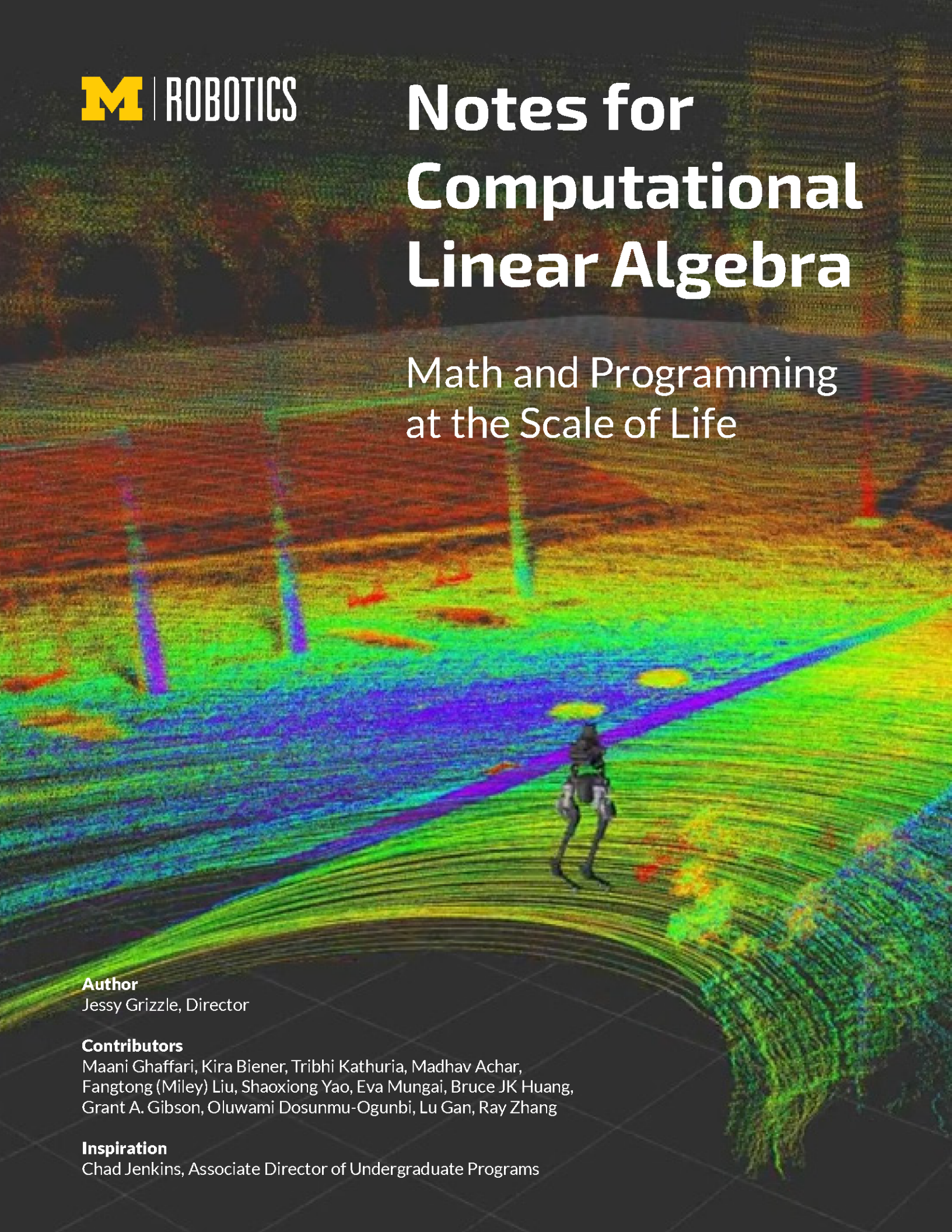
M | ROBOTICS

Notes for Computational Linear Algebra

Math and Programming at the Scale of Life



Author
Jessy Grizzle, Director

Contributors
Maani Ghaffari, Kira Biener, Tribhi Kathuria, Madhav Achar, Fangtong (Miley) Liu, Shaoxiong Yao, Eva Mungai, Bruce JK Huang, Grant A. Gibson, Oluwami Dosunmu-Ogunbi, Lu Gan, Ray Zhang

Inspiration
Chad Jenkins, Associate Director of Undergraduate Programs

Cover design by Dan Newman, Head of Communications, Michigan Robotics

Jerry W. and Carol L. Levin Professor of Engineering
Elmer G. Gilbert Distinguished University Professor
Professor of EECS and ME

Composed between April 2020 and August 2020 for use in ROB 101, Computational Linear Algebra, a first-semester first-year course. Revised Fall 2020 when put to the test with our Pilot Class. Revised Summer and Fall 2021.

First release, August 31, 2020. Second release, August 30, 2021

Contents

*___

Preface

This collection of course notes is dedicated to the group of students who were brave enough to take the pilot offering of ROB 101, Computational Linear Algebra in Fall 2020.

ROB 101 was conceived by Prof. Chad Jenkins as one part of a complete undergraduate curriculum in Robotics. Chad and I both thank Associate Dean for Undergraduate Education, Prof. Joanna Mirecki Millunchick, for her support in the offering of this course as ROB 101.

The following remarks are adapted from an Education Proposal led by Prof. Jenkins, Dr. Mark Guzdial, Ella Atkins, and myself.

A challenge for the current undergraduate curricular structure at the University of Michigan (and elsewhere) is that the best Robotics major is a quadruple major in ME, EE, CS, and Math with a minor in Psychology. The Robotics Institute at Michigan is poised to address this challenge in revolutionary ways as it evolves toward a department. The Robotics faculty are using the opportunity of a clean slate in the area of undergraduate robotics education-—and the absolute necessity to integrate learning across a wide range of traditional disciplines—to design a new curricular system where computation, hardware, and mathematics are on an equal footing in preparing a future innovation workforce.

By integrating linear algebra, optimization, and computation in the first semester, students will experience mathematics as a means of making predictions and reasoning about experimental outcomes and robot designs. They will be prepared to encounter physics as a means to build mathematical models that describe the movement of objects, those with a palpable mass as well as electrons and waves, while grounding all of this in design. And computation? They will find it is the engine that drives discovery and the means of extracting information from data, allowing their machines to make decisions in real-time.

In addition to **ROB 101 (*Computational Linear Algebra*)** in the first year, we are planning **ROB 102 (*Graph Search for Robotics and AI*)**, which will show how computers can reason autonomously through graph search algorithms. The objective of the course is for students to implement a path planner for autonomous navigation with a given mobile robot at a known location in a known environment. The course will build towards providing a broader conceptual foundation for modeling problems as graphs and inferring solutions through search.

In **ROB 103 (*Robotic Mechanisms*)**, students will experience hands-on robotic systems design, build, and operation. The objective of the course is for students, in teams, to build an omni-drive mobile robot that can be teleoperated, as a step towards the gateway course: ROB 204. Students will learn to safely and efficiently operate modern shop tools such as 3D printers and laser cutters as well as traditional machining tools such as lathes, mills, and drill presses. Students will learn basic electronic and power systems principles including safe battery management, wiring harness design and assembly, and signal measurement for test and debugging. Students will design and build their real-world mobile robot based on application requirements from conception with CAD software through manufacturing, assembly, and test. Each student team will be given a "kit" of servos, sensors, and low-cost embedded processing components to use in their design.

The new first-year curriculum allows for major innovation in the second and third year curriculum, and the advancing of many graduate topics to much earlier places in Robotics education. We hope that you will follow our efforts in bringing this curriculum to the University of Michigan.

Jessy Grizzle
Fall Term, 2020

Philosophy of the Course

The Robotics faculty want out of the Sputnik era of educating engineers, where we are forced to pound our students with four semesters of Calculus before we are able to engage them in any interesting engineering. We seek to prepare students for the era of Information, AI, Data, and of course, Robotics. While we believe our ideas for this Linear Algebra course will work for most engineering departments, we understand that the College of Engineering needs a skunkworks to test some of our more revolutionary suggestions before turning them loose on everyone. We are proud to serve that role.

ROB 101 assumes a High School course in Algebra and no background in programming. With these entry requirements, we seek to open up mathematics and programming to everyone with the drive and skills to arrive at the University of Michigan's College of Engineering. From its inception in December 2019, the plan has always been to teach the course in a hybrid mode. We are being very intentional to design the course for inclusivity with a focus on making sure that one's zip code is not the best predictor of success. To do that, we are re-imagining the way mathematics is introduced to first-semester Y1 undergrads. We want to break the Sputnik era 4-semester calculus chain where AP credits are a huge predictor of success.

We will begin mathematics with Linear Algebra, the workhorse of modern autonomous systems, machine learning, and computer vision. We are integrating it with computation, and to make sure that students without access to high-end tools can be successful, all the programming will be run in a web browser through cloud services that are hidden from the student. If you can google at home or the library, then you can complete our programming assignments. Setting this up is a challenge, but we feel it is very important to have a platform that has equity in mind. Our plans for a hybrid mode of teaching are motivated by a desire to have students from Minority Serving Institutions join the course this fall.

The material in the course leaves out many traditional Linear Algebra topics, such as eigenvalues, eigenvectors, or how to compute determinants for matrices larger than 2×2. Sounds crazy! Good. The course focuses on solving systems of linear equations at scale, meaning hundreds or thousands of variables, and all the mathematics in the book should be implementable in HW sets in Julia. With that as a premise, we chose to focus on a few things in Linear Algebra that work well at scale, such as triangular systems of equations. This led to the conviction that LU Factorization should be introduced early and in an understandable manner, which was made possible by a magical video by Prof. Gilbert Strang (MIT). Once you understand that matrix multiplication $C = A \cdot B$ can be done by multiplying the columns of A by the rows of B and summing them up, the LU Factorization becomes completely transparent, allowing triangular matrices to rule the day. And once you focus on triangular structures, even linear independence and linear combinations become much more approachable. Of course, some basic geometry is good, and thus Gram-Schmidt is featured along with the QR Factorization. In between, least squared error solutions to linear equations and regression are treated along with other useful tools.

The book wraps up with a treatment of root finding for both scalar and vector nonlinear equations, and a users' view of optimization that highlights the role that the abstract function "$\arg\min$" is playing in modern engineering.

Readers of the book will not find many applications of the juicy computational tools as they are treated in the HW sets, via jupyter notebooks, and in three amazing Projects. The first project leads students through the process of building a map for robot navigation from LiDAR data collected on the UofM North Campus Grove. The main Linear Algebra concept being explored is the transformation of points and collections of points in $\mathbb{R}^3$ under rigid body transformations. The second project is built around regression and will give students insight into the power of Machine Learning. The third project will focus on the control of a planar Segway using a simplified version of Model Predictive Control. Students will experience the excitement of balancing a challenging ubiquitous mobile platform, and while doing so, will learn about ODEs and their relation to iterative discrete-time models.

Finally, we thank Prof. Steven Boyd for making his Y1 Applied Linear Algebra course material open source (http://vmls-book.stanford.edu/). We have done the same (https://tinyurl.com/sxk8n4u9).

Jessy Grizzle and Maani Ghaffari Ann Arbor, Michigan USA

List of Algorithms and Methods or Things You Need Code to Do Well

The book covers the following algorithms. In HW, you will subsequently code most of them in the Julia Programming Language and apply them "at the scale of life"! The class projects will bring Linear Algebra to life.

- Forward substitution; see Chap. 3.5
- Back substitution; see Chap. 3.6
- Standard matrix multiplication; see Chap. **??**
- A second way to do matrix multiplication; see Chap. **??**
- LU Factorization without row permutations; see Chap. **??**
- Using LU to solve $Ax = b$; see Chap. **??** and Chap. **??**
- (Optional Read): LU Factorization with row permutations; see Chap. **??**
- Using LU to compute $\det(A)$; see Chap. **??**
- Building a row permutation matrix; see Chap. **??**
- Checking linear independence; see Chap. **??**
- (Optional Read): LDLT Factorization (aka Cholesky Factorization); see Chap. **??**
- Counting number of linearly independent columns of a matrix; see Chap. **??**
- Checking linear combinations; see Chap. **??**
- Linear regression for overdetermined equations; see Chap. **??**
- Gram-Schmidt; see Chap. **??** and (Optional Read): Modified Gram-Schmidt; see Chap. **??**
- QR Factorization; see Chap. **??**
- Linear regression for underdetermined equations; see Chap. **??**
- Null space of a matrix; see Chap. **??**
- Bisection Algorithm; see Chap. **??**
- Numerical derivatives; see Chap. **??** and Chap. **??**
- Newton's Method; see Chap. **??** and Newton-Raphson Algorithm; see Chap. **??**
- Gradient Descent; see Chap. **??**
- Second-order optimization using the Hessian; see Chap. **??**
- (Optional Read): Quadratic Program (QP) and max-margin classifier; see Chap. **??** and Chapter **??**
- (Optional Read): Solving simple ODEs; see Appendix B

Chapter 1

Introduction to Systems of Linear Equations

Learning Objectives

- Get you going on Algebra, just in case Calculus has erased it from your mind
- Review on your own the quadratic equation.
- Set the stage for cool things to come.

Outcomes

- Refresher on the quadratic equation.
- Examples of systems of linear equations with two unknowns. Show that three things are possible when seeking a solution:
  - there is one and only one solution (one says there is a unique solution, which is shorthand for "there is a solution and it is unique");
  - there is no solution (at all); and
  - there are an infinite number of solutions
- Notation that will allow us to have as many unknowns as we'll need.
- Remarks that you can mostly ignore on why some numbers are called counting numbers, rational numbers, irrational numbers, or complex numbers
- Remarks on your first project.

1.1 For Review on Your Own: You Have Done Algebra Before

Quadratic Equation

$ax^2 + bx + c = 0$, where x is an *unknown variable* and *typically* a, b, and c are fixed real numbers, called *constants*. If $a \neq 0$, the solutions to this *nonlinear algebraic equation* are

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a},$$

where the symbol $\pm$ (read, plus minus) indicates that there is one solution x corresponding to the plus sign and a second solution x corresponding to the minus sign.

The *discriminant* of a quadratic equation is $\Delta := b^2 - 4ac$. If $\Delta \geq 0$, the two solutions are *real numbers*, while if $\Delta < 0$, the two solutions are *complex numbers*.

Complex Numbers

If you have not learned complex numbers (**a**lso **k**nown **a**s (**aka**) ***imaginary numbers***) or are fuzzy on the details, let your GSI know. We may not use complex numbers at all in the graded portion of the course. We're not sure yet! We will need complex numbers when we study eigenvalues of matrices, which could happen at the end of the term, or it could be that we do not get that far. Piloting a Linear Algebra course during a pandemic has never been done before!

Example 1.1 ***(two distinct real solutions)*** *Find the roots of* $2x^2 + 8x - 10 = 0$.

Solution:

$$\begin{aligned} a = 2, b = 8, c = -10 \\ b^2 - 4ac = 144 > 0 \\ x = \frac{-8 \pm \sqrt{144}}{4} \\ = \frac{-8 \pm 12}{4} \\ = -2 \pm 3 \end{aligned}$$

The two solutions are $x = 1$ and $x = -5$. ■

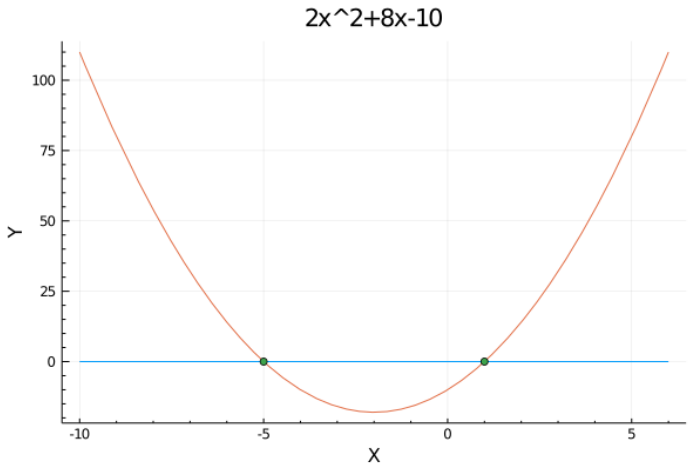


Figure 1.1: The two roots are at the intersection of the quadratic with the line $y = 0$. Why? Because, by definition, the roots are values of x where $ax^2 + bx + c$ equals zero!

Example 1.2 ***(two repeated real solutions)*** *Find the roots of* $2x^2 + 8x + 8 = 0$.

Solution:

$$\begin{aligned} a = 2, b = 8, c = 8 \\ b^2 - 4ac = 0 \\ x &= \frac{-8 \pm \sqrt{0}}{4} \\ &= \frac{-8 \pm 0}{4} \\ &= -2 \pm 0 \end{aligned}$$

The two solutions are $x = -2$ and $x = -2$. ■

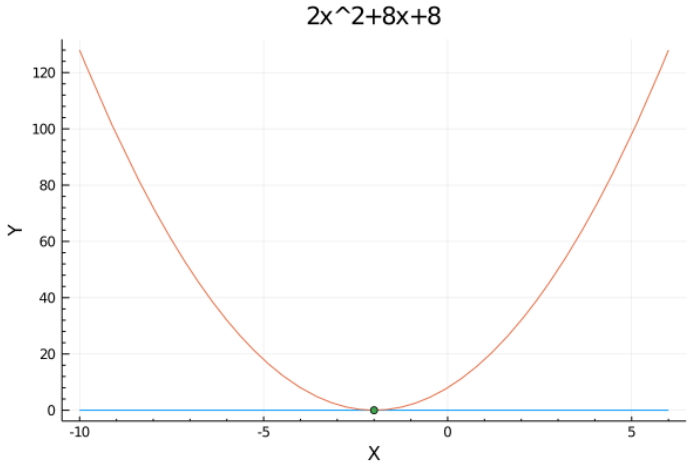


Figure 1.2: Note that there is now only a single intersection with $y = 0$ at $x = -2$. Because quadratic equations have two solutions, we say **the root at $\mathbf{x = -2}$ is repeated**.

Example 1.3 ***(two distinct complex solutions)*** *Find the roots of* $2x^2 + 8x + 10 = 0$.

Solution:

$$\begin{aligned} a = 2, b = 8, c = 10 \\ b^2 - 4ac &= -16 \\ x &= \frac{-8 \pm \sqrt{-16}}{4} \\ &= \frac{-8 \pm \sqrt{16}\sqrt{-1}}{4} \\ &= \frac{-8 \pm 4\sqrt{-1}}{4} \\ &= -2 \pm \sqrt{-1} \\ &= -2 \pm i \end{aligned}$$

The two solutions are $x = -2 + i$ and $x = -2 - i$, where $i := \sqrt{-1}$ is an *imaginary number*. If you have not already learned complex numbers, do not sweat it. If we need them at all in ROB 101, it will be at the end of the term and we will teach complex numbers before we use them! ■

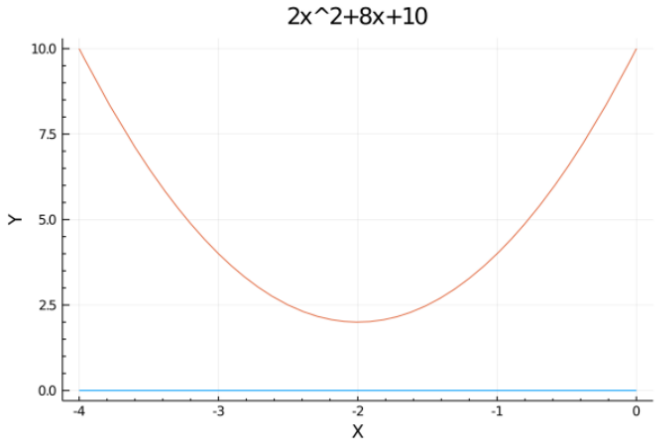


Figure 1.3: Note that there are no intersections with $y = 0$ and hence there are not any real solutions.

Remark: As in many subjects, the vocabulary in mathematics reflects the many twists and turns of history. If you are expecting mathematics to be a *pure science* in the sense that it is 100% logical, then you will often be disappointed! The Greeks and Egyptians called $1, 2, 3, \ldots$ *natural numbers* or *counting numbers* because they arose "naturally" in "counting objects": one cow, two sheep, three coins, four sacks of flour, etc. They were also comfortable with *fractions*, $\frac{m}{n}$, where both m and n were counting numbers, and hence, n could never be zero. Fractions were called *rational numbers* based on the word *ratio*. The square root of two, $\sqrt{2}$, was known to the Babylonians, Indians and Greeks. Our notion of $\sqrt{2}$ is probably thanks to Pythagoras, the Ionian Greek philosopher known to you for developing the Pythagorean Theorem relating the sides of right triangles, yes, the famous formula $a^2 + b^2 = c^2$, which gives $1^2 + 1^2 = (\sqrt{2})^2$, where the symbol $\sqrt{2}$ stands for the "quantity that when squared gives the counting number 2". It took a long time for the Greeks to accept that $\sqrt{2}$ could not be written as a *ratio of two counting numbers*. Eventually, it was proved to be *irrational*, that is, *not rational*, which means precisely that it cannot be expressed as the ratio of two counting numbers. In case you are interested, while Euclid was not the first to prove $\sqrt{2}$ was irrational, the beautiful reasoning he invented for the proof of $\sqrt{2}$ not being a rational number is still taught today. It is called *proof by contradiction*[1].

So far, so good with regards to vocabulary. But why call things involving $\sqrt{-1}$ complex numbers? Initially, mathematicians could not justify the existence of *quantities involving square roots of negative numbers.* Using them in calculations was a sign of "careless" mathematics and such numbers were treated as being *figments of one's imagination*, literally, *imaginary numbers*. Nevertheless, some (brave) mathematicians found them convenient for solving equations and others even for describing physical phenomena. Eventually, formal algebra caught up with the notion of imaginary numbers and their rules of use were rigorously justified. The name imaginary numbers stuck, however! Here is a link to a slightly simplified explanation of how mathematicians "codify" the existence of complex numbers: `http://www.math.toronto.edu/mathnet/answers/imagexist.html` Your instructors are unsure if they could have followed the reasoning in this document when they were at your stage of college, so do not sweat it.

Just for fun, you might want to learn more about numbers on Wikipedia `https://en.wikipedia.org/wiki/Number`. Were negative numbers always accepted? What about the notion of zero? **None of these remarks on numbers are required reading**. You will not be tested on the history of numbers!

To Know

- What are the counting numbers (also known as the natural numbers)?
- What are rational numbers?
- Be able to give an example of an irrational number, but of course, you are not expected to prove it is irrational.
- Later in the course (though it is not sure): what are imaginary numbers?

[1]It may seem remarkable to you that the two words "proof" and "contradiction" can coexist in logical statements. As you advance in your mathematical training, you may come across the term again. "Proof by contradcition" is not for amateurs...think about it as semi-professional-grade logic and math.

In ROB 101, the first ten Chapters focus on linear equations, hence, equations that do not have quadratic terms x^2, cubic terms x^3, nor terms with higher powers; they also do not have $\sin(x)$, $\cos(x)$, $\sqrt{x}$, or e^x or anything like that. It is perhaps hard to believe that equations with only order-one terms and constants could be interesting or important, but they are both interesting and important!

1.2 Linear Systems of Equations: What can happen?

We begin with several examples to illustrate what can happen.

Same number of equations as unknowns, and there is a unique solution:

$$\begin{aligned} x + y &= 4 \\ 2x - y &= -1 \end{aligned} \tag{1.1}$$

One way to compute a solution is to solve for x in terms of y in the first equation and then substitute that into the second equation,

$$\begin{aligned} x + y = 4 &\implies x = 4 - y \\ 2x - y = -1 &\implies 2(4 - y) - y = -1 \\ &\implies -3y = -9 \\ &\implies y = 3 \\ &\text{going back to the top} \\ x = 4 - y &\implies x = 1 \end{aligned}$$

You can try on your own solving for y in terms of x and repeating the above steps. You will obtain the same answer, namely $x = 1, y = 3$.
Another "trick" you can try, just to see if we can generate a different answer, is to add the first equation to the second, which will eliminate y,

$$\begin{aligned} x + y &= 4 \\ + \; 2x - y &= -1 \\ \hline 3x + 0y &= 3 \\ &\implies x = 1 \end{aligned}$$

Going back to the top and using either of the two equations

$$\begin{aligned} x + y = 4 &\implies y = 4 - x \\ &\implies y = 3 \\ &\text{or} \\ 2x - y = -1 &\implies -y = -2x - 1 \\ &\implies -y = -3 \\ &\implies y = 3 \end{aligned}$$

gives the same answer as before, namely, $x = 1, y = 3$.

In fact, the set of equations (1.1) has one, and only one, solution. In math-speak, one says the set of equations (1.1) has a *unique solution*. Often, we will stack x and y together and write the solution as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}.$$

Same number of equations as unknowns, and there is no solution:

$$\begin{aligned} x - y &= 1 \\ 2x - 2y &= -1 \end{aligned} \tag{1.2}$$

Because there are only two equations with a very nice set of numbers you might notice almost immediately that the left-hand side of the second equation is twice the left-hand side of the first equation, namely, $2x - 2y = 2(x - y)$, but when we look to the right-hand sides, $-1 \neq 2 \cdot 1$, and hence the equations (1.2) are *inconsistent*.

While the above is one way to analyze the problem, let's try to find a solution just as we did for the first set of equations,

$$\begin{aligned} x - y = 1 &\implies x = y + 1 \\ 2x - 2y = -1 &\implies 2(y + 1) - 2y = -1 \\ &\implies 2 + 2y - 2y = -1 \\ &\implies 2 = -1. \end{aligned}$$

Hence, trying to solve the equations has led us to a *contradiction*, namely $2 = -1$. Perhaps if we had solved for y in terms of x, we could have found a solution? Let's try

$$\begin{aligned} x - y = 1 &\implies -y = -x + 1 \\ &\implies y = x - 1 \\ 2x - 2y = -1 &\implies 2x - 2(x - 1) = -1 \\ &\implies 2x - 2x + 2 = -1 \\ &\implies 2 = -1 \end{aligned}$$

again! No matter how you manipulate the equations (1.2), while obeying "the rules of algebra" (which we have **not yet learned** for systems of linear equations), you cannot extract a sensible answer from these equations. They really are *inconsistent*.

Same number of equations as unknowns, and there are an infinite number of solutions:

$$\begin{aligned} x - y &= 1 \\ 2x - 2y &= 2 \end{aligned} \tag{1.3}$$

Because there are only two equations with a very nice set of numbers, you might notice that the left-hand side of the second equation is twice the left-hand side of the first equation, namely, $2x - 2y = 2(x - y)$, and this time, when we look to the right-hand sides, $2 = 2 \cdot 1$, and hence the two equations are actually the "same" in the sense that is one equation can be obtained from the other equation by multiplying both sides of it by a non-zero constant. We could elaborate a bit, but it's not worth it at this point. Instead, let's approach the solution just as we did in the first case.

We solve for x in terms of y in the first equation and then substitute that into the second equation,

$$\begin{aligned} x - y = 1 &\implies x = y + 1 \\ 2x - 2y = 2 &\implies 2(y + 1) - 2y = 2 \\ &\implies 2y + 2 - 2y = 2 \\ &\implies 2 = 2. \end{aligned}$$

The conclusion $2 = 2$, is perfectly correct, but tells us nothing about y. In fact, we can view the value of y as an arbitrary *free parameter* and hence the solution to (1.3) is

$$x = y + 1, \quad -\infty < y < \infty.$$

The solution can also be expressed as

$$y = x - 1, \quad -\infty < x < \infty,$$

which perhaps inspires you to plot the solution as a line in $\mathbb{R}^2$, with slope $m = 1$ and y-intercept $b = -1$.

Summary So Far

Consider a set of two equations with two unknowns x and y

$$\begin{aligned} a_{11}x + a_{12}y &= b_1 \\ a_{21}x + a_{22}y &= b_2, \end{aligned} \tag{1.4}$$

constants $a_{11}, a_{12}, a_{21}, a_{22}$ and b_1, b_2. Depending on the values of the constants, the linear equations (1.4) can have a unique solution, no solution, or an infinity of solutions.

More equations than unknowns typically means that there are no solutions: The system of equations

$$\begin{aligned} x &= 1 \\ y &= 2 \\ x + y &= a \end{aligned} \tag{1.5}$$

will only have a solution when $a = 3$. For all other values of a, it will not have a solution. We will learn to recognize later when the set of equations are consistent. At this point in the course, we do not have adequate mathematical tools to address the issue.

Limit of Hand Solutions

When there are only two equations and two unknowns, determining *by hand* if the equations have one solution, no solution, or an infinity of solutions is quite do-able. With sufficient motivation and "nice numbers", three equations and three unknowns is also not too bad. At four, it becomes super tedious and errors begin to sprout like weeds. Hence, what about 100 equations and 100 unknowns? **Our four-week goal is for you to handle systems of linear equations with hundreds of variables with confidence and ease** As you can imagine, this is where the "computational" part of ROB 101's name comes into play!

1.3 Naming Variables

If we have two variables (also called unknowns), it is natural to call them x and y. If we have three variables, naming them x, y, and z works fine. If we have 26 variables, would we start with a and go all the way to z? What if we have more variables? Well, there are 24 Greek letters, do we add them to the list? Throw in Mandarin Characters to get us to several thousand variables? Clearly, this is not a practical way to go. The only real possibility is to add counting numbers, because there are as many of them as we need, and computers understand numbers!

Welcome to $x_1, x_2, x_3, \ldots, x_n$, or $y_1, y_2, \ldots, y_m$ etc.

