectorize the code, meaning to keep track of all values of the sequence as a vector. This will probably make the code simpler to write since we don't have to worry about using a "temp" variable within the `for` loop.

First, we need to start a new blank MATLAB script. Define  $N$  as before, setting it equal to 20. Then define a vector of size 20 by using `x = zeros(N+1, 1)`. This creates a vector of size 21, and it will initially contain all zeros. You could make a vector of size 20 but remember that everything is "off by one" since you cannot use 0 as an index in MATLAB - so  $x_0$  will have to go in the `x(1)` position,  $x_1$  will be `x(2)`, and  $x_N$  will be `x(N+1)`. Initialize the first two values of the sequence using `x(1) = 2` and `x(2) = 3`. Think about where the `for` loop would start and end. Inside the `for` loop, you should only need one line to set the next value of `x` in terms of the previous two values. Run the code and make sure that you are getting the same values as you did previously - you may need to make an adjustment to where the loop begins and ends.

**Exercise A.0.12**

Write a `for` loop used to compute the product

$$\prod_{k=1}^7 \frac{k^2}{2k+3}$$



**Exercise A.0.13**

Given the following code segment, determine values of  $i$  and  $s$  at each iteration of the loop, completing the trace table below.

```

i = 1;
s = 0;
while i <=3
    s = s + i;
    i = i + 1;
end

```

$i$				
$s$				
$i$				

**Activity A.0.14**

Referring back to [Activity A.0.10](#), suppose we want to write a file that computes solutions to  $M_{t+1} = 0.5M_t + 1$  and continues until the iterates are not changing that much. In other words, we want to continue the loop and stop once the difference  $|m_{t+1} - m_t|$  is small. To implement this, we should keep track of not just the current iterate  $m_t$  but also the previous iterate  $m_{t-1}$ . Another option is to keep track of all iterates, but for now let's just keep track of the most recent two.

For the code segment below, `tol` is the “tolerance”, which indicates when we should stop running the loop. In this case, it is set to  $10^{-3}$ . We let `m_old` be the previous iterate and `m_new` be the current iterate. At some point in the while loop, we do need to replace `m_old` with the new value `m_new`. Notice that `i` is an optional variable that is used to keep track of the number of times that the loop is run.

```

tol = 1e-3;           %set tolerance
i = 0;               %i = number of iterations
m_old = 1;          %initialize m
m_new = 0.5*m_old + 1; %compute next m
while abs(m_new - m_old) > tol %loop until terms are not changing by much
    m_old = m_new;    %update m_old
    m_new = 0.5*m_old + 1; %update m_new
    i = i + 1;       %add 1 to the number of iterations
end
i                    %display the number of iterations used
m_new                %display the final iterate

```

Experiment with different values of `tol` to see how this affects the final iterate  $m_t$  as well as the number of times that the loop is run.

**Exercise A.0.15**

Given the following code segment, determine values of  $a$ ,  $b$ ,  $c$  at each iteration of the `while` loop, completing the trace table below.

```

a = 2;
b = 3;
c = 20;
while a <= b && c > a
    a = a+b;
    b = a+5;
end

```

$a$				
$b$				
$c$				

**Exercise A.0.16**

Consider the code segment:

```

if x > 2
    y = x + 1;
else
    x = x - 1;
    y = 5;
end

```

Determine the value of  $x$  and  $y$  after executing the statement if:

(a)  $x = 4$

(b)  $x = 2$

**Exercise A.0.17**

The command `rem(a,b)` gives the remainder obtained if  $a$  is divided by  $b$ . For example,  $\text{rem}(14,3)=2$ , since  $14 = 3 \cdot 4 + 2$ .

(a) Write a program using `for` loops, `while` loops, and/or conditional statements to list the integers that are divisible by both 3 and 8 in the set of whole numbers 1 to 1000.

(b) Write a program using `for` loops, `while` loops, and/or conditional statements to list the integers that are divisible by either 3 or 8 (or both) in the set of whole numbers 1 to 1000.

## Chapter 3 Iteration

### Activity A.0.18

Find  $a$  and  $b$  that allows you to use Theorem 3.1.4, and show that  $f(x) = \cos(x)$  has a unique fixed point in  $[a, b]$ . Then write a MATLAB program to find it - if you are stuck refer to Example 3.1.2.

Experiment with different values of TOL, and how it affects the number of iterations. Fill in the following table.

TOL	$10^{-2}$	$10^{-3}$	$10^{-4}$	$10^{-5}$	$10^{-6}$
num.iterations					

How does the number of iterations,  $n$ , increase as TOL decreases? What function could be used to model this relationship?

### Exercise A.0.19

How could one use a fixed-point iteration to approximate a root of  $f(x) = e^x - 2x^2$ ?

We would like to solve  $e^x - 2x^2 = 0$ , but the fixed-point iteration is used to solve an equation in the form  $g(x) = x$ .

(a) One option would be to add  $x$  to both sides. What is the resulting  $g$ ?

(b) Another option would be to solve for one of the  $x$ 's. What is the resulting  $g$ ?

**Exercise A.0.20**

Consider the problem of finding roots of  $f(x) = x^3 + 4x^2 - 10$ , in the interval  $[1, 2]$ . Since  $f(1)$  and  $f(2)$  have opposite signs and  $f$  is continuous, there must be a root in the interval  $[1, 2]$ . To approximate the solution using a fixed-point iteration, there are many ways to change this equation to the form  $x = g(x)$  by simple algebraic manipulation.

For each function  $g_i$  below, show that a fixed point of  $g_i$  must be a root of  $f$ . That is, show that  $f(x^*) = 0$  whenever  $g_i(x^*) = x^*$ . Then, find  $g'_i(x)$  on the interval  $[1, 2]$ . With the help of a plot, determine if there is a constant  $k$  with  $|g'_i(x)| < k$  on an interval around the fixed point. Recall Theorem 3.1.5 requires several conditions to be satisfied in order to guarantee convergence to a fixed point. For each function, determine which of these conditions are met and which are not.

Finally, run your FixedPoint.m code for each function  $g_i(x)$  to determine the fixed points, if they exist. Use  $x_0 = 1.5$  as your initial guess. If a fixed point exists, how many iterations does it take?

(a)  $x = g_1(x) = x - x^3 - 4x^2 + 10$

(b)  $x = g_2(x) = \left(\frac{10}{4+x}\right)^{\frac{1}{2}}$

(c)  $x = g_3(x) = (10/x - 4x)^{1/2}$

(d)  $x = g_4(x) = \frac{1}{2}(10 - x^3)^{\frac{1}{2}}$

(e)  $x = g_5(x) = x - \frac{x^3 + 4x^2 - 10}{3x^2 + 8x}$

**Exercise A.0.21**

For the function  $f(x) = x^3 + 4x^2 - 10$ , apply the Bisection method to approximate the solution to  $f(x) = 0$  in the interval  $[1, 2]$ .

Complete the first four iterations, giving results in the table below.

$a$	$c$	$b$	$f(a)$	$f(c)$	$f(b)$

You should find that the iterates are 1.5, 1.25, 1.375, 1.3125. Notice that  $f(1.375)$  is closer to zero than  $f(1.3125)$ , so  $|f(x)|$  is not certain to decrease with each iteration. It turns out that the root is closer to 1.365, which one may verify by performing more iterations.

**Exercise A.0.22**

Use Theorem 3.3.1 to find the number of iterations necessary to solve  $f(x) = x^3 + 4x^2 - 10 = 0$  with an accuracy of  $10^{-3}$  using  $a = 1$  and  $b = 2$ .

**Exercise A.0.23**

Show that the sequence  $x_n = \frac{1}{n}$ ,  $n \geq 1$  converges linearly to 0.

**Exercise A.0.24**

Using  $f(x) = \cos x - x$ ,  $x_0 = 1$ , use Newton's method to find  $x_1$  and  $x_2$  by hand.

**Exercise A.0.25**

Consider solving  $f(x) = (x - 1)^3 + 0.512 = 0$ . Does this problem satisfy the hypotheses of Theorem 3.4.1? Applying Newton's Method with an initial guess of 3, what happens? Sketch a picture, showing the first several iterations with tangent lines to help see this.

**Exercise A.0.26**

For  $f(x) = \cos x - x$ , determine the formula for the secant formula and use it to find  $x_2$  and  $x_3$  with  $x_0 = 0.5, x_1 = 0.7$ .

## Chapter 4 Matrices

**Exercise A.0.27**

Let

$$A = \begin{bmatrix} 1 & 2 & 0 \\ -2 & -3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 0 & 2 \\ 2 & -5 & 1 \end{bmatrix}.$$

Using MATLAB, define matrices  $A$  and  $B$  and compute:

- (a)  $A + B$
- (b)  $A - B$
- (c)  $2A$
- (d)  $2A - 3B$

**Exercise A.0.28**

Solve the following systems of equations by using the `rref()` command in MATLAB:

(a)

$$\begin{aligned}x + 2y + z &= 9 \\2x + 5y + 4z &= 8 \\-3x - 6y - 2z &= -35\end{aligned}$$

(b)

$$\begin{aligned}-x_1 - x_3 - 2x_4 &= -2 \\x_1 + x_2 + x_3 + 4x_4 &= 3 \\-2x_1 - 3x_2 - 11x_4 &= -6\end{aligned}$$

**Exercise A.0.29**

Compute the matrix expression, both by hand and by using MATLAB:

$$\begin{bmatrix} 8 & 3 & -4 \\ 5 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

**Exercise A.0.30**

Write the following matrix equation as a system of linear equations:

$$\begin{bmatrix} 7 & -3 \\ 2 & 1 \\ 9 & -6 \\ -3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -9 \\ 12 \\ -4 \end{bmatrix}$$

**Exercise A.0.31**

Let  $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

- (a) Use the command `A^2` in MATLAB to compute  $A^2$ .
- (b) Use the command `A.^2` in MATLAB to square each of the nine elements.

**Exercise A.0.32**

Use MATLAB to solve  $A\mathbf{x} = \mathbf{b}$ , with  $A = \begin{bmatrix} 1 & 2 & 1 \\ -3 & -1 & 2 \\ 0 & 5 & 3 \end{bmatrix}$ ,  $\mathbf{b} = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$ .

- (a) Use the command `inv(A)` to compute the inverse of  $A$ , then verify  $AA^{-1} = A^{-1}A$ .
- (b) Use the command `inv(A)*b` to compute the solution by computing the inverse.
- (c) Use the command `A \ b` to compute the solution by using Gaussian elimination.



**Exercise A.0.33**

Use MATLAB to find the eigenvectors and eigenvalues for the following matrices.

$$(a) B = \begin{bmatrix} 0 & 1 & -1 \\ 1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

$$(b) C = \begin{bmatrix} 4 & 0 & 1 \\ 2 & 1 & 0 \\ 2 & 2 & 3 \end{bmatrix}$$

**Activity A.0.34**

Consider the  $2 \times 2$  matrix,  $A = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$ . Save the matrix to variable  $A$  in MATLAB, using the command `A = [5 3; 3 5]`. From our earlier discussion, the eigenvalues must satisfy  $\det(A - \lambda I) = 0$ . To generate the  $2 \times 2$  identity matrix in MATLAB, use the command `eye(2)`. Define  $x$  to be a symbolic variable (using `syms x`). To create the matrix  $A - xI$  in MATLAB, use the command `A - x*eye(2)`. To find the characteristic equation, we use the command `d = det(A - x*eye(2))`. We want to solve the characteristic equation  $\det(A - xI) = 0$  for  $x$ , which may be done using the command `solve(d)`. MATLAB should output the two eigenvalues 8 and 2. *If you do not have the symbolic toolbox, you can solve for these eigenvalues by hand.*

Remember that by Remark 4.2, the eigenvector  $\mathbf{v}$  associated with the eigenvalue  $\lambda$  satisfies  $(A - \lambda I)\mathbf{v} = \mathbf{0}$ . We chose  $\lambda$  so that the matrix  $A - \lambda I$  was not invertible. However, we may solve for  $\mathbf{v}$  by row reducing the augmented matrix  $[A - \lambda I | \mathbf{0}]$ . We need to do this twice, for both eigenvalues 8 and 2.

