Introductory Numerical Analysis

Thomas Trogdon
University of Washington
trogdon@uw.edu

Contents

Preface

Part I of these notes are for AMATH 352 taught from [OS18]. Most of the theoretical flow of the subject of linear algebra directly shadows [OS18]. Although, the ordering is changed to focus on implementing algorithms in MATLAB immediately.

Throughout this text the results that are deemed the most important in the sense that they are likely useful in solving problems and for following the main development of the text are highlighted by being boxed.

Chapter 0

Matlab basics

MATLAB is a scripting language without types. That means a few things.

1. There is no need to declare variables before you assign values to them. This, for example, is something that is necessary in JAVA.
2. A variable initially defined to be a scalar than then be assigned to be an array and then assigned to be a function. This can be convenient but also makes it tougher to catch bugs.
3. Unless a specific environment is loaded, MATLAB will do all of its calculations in floating point arithmetic. In a language like PYTHON an expression like `1/4` will return `0` because `0` is the closest integer and `1` and `4` are integers. MATLAB will return `0.25`.

MATLAB code is stored in `.m`-files, or m-files for short. These are also referred to as MATLAB scripts. It is good practice to include the following at the top of every script that your write:

```
1 clear all;  close all;
```

This will help ensure that

1. The ability of your code to run is not affected by other previously set variables.
2. If you close MATLAB, your script will behave the same when the next time you open it.

MATLAB will also print the output of any line if you do not append a semicolon `;` at the end of the line. For example, executing the following produces output to the Command Window

```
1 x = 10
```

```
x =

    10
```

while

```
x = 10;
```

produces no output. In order to debug your code you will want to have nearly every line end with a semicolon. And then if the script does not run as expected you can remove one or two semicolons at a time to monitor the output.

The following code uses a *for loop* to add up all the positive integers that are less than or equal to n

```
n = 10;
SUM = 0; % using capital letters because sum() is a built-in function
for i = 1:n
    SUM = SUM + i;
end
SUM
n*(n+1)/2  % known answer to check
```

```
SUM =

    55


ans =

    55
```

Another type of loop is the *while loop*. Here is an example of performing the same sum as above using a while loop.

```
n = 10;
SUM = 0;
i = 0;
while i < n
    i = i + 1;
    SUM = SUM + i;
end
SUM
```

```
SUM =

    55
```

0.1 ▪ Using Matlab with some examples from calculus

Because Part I of this text concerns numerical linear algebra, there are few opportunities introduce the reader to the plotting functionality in MATLAB. So, we take some time to review some theorems from calculus and illustrate them using MATLAB.

0.1.1 ▪ Demonstrations from differential calculus

Definition 0.1. *A function f defined on a set X has limit L at x_0*

$$\lim_{x \to x_0} f(x) = L$$

if, given any real number $\epsilon > 0$*, there exists a real number* $\delta > 0$ *such that*

$$|f(x) - L| < \epsilon, \quad \textit{whenever} \quad x \in X \quad \textit{and} \quad 0 < |x - x_0| < \delta.$$

Definition 0.2. *Let* f *be a function defined on a set* X *of real numbers and* $x_0 \in X$*. Then* f *is continuous at* x_0 *if*

$$\lim_{x \to x_0} f(x) = f(x_0).$$

The function

$$f(x) = \begin{cases} \cos(\pi/x), & x \neq 0, \\ 1 & x = 0, \end{cases}$$

is not continuous at $x = 0$. To show this, let $x_n = 1/n$. If f is continuous then $\lim_{n\to\infty} f(1/n) = 1$

```
f = @(x) cos(pi./x);
ns = linspace(1,10,100);
plot(ns,f(1./ns))
xlabel('n'); ylabel('f(1/n)') %label axes
```

The plot that results is shown in Figure 1

Definition 0.3. *Let* $C^n[a,b] = \{f : [a,b] \to \mathbb{C} \mid f^{(n)}$ *exists and is continuous*$\}$.

We use the notation $C[a,b] := C^0[a,b]$.