1.4 A 3 x 3 Example

Here is a *system of linear equations* with variables x_1, x_2, x_3.

$$\begin{aligned} x_1 + x_2 + 2x_3 &= 7 \\ 2x_1 - x_2 + x_3 &= 0.5 \\ x_1 + 4x_3 &= 7 \end{aligned} \tag{1.6}$$

The strategy for seeking a solution is the same as with two equations and two unknowns: solve for one of the variables in terms of the other variables, substitute into the remaining equations, simplify them, and repeat. We can start anywhere, so let's solve for x_1 in the bottom equation and plug that back into the two equations above it. Doing so gives us,

$$x_1 = 7 - 4x_3 \tag{1.7}$$

and then

$$\begin{aligned} \underbrace{(7 - 4x_3)}_{x_1} + x_2 + 2x_3 = 7 &\implies x_2 - 2x_3 = 0 \\ \underbrace{2(7 - 4x_3)}_{2x_1} - x_2 + x_3 = 0.5 &\implies -x_2 - 7x_3 = -13.5 \end{aligned}$$

$$\begin{aligned} x_2 - 2x_3 = 0 &\implies x_2 = 2x_3 \\ -x_2 - 7x_3 = -13.5 &\implies \underbrace{-(2x_3)}_{-x_2} - 7x_3 = -13.5 \\ &\implies -9x_3 = -13.5 \\ &\implies x_3 = 1.5 \end{aligned} \tag{1.8}$$

Plugging "$x_3 = 1.5$" into (1.7) gives

$$x_1 = 7 - 4 \cdot (1.5) = 7 - 6 = 1.$$

And then from (1.7), we have that

$$x_2 = 2x_3 = 2 \cdot (1.5) = 3.$$

Hence, the solution is

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 3.0 \\ 1.5 \end{bmatrix}. \tag{1.9}$$

Tedium and Pain $\Longrightarrow$ Motivation!

The hope is that you found the 3×3 example super tedious and something you'd prefer to avoid! The more tedious and painful you found it, the more motivation you will have to learn some new math that will make solving systems of linear equations a snap.

1.5 Greek Alphabet

It's hard to believe that the 26 Latin letters we know and love, $a, b, c, \ldots, x, y, z$, especially after being augmented with numerical subscripts, do not provide enough symbols! You might as well get used to it, **engineers and mathematicians often use the 24 Greek letters as well:** $\alpha, \beta, \gamma, \ldots, \chi, \psi, \omega$. Here is a link to the Greek alphabet with names of the letters and their pronunciations: https://web.mit.edu/jmorzins/www/greek-alphabet.html. 99.9% of all engineers, including your author, mispronounce them, so do not feel intimidated by that. Just gleefully join the club!

There are several accepted modern pronunciations of the Greek alphabet, it seems, giving you even less worry about your own pronunciation. The links below give a few of them:

1. Greek Alphabet - Pronunciation of each letter: https://www.youtube.com/watch?v=1FyEWbwBarQ
2. Greek Alphabet Rap Song: https://www.youtube.com/watch?v=w3D5ERMOpMk
3. Greek Alphabet Song (Nursery-rhyme Style): https://www.youtube.com/watch?v=3gaeIUsPJ-Y
4. Learn the Greek Alphabet in Less Than 10 Minutes: https://www.youtube.com/watch?v=BQVoz-HX2cA

If you lookup the origin of the word "AlphaBet", you will learn that it comes from Alpha Beta, the first two letters of the Greek Alphabet.

1.6 (Optional Read) Where is Computational Linear Algebra Used?

Figure 1.5 shows numerous areas where Computational Linear Algebra is used. In ROB 101, we have built projects around three different themes to give you a chance to really dive into a subject and feel that you learned something beyond the math itself:

- For **Project 1: Map Building from LiDAR Data** you will be given data collected on the Cassie Blue bipedal robot during an experiment on the North Campus Grove and are asked to transform the data for building a map and visualizing it using Julia. The underlying work you will do is very similar to what is done in real-time[2] on Cassie in order to build a map for autonomous navigation. You may wish to view the videos https://www.youtube.com/watch?v=pNyXsZ5zVZk and https://youtu.be/gE3Y-2Q3gco to see mapping and navigation done in real-time. Yes, this is very similar to what is done by AVs (Autonomous Vehicles). One difference is that an AV has 200 Kg of electronics in its trunk to process all the data, while Cassie's entire autonomy package weighs in at 9 Kg and runs off a hobbyist LiPo battery.

[2]Real-time means the computations are done quickly enough on the robot that they can be used almost immediately. This is opposed to "off-line" where you collect data on the robot and then do the processing on a desktop in the lab. You are clearly doing offline data processing, which is easier, because there are no computation time requirements.

- For **Project 2: Precipitation Data in Alaska (A True Story)**, you will use linear regression methods taught in Chap. **??** and apply them to a much larger dataset from the U.S. National Oceanic and Atmospheric Administration (NOAA). The project will give you insight into Machine Learning, one facet of the burgeoning area of Artificial Intelligence or AI for short. You will lean an important new function called the radial basis function and apply it towards building a mathematical model of a surface.
- For **Project 3: Feedback Control of a Segway**, you will learn how to balance and "drive" an inherently unstable mobile robot! If you have not seen a Segway, here is a video of the bideal robot Cassie Blue riding a Segway https://youtu.be/0gauVSUJzd0. Your project will not be this awesome, but the project will put you on the road to awesome things!

Now, in case just reading about these projects intimidates you, we want to assure you that their difficulty has been thoughtfully scaled to a first-semester Y1 experience, while also giving you insight into how the real things are done. This is possible because we will teach you math and programming in an integrated manner. The math reinforces the programming and the programming reinforces the math. At each step of the way, you will increase your confidence in Engineering, Mathematics, or the Sciences as a career choice. In High School, you were mostly taught math as being disconnected from real applications. There was a huge emphasis on memorizing formulas or theorems, without ever really using the concepts to do something fun. In ROB 101, we hope to show you how empowering it is to do mathematics at the scale of life.

1.7 Looking Ahead

We need a straightforward way to check whether a set of equations falls into the nice case of having a unique solution or is it one of the "problematic cases" where there may be no solution at all or an infinity of solutions. To get there, we need to learn:

- what are *matrices* and *vectors*;
- how to express a system of linear equations in terms of *matrices* and *vectors*;
- what it means for a set of linear equations to be *triangular* or *square*; and
- what is the *determinant* of a matrix.

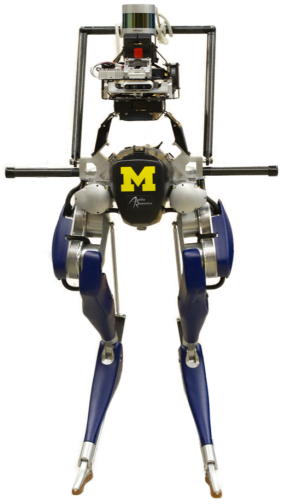
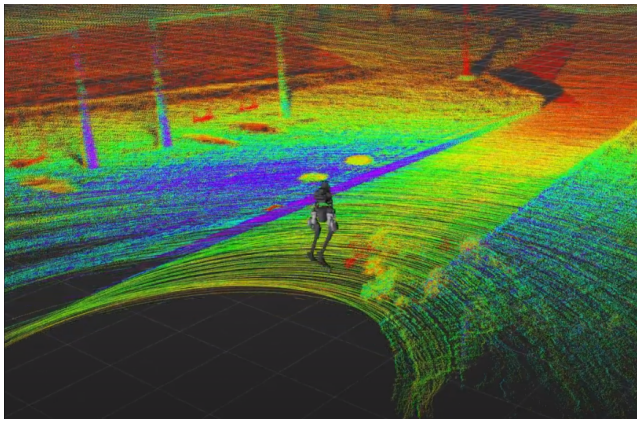
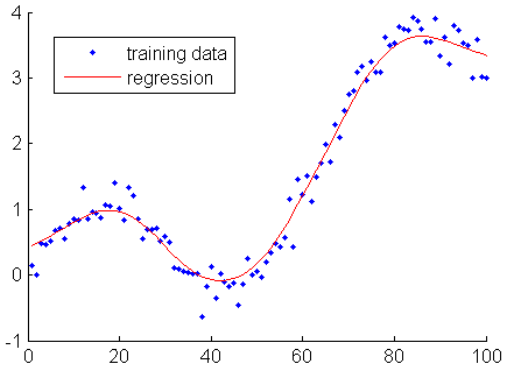


Figure 1.4: Cassie Blue with her perception package: a 32-Beam Velodyne LiDAR and an Intel RealSense Camera. In your first project, you will learn what is a LiDAR sensor and how to process the LiDAR data to build a "map" that a robot can use for navigation. Each colored dot in Fig. 1.5a is a LiDAR measurement. How many are there? Ten thousand, maybe? The image is the result of combining multiple LiDAR scans into a single image. You will learn that this involves applying matrices to vectors. You will learn about coordinate frames. Right now, this seems like magic. In a few weeks, you will be comfortable enough with the Julia programming language to manipulate a dataset with 10,000 points or more. **Remember, one of our goals is mathematics and programming at the scale of life!**

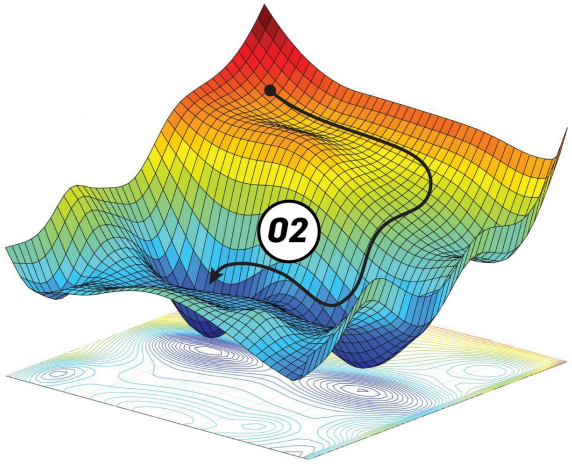


(a)

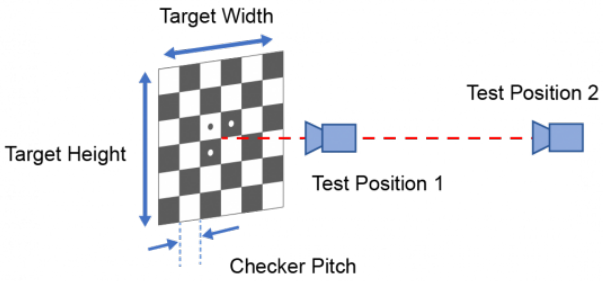


(b)

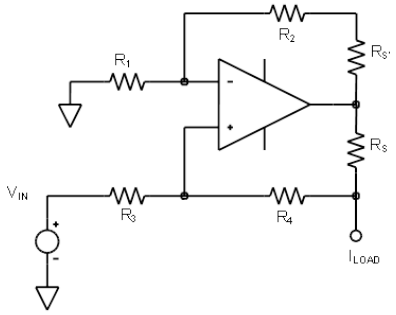
(c)



(d)



(e)



(f)

Figure 1.5: Just a few of the almost infinitely many application areas of linear algebra and programming.(a) Cassie Blue building a map on the UofM North Campus Grove. You will study this in Project 1. (b) Fitting a function to data allows one to model new situations that no one has seen before; image courtesy of Wikimedia Commons. You will study this in Project 2. (c) A Segway-like structure that requires a feedback loop to balance it; image courtesy of www.soarboards.com. You will study this in Project 3. (d) Gradient descent is used to find minima of functions, image courtesy of https://www.analyticsvidhya.com. You will study this in Chap. **??**. (e) Checkerboard or other regular patterns are used to calibrate cameras so that they can measure the position of objects in a scene; image courtesy of https://www.imatest.com/ (f) An electric circuit with a non-inverting amplifier that converts a voltage signal to a desired current, such as one might use to drive a motor or an electromagnet; image courtesy of https://wiki.analog.com/university/courses/electronics/text/chapter-4. Circuits are studied in EECS 215.

Chapter 2

Vectors, Matrices, and Determinants

Learning Objectives

- Begin to understand the vocabulary of mathematics and programming
- Answer the question: why make a distinction between integers and decimal numbers.
- An introduction to the most important tools of linear algebra: vectors and matrices.
- Find out an easy way to determine when a set of linear equations has a unique answer.

Outcomes

- Scalars vs Array
- Row vectors and column vectors
- Rectangular matrices and square matrices
- Learn new mathematical notation $x := y$, which means that x is by definition equal to y. You may be more used to $x \triangleq y$.
- Using matrices and vectors to express systems of linear equations
- Determinant of a square matrix and its relation to uniqueness of solutions of systems of linear equations

2.1 Scalars and Arrays

Scalars are simply numbers such as the ones you have been using for a long time: 25.77763, $\sqrt{17}, 10, -4, \pi$. In the Julia programming language, you will soon learn that for computational and storage efficiency, Julia differentiates between scalars that require decimal points and those that do not. Get ready for that! In ROB 101, when we do math, scalars are just numbers. When we do programming, scalars that do not require decimal points are called INTEGERS and those that do require decimal points are called FLOATING POINT NUMBERS because, with the same number of zeros and ones[1], Julia has to represent very large numbers such as 5.972×10^{24}, the mass of the earth in kilograms, and very small numbers, such as the mass of one atom of lead in kilograms $3.4406366 \times 10^{-22}$. To do that, where the decimal point appears in a list of zeros and ones has to "float", that is, it varies from number to number.

In ROB 101, we will not need to learn how computers represent numbers in *binary*. We will simply accept that computers use a different representation for numbers than we humans use. The more zeros and ones we allow in the representation of number, the more space it takes and the longer it takes to add them or multiply them, etc. The computer that provided navigational assistance for the moon landing on 20 July 1969 had a 16 bit word length, meaning its computations were based on groups of 16 binary digits (zeros and ones), called *bits*. In Julia, we'll typically use 64 zeros and ones to represent numbers:

- Int64
- Float64

Remark (can skip): In case you are wondering, ∞ (infinity) is a concept: it is NOT a number. Because it is not a number, ∞ is neither an integer nor a decimal number. The symbol ∞ is used to indicate that there is no positive upper bound for how big something can grow (i.e., it can get bigger and bigger and bigger . . .), while $-\infty$ is similar to indicate that something can get more and more negative without end. In ROB 101, you will only encounter ∞ when you try to divide something by zero. Julia has been programmed to rock your world when you do that! (Just kidding).

	A	B	C	D	E	F	G
1	36	0.157822	114.2628	2536	10675	4808	14122
2	36	0.157822	114.2628	2536	10672	4808	14122
3	36	0.157822	114.2628	2536	10672	4808	14122
4	36	0.157822	114.2628	2536	10672	4808	14089
5	35	0.153377	111.0451	2535	10672	4808	14061
6	35	0.153377	111.0451	2535	10672	4808	14001
7	34	0.148936	107.8298	2534	10672	4808	13982
8	33	0.144499	104.617	2533	10672	4808	13911
9	32	0.140065	101.4068	2532	10672	4808	13823
10	30	0.131207	94.99374	2530	10672	4808	13756
11	30	0.131207	94.99374	2530	10672	4808	13683
12	29	0.126783	91.79099	2529	10672	4808	13609
13	24	0.104717	75.81477	2524	10672	4808	13494
14	24	0.104717	75.81477	2524	10672	4808	13470
15	20	0.087125	63.07885	2520	10672	4808	13290
16	6	0.025992	18.81852	2506	10673	4808	13209
17	-1	-0.00432	-3.12766	2499	10670	4807	13155
18	-18	-0.07723	-55.9149	2482	10666	4806	13119
19	-27	-0.11543	-83.5682	2473	10666	4806	13059
20	-56	-0.23659	-171.294	2444	10666	4806	12965

Figure 2.1: Here is an array of numbers from a spreadsheet. The rows are numbered while the columns are labeled with letters.

[1]Zeros and ones are sometimes called the binary language of computers. If you are interested in binary numbers and binary arithmetic, you can find many sources on the web.

Arrays are scalars that have been organized somehow into lists. The lists could be rectangular or not, they could be made of columns of numbers, or rows of numbers. If you've ever used a spreadsheet, then you have seen an array of numbers. In Fig. 2.1, columns A, D, E, F, and G have only integer numbers in the them, while columns B and C use decimal numbers. Each row has both integer numbers and decimal numbers.

2.2 Row Vectors and Column Vectors

For us, a *vector* is a finite ordered list of numbers or of unknowns. In Julia, a vector is a special case of an *array*. For example

$$v = \begin{bmatrix} 1.1 \\ -3 \\ 44.7 \end{bmatrix} \tag{2.1}$$

is a vector of *length* three; sometimes we may call it a *3-vector*. By *ordered* we mean the list has a first element $v_1 = 1.1$, a second element $v_2 = -3$, and a third element $v_3 = 44.7$, and we also mean that if we change the order in which we list the elements, we (usually) obtain a different vector. For example, the vector

$$w = \begin{bmatrix} 1.1 \\ 44.7 \\ -3 \end{bmatrix} \tag{2.2}$$

is not equal to the vector v, even though they are made of the same set of three numbers.

Vectors written from "top" to "bottom" as in (2.1) and (2.2) are called *column* vectors. It is also useful to write vectors from "left" to "right" as in

$$v^{\text{row}} = \begin{bmatrix} 1.1 & -3 & 44.7 \end{bmatrix} \tag{2.3}$$

and we call them *row* vectors. What we said about the word "ordered" applies equally well to row vectors in that the row vector v has a first element $v_1^{\text{row}} = 1.1$, a second element $v_2^{\text{row}} = -3.0$, and a third element $v_3^{\text{row}} = 44.7$. Moreover, if we change the order in which we list the elements, we (usually) obtain a different row vector. Here, we were super careful and added the superscript $^{\text{row}}$ to v^{row} to clearly distinguish the symbol v being used for the column vector (2.1) from the row vector (2.3). Normally, we will not be that fussy with the notation, BUT, you must be aware that column vectors and row vectors are different "animals" except in the case that they have length one, as in $v = [v_1]$ is both a row vector and a column vector.

A general length n column vector is written like this,

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}, \tag{2.4}$$

while a general length n row vector is written like this

$$v = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix}. \tag{2.5}$$

This general notation allows for the case that $n = 1$, yielding

$$v = [v_1],$$

which as we noted above, is both a row vector and a column vector.

Here are some examples in Julia, thanks to Prof. Maani Ghaffari:

```
# we define an array of numbers.
# a is a 1x5 array
a = [1 -2 4 8.1 2^0.5]
```

Output

```
1×5 Matrix{Float64}:
 1.0  -2.0  4.0  8.1  1.41421
```

```
# b is a 5x1 array
b = [1, -2, 4, 8.1, 2^0.5]
```

Output

```
5-element Vector{Float64}:
  1.0
 -2.0
  4.0
  8.1
  1.4142135623730951
```


```
# b is a 5x1 array
b = [1, -2, 4, 8.1, 2^0.5]
```

Output

```
5-element Vector{Float64}:
  1.0
 -2.0
  4.0
  8.1
  1.4142135623730951
```


```
# or
c = [1; -2; 4; 8.1; 2^0.5]
```

Output

```
5-element Vector{Float64}:
  1.0
 -2.0
  4.0
  8.1
  1.4142135623730951
```

2.3 Remark on Brackets

Usually, in this course and in most books, *square brackets* [ ] are used to enclose vectors, but that is mainly *a matter of taste* as you can also use *parentheses* ( ) as in

$$v = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \tag{2.6}$$

and

$$v = \begin{pmatrix} v_1 & \cdots & v_n \end{pmatrix}. \tag{2.7}$$

In most programming languages, and Julia is no exception, you can only use square brackets! The choice of symbols, words, and their allowed arrangements in a language is called *syntax*. In a programming language, syntax is typically much more restrictive than in a written version of a spoken language. At some point, you may appreciate that throwing errors for bad syntax helps us to reduce *ambiguity* and *bugs* when we program. *You have likely experienced that ambiguities in spoken language can sometimes lead to bad outcomes.*

2.4 Matrices, Rectangular and Square, and the Matrix Diagonal

Matrices are generalizations of vectors that allow multiple columns and rows, where each row must have the same number of elements[2]. Here is a 3×2 matrix,

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, \tag{2.8}$$

meaning it has three rows and two columns, while here is a 2×3 matrix,

$$A = \begin{bmatrix} 1.2 & -2.6 & 11.7 \\ 3.1 & \frac{11}{7} & 0.0 \end{bmatrix}, \tag{2.9}$$

meaning it has two rows and three columns. It is customary to call a $1 \times n$ matrix a row vector and an $n \times 1$ matrix a column vector, but it is perfectly fine to call them matrices too!

A general $n \times m$ matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}, \tag{2.10}$$

is said to be *rectangular of size* $n \times m$ (one reads this as "n by m"), and when $n = m$, we naturally say that the $n \times n$ matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \tag{2.11}$$

is *square*. We note that a_{ij}, the ij-element of A, lies on the intersection of the i-th row and the j-th column.

Definition: The **diagonal** of the square matrix A in (2.11) is

$$\text{diag}(A) = \begin{bmatrix} a_{11} & a_{22} & \dots & a_{nn} \end{bmatrix} \tag{2.12}$$

What we are calling the diagonal is sometimes called the *main diagonal of a matrix*. We now highlight the diagonal in red to help those with a "visual memory"

$$A = \begin{bmatrix} \color{red}{a_{11}} & a_{12} & \cdots & a_{1n} \\ a_{21} & \color{red}{a_{22}} & \cdots & a_{2n} \\ \vdots & \vdots & \color{red}{\ddots} & \vdots \\ a_{n1} & a_{n2} & \cdots & \color{red}{a_{nn}} \end{bmatrix} \iff \text{diag}(A) = \begin{bmatrix} \color{red}{a_{11}} & \color{red}{a_{22}} & \dots & \color{red}{a_{nn}} \end{bmatrix}.$$

While it is possible to define the diagonal of general rectangular matrices, we will not do so at this time. Here are some examples in Julia, once again thanks to Prof. Maani Ghaffari

```
# Let's define a 3x2 matrix
A = [1 2; 3 4; 5 6]
```

Output

```
3×2 Array{Int64,2}:
 1  2
 3  4
 5  6
```

[2] Equivalently, each column has the same number of elements.

```
# Let's define a 2x3 matrix
A = [1.2 -2.6 11.7; 3.1 11/7 0]
```

Output

```
2×3 Array{Float64,2}:
 1.2  -2.6      11.7
 3.1   1.57143   0.0
```

```
# Here is a big matrix
using Random
A = randn(6,6)
```

Output

```
6×6 Matrix{Float64}:
 -0.518479   0.693952  -0.137698    0.200556   2.03276   -1.09174
  0.575811   0.999473  -0.0427593  -0.903915   2.14338    0.240728
  0.570255   0.477523  -0.503201   -0.864054  -0.661544   0.0821051
 -0.681514  -1.02591   -0.418878    0.248959  -0.776872   0.466698
  0.395184   1.72782    0.976437    1.10196    0.892258  -1.48822
  1.15146   -0.161273  -0.691775   -1.07168    0.909486  -1.02277
```

```
# compute its diagonal
using LinearAlgebra
diagA=diag(A)
```

Output

```
6-element Vector{Float64}:
 -0.5184785168661076
  0.9994728076152639
 -0.5032006177084569
  0.24895902626040853
  0.8922580664116776
 -1.022769405730864
```

2.5 Expressing a System of Linear Equations in terms of Vectors and Matrices

Before we even talk about "matrix-vector multiplication", we can address the task of writing a system of linear equations in "matrix-vector form". In the beginning, we will diligently follow the notation $Ax = b$, so let's see how we can identify a matrix A, a column vector x, and a column vector b. We'll do a few examples before giving a "general method".

Example 2.1 *Express the System of Linear Equations in Matrix Form:*

$$\begin{aligned} x_1 + x_2 &= 4 \\ 2x_1 - x_2 &= -1. \end{aligned} \tag{2.13}$$

Solution: When your instructors look at this equation, they see two unknowns x_1 and x_2 and *coefficients*[3] associated with them on the left-hand and right-hand sides of the two equations.

[3]Coefficients in this case are the numbers multiplying x_1 and x_2. An equations typically has variables (unknowns) and coefficients (numbers) that multiply the unknowns or that determine what the equation is supposed to equal, as in $3x_1 + 4x_2 = 7$.

We can use the unknowns to define a column vector of length two, the four coefficients multiplying the unknowns to build a 2×2 matrix A, and the two coefficients on the right-hand side to define another column vector of length two,

$$\begin{array}{c} x_1 + x_2 = 4 \\ 2x_1 - x_2 = -1 \end{array} \iff \underbrace{\begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 4 \\ -1 \end{bmatrix}}_{b}. \tag{2.14}$$

■

Example 2.2 *Express the System of Linear Equations in Matrix Form:*

$$\begin{array}{c} x_1 - x_2 = 1 \\ 2x_1 - 2x_2 = -1. \end{array} \tag{2.15}$$

Solution: We now jump straight into it. We form the 2×1 vector x of unknowns as before and place the coefficients by rows into the 2×2 matrix A,

$$\begin{array}{c} x_1 - x_2 = 1 \\ 2x_1 - 2x_2 = -1 \end{array} \iff \underbrace{\begin{bmatrix} 1 & -1 \\ 2 & -2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 1 \\ -1 \end{bmatrix}}_{b}. \tag{2.16}$$

■

Example 2.3 *Express the System of Linear Equations in Matrix Form:*

$$\begin{array}{c} 3x_1 + x_2 + 2x_3 = 7 \\ 2x_1 - x_2 + 4x_3 = 4 \end{array} \tag{2.17}$$

Solution: We can have the number of equations different from the number of unknown variables. In this case, we have more unknowns than equations.

$$\begin{array}{c} 3x_1 + x_2 + 2x_3 = 7 \\ 2x_1 - x_2 + 4x_3 = 4 \end{array} \iff \underbrace{\begin{bmatrix} 3 & 1 & 2 \\ 2 & -1 & 4 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 7 \\ 4 \end{bmatrix}}_{b} \tag{2.18}$$

■

From Multiple Linear Equations to Matrices and Vectors

We will work an example with three equations and four variables. The same method works for n equations in m variables.
Goal: Transform a system of linear equations into matrix form $Ax = b$

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= b_1 \\ 0a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 &= b_3. \end{aligned} \tag{2.19}$$

Two parts are really easy. We define two column vectors

$$x := \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \text{ and } b := \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}. \tag{2.20}$$

The vector x has four rows because there are $m = 4$ unknowns, x_1, x_2, x_3 and x_4. The vector b has three rows because there are $n = 3$ equations.

Now, what about the matrix A? First of all, it's size is 3×4 because it has one row for each equation and one column for each unknown. One way to think about is as follows. We place each term $a_{ij}x_j$ in the entry corresponding to the i-th row and j-th column. For example $a_{23}x_3$ goes in the second row and third column; we highlight it below

$$\begin{matrix} row_1 \\ row_2 \\ row_3 \end{matrix} \underbrace{\begin{bmatrix} a_{11}x_1 & a_{12}x_2 & a_{13}x_3 & a_{14}x_4 \\ a_{21}x_1 & a_{22}x_2 & a_{23}x_3 & a_{24}x_4 \\ a_{31}x_1 & a_{32}x_2 & a_{33}x_3 & a_{34}x_4 \end{bmatrix}}_{\begin{matrix} col_1 & col_2 & col_3 & col_4 \end{matrix}} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}. \tag{2.21}$$

We note that row_1 of the matrix has the terms for the first equation, row_2 has the terms for the second equation, etc, while col_1 is for the x_1 terms of each of the equations, col_2 is for the x_2 terms, etc. If we add up all of the entries in a given row, we obtain the corresponding equation.

Next, we move the x_i's out of the matrix to the right-hand side while leaving the coefficients where they are, like so

$$\begin{matrix} row_1 \\ row_2 \\ row_3 \end{matrix} \underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}}_{\begin{matrix} col_1 & col_2 & col_3 & col_4 \end{matrix}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \iff \begin{matrix} Eq_1 \\ Eq_2 \\ Eq_3 \end{matrix} \underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}}_{\begin{matrix} var_1 & var_2 & var_3 & var_4 \end{matrix}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{2.22}$$

to obtain

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 &= b_3. \end{aligned} \iff \underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}}_{b} \tag{2.23}$$

Remark: After a bit of practice, you will go straight from (2.19) to (2.22) (in other words, from the left-hand side of (2.23) to the right-hand side of (2.23)), without passing through the intermediate step given in (2.21).

Example 2.4 *Express the System of Linear Equations in Matrix Form:*

$$\begin{aligned} x_1 + x_2 + 2x_3 &= 7 \\ 2x_1 - x_2 + x_3 &= 0.5 \\ x_1 + 4x_3 &= 7 \end{aligned} \tag{2.24}$$

Solution: Note that we treat "any missing coefficients" (as in the "missing x_2" in the third equation) as being zeros! This is super

important to remember.

$$\begin{aligned} x_1 + x_2 + 2x_3 &= 7 \\ 2x_1 - x_2 + x_3 &= 0.5 \\ x_1 + 4x_3 &= 7 \end{aligned} \iff \underbrace{\begin{bmatrix} 1 & 1 & 2 \\ 2 & -1 & 1 \\ 1 & \boxed{0} & 4 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 7 \\ 0.5 \\ 7 \end{bmatrix}}_{b} \tag{2.25}$$

■

Example 2.5 *We redo Example 2.4 with the equations in a different order*

$$\begin{aligned} x_1 + 4x_3 &= 7 \\ x_1 + x_2 + 2x_3 &= 7 \\ 2x_1 - x_2 + x_3 &= 0.5 \end{aligned} \tag{2.26}$$

Solution: Once again, we treat "any missing coefficients" (as in the "missing x_2" in the first equation) as being zeros! This is super important to remember.

$$\begin{aligned} x_1 + \boxed{0x_2} + 4x_3 &= 7 \\ x_1 + x_2 + 2x_3 &= 7 \\ 2x_1 - x_2 + x_3 &= 0.5 \end{aligned} \iff \underbrace{\begin{bmatrix} 1 & \boxed{0} & 4 \\ 1 & 1 & 2 \\ 2 & -1 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 7 \\ 7 \\ 0.5 \end{bmatrix}}_{b} \tag{2.27}$$

■

Example 2.6 *(Optional Read) We redo Example 2.4 with the equations* **and variables** *in a different order*

$$\begin{aligned} x_1 + x_2 + 2x_3 &= 7 \\ x_1 + 4x_3 &= 7 \\ 2x_1 - x_2 + x_3 &= 0.5 \end{aligned} \tag{2.28}$$

Solution: This time, just to drive home the point that you can place the variables in any order that you wish, we will define x by

$$x_{\text{odd}} = \begin{bmatrix} x_2 \\ x_3 \\ x_1 \end{bmatrix}.$$

With the variables x_1, x_2, and x_3 arranged in this deliberately strange order, the matrices become

$$\begin{aligned} x_1 + x_2 + 2x_3 &= 7 \\ x_1 + \boxed{0x_2} + 4x_3 &= 7 \\ 2x_1 - x_2 + x_3 &= 0.5 \end{aligned} \iff \underbrace{\begin{bmatrix} 1 & 2 & 1 \\ \boxed{0} & 4 & 1 \\ -1 & 1 & 2 \end{bmatrix}}_{A_{\text{odd}}} \underbrace{\begin{bmatrix} x_2 \\ x_3 \\ x_1 \end{bmatrix}}_{x_{\text{odd}}} = \underbrace{\begin{bmatrix} 7 \\ 7 \\ 0.5 \end{bmatrix}}_{b} \tag{2.29}$$

Re-ordering the components of x re-orders the columns of A. ■

How to Handle "Missing" Variables or Coefficients

A zero in row i and column j of a matrix corresponds to the variable x_j being absent from the i-th equation

$$\underbrace{\begin{bmatrix} 0 & 0 & 2 \\ 2 & 0 & 0 \\ 0 & -1 & 4 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 7 \\ 0.5 \\ 7 \end{bmatrix}}_{b} \iff \begin{aligned} 2x_3 &= 7 \\ 2x_1 &= 0.5 \\ -x_2 + 4x_3 &= 7 \end{aligned} \iff \underbrace{\begin{aligned} 0x_1 + 0x_2 + 2x_3 &= 7 \\ 2x_1 + 0x_2 + 0x_3 &= 0.5 \\ 0x_1 - x_2 + 4x_3 &= 7 \end{aligned}}_{\text{we'd never write it like this}} \tag{2.30}$$

2.6 The Matrix Determinant

One of the more difficult topics in *Linear Algebra* is the notion of the determinant of a matrix. Most people who claim to understand it are wrong. Instead of teaching you right away a bunch of details of the matrix determinant that are rather useless in the actual practice of Linear Algebra, we are going to cut directly to the "good stuff". Moreover, we'll only give you part of the good stuff here, and save some "really excellent stuff" for Chapter **??**.