First, to find the eigenvector associated with  $\lambda_1 = 8$ , we note that

$$A - 8I = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix} - 8 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -3 & 3 \\ 3 & -3 \end{bmatrix}.$$

To solve  $(A - 8I)\mathbf{v} = \mathbf{0}$ , we row reduce the augmented matrix

$$[A - 8I | \mathbf{0}] = \left[ \begin{array}{cc|c} -3 & 3 & 0 \\ 3 & -3 & 0 \end{array} \right]$$

However, this augmented column of zeros on the right never changes on performing any row operation. So we could in fact ignore it when we do the row reduction, and simply remember that there is an augmented column of zeros. So we will row reduce the matrix  $A - 8I$  by using the command `rref(A - 8*eye(2))`, resulting in the matrix  $\begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix}$ . Remember, there is an augmented column of zeros, so the augmented

matrix is actually  $\begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ . From the first row of the matrix, we get the equation  $x_1 - x_2 = 0$ , or  $x_1 = x_2$ . We may write the eigenvector as

$$\mathbf{v}_1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_1 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad x_1 \in \mathbb{R}.$$

There are infinitely many solutions, in agreement with Proposition 4.2.3. That is, any multiple of the vector  $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$  is an eigenvector associated with  $\lambda_1 = 8$ .

Repeat the steps above to find the eigenvector  $\mathbf{v}_2$  associated with the eigenvalue  $\lambda_2 = 2$ .

### Exercise A.0.35

Find the eigenvectors and eigenvalues for the matrix from Activity A.0.34,  $A = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$ . Check to see that the MATLAB output agrees with the solutions that we found.

## Chapter 5 Discrete Models

### Activity A.0.36

Consider the population (in millions) of the U.S. obtained by the U.S. Census:

<b>1790</b>	<b>1800</b>	<b>1810</b>	<b>1820</b>	<b>1830</b>	<b>1840</b>	<b>1850</b>	<b>1860</b>	<b>1870</b>	<b>1880</b>	<b>1890</b>	<b>1900</b>
3.9	5.3	7.2	9.6	12.9	17.1	23.2	31.4	39.8	50.2	62.9	76.2
<b>1910</b>	<b>1920</b>	<b>1930</b>	<b>1940</b>	<b>1950</b>	<b>1960</b>	<b>1970</b>	<b>1980</b>	<b>1990</b>	<b>2000</b>	<b>2010</b>	<b>2020</b>
92.2	106.0	122.8	132.2	150.7	179.3	203.3	226.5	248.7	281.4	308.7	TBD

Table A.1: U.S. Population (millions), years 1790 - 2010

We may find the growth rate for the decade of 1790 - 1800 by taking  $\frac{5.3-3.9}{3.9} = 0.36$ , to get a growth rate of 36% for the 10-year period.

- Open the data file census.csv in Excel or create two columns of your own with the data above. Create a new column that gives the growth rate for each decade. If this rate is relatively consistent, we may want to model this data subset by using a constant growth rate model. Is this the case here? What is the approximate the growth rate  $r$  during the years 1790 to 1860?
- Now let  $r$  be the growth rate, and  $P_0$  the population in 1790. We may predict the population each decade during the time period 1790 - 1860 using the Malthusian model  $P_{n+1} = (1+r)P_n$ . Find  $P_1$ ,  $P_2$ , and  $P_3$  in terms of  $r$  and  $P_0$ . Use this to deduce a formula for  $P_n$  for the  $n^{\text{th}}$  decade since 1790.

(c) Notice that the formula you derived above gives the population  $P_n$  for the  $n^{\text{th}}$  decade since 1790. If we interpolate between decades and assume the same annual growth rate, we can expand our formula to include any year  $t$  between 1790 -1860. Let  $N(t)$  be the population at year  $t$ , for example  $N(1790) = 3.9$ , and re-write your formula to give the population  $N(t)$  for any year  $1790 \leq t \leq 1860$ .

(d) We will now compare the actual data with that predicted by the Malthusian model,  $N(t)$ . In the Excel data sheet, add a column labeled "Prediction". In this column, enter the predicted population using your model for  $N(t)$  for each decade. Plot both the actual and predicted population against time for the years 1790 to 1860. Add another column to compute the percent error between the actual and predicted population values (i.e.  $\frac{\text{Actual}-\text{Predicted}}{\text{Actual}}$ ). How well does the model fit the data?

(e) We have designed the model above to represent population growth from 1790 - 1860 because the growth rate appeared to be relatively constant between those years. You may have noticed the growth rates become significantly lower after 1860 so an adjustment to the model seems appropriate if we are to include all years 1790 - 2010. To take into account a changing rate, consider the next level of complication and assume a *linear* growth rate  $r_n = a + bn$ . Let the population  $Q_n$  in the  $n^{\text{th}}$  decade since 1790 in this new model be given by  $Q_{n+1} = (1 + r_n)Q_n$ , where  $r_n = a + bn$ .

If the growth rate appears to be linear in a scatter plot, we can use linear regression to estimate the parameters  $a$  and  $b$ . Plotting the growth rate  $r_n$  against the decade  $n$  should give you the linear regression line  $r_n = 0.376 - 0.0145n$ . This gives us  $Q_{n+1} = (1 + 0.376 - 0.0145n)Q_n$ . Make a new column in your Excel file that gives the predicted population  $Q_n$  for each decade from 1790 - 2010. Plot both the actual and predicted population against time for  $1790 \leq n \leq 2010$ . Finally, add another column to compute the percent error between the actual and predicted population values.

(f) If we continue to use the model  $Q_{n+1} = (1 + 0.376 - 0.0145n)Q_n$ , will the population ever stop growing? If so, when?

**Activity A.0.37**

Suppose owl and wood rat populations at week  $k$  are modeled according to

$$\begin{aligned}L_{k+1} &= 0.5L_k + 0.4R_k \\R_{k+1} &= -0.104L_k + 1.1R_k\end{aligned}$$

where  $L_k$  is the number of owls in the region studied, and  $R_k$  is the number of thousands of rats in the area.

(a) Write this system of equations in the form of a matrix-vector equation  $\mathbf{x}_{k+1} = A\mathbf{x}_k$ .

(b) Find the eigenvalues and eigenvectors of  $A$ .

(c) Use Proposition 5.2.2 to determine the long term behavior of the owl and rat populations.

**Exercise A.0.38**

Classify the origin as an attractor, repeller, or a saddle point for the dynamical system  $\mathbf{x}_{k+1} = A\mathbf{x}_k$ . Plot several trajectories.

(a)  $A = \begin{bmatrix} 2/3 & 2/15 \\ 4/15 & 8/15 \end{bmatrix}, \mathbf{x}_0 = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$

(b)  $A = \begin{bmatrix} 0.8 & 0.5 \\ -0.1 & 1 \end{bmatrix}, \mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

**Exercise A.0.39**

Consider a population of size  $x$  that grows a factor of  $2/(1+.001x)$  at each time step. The dynamical system is then

$$x_{t+1} = \frac{2x_t}{1 + 0.001x_t}.$$

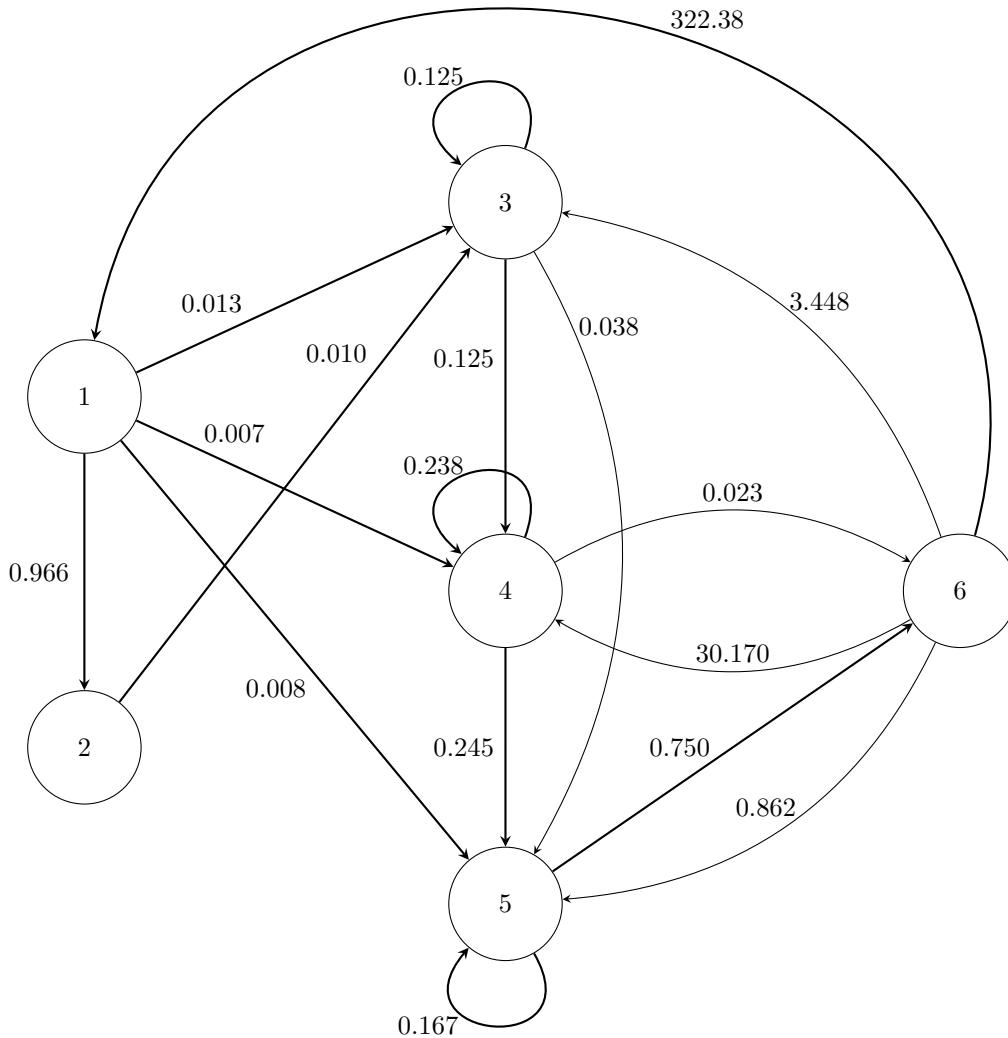
Find the equilibria, and experiment with different initial conditions to determine whether each of them is stable or unstable.

**Exercise A.0.40**

Use Theorem 5.5.2 to determine whether each equilibrium in [Exercise A.0.39](#) is stable or unstable.

**Activity A.0.41**

(Adapted from [5]) For this activity, we consider a monocarpic perennial plant called the **teasel**, *Dipsacus sylvestris*. This type of plant will only flower once (monocarpic) in its life though it will come back year after year (perennial) until it does so. The plant then dies after it has flowered and set its seeds. Keeping this in mind, we know the teasel has up to six stages of development: first-year dormant seeds (1), second-year dormant seeds (2), small rosettes (a round grouping of leaves) (3), medium rosettes (4), large rosettes (5), and flowering plants (6). The state diagram below shows the proportion of plants in each stage that progress to each additional stage as well as the average number of new plants or rosettes contributed per flowering plant.



- Determine a transition matrix for the teasel. Does it represent a Leslie matrix? Why or why not?
- Determine the long term distribution of plant stages. Do your results match the model assumptions?
- Does the population of teasel plants continue to grow or eventually die out?





## Chapter 6 Continuous Models

### Exercise A.0.43

Derive the discrete model  $P_{n+1} = P_n + rP_n = (1 + r)P_n$  from the continuous model  $\frac{dN}{dt} = kN$  by using a derivative approximation formula. How does the continuous parameter  $k$  relate to the discrete parameter  $r$ ?

### Exercise A.0.44

Verify that the solution to (6.1) is  $N(t) = N_0e^{kt}$ , both by checking that it satisfies (6.1) and by using separation of variables.

What happens to  $N(t)$  as  $t \rightarrow \infty$ ?

### Exercise A.0.45

The logistic model in Equation 6.2 is restated here

$$\frac{dN}{dt} = rN \left( 1 - \frac{N}{K} \right), N(0) = N_0. \quad (\text{A.3})$$

Here,  $K$  is the carrying capacity, and  $N(t)$  is the population at time  $t$ .

(a) What happens to the rate of change if the ratio  $N/K$  is close to zero?

(b) When the ratio  $N/K$  is close to one?

**Exercise A.0.46**

Suppose a drug is administered to a patient intravenously for 12 hours continuously, and then stopped. We assume the body begins to process or metabolize the drug at a rate proportional to the amount of drug immediately upon injection and continues until there is no drug left in the body. Once the drug is metabolized (usually in the liver) it is no longer present in its original form. Assume there is no drug present in the body initially. We wish to determine the amount of drug  $A(t)$  present in the body at any time  $t$  after administration begins.