2.6.1 First Contact with the Matrix Determinant

What you want and need to know about the determinant function: the first five points are extra important.

Enough Facts about the Determinant to Get Us Going

Fact 1 The determinant of a square matrix A is a real number, denoted $\det(A)$.

Fact 2 A square system of linear equations (i.e., n equations and n unknowns), $Ax = b$, has a unique solution x for any $n \times 1$ vector b if, and only if, $\det(A) \neq 0$.

Fact 3 When $\det(A) = 0$, the set of linear equations $Ax = b$ may have either no solution or an infinite number of solutions. To determine which case applies (and to be clear, only one case can apply), depends on how "b is related to A", which is another way of saying that we will have to address this later in the course.

Fact 4 The determinant of the 1×1 matrix $A = [a]$ is $\det(A) := a$.

Fact 5 The determinant of the 2×2 square matrix $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is $\det(A) := ad - bc$.

Fact 6 The determinant is only defined for square matrices; see Fact 1.

Fact 7 We will learn later how to compute by hand the determinant of $n \times n$ matrices with special structure, but for general matrices, you only need to know how to *compute by hand the determinant of* a 2×2 matrix.

Remark: In case you are "sad" that you can only compute determinants for 2×2 matrices, do not worry. *Very soon, you will also be able to do it for a* 100×100 *matrix and understand what you are doing.* In this book, we take a "structural approach" to Computational Linear Algebra in general, and to the matrix determinant in particular. In Chap. 3, we will first focus on matrices with a special "triangular" structure and then we'll see how to "factor" or "decompose" a non-triangular matrix into (the product of) two triangular matrices. This structural approach has many advantages when it comes to scaling up computations to problems with hundreds or even thousands of variables. In particular, it will provide a practical way of understanding and computing the matrix determinant for really big matrices.

2.6.2 Examples of Using the Determinant

We reuse some previous examples and add in a few more to illustrate Fact 2 of Sec. 2.6.1

Example 2.7 *Check for existence and uniqueness of solutions:*

$$\begin{aligned} x_1 + x_2 &= 4 \\ 2x_1 - x_2 &= -1 \end{aligned} \iff \underbrace{\begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 4 \\ -1 \end{bmatrix}}_{b}. \tag{2.31}$$

Solution: We compute $\det(A) = (1) \cdot (-1) - (1) \cdot (2) = -3 \neq 0$ and conclude that (2.31) has a unique solution. ■

Example 2.8 *Check for existence and uniqueness of solutions:*

$$\begin{aligned} x_1 - x_2 &= 1 \\ 2x_1 - 2x_2 &= -1 \end{aligned} \iff \underbrace{\begin{bmatrix} 1 & -1 \\ 2 & -2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 1 \\ -1 \end{bmatrix}}_{b} \tag{2.32}$$

Solution: We compute $\det(A) = (1) \cdot (-2) - (-1) \cdot (2) = 0$. We therefore conclude that (2.32) does not have a unique solution. In fact, it may have either no solution at all or it may have an infinite number of solutions. At this point in the course, we do not have the tools to determine which case applies here without grinding through the equations. Referring back to (1.2), where we did grind through the equations, we know that this system of linear equations has no solution. ■

Example 2.9 *Check for existence and uniqueness of solutions:*

$$\begin{aligned} x_1 - x_2 &= 1 \\ 2x_1 - 2x_2 &= 2 \end{aligned} \iff \underbrace{\begin{bmatrix} 1 & -1 \\ 2 & -2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 1 \\ 2 \end{bmatrix}}_{b} \tag{2.33}$$

Solution: We compute $\det(A) = (1)\cdot(-2) - (1)\cdot(2) = 0$. We therefore conclude that (2.33) does not have a unique solution. In fact, it may have either no solution at all or it may have an infinite number of solutions. At this point in the course, we do not have the tools to determine which case applies here without grinding through the equations. Referring back to (1.3), we know that this system of linear equations has an infinite number of solutions. ■

Example 2.10 *Check for existence and uniqueness of solutions:*

$$\begin{aligned} x_1 + x_2 + 2x_3 &= 7 \\ 2x_1 - x_2 + x_3 &= 0.5 \\ x_1 + 4x_3 &= 7 \end{aligned} \iff \underbrace{\begin{bmatrix} 1 & 1 & 2 \\ 2 & -1 & 1 \\ 1 & 0 & 4 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 7 \\ 0.5 \\ 7 \end{bmatrix}}_{b} \tag{2.34}$$

Solution: Using Julia, we compute $\det(A) = -9 \neq 0$. We therefore conclude that (2.34) has a unique solution for x.

```
# using LinearAlgebra
A=[1 1 2; 2 -1 1; 1 0 4]
det(A)
```

Output

```
-9.0
```

■

Example 2.11 *Check for existence and uniqueness of solutions in an example where the unknowns are not in the "correct" order and we have a bunch of "missing coefficients":*

$$\begin{aligned} x_1 + x_2 + 2x_3 &= 7 \\ -x_2 + x_3 + 2x_1 &= 0.5 \\ x_1 + 4x_3 &= 7 \\ x_4 + 2x_3 - 5x_5 - 11 &= 0 \\ -4x_2 + 12x_4 &= 0 \end{aligned} \iff \underbrace{\begin{bmatrix} 1 & 1 & 2 & 0 & 0 \\ 2 & -1 & 1 & 0 & 0 \\ 1 & 0 & 4 & 0 & 0 \\ 0 & 0 & 2 & 1 & -5 \\ 0 & -4 & 0 & 12 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 7 \\ 0.5 \\ 7 \\ 11 \\ 0 \end{bmatrix}}_{b} \tag{2.35}$$

Solution: Using Julia, one computes $\det(A) = -540 \neq 0$. We therefore conclude that (2.35) has a unique solution. Grinding through the equations would have been no fun at all!

```
A=[1 1 2 0 0; 2 -1 1 0 0; 1 0 4 0 0; 0 0 2 1 -5; 0 -4 0 12 0]
det(A)
```

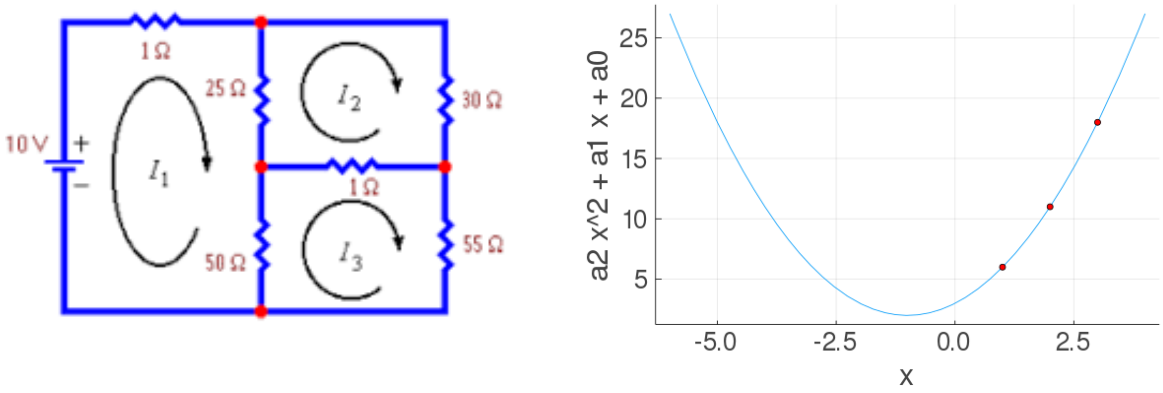
Output

```
-540.0
```

■

2.7 (Optional Read): Other Facts on the Matrix Determinant Covered in "Standard" Courses

Here are some other facts that are sometimes useful but are not required in ROB 101.



(a) (b)

Figure 2.2: (a) Image borrowed from the Khan Academy. A simple electric circuit as you would learn to analyze in the first month of EECS 215 using Kirchhoff's Current and Voltage Laws, which can be thought of as the electrical versions of Newton's equations for balancing forces. (b) A quadratic function that you need to find based upon someone having measured for you the three values given in red! Can you do it? Does it matter that the data were selected from only one side of the parabola?

- In written mathematics, but not in programming, you will also encounter the notation $\det(A) = |A|$, so that

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

- In Julia, once you state that you are using the linear algebra package, that is `using LinearAlgebra`, the command is `det(A)` , if A is already defined, or, for example, `det([1 2 3; 4 5 6; 7 8 9])`
- The determinant of a 3×3 matrix is

$$\det\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}\right) := a \cdot \det\left(\begin{bmatrix} e & f \\ h & i \end{bmatrix}\right) - b \cdot \det\left(\begin{bmatrix} d & f \\ g & i \end{bmatrix}\right) + c \cdot \det\left(\begin{bmatrix} d & e \\ g & h \end{bmatrix}\right). \tag{2.36}$$

  In ROB 101, you do not need to use, much less memorize, the formula (2.36).

Optional, and for sure, skip on your first read: Khan Academy

Here are the (Tiny URLs) of two videos by the Khan Academy that provide a "clasical" introduction to the matrix determinant.

- `https://tinyurl.com/gqt93l3` works a 3×3 example based on (2.36).
- `https://tinyurl.com/zhlwb5v` shows an alternative way to compute the determinant of a 3×3 matrix.
- **In ROB 101**, you do not need to use either of these methods.

2.8 (Optional Read): A Few "Practical" Examples Using Linear Algebra

You are not responsible for any of these examples. They are given here to provide a sense of how vectors and matrices are used in engineering. Because we are so early in the course, the examples we can work are rather boring and are not on par with what you will do in the projects.

Example 2.12 ***[Circuit Equations]*** *Write the equations satisfied by the currents I_1, I_2, and I_3 in Fig. 2.2a and then solve them.*

Solution: Once you have taken EECS 215, you would quickly write down the following equations which are based on three facts called Kirchhoff's Current and Voltage Laws:

- the sum of the voltages around any loop in the circuit is zero;
- current I flowing through resistor R creates a voltage $V = IR$; and
- when two loops touch, as for the 25 Ω resistor, the currents add with their sign determined by their directions.

At the 25 Ω resistor, for example, the current is $I_1 - I_2$ because they flow in opposite directions. In short, after taking EECS 215 and applying Kirchhoff's Current and Voltage Laws, you would obtain

$$\begin{aligned} -10 + I_1 + 25(I_1 - I_2) + 50(I_1 - I_3) &= 0 \\ 25(I_2 - I_1) + 30I_2 + (I_2 - I_3) &= 0 \\ 55I_3 + 50(I_3 - I_1) + (I_3 - I_2) &= 0 \end{aligned} \iff \underbrace{\begin{bmatrix} 76 & -25 & -50 \\ -25 & 56 & -1 \\ -50 & -1 & 106 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 10 \\ 0 \\ 0 \end{bmatrix}}_{b} \tag{2.37}$$

While it is not important to you, the first equation is obtained by adding up the voltages in the loop with I_1, proceeding in the direction of I_1, the second equation is obtained by adding up the voltages in the loop with I_2, proceeding in the direction of I_2, and similarly for the last equation. In Julia, we compute that $\det(A) = 242,310$, kind of a wild number, but it's non-zero. Solving the equation results in

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} 0.2449 \\ 0.1114 \\ 0.1166 \end{bmatrix},$$

in units of Amperes in case you are curious! ■

Example 2.13 ***[Fitting a Function to Data]*** *Figure 2.2b shows a quadratic function of the form*

$$y = a_2x^2 + a_1x + a_0. \tag{2.38}$$

You are given the data in Table 2.1 and need to find the coefficients a_2, a_1, and a_0 that define the quadratic.

Table 2.1: Data for a Quadratic Function.

x	y
1	6
2	11
3	18

Solution: Our unknowns are the coefficients a_2, a_1, and a_0. Hence, we rewrite the quadratic as

$$y = a_2x^2 + a_1x + a_0 = x^2a_2 + xa_1 + a_0,$$

and note that this equation is linear in the unknowns. From Table 2.1, we have the following equations

$$\begin{aligned} 6 &= a_2 + a_1 + a_0 \\ 11 &= 4a_2 + 2a_1 + a_0 \\ 18 &= 9a_2 + 3a_1 + a_0 \end{aligned} \iff \underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 6 \\ 11 \\ 18 \end{bmatrix}}_{b} \tag{2.39}$$

Using Julia, we compute $\det(A) = -2 \neq 0$ and hence a solution to (2.39) exists and is unique. Using our current methods, we compute that the answer is

$$\begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 2.0 \\ 3.0 \end{bmatrix}.$$

In other words, the data are compatible with the function $y = x^2 + 2x + 3$. ■

Example 2.14 ***[Fitting a Function to Data: Take 2]*** *We re-visit the quadratic function in Fig. 2.2b with unknown coefficients a_2, a_1, and a_0, namely $y = a_2x^2 + a_1x + a_0$. This time, however, we use the data in Table 2.2*

Table 2.2: Data for a Quadratic Function.

x	y
-3	6
1	6
2	11
3	18

Solution: Our unknowns are once again the coefficients a_2, a_1, and a_0. We rewrite the quadratic as

$$y = a_2x^2 + a_1x + a_0 = x^2a_2 + xa_1 + a_0,$$

and note again that this equation is linear in the unknowns. From Table 2.2, we have the following equations

$$\begin{aligned} 6 &= 9a_2 - 3a_1 + a_0 \\ 6 &= a_2 + a_1 + a_0 \\ 11 &= 4a_2 + 2a_1 + a_0 \\ 18 &= 9a_2 + 3a_1 + a_0 \end{aligned} \iff \underbrace{\begin{bmatrix} 9 & -3 & 1 \\ 1 & 1 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 6 \\ 6 \\ 11 \\ 18 \end{bmatrix}}_{b} \tag{2.40}$$

In this case, we have more equations than unknowns, which translates into A being non-square. Because A is non-square, we cannot compute its determinant and then use the answer to determine whether solutions exist or are unique, which is kind of a bummer, because it kind of says that more data is not necessarily better! Later in the course, we will be able to confront this issue head on and learn that, yes, in most cases, more data is better. For now, we punt! ■

2.9 (Optional Read) Yet Another Determinant Example

In the following examples, the determinant is computed using the method of (2.36). **You are not required to know this method because you would never want to use it on a** 10×10 **matrix, for example,** whereas we will learn a method that scales easily to matrices that are 100×100. **The method in (2.36) is only illustrated here to prove a point: it's painful.**

Example 2.15 *Compute the determinant of the matrix below using the method that is normally taught in Linear Algebra, namely,* (2.36)*.*

$$A = \begin{bmatrix} 4 & -1 & 1 \\ 4 & 5 & 3 \\ -2 & 0 & 0 \end{bmatrix}$$

Solution:

$$\begin{aligned} \det(A) &= \begin{vmatrix} 4 & -1 & 1 \\ 4 & 5 & 3 \\ -2 & 0 & 0 \end{vmatrix} \\ &= 4 \cdot \begin{vmatrix} 5 & 3 \\ 0 & 0 \end{vmatrix} - (-1) \cdot \begin{vmatrix} 4 & 3 \\ -2 & 0 \end{vmatrix} + 1 \cdot \begin{vmatrix} 4 & 5 \\ -2 & 0 \end{vmatrix} \\ &\text{(after applying our determinant formula to each of the 2 x 2 matrices above)} \\ &= 4 \cdot (0) + 1 \cdot (6) + 1 \cdot 10 \\ &= 16. \end{aligned}$$

■

For 3×3 it's not so bad, but already at 4×4, it becomes tedious.

Example 2.16 *Compute the determinant of the matrix below using the method that is normally taught in Linear Algebra.*

$$A = \begin{bmatrix} 4 & -1 & 1 & 2 \\ 4 & 5 & 3 & 7 \\ -2 & 0 & 0 & 4 \\ -2 & 8 & 1 & 4 \end{bmatrix}$$

Solution:

$$\begin{aligned} \det(A) &= \begin{vmatrix} 4 & -1 & 1 & 2 \\ 4 & 5 & 3 & 7 \\ -2 & 0 & 0 & 4 \\ -2 & 8 & 1 & 4 \end{vmatrix} \\ &= 4 \cdot \begin{vmatrix} 5 & 3 & 7 \\ 0 & 0 & 4 \\ 8 & 1 & 4 \end{vmatrix} - (-1) \begin{vmatrix} 4 & 3 & 7 \\ -2 & 0 & 4 \\ -2 & 1 & 4 \end{vmatrix} + (1) \begin{vmatrix} 4 & 5 & 7 \\ -2 & 0 & 4 \\ -2 & 8 & 4 \end{vmatrix} - (2) \begin{vmatrix} 4 & 5 & 3 \\ -2 & 0 & 0 \\ -2 & 8 & 1 \end{vmatrix} \\ &= \text{(after applying our determinant formula to each of the 3 x 3 matrices above)} \\ &= 4(76) + 1(-30) + 1(-240) - 2(-38) \\ &= 110. \end{aligned}$$

■

Once again, the point is not to be understand why this method works or how to do it. The point is how unwieldy it is. Since you have read this far, we'll let you in on a secret. In Chapter **??**, we'll learn how to "factor" or "decompose" the matrix A above into two matrices called

$$L = \begin{bmatrix} 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ -0.5000 & 1.0000 & 0.0000 & 0.0000 \\ 1.0000 & 0.8000 & 1.0000 & 0.0000 \\ -0.5000 & -0.0667 & 0.7500 & 1.0000 \end{bmatrix} \text{ and } U = \begin{bmatrix} 4.0 & -1.0 & 1.0 & 2.0 \\ 0.0 & 7.5 & 1.5 & 5.0 \\ 0.0 & 0.0 & 0.8 & 1.0 \\ 0.0 & 0.0 & 0.0 & 55/12 \end{bmatrix}.$$

In Chapter **??**, we'll learn that all of the information concerning the determinant is contained in U and that $\det(A) = \det(U) = 4 \times 7.5 \times 0.8 \times 55/12 = 110.0$, the product of the terms on the diagonal of U. How mind blowing is that?

2.10 Looking Ahead

We want to solve very large sets of linear equations, say more than 100 variables. We could teach you the matrix inverse command in Julia, but then you'd understand nothing. In the next chapter, we will begin exploring how to solve problems with special structure. Soon after that, we'll show how to transform all linear systems of n-equations in n-variables to a form where they are solvable with "special strucutre". To get there we need to understand:

- what are *square and triangular matrices*;
- what is *forward* and *back substitution*;
- what does it mean to multiply two matrices; and
- how to *factor* a square matrix as the product of two triangular matrices.

Help! Help! How am I supposed to remember all of this?

You probably can't. In any case, we don't want you to memorize the ROB 101 material. Instead, open up a `google doc or google sheet` and make notes! You need an organized method for keeping track of stuff. In High School, you may have been able to remember all the new notation without any special effort. In College, it's a bit different.

Chapter 3

Triangular Systems of Equations: Forward and Back Substitution

Learning Objectives

- Equations that have special structure are often much easier to solve
- Some examples to show this.

Outcomes

- Recognize triangular systems of linear equations and distinguish those with a unique answer.
- Learn that the determinant of a square triangular matrix is the product of the terms on its diagonal.
- How to use forward and back substitution.
- Swapping rows of equations and permutation matrices.

3.1 Background

- A system of linear equations is **square** if it has the same number of equations as unknowns.
- If the system of linear equations is expressed in the form $Ax = b$, then it is **square** if, and only if, A has the same number of rows as it has columns; that is, A is $n \times n$.
- A square system of linear equations $Ax = b$, has a **unique solution** x for any $n \times 1$ vector b if, and only if, the $n \times n$ matrix A satisfies $\det(A) \neq 0$.
- Computing the **determinant** of a general square matrix is tedious.
- The **diagonal** of the square matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

  is

$$\operatorname{diag}(A) = \begin{bmatrix} a_{11} & a_{22} & \ldots & a_{nn} \end{bmatrix}.$$

  The diagonal consists of all elements a_{ij} of A such that $j = i$.
- The elements of A that are **not on the diagonal** are called the **off-diagonal elements**.

3.2 A Warm-up with Diagonal Systems of Linear Equations

As a warm-up to the main topic, we want to illustrate that some systems of linear equations are easy to solve, independently of the number of unknowns. We'll start small, and then go big!

This is an example of a square system of linear equations that is *Diagonal*,

$$\begin{aligned} 4x_1 &= 6 \\ -x_2 &= 7 \\ 3x_3 &= 2. \end{aligned} \tag{3.1}$$

Computing the solution of the set of diagonal equations is trivial because we just have to divide each row of the equation by the element multiplying the unknown, namely

$$\begin{aligned} 4x_1 &= 6 \\ -x_2 &= 7 \\ 3x_3 &= 2. \end{aligned} \iff \begin{aligned} x_1 &= \frac{6}{4} = \frac{3}{2} \\ x_2 &= \frac{7}{-1} = -7 \\ x_3 &= \frac{2}{3}. \end{aligned} \tag{3.2}$$

In this example, all of the constants multiplying the unknowns were non-zero. If one of them had been zero, for example, if the second row were $0x_2 = 7$, then the equations would not have a solution; if the second row were $0x_2 = 0$, then the equations would have an infinite number of solutions because x_2 could take on any value.

When we write the system as $Ax = b$, we have

$$\begin{aligned} 4x_1 &= 6 \\ -x_2 &= 7 \\ 3x_3 &= 2. \end{aligned} \iff \underbrace{\begin{bmatrix} 4 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 3 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 6 \\ 7 \\ 2 \end{bmatrix}}_{b}. \tag{3.3}$$

where we note that all of the off-diagonal terms of the matrix A are zero. More precisely, the condition is $a_{ij} = 0$ for all $i \neq j$. Such matrices are called *Diagonal Matrices*.

The corresponding system of linear equations is easy to solve because each of the coefficients $a_{11} = 4$, $a_{22} = -1$, and $a_{33} = 3$ multiplying the unknowns x_1, x_2, and x_3 is non-zero. For later use, we note that n real numbers a_{11}, a_{22}, ..., a_{nn} are all non-zero if, and only if, their product is non-zero, that is,

$$a_{11} \neq 0, a_{22} \neq 0, \ldots, a_{nn} \neq 0 \iff a_{11}a_{22}\cdots a_{nn} \neq 0.$$

This result is true because the product to any two real numbers is zero if, and only if, at least one of the numbers is zero.

More generally, a square $n \times n$ matrix A where all of the **off-diagonal** elements are zero is said to be a **diagonal matrix**,

$$A = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix};$$

in other words, if $a_{ij} = 0$ for all $i \neq j$, then A is diagonal. Computing the **determinant** of a diagonal matrix is very easy,

$$\det(A) = \begin{vmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{vmatrix} = a_{11}a_{22}\cdots a_{nn},$$

the product of the terms on its diagonal.

A diagonal system of n linear equations

$$\begin{aligned} a_{11}x_1 &= b_1 \\ a_{22}x_2 &= b_2 \\ \vdots \quad &= \vdots \\ a_{nn}x_n &= b_n \end{aligned} \iff \underbrace{\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_{b} \tag{3.4}$$

is straightforward to solve, no matter its size, because, whenever $a_{ii} \neq 0$, we have

$$x_i = \frac{b_i}{a_{ii}}. \tag{3.5}$$

If one of the coefficients a_{ii} on the diagonal is zero, then, if the corresponding element b_i is non-zero, we deduce that the equations do not have a solution, and if the corresponding element b_i is zero, we deduce that the equations can have an infinite number of solutions. This is identical to the discussion below (3.2).

For diagonal matrices, it follows that if $\det(A) \neq 0$, then $Ax = b$ has a unique solution; moreover the solution is very easy to compute via (3.5). In the next Section, we explore other types of matrices that make computing a solution very fast and easy.

3.3 Lower Triangular Systems of Linear Equations

This is an example of a square system of linear equations that is *Lower Triangular*

$$\begin{aligned} 3x_1 &= 6 \\ 2x_1 - x_2 &= -2 \\ x_1 - 2x_2 + 3x_3 &= 2. \end{aligned} \tag{3.6}$$

When we write the system as $Ax = b$, in the lower triangular case we have

$$\begin{aligned} 3x_1 &= 6 \\ 2x_1 - x_2 &= -2 \\ x_1 - 2x_2 + 3x_3 &= 2 \end{aligned} \iff \underbrace{\begin{bmatrix} 3 & 0 & 0 \\ 2 & -1 & 0 \\ 1 & -2 & 3 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 6 \\ -2 \\ 2 \end{bmatrix}}_{b}. \tag{3.7}$$

where we note that all terms "above" the diagonal of the matrix A are zero. More precisely, the condition is $a_{ij} = 0$ for all $j > i$. Such matrices are called *lower-triangular*.

Here are two more examples of square lower triangular systems

$$\begin{aligned} 3x_1 &= 6 \\ 2x_1 - x_2 &= 0 \end{aligned} \iff \underbrace{\begin{bmatrix} 3 & 0 \\ 2 & -1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 6 \\ 0 \end{bmatrix}}_{b} \tag{3.8}$$

$$\begin{aligned} 3x_1 &= 6 \\ 2x_2 &= -2 \\ x_1 - 2x_2 &= 2 \\ x_1 - x_3 + 2x_4 &= 10 \end{aligned} \iff \underbrace{\begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 1 & -2 & 0 & 0 \\ 1 & 0 & -1 & 2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 6 \\ -2 \\ 2 \\ 10 \end{bmatrix}}_{b} \tag{3.9}$$

The following systems of linear equations are square, but they are not lower triangular. The "offending term" is boxed,

$$\begin{aligned} 3x_1 &= 6 \\ 2x_1 - x_2 - \boxed{x_3} &= -2 \\ x_1 - 2x_2 + 3x_3 &= 2 \end{aligned} \iff \underbrace{\begin{bmatrix} 3 & 0 & 0 \\ 2 & -1 & \boxed{-1} \\ 1 & -2 & 3 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 6 \\ -2 \\ 2 \end{bmatrix}}_{b} \tag{3.10}$$

and

$$\begin{aligned} 3x_1 + \boxed{3x_2} &= 6 \\ 2x_1 - 2x_2 &= 0 \end{aligned} \iff \underbrace{\begin{bmatrix} 3 & \boxed{3} \\ 2 & -1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 6 \\ 0 \end{bmatrix}}_{b} \tag{3.11}$$

Triangular Matrices

The key ingredients of a square lower triangular system are

- The unknowns are ordered, as in $\begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix}$ or $\begin{pmatrix} u & v & w & x & y & z \end{pmatrix}$
- The first equation only involves the first unknown.
- The second equation involves only the first two unknowns
- More generally, the i-th equation involves only the first i unknowns.
- The coefficients a_{ij} on or below the diagonal can be zero or non-zero.

From a matrix point of view, the condition is, all terms above the diagonal are zero. What does this mean? It means that A looks like this,

$$A = \begin{bmatrix} \mathbf{a_{11}} & 0 & 0 & 0 & \cdots & 0 & 0 \\ a_{21} & \mathbf{a_{22}} & 0 & 0 & \cdots & 0 & 0 \\ a_{31} & a_{32} & \mathbf{a_{33}} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ a_{(n-1)1} & a_{(n-1)2} & a_{(n-1)3} & a_{(n-1)4} & \cdots & \mathbf{a_{(n-1)(n-1)}} & 0 \\ a_{n1} & a_{n2} & a_{n3} & a_{n4} & \cdots & a_{n(n-1)} & \mathbf{a_{nn}} \end{bmatrix}, \tag{3.12}$$

namely, $a_{ij} = 0$ for $j > i$. Note that the diagonal is in bold and the terms above the diagonal are in red. The matrices in (3.7) through (3.9) are *lower triangular*, while the matrices in (3.10) and (3.11) are not lower triangular.

3.4 Determinant of a Lower Triangular Matrix

Highly Useful Fact: The matrix determinant of a square lower triangular matrix is equal to the product of the elements on the diagonal. For the matrix A in (3.12), its determinant is

$$\det(A) = a_{11} \cdot a_{22} \cdot \ldots \cdot a_{nn}. \tag{3.13}$$

Hence, for lower triangular matrices of size 10×10 or so, the determinant can be computed by inspection! Let's do a few:

$$\det\left(\begin{bmatrix} 3 & 0 & 0 \\ 2 & -1 & 0 \\ 1 & -2 & 3 \end{bmatrix}\right) = 3 \cdot (-1)3 = -9 \neq 0.$$

Hence, the system of equations (3.7) has a unique solution.

Let's now use the alternative notation for the determinant of a matrix,

$$\begin{vmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 1 & -2 & 0 & 0 \\ 1 & 0 & -1 & 2 \end{vmatrix} = 3 \cdot 2 \cdot 0 \cdot 2 = 0.$$

Hence, the system of equations (3.9) is one of the problem cases: it may have no solution or it may have an infinite number of solutions.

Remark: For a square lower triangular matrix, the matrix determinant is nonzero if, and only if, all of the elements on the diagonal of the matrix are non-zero. Equivalently, for a square lower triangular matrix, the matrix determinant is zero if, and only if, at least one of the elements on the diagonal of the matrix is zero. In other words, we do not really need to multiply them out to check for $\det(A) \neq 0$, the condition for uniqueness of solutions of square systems of linear equations.

3.5 Lower Triangular Systems and Forward Substitution

We develop a solution to a lower triangular system by starting at the top and working our way to the bottom via a method called **forward substitution**. As an example, we use (3.7), which for convenience, we repeat here:

$$\begin{aligned} 3x_1 &= 6 \\ 2x_1 - x_2 &= -2 \\ x_1 - 2x_2 + 3x_3 &= 2 \end{aligned} \iff \underbrace{\begin{bmatrix} 3 & 0 & 0 \\ 2 & -1 & 0 \\ 1 & -2 & 3 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 6 \\ -2 \\ 2 \end{bmatrix}}_{b}. \tag{3.14}$$

Because we have ordered the variables as $\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix}$, we isolate x_1 in the first equation, x_2 in the second equation, and x_3 in the third equation by moving the other variables to the right-hand side,

$$\begin{aligned} 3x_1 &= 6 \\ -x_2 &= -2 - 2x_1 \\ 3x_3 &= 2 - x_1 + 2x_2. \end{aligned} \tag{3.15}$$

You can see how the first equation is trivial, and so is the second one, once the first is solved, etc. Next, we can make the coefficients of the leading variables all equal to one by dividing through by the coefficients that multiply them, yielding

$$\begin{aligned} x_1 &= \frac{1}{3}6 = 2 \\ x_2 &= -\left[-2 - 2x_1\right] = 2 + 2x_1 \\ x_3 &= \frac{1}{3}\left[2 - x_1 + 2x_2\right]. \end{aligned} \tag{3.16}$$

Next, we substitute in, working from top to bottom:

$$\begin{aligned} x_1 &= \frac{1}{3}6 = 2 \\ x_2 &= 2 + 2x_1 = 6 \\ x_3 &= \frac{1}{3}\left[2 - x_1 + 2x_2\right] = \frac{1}{3}\left[12\right] = 4, \end{aligned} \tag{3.17}$$

Forward Substitution

The method is called **forward substitution** because once we have solved x_1, we take its value forward to the next equation where we solve for x_2, and once we have solved for x_1 and x_2, we take their values forward to the next equation where we solve for x_3. I am guessing that you see the pattern.

When can forward substitution go wrong? Well first of all, we only use it for lower triangular systems of linear equations. If the diagonal of the matrix corresponding to the system of linear equations has a zero on it, then the matrix determinant is zero ($\implies$ no solution or an infinite number of solutions) and forward substitution would lead us to **divide by zero, which we know is a major error in mathematics**.

As an example, we take (3.9), which we repeat here

$$\begin{aligned} 3x_1 &= 6 \\ 2x_2 &= -2 \\ x_1 - 2x_2 &= 2 \\ x_1 - x_3 + 2x_4 &= 10. \end{aligned} \iff \underbrace{\begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 1 & -2 & \boxed{0} & 0 \\ 1 & 0 & -1 & 2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 6 \\ -2 \\ 2 \\ 10 \end{bmatrix}}_{b} \tag{3.18}$$

When we try to isolate x_3 in the third equation, we have a problem. To make it more obvious, we make x_3 explicit with its corresponding coefficient of zero

$$\left.\begin{aligned} 3x_1 &= 6 \\ 2x_2 &= -2 \\ x_1 - 2x_2 + \boxed{0x_3} &= 2 \\ x_1 - x_3 + 2x_4 &= 10 \end{aligned}\right\} \implies \left.\begin{aligned} 3x_1 &= 6 \\ 2x_2 &= -2 \\ \boxed{0x_3} &= 2 - (x_1 - 2x_2) \\ 2x_4 &= 10 - (x_1 - x_3) \end{aligned}\right\} \overset{???}{\implies} \begin{aligned} x_1 &= 2 \\ x_2 &= -1 \\ x_3 &= \boxed{\frac{1}{0}} \cdot (-2) \quad \text{?!? Divide by zero: not allowed!} \\ x_4 &= \frac{1}{4}(10 - 2 + \frac{-2}{0}) \quad \text{?!?!? } \textbf{Wrong!} \end{aligned} \tag{3.19}$$

3.6 Upper Triangular Systems, Upper Triangular Matrices, Determinants, and Back Substitution

This is an example of a square **upper triangular system of equations** and its corresponding **upper triangular matrix**

$$\begin{aligned} x_1 + 3x_2 + 2x_3 &= 6 \\ 2x_2 + x_3 &= -2 \\ 3x_3 &= 4, \end{aligned} \iff \underbrace{\begin{bmatrix} 1 & 3 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 6 \\ -2 \\ 4 \end{bmatrix}}_{b}. \tag{3.20}$$

We note that all of the coefficients of A *below the diagonal* are zero; that is $a_{ij} = 0$ for $i > j$.

Highly Useful Fact: The matrix determinant of a square upper triangular matrix is equal to the product of the elements on the diagonal. For the matrix A in (3.20), its determinant is

$$\det(A) = 1 \cdot 2 \cdot 3 = 6. \tag{3.21}$$

Hence, we know the system of equations has a unique solution.

Back Substitution

We develop a solution to an **upper triangular system of linear equations** by starting at the bottom and work our way to the top via **back substitution**. We first solve for the leading variables, which here we do in one step,

$$\begin{aligned} x_1 &= 6 - (3x_2 + 2x_3) \\ x_2 &= \frac{1}{2}(-2 - x_3) \\ x_3 &= \frac{4}{3}, \end{aligned} \tag{3.22}$$

but of course, you can do it in two steps as we did for lower triangular systems. Next, we do back substitution, from bottom to top, sequentially plugging in numbers from the previous equations

$$\begin{array}{llll} x_1 & = 6 - (3x_2 + 2x_3) & = 6 - (3 \cdot (-\frac{5}{3}) + 2 \cdot \frac{4}{3}) & = \frac{18}{3} + \frac{7}{3} = \frac{25}{3} = 8\frac{1}{3} \\ x_2 & = \frac{1}{2} \cdot (-2 - x_3) & = \frac{1}{2} \cdot \left(-2 - \frac{4}{3}\right) & = \frac{1}{2} \cdot \left(-\frac{10}{3}\right) = -\frac{5}{3} = -1\frac{2}{3} \\ x_3 & = \frac{4}{3}. & & \end{array} \tag{3.23}$$

3.7 General Cases

The general form of a lower triangular system with a non-zero determinant is

$$\begin{aligned} a_{11}x_1 &= b_1 \quad (a_{11} \neq 0) \\ a_{21}x_1 + a_{22}x_2 &= b_2 \quad (a_{22} \neq 0) \\ &\vdots = \vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n &= b_n \quad (a_{nn} \neq 0) \end{aligned} \tag{3.24}$$

and the solution proceeds from top to bottom, like this

$$\begin{aligned} x_1 &= \frac{b_1}{a_{11}} \quad (a_{11} \neq 0) \\ x_2 &= \frac{b_2 - a_{21}x_1}{a_{22}} \quad (a_{22} \neq 0) \\ &\vdots = \vdots \\ x_n &= \frac{b_n - a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{n,n-1}x_{n-1}}{a_{nn}} \quad (a_{nn} \neq 0). \end{aligned} \tag{3.25}$$

The general form of an upper triangular system with a non-zero determinant is

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \quad (a_{11} \neq 0) \\ a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \quad (a_{22} \neq 0) \\ a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \quad (a_{33} \neq 0) \\ &\vdots = \vdots \\ a_{nn}x_n &= b_n \quad (a_{nn} \neq 0), \end{aligned} \tag{3.26}$$

and the solution proceeds from the **bottom** of (3.26) to its **top**, like this,

$$\begin{aligned} x_n &= \frac{b_n}{a_{nn}} && (a_{nn} \neq 0) \\ x_{n-1} &= \frac{b_{n-1} - a_{n-1,n}x_n}{a_{n-1,n-1}} && (a_{n-1,n-1} \neq 0) \\ \vdots &= \vdots && \vdots \\ x_2 &= \frac{b_2 - a_{23}x_3 - \cdots - a_{2n}x_n}{a_{22}} && (a_{22} \neq 0) \\ x_1 &= \frac{b_1 - a_{12}x_2 - \cdots - a_{1n}x_n}{a_{11}} && (a_{11} \neq 0) \end{aligned} \tag{3.27}$$

Remark: For $1 \leq i < n$, we need to form the product

$$a_{i,i+1}x_{i+1} + a_{i,i+2}x_{i+2} + \cdots + a_{i,n}x_n.$$

In Julia, there are two ways to do it

```
method1 = A[i, i+1:n]' *x[i+1:n]
method2 = (A[i:i, i+1:n] *x[i+1:n])[1]
```

In method one, $A[i, i+1:n]$ is a column vector while $A[i, i+1:n]'$ is a row vector. When it is multiplied by the column vector, $x[i+1:n]$, it produces a scalar. In method two, $A[i:i, i+1:n]$ is a $1 \times (n-i)$ matrix, which in Julia, is different than a row vector. When it is multiplied by the column $x[i+1:n]$, it produces a 1×1 matrix. You then have to extract its value, which is done via $(A[i:i, i+1:n] * x[i+1:n])[1]$. See the "tinyMatrix" example in Lab #1.

For those of you with a "visual memory", here is a graphical representation for upper and lower triangular matrices

$$\text{(lower triangular)} \begin{bmatrix} \times & & & & \\ \times & \times & & \mathbf{0} & \\ \times & \times & \times & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \end{bmatrix} \qquad \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & \mathbf{0} & & \times & \times \\ & & & & \times \end{bmatrix} \text{(upper triangular).} \tag{3.28}$$

Lower: everything above the diagonal is zero. **Upper:** everything below the diagonal is zero. For us to be able to solve the equations for arbitrary values b on the right-hand side of $Ax = b$, we need the elements on the diagonal to be non-zero.

Example 3.1 *Use back substitution to solve the upper triangular system of linear equations $Ux = b$, where*

$$U = \begin{bmatrix} 0.9555 & -0.8218 & -1.2433 & -0.5536 & 0.9102 & 1.2047 \\ 0.0000 & -0.2728 & 0.3770 & 2.0805 & -1.1050 & 1.0576 \\ 0.0000 & 0.0000 & 0.2126 & 1.0730 & -1.3323 & 2.3487 \\ 0.0000 & 0.0000 & 0.0000 & -0.2295 & 0.9807 & 0.3360 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & -1.2425 & -1.5521 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & -0.7935 \end{bmatrix} \text{ and } b = \begin{bmatrix} 0.8399 \\ -0.8898 \\ 0.0069 \\ -1.1286 \\ -0.0115 \\ -1.1136 \end{bmatrix}.$$

The diagonal of U is in bold font. We note that U is indeed upper triangular because all of its elements below the diagonal are zero. Moreover, $\det(U) \neq 0$ because all of the elements on the diagonal are non-zero.

Solution: We first program a function in Julia that implements back substitution. In HW you will develop Julia code to solve triangular systems of equations. Your data will be given to you sometimes as systems of equations written out as formulas and sometimes directly as matrices. The function below checks that there are no "tiny" entries on the diagonal of U, but it does not check that U is really upper triangular. In HW, we'll have you do that check.

```
function backwardsub(U, b)
    # U a square upper triangular matrix
```

```
    # b has same number of rows as U
    #
    # Assert no entries on the diagonal of U
    # are 0 (or very close to 0)
    if minimum(abs.(diag(U))) < 1e-6
        return false
    end
    n = length(b)
    x = Vector{Float64}(undef, n)
    # Start from the bottom and work our way to the top
    x[n] = b[n] / U[n,n]
    for i = n-1:-1:1
        #x[i]=(b[i] - U[i,(i+1):n]' * x[(i+1):n]) / U[i,i]
        x[i]=( b[i] - (U[i:i,(i+1):n] * x[(i+1):n])[1] ) / U[i,i]
    end
    # The for loop works with either line 15 or line 16. They differ in how
    # a row is extracted from a matrix in Julia
    return x
end

U=[ 0.955467  -0.821842  -1.24331   -0.553594   0.910181   1.20471
  0.0       -0.272776   0.376981   2.08047   -1.10505    1.05765
  0.0        0.0        0.212559   1.07301   -1.33234    2.3487
 -0.0        0.0        0.0       -0.229487   0.980719   0.336002
 -0.0        0.0       -0.0        0.0       -1.24249   -1.55205
 -0.0        0.0        0.0        0.0       -0.0       -0.793501]

b=[0.8398952455773964
 -0.8897505302659705
  0.006884706336738545
 -1.1285718398040936
 -0.011546427596053652
 -1.1135689635657877]

x=backwardsub(U, b)
```

```
6-element Vector{Float64}:
 -48.810775767214956
 -21.035398066599555
 -23.98466204387721
  -0.47925601967730014
  -1.7437077113383561
   1.40336197583662
```

```
U*x-b
```

```
6-element Vector{Float64}:
 -7.771561172376096e-16
 -8.881784197001252e-16
  2.211772431870429e-16
  0.0
  3.122502256758253e-17
  0.0
```

It looks like the computed x is a pretty good solution of $Ux = b$! ■

3.8 A Simple Trick with Systems of Equations: Re-arranging their Order

This system of equations is neither upper triangular nor lower triangular

$$\begin{aligned} 3x_1 &= 6 \\ x_1 - 2x_2 + 3x_3 &= 2 \\ 2x_1 - x_2 &= -2. \end{aligned} \tag{3.29}$$

We can simply re-arrange the order of the equations to arrive at

$$\begin{aligned} 3x_1 &= 6 \\ 2x_1 - x_2 &= -2 \\ x_1 - 2x_2 + 3x_3 &= 2, \end{aligned} \tag{3.30}$$

which happens to be lower triangular. We still have the same equations and hence their solution has not changed. The equations have just been re-arranged to make the process of computing their solution fall into a nice pattern that we already know how to handle.

This is a really useful trick in mathematics: re-arranging a set of equations without changing the answer. Right here, it may seem kind of trivial, but when we look at this in terms of matrices in the next Chapter, we get a cool piece of insight.

3.9 Looking Ahead

Once again, our goal is to solve very large sets of linear equations, say more than 100 variables. In this chapter, we saw how to solve problems with triangular structure. Our next major way point is to reduce all linear systems of n-equations in n-variables to being solvable with forward substitution and back substitution. To get there we need to understand:

- the standard way to multiply two matrices
- a little known way to multiply two matrices
- how to *factor* a square matrix as the product of two triangular matrices

Help! Help! How am I supposed to remember all of this?

You probably can't. In any case, we don't want you to memorize the ROB 101 material. Instead, open up a `google doc or google sheet` and make notes! You need an organized method for keeping track of stuff. In High School, you may have been able to remember all the new notation without any special effort. In College, it's a bit different.

Appendix A

To Learn on Your Own (if you want to): Cool and Important Things We Omitted From our Linear Algebra Introduction

Learning Objectives

- Introduce material that is commonly included in a second or third year Linear Algebra Course
- Provide a resource for use after you leave ROB 101.

Outcomes

- Complex numbers obey the same rules of arithmetic as the real numbers, if you really understand the real numbers!
- Eigenvalues and eigenvectors of square matrices
- Symmetric matrices have real eigenvalues and admit orthonormal eigenvectors
- Positive definite matrices allow one to generalize the Euclidean norm
- The Singular Value Decomposition (SVD) allows one to quantify the degree to which vectors are linearly independent. This is super useful in engineering practice.
- Matrices are good for other things than representing systems of equations: they also allow one to transform vectors in interesting ways, giving rise to the concept of *linear transformations*.
- Many more facts about basis vectors.

A.1 Complex Numbers and Complex Vectors

Here are some video resources that you may enjoy consulting:

- https://youtu.be/T647CGsuOVU
- https://youtu.be/2HrSG0fdxLY
- https://youtu.be/N9QOLrfcKNc
- https://youtu.be/DThAoT3q2V4
- https://youtu.be/65wYmy8Pf-Y

The story of complex numbers begins with the quadratic equation $x^2+1=0$, which has no real solutions! After much soul searching, the mathematics community finally embraced the notion of an **imaginary quantity** $\mathbb{i}$ defined by

$$(\mathbb{i})^2 := -1. \tag{A.1}$$

More commonly, we write this as

$$\mathbb{i} = \sqrt{-1}. \tag{A.2}$$

The set of **complex numbers** is then defined as

$$\mathbb{C} := \{x + \mathbb{i}\, y \,|x \in \mathbb{R}, y \in \mathbb{R}\}\,. \tag{A.3}$$

If $z = x + \mathbb{i}\, y \in \mathbb{C}$, then we define

$$\begin{aligned} x &:= \text{real}(z) \;\; \text{the } \textbf{real part} \text{ of } \; z \\ y &:= \text{imag}(z) \;\; \text{the } \textbf{imaginary part} \text{ of } \; z. \end{aligned} \tag{A.4}$$

We note that both x and y are real numbers. Complex numbers of the form $0 + \mathbb{i}\, y$ are called **imaginary numbers**. We view a **real number** $x \in \mathbb{R}$ as being a complex number of the form $x + \mathbb{i}\, 0$. In other words, we view $\mathbb{R} \subset \mathbb{C}$. In addition, we define the **complex conjugate** of $z = x + \mathbb{i}\, y$ to be

$$z^* := x - \mathbb{i}\, y, \tag{A.5}$$

that is, $\text{imag}(z^*) = -\text{imag}(z)$, while $\text{real}(z^*) = \text{real}(z)$.

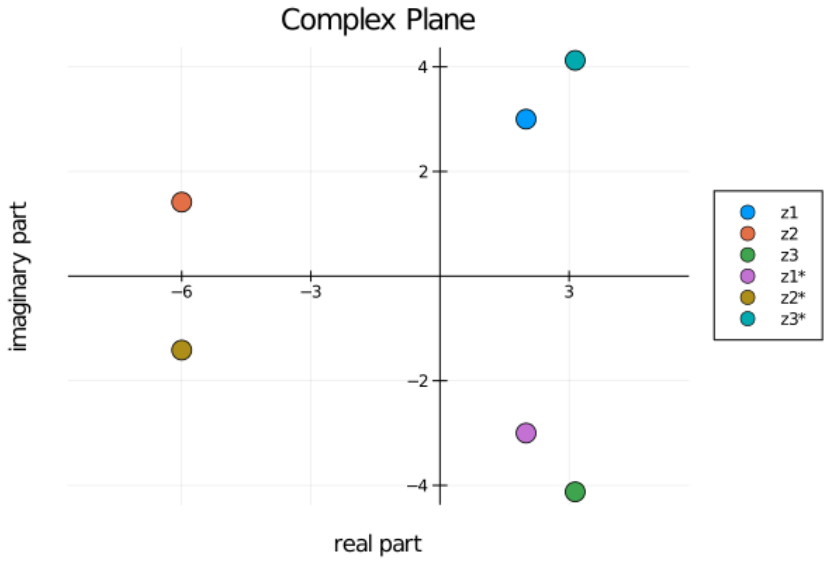


Figure A.1: The complex plane has x-axis given by the real part of a complex number and y-axis given by the imaginary part of a complex number. Here, we plot z_1, z_2, z_3 from Example A.1 and their complex conjugates.

Example A.1 *For the following are complex numbers, compute their real and imaginary parts as well as their complex conjugates. Also, plot them in the complex plane.*

$$z_1 = 2 + \mathbb{i}\, 3$$
$$z_2 = -6 + \mathbb{i}\, \sqrt{2}$$
$$z_3 = \pi - \mathbb{i}\, \sqrt{17}.$$

Solution

$$\begin{array}{lll} \text{real}(z_1) = 2 & \text{imag}(z_1) = 3 & z_1^* = 2 - \mathbb{i}\, 3 \\ \text{real}(z_2) = -6 & \text{imag}(z_2) = \sqrt{2} & z_2^* = -6 - \mathbb{i}\, \sqrt{2} \\ \text{real}(z_3) = 2\pi & \text{imag}(z_3) = -\sqrt{17} & z_2^* = \pi + \mathbb{i}\, \sqrt{17}. \end{array}$$

All of these values are plotted in Fig. A.1.

■

A.1.1 Arithmetic of Complex Numbers: Enough to Get You By

We'll define all of the major arithmetic operations. Just like operations with vectors and matrices, however, it's much more fun to do the calculations in Julia than by hand!

The **addition of two complex numbers** is defined by adding their respective real and imaginary parts,

$$(x_1 + \mathbb{i}\, y_1) + (x_2 + \mathbb{i}\, y_2) := (x_1 + x_2) + \mathbb{i}\, (y_1 + y_2)\,. \tag{A.6}$$

This is very similar to how we add two vectors

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} := \begin{bmatrix} x_1 + x_2 \\ y_1 + y_2 \end{bmatrix}$$

by adding their respective components.

The **multiplication** of two complex numbers is defined by

$$(x_1 + \mathbb{i}\, y_1) \cdot (x_2 + \mathbb{i}\, y_2) := (x_1 x_2 - y_1 y_2) + \mathbb{i}\, (x_1 y_2 + y_1 x_2)\,. \tag{A.7}$$

This formula comes from applying basic algebra to the "symbolic expression"

$$(a_1 + b_1)(a_2 + b_2) = a_1 a_2 + b_1 b_2 + a_1 b_2 + b_1 a_2$$

and then substituting in

$$a_1 := x_1$$
$$a_2 := x_2$$
$$b_1 := \mathbb{i}\, y_1$$
$$b_2 := \mathbb{i}\, y_2.$$

The term $b_1 b_2 = (\mathbb{i}\, y_1) \cdot (\mathbb{i}\, y_2) = (\mathbb{i})^2 y_1 y_2 = (-1) y_1 y_2$, which explains how the minus sign appears!

Let's note that if we multiply a complex number by its complex conjugate, then we obtain a real number. Indeed,

$$z \cdot z^* = (x + \mathbb{i}\, y) \cdot (x - \mathbb{i}\, y) = ((x)(x) - (y)(-y)) + \mathbb{i}\, ((x)(-y) + (y)(x)) = x^2 + y^2. \tag{A.8}$$

The **magnitude of a complex number** $z = x + \mathbb{i}\, y$ is denoted by $|z|$ and is defined by

$$|z| := \sqrt{x^2 + y^2}, \tag{A.9}$$

or equivalently, by

$$|z| := \sqrt{z \cdot z^*}. \tag{A.10}$$

Both definitions are common and we note that the square root makes sense[1] because the magnitude is a non-negative real number.

Using the complex conjugate, the **division of one complex number by another** can be defined, and subsequently, understood. We define

$$\frac{x_1 + \mathbb{i}\, y_1}{x_2 + \mathbb{i}\, y_2} := \frac{(x_1 x_2 + y_1 y_2) + \mathbb{i}\, (y_1 x_2 - x_1 y_2)}{(x_2)^2 + (y_2)^2}, \tag{A.11}$$

and note that the denominator is real, and thus the indicated division can be treated as multiplication by one over the denominator. It follows that when $|z_2| \neq 0$, z_1/z_2 is a well-defined complex number. The formula (A.11) is best understood from an alternative definition of complex division

$$\frac{z_1}{z_2} := \frac{z_1 \cdot z_2^*}{z_2 \cdot z_2^*} = \frac{z_1 \cdot z_2^*}{|z_2|^2}. \tag{A.12}$$

Personally, we try to avoid using either one of these formulas and do the computations in Julia! Multiplication and division of complex numbers by hand is very error prone. For probably a century, engineering faculty have been torturing students by making them do such calculations by hand; at some point, it has to stop!

Example A.2 *For the following complex numbers, compute their sum, product, division, and magnitudes,*

$$z_1 = 2 + \mathbb{i}\, 3$$
$$z_2 = -6 + \mathbb{i}\, \sqrt{2}$$

Solution

■

A.1.2 Angles of Complex Numbers and Euler's Formula: More Advanced Aspects

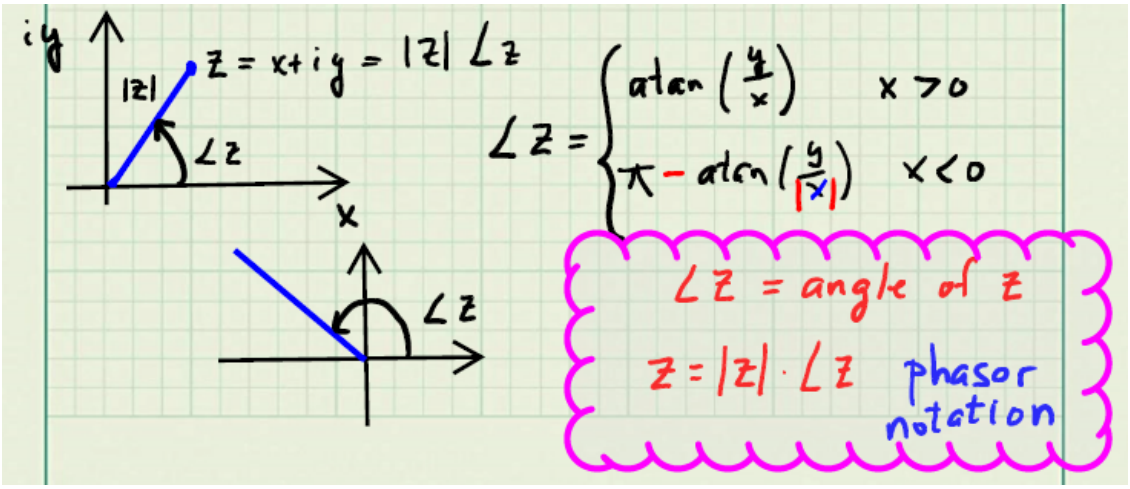


Figure A.2: It is often helpful to understand complex numbers as having a magnitude and an angle. This is similar to using polar coordinates in $\mathbb{R}^2$. Here, the angle of z is denoted by $\angle z$ instead of θ. Both conventions are common.

For a pair of real numbers (x, y), we were taught in High School how to express them in **polar coordinates** (ρ, θ), where

$$\rho := \sqrt{x^2 + y^2}$$
$$\theta := \begin{cases} \arctan(y/x) & x > 0 \\ \pi - \arctan(y/|x|) & x < 0 \\ \text{sign}(y)\ \frac{\pi}{2} & x = 0, y \neq 0 \\ \text{undefined} & x = 0, y = 0. \end{cases} \tag{A.13}$$

[1] Julia will recognize $\text{real}(z)^2 + \text{imag}(z)^2$ as being a real number. It does not recognize $z \cdot z^*$ as a real number. In Julia, the command is $\text{abs}(z)$, just as with a real number.

From polar coordinates (ρ, θ), we computed the **Cartesian Coordinates** (x, y) as

$$\begin{aligned} x &= \rho\cos(\theta) \\ y &= \rho\sin(\theta). \end{aligned} \tag{A.14}$$

Moving beyond High School, we can also express the above in terms of the canonical basis vectors $\{e_1, e_2\}$ for $\mathbb{R}^2$ as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \rho\cos(\theta)e_1 + \rho\sin(\theta)e_2. \tag{A.15}$$

In fact, any (non-zero) vector $v \in \mathbb{R}^2$ can be expressed as

$$v = ||v||\cos(\theta)e_1 + ||v||\sin(\theta)e_2. \tag{A.16}$$

Equation (A.16) hints at a natural way of expressing complex numbers.

Polar Coordinates Meet Complex Numbers and their Multiplication

For a non-zero complex number $z = x + \mathbb{i}\, y$, we define its angle as in (A.13). Doing so allows us to express every (non-zero) $z \in \mathbb{C}$ as

$$z = |z|\cos(\theta) + \mathbb{i}\,|z|\sin(\theta). \tag{A.17}$$

The real and imaginary parts of z are playing the role of the basis vectors $\{e_1, e_2\}$. One can also think of $\{1.0, \mathbb{i}\}$ as being a basis for C, though this is beyond our scope. Equation (A.17) leads to a very nice way to understand the multiplication of two complex numbers.

Fact: Suppose $z_1 = |z_1|\cos(\theta_1) + \mathbb{i}\,|z_1|\sin(\theta_1)$ and $z_2 = |z_2|\cos(\theta_2) + \mathbb{i}\,|z_2|\sin(\theta_2)$ are non-zero complex numbers. Then,

$$\boxed{\begin{aligned} z_1 \cdot z_2 &= |z_1||z_2|\cos(\theta_1 + \theta_2) + \mathbb{i}\,|z_1||z_2|\sin(\theta_1 + \theta_2) \\ \frac{z_1}{z_2} &= \frac{|z_1|}{|z_2|}\cos(\theta_1 - \theta_2) + \mathbb{i}\,\frac{|z_1|}{|z_2|}\sin(\theta_1 - \theta_2). \end{aligned}} \tag{A.18}$$

When expressed in "polar form", multiplying two complex numbers is equivalent to multiplying their magnitudes and adding their angles (or phases), while dividing two complex numbers is equivalent to dividing their magnitudes and subtracting their angles (or phases). Proving (A.18) involves some trigonometric identities. You may want to give it a go. We'll provide a simpler way to understand it in a few more lines!

Remark: A positive real number has angle zero, while a negative real number has angle π (or, equivalently, $-\pi$). The angle of $\mathbb{i}$ is $\pi/2$ and the angle of $-\mathbb{i}$ is $-\pi/2$ (or, equivalently, $3\pi/2$).

The exponential function of a real number x is defined by the infinite series

$$\begin{aligned} e^x &:= \sum_{n=0}^{\infty} \frac{x^n}{k!} \\ &= 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \cdots \\ &= 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \cdots, \end{aligned} \tag{A.19}$$

where $n! := 1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n$, the product of all integers from 1 to n.