- (a) First let  $D(t)$  be the drug administration rate and let  $P(A)$  be the drug processing (or metabolizing) rate. Construct a compartment diagram with these rates.
- (b) Now suppose the drug is administered at a constant rate of 10 mg/hr for the first 12 hours and is metabolized with a 10% constant of proportionality, that is  $P(A) = 0.1A$ . Define  $D(t)$  and  $P(A)$  for any time  $t \geq 0$  and determine the initial value problem for  $A(t)$ .
- (c) Determine the intermediary conditions for  $t = 12$  and solve for an explicit solution of  $A(t)$  at any time  $t \geq 0$ . You should have a piecewise function.

- (d) Plot the amount of drug in the body versus time. Use calculus to verify the concavity of your graph. What happens to the amount of the drug in the body as time gets large?
- (e) As with any model, it's again useful to think of any potential ethical considerations or unintended consequences the results of the model may provide. What questions should one consider as the doctor? What questions should one consider as the patient?
- (f) As an extension of this exercise, consider a new model that would allow for additional drugs to be administered once the amount in the body falls below a certain threshold. How would your differential equation change? Initial and intermediate conditions?

**Activity A.0.47**

Consider the initial value problem  $\frac{dx}{dt} = t(x - 1)$ ,  $x(0) = 2$ .

(a) Verify that the exact solution is  $x(t) = e^{\frac{1}{2}t^2} + 1$

(b) Apply Euler's method by hand with  $h = 0.1$  to approximate  $x(0.5)$  by completing the table below.

$i$	$t$	$x_i$	Actual Value, $x = e^{\frac{1}{2}t^2} + 1$	Error
0	0	2	2	0
1	0.1			
2	0.2			
3	0.3			
4	0.4			
5	0.5			

**Exercise A.0.48**

Recall the logistic growth model,

$$\frac{dN}{dt} = rN \left( 1 - \frac{N}{K} \right), \quad N(0) = N_0,$$

where  $r, K > 0$ . Draw the phase line diagram for this problem. Find the equilibria and classify these as stable or unstable.

**Activity A.0.49**

Refer back to Example 6.1.5. Consider a population with natural logistic growth and a linear harvest rate proportional to the population size  $x$ .

$$\frac{dx}{dt} = rx \left(1 - \frac{x}{K}\right) - hx$$

Perform a phase line analysis for this population using low,  $h_1$ , and high,  $h_2$ , harvest rates for the line  $H(x) = hx$ . Specifically, let your choice of low and high harvest rates be such that  $G(h_1) = G(h_2)$ . Is a high or low harvest rate preferable? Explain. To answer this question, you may choose a species for which you know something about their harvest methods (or can research online), such as for deer, salmon, trees, etc.

**Activity A.0.50**

The following is a Lotka-Volterra model that models the interaction between hares and foxes. Let  $H(t)$  be the number of hares, in hundreds, and  $F(t)$  be the number of foxes (in tens) at time  $t$ . This model includes the term  $-a_3H(t)^2$ , which takes into account competition among the prey. When  $H$  is small, the birth rate term  $a_1H$  cause growth to be roughly exponential in the absence of predators. However, when  $H$  is large, the term  $-a_3H^2$  dominates, causing the rate of change to eventually decrease.

$$\begin{aligned}H'(t) &= a_1H(t) - a_2H(t)F(t) - a_3H(t)^2 \\F'(t) &= -b_1F(t) + b_2H(t)F(t),\end{aligned}$$

- (a) Use  $a_1 = 3, a_2 = 1, a_3 = 1, b_1 = 0.5,$  and  $b_2 = 1$  as parameters in the model. Starting with 200 hares and 10 foxes, use Euler's method to approximate  $H(t)$  and  $F(t)$  on the interval  $0 \leq t \leq 20$  using Euler's method. Use a step size of  $h = 0.1$ . Plot  $H(t)$  and  $F(t)$  against  $t$ . Then make a plot showing the trajectories with  $H$  on the horizontal axis and  $F$  on the vertical axis.
- (b) Estimate what happens to the population of hares and foxes as  $t$  gets large.

**Exercise A.0.51**

Reconsider the predator prey model from **Activity A.0.50** :

$$\begin{aligned}H'(t) &= 3H - FH - H^2 \\F'(t) &= -0.5F + HF\end{aligned}$$

Plot the phase plane diagram for this problem.

**Exercise A.0.52**

Consider the initial value problem

$$\begin{aligned}x_1'(t) &= -1.5x_1 + 0.5x_2 + 3, & x_1(0) &= 5 \\x_2'(t) &= x_1 - x_2 + 1, & x_2(0) &= 4.\end{aligned}$$

Find the equilibrium, and use Theorem 6.4.2 to solve the system. Then use Theorem 6.4.7 to determine whether the equilibrium is stable.

**Activity A.0.53**

In Activity A.0.50 we made a phase plane diagram for the predator-prey system,

$$\begin{aligned}H'(t) &= 3H(t) - H(t)F(t) - H(t)^2 \\F'(t) &= -0.5F(t) + H(t)F(t),\end{aligned}$$

and found that the equilibria are  $(0, 0)$ ,  $(3, 0)$ , and  $(0.5, 2.5)$ . Use the Jacobian matrix to determine which of these points are stable. The results should agree with the phase plane diagram, as well as solution plots obtained from using Euler's method.

## Chapter 7 Stochastic Models

### Activity A.0.54

(A Stochastic Population Model, [17]) Consider the model of a population in which the rate of increase varies with some known distribution. Assume that the population at time step  $k$  is modeled using

$$P_{k+1} = P_k(1 + r_k), \quad (\text{A.4})$$

where  $r_k$  is a discrete random variable with probability mass function given in the table below.

Value	Probability
-0.01	0.3
0.00	0.1
0.01	0.4
0.02	0.2

As there are 4 possibilities for  $r_k$  at each time step, there are 16 possible computations of  $r_k$  for two time steps, 64 possibilities for three time steps, and  $4^k$  possibilities for  $k$  time steps. Therefore, finding an analytic solution to (A.4) is difficult to do for more than a few time steps.

Suppose that  $P_0 = 100$ . Use simulation to approximate the distribution of values for  $P_{20}$ , then make a histogram of those values. Experiment with the number of simulations.

The code will look something like this:

```

Initialize T:=20 to be the number of time steps
Initialize numSims to be the number of simulations
Initialize finalPop to be a vector of size numSims
For k = 1:numSims
    Initialize P to be a vector of size T
    Set P(1):=100
    For i=1 to T
        Pick a random number in  $U \in [0, 1]$ 
        Set r to -0.01, 0, .01, or .02 depending on the value of U
        P(i+1) := P(i)*(1+r)
    End For
    Set finalPop(k):=P(T+1)
End For
Make a histogram of finalPop

```

What are the mean and standard deviation of `finalPop`?

### Exercise A.0.55

Suppose a fair coin is tossed until a head comes up. Let  $X$  be the trial on which the coin first comes up heads.

- Give the sample space of  $X$ .
- Find the probability mass function of  $X$ .
- Find the probability that  $X$  is odd.



- (d) Find  $E(X)$ .
- (e) Write a MATLAB simulation code to verify your results for parts (b) - (d).

**Activity A.0.56**

In roulette, betting on 00 has a 35 to 1 payout. Since an American roulette wheel has 38 possible slots, there is a  $1/38$  chance of winning. The 35 to 1 payout means that for a \$1.00 bet, you would win \$35 if it lands on 00 and lose your \$1 otherwise. Let  $X$  be the amount gained after betting \$1.00 on 00.

- (a) Find the probability mass function of  $X$ .
- (b) Find the expected value and standard deviation of  $X$ .
- (c) Write a code in MATLAB to simulate making many \$1.00 bets on 00. The code will look something like this:

```

Initialize numSims
Initialize X to be a vector of outcomes (winnings)
For i = 1 to numSims
    Pick a random number in  $p \in [0, 1]$ 
    If  $p < 1/38$ 
         $X(i) = 35$ 
    Else
         $X(i) = -1$ 
    End If
End For
Calculate mean of winnings
Calculate standard deviation of winnings
Calculate proportion of values of winnings that are  $> 0$ 

```

- (d) Use your code to approximate the probability of winning money overall (having a positive amount) after playing  $n$  times.
- (e) Using the Central Limit Theorem, what are the mean and standard deviation of  $\bar{x}_n$ ? (Note  $n = 1$  in this case.) Then fill in the following table.

$n$	10	20	40	80	160
$E(\bar{x}_n)$					
Expected value from simulation					
$\text{Var}(\bar{x}_n)$					
Variance from simulation					
$P(\bar{x}_n > 0)$					
Probability of winning money from simulation					

Compare your answers to the analytic solutions you found in part (b).

**Activity A.0.57**

This activity is an extension of [Activity A.0.56](#). In roulette, betting on 00 has a 35 to 1 payout. Since an American roulette wheel has 38 possible slots, there is a  $1/38$  chance of winning. The 35 to 1 payout means that for a \$1.00 bet, you would win \$35 if it lands on 00 and lose \$1 otherwise.

- (a) Suppose you place now place  $n$  \$1.00 bets on 00. Let  $\bar{x}_n$  be the average amount won or lost per game after the  $n$  bets have been made. Find the expected value and standard deviation of  $\bar{x}_n$  using the CLT.
- (b) Write a code in MATLAB to simulate making  $n$  \$1.00 bets on 00. Report the mean and standard deviation of the winnings for  $n$  bets. Again, use your simulation to estimate the probability of winning money (having a positive amount) overall after playing  $n$  times. The code will look something like this:

```

Initialize numSims
Initialize winnings to be a vector of numSims zeros
Initialize numGames
For k = 1:numSims
    For i = 1 to numGames
        Pick a random number in  $p \in [0, 1]$ 
        If  $p < 1/38$ 
            winnings(k) = winnings(k) + 35
        Else
            winnings(k) = winnings(k) - 1
        End If
    End For
End For
Calculate mean of winnings in numGames
Calculate standard deviation of winnings in numGames
Calculate proportion of values of winnings in numGames that are  $> 0$ 

```

- (c) Use your answers to (b) as well as your simulation code to complete the following table. Also, use your simulation to estimate the probability of winning money overall after playing  $n$  times.

$n$	10	20	40	80	160
$E(\bar{x}_n)$					
Expected value from simulation					
$\text{Var}(\bar{x}_n)$					
Variance from simulation					
$P(\bar{x}_n > 0)$					
Probability of winning money					

- (d) Find a 95% confidence interval for the amount won (or lost) if you play 160 times.

**Exercise A.0.58**

Write a MATLAB program using the Monte Carlo method to approximate  $\int_{-2}^1 e^{x^2} dx$ . Although there is no analytic solution to test against, we may use technology to show that the solution is about 17.915. Test your Monte Carlo code with the `sum` function below. Note it is possible to write the code without using an explicit loop.

```
N = 1000; %Initialize number of sample points
x = rand(N,1)*3-2; %Generate N random numbers in [-2, 1]
approxInt = sum(exp(x.^2))*3/N %compute approximate integral
```

**Exercise A.0.59**

Suppose you are taking a 10 question multiple choice test, each having 4 answers (A, B, C, D). You forgot to study so you are randomly guessing.

(a) What is the probability of getting exactly 2 answers correct?

(b) What is the probability of getting at least 3 answers correct?

**Activity A.0.60**

Using MATLAB's `randn` Command Suppose we want to take  $n$  random samples that are normally distributed with mean 0 and standard deviation 1. In other words, we want to generate random variables  $X_i \sim N(0, 1)$  for  $i = 1, \dots, n$ . We may use the command `randn` to generate a random number with mean 0 and standard deviation 1. Try this several times from the command prompt.

Note that in order to generate a random number with mean  $m$  and standard deviation  $s$  you would simply use the command `m+s*randn`. Try this several times to generate a random variable with mean 100 and standard deviation 15.

To generate several values with a normal distribution, you can avoid writing a for loop by using the command `randn(n,1)`, where  $n$  is the number of samples. Generate 200 random samples from the  $N(100, 15)$  distribution. Plot the results in a histogram. Verify that the histogram is roughly bell shaped, centered at 100, and that roughly 95% of the data falls within two standard deviations of the mean.