Euler's Formula

A very famous result due to the Swiss Mathematician Leonhard Euler (https://en.wikipedia.org/wiki/Leonhard_Euler), asserts that for a given real number θ,

$$e^{\mathbb{i}\,\theta} = \cos(\theta) + \mathbb{i}\,\sin(\theta) \quad \textbf{Euler's Formula}. \tag{A.20}$$

Hence, every complex number can be written as

$$z = |z| e^{\mathbb{i}\,\theta}, \tag{A.21}$$

which leads to

Fact: Suppose $z_1 = |z_1| e^{\mathbb{i}\,\theta_1}$ and $z_2 = |z_2| e^{\mathbb{i}\,\theta_2}$ are non-zero complex numbers. Then,

$$\boxed{\begin{aligned} z_1 \cdot z_2 &= |z_1||z_2| e^{\mathbb{i}\,(\theta_1+\theta_2)} \\ \\ \frac{z_1}{z_2} &= \frac{|z_1|}{|z_2|} e^{\mathbb{i}\,(\theta_1-\theta_2)}. \end{aligned}} \tag{A.22}$$

Deriving this result for multiplication and division is much easier than (A.18), but Euler's Formula (A.20) assures us they are the same.

Remark: Deriving Euler's formula is not that hard. When you substitute $\mathbb{i}\,\theta$ into (A.19), you must first note that $(\mathbb{i})^{2n} = (-1)^n$ and $(\mathbb{i})^{2n+1} = (-1)^n\,\mathbb{i}$. If you then separate the power series into its real and imaginary parts, you will recognize the power series for $\cos(\theta)$ and $\sin(\theta)$.

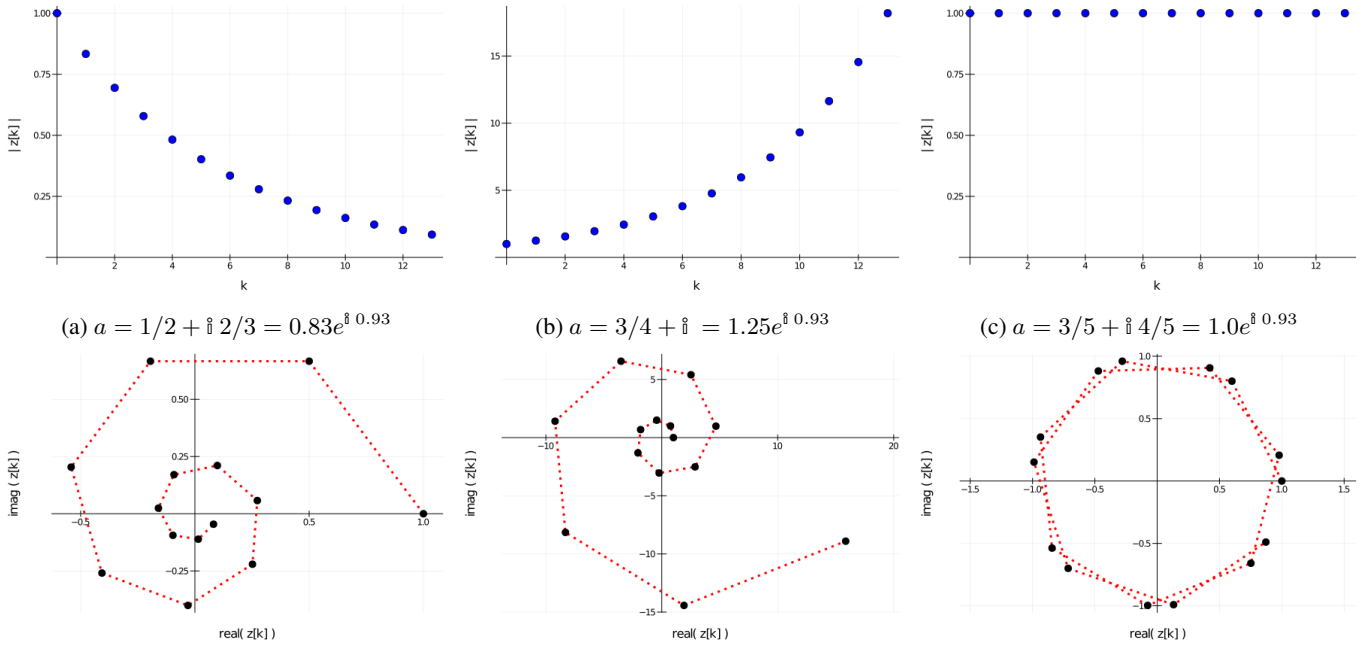
A.1.3 Iterating with Complex Numbers: Background for Eigenvalues

Consider the equation

$$z_{k+1} = a z_k, \tag{A.23}$$

with $a \in \mathbb{C}$ and $z_0 \in \mathbb{C}$. Equation (A.23) is technically called a **scalar linear difference equation**, but for us, it looks not so different than iterating with the bisection method or Newton's Algorithm. We compute a few steps until the general pattern of its solution becomes clear:

$$\begin{aligned} z_1 &= a z_0 \\ z_2 &= a z_1 = a^2 z_0 \\ z_3 &= a z_2 = a^3 z_0 \\ &\vdots \\ z_k &= a^k z_0. \end{aligned} \tag{A.24}$$



(a) $a = 1/2 + \mathbb{i}\, 2/3 = 0.83e^{\mathbb{i}\, 0.93}$ (b) $a = 3/4 + \mathbb{i}\, = 1.25e^{\mathbb{i}\, 0.93}$ (c) $a = 3/5 + \mathbb{i}\, 4/5 = 1.0e^{\mathbb{i}\, 0.93}$

(d) $a = 3/5 + \mathbb{i}\, 4/5 = 1.0\angle 53.1$ degrees (e) $a = 3/4 + \mathbb{i}\, = 1.25\angle 53.1$ degrees (f) $a = 3/5 + \mathbb{i}\, 4/5 = 1.0\angle 53.1$ degrees

Figure A.3: The solid dots illustrate the evolution of z_k in (A.23) and (A.24) when a has magnitude less than one, greater than one, and equal to one, respectively. In each case, $z_0 = 1.0 + \mathbb{i}\, 0.0$ and the angle of a was selected to be 53.1 degrees; therefore the dots rotate counterclockwise. The red dashes are present to guide the eye in connecting the dots. In (f), the points lie on a circle of radius one.

Scalar Linear Difference Equation

The general solution to $z_{k+1} = az_k$, $z_0 \in \mathbb{C}$ is $z_k = a^k z_0$. We write $a = |a|e^{\mathbb{i}\, \theta}$, where $\theta = \angle a$, the angle of a as computed in (A.13). Then from (A.22), we conclude that

$$z_k = |a|^k e^{\mathbb{i}\, k\angle a} z_0. \tag{A.25}$$

Below, we analyze three cases and show the following for $z_0 \neq 0$

- $|a| < 1 \implies |z_k| \underset{k\to\infty}{\longrightarrow} 0$
- $|a| > 1 \implies |z_k| \underset{k\to\infty}{\longrightarrow} \infty$
- $|a| = 1 \implies |z_k| = |z_0|, k \geq 0.$

See also Fig. A.3.

The following analysis supports the illustrations in Fig. A.3.

Case 1: |a|<1 We note that $\log(|a|^k) = k\log(|a|)$, and that $|a| < 1 \implies \log(|a|) < 0$. Hence,

$$\lim_{k\to\infty} |a|^k = \lim_{k\to\infty} e^{k\log(|a|)} = 0.$$

Case 2: |a|>1 We note that $\log(|a|^k) = k\log(|a|)$, and that $|a| > 1 \implies \log(|a|) > 0$. Hence,

$$\lim_{k\to\infty} |a|^k = \lim_{k\to\infty} e^{k\log(|a|)} = \infty.$$

Case 3: |a|=1 We note that when $|a| = 1$, then $|a|^k = 1$ for all $k \geq 1$, and thus this case is clear.

A.1.4 $\mathbb{C}^n$, the Space of Complex Vectors

$\mathbb{C}^n$ sounds harder than it is. It's exactly $\mathbb{R}^n$ where the scalars are complex numbers instead of real numbers. All the definitions of vector addition, linear combinations, spans, and linear independence are the same. We'll cover just a few of the basic ideas so that you get the idea.

Recall that we started by defining $\mathbb{R}^n$ as n-tuples of real numbers and then we identified it with column vectors of length n. We do that same here.

$$\boxed{\mathbb{C}^n := \{(\alpha_1, \alpha_2, \ldots, \alpha_n) \mid \alpha_i \in \mathbb{C}, 1 \leq i \leq n\} \iff \left\{ \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} \;\middle|\; \alpha_i \in \mathbb{C}, 1 \leq i \leq n \right\} =: \mathbb{C}^n} \tag{A.26}$$

Consider two vectors $v_1 \in \mathbb{C}^n$ and $v_2 \in \mathbb{C}^n$. We define their **vector sum** by

$$v_1 + v_2 = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} := \begin{bmatrix} \alpha_1 + \beta_1 \\ \alpha_2 + \beta_2 \\ \vdots \\ \alpha_n + \beta_n \end{bmatrix},$$

that is, we sum their respective components or entries. Let γ be a complex number. Then we define

$$\gamma v := \begin{bmatrix} \gamma\alpha_1 \\ \gamma\alpha_2 \\ \vdots \\ \gamma\alpha_n \end{bmatrix},$$

that is, to **multiply a complex vector by a complex number**, we multiply each of the components of the vector by the number, just as we do for real vectors.

Let $\{v_1, v_2, \ldots, v_k\}$ be a collection of vectors in $\mathbb{C}^n$. Then we define their **span** as

$$\text{span}\{v_1, v_2, \ldots, v_k\} := \{\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k \mid \alpha_i \in \mathbb{C}, 1 \leq i \leq k\}.$$

The set of vectors $\{v_1, v_2, \ldots, v_k\}$ is **linearly independent** in the vector space $\mathbb{C}^n$ if the only solution to

$$\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k = 0$$

is $\alpha_1 = 0 + \mathbb{i}\, 0, \alpha_2 = 0 + \mathbb{i}\, 0, \ldots, \alpha_k = 0 + \mathbb{i}\, 0$.

The **norm** of a complex vector

$$v = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix}$$

is

$$||v|| := \sqrt{\sum_{k=1}^{n} |\alpha_k|^2},$$

where, to be extra clear, $|\alpha_k|^2 = \alpha_k \cdot \alpha_k^*$. Moreover, if one defines the **complex conjugate of a vector** by taking the complex conjugate of each of its components, then

$$||v||^2 = (v^*)^\top \cdot v,$$

and yes, the transpose simply takes the column vector to a row vector.

A.1.5 Iterating with Matrices: The Case for Eigenvalues and Eigenvectors

We now attempt to analyze the matrix versions of (A.23) and (A.24). Recall that you saw matrix difference equations in Project 3. Our real goal is to understand

$$x_{k+1} = Ax_k, \tag{A.27}$$

with A an $n \times n$ real matrix and $x_0 \in \mathbb{R}^n$. But we'll see that allowing the entries of A to be complex and $x_0 \in \mathbb{C}^n$ does not change anything.

With this in mind, we rewrite (A.27) first as

$$x[k+1] = Ax[k],$$

with the time index denoted in square brackets, Julia style! This will allow us to use a subscript for the components of x. Next, we replace $x[k]$ with $z[k]$ to emphasize that the we allow $z[k]$ to be a complex vector. We compute a few steps of $z[k+1] = Az[k]$ until the general pattern of its solution becomes clear:

$$\begin{aligned} z[1] &= Az[0] \\ z[2] &= Az[1] = A^2 z[0] \\ z[3] &= Az[2] = A^3 z[0] \\ &\vdots \\ z[k] &= A^k z[0]. \end{aligned} \tag{A.28}$$

So far so good! Now, our challenges are:

- give conditions on A so that $|| \; z[k] \; ||$ contracts, blows up, or stays bounded as k tends to infinity;
- even better, for a given initial condition $z[0]$, describe in detail the evolution of $z[k]$ for $k > 0$.

We'll start with a diagonal $n \times n$ matrix A, and for reasons that will become clear in the next section, we'll denote the entries on the diagonal by λ,

$$A = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \lambda_n \end{bmatrix}. \tag{A.29}$$

We leave it as an exercise to compute that

$$A^2 = \begin{bmatrix} (\lambda_1)^2 & 0 & 0 & 0 \\ 0 & (\lambda_2)^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & (\lambda_n)^2 \end{bmatrix}, \tag{A.30}$$

and once you have established (A.30), you will have no trouble believing that

$$A^k = \begin{bmatrix} (\lambda_1)^k & 0 & 0 & 0 \\ 0 & (\lambda_2)^k & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & (\lambda_n)^k \end{bmatrix}. \tag{A.31}$$

One thing we can do is observe that

$$\begin{bmatrix} z_1[k] \\ z_2[k] \\ \vdots \\ z_n[k] \end{bmatrix} = \begin{bmatrix} (\lambda_1)^k & 0 & 0 & 0 \\ 0 & (\lambda_2)^k & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & (\lambda_n)^k \end{bmatrix} \begin{bmatrix} z_1[0] \\ z_2[0] \\ \vdots \\ z_n[0] \end{bmatrix} \tag{A.32}$$

results in n scalar equations of the form (A.23), namely,

$$\boxed{z_j[k] = (\lambda_j)^k z_j[0],\ \ 1 \leq j \leq n.} \tag{A.33}$$

Linear Difference Equation with a Diagonal Matrix

The general solution to $z[k+1] = Az[k]$, $z[0] \in \mathbb{C}^n$ is $z[k] = A^k z[0]$. When A is diagonal, the solution is given in (A.32) and (A.33). Based on these results and Chapter A.1.3, we analyze three cases for $z_j[0] \neq 0$,

- $|\lambda_j| < 1 \implies |z_j[k]| \underset{k\to\infty}{\longrightarrow} 0$
- $|\lambda_j| > 1 \implies |z_j[k]| \underset{k\to\infty}{\longrightarrow} \infty$
- $|\lambda_j| = 1 \implies |z_j[k]| = |z_j[0]|, k \geq 0.$

Being a ROB 101 student, having "real" matrices replaced by diagonal matrices must be a bit disconcerting! You'll be glad to know that it is really just a step to something kind of magical: most matrices can be factored as $A = M\Lambda M^{-1}$, where $\det(M) \neq 0$ and Λ is diagonal, as in (A.29). But we get ahead of ourselves!

Key features of a Diagonal Matrix: One way that Eigenvalues and Eigenvectors come about

Let $v_j = e_j$, where e_j are the canonical basis vectors for either $\mathbb{R}^n$ or $\mathbb{C}^n$ (aka, columns of the $n \times n$ identity matrix). We have noted before that $Ae_j = a_j^{\text{col}}$. In our case, $a_j^{\text{col}} = \lambda_j e_j$. Hence, substituting in $v_j = e_j$, we arrive at the equation

$$\boxed{Av_j = \lambda_j v_j,\ \ 1 \leq j \leq n.} \tag{A.34}$$

Equation (A.34) is the defining relation for eigenvalues (denoted here by λ_j) and eigenvectors (denoted here by v_j). We further note that the set of vectors $\{v_1, v_2, \ldots, v_n\}$ is linearly independent and spans both $\mathbb{R}^n$ and $\mathbb{C}^n$. Having a set of eigenvectors that forms a basis turns out to be a defining characteristic of matrices that are related to a diagonal matrix Λ by a transformation of the form $A = M\Lambda M^{-1}$.

Remark: If $v \in \mathbb{C}^n$ is an eigenvector, meaning $v \neq 0$ and there exists a $\lambda \in \mathbb{C}$ such that (A.34) holds, then we have that

$$\begin{aligned} Av &= \lambda v \\ A^2 v &= A(\lambda v) = \lambda Av = (\lambda)^2 v \\ &\vdots \\ A^k v &= (\lambda)^k v \end{aligned}$$

and hence we can analyze convergence for the difference equation $z[k+1] = Az[k]$, $z[0] = v$, even when A is not diagonal.

Remark: Suppose that A is real and that $\lambda \in \mathbb{C}$ and $v \in \mathbb{C}^n$, satisfy $Av = \lambda v$ and $v \neq 0$. Even though the eigenvalue and eigenvector are complex, their real and imaginary parts are very relevant to computations in $\mathbb{R}^n$. Decompose λ and v into their real and imaginary parts, viz

$$\begin{aligned} \lambda &=: \lambda_{\text{Re}} + \mathbb{i}\, \lambda_{\text{Im}} \\ v &=: v_{\text{Re}} + \mathbb{i}\, v_{\text{Im}}. \end{aligned} \tag{A.35}$$

Then

$$\begin{aligned} Av_{\text{Re}} &= \text{real}(Av) = \lambda_{\text{Re}} \cdot v_{\text{Re}} - \lambda_{\text{Im}} \cdot v_{\text{Im}} \\ Av_{\text{Im}} &= \text{imag}(Av) = \lambda_{\text{Im}} \cdot v_{\text{Re}} + \lambda_{\text{Re}} \cdot v_{\text{Im}}. \end{aligned} \tag{A.36}$$

Hence,

$$A \begin{bmatrix} v_{\text{Re}} \\ v_{\text{Im}} \end{bmatrix} = \begin{bmatrix} \lambda_{\text{Re}} I_n & -\lambda_{\text{Im}} I_n \\ \lambda_{\text{Im}} I_n & \lambda_{\text{Re}} I_n \end{bmatrix} \begin{bmatrix} v_{\text{Re}} \\ v_{\text{Im}} \end{bmatrix}. \tag{A.37}$$

If we write $\lambda = |\lambda| e^{\mathbb{i}\,\theta} = |\lambda|\cos(\theta) + \mathbb{i}\,|\lambda|\sin(\theta)$, then (A.37) can be rewritten as

$$A \begin{bmatrix} v_{\mathrm{Re}} \\ v_{\mathrm{Im}} \end{bmatrix} = |\lambda| \underbrace{\begin{bmatrix} \cos(\theta) I_n & -\sin(\theta) I_n \\ \sin(\theta) I_n & \cos(\theta) I_n \end{bmatrix}}_{R(\theta)} \begin{bmatrix} v_{\mathrm{Re}} \\ v_{\mathrm{Im}} \end{bmatrix}, \tag{A.38}$$

where $R(\theta)^\top \cdot R(\theta) = R(\theta) \cdot R(\theta)^\top = I_{2n}$, and hence $R(\theta)$ is an orthogonal matrix. This shows how the complex aspect of the eigenvalue and eigenvector manifests itself as a "kind of rotation" of vectors in the two dimensional subspace

$$\mathrm{span}\{v_{\mathrm{Re}}, v_{\mathrm{Im}}\}$$

by $R(\theta)$, in addition to the scaling by $|\lambda|$. A second benefit of the latter expression is that we then have

$$A^k \begin{bmatrix} v_{\mathrm{Re}} \\ v_{\mathrm{Im}} \end{bmatrix} = |\lambda|^k \underbrace{\begin{bmatrix} \cos(k\theta) I_n & -\sin(k\theta) I_n \\ \sin(k\theta) I_n & \cos(k\theta) I_n \end{bmatrix}}_{R(k\theta)} \begin{bmatrix} v_{\mathrm{Re}} \\ v_{\mathrm{Im}} \end{bmatrix}. \tag{A.39}$$

Figure A.4 illustrates a case where $\lambda = 0.9803 \pm \mathbb{i}\, 0.0965 = 0.985 \angle 5.6$ degrees. The rotating and decaying nature of the solution is clearly seen in the figure. The reader should compare Figs. A.3-(a) and -(d) with Fig. A.4.

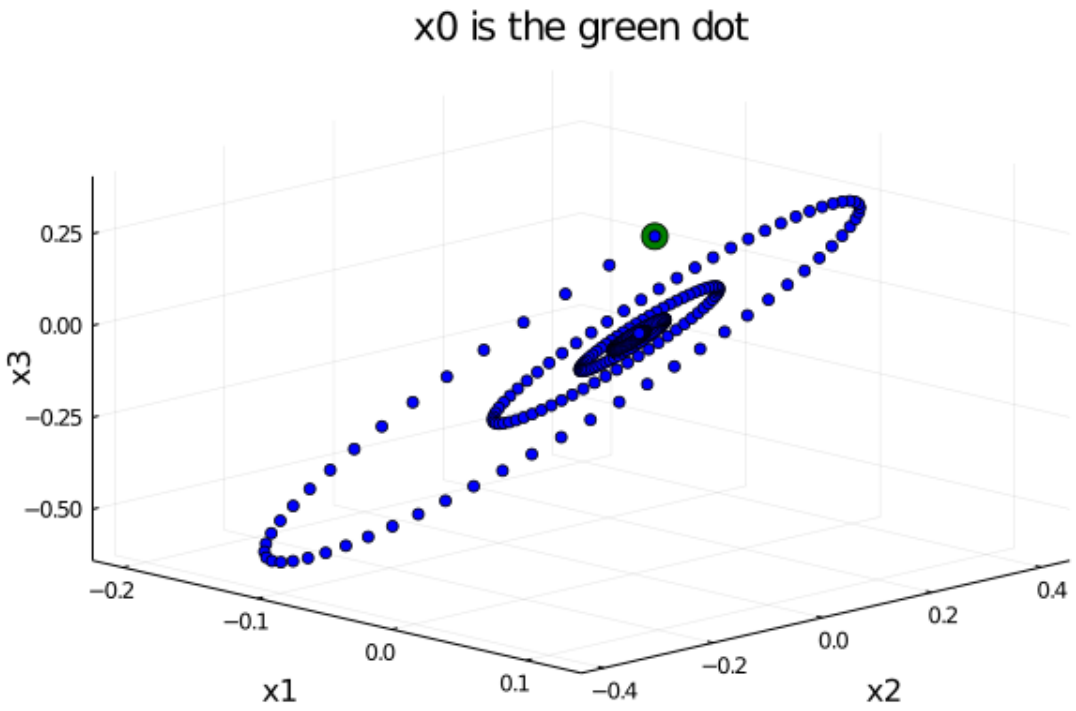


Figure A.4: The eigenvalues of a real 3×3 matrix are computed to be $0.9803 \pm \mathbb{i}\, 0.0965$ and 1.100. The initial condition in green was chosen to be a linear combination of the real and imaginary parts of an eigenvector corresponding to the complex pair of eigenvalues. The resulting solution of (A.28) evolves in the plane defined by $\mathrm{span}\{v_{\mathrm{Re}}, v_{\mathrm{Im}}\}$ as indicated by (A.37) through (A.39). This is very analogous to how complex numbers, when iterated, evolve in the Complex Plane.

A.2 Eigenvalues and Eigenvectors

The study of eigenvalues and eigenvectors is very traditional in Linear Algebra courses. We skipped very important aspects of them in the main portion of the book for a few reasons: (1) time is limited; (2) they get complicated really fast; and (3) their most important applications are the evolution of linear difference equations and the Singular Value Decomposition (SVD), neither of which were covered in the main portion of the text. The usual illustrative application of eigenvalues and eigenvectors is to "diagonalize" a matrix,

which we treated indirectly in Chapter **??**. In the context of Chapter A.1.5 and your Segway Project, it does make sense.

Just in case you are starting here and skipped Appendix A.1 entirely, we start from the beginning.

A.2.1 General Square Matrices

Temporary Def. Let A be an $n \times n$ matrix with real coefficients. A scalar $\lambda \in \mathbb{R}$ is an **eigenvalue** (e-value) of A, if there exists a non-zero vector $v \in \mathbb{R}^n$ such that $A \cdot v = \lambda v$. Any such vector v is called an **eigenvector** (e-vector) associated with λ.

We note that if v is an e-vector, then so is αv for any $\alpha \neq 0$, and therefore, e-vectors are not unique. To find eigenvalues, we need to have conditions under which there exists $v \in \mathbb{R}^n$, $v \neq 0$, such that $A \cdot v = \lambda v$. Here they are,

$$A \cdot v = \lambda v \iff (\lambda I - A) \cdot v = 0 \overset{v \neq 0}{\iff} \det(\lambda I - A) = 0.$$

Example A.3 *Let A be the 2×2 real matrix $A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$. Determine, if any, its e-values and e-vectors.*

Solution: To find e-values, we need to solve

$$\det(\lambda I - A) = \begin{vmatrix} \lambda & -1 \\ 1 & \lambda \end{vmatrix} = \lambda^2 + 1 = 0.$$

We compute the discriminant of this quadratic equation and we find

$$b^2 - 4ac = -4 < 0,$$

and therefore there are no real solutions. Hence, by our *temporary definition*, this 2×2 real matrix does not have any e-values, and hence, neither does it have any e-vectors.

If we were to allow e-values to be complex numbers, then we'd have two e-values corresponding to the two complex solutions of the quadratic equation $\lambda^2 + 1 = 0$, namely, $\lambda_1 = \mathbb{i}$ and $\lambda_2 = -\mathbb{i}$.

We'll see shortly that we'll also need to allow the e-vectors to have complex entries. Hence, we need to generalize our temporary definition. ■

Permanent Definition of Eigenvalues and Eigenvectors

Let A be an $n \times n$ matrix with real or complex coefficients. A scalar $\lambda \in \mathbb{C}$ is an **eigenvalue** (e-value) of A, if there exists a non-zero vector $v \in \mathbb{C}^n$ such that $Av = \lambda v$. Any such vector v is called an **eigenvector** (e-vector) associated with λ.

Eigenvectors are not unique.

- To find e-values, we solve $\det(\lambda I - A) = 0$ because

$$A \cdot v = \lambda v \iff (\lambda I - A) \cdot v = 0 \overset{v \neq 0}{\iff} \det(\lambda I - A) = 0. \tag{A.40}$$

- To find e-vectors, we find any non-zero $v \in \mathbb{C}^n$ such that

$$(\lambda I - A) \cdot v = 0. \tag{A.41}$$

Of course, if you prefer, you can solve $(A - \lambda I)v = 0$ when seeking e-vectors.

Fundamental Theorem of Algebra (and a bit More)

Let A be an $n \times n$ matrix with real or complex coefficients. Then the following statements are true

- $\det(\lambda I - A) = \lambda^n + \alpha_{n-1}\lambda^{n-1} + \cdots \alpha_1\lambda + \alpha_0$, and if A is real, so are the coefficients $\alpha_{n-1}, \ldots, \alpha_0$.
- The degree n polynomial $\lambda^n + \alpha_{n-1}\lambda^{n-1} + \cdots \alpha_1\lambda + \alpha_0$ has n roots $\lambda_1, \ldots, \lambda_n \in \mathbb{C}$ such that

$$\det(\lambda I - A) = (\lambda - \lambda_1)(\lambda - \lambda_2)\cdots(\lambda - \lambda_n).$$

  Each of the roots λ_i, $1 \leq i \leq n$, is an e-value of A.
- The e-values $\{\lambda_1, \ldots, \lambda_n\}$ are said to be **distinct** if $\lambda_i \neq \lambda_k$ for all $i \neq k$.
- If $\lambda_i = \lambda_k$ for some $i \neq k$, then λ_i is a **repeated e-value**. The e-values can then be grouped into $1 \leq p \leq n$ sets of distinct roots $\{\lambda_1, \ldots, \lambda_p\}$ such that

$$\det(\lambda I - A) = (\lambda - \lambda_1)^{m_1}(\lambda - \lambda_2)^{m_2}\cdots(\lambda - \lambda_p)^{m_p}.$$

  The integer m_i is called the **algebraic multiplicity** of λ_i and their sum satisfies $m_1 + m_2 + \cdots + m_p = n$.
- An e-vector associated with λ_i is computed by finding non-zero solutions to (A.41).
- If the matrix A is real, then the e-values occur in **complex conjugate pairs**, that is, if λ_i is an e-value then so is λ_i^*.
- If the matrix A is real and the e-value λ_i is real, then the e-vector v_i can always be chosen to be real, that is, $v_i \in \mathbb{R}^n$ instead of $v_i \in \mathbb{C}^n$.
- There will always be at least one non-zero solution to (A.41), and because any non-zero multiple of a solution is also a solution, there will always be an infinite number of solutions to (A.41).
- If λ_i is a repeated e-value with algebraic multiplicity m_i, then the number of linearly independent e-vectors associated with λ_i is upper bounded by m_i. Another way to say this is, $1 \leq \dim\left(\text{Null}(A - \lambda_i I)\right) \leq m_i$.
- **In Julia**, after `using LinearAlgebra`, the commands are $\Lambda =$ `eigvals(A)` and $V =$ `eigvecs(A)`

Example A.4 *Let A be the 2×2 real matrix that we treated in Example A.3, namely, $A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$. Determine its e-values and e-vectors in the sense of our "permanent" definition.*

Solution: As in Example A.3, to find e-values, we solve

$$\det(\lambda I - A) = \begin{vmatrix} \lambda & -1 \\ 1 & \lambda \end{vmatrix} = \lambda^2 + 1 = 0.$$

We apply the quadratic equation and determine $\lambda_1 = \mathbb{i}$ and $\lambda_2 = -\mathbb{i}$. To find the eigenvectors, we solve

$$(A - \lambda_i I)v_i = 0.$$

The eigenvectors are

$$v_1 = \begin{bmatrix} 1 \\ \mathbb{i} \end{bmatrix}, v_2 = \begin{bmatrix} 1 \\ -\mathbb{i} \end{bmatrix}.$$

Note that the eigenvalues and eigenvectors each form complex conjugate pairs. Indeed,

$$\lambda_2 = \lambda_1^* \text{ and } v_2 = v_1^*.$$

■

Example A.5 *Let A be the $n \times n$ identity matrix. Determine its e-values and e-vectors.*

Solution: $\det(\lambda I - I) = \det((\lambda - 1)I_{=}0 \iff \lambda = 1$. Alternatively, you can compute that $\det(\lambda I - I) = (\lambda - 1)^n$. Hence, the e-value $\lambda = 1$ is repeated n times, that is, $m_1 = n$. What are the e-vectors? We seek to solve

$$(A - \lambda I) \cdot v = 0 \iff (I - 1 \cdot I) \cdot v = 0 \iff 0_n \cdot v = 0,$$

where 0_n is the $n \times n$ matrix of all zeros! Hence, any non-zero vector $v \in \mathbb{R}^n$ is an e-vector. Moreover, if $\{v_1, \ldots, v_n\}$ is a basis for $\mathbb{R}^n$, then $\{v_1, \ldots, v_n\}$ is a set of n linearly independent e-vectors associated with $\lambda_1 = 1$. ■

Example A.6 *Let $a \in \mathbb{R}$ be a constant and let A be the 4×4 matrix below. Determine its e-values and e-vectors.*

$$A = \begin{bmatrix} a & 1 & 0 & 0 \\ 0 & a & 1 & 0 \\ 0 & 0 & a & 1 \\ 0 & 0 & 0 & a \end{bmatrix}.$$

Solution: To find the e-values, we solve

$$\det(\lambda I - A) = \det\left(\begin{bmatrix} (\lambda - a) & -1 & 0 & 0 \\ 0 & (\lambda - a) & -1 & 0 \\ 0 & 0 & (\lambda - a) & -1 \\ 0 & 0 & 0 & (\lambda - a) \end{bmatrix}\right) = (\lambda - a)^4 = 0,$$

and hence there is one distinct e-value $\lambda_1 = a$. To solve for e-vector(s) we consider

$$0 = (A - aI) \cdot v = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot v$$

and we find that the only solutions are multiples of

$$v = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

■

We've seen the extremes! A matrix with a single distinct e-value and a complete set of e-vectors (there were n linearly independent e-vectors associated with the e-value), and another matrix with a single distinct e-value, but only one linearly independent e-vector associated with it.

When the e-values are Distinct, the e-vectors form a Basis

Let A be an $n \times n$ matrix with coefficients in $\mathbb{R}$ or $\mathbb{C}$. If the e-values $\{\lambda_1, \ldots, \lambda_n\}$ are distinct, that is, $\lambda_i \neq \lambda_j$ for all $1 \leq i \neq j \leq n$, then the e-vectors $\{v_1, \ldots, v_n\}$ are linearly independent in ($\mathbb{C}^n$,$\mathbb{C}$).

Restatement of the result: If $\{\lambda_1, \ldots, \lambda_n\}$ are distinct, then $\{v_1, \ldots, v_n\}$ is a basis for ($\mathbb{C}^n$,$\mathbb{C}$). If A is real and its e-values are real, then the e-vectors can be chosen to be real and they form a basis for $\mathbb{R}^n$.

A.2.2 Real Symmetric Matrices

We recall that a real $n \times n$ matrix A is **symmetric** if $A^\top = A$. E-values and e-vectors of symmetric matrices have nicer properties than those of general matrices.

E-values and E-vectors of Symmetric Matrices

- The e-values of a symmetric matrix are real. Because the e-values are real and the matrix is real, we can always chose the e-vectors to be real. Moreover, we can always normalize the e-vectors to have **norm one**.
- Just as with general matrices, the e-values of a symmetric matrix may be distinct or repeated. However, even when an e-value λ_i is repeated m_i times, there are always m_i linearly independent e-vectors associated with it. By applying Gram-Schmidt, we can always chose these e-vectors to be **orthonormal**.
- E-vectors associated with distinct e-values are automatically orthogonal. To be clear,

$$\left(A^\top = A,\ Av_i = \lambda_i v_i, Av_k = \lambda_k v_k,\ \text{and}\ \lambda_i \neq \lambda_k\right) \implies v_i \perp v_k.$$

  Since we can assume they have length one, we have that the e-vectors are **orthonormal**.
- **In summary**, when A is symmetric, there is always an orthonormal basis $\{v_1, v_2, \ldots, v_n\}$ for $\mathbb{R}^n$ consisting of e-vectors of A. In other words, for all $1 \leq i \leq n$, $Av_i = \lambda_i v_i$, $||v_i|| = 1$, and for $k \neq i$, $v_k \perp v_i$.

Factoring a Symmetric Matrix

For every real $n \times n$ symmetric matrix A, there exists an $n \times n$ diagonal matrix Λ and an $n \times n$ orthogonal matrix Q such that

$$A = Q \cdot \Lambda \cdot Q^\top. \tag{A.42}$$

Moreover,

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \quad \text{and} \quad Q = \begin{bmatrix} v_1 & v_2 & \ldots & v_n \end{bmatrix}$$

are constructed from the e-values of A and a corresponding set of orthonormal e-vectors.

Remark 01: From (A.42), $\det(A) = \det(Q) \cdot \det(\Lambda) \cdot \det(Q^\top) = \det(\Lambda) = \lambda_1 \cdot \lambda_2 \cdots \lambda_n$. Hence, a symmetric real matrix A is invertible if, and only if, all of its e-values are non-zero. Moreover, in this case

$$A^{-1} = Q \cdot \Lambda^{-1} \cdot Q^\top.$$

While a similar result holds for general square matrices, it requires inverting the matrix formed by stacking the e-vectors as columns, and hence is not numerically attractive. For symmetric matrices, the corresponding inverse is computed via a matrix transpose.

Remark 02: Using the fact that matrix multiplication can be realized by summing over the product of columns times rows, (A.42) can be rewritten as

$$A = \sum_{i=1}^{n} \lambda_i \left(v_i \cdot v_i^\top\right). \tag{A.43}$$

Equations (A.42) and (A.43) parallel results we will develop for the Singular Value Decomposition or (SVD). Equation (A.42) factors A into a product of three terms consisting of two orthogonal matrices and a diagonal matrix, while (A.43) is an expansion of A into "rank one" matrices.

A.3 Positive Definite Matrices

Some Definitions and Facts

Def. Let P be an $n \times n$ real matrix and $x \in \mathbb{R}^n$. Then $x^\top P x$ is called a **quadratic form**.

Def. An $n \times n$ matrix S is **skew symmetric** if $S^\top = -S$.

Fact If S is skew symmetric, then $x^\top S x = 0$ for all $x \in \mathbb{R}^n$.

Fact Let P an $n \times n$ real matrix and write

$$P = \frac{1}{2}\left(P + P^\top\right) + \frac{1}{2}\left(P - P^\top\right).$$

Then $\left(P + P^\top\right)$ is symmetric, $\left(P - P^\top\right)$ is skew symmetric, and we see that every (real) square matrix can be written as the sum of a symmetric matrix and a skew symmetric matrix.

Fact Let P an $n \times n$ real matrix. Then, for all $x \in \mathbb{R}^n$

$$x^\top P x = \frac{1}{2} x^\top \left(P + P^\top\right) x.$$

Hence, a quadratic form only depends on the symmetric part of a matrix.

Consequence: When working with a quadratic form, $x^\top P x$, one **ALWAYS** assumes that the matrix P is symmetric. Allowing the matrix to be non-symmetric does not increase the generality of the notion of a quadratic form. This is because $x^\top P x = \frac{1}{2} x^\top \left(P + P^\top\right) x$ implies that one can always replace P with its symmetric part!

Fact For an $n \times n$ symmetric real matrix P with e-values $\lambda_1, \ldots, \lambda_n$, let $\lambda_{\max} := \max_{1 \le i \le n} \lambda_i$ and $\lambda_{\min} := \min_{1 \le i \le n} \lambda_i$ be the max and min, respectively over the e-values. Then, for all $x \in \mathbb{R}^n$,

$$\lambda_{\min}\, x^\top x \le x^\top P x \le \lambda_{\max}\, x^\top x. \tag{A.44}$$

Because $x^\top x = ||x||^2$, the above expression is also commonly written as

$$\lambda_{\min}\, ||x||^2 \le x^\top P x \le \lambda_{\max}\, ||x||^2.$$

Both are useful.

Equation (A.44) is established by choosing an orthonormal set of e-vectors for P, $\{v_1, \ldots, v_n\}$, which we know forms a basis for $\mathbb{R}^n$. Hence, for all $x \in \mathbb{R}^n$, there exist coefficients $\alpha_1, \ldots, \alpha_n$ such that $x = \alpha_1 v_1 + \cdots + \alpha_n v_n$. Then, using the two facts we have at our disposal, namely (a) $\{v_1, \ldots, v_n\}$ is orthonormal and (b), $A v_i = \lambda_i v_i$, we compute

$$\begin{aligned} x^\top x &= \alpha_1^2 + \cdots + \alpha_n^2 \\ x^\top P x &= \lambda_1 \alpha_1^2 + \cdots + \lambda_n \alpha_n^2. \end{aligned}$$

It follows that

$$\lambda_{\min} x^\top x = \lambda_{\min}\alpha_1^2 + \cdots + \lambda_{\min}\alpha_n^2 \le \lambda_1\alpha_1^2 + \cdots + \lambda_n\alpha_n^2 \le \lambda_{\max}\alpha_1^2 + \cdots + \lambda_{\max}\alpha_n^2 = \lambda_{\max} x^\top x,$$

showing that (A.44) holds.

Positive Definite and Semidefinite Matrices

Def. A real symmetric matrix P is **positive definite**, if for all $x \in \mathbb{R}^n$, $x \neq 0 \implies x^\top P x > 0$. The common notation for such matrices is $P > 0$.

Def. A real symmetric matrix P is positive semidefinite, if for all $x \in \mathbb{R}^n \implies x^\top P x \geq 0$. The common notation for such matrices is $P \geq 0$.

From (A.44)**, we arrive at the following facts.**

Fact A symmetric matrix P is positive definite if, and only if, all of its eigenvalues are greater than 0.

Fact A symmetric matrix P is positive semidefinite if, and only if, all of its eigenvalues are greater than or equal to 0.

Example A.7 *Determine, which, if any, of the following matrices are positive definite or positive semidefinite.*

$$P_1 = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, P_2 = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, P_3 = \begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix}, P_4 = \begin{bmatrix} 4 & 1 \\ 3 & 4 \end{bmatrix}$$

Solution: Because P_4 is not symmetric, it cannot be positive definite or positive semidefinite! Using Julia, we compute the e-values of P_1, P_2, and P_3

$$\begin{aligned} P_1 &\implies \lambda_1 = 1, \lambda_2 = 3 \implies P > 0 \\ P_2 &\implies \lambda_1 = -1, \lambda_2 = 3 \implies P \not\geq 0 \text{ (neither positive semidefinite nor positive definite)} \\ P_3 &\implies \lambda_1 = 0, \lambda_2 = 5 \implies P \geq 0. \end{aligned}$$

We note that P being positive definite does NOT mean that all of its entries have to be positive! P can have entries with negative values and still be positive definite. We note that all of the entries of P being positive does NOT imply that P is even positive semidefinite. ■.

Computing e-values is a terrible way to determine if a matrix is positive definite or not. The following facts imply that LDLT Factorization can be applied to positive definite and positive semidefinite matrices.

More on Positive Definite and Semidefinite Matrices

Fact A symmetric $n \times n$ matrix P is positive semidefinite if, and only if, there exists a $k \times n$ matrix N such that $N^\top \cdot N = P$.

Fact A symmetric $n \times n$ matrix P is positive definite if, and only if, there exists an $n \times n$ matrix N with linearly independent columns such that $N^\top \cdot N = P$.

Turning these ideas into algorithmic form gives the following:

LDLT or LU Factorization to Check $\mathrm{P} > 0$

Here are two better ways to test whether a matrix is positive definite, positive semidefinite, or neither:

- From Chapter **??**, we can do the LDLT factorization of P, namely

$$Q \cdot P \cdot Q^\top = L \cdot D \cdot L^\top,$$

  where we have used Q to denote the row permutation matrix because P is being used for a symmetric matrix. Then $P > 0 \iff D > 0$, where D is diagonal.

- Straight up LU with no permutations at all: The key fact is that, if an $n \times n$ matrix P is symmetric and invertible, then it can be written as

$$P = L \cdot U;$$

  you can do the factorization without permuting any of the rows of P. Moreover, there is always a diagonal matrix D such that

$$U = D \cdot L^\top.$$

  Determining D from U is trivial: you just normalize by the diagonal of U. Then, $P = L \cdot D \cdot L^\top$ and

$$P > 0 \iff L \cdot D \cdot L^\top > 0 \iff D > 0, \text{ that is, all of the entries on the diagonal of } D \text{ are positive.}$$

- If the LU Factorization without permutations fails, then P is not positive definite, but could be positive semidefinite. To rule out the latter, you need to run the full LDLT algorithm and checkl that the diagonal of D has at least one negative entry or not.

Example A.8 *Using the LU Factorization without row permutations, determine if the randomly generated symmetric matrix P is positive definite or not.*

$$P = \begin{bmatrix} 8.241e-01 & 1.171e+00 & 1.117e+00 & 1.706e+00 & 1.021e+00 \\ 1.171e+00 & 1.574e+00 & 8.547e-01 & 1.102e+00 & 2.871e-01 \\ 1.117e+00 & 8.547e-01 & 1.238e+00 & 7.506e-01 & 1.291e+00 \\ 1.706e+00 & 1.102e+00 & 7.506e-01 & 2.943e-01 & 9.570e-01 \\ 1.021e+00 & 2.871e-01 & 1.291e+00 & 9.570e-01 & 1.448e+00 \end{bmatrix}.$$

Solution: We do the LU Factorization without permutations and obtain

$$L = \begin{bmatrix} 1.000e+00 & 0.000e+00 & 0.000e+00 & 0.000e+00 & 0.000e+00 \\ 1.421e+00 & 1.000e+00 & 0.000e+00 & 0.000e+00 & 0.000e+00 \\ 1.356e+00 & 8.151e+00 & 1.000e+00 & 0.000e+00 & 0.000e+00 \\ 2.070e+00 & 1.469e+01 & 1.616e+00 & 1.000e+00 & 0.000e+00 \\ 1.239e+00 & 1.294e+01 & 1.649e+00 & 5.893e-01 & 1.000e+00 \end{bmatrix}$$

$$U = \begin{bmatrix} 8.241e-01 & 1.171e+00 & 1.117e+00 & 1.706e+00 & 1.021e+00 \\ 0.000e+00 & -8.990e-02 & -7.328e-01 & -1.321e+00 & -1.164e+00 \\ 0.000e+00 & 0.000e+00 & 5.696e+00 & 9.203e+00 & 9.393e+00 \\ 0.000e+00 & 0.000e+00 & 0.000e+00 & 1.296e+00 & 7.636e-01 \\ 0.000e+00 & 0.000e+00 & 0.000e+00 & 0.000e+00 & -6.909e-01 \end{bmatrix}$$

We extract the diagonal of U and we form $D \cdot L^\top$

$$D = \begin{bmatrix} 8.241e-01 & 0.000e+00 & 0.000e+00 & 0.000e+00 & 0.000e+00 \\ 0.000e+00 & -8.990e-02 & 0.000e+00 & 0.000e+00 & 0.000e+00 \\ 0.000e+00 & 0.000e+00 & 5.696e+00 & 0.000e+00 & 0.000e+00 \\ 0.000e+00 & 0.000e+00 & 0.000e+00 & 1.296e+00 & 0.000e+00 \\ 0.000e+00 & 0.000e+00 & 0.000e+00 & 0.000e+00 & -6.909e-01 \end{bmatrix}$$

$$D \cdot L^\top = \begin{bmatrix} 8.241e-01 & 1.171e+00 & 1.117e+00 & 1.706e+00 & 1.021e+00 \\ 0.000e+00 & -8.990e-02 & -7.328e-01 & -1.321e+00 & -1.164e+00 \\ 0.000e+00 & 0.000e+00 & 5.696e+00 & 9.203e+00 & 9.393e+00 \\ 0.000e+00 & 0.000e+00 & 0.000e+00 & 1.296e+00 & 7.636e-01 \\ 0.000e+00 & 0.000e+00 & 0.000e+00 & 0.000e+00 & -6.909e-01 \end{bmatrix}$$

and we recognize, that indeed, $U = D \cdot L^\top$.

Back to the question of determining whether P is positive definite? We see that D has non-positive entries and therefore P is not positive definite. We only formed $D \cdot L^\top$ to illustrate that $P = L \cdot D \cdot L^\top$. ■

The following results are primarily of use for "hand calculations" or proving results about positive definite matrices. We include them for completeness.

Schur Complement Theorem: A way to Decompose the Test for Being Positive Definite

Suppose that A is $n \times n$, symmetric, and invertible, B is $n \times m$, C is $m \times m$, symmetric, and invertible, and

$$M := \begin{bmatrix} A & B \\ B^\top & C \end{bmatrix},$$

which is then $(n+m) \times (n+m)$ and symmetric. Under these conditions, the following three statements are equivalent:

(a) $M > 0$.

(b) $A > 0$, and $C - B^\top \cdot A^{-1} \cdot B > 0$.

(c) $C > 0$, and $A - B \cdot C^{-1} \cdot B^\top > 0$.

Remarks:

- $C - B^\top \cdot A^{-1} \cdot B$ is called the Schur Complement of A in M.
- $A - B \cdot C^{-1} \cdot B^\top$ is called the Schur Complement of C in M.

A.4 Singular Value Decomposition or SVD

The material here is inspired by a handout prepared by Prof. James Freudenberg, EECS, University of Michigan.

A.4.1 Motivation

In abstract linear algebra, a set of vectors is either linearly independent or not. There is nothing in between. For example, the set of vectors

$$\left\{ v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, v_2 = \begin{bmatrix} 0.999 \\ 1 \end{bmatrix} \right\}$$

is linearly independent. In this case, one looks at the set of vectors and says, yes, BUT, the vectors are "almost" dependent because when one computes the determinant

$$\det \begin{bmatrix} 1 & 0.999 \\ 1 & 1 \end{bmatrix} = 0.001,$$

the result is pretty small, so it should be fine to call them dependent.

Well, what about the set

$$\left\{ v_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, v_2 = \begin{bmatrix} 10^4 \\ 1 \end{bmatrix} \right\}?$$

When you form the matrix and check the determinant, you get

$$\det \begin{bmatrix} 1 & 10^4 \\ 0 & 1 \end{bmatrix} = 1,$$

which seems pretty far from zero. So are these vectors "adequately" linearly independent?

Maybe not! Let's note that

$$\begin{bmatrix} 1 & 10^4 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 10^{-4} & 0 \end{bmatrix} = \begin{bmatrix} 1 & 10^4 \\ 10^{-4} & 1 \end{bmatrix},$$

and its determinant is zero! Hence, it's possible to add a very small perturbation to one of the vectors and make the set linearly dependent! This cannot be good.

A.4.2 Definition and Main Theorem

Rectangular Diagonal Matrix

An $n \times m$ matrix Σ is a **Rectangular Diagonal Matrix** if

$$\Sigma_{ij} = 0 \text{ for } i \neq j.$$

Alternative and equivalent way to define Rectangular Diagonal is

(a) (tall matrix) $n > m$ $\quad \Sigma = \begin{bmatrix} \Sigma_d \\ 0 \end{bmatrix}$, where Σ_d is an $m \times m$ diagonal matrix.

(b) (wide matrix) $n < m$ $\quad \Sigma = \begin{bmatrix} \Sigma_d & 0 \end{bmatrix}$, where Σ_d is an $n \times n$ diagonal matrix.

The **diagonal** of Σ is defined to be the diagonal of Σ_d.

Singular Value Decomposition (Main Theorem)

Every $n \times m$ real matrix A can be factored as

$$A = U \cdot \Sigma \cdot V^\top,$$

where U is an $n \times n$ orthogonal matrix, V is an $m \times m$ orthogonal matrix, Σ is an $n \times m$ rectangular diagonal matrix, and the diagonal of Σ,

$$\text{diag}(\Sigma) = [\sigma_1, \sigma_2, \cdots, \sigma_p],$$

satisfies $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$, for $p := \min(n, m)$.

Moreover, the columns of U are eigenvectors of $A \cdot A^\top$, the columns of V are eigenvectors of $A^\top \cdot A$, and $\{\sigma_1^2, \sigma_2^2, \ldots, \sigma_p^2\}$ are eigenvalues of both $A^\top \cdot A$ and $A \cdot A^\top$.

The **Singular Values of A** are the elements $\{\sigma_1, \ldots, \sigma_p\}$ from the diagonal of Σ.

Another way to write the SVD of A is

$$A = \sigma_1 u_1 \cdot v_1^\top + \sigma_2 u_2 \cdot v_2^\top + \cdots + \sigma_p u_p \cdot v_p^\top,$$

where u_i and v_i are columns of U and V respectively.

$$U = \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix} \text{ and } V = \begin{bmatrix} v_1 & v_2 & \cdots & v_m \end{bmatrix}. \tag{A.45}$$

This formula follows from our matrix multiplication formulation through the sum over columns times rows, where we note that the columns of V are the rows of $V^\top$.

Rank and Nullity of a Matrix

The **rank** of an $n \times m$ matrix A is the dimension of its column span and the **nullity** of A is the dimension of its null space. Let r be the number of non-zero singular values of A. Then

Fact $\text{rank}(A) := \dim \text{col span}\{A\} = r$.

Fact $\text{nullity}(A) := \dim \text{null}(A) = m - r$.

Example A.9 *Determine the SVD of A as well as its rank and nullity,*

$$A = \begin{bmatrix} 1 & 10^4 \\ 0 & 1 \end{bmatrix}.$$

Solution: Using the LinearAlgebra package in Julia, we find

$$U = \begin{bmatrix} 1.000e+00 & -1.000e-04 \\ 1.000e-04 & 1.000e+00 \end{bmatrix}$$
$$\Sigma = \begin{bmatrix} 1.000e+04 & 0.000e+00 \\ 0.000e+00 & 1.000e-04 \end{bmatrix}$$
$$V = \begin{bmatrix} 1.000e-04 & -1.000e+00 \\ 1.000e+00 & 1.000e-04 \end{bmatrix}$$

There are two non-zero singular values, and thus $r = 2$. It follows that $\text{rank}(A) = 2$ and $\text{nullity}(A) = 0$.

Information about the "near" linear dependence of the columns of A is in the diagonal matrix Σ. There are two singular values, $\sigma_1 = 10^4$ and $\sigma_2 = 10^{-4}$. Their ratio is 10^8, which is an indicator that these vectors are "nearly linearly dependent". "Numerically", one would say that $r = 1$ and hence $\text{rank}(A) = r = 1$ and $\text{nullity}(A) = 2 - r = 1$. ■

A.4.3 Numerical Linear Independence

Illustration: 5×5 matrix. For

$$A = \begin{bmatrix} -32.57514 & -3.89996 & -6.30185 & -5.67305 & -26.21851 \\ -36.21632 & -11.13521 & -38.80726 & -16.86330 & -1.42786 \\ -5.07732 & -21.86599 & -38.27045 & -36.61390 & -33.95078 \\ -36.51955 & -38.28404 & -19.40680 & -31.67486 & -37.34390 \\ -25.28365 & -38.57919 & -31.99765 & -38.36343 & -27.13790 \end{bmatrix},$$

and the Julia commands

```
using LinearAlgebra

A=[-32.57514 -3.89996 -6.30185 -5.67305 -26.21851;
-36.21632 -11.13521 -38.80726 -16.86330 -1.42786;
-5.07732 -21.86599 -38.27045 -36.61390 -33.95078;
-36.51955 -38.28404 -19.40680 -31.67486 -37.34390;
-25.28365 -38.57919 -31.99765 -38.36343 -27.13790 ]

(U ,Sigma, V) = svd(A)
```

one obtains

$$U = \begin{bmatrix} -2.475e-01 & -5.600e-01 & 4.131e-01 & 5.759e-01 & 3.504e-01 \\ -3.542e-01 & -5.207e-01 & -7.577e-01 & -1.106e-02 & -1.707e-01 \\ -4.641e-01 & 6.013e-01 & -1.679e-01 & 6.063e-01 & -1.652e-01 \\ -5.475e-01 & -1.183e-01 & 4.755e-01 & -3.314e-01 & -5.919e-01 \\ -5.460e-01 & 1.992e-01 & -2.983e-02 & -4.369e-01 & 6.859e-01 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 1.325e+02 & 0.000e+00 & 0.000e+00 & 0.000e+00 & 0.000e+00 \\ 0.000e+00 & 3.771e+01 & 0.000e+00 & 0.000e+00 & 0.000e+00 \\ 0.000e+00 & 0.000e+00 & 3.342e+01 & 0.000e+00 & 0.000e+00 \\ 0.000e+00 & 0.000e+00 & 0.000e+00 & 1.934e+01 & 0.000e+00 \\ 0.000e+00 & 0.000e+00 & 0.000e+00 & 0.000e+00 & 7.916e-01 \end{bmatrix}$$

$$V = \begin{bmatrix} 4.307e-01 & 8.839e-01 & -5.303e-02 & 8.843e-02 & -1.503e-01 \\ 4.309e-01 & -2.207e-01 & -1.961e-01 & 7.322e-01 & 4.370e-01 \\ 4.617e-01 & -8.902e-02 & 7.467e-01 & -3.098e-01 & 3.539e-01 \\ 4.730e-01 & -3.701e-01 & 7.976e-02 & 1.023e-01 & -7.890e-01 \\ 4.380e-01 & -1.585e-01 & -6.283e-01 & -5.913e-01 & 1.968e-01 \end{bmatrix}$$

Because the **smallest singular value $\sigma_5 = 0.7916$ is less than 1% of the largest singular value $\sigma_1 = 132.5$**, in many cases, one would say that the numerical rank of A was 4 instead of 5.

This notion of numerical rank can be formalized by asking the following question: Suppose $\text{rank}(A) = r$. How far away is A from a matrix of rank strictly less than r?

The numerical rank of a matrix is based on the expansion in (A.4.2), which is repeated here for convenience,

$$A = U \cdot \Sigma \cdot V^\top = \sum_{i=1}^{p} \sigma_i u_i \cdot v_i^\top = \sigma_1 u_1 \cdot v_1^\top + \sigma_2 u_2 \cdot v_2^\top + \cdots + \sigma_p u_p \cdot v_p^\top,$$

where $p = \min\{m, n\}$, and once again, the singular values are ordered such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$. Each term $u_i \cdot v_i^\top$ is a rank-one matrix. The following will help you understand the expansion.

Exercises or Facts:

- $A \cdot A^\top = U \cdot \Sigma \cdot \Sigma^\top \cdot U^\top = \sum_{i=1}^{p} (\sigma_i)^2 \; u_i \cdot u_i^\top$
- $A^\top \cdot A = V \cdot \Sigma^\top \cdot \Sigma \cdot V^\top = \sum_{i=1}^{p} (\sigma_i)^2 \; v_i \cdot v_i^\top$
- $\left(u_i \cdot v_i^\top\right) \cdot v_j = \begin{cases} u_i & j = i \\ 0 & j \neq i \end{cases}$ and hence rank$(u_i \cdot v_i^\top) = 1$ and nullity$(u_i \cdot v_i^\top) = m - 1$
- $\left(u_i \cdot u_i^\top\right) \cdot u_j = \begin{cases} u_i & j = i \\ 0 & j \neq i \end{cases}$ and hence rank$(u_i \cdot u_i^\top) = 1$ and nullity$(u_i \cdot u_i^\top) = n - 1$
- $\left(v_i \cdot v_i^\top\right) \cdot v_j = \begin{cases} v_i & j = i \\ 0 & j \neq i \end{cases}$ and hence rank$(v_i \cdot v_i^\top) = 1$ and nullity$(v_i \cdot v_i^\top) = m - 1$
- $v_i \cdot v_i^\top$, and $u_i \cdot u_i^\top$ have e-values $\lambda_1 = 1$ distinct and $\lambda_2 = 0$ repeated $m - 1$ and $n - 1$ times, respectively.
- **Hint:** $\left(u_i \cdot v_i^\top\right) \cdot v_j = u_i \cdot \left(v_i^\top \cdot v_j\right) = \begin{cases} u_i & j = i \\ 0 & j \neq i \end{cases}$ because the $\{v_1, v_2, \ldots, v_m\}$ are orthonormal.

So far, we have only defined the norm of a vector. However, it is also useful to measure the "length" of matrices.

Def. (Induced Matrix Norm) Given an $n \times m$ real matrix A, the **matrix norm induced by the Euclidean vector norm** is given by:

$$||A|| := \max_{x^\top x = 1} ||Ax|| = \sqrt{\lambda_{\max}(A^\top A)}$$

where $\lambda_{\max}(A^\top A)$ denotes the largest eigenvalue of the matrix $A^\top A$. **(Recall that the matrices of the form $A^\top A$ are at least positive semidefinite and hence their e-values are real and non-negative.)** Therefore, the square root exists.

Numerical Rank

Facts: Suppose that rank$(A) = r$, so that σ_r is the smallest non-zero singular value of A.

(i) If an $n \times m$ matrix E satisfies $||E|| < \sigma_r$, then rank$(A + E) \geq r$.

(ii) There exists an $n \times m$ matrix E with $||E|| = \sigma_r$ and rank$(A + E) < r$.

(iii) In fact, for $E = -\sigma_r u_r v_r^\top$, rank$(A + E) = r - 1$.

(iv) Moreover, for $E = -\sigma_r u_r v_r^\top - \sigma_{r-1} u_{r-1} v_{r-1}^\top$, rank$(A + E) = r - 2$.

Corollary: Suppose A is square and invertible. Then σ_r measures the distance from A to the nearest singular matrix.

Illustration Continued

```
u5=U[:,5]; v5=V[:,5]; sig5=Sigma[5]
E=-sig5*u5*v5'
# Induced Norm
M=E'*E
SquareRootEigs=(abs.(eigvals(E'*E))).^0.5
#
(U ,Sigma2, V) = svd(A+E)
```

$$E = \begin{bmatrix} 4.169e-02 & -1.212e-01 & -9.818e-02 & 2.189e-01 & -5.458e-02 \\ -2.031e-02 & 5.906e-02 & 4.784e-02 & -1.066e-01 & 2.659e-02 \\ -1.966e-02 & 5.716e-02 & 4.629e-02 & -1.032e-01 & 2.574e-02 \\ -7.041e-02 & 2.048e-01 & 1.658e-01 & -3.697e-01 & 9.220e-02 \\ 8.160e-02 & -2.373e-01 & -1.922e-01 & 4.284e-01 & -1.068e-01 \end{bmatrix}$$

$$\sqrt{\lambda_i(E^\top \cdot E)} = \begin{bmatrix} 7.376e-09 \\ 2.406e-09 \\ 1.977e-09 \\ 4.163e-09 \\ 7.916e-01 \end{bmatrix}$$

$$\Sigma_2 = \begin{bmatrix} 1.325e+02 & 0.000e+00 & 0.000e+00 & 0.000e+00 & 0.000e+00 \\ 0.000e+00 & 3.771e+01 & 0.000e+00 & 0.000e+00 & 0.000e+00 \\ 0.000e+00 & 0.000e+00 & 3.342e+01 & 0.000e+00 & 0.000e+00 \\ 0.000e+00 & 0.000e+00 & 0.000e+00 & 1.934e+01 & 0.000e+00 \\ 0.000e+00 & 0.000e+00 & 0.000e+00 & 0.000e+00 & 1.775e-15 \end{bmatrix}$$

We added a matrix with norm 0.7916 and made the (exact) rank drop from 4 to 5! How cool is that? This example shows that SVD can exactly measure how close a matrix is to being singular. We also see that $E^\top \cdot E$ has rank one: there is one non-zero e-value and the rest are (essentially) zero as the theory promised.