**Exercise A.0.61**

Suppose that lengths of human pregnancies are normally distributed with a mean of 268 days and a standard deviation of 15 days. Write a MATLAB file to generate 1000 random pregnancy lengths, then make a histogram of these to help visualize the distribution. Recall the command `hist(x,n)` to make a histogram of an array  $x$ , with  $n$  bins. Consider  $n \approx 20$  to start. Finally, use MATLAB's `int` command to approximate the probability that a pregnancy lasts longer than 290 days.

**Activity A.0.62**

In **Activity A.0.54** we modeled a population  $P_k$  at time step  $k$  using

$$P_{k+1} = P_k(1 + r_k),$$

where  $r_k$  is a discrete random variable with a given probability mass function. Suppose this rate  $r_k$  is instead a normal random variable with mean 0.01 and standard deviation 0.008.

Suppose that  $P_0 = 100$ . Use simulation to approximate the distribution for  $P_{20}$ , then make a histogram of its values.

The code will look something like this:

```

Initialize T:=20 to be the number of time steps
Initialize numSims to be the number of simulations
Initialize finalPop to be a vector of size numSims
For k = 1:numSims
    Initialize P to be a vector of size T
    Set r to be a vector of T normal random variables with mean 0.01 and standard deviation 0.008.
    Set P(1):=100
    For i=1 to T
        P(i+1) := P(i)*(1+r(i))
    End For
    Set finalPop(k):=P(T+1)
End For
Make a histogram of finalPop

```

What are the mean and standard deviation of **finalPop**?

**Activity A.0.63**

Suppose that the time spent waiting for a car to pass on a highway is distributed according to the exponential distribution with an average time of 30 seconds.

- After a car passes, what is the probability of waiting more than 45 seconds for the next car to pass?
- Suppose you have waited a minute without a car passing by. What is the probability of having to wait an additional 45 seconds before another car passes by?
- What is the average number of cars that will pass by during a 5 minute period?
- What is the probability that during a 5 minute period, there will be at least 8 cars that pass by?

**Activity A.0.64**

Continuing from **ActivityA.0.63**, suppose that cars pass by on a highway every 30 seconds on average. Write a MATLAB code to simulate waiting for 200 cars to pass by, and recording how much time passes between successive cars. Then modify your code segment to simulate waiting for 10 minutes and recording how many cars pass by. Generate a density histogram for each simulation, and compare results to the theoretical distributions.

**Activity A.0.65**

Continuing from **ActivityA.0.63**, suppose that cars pass by on a highway every 30 seconds on average. Write a program to simulate the Poisson process for  $X(t)$ , where  $X$  is the number of cars that have passed by after  $t$  seconds. Simulate the process for a ten minute period of time. Plot  $X(t)$ , using 20 simulations. Add the expected value  $E(X(t))$  to the plot, using a slightly thicker line.

**Exercise A.0.66**

Consider the following functions:  $f(x) = x^2$ ,  $g(x) = x^2 + x$ ,  $h(x) = x^3$ ,  $k(x) = x^3 - 3x^2$ . Which functions are  $o(h)$ , and which are  $o(h^2)$ ? If two functions are both  $o(h)$ , what can we say about their sum and difference?

## Appendix B

# Solutions to in-class Exercises

### Chapter 2

**Exercise A.0.19:**

(a)  $g(x) = x + e^x - 2x^2$  (b)  $g(x) = \sqrt{\frac{e^x}{2}}$  or  $g(x) = \ln(2x^2)$

**Exercise A.0.21:**

$a$	$c$	$b$	$f(a)$	$f(b)$	$f(c)$
1	1.5	2	-5	2.375	14
1	1.25	1.5	-5	-1.80	2.375
1.25	1.375	1.5	-1.80	0.162	2.375
1.25	1.3125	1.375	-1.80	-0.848	0.162

**Exercise A.0.22:** Use Theorem 3.3.1 to find the number of iterations necessary to solve  $f(x) = x^3 + 4x^2 - 10 = 0$  with an accuracy of  $10^{-3}$  using  $a = 1$  and  $b = 2$ .

Here, we want to find the minimum  $n$  required in order to guarantee that  $|x_n - x| \leq 10^{-3}$ . We can combine this with Theorem 3.3.1 to state that  $|x_n - x| \leq (2 - 1)/2^n \leq 10^{-3}$ . Next, we use some algebra to solve  $(2 - 1)/2^n \leq 10^{-3}$  for  $n$ , which gives  $2^n \geq 10^3$ , or  $n \geq \log_2 10^3 \approx 9.97$ . So we need  $n \geq 10$ .

**Exercise A.0.23:** Show that the sequence  $x_n = \frac{1}{n}, n \geq 1$  converges linearly to 0.

Using Definition 3.3.2, we must show that  $\frac{|x_{n+1} - 0|}{|x_n - 0|} = \lambda$  for some constant  $\lambda$ . Indeed,

$$\lim_{n \rightarrow \infty} \frac{|x_{n+1} - 0|}{|x_n - 0|} = \lim_{n \rightarrow \infty} \frac{\frac{1}{n+1}}{\frac{1}{n}} = \lim_{n \rightarrow \infty} \frac{n}{n+1} = 1.$$

**Exercise A.0.24:** Using  $f(x) = \cos x - x, x_0 = 1$ , use Newton's method to find  $x_1$  and  $x_2$  by hand.

We need the derivative,  $f'(x) = -\sin x - 1$ . We get

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 1 - \frac{\cos 1 - 1}{-\sin 1 - 1} \approx 0.7504,$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 0.7504 - \frac{\cos(0.7504) - 0.7504}{-\sin(0.7504) - 1} \approx 0.7390.$$

**Exercise A.0.26:** For  $f(x) = \cos x - x$ , use the secant method with  $x_0 = 0.5, x_1 = 0.7$  to find  $x_2$  and  $x_3$ . Using the fact that  $f(x_0) = f(0.5) = \cos 0.5 - 0.5 \approx 0.3776$ , and  $f(x_1) = 0.7 = \cos 0.7 - 0.7 \approx 0.0648$ , we get the first iterate

$$x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} \approx 0.7 - \frac{0.0648 \cdot (0.7 - 0.5)}{0.0648 - 0.3776} \approx 0.7415.$$

Using the fact that  $f(x_2) = \cos 0.7415 - 0.7415 \approx -0.003876$ ,

$$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)} \approx 0.7415 - \frac{(-0.003876)(0.7415 - 0.7)}{-0.003876 - 0.0648} \approx 0.7391.$$

## Chapter 4

### Exercise A.0.28

(a)  $x = 5, y = 6, z = -8$

(b)  $x_1 = \frac{3}{2} - \frac{5}{2}x_4, x_2 = 1 - 2x_4, x_3 = \frac{1}{2} + \frac{1}{2}x_4, x_4$  is free.

### Exercise A.0.29

$$\begin{bmatrix} 2 \\ 13 \end{bmatrix}$$

### Exercise A.0.30

$$\begin{aligned} 7x_1 - 3x_2 &= 1 \\ 2x_1 + x_2 &= -9 \\ 9x_1 - 6x_2 &= 12 \\ -3x_1 + 2x_2 &= -4 \end{aligned}$$

### Exercise A.0.33

a) Using  $[x \ y] = \text{eig}([0 \ 1 \ -1; \ 1 \ 1 \ 0; \ -1 \ 0 \ 1])$  gives

$$\begin{aligned} x = \\ 0.8165 & \quad -0.0000 & \quad -0.5774 \\ -0.4082 & \quad 0.7071 & \quad -0.5774 \\ 0.4082 & \quad 0.7071 & \quad 0.5774 \end{aligned}$$

$$\begin{aligned} y = \\ -1.0000 & \quad 0 & \quad 0 \\ 0 & \quad 1.0000 & \quad 0 \\ 0 & \quad 0 & \quad 2.0000 \end{aligned}$$

The eigenvalues are  $\lambda_1 = -1, \lambda_2 = 1, \lambda_3 = 2$ . The corresponding eigenvectors  $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$  are given by the columns of  $\mathbf{x}$ .

b) Using  $[x \ y] = \text{eig}([4 \ 0 \ 1; \ 2 \ 1 \ 0; \ 2 \ 2 \ 3])$  gives

$$\begin{aligned} x = \\ 0.5906 & \quad -0.2619 - 0.0890i & \quad -0.2619 + 0.0890i \\ 0.2757 & \quad -0.3718 + 0.4353i & \quad -0.3718 - 0.4353i \\ 0.7584 & \quad 0.7718 & \quad 0.7718 \end{aligned}$$

$$\begin{aligned} y = \\ 5.2843 & \quad 0 & \quad 0 \\ 0 & \quad 1.3579 + 0.8975i & \quad 0 \\ 0 & \quad 0 & \quad 1.3579 - 0.8975i \end{aligned}$$

Note that two of the eigenvalues are complex, and their associated eigenvectors are also complex. The eigenvalues are 5.2843 and  $1.3579 \pm 0.8975i$ . The corresponding eigenvectors are again given by the three columns of  $\mathbf{x}$ .

**Exercise A.0.37** (a)  $A = \begin{bmatrix} 0.5 & 0.4 \\ -0.104 & 1.1 \end{bmatrix}$  and  $\mathbf{x}_k = \begin{bmatrix} L_k \\ R_k \end{bmatrix}$ .

(b)  $\lambda_1 = 0.58, \lambda_2 = 1.02, \mathbf{v}_1 = (5, 1)^T, \mathbf{v}_2 = (0.7692, 1)^T$ .

(c)

$$\mathbf{x}_k = c_1(0.58)^k \begin{bmatrix} 5 \\ 1 \end{bmatrix} + c_2(1.02)^k \begin{bmatrix} 0.7692 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} 5c_1(0.58)^k + 0.7692c_2(1.02)^k \\ c_1(0.58)^k + c_2(1.02)^k \end{bmatrix}$$

As  $k$  gets large, since  $(0.58)^k \rightarrow 0$ , we see that  $\mathbf{x}_k \rightarrow \begin{bmatrix} .7692c_2(1.02)^k \\ c_2(1.02)^k \end{bmatrix}$ .

The populations both increase, with an asymptotic growth rate of 2%. The ratio  $L_k/R_k$  approaches 0.769, meaning for every owl there are approximately 769 rats.

(d) From part(c), we saw that both populations increase without bound, so the trajectories approach infinity. Also, since  $L_k/R_k \rightarrow .769$ , the points  $(L_k, R_k)$  approach a straight line through the origin with slope  $1/.769 \approx 1.3$ , see Figure B.1.

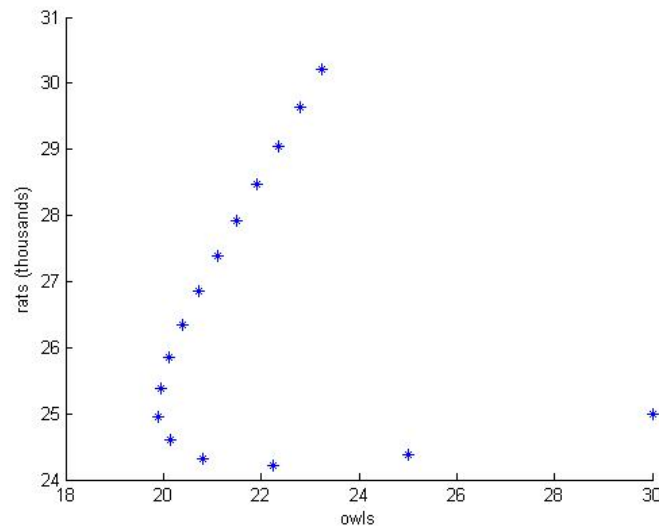


Figure B.1: Trajectories  $\mathbf{x}_k, k = 0, 1, \dots, 15$

**Exercise A.0.38** (a) The eigenvalues are  $\lambda_1 = 0.8, \lambda_2 = 0.4$ , with eigenvectors  $\mathbf{v}_1 = (1, 1)^T, \mathbf{v}_2 = (-1, 2)^T$ . Using Proposition 5.2.2, we have

$$\mathbf{x}_k = \begin{bmatrix} c_1(0.8)^k - c_2(0.4)^k \\ c_1(0.8)^k + 2c_2(0.4)^k \end{bmatrix}.$$

Since we also know that  $\mathbf{x}_0 = (1, 5)^T$ , we have

$$\begin{aligned} 1 &= c_1(0.8)^0 - c_2(0.4)^0 \\ 5 &= c_1(0.8)^0 + 2c_2(0.4)^0, \end{aligned}$$



and

$$1 = c_1 - c_2, 5 = c_1 + 2c_2.$$

Solving this linear system for  $c_1$  and  $c_2$  gives us  $c_1 = 8/3, c_2 = 4/3$ . Hence, the trajectories satisfy

$$\mathbf{x}_k = \begin{bmatrix} 8/3(0.8)^k - 4/3(0.4)^k \\ 8/3(0.8)^k + 8/3(0.4)^k \end{bmatrix} = \frac{4}{3} \begin{bmatrix} 2(0.8)^k - (0.4)^k \\ 2(0.8)^k + 2(0.4)^k \end{bmatrix}.$$

It is clear that  $\mathbf{x}_k \rightarrow (0, 0)^T$  as  $k \rightarrow \infty$  (see Figure B.2).