Other Interesting and Useful Facts

(a) **Null space:** $\text{null}(A) := \{x \in \mathbb{R}^m \mid Ax = 0\}$

(b) **Range:** $\text{range}(A) := \{y \in \mathbb{R}^n \mid \text{ such that } \ y = Ax \ \text{ for some } \ x \in \mathbb{R}^m\}$

(c) **Fact:** Suppose $A = U \cdot \Sigma \cdot V^\top$. Then the columns of U corresponding to non-zero singular values are a basis for $\text{range}(A)$ and the columns of V corresponding to zero singular values are a basis for $\text{null}(A)$, viz

$$\begin{aligned} \text{range}(A) &:= \text{span}\{u_1, ..., u_r\}, \text{and} \\ \text{null}(A) &:= \text{span}\{v_{r+1}, ..., v_m\}. \end{aligned}$$

(d) The SVD can also be used to compute an "effective" range and an "effective" null space of a matrix.

(e) **Fact:** Suppose that $\sigma_1 \geq ... \geq \sigma_r > \delta \geq \sigma_{r+1} \geq ...\sigma_p \geq 0$, so that r is the "effective" or "numerical rank" of A. (Note the δ inserted between σ_r and σ_{r+1} to denote the break point.)

(f) **Fact:** Let $\text{range}_{\text{eff}}(A)$ and $\text{null}_{\text{eff}}(A)$ denote the effective range and effective null space of A, respectively. Then we can calculate bases for these subspaces by choosing appropriate singular vectors:

$$\begin{aligned} \text{range}_{\text{eff}}(A) &:= \text{span}\{u_1, ..., u_r\}, \text{and} \\ \text{null}_{\text{eff}}(A) &:= \text{span}\{v_{r+1}, ..., v_m\}. \end{aligned}$$

A.5 Linear Transformations and Matrix Representations

Def. A function $L : \mathbb{R}^m \to \mathbb{R}^n$ is a **linear transformation** if for all $x, z \in \mathbb{R}^m$, $\alpha, \beta \in \mathbb{R}$,

$$L(\alpha x + \beta z) = \alpha L(x) + \beta L(z). \tag{A.46}$$

Just as with checking the subspace property, one can break the condition (A.46) into two separate properties,

$$\begin{aligned} L(x + z) &= L(x) + L(z) \\ L(\alpha x) &= \alpha L(x). \end{aligned}$$

You already know at least one linear transformation from $\mathbb{R}^m$ to $\mathbb{R}^n$, namely, let A be an $n \times m$ real matrix and for $x \in \mathbb{R}^m$, define $L : \mathbb{R}^m \to \mathbb{R}^n$ by

$$L(x) = Ax. \tag{A.47}$$

It is straightforward to show that (A.46) holds and thus we leave that to you. If you are trying to think of a clever linear transformation that is not built from a matrix, but you cannot come up with one, well, there is a reason for that: there are none!

Linear Transformations from $\mathbb{R}^m$ to $\mathbb{R}^n$ are Kind of Boring

Fact: Let $L : \mathbb{R}^m \to \mathbb{R}^n$ be a linear transformation. Then there exists an $n \times m$ real matrix A such that, for every $x \in \mathbb{R}^m$

$$L(x) = Ax.$$

The matrix A has a name: it is called the **matrix representation of** L. It's computation is relatively easy. Let $\{e_1, e_2, \ldots, e_m\}$ be the natural basis vectors for $\mathbb{R}^m$. Define

$$a_j^{\text{col}} := L(e_i) \text{ and } A := \begin{bmatrix} a_1^{\text{col}} & a_2^{\text{col}} & \cdots & a_m^{\text{col}} \end{bmatrix} = \begin{bmatrix} L(e_1) & L(e_2) & \cdots & L(e_m) \end{bmatrix}.$$

Then $a_j^{\text{col}} \in \mathbb{R}^n$ because $L(x) \in \mathbb{R}^n$ for all $x \in \mathbb{R}^m$. Now, because $\{e_1, e_2, \ldots, e_m\}$ is a basis, we have

$$x \in \mathbb{R}^m \iff x = x_1e_1 + x_2e_2 + \cdots + x_me_m.$$

Because L is a linear transformation,

$$\begin{aligned} L(x) &= L(x_1e_1 + x_2e_2 + \cdots + x_me_m) \\ &= x_1L(e_1) + x_2L(e_2) + \cdots + x_mL(e_m) \\ &= x_1a_1^{\text{col}} + x_2a_2^{\text{col}} + \cdots + x_ma_m^{\text{col}} \\ &= Ax. \end{aligned}$$

We will give you just a hint that **there are interesting linear transformations**, but to do that, we need to build an interesting vector space, the set of polynomials of degree less than or equal to n, namely

$$P_n(t) := \{a_0 + a_1t + a_2t^2 + \cdots + a_nt^n \,|a_i \in \mathbb{R}, 0 \leq i \leq n\}.$$

We note that everything in $P_n(t)$ appears to be a linear combination of the set of monomials, $\{1, t, t^2, \ldots, t^n\}$. Indeed, if you take a bit more Linear Algebra, such as Math 217, you can make sense of the monomials as being vectors, and not just any vectors, but **basis vectors** for $P_n(t)$. Hence,

$$P_n(t) = \text{span}\{1, t, t^2, \ldots, t^n\}.$$

It is clear that if you add any two polynomials of degree less than or equal to n, you get another polynomial of degree less than or equal to n. If you multiply a polynomial of degree less than or equal to n by a real number, you get another polynomial of degree less than or equal to n. This tells you that $P_n(t)$ satisfies all the properties of being a **vector space**: it is **closed under linear combinations**! And yes, in abstract Linear Algebra, we call the elements of $P_n(t)$ vectors.

We define $L : P_n(t) \to P_n(t)$ by $L(p(t)) := \frac{dp(t)}{dt}$, the first derivative of the polynomial $p(t)$, that is, for those of you who have taken Calculus I,

$$L(a_0 + a_1t + a_2t^2 + \cdots + a_nt^n) := a_1 + 2a_2t + 3a_3t^2 + \cdots + na_nt^{n-1}. \tag{A.48}$$

The rules of differentiation imply that L satisfies (A.46), that is, the rules of being a linear transformation. We note that L defined in (A.48) does not look like it comes from a matrix.

Remark: Bummer! Since L does not appear to come from a matrix, does that mean that we cannot apply to it any of the computational tools that we have developed in ROB 101? The emphatic answer is: we absolutely can apply them! How? We'll show below that there is a matrix hiding inside of L! While this is quite a bit beyond the scope of ROB 101, we feel that it might provide additional motivation for you to take a more advanced Linear Algebra course!

Definition: Let $\{v\} := \{v_1, v_2, \ldots, v_k\}$ be a basis for a (real) vector space V. We know that for any vector $x \in V$, there exist[2] real coefficients $\alpha_1, \alpha_2, \ldots, \alpha_k$ such that

$$x = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k. \tag{A.49}$$

The column vector built by expressing x as a linear combination of the basis vectors in $\{v\}$,

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{bmatrix} \in \mathbb{R}^k, \tag{A.50}$$

is called the **representation of x with respect to the basis** $\{v_1, v_2, \ldots, v_k\}$. A nice notation for it is

$$[x]_{\{v\}} := \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{bmatrix} \in \mathbb{R}^k. \tag{A.51}$$

The representation of x in V is a vector in $\mathbb{R}^k$.

Let's apply this idea to the vector space $V := P_n(t)$ and its very nice basis

$$\{v\} = \{1, t, t^2, \ldots, t^n\}.$$

We noted earlier that any vector in "$x \in V$" (that is, $p(t) \in P_n(t)$) can be written as

$$x = p(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n,$$

and hence,

$$[x]_{\{v\}} := \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \in \mathbb{R}^{n+1}. \tag{A.52}$$

It's kind of obvious, we add two polynomials of the same degree by adding their coefficients, and (A.52) is simply enticing us to do just that. [In other words, do not overthink it!]

Hence, if polynomials of degree less than or equal to n can be represented by columns of numbers, can differentiation be represented by a matrix? The answer is yes. As in (A.48), we define $L : V \to V$ by if $x = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n \in V$,

$$L(x) := a_1 + 2a_2 t + 3a_3 t^2 + \cdots + n a_n t^{n-1}. \tag{A.53}$$

We now break this up and apply L to each of the basis elements

$$\{v\} := \{v_1 = 1, v_2 = t, v_3 = t^2, \ldots, v_{n+1} = t^n\},$$

yielding

$$L(v_1) = 0, L(v_2) = v_1, L(v_3) = 2v_2, \ldots, L(v_{n+1}) = n v_n, \tag{A.54}$$

and note that

$$[L(v_1)]_{\{v\}} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}, [L(v_2)]_{\{v\}} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}, [L(v_3)]_{\{v\}} = \begin{bmatrix} 0 \\ 2 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}, \ldots, [L(v_n)]_{\{v\}} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ n-1 \\ 0 \\ 0 \end{bmatrix}, [L(v_{n+1})]_{\{v\}} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ n \\ 0 \end{bmatrix}. \tag{A.55}$$

[2]The coefficients are actually unique. You should you prove that if you have two such sets of coefficients that they have to be equal. It follows form the definition of linear Independence!

We use the above vectors as the columns of a matrix A, where $a_j^{\text{col}} := [L(v_j)]_{\{v\}}$,

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & n-1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & n \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{A.56}$$

We next note that for $x = a_0 + a_1 t + \cdots + a_{n-2}t^{n-2} + a_{n-1}t^{n-1} + a_n t^n \in V$, its representation is given in (A.52) and that

$$\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & n-1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & n \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-2} \\ a_{n-1} \\ a_n \end{bmatrix} = \begin{bmatrix} a_1 \\ 2a_2 \\ \vdots \\ (n-1)a_{n-1} \\ na_n \\ 0 \end{bmatrix}; \tag{A.57}$$

moreover, the right hand side of the equation is the representation of

$$\frac{d}{dt}\left(a_0 + a_1 t + \cdots + a_{n-2}t^{n-2} + a_{n-1}t^{n-1} + a_n t^n\right) = a_1 + 2a_2 t + \cdots + (n-1)a_{n-1}t^{n-2} + na_n t^{n-1}$$

with respect to the monomials. In other symbols,

$$A\,[x]_{\{v\}} = [L(x)]_{\{v\}}. \tag{A.58}$$

A is called the **matrix representation of L with respect to the basis $\{v\}$**. In Robotics, we often need to differentiate signals in real-time on our robots. We do it using versions of (A.57) and (A.58), which transform differentiation into matrix multiplication!

A.6 Affine Transformations

All functions $f : \mathbb{R}^m \to \mathbb{R}^n$ that fail (A.46) are nonlinear! That does not seem very discerning. There are a few more classes of functions called out, and one in particular is very important in Linear Algebra is **Definition:** A function $f : \mathbb{R}^m \to \mathbb{R}^n$ is **affine** if there exists a constant vector $b \in \mathbb{R}^n$ and an $n \times m$ real matrix A such that

$$f(x) = Ax + b. \tag{A.59}$$

A bit more abstract definition is $f : \mathbb{R}^m \to \mathbb{R}^n$ is **affine** if there exists a constant vector $b \in \mathbb{R}^n$ such that the function $L : \mathbb{R}^m \to \mathbb{R}^m$ by

$$L(x) := f(x) - b$$

is linear. When dealing with $\mathbb{R}^n$ and $\mathbb{R}^m$, the two definitions are equivalent, while the more abstract definition extends to more general settings.

Appendix B

What is an Ordinary Differential Equation?

Learning Objectives

- Provide motivation for considering equations with derivatives.
- Provide a numerical means to approximate a solution to a differential equation.
- Prepare you a bit for Math 216.

Outcomes

- See $F = ma$ in a new light
- Define an ordinary differential equation, aka an ODE
- Learn how to pass from a differential equation to a difference equation
- Relate the process of iterating a difference equation to the process of solving an ODE

Equations with a derivative in them are called **differential equations**. In Robotics, we use differential equations to understand the motion of robots, such as Cassie Blue, or in the case of Project #3, a Segway. The topic of differential equations is typically delayed until a fourth-semester Calculus course. Moreover, such courses focus almost exclusively on closed-form solutions to differential equations. As you can imagine, we're not a big fan of that. Here, you will see that tools we have developed so far in ROB 101 allow us to develop elementary numerical tools for analyzing the solutions of interesting differential equations.

B.1 Preliminaries: Expanding our Concept of an Equation

Equations in a scalar variable $x \in \mathbb{R}$ probably seem pretty trivial to you by now. Examples we have seen include:

- Linear equation: $ax = b$;
- Quadratic equation: $ax^2 + bx + c = 0$;
- Cubic equation: $ax^3 + bx^2 + cx + d = 0$; or
- Trigonometric equation: $A\cos(x) + B\sin(x) + C = 0$.

One obvious way to increase the generality of our analysis tools for equations is to include several variables, $x \in \mathbb{R}^n$, $n > 1$, and we've done a bunch of that in ROB 101 as well, where we studied how to find solutions to

- System of linear equations: $Ax = b$; and
- System of nonlinear equations: $F(x) = 0$.

In the case of linear systems, we explored map building and linear regression (easy version of Machine Learning) as cool application problems. For nonlinear systems, we looked at the position of a robot gripper in $\mathbb{R}^2$ and sought values for the angles of its joints so that the gripper could be placed in a desired position.

Another way to increase the generality of our analysis tools is to expand our concept of equations to include time as a variable! This is a big conceptual step: instead of the values in our equation depending on constants or other powers of our variable x, as in the examples we listed above, we will have the value of our variable x at time t depend on x at some other time, say $t - \delta t$, for some $\delta t > 0$.

B.2 Time in a Digital Computer is Discrete

To get our heads around functions that depend on time, we'll start with "time" as it is treated in a digital computer, namely, time is a sequential variable, such as

$$k = 1, 2, 3, \ldots. \tag{B.1}$$

Digital time is similar to our *human* notion of time in that it is strictly increasing, but whereas our human notion of time is continuous (it can be divided into ever smaller increments as long as we ignore the rules of Quantum Mechanics), *digital* time is fundamentally discrete in that it increases by non-zero increments.

Suppose we denote the value of our variable x at time k by $x[k]$, and we define an *equation for x at the next time*, $k + 1$, by

$$x[k + 1] = \frac{1}{2}x[k] + 1. \tag{B.2}$$

Equation (B.2) says that if we know the value of our variable x at time k, we can find the value of x at time $k+1$ by multiplying $x[k]$ by one-half and adding one to it. That seems easy enough. OK, then, what is the value of x at time $k = 4$?

To compute $x[4]$, we need to know $x[3]$. To compute $x[3]$, we need to know $x[2]$. To compute $x[2]$, we need to know $x[1]$. If we suppose that time starts at $k = 1$, then we cannot go back any further and we realize that somehow $x[1]$ must be given to us!

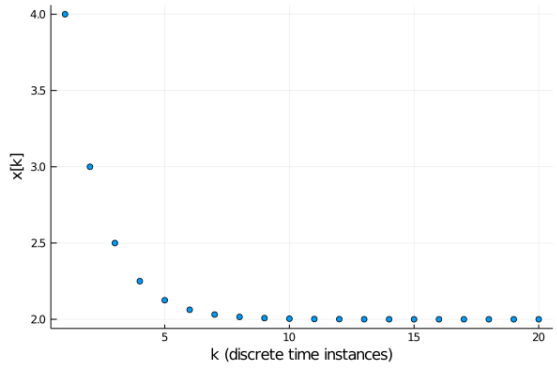


Figure B.1: Plot of the function $x : [1, 2, \ldots, 20] \to \mathbb{R}$ at discrete instances of time.

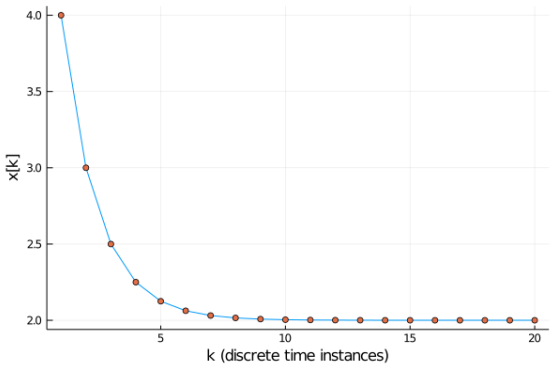


Figure B.2: Plot of the function $x : [1, 2, \ldots, 20] \to \mathbb{R}$, with the "dots" connected, giving us the impression that time is almost continuous.

Equation (B.2) is called a **difference equation** and $\mathbf{x}[\mathbf{1}]$ is called its **initial value**. If we are given that $x[1] = 4$, we'd then compute that

$$\begin{aligned} x[1] &= \frac{1}{2}x[0] + 1 = 3 \\ x[2] &= \frac{1}{2}x[1] + 1 = \frac{7}{2} \\ x[3] &= \frac{1}{2}x[2] + 1 = \frac{11}{4} \\ x[4] &= \frac{1}{2}x[3] + 1 = \frac{19}{8} \\ \vdots &= \qquad \vdots \\ x[k+1] &= \frac{1}{2}x[k] + 1 \end{aligned}$$

In Fig. B.1, we plot the function x for $k = 1 : 20$. We observe that it seems to converge to 2.0, which is intriguing. In Fig. B.2, we have "cheated" and made time look quasi-continuous by "connecting the dots". The Julia code for generating x and the plots is given in Sec. B.6.1.

It would seem that if we could put the "dots closer together", the effect in Fig. B.2 would be even better. Let's try!

B.3 Digital Time may be Discrete, but We Can Make the Time Increment δt Quite Small

In our previous model and plot, we implicitly took $\delta t = 1$ "unit" of time, where "unit" could have been seconds, milliseconds, months, or fortnights. We just plotted the index k and had no idea how it was directly related to a physical notion of "time". This

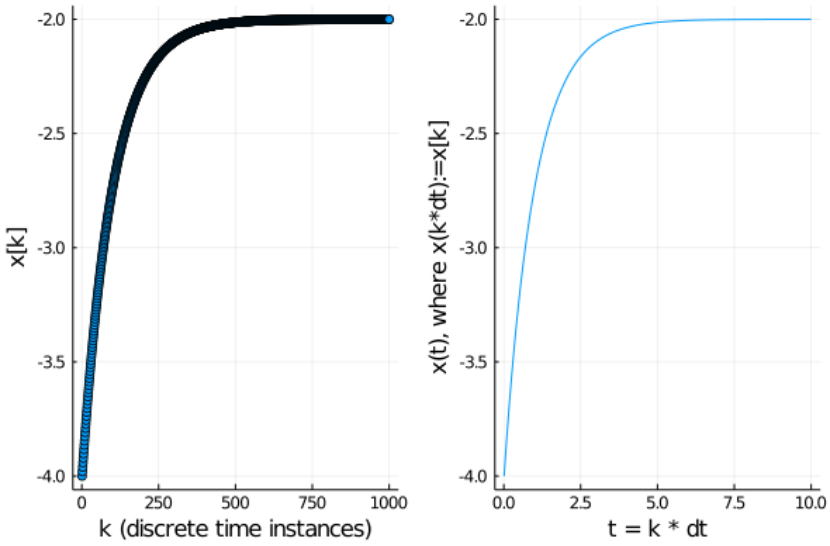


Figure B.3: Plot of the function $x : [0, 10] \to \mathbb{R}$ that solves (B.3) when its initial condition is $x[1] = 4$. The discrete interval of time $\delta t = 0.01$ is small enough that it looks continuous, and in fact, if one connects the dots, our function versus time looks like a continuous curve.

time we'll define $\delta t = 0.01$ seconds, that is, 10 milliseconds, and we'll think of $x[k]$ as representing $x(k\delta t)$. To be super explicit,

$$\begin{aligned}
x[1] &:= x(t)|_{t=\delta t} = x(0.01) \\
x[2] &:= x(t)|_{t=2\delta t} = x(0.02) \\
x[3] &:= x(t)|_{t=3\delta t} = x(0.03) \\
&\vdots \\
x[99] &:= x(t)|_{t=99\delta t} = x(0.99) \\
&\vdots \\
x[301] &:= x(t)|_{t=301\delta t} = x(3.01) \\
&\text{etc.}
\end{aligned}$$

We introduce δt into the model in a particular manner that will become clear in Sec. B.4,

$$x[k+1] = x[k] - \delta t \cdot (x[k] + 2). \tag{B.3}$$

Fig B.3 shows two plots side by side. One shows the function x plotted against the index k, while the second plot shows x plotted against time, for $t = k\delta t$. The Julia code for computing the function and making the plots is given in Sec. B.6.2.

B.4 Equations with Derivatives in them are called Differential Equations

All of us hear about "$F = ma$" in High School, one of Newton's laws for how a body of mass m accelerates under the action of applied forces F. We might also learn that "acceleration", a, is the rate of change of "velocity", v, with respect to time, t.

The notation $\frac{\mathrm{d}}{\mathrm{dt}}$

In Calculus, the rate of change of one quantity, say v, with respect to another quantity, say t, is denoted

$$\frac{dv(t)}{dt}.$$

Hence, in the language of Calculus, acceleration is related to velocity by

$$a(t) := \frac{dv(t)}{dt},$$

which, once again, is just shorthand for "acceleration is the rate of change of velocity with respect to time". This course does not assume knowledge of Calculus, so please don't sweat the introduction of this notation. For us, it is just a shorthand way of saying "rate of change with respect to time".

Using this shorthand notation, we can express Newton's Law as

$$m\frac{dv(t)}{dt} = F(t). \tag{B.4}$$

If we suppose that the body is falling in air, we might model the total force $F(t)$ acting on the body as being due to gravity and air friction,

$$F(t) = -mg - k_d v(t).$$

Doing so leads us to the equation

$$\boxed{m\frac{dv(t)}{dt} = -k_d v(t) - g}. \tag{B.5}$$

This equation says that the rate of change of the velocity as a function of time is given by a sum of two terms, one of which corresponds to gravity and the other to air resistance. Equation (B.5) is called a **differential equation**, that is, an equation that depends on "derivatives."

In (B.5), let's now replace the shorthand Calculus symbol for rate of change, $\frac{dv(t)}{dt}$, with the same kind of numerical approximation we used in our studies of root finding and optimization, namely

$$\frac{dv(t)}{dt} \approx \frac{v(t+\delta t) - v(t)}{\delta t}. \tag{B.6}$$

Substituting (B.6) into (B.5) and assuming the approximation is "good enough" (that we can change $\approx$ into $=$) give

$$\begin{aligned}
\frac{dv(t)}{dt} &= \frac{1}{m}\left(-k_d v(t) - g\right) \\
&\Updownarrow \\
\frac{v(t+\delta t) - v(t)}{\delta t} &= \frac{1}{m}\left(-k_d v(t) - g\right) \\
&\Updownarrow \\
v(t+\delta t) - v(t) &= \delta t\frac{1}{m}\left(-k_d v(t) - g\right) \\
&\Updownarrow \\
v(t+\delta t) &= v(t) + \delta t\frac{1}{m}\left(-k_d v(t) - g\right).
\end{aligned} \tag{B.7}$$

If we then define $t = k \cdot \delta t$ and $v[k] := v(k \cdot \delta t)$, the equation looks just like our *difference equations* in Sec. B.3. Indeed, we have $v(t + \delta t) = v(k \cdot \delta t + \delta t) = v((k + 1) \cdot \delta t) = v[k + 1]$, and (B.7) becomes

$$v[k+1] = v[k] - \delta t \frac{k_d}{m} v[k] - \delta t \frac{g}{m}, \tag{B.8}$$

which is very much like (B.3) and hence, we now know one way to (approximately) solve the differential equation (B.5): we set an initial condition and iterate based on (B.8).

In fact, if the mass of our falling body is $m = \frac{2}{g}$ and its drag is $k_d = m$, then (B.8) is exactly the same as (B.3). Moreover, we can physically interpret the solution of the differential equation (B.5), which is plotted in Fig. B.3, as a package is tossed out of plane. Our model tracks its evolution from the moment the parachute deploys, greatly increasing its drag, thereby slowing its velocity from an initial value of -4 to -2 units of distance per second (we never specified the units).

B.5 Discretization of ODEs of Higher Dimension

ODE

ODE is short for Ordinary Differential Equation. The word "ordinary" is used because there are other kinds of differential equations, which apparently, are less "ordinary". Everyone in the know uses the terminology ODE, hence you will too!

ODEs can be vector valued too. A linear ODE may look like this,

$$\frac{dx(t)}{dt} = Ax(t) + b.$$

Just as before, we select $\delta t > 0$ and define $x[k] := x(k \cdot \delta t)$, and re-write the differential equation as a difference equation

$$\begin{aligned}
\frac{dx(t)}{dt} &\approx \frac{x(t+\delta t) - x(t)}{\delta t} \\
&\Downarrow \\
\frac{x(t+\delta t) - x(t)}{\delta t} &\approx Ax(t) + b \\
&\Downarrow \\
x(t+\delta t) &= x(t) + \delta t \left(Ax(t) + b\right) \\
&\Downarrow \\
x[k+1] &= x[k] + \delta t A x[k] + \delta t b
\end{aligned}$$

A two-dimensional example is

$$\underbrace{\begin{bmatrix} x_1[k+1] \\ x_2[k+1] \end{bmatrix}}_{x[k+1]} = \underbrace{\begin{bmatrix} x_1[k] \\ x_2[k] \end{bmatrix}}_{x[k]} + \delta t \cdot \underbrace{\begin{bmatrix} 0.0 & 1.0 \\ -2.0 & -1.0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1[k] \\ x_2[k] \end{bmatrix}}_{x[k]} + \delta t \cdot \underbrace{\begin{bmatrix} 0.0 \\ 1.0 \end{bmatrix}}_{b}. \tag{B.9}$$

This time, our $x[1]$ is a 2×1 vector. We'll arbitrarily specify it as $x[1] = [2 \;\; 0]^\top$. The plots of x as a function of index k and of time $t = k \cdot \delta t$ are given in Fig. B.5. The associated Julia code is given in Sec. B.6.4.

Finally, not only can the function x of time can be vector valued, it can have nonlinear terms. Here is a common example from physics: a pendulum of mass m and length ℓ swinging from a frictionless pivot satisfies the equation

$$\begin{aligned}
\frac{dx_1(t)}{dt} &= x_2(t) \\
\frac{dx_2(t)}{dt} &= -\frac{g}{\ell} \sin(x_1(t)),
\end{aligned} \tag{B.10}$$

where x_1 is the angle of the pendulum and x_2 is its angular velocity. We'll rewrite the model in vector form by defining

$$x := \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \tag{B.11}$$

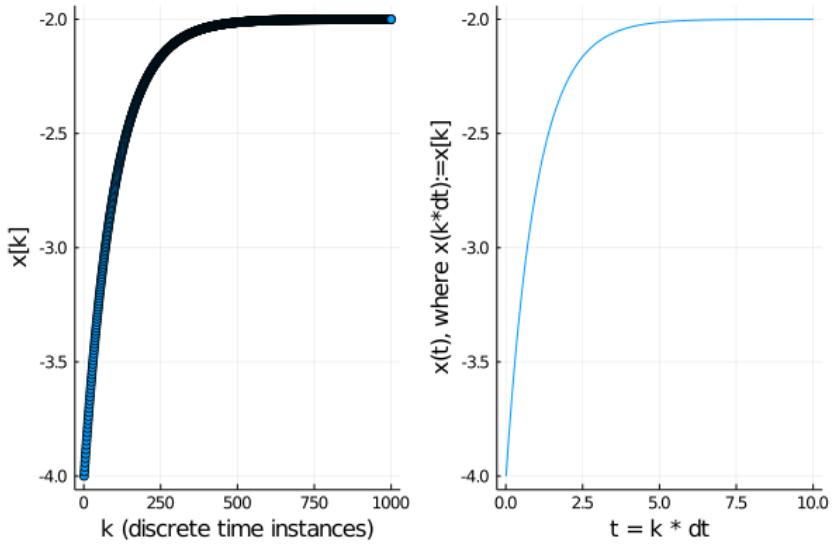


Figure B.4: Plot of the function $x : [0, 10] \to \mathbb{R}$ that solves (B.3) when its initial condition is $x[1] = 4$. The discrete interval of time $\delta t = 0.01$ is small enough that it looks continuous, and in fact, if one connects the dots, our function versus time looks like a continuous curve.

so that

$$\frac{dx(t)}{dt} = \begin{bmatrix} \frac{dx_1(t)}{dt} \\ \frac{dx_2(t)}{dt} \end{bmatrix} := \underbrace{\begin{bmatrix} x_2(t) \\ -\frac{g}{\ell}\sin(x_1(t)) \end{bmatrix}}_{f(x(t))} =: f(x(t)) \tag{B.12}$$

To find a solution, we discretize the model with $\delta t > 0$. To show the flexibility we have in approximating the derivative, we'll use a symmetric difference approximation this time, namely

$$\begin{aligned} \frac{dx(t)}{dt} &\approx \frac{x(t+\delta t) - x(t-\delta t)}{2\delta t} \quad \text{and} \quad \frac{dx(t)}{dt} = f(x(t)) \\ &\Updownarrow \\ x(t+\delta t) &= x(t-\delta t) + 2\delta t f(x(t)) \quad \text{and} \quad t = k \cdot \delta t \\ &\Updownarrow \\ x[k+1] &= x[k-1] + \delta t f(x[k]) \end{aligned} \tag{B.13}$$

We use the last line of (B.13) to iterate and compute a solution to the pendulum ODE (B.10). The solution is plotted in Fig. B.6. The associated Julia code is given in Sec. B.6.5.