(b) The eigenvalues are  $\lambda = \pm 0.9 + 0.2i$ , with eigenvectors  $v = (0.9129, 1.826 \pm .3651i)^T$ . Note that the eigenvalues come in complex conjugate pairs - that is, they are in the form  $a \pm bi$ . They therefore have the same length,  $|\lambda| = \sqrt{.9^2 + .2^2} = .922$ . Since  $0.922 < 1$ , the origin is an attractor and solutions tend towards the origin. Also, it turns out that if the eigenvalues are complex, the solutions follow a spiral pattern (see Figure B.3).

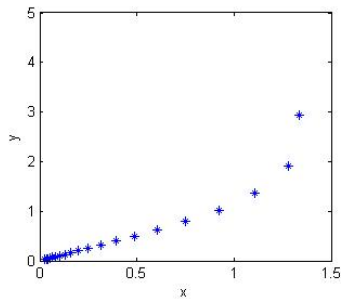


Figure B.2: (a) Trajectories  $\mathbf{x}_k$

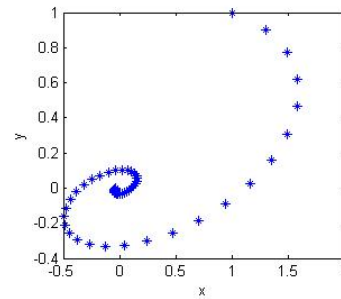


Figure B.3: (b) Trajectories  $\mathbf{x}_k$

## Chapter 6

**Activity A.0.53**  $J(H, F) = \begin{bmatrix} 3 - F - 2H & -H \\ F & -.5 + H \end{bmatrix}$ ;  $(0,0)$  and  $(3,0)$  are unstable;  $(0.5, 2.5)$  is stable.

## Chapter 7

### Exercise A.0.55

(b)  $P(X = k) = \frac{1}{2^k}, k = 1, 2, \dots$

Recall that the geometric series  $\sum_{i=1}^{\infty} r^n$  converges to  $r \frac{1}{1-r}$  in the case that  $|r| < 1$ .

We can then verify that the probabilities add to one,

$$\sum_{k=1}^{\infty} \left(\frac{1}{2}\right)^k = \frac{1}{2} \left(\frac{1}{1-\frac{1}{2}}\right) = \frac{1}{2} \cdot 2 = 1.$$

(c)

$$\begin{aligned} P(X \text{ is odd}) &= \frac{1}{2} + \frac{1}{2^3} + \frac{1}{2^5} + \dots = \frac{1}{2} \left(1 + \frac{1}{2^2} + \frac{1}{2^4} + \frac{1}{2^6} \dots\right) \\ &= \frac{1}{2} \left(1 + \frac{1}{4} + \frac{1}{4^2} + \frac{1}{4^3} + \dots\right) = \frac{1}{2} \left(\frac{1}{1-\frac{1}{4}}\right) = \frac{2}{3} \end{aligned}$$

(d)

$$E(X) = \sum_{k=1}^{\infty} k \cdot \frac{1}{2^k} = \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + \dots$$

Notice that

$$\frac{1}{2}E(X) = \frac{1}{4} + 2 \cdot \frac{1}{8} + 3 \cdot \frac{1}{16} + 4 \cdot \frac{1}{32} + \dots$$

We subtract these expressions in a way so that the first term in  $E(X)$  is left alone, but the second term of  $E(X)$  is grouped with the first term of  $\frac{1}{2}E(X)$ , the third term of  $E(X)$  is grouped with the second term of  $\frac{1}{2}E(X)$ , and so on.

$$\frac{1}{2}E(X) = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots = 1,$$

Hence  $E(X) = 2$ .

$E(Y) = \sum_{n=1}^{\infty} 2^n \left(\frac{1}{2^n}\right) = \sum_{n=1}^{\infty} 1$ , so the sum does not converge. Therefore,  $E(Y)$  is not defined.

**Exercise A.0.58:**

In the for loop, we need to choose random numbers in the interval  $[-2, 1]$ . To do this, we could simply use the fact that if  $x \in U(0, 1)$ , then  $(b-a)x + a \in U(a, b)$ .

```
sum=0;
a=-2;
b=1;
n=10000;
for i=1:n
x=(b-a)*rand+a;
sum = sum + exp(x^2);
end
A=sum*(b-a)/n
```

**Exercise A.0.59:** This is an example of a Binomial experiment: there are exactly two outcomes (correct or incorrect), and we may assume the probability of getting a question correct is  $1/2$ . The experiment is repeated 10 times, so the number of heads  $X$  has the Binomial distribution  $B(10, 1/4)$ . Using (7.10), we see that  $P(X = 2) = {}_{10}C_2 \cdot (1/4)^2(3/4)^8 \approx 0.2816$

**Activity A.0.63**

(a) 0.223, (b) 0.223, (c) 10, (d) 0.780

## Appendix C

# Selected Solutions

### Chapter 1 Introduction

10 (b)  $V(t) = 10,000e^{0.05t} + 15,000$

16. (c)  $A \approx 88.0$ , (d) About every 5.3 hours

### Chapter 2 MATLAB Programming

8

```
n=100;
x=linspace(0, 6, n);
f=3*exp(-.1*x.^2);
g=-2*cos(x./2);
plot(x,f,x,g)
hold on
plot([0, 6],[0, 0], 'k')
xlabel('x')
ylabel('y')
legend('f(x)', 'g(x)')
```

16.

```
x=1; %x is an array of Fibonacci numbers - we set x(1) = 1
x(2)=1; %set x(2) = 1
n=20; %pick some number
for i=3:n
    x(i)=x(i-1)+x(i-2);
end
x' %output all n Fibonacci numbers
```

17.

```
function Divisors(n)
div=0; %div is an array that contains all the divisors of n
k=1; %k keeps track of the number of divisors
for i=1:n
```

```

    if rem(n,i)==0 %if n is divisible by i
        div(k)=i; %set i to be a divisor
        k = k + 1; %increment k
    end
end
div' %output all divisors

```

21.  $N = 100000$

20. 1,835,421

25.

```

g = zeros(1000,1);
k = 0;
i = 2;
while k < 1000
    if isprime(i)==1
        k = k + 1;
    else
        g(k)=g(k)+1;
    end
    i = i + 1;
end
g

```

24. (b) 1.61803

## Chapter 3 Iteration

6. (a) 2.6906, (b) 1.8294 (Note that  $\frac{d}{dx}(e^x + 2^{-x} + 2 \cos x - 6) = e^x - \ln 2 \cdot 2^{-x} - 2 \sin x$ ). (c) 0.91, 3.7331

7. Hint: minimize the function  $d(x) = (x - 1)^2 + (x^2 - 0)^2 = x^4 + x^2 - 2x + 1$ . The minimum occurs at approximately (0.5898, 0.3479)

8. To obtain this solution, we checked every 0.1 units on the range  $[0, 4]$ . The iteration converges to -1.379 for  $x_0 \in \{-3.7, -3.2\} \cup [-3, -0.2] \cup 0.1 \cup [3.3, 3.6] \cup [3.8, 4]$  The iteration converges to 1.379 for  $x_0 \in [-4, -3.8] \cup -3.6, -3.3 \cup -0.1 \cup [0.2, 3] \cup 3.2, 3.7$  The iteration did not converge for  $x_0 = \pm 3.1, x_0 = 0$ .

9. -2.87939, -0.652704, 0.532089

15. (a) 0.64118, (b) 0.25753, (c) -2.19131, -0.79816

16. Using Theorem 3.3.1, we want  $|x_n - x^*| \leq (4 - 1)/2^n < 10^{-3}$ . Solving for  $n$  gives  $n > \log_2 3000 \approx 11.55$ . We need  $n \geq 12$ , and  $x_{12} \approx 1.378$ .

17. For the first part,

$$|f(x_n)| = |(x_n - 1)^{10}| = \left| \left( 1 + \frac{1}{n} - 1 \right)^{10} \right| \leq \frac{1}{2^{10}} < 10^{-3}.$$

Since  $1/n^{10}$  is a decreasing function, it holds that  $|f(x_n)| < 10^{-3}$  when  $n \geq 2$ . For the second part,

$$|1 - x_n| < 10^{-3} \Rightarrow |1 - 1 + 1/n| < 10^{-3} \Rightarrow \frac{1}{n} < 10^{-3} \Rightarrow n > 1000.$$

This example demonstrates that  $|f(x_n)|$  can be very small even when  $|x^* - x_n|$  is not (where  $x^*$  is the exact solution). Thus, examining  $|f(x_n)|$  may not be the best way to determine if the sequence is converging to the root.

18. First, we will show that  $x_n = 1/n^k$  converges linearly to 0 for all  $k \in \mathbb{N}$ .

$$\frac{|x_{n+1} - 0|}{|x_n - 0|} = \frac{\frac{1}{(n+1)^k}}{\frac{1}{n^k}} = \frac{n^k}{(n+1)^k} = \frac{n^k}{n^k + g(n)},$$

where by the Binomial theorem,  $g(n)$  is a polynomial of degree at most  $k - 1$ . From calculus, we know that  $\lim_{n \rightarrow \infty} \frac{n^k}{n^k + g(n)} = 1$ . Hence,  $1/n^k$  converges linearly to zero. For the second part, we wish to find an  $N$  such that  $1/N^k < 10^{-m}$ . Note that

$$\frac{1}{N^k} < 10^{-m} \Rightarrow N^k > 10^m \Rightarrow N > (10^m)^{1/k} = 10^{m/k}.$$

So,  $N = 10^{m/k} + 1$  works.

19.

$$\frac{|x_{n+1} - 0|}{|x_n - 0|^2} = \frac{10^{-2^{n+1}}}{(10^{-2^n})^2} = \frac{10^{-2^{n+1}}}{10^{-2^n \cdot 2}} = \frac{10^{-2^{n+1}}}{10^{-2^{n+1}}} = 1.$$

## Chapter 4 Matrices

1. (a)  $x_1 = 1, x_2 = 3, x_3 = 0$ . The solution is unique. (b) No solution. (c)  $x_1 = 1 - 4x_3, x_2 = 3 + 2x_3, x_3$  is

free. In vector form, the solution is  $\mathbf{x} = x_3 \begin{bmatrix} -4 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}, x_3 \in \mathbb{R}$ .

11. (c) Yes, there is sufficient food, (d) 200, (e) 650

## Chapter 5 Discrete

1. (b) About 165.3 million

4. (a)  $a_{t+1} = a_t - a_t V + \gamma q V$  (b)  $c_{t+1} = c_t - c_t q + \gamma q = c_t(1 - q) + \gamma q$ . (c)  $q = 0.2, \gamma = 5.0, c_{t+1} = 0.8c_t + 1$ . (d) 5.0 millimoles per liter; stable

11. (d) The growth rate is eventually about 2.38% per year. Long term distribution of age classes is about 50.3% juvenile, 29.5% adult, 20.2% mature.

18. 37.5%

## Chapter 6 Continuous Models

7. (b) 61.365, 104.258. Errors: 49.831, 6.938, (c) 103.338, 111.104. Errors: 7.859, 0.0922

8. (a) 2.3894, 2.6707, (b) 2.7469, 2.7693

11. The root is about 1.27 (see Figure C.1).

12. The maximum is roughly  $y(0.78) = 2.36$ .

13. The asymptote is just below  $x = 1.3$ .

19. There are stable equilibria when  $x$  is an odd integer, and unstable equilibria when  $x$  is an even integer. See Figure C.2

25. (a) 7887 is a stable equilibrium, and 2113 is an unstable equilibrium. (b) About 5.1 years. (d) 2250 fish can be harvested each year.

26. (a) 0, 200, 4200

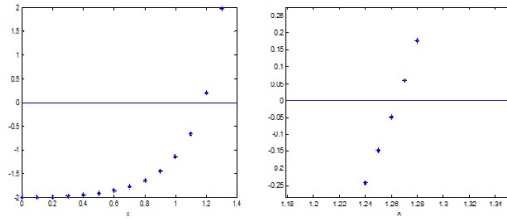


Figure C.1: Left: approximated  $y$  on  $[0, 1.4]$  with  $h = 0.1$ . Right: approximated  $y$  on  $[1.18, 1.34]$  with  $h = 0.01$ .

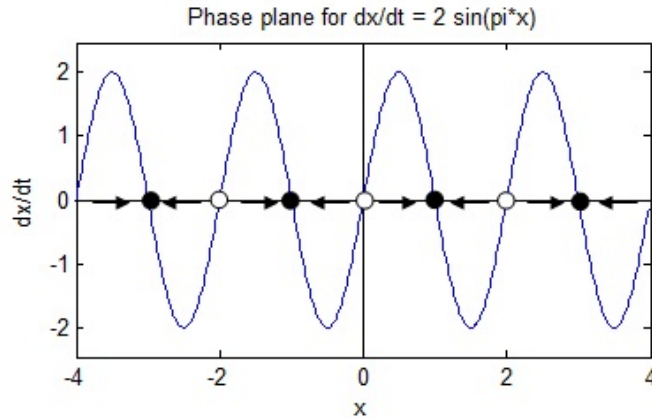


Figure C.2

28. (a)  $c'(t) = -.0175 - 0.002375c(t)$ , (b) About 133 days, (c)  $7.368 \text{ mg/m}^3$ , (d) About 549 days.  
 29.  $dQ/dt = 1.8 - 0.347Q$ . As  $t \rightarrow \infty$ ,  $Q \rightarrow 5.2 \text{ mg}$ .  
 30. (d, i)  $0.00222 \text{ L/mmHg}$ , (d, iii)  $15.97 \text{ mmHg/L/min}$   
 27.  $Q(t) = -3.9 \cdot 10^{-7}(100 - t)^4 - 0.4t + 40$ , and the concentration is  $c(t) = Q(t)/(200 - 2t)$   
 42. (a)  $\frac{dX}{dt} = X - \frac{QX}{1+X}$ ,  
 43. (b) Equilibria are  $(0,0,0)$  and  $(\sqrt{\beta(\rho - 1)}, \sqrt{\beta(\rho - 1)}, \rho - 1)$ .

## Chapter 7 Stochastic Models

1. (c) The mean is about 20, the standard deviation is about 53 (answers will vary, especially the standard deviation), (d) About 0.10  
 3. (a)  $E(X) = 5/3, Var(X) = 5/9$ , (b)  $E(Y) = -1/6, Var(Y) \approx 3.806$   
 4. (a)  $E(X) = 3.5, Var(X) \approx 2.917$ , (b)  $E(Y) = 3.5, Var(Y) \approx 0.1215$ , (d) About 0.849  
 7. (c) Since  $x^2 - 12x + 35 = (x - 5)(x - 7)$  is positive for  $x < 5$  and  $x > 7$  then  $P(X^2 - 12X + 35) > 0 = 8/10$ .  
 9. (a) 91st, (b) 529  
 11. (a)  $E(X) = 0, Var(X) = 2$ , (b) About 0.9973  
 13. (a) 0.950, (b) 0.0315, (c) 0.183  
 16.

(b) 0.3032

17. (a) 0.0273, (b) 0.0895

28 (b) The distribution is approximately normal with a mean around 254 million and a standard deviation around 12 million. (c) In the long term, the endowment approaches a normal distribution with a mean of 280 million and standard deviation close to 15 million.

29.  $1/e \approx 0.368$ .

## Appendix D

# MATLAB Commands

### Working with Matrices

MATLAB command	What it does									
<code>x = some number</code>	Assigns a number to variable <code>x</code>									
<code>x = start:increment:end</code>	Assigns an array of numbers to variable <code>x</code> . The first value is <i>start</i> , and the subsequent values are obtained by taking increments of size <i>increment</i> , and the array ends when value <i>end</i> has been reached. Example: <code>x = 2:3:17</code> yields <code>x = [2, 5, 8, 11, 14, 17]</code>									
<code>A=[1 2 3; 4 5 6; 7 8 9]</code>	Assigns matrix <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td></tr></table> to variable <code>A</code>	1	2	3	4	5	6	7	8	9
1	2	3								
4	5	6								
7	8	9								
<code>R = [1 2 3]</code>	Assigns the row vector <code>[1 2 3]</code> to variable <code>R</code>									
<code>C = [1; 4; 7]</code>	Assigns the column vector <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1</td></tr><tr><td>4</td></tr><tr><td>7</td></tr></table> to variable <code>C</code>	1	4	7						
1										
4										
7										
<code>A(n,m)</code>	Output $a_{nm}$ (row $n$ and column $m$ of $A$ )									
<code>A(n,:)</code>	Output row $n$ of $A$									
<code>A(:,m)</code>	Output column $m$ of $A$									
<code>size(A)</code>	Output the size of $A$ , in the form: [number of columns    number of rows]									
<code>R = [1 2 3]</code>	Generates a row matrix <code>[123]</code> and assigns it to <code>R</code>									
<code>C = [1; 4; 7]</code>	Generates a column matrix <code>[147]<sup>T</sup></code> and assigns it to <code>C</code>									
<code>x = 2:3:17</code>	Assigns an array of numbers to variable <code>x</code> , from 2 to 17, in increments of 3.									
<code>A'</code>	Computes the transpose of $A$									
<code>A^n</code>	Computes the product $A \cdot A \dots A$ (with $n$ factors of $A$ )									
<code>A.^n</code>	Raises each element of $A$ to the power $n$									
<code>rref(A)</code>	Reduces $A$ to row reduced echelon form (performs Gaussian elimination)									
<code>inv(A)</code>	Computes the matrix inverse $A^{-1}$									
<code>A\b</code>	Solves the equation $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$ by using Gaussian elimination									
<code>det(A)</code>	Computes the determinant of $A$									
<code>eye(n)</code>	Produces the $n \times n$ identity matrix, $I_n$ .									
<code>eig(A)</code>	Computes the eigenvalues of $A$									
<code>[x y] = eig(A)</code>	Computes the eigenvectors and eigenvalues of $A$									



## MATLAB Operators and Functions

Mathematical operator	MATLAB Command
Multiplication: $xy$	<code>x*y</code>
Radicals: $\sqrt{x}$	<code>sqrt(x)</code>
Exponents: $x^n$	<code>x^n</code>
Logarithms: $\ln x$ , $\log_{10}(x)$	<code>log(x)</code> , <code>log10(x)</code>
Absolute value: $ x $	<code>abs(x)</code>
Exponential function: $e^x$	<code>exp(x)</code>
Trigonometric functions	<code>sin(x)</code> , <code>cos(x)</code> , <code>tan(x)</code>
Inverse Trigonometric functions	<code>asin(x)</code> , <code>acos(x)</code> , <code>atan(x)</code>
Floor (greatest integer below)	<code>floor(x)</code>
Ceiling (smallest integer above)	<code>ceil(x)</code>
Round to nearest integer	<code>round(x)</code>
Sign of $x$ (returns 1 if $x > 0$ , $-1$ if $x < 0$ , and 0 if $x = 0$ )	<code>sign(x)</code>
Random number in $[0, 1]$	<code>rand(x)</code>

## Miscellaneous commands

MATLAB command	What it does
<code>disp('text')</code>	Output text to command line
<code>format long</code>	Outputs numbers with 15 digit accuracy
<code>vpa(fraction)</code>	Converts fractions into decimal approximations
<code>size(R)</code>	Outputs the size of the matrix R. The first number is the number of rows, and the second is the number of columns.
<code>figure(n)</code>	Opens up figure number n (useful for working with multiple figures)
<code>hold on</code>	Saves plots so that the next time there is a plot command, the command is applied to the current plot (instead of a brand new plot).
<code>break</code>	Forces MATLAB to exit a while or for loop
<code>return</code>	Forces MATLAB to exit the entire script or function
<code>CTRL+C</code>	Exits execution of code

## Plotting

To plot symbolic functions (for example,  $f(x) = \cos x$ )

```
ezplot('cos(x)')
ezplot('cos(x)', [xmin,xmax])
ezplot('cos(x)', [xmin,xmax,ymin,ymax])
```

To plot a single point  $(x, y)$ ,

```
plot(x,y)
plot(x,y,'s')
```

where **s** may consist of one element from any or all of the these 3 columns:

<b>b</b>	blue	<b>.</b>	point	<b>-</b>	solid
<b>g</b>	green	<b>o</b>	circle	<b>:</b>	dotted
<b>r</b>	red	<b>x</b>	x-mark	<b>-.</b>	dashdot
<b>c</b>	cyan	<b>+</b>	plus	<b>--</b>	dashed

```

        m    magenta      *    star          (none) no line
        y    yellow      s    square
        k    black       d    diamond
        w    white       v    triangle (down)
                                ^    triangle (up)
<    triangle (left)
>    triangle (right)

        p    pentagram
        h    hexagram

```

Notes

- If  $x$  and  $y$  are both vectors, then  $x$  is plotted on the horizontal axis.
- Plotting in MATLAB takes a lot of practice and persistence. You may find yourself often doing internet searches or asking questions to your instructor.

## For Loops

The set of statements inside the loop are executed as the variable `var` iterates through the integers `start` to `end`.

```

for var=start:end
    statements
:
end

```

## Logical Expressions

A logical expression is some proposition that is true or false. Examples of logical expressions are in the table below.

Logical Expression to test	MATLAB notation
$x < y$	<code>x &lt; y</code>
$x \leq y$	<code>x &lt;= y</code>
$x = y$	<code>x == y</code>
$x \neq y$	<code>x != y</code>

Let  $p$  and  $q$  be two logical expressions. We can combine these to form a compound logical expression.

Compound logical Expression	MATLAB notation	When is it true?
$p$ and $q$	<code>p &amp;&amp; q</code>	When both $p$ and $q$ are true
$p$ or $q$	<code>p   q</code>	When at least one of $p$ or $q$ are true

## While Loops

The set of statements inside the loop are executed as long as *logical\_expression* is true.

```

while logical_expression
    statements
:
end

```

## Conditional Statements

**Standard conditional statement:** The set of statements are executed if *logical\_expression* is true. If *logical\_expression* is not true, then MATLAB will skip the set of statements and continue execution after the **end** statement.

```
if logical_expression
    statements
    :
end
```

**If-else statement:** if *logical\_expression* is true, then statements A are executed. If *logical\_expression* is not true, then statements B are executed.

```
if logical_expression
    statements A
else
    statements B
end
```

**Multiple if-else statement:** If *logical\_expression 1* is true, then *statements A* are executed. Otherwise, if *logical\_expression 2* is true, *statements B* are executed. If *logical\_expression 1* and *logical\_expression 2* are false, and *logical\_expression 3* is true, then *statements C* are executed. If none of these logical expressions are true, then *statements D* are executed. Note that only one set of *statements A*, *B*, *C*, and *D* are allowed to be executed. Also, the very last **else** statement is not required, you may not want to do anything if none of the logical expression are true. There are many possible variations of multiple if-else statements.

```
if logical_expression 1
    statements A
elseif logical_expression 2
    statements B
elseif logical_expression 3
    statements C
else
    statements D
end
```

# Appendix E

## Debugging

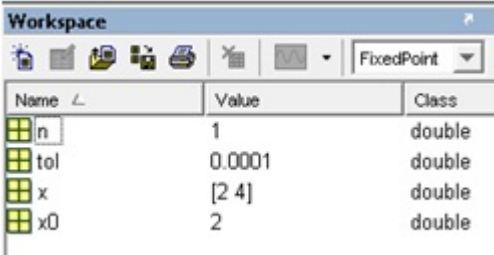
When running a program, you may set a breakpoint at any line of code to temporarily pause the execution of the code. To do this, click to the left of the line number, and you should see a red circle appear. When you run the code, you should see a green arrow at the corresponding line, telling you the line that is about to be executed.

```

1  function FixedPoint(x0)
2  -  tol=1e-4;
3  -  x(1)=x0;
4  -  x(2)=g(x(1));
5  -  n=1;
6  -  while abs(x(n+1)-x(n)) > tol
7  -      n=n+1;

```

In the Workspace window, you should be able to see all of the variables that have been declared. For this program, there are four variables so far.