Our transformation of ODEs into difference equations are examples of numerical integration methods. When we use the approximation

$$\frac{dx(t)}{dt} \approx \frac{x(t+\delta t) - x(t)}{\delta t}$$

we end up with what is called a first-order forward Euler integration method. In general, it's a pretty bad way to solve ODEs, but for a first introduction, it is great! We can do a lot with it.

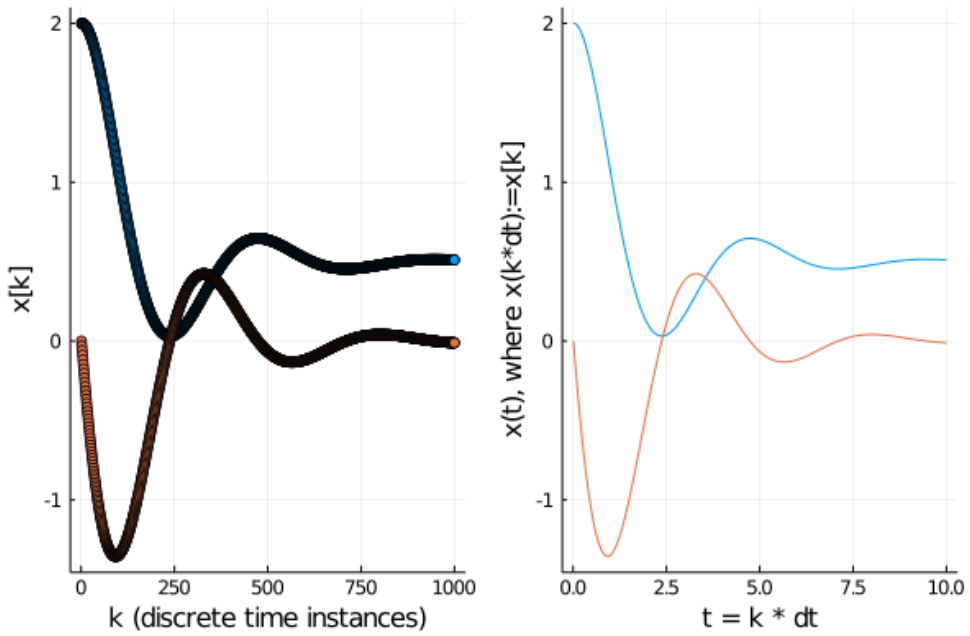


Figure B.5: Plot of the vector valued function $x : [0, 10] \to \mathbb{R}^2$ solving the ODE $\frac{dx(t)}{dt} = Ax + b$, with the solution approximated in (B.9). The discrete interval of time is small enough that it looks continuous!

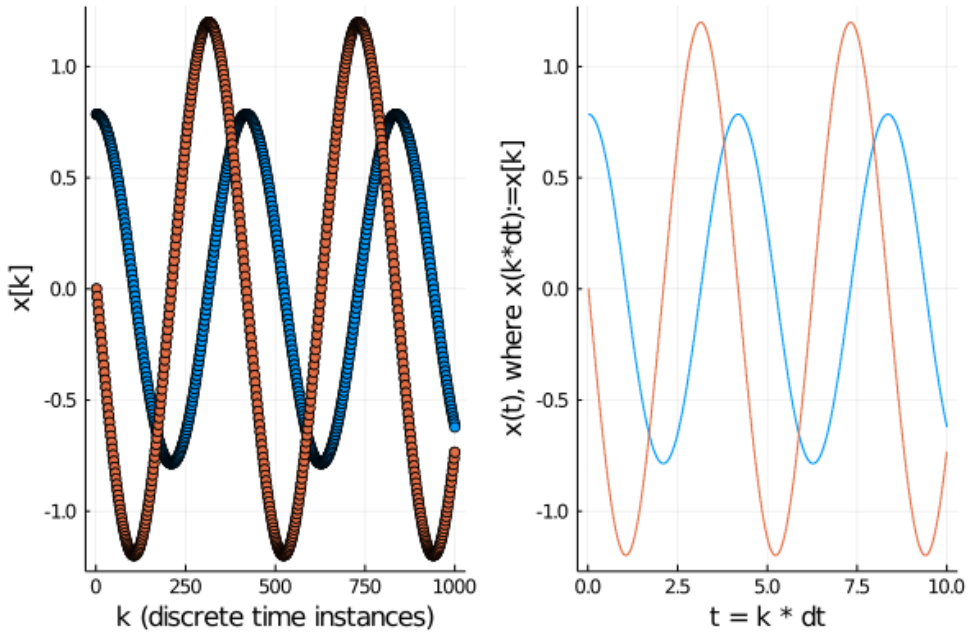


Figure B.6: Plot of a vector valued function $x : [0, 10] \to \mathbb{R}^2$ that oscillates back and forth because it corresponds to the solution the pendulum ODE given in (B.10). The blue lines are the angle of the pendulum and the brown(ish) lines are its angular velocity.

B.6 Julia Code for Generating the Figures

We provide code in Julia 1.41 for the figures appearing in this Appendix.

B.6.1 For Figures B.1 and B.2

```
# Model
dt=0.01
T=10
N=floor(Int,T/dt);
K=1:N
time = K*dt
x=zeros(N,1)
x[1]=-4
for k=1:(N-1)
    x[k+1]=(1-dt)*x[k]-2*dt
end
# Plotting
plot1=scatter(K,x,xlabel="k (discrete time instances)", ylabel="x[k]", leg=false)
plot2=plot(time,x,xlabel="t = k * dt", ylabel="x(t), where x(k*dt):=x[k]", leg=false)
plot(plot1, plot2, layout = (1, 2), legend = false)
#Turn the plot into an image that one can copy
plot!(fmt = :png)
```

B.6.2 Code for Figure B.3

```
using Plots
gr()
# Model
N=20
time=1:N
x=zeros(N,1)
x[1]=4
for k=1:(N-1)
    x[k+1]=0.5*x[k]+1
end
# Plotting
scatter(time,x)
#plot!(xlabel="k (discrete time instances)", ylabel="x[k]",
#title="Plot of as a function of time: Discrete-time Model", leg=false)
plot!(xlabel="k (discrete time instances)", ylabel="x[k]", leg=false)
#Turn the plot into an image that one can copy
plot!(fmt = :png)
#
# Second plot
#
plot(time,x)
#plot!(xlabel="k (discrete time instances)", ylabel="x[k]",
#title="Plot of as a function of time: Discrete-time Model", leg=false)
plot!(xlabel="k (discrete time instances)", ylabel="x[k]", leg=false)
scatter!(time,x)
#Turn the plot into an image that one can copy
plot!(fmt = :png)
```

B.6.3 Code for Figure B.4

```
# Model
dt=0.01
T=10
N=floor(Int,T/dt);
K=1:N
time = K*dt
x=zeros(N,1)
x[1]=4
for k=1:(N-1)
    x[k+1]=(1-dt)*x[k]+2*dt
end
# Plotting
plot1=scatter(K,x,xlabel="k (discrete time instances)", ylabel="x[k]", leg=false)
plot2=plot(time,x,xlabel="t = k * dt", ylabel="x(t), where x(k*dt):=x[k]", leg=false)
plot(plot1, plot2, layout = (1, 2), legend = false)
#Turn the plot into an image that one can copy
plot!(fmt = :png)
```

B.6.4 Code for Figure B.5

```
# Model
dt=0.01
T=10
N=floor(Int,T/dt);
K=1:N
time = K*dt
A=[0 1; -2 -1]
b=[0;1]
x=zeros(N,2)
x[1, :]=[2 0]
for k=1:(N-1)
    x[k+1,:]=x[k,:]+ dt*A*x[k,:] + dt*b
end
# Plotting
plot1=scatter(K,x,xlabel="k (discrete time instances)", ylabel="x[k]", leg=false)
plot2=plot(time,x,xlabel="t = k * dt", ylabel="x(t), where x(k*dt):=x[k]", leg=false)
plot(plot1, plot2, layout = (1, 2), legend = false)
#Turn the plot into an image that one can copy
plot!(fmt = :png)
```

B.6.5 Code for Figure B.6

```
# Model
dt=.01
T=50
N=floor(Int,T/dt);
K=1:N
time = K*dt
g=9.81 #m/s^2
l=4 # m
x=zeros(N,2)
```

```
#Pendulum equations
f(x1,x2)=[x2;-(g/l)*sin(x1)]
#Initial conditions
x[1, :]=[pi/4 0]
x[2, :]=x[1,:] + dt*f(x[1,1],x[1,2])
#Euler integration based on Symmetric Difference for dx/dt
for k=2:(N-1)
x[k+1,:]=x[k-1,:] +2*dt*f(x[k,1],x[k,2])
end
# Plotting
plot1=scatter(K,x,xlabel="k (discrete time instances)", ylabel="x[k]", leg=false)
plot2=plot(time,x,xlabel="t = k * dt", ylabel="x(t), where x(k*dt):=x[k]", leg=false)
plot(plot1, plot2, layout = (1, 2), legend = false)
#Turn the plot into an image that one can copy
plot!(fmt = :png)
```

From here on, the book is a work in progress.
Please pardon our mess during construction!

Appendix C

Camera and LiDAR Models for Students of Robotics

Learning Objectives

- [**JWG: ???**].

Outcomes

- [**JWG: ???**]

[**JWG: Jessy to write an Intro. Bruce to add background on LiDAR sensors**]

C.1 Pinhole Camera Model

Camera calibration is the process of finding the quantities internal to the camera that affect the imaging process. Today's cheap camera lenses or camera production errors may introduce a lot of distortion to images. Precise camera calibration is important when we need to deal with 3D interpretation of images, reconstruction of world models, and robot interaction with the real world. We will apply what we have learnt so far to discover the charm of camera calibration and its usage in our project.

C.2 Preliminaries: Geometrical Transformations

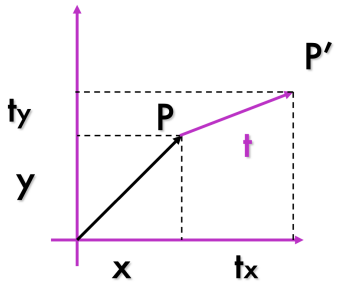
C.2.1 Homogeneous Coordinates

Homogeneous coordinates are widely used in computer vision and graphics. They are a nice extension of standard 3D vectors and allow us to simplify and compute various vector operations, such as translation, rotation, scaling and perspective projection. The sequence of such operations can be multiplied out into a single matrix with simple and efficient processing.Besides, homogeneous coordinates also allow to represent infinity. The Euclidean coordinate system denotes the location of an object by a triplet of numbers. However, we can't treat infinity as a regular number. Homogeneous coordinates allows by denoting infinity by $[x, y, z, 0] = \frac{[x,y,z]}{0}$ in 3D.

How to transfer between Cartesian Coordinates and Homogeneous Coordinates?

- Cartesian coordinates $\rightarrow$ Homogeneous coordinates : Multiply the coordinates by a non-zero scalar and add an extra coordinate equal to that scalar. For example, $[x, y] \rightarrow [x \cdot z, y \cdot z, z], z \neq 0$. We usually multiply the coordinates by 1.
- Homogeneous coordinates $\rightarrow$ Cartesian coordinates: Divide the Cartesian coordinates by the last coordinate and eliminate it. For example, $[x, y, z], z \neq 0 \rightarrow [x/z, y/z]$.

C.2.2 2D Translation



$$\mathbf{P} = (x, y)$$
$$\mathbf{t} = (t_x, t_y)$$

Figure C.1: Example of 2D Translation

If point **P'**(x', y') is obtained by translating **P** by **t**(t_x, t_y), then the relation between **P'** and **P** could be written as:

$$\mathbf{P'} = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \mathbf{I} \cdot \mathbf{P} + \mathbf{t}$$

If we use homogeneous coordinates, $P = (x, y, 1)$, and $t = (t_x, t_y, 1)$. The the relation between P' and P becomes:

$$\mathbf{P'} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \cdot \mathbf{P} = \mathbf{T} \cdot \mathbf{P}$$

C.2.3 2D Scaling

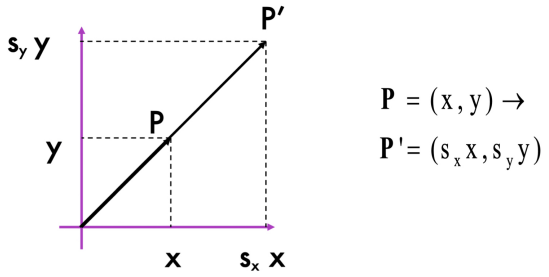


Figure C.2: Example of 2D Scaling

If point **P'**(x', y') is obtained by scaling **P** by **s**(s_x, s_y), then the relation between **P'** and **P** could be written as:

$$\textbf{P'} = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \textbf{S'} \cdot \textbf{P}$$

Similar before, the relation between **P'** and **P** could be expressed in homogeneous coordinates:

$$\textbf{P'} = \begin{bmatrix} s_x x \\ s_y y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \textbf{S'} & 0 \\ 0 & 1 \end{bmatrix} \textbf{P} = \textbf{S} \cdot \textbf{P}$$

C.2.4 2D Rotation

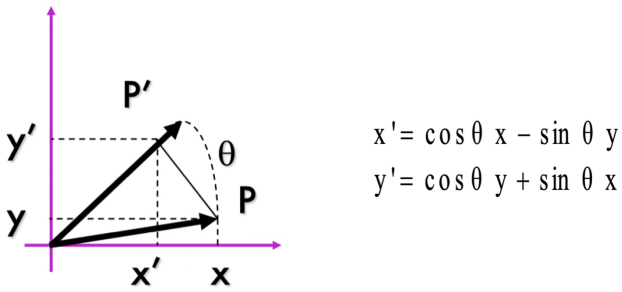


Figure C.3: Example of 2D Rotation

If point **P'**(x', y') is obtained by rotating **P** counterclockwise by θ degree, then the relation between **P'** and **P** could be written as:

$$\textbf{P'} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \textbf{R'} \cdot \textbf{P}$$

The relation between **P'** and **P** could be expressed in homogeneous coordinates:

$$\textbf{P'} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \textbf{R'} & 0 \\ 0 & 1 \end{bmatrix} \textbf{P} = \textbf{R} \cdot \textbf{P}$$

R is an orthogonal matrix and it satisfies the following properties:

- $R \cdot R^\top = R^\top \cdot R = I$
- det(R) = 1

C.2.5 Other 2D Geometrical Transformations in Homogeneous Coordinates

- Shear in x: $\begin{bmatrix} 1 & k_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- Shear in y: $\begin{bmatrix} 1 & 0 & 0 \\ k_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- Reflect about y: $\begin{bmatrix} 1 & 0 & 0 \\ k_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- General Affine Transformation: $\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$

If there are more than one type of geometric transformation, one can multiply the corresponding transformation matrix to get the final transformation matrix.
For example, if we want to find the transformation matrix after scaling P by s firstly, then rotating counterclockwise by θ degree, and finally translate by t. The position of P' could be compute by:

$$
\begin{aligned}
P' = \mathbf{T} \cdot \mathbf{R} \cdot \mathbf{S} \cdot P &= \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} R' & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} S & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} R'S & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\end{aligned}
$$

C.3 Pinhole Model

Last section we mainly discussed the affine transformation, which preserves the parallel lines, ratio of areas, ratio of lengths on colinear lines and a few others. However, the affine transformation is not a valid assumption in general for images, nor is it valid around the image edges. The mapping from 3D object to 2D image is a perspective projection as shown in the following image:

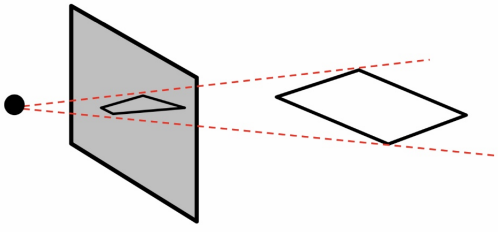


Figure C.4: Perspective schematic diagram

Figure C.5: Real-life perspective example: parallel rails

The **pinhole camera model** describes the mathematical relationship between the coordinates of a point in 3D space and its projection onto the image plane of an ideal pinhole camera. The following diagram illustrates the pinhole model and defines some intrinsic parameters of pinhole camera:

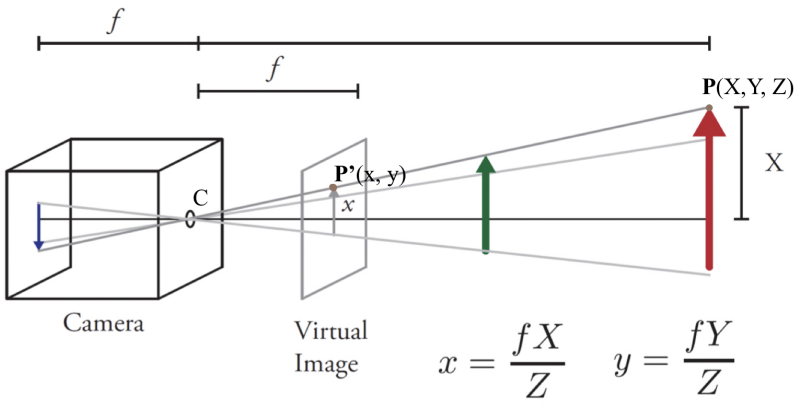


Figure C.6: Pinhole camera model

There are some key assumptions of the pinhole model:

- Aperture, i.e. the pinhole, is described as a point (point C in the image)
- No lenses are used to focus light, therefore no geometric distortions (radial distortion)
- Does not take into account that most practical cameras have only discrete image coordinates
- This model can only be used as a first order approximation of the mapping from 3D scene to 2D image. This is a perspective projection
- Ideal model that we use has identity matrix for the perspective portion of the projection matrix (**projection matrix** = [**Perspective**|0] = $[\mathbf{I}_{3\times3}|0]_{3\times4}$ which assumes no perspective since the surface is close to the camera)

If **P** (X, Y, Z) is a 3D scene point, and it is projected into the 2D image by pinhole. **P'** $(x, y$ is the projective point of **P** on the 2D virtual image plane. We could find the following equations derived using similar triangles.
Remark: f is the focal length of the camera.

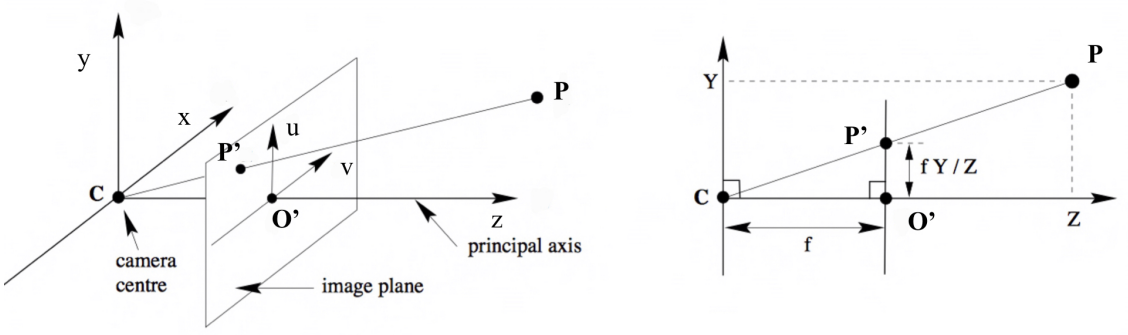


Figure C.7: Pinhole model

$$\frac{f}{Z} = \frac{y}{Y} = \frac{x}{X} \implies \begin{cases} x = \frac{fX}{Z} \\ y = \frac{fY}{Z} \end{cases} \tag{C.1}$$

The image plane is usually read as $n \times m$ pixel matrix in computer. Points in the image is then represented by pixel coordinates.

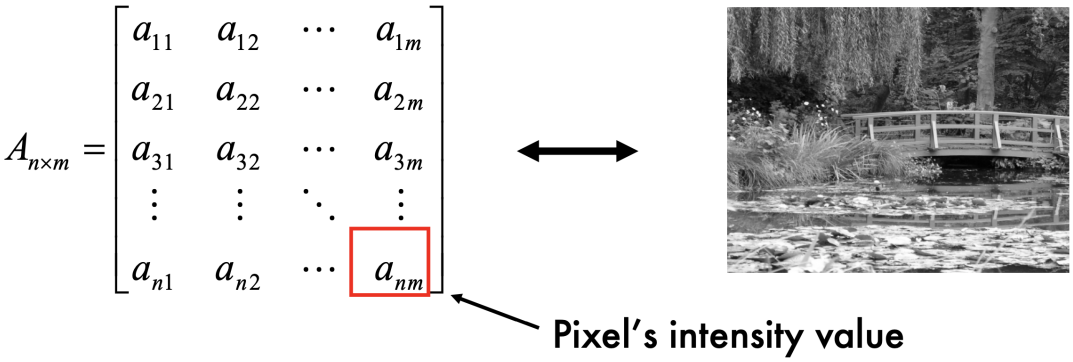


Figure C.8: Relation of image plane and pixel matrix

The following picture is an example of image plane, which indicates the relation between camera coordinate origin and image coordinate origin

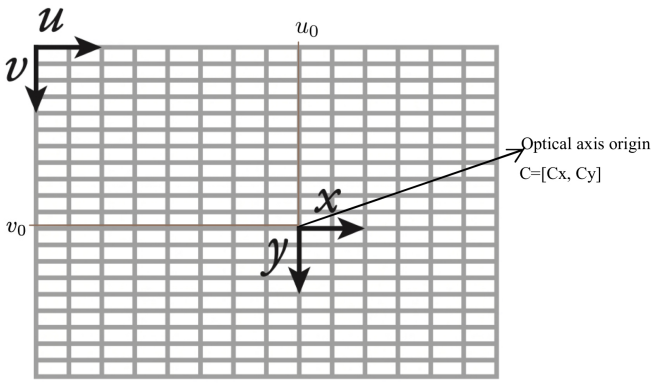


Figure C.9: Pixel coordinates plane

By expanding equation C.1, the relation between positions on the image scene (u, v) and positions in the real world (X, Y, Z) could be written as:

$$u = fs_x x^c + u_0 = \alpha \frac{X}{Z} + u_0$$
$$v = fs_y y^c + v_0 = \beta \frac{Y}{Z} + v_0$$

We assume the world frame and camera frame is aligned. s_x, s_y are the scale in x and y direction to transfer the units from metric to pixel. x^c, y^c are projective points of 3D objects in camera coordinates.

However, there are other considerations:

- Optical axis is not centered on sensor
- Pixel coordinate system orgin is in the corner of the sensor
- Pixels aren't square
- Pixel size is unknown

Therefore, we need to find a transformation matrix to consider all the possible situations to transform the camera frame to the image plane. Review 2D geometric transformations in previous section and recall the general affine transformation $\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$. We will explore more about it in the next section.

C.4 Geometric and Mathematical Relations

We would like to represent image plane coordinates in terms of real world coordinates. This is exactly the same problem as finding the conversion of 3D world coordinates to image coordinates and then inverting this transformation. In this section, we will examine the three components of this transformation.

C.4.1 Intrinsic Matrix K

Intrinsic matrix is the affine transformation matrix we mentioned before to represent the linear properties of the pinhole camera model (focal length to visual image, scale, shear, translation, and so on). The intrinsic matrix can be used to transform 2D camera coordinates (x^c, y^c) into image plane coordinates (u, v):

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} x^c \\ y^c \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & ks_y & u_0 \\ 0 & fs_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^c \\ y^c \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^c \\ y^c \\ 1 \end{bmatrix}$$

Where s_x is the scale in x, s_y is the scale in y, f is the focal length, k is the ration of shear in the y direction to that in x, and (u_0, v_0) is the distance from the image center to the image plane coordinate system origin.

C.4.2 Projection Matrix

This matrix transforms the coordinates from 3D to 2D. The perspective portion of this matrix (left) is equal to the identity because we make the assumption that all captured positions are close.

$$\begin{bmatrix} x^c \\ y^c \\ 1 \end{bmatrix} = \frac{1}{Z_c} [P_{\text{perspective}} \quad 0] \begin{bmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{bmatrix} = \frac{1}{Z_c} [I_{3\times 3} \quad 0]_{3\times 4} \begin{bmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{bmatrix}$$

The positions are now shown in lower case and lie now on a 2D plane.

C.4.3 Extrinsic Matrix

This matrix belongs in SE(3) and it represents the transformation of position coordinates from the real world frame to the camera coordinate frame.

$$\begin{bmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{bmatrix} = [R \quad t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

C.4.4 Full Projection Matrix Workspace to Camera

Grouping everything together we get a final full projection matrix, P, such that

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\begin{aligned} P &= K[I \quad 0][R \quad t] \\ &= \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} [I \quad 0][R \quad t] \end{aligned}$$

Or with the general affine transformation as the intrinsic matrix

$$P = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} [I \quad 0][R \quad t]$$

C.5 Intrinsic Matrix Calibration

Solve the simpler problem of only finding the intrinsic matrix if the extrinsic is known. If you are already given the 2D camera coordinates (post extrinsic and projective transformation, then you can select at least 3 known points in the real world and image plane (which do not all lie within the same plane). Remember the general form

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^c \\ y^c \\ 1 \end{bmatrix}$$

For example, let's say that we know two different points then

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1^c \\ y_1^c \\ 1 \end{bmatrix}, \quad \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2^c \\ y_2^c \\ 1 \end{bmatrix}$$

The system of equations become

$$\begin{aligned} ax_1 + by_1 + c &= u_1 \\ dx_1 + ey_1 + f &= v_1 \\ ax_2 + by_2 + c &= u_2 \\ dx_2 + ey_2 + f &= v_2 \end{aligned}$$

Rearranging into the form $Ax = b$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \vdots \end{bmatrix}$$

A quick and less rigorous derivation of the solution: We want to approximate b with the subspace of A. This means through the projection theorem that $b - Ax$ is perpendicular to Ax

$$\begin{aligned} b - Ax &\perp Ax \\ (b - Ax) \cdot (Ax) &= 0 \\ (Ax) \cdot (Ax) &= (Ax) \cdot b \\ x^T A^T Ax &= x^T A^T b \\ x &= (A^T A)^{-1} A^T b \end{aligned}$$

Then plug the corresponding values into the complete intrinsic matrix.

C.6 Nonlinear Phenomena in Calibration

In real life, because of the imperfect lens or inaccurate placement of the lens, the image would be distorted. We can use nonlinear root finding methods like Newton-Raphson to account for nonlinear phenomena. $(x_{distorted} = x_{linear} + g(x)$

- Radial Distortion
  This phenomenon occurs when light rays bend more near the edges of a lens than they do at this optical center. Smaller lens can lead to greater distortion.

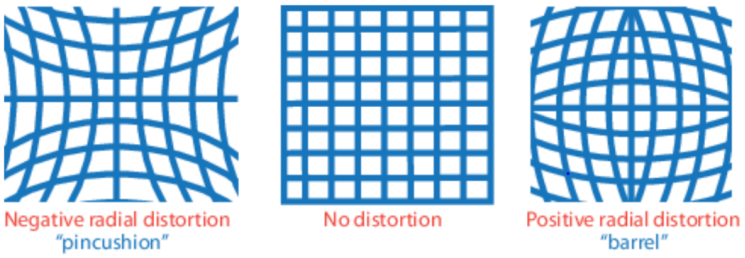


Figure C.10: Types of radial distortion

- Tangential Distortion This phenomenon occurs when the lens and the image plane are not parallel

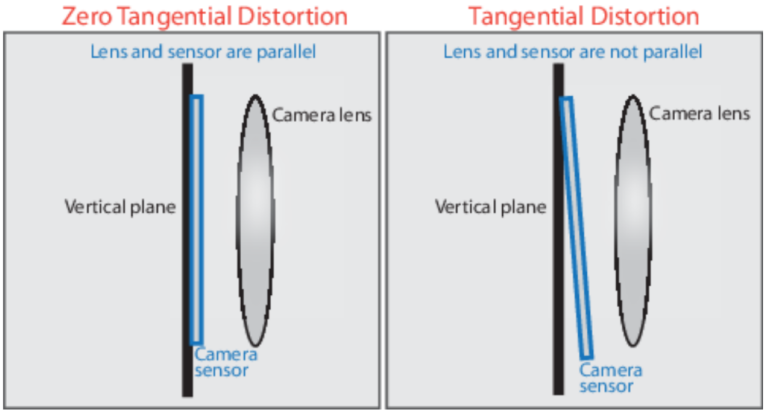


Figure C.11: Tangential distortion

C.7 Projection Map from LiDAR to Camera

C.8 Projection Map

The projection map is a mapping between a 3D and a 2D world, and is used to project a LiDAR point cloud (3D) into an image (2D). Let X be the LiDAR points and Y be the projected points on the image-plane of a camera. The standard relations are [1]

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsic parameters}} \begin{bmatrix} \mathbb{1}_{3\times3} \\ \mathbf{0}_{1\times3} \end{bmatrix}^{\mathsf{T}} \underbrace{\begin{bmatrix} R_L^C & t_L^C \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}}_{\text{extrinsic parameters}} \underbrace{\begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}}_{\text{LiDAR points}} \tag{C.2}$$

$$= K \begin{bmatrix} \mathbb{1}_{3\times3} \\ \mathbf{0}_{1\times3} \end{bmatrix}^{\mathsf{T}} H_L^C \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{C.3}$$

$$Y_i = \begin{bmatrix} u & v & 1 \end{bmatrix}^{\mathsf{T}} = \begin{bmatrix} \frac{u'}{w'} & \frac{v'}{w'} & 1 \end{bmatrix}^{\mathsf{T}}, \tag{C.4}$$

where (C.2) includes the camera's intrinsic parameters (K) and the extrinsic parameters (R_L^C, T_L^C).

[1] More information about the projection map, we refer the readers to " Multiple view geometry in computer vision second edition," by Hartley Richard and Zisserman Andrew, and "Computer vision: a modern approach," by Forsyth David A and Ponce Jean.

For later use, we combine (C.2) and (C.4) to define

$$\Pi\left(X_i; R, t\right) := Y_i, \tag{C.5}$$

the projection map from LiDAR coordinates to image coordinates. Note that it is a function of both the extrinsic variables and the LiDAR vertices.