The screenshot shows the MATLAB Workspace window with the following variables:

Name	Value	Class
n	1	double
tol	0.0001	double
x	[2 4]	double
x0	2	double

You can also enter variable names into the command line to find what the value of the variable is, or other information about the variable.

You can also step through a program line-by-line. After the code has reached a breakpoint, hit F10 to go to the next line. When you do this, you should see the green arrow advance to the next line. When you reach a conditional (if/then) statement, you can see if the condition is satisfied. For the example below, n is 2 and x(3) is 16, so the condition is not satisfied. If you hit F10 again, the green arrow skips ahead to the "end" statement.

```
EDU>> FixedPoint(2)
K>> x

x =
     2     4
      ← x is the row vector [2 4]

K>> size(x)

ans =
     1     2
      ← x is a 1 × 2 matrix.

K>> | _____
```

```
6 ♦ while abs(x(n+1)-x(n)) > tol
7 -     n=n+1;
8 -     x(n+1)=g(x(n));
9 - →     if n > 100 || abs(x(n+1))>100
10 -         disp('Fixed point iteration appears to be diverging')
11 -         return
12 -     end
13 - end
```

# Appendix F

## Completed Proofs

### Chapter 3

**Theorem 3.1.4** If  $f$  is continuous on  $[a, b]$ , and  $a \leq f(x) \leq b$  for all  $x \in [a, b]$ , then  $f$  has a fixed point in  $[a, b]$ .

*Proof.* If  $a = f(x)$  or  $b = f(x)$  for any  $x \in [a, b]$ , we have found a fixed point and are done. Assume then that  $a < f(x) < b$ . Let  $g(x) = f(x) - x$ . Then  $g(a) = f(a) - a > a - a = 0$ , and  $g(b) = f(b) - b < b - b = 0$ . Since  $g(a) > 0$  and  $g(b) < 0$ , then by the Intermediate Value Theorem, there must be some  $c \in [a, b]$  such that  $g(c) = 0$ . This implies that  $f(c) - c = 0$ , or  $f(c) = c$ .  $\square$

**Theorem 3.1.5:** If  $f$  is smooth on  $[a, b]$ ,  $a \leq f(x) \leq b$ , and  $|f'(x)| \leq k$  for some  $k < 1$  on  $(a, b)$ , then  $f$  has a unique fixed point  $x^* \in [a, b]$ . In addition, for any number  $x_0$  in  $[a, b]$ , the sequence defined by  $x_n = f(x_{n-1})$ ,  $n \geq 1$ , converges to this unique fixed point  $x^*$  in  $[a, b]$ . Moreover, the error  $|x_n - x^*|$  satisfies the inequality  $|x_n - x^*| \leq k^n \max\{x_0 - a, b - x_0\}$ .

*Proof.* Clearly,  $f$  has a fixed point in  $[a, b]$  by Theorem 3.1.4. To show uniqueness, suppose that  $p$  and  $q$  are distinct fixed points in  $[a, b]$ , and  $p < q$ . Then  $f(p) = p$  and  $f(q) = q$ . By the mean value theorem, there is a  $\xi \in (p, q)$  such that  $f(q) - f(p) = f'(\xi)(q - p)$ . By assumption,  $|f'(\xi)| \leq k < 1$ . Hence,

$$|q - p| = |f(q) - f(p)| = |f'(\xi)(q - p)| = |f'(\xi)||q - p| \leq k|q - p| < |q - p|,$$

and a contradiction occurs since a number cannot be less than itself. This shows that  $f$  must have a unique fixed point in  $[a, b]$ . For the second part, we will show that the sequence  $x_n = f(x_{n-1})$  converges to this unique fixed point  $x^* \in [a, b]$ . Let  $n \geq 1$  be an integer and  $x_0 \in [a, b]$ . Again by the mean value theorem, there is a  $\xi_0 \in (a, b)$  such that  $f(x_0) - f(x^*) = f'(\xi_0)(x_0 - x^*)$ . Substituting  $x_1$  for  $f(x_0)$  and taking absolute values, we may write this as  $|x_1 - x^*| = |f'(\xi_0)||x_0 - x^*|$ . By assumption, we have  $|f'(\xi_0)| \leq k$ , so we may conclude

$$|x_1 - x^*| \leq k|x_0 - x^*|.$$

At the next step, there is a  $\xi_1 \in (a, b)$  such that  $f(x_1) - f(x^*) = f'(\xi_1)(x_1 - x^*)$ , and following the steps above we arise at

$$|x_2 - x^*| \leq k|x_1 - x^*| \leq k^2|x_0 - x^*|.$$

Using induction, we can see that

$$|x_n - x^*| \leq k^n|x_0 - x^*| \text{ for all } n \geq 1.$$

Since  $0 < k < 1$ , the right hand side satisfies  $k^n|x_0 - x^*| \rightarrow 0$  as  $n \rightarrow \infty$ . Hence,  $|x_{n+1} - x^*| \rightarrow 0$ , and  $x_n \rightarrow x^*$ .

To prove the error bound, we use the result above to see that  $|x_n - x^*| \leq k^n|x_0 - x^*| \leq k^n \max\{x_0 - a, b - x_0\}$   $\square$

**Theorem 3.3.1:** Suppose that  $f$  is smooth on  $[a, b]$ , and that  $f(a)$  and  $f(b)$  have opposite signs. The Bisection method generates a sequence  $x_n$  approximating a zero  $x$  of  $f$  with  $|x_n - x| \leq (b - a)/2^n, n \geq 1$ . To see why this theorem is true, consider the initial case  $n = 1$ . We choose  $x_1 = (a + b)/2$ . Without loss of generality, suppose that  $f(x_1)$  and  $f(a)$  have opposite signs. By the intermediate value theorem, there is a  $x \in [a, x_1]$  such that  $f(x) = 0$ . We easily see that

$$|x_1 - x| = x_1 - x \leq (x_1 - x) + (x - a) = x_1 - a = \frac{b - a}{2}.$$

We may use induction to prove the theorem holds for the general case  $n \geq 1$ .

**Theorem 3.4.1:** Suppose  $f(x)$  has a root  $p$  in  $[a, b], f'(p) \neq 0$ , and  $f'$  is smooth on  $[a, b]$ . Then there is some closed interval  $I$  containing  $p$  such that Newton's method converges to  $p$  for any initial guess  $x_0 \in I$ .

*Proof.* Let the sequence  $\{x_n\}_{n=0}^\infty$  be given using Newton's method,  $x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$  for  $n \geq 1$ . Let  $g(x) = x - \frac{f(x)}{f'(x)}$ . Then the sequence may be rewritten in the form  $x_n = g(x_{n-1})$ . Since  $f(p) = 0$ , it follows that  $g(p) = p$ , so  $g$  has a fixed point in  $[a, b]$ . Next,

$$g' = 1 - \frac{f'f' - ff''}{(f')^2} = \frac{(f')^2 - (f')^2 + ff''}{(f')^2} = \frac{ff''}{(f')^2}.$$

Since  $f(p) = 0$  and  $f'(p) \neq 0$  by assumption, we see that  $g'(p) = \frac{f(p)f''(p)}{(f'(p))^2} = 0$  as well. Since  $g'$  is continuous, there is a closed interval  $I$  such that  $p \in I$  and  $|g'(x)| \leq k, k < 1$ , for all  $x \in I$ . Following the proof of Theorem 3.1.5, we could show that the fixed point in  $I$  is unique, and the sequence  $x_{n+1} = g(x_n)$  converges to the fixed point  $p$  for any initial guess  $x_0 \in I$ . Hence, the sequence  $x_n$  converges to  $p$  for any initial guess  $x_0 \in I$ .  $\square$

## Chapter 5

**Proposition 5.4.2:** If  $\mathbf{x}$  is a probability vector, and  $P$  is a transition matrix, then  $P\mathbf{x}$  is a probability vector.

*Proof.* We will prove the case when  $n = 2$ . Let  $\mathbf{x} = (x_1, x_2)^T$  be a probability vector, and  $P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$  be a stochastic matrix. Then,

$$P\mathbf{x} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} p_{11}x_1 + p_{12}x_2 \\ p_{21}x_1 + p_{22}x_2 \end{bmatrix}.$$

To show that  $P\mathbf{x}$  is a probability vector, we need to show its entries add to one. Indeed,

$$\begin{aligned} p_{11}x_1 + p_{12}x_2 + p_{21}x_1 + p_{22}x_2 &= (p_{11} + p_{21})x_1 + (p_{12} + p_{22})x_2 \\ &= x_1 + x_2 \quad (\text{since } P \text{ is a stochastic matrix}) \\ &= 1 \quad (\text{since } \mathbf{x} \text{ is a probability vector}) \end{aligned}$$

$\square$

## Chapter 6

**Theorem 6.1.3:** Let  $y$  be the solution to the initial value problem  $y'(t) = f(t, y), y(t_0) = y_0$ . Suppose that there is a constant  $M$  such that  $|y''(t)| \leq M$  for all  $t \in [t_0, t_n]$ . In addition, suppose that there is a constant  $L$  such that  $|f(t, x) - f(t, y)| \leq L|x - y|$  for all  $x, y$ . Then the error at time  $t_n$  satisfies

$$|y(t_n) - y_n| \leq \frac{hM}{2L} \left( e^{L(t-t_0)} - 1 \right).$$

*Proof.* Consider the error at the first step,  $T_1 := y(t_1) - y_1$ , where  $y_1 = y(t_0) + y'(t_0)(t_1 - t_0)$  approximation of the solution at  $t_1$ , and  $y(t_1)$  is the true value of the solution at  $t_1$ . Using the fundamental theorem of Calculus,

$$y(t_1) - y(t_0) = \int_{t_0}^{t_1} y'(s) ds.$$

Using the integration by parts formula  $\int u dv = uv - \int v du$  with  $u = y'(s)$ ,  $dv = ds$ ,  $du = y''(s)ds$  and  $v = s - t_1$ , we get

$$\begin{aligned} y(t_1) - y(t_0) &= y'(s)(s - t_1)|_{t_0}^{t_1} - \int_{t_0}^{t_1} y''(s)(s - t_1) ds \\ &= y'(t_0)(t_1 - t_0) - \int_{t_0}^{t_1} y''(s)(s - t_1) ds. \end{aligned}$$

Using the fact that  $y_1 = y(t_0) + y'(t_0)(t_1 - t_0)$ , we have

$$y(t_1) - y(t_0) = y_1 - y(t_0) = \int_{t_0}^{t_1} y''(s)(s - t_1) ds.$$

Canceling  $y(t_0)$  and rearranging terms we see that

$$T_1 = y(t_1) - y_1 = \int_{t_0}^{t_1} y''(s)(s - t_1) ds.$$

Since  $|y''(t)| \leq M$ , it follows that

$$|T_1| \leq \int_{t_0}^{t_1} |M||s - t_1| ds = |M| \int_{t_0}^{t_1} (t_1 - s) ds = \frac{1}{2}Mh^2.$$

Next let  $T_i = y(t_{i+1}) - y(t_i) - hf(t_i, y(t_i))$  be the additional error from step  $i$  to step  $i + 1$  and  $e_i := y(t_i) - y_i$  be the accumulated error at step  $i$ .

$$e_{i+1} = y(t_i) + hf(t_i, y(t_i)) + T_i - (y_i + hf(t_i, y_i)) = e_i + h[f(t_i, y(t_i)) - f(t_i, y_i)] + T_i.$$

By the mean value theorem there is a  $y_i^*$  between  $y_i$  and  $y(t_i)$  such that

$$f(t_i^*, y(t_i)) - f(t_i, y_i) = f_y(t_i, y_i^*)(y(t_i) - y_i).$$

Thus,

$$|f(t_i, y(t_i)) - f(t_i, y_i)| \leq R|e_i|.$$

Combining terms,

$$e_{i+1} \leq |e_i| + h|f(t_i, y(t_i)) - f(t_i, y_i)| + T_i \leq |e_i| + hR|e_i| + \frac{Mh^2}{2},$$

which finally leads to

$$e_{i+1} = (1 + hR)|e_i| + \frac{Mh^2}{2}.$$

Letting  $C = 1 + hR$  and computing the first few errors,

$$\begin{aligned} |e_1| &\leq C|e_0| + \frac{1}{2}Mh^2 = \frac{1}{2}Mh^2, \\ |e_2| &\leq C|e_1| + \frac{Mh^2}{2} \leq C\frac{Mh^2}{2} = (1 + C)\frac{1}{2}Mh^2, \\ |e_3| &\leq C|e_2| + \frac{Mh^2}{2} \leq C(1 + C)\frac{Mh^2}{2} + \frac{Mh^2}{2} = (1 + C + C^2)\frac{Mh^2}{2}. \end{aligned}$$



Inductively, we see that

$$|e_n| \leq (1 + C + \cdots + C^{n-1}) \frac{Mh^2}{2}.$$

Recalling the formula for a geometric sum,  $1 + C + \cdots + C^{n-1} = \frac{1-C^n}{1-C} = \frac{(1+Rh)^n - 1}{Rh}$ . Also, Taylor's Theorem can be used to easily prove that  $1 + Rh < e^{Rh}$ , so we have

$$|e_n| \leq \frac{e^{Rhn} - 1}{R} \cdot \frac{Mh}{2}.$$

Since  $nh = b - t_0$  we also have

$$|e_n| \leq Mh \frac{e^{R(b-t_0)} - 1}{2R}.$$

□

## Chapter 7

**Proposition 7.3.1:** Let  $X$  and  $Y$  be random variables, and let  $a$  be a real number. Then,

- (a)  $E(a) = a$
- (b)  $E(aX) = aE(X)$
- (c)  $E(X + Y) = E(X) + E(Y)$
- (d)  $\text{Var}(aX) = a^2\text{Var}(X)$
- (e)  $\text{Var}(X) = E(X^2) - E(X)^2$
- (f)  $\text{Var}(X + a) = \text{Var}(X)$

*Proof.* (b) If  $X$  is a continuous random variable,  $E(aX) = \int_{\mathbb{R}} af(x) dx = a \int_{\mathbb{R}} f(x) dx = aE(X)$ . If  $X$  is discrete, simply replace the integrals with sums.

(c) See [25] for a proof.

(d)  $\text{Var}(aX) = E[(E(aX) - aX)^2] = E[(aE(X) - aX)^2] = E[a^2(E(X) - X)^2] = a^2E[(E(X) - X)^2] = a^2\text{Var}(X)$ , where we have used (b) at several points.

(e) Using parts (b) and (c),

$$\begin{aligned} \text{Var}(X) &= E((X - E(X))^2) = E(X^2 - 2XE(X) + (E(X))^2) \\ &= E(X^2) - E(2XE(X)) + E((E(X))^2) = E(X^2) - 2E(X)E(X) + (E(X))^2 \\ &= E(X^2) - (E(X))^2 \end{aligned}$$

(f)  $\text{Var}(X + a) = E[(E(X + a) - (X + a))^2] = E[(E(X) + a - X - a)^2] = E[(E(X) - X)^2]$ . □

**Theorem 7.3.4: The Central Limit Theorem:** Let  $\{X_i\}_{i=1}^{\infty}$  be a sequence of independent random variables, each having the same distribution with  $E(X_i) = \mu$  and  $\text{Var}(X_i) = \sigma^2$ . Let  $\bar{x}_n = \frac{1}{n} \sum_{i=1}^n X_i$ . Then

- (a)  $E(\bar{x}_n) = \mu$
- (b)  $\text{Var}(\bar{x}_n) = \sigma^2/n$
- (c) The distribution of  $\bar{x}_n$  approaches a normal distribution as  $n \rightarrow \infty$ .

*Proof.* Notice that by Proposition 7.3.1(b) that

$$E[\bar{x}_n] = E\left(\frac{1}{n} \sum_{i=1}^n X_i\right) = \frac{1}{n} E[X_1 + X_2 + \cdots + X_n].$$

Next, using Proposition 7.3.1(c) repeatedly gives

$$E[\bar{x}_n] = \frac{1}{n} (E[X_1] + E[X_2] + \cdots + E[X_n]) = \frac{1}{n} (\mu + \mu + \cdots + \mu) = \frac{1}{n} (n\mu) = \mu.$$

To prove (b), we use Proposition 7.3.1(d):

$$\text{Var}(\bar{x}_n) = \text{Var}\left(\frac{1}{n} \sum_{i=1}^n X_i\right) = \frac{1}{n^2} \text{Var}(X_1 + X_2 + \cdots + X_n).$$

Since the variables are independent, we may use Proposition 7.3.3 repeatedly to obtain

$$\text{Var}(\bar{x}_n) = \frac{1}{n^2} (\text{Var}(X_1) + \text{Var}(X_2) + \cdots + \text{Var}(X_n)) = \frac{1}{n^2} (\sigma^2 + \sigma^2 + \cdots + \sigma^2) = \frac{1}{n^2} \cdot n\sigma^2 = \frac{\sigma^2}{n}.$$

For a formal proof of (c) see [12]. □

# Index

- Alee effect, 106
- algorithm, 17
- Allee effect, 104
- argument, 21, 38
- assignment statement, 18
- attractor, 63
- augmented matrix, 47
- autonomous differential equation, 88, 94, 99
  
- Bernoulli trial, 123
- binomial distribution, 123
- birth death process, 137
- Birth process, 136
- bisection method, 39
- boolean expression, 20
  
- calling, 21
- carrying capacity, 61, 81
- Central limit theorem, 120
- characteristic equation, 55
- column vector, 19
- Command Window, 18
- compartment diagram, 82
- competition model, 108, 109
- conditional probability, 119
- conditional statement, 28
- continuous random variable, 113, 117
- critical point, 93
- cumulative distribution function, 118
  
- determinant, 52
- discrete dynamical system, 60
- discrete random variable, 113
- disease model, 11, 110
  
- eigenvalue, 54
- eigenvalue, dominant, 63
- eigenvector, 54
- eigenvector, dominant, 63
- elementary row operations, 47
- equilibrium, 63, 71, 88, 93
- equilibrium, stability, 63
- Euler's method, 84
- Euler's method for systems, 91
- expected value, 114, 118
- explicit formula, 10
  
- exponential distribution, 127, 129
- ezplot, 21
  
- Fibonacci sequence, 71
- fixed point, 35
- fixed point iteration, 36
- for loop, 22
- function, 21
- fundamental matrix, 70
  
- Gaussian distribution, 126
- Gaussian elimination, 47
- Gompertz growth, 14
  
- Heart output model, 106
- Heun's method, 103
  
- if-then-else statement, 28
- Improved Euler's method, 86
- independent random variables, 119
- Intermediate Value Theorem, 39
  
- Jacobian matrix, 99
  
- Leslie matrix, 64
- Leslie model, 76, 77
- limiting distribution, 67
- limiting matrix, 67
- linear convergence, 41
- little-oh, 135
- logical expression, 20, 25
- logistic model, continuous, 81
- logistic model, discrete, 61
- Lorenz oscillator, 110
  
- M-file, 23
- Malthusian model, 60, 81
- Mandelbrot set, 45
- Markov chain, 65
- Markov chain, absorbing, 68
- Markov property, 131
- mass action, 11
- MATLAB function, 38
- matrix, 19, 46
- matrix, diagonal matrix, 51
- matrix, identity matrix, 51

- matrix, invertible or nonsingular, 51
- matrix-vector multiplication, 50
- Monte Carlo integration, 122
- Monte Carlo method, 115
  
- Newton's Law of Cooling, 14, 104, 107
- Newton's method, 41
- nondimensionalization, 90
- normal distribution, 126
- nullcline, 93
  
- phase line diagram, 88
- phase plane diagram, 93
- plot, 21
- Poisson distribution, 127–129
- Poisson process, 135
- preallocation, 24
- predator prey model, 90, 147
- probability density function, 118
- probability distribution, 113
- probability histogram, 132
- probability mass function, 114
- probability vector, 66
- projection matrix, 64
  
- quadratic convergence, 41
  
- random variable, 113
- Random variables, properties, 119
- recurrence relation, 10, 60
- reduced row echelon form, 48
- repeller, 63
- return, 38
- Ricker's model, 75
- rounding, 20
- row echelon form, 48
- row vector, 19
- Runge-Kutta method, 87, 103
  
- saddle point, 63
- sample space, 113
- scalar, 19, 47
- secant method, 43
- singular, 51
- smooth, 37
- spectral radius, 63
- stability, 72, 88, 93, 98, 99
- standard deviation, 114
- state diagram, 64
- state vector, 64
- stationary distribution, 67
- stochastic model, 113
- stochastic process, 134
- stopping condition, 38
- switch statement, 29
  
- Symbolic Toolkit, 22
  
- teasel, 173
- trajectory, 63
- transition matrix, 66
- transpose, 19, 46
- Tumor growth model, 108
  
- uniform random variable, 113
  
- variance, 114, 118
- vector, 19, 46
- vectorizing code, 24
  
- waiting time, 127
- while loop, 25
- Workspace, 18

# Bibliography

- [1] Frederick Adler. *Modeling the Dynamics of Life: Calculus and Probability for Life Scientists*. Brooks/Cole, 2004.
- [2] Edward A. Bender. *An Introduction to Mathematical Modeling*. John Wiley & Sons, New York-Chichester-Brisbane, 1978. A Wiley-Interscience Publication.
- [3] S. Brault and H. Caswell. Pod-specific demography of killer whales (ornicus orca). *Ecology*, 74:5:1444–1454, 07 1993.
- [4] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. Brooks/Cole, 6th edition, 1997.
- [5] Hal Caswell. *Matrix Population Models: Construction, Analysis, and Interpretation*. Sinauer Associates, Inc., Massachusetts, second edition, 2001.
- [6] Steven C. Chapra. *Applied Numerical Methods with MATLAB for Engineers and Scientists*. McGraw Hill Companies, Inc., New York, 2nd edition, 2008.
- [7] Jeffrey R. Chasnov. *Mathematical Biology*.
- [8] James L. Cornette and Ralph A. Ackerman. *Calculus for the Life Sciences: A Modeling Approach, Volume I*. Cornette and Ackerman, 2011.
- [9] William P. Fox. *Mathematical Modeling with Maple*. Brooks/Cole, Cengage Learning, Boston, 2012.
- [10] Geoffrey R. Grimmett and David R. Stirzaker. *Probability and Random Processes*. Oxford University Press, New York, third edition, 2001.
- [11] Charles M. Grinstead and J. Laurie Snell. *Introduction to Probability*.
- [12] Robert V. Hogg and Elliot A. Tanis. *Probability and Statistical Inference*. Pearson, 2015.
- [13] Mark Kot. *Elements of Mathematical Ecology*. Cambridge University Press, Cambridge, 2001.
- [14] David C. Lay. *Linear Algebra and Its Applications*. Brooks/Cole, 2011.
- [15] Glenn Ledder. *Mathematics for the Life Sciences*.
- [16] Joseph M. Mahaffy. Calculus for the life sciences i, lecture notes - discrete malthusian growth.
- [17] Daniel Maki and Maynard Thompson. *Mathematical Modeling and Computer Simulation*. Brooks/Cole, 2006.
- [18] The Mathworks, Inc., Natick, Massachusetts. *MATLAB version 9.10.0.1649659 (R2021a) Update 1*, 2021.
- [19] Cleve B. Moler. *Numerical Computing with Matlab*. The MathWorks, Inc., Natick, 2004.
- [20] Douglas D. Mooney and Randall J. Swift. *A Course in Mathematical Modeling*. Classroom Resource Materials Series. Mathematical Association of America, Washington, DC, 1999.

- [21] A. J. Nicholson and V. A. Bailey. The balance of animal populations. *Proceedings of the Zool. Soc. of London*, 105:551–598, 1935.
- [22] Juan Pinasco and Lilia Romanelli. Coexistence of languages is possible. *Physica A: Statistical Mechanics and its Applications*, 361:355–360, 02 2006.
- [23] Michael B. Rabinowitz, George W. Wetherill, and Joel D. Kopple. Kinetic analysis of lead metabolism in healthy humans. *The Journal of Clinical Investigation*, 58:260–270, 08 1976.
- [24] Eric Renshaw. *Modelling Biological Populations in Space and Time*, volume 11 of *Cambridge Studies in Mathematical Biology*. Cambridge University Press, Cambridge, 1991.
- [25] Sheldon Ross. *A First Course in Probability*. Pearson, 2006.
- [26] John D. Rowatt. Applications of markov chains to the critical element model for determining the fatigue life of composites. 1995.
- [27] James Stewart. *Calculus: Concepts and Contexts*. Cengage Learning, 2009.
- [28] William F. Trench. *Elementary Differential Equations*. Trench, 2013.
- [29] K. K. Tung. *Topics in Mathematical Modeling*. Princeton University Press, 2007.