Theorem 0.4 (Intermediate Value Theorem). *Let* $f \in C[a,b]$*. Assume* $f(a) \neq f(b)$*. For every real number* y*,* $f(a) \leq y \leq f(b)$*, there exists* $c \in [a,b]$ *such that* $f(c) = y$*.*

```
x = linspace(-3,3,100);
f = @(x) sin(x);
c = @(x) 0*x+.1;
plot(x,f(x),'k')
hold on
plot(x,c(x)) %every value between f(-3) and f(3) is attained at least once
```

The plot that results is shown in Figure 2

Definition 0.5. *Let* f *be a function defined on an open interval containing* x_0*. The function* f *is differentiable at* x_0 *if*

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}$$

exists. In which case, $f'(x_0)$ *is the derivative of* $f(x)$ *at* x_0*. If* f *has a derivative at each point in a set* X *then* f *is said to be differentiable on* X*.*

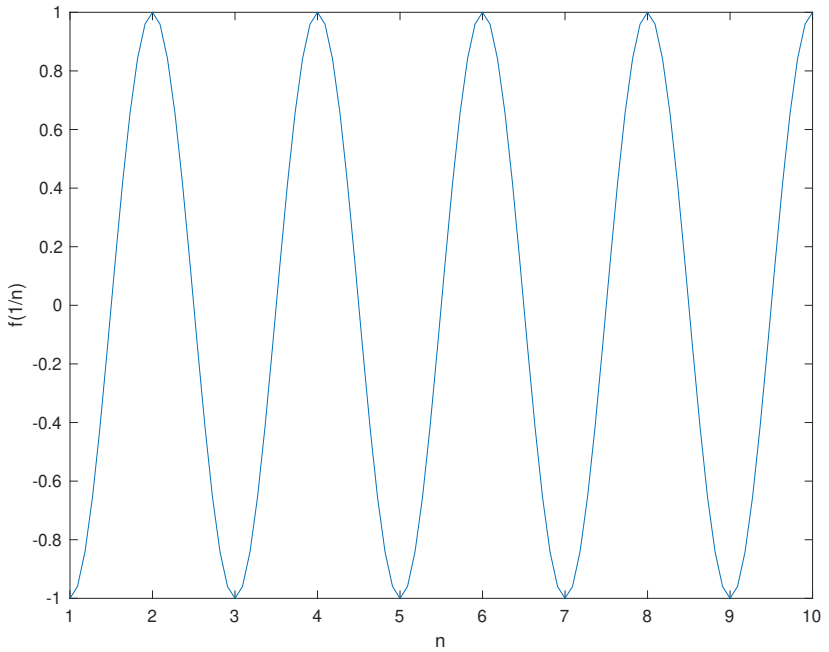


Figure 1: A plot of the function $f(1/n)$

```
format long %to see more digits
f = @(x) sin(x); df = @(x) cos(x);
x = .0001; x0 = 0;
(sin(x)-sin(x0))/(x-x0)-cos(x0)
```

```
ans =

   -1.666666582522680e-09
```

Here are some of the most important theorems from single-variable calculus:

Theorem 0.6. *If a function f is differentiable at x_0, it is continuous at x_0.*

Theorem 0.7 (Rolle's Theorem). *Suppose $f \in C[a,b]$ and f is differentiable on $[a,b]$. If $f(a) = f(b)$, the a number c in (a,b) exists with $f'(c) = 0$.*

Theorem 0.8 (Mean Value Theorem). *Suppose $f \in C[a,b]$ and f is differentiable on $[a,b]$. There exists a point $c \in (a,b)$ such that*

$$f'(c) = \frac{f(b) - f(a)}{b - a}.$$

Theorem 0.9 (Extreme Value Theorem). *If $f \in C[a,b]$, then $c_1, c_2 \in [a,b]$ exist with $f(c_1) \leq f(x) \leq f(c_2)$, for all $x \in [a,b]$. Furthermore, if f is differentiable on $[a,b]$ then c_1, c_2 are either the endpoints (a or b) or at a point where $f'(x) = 0$.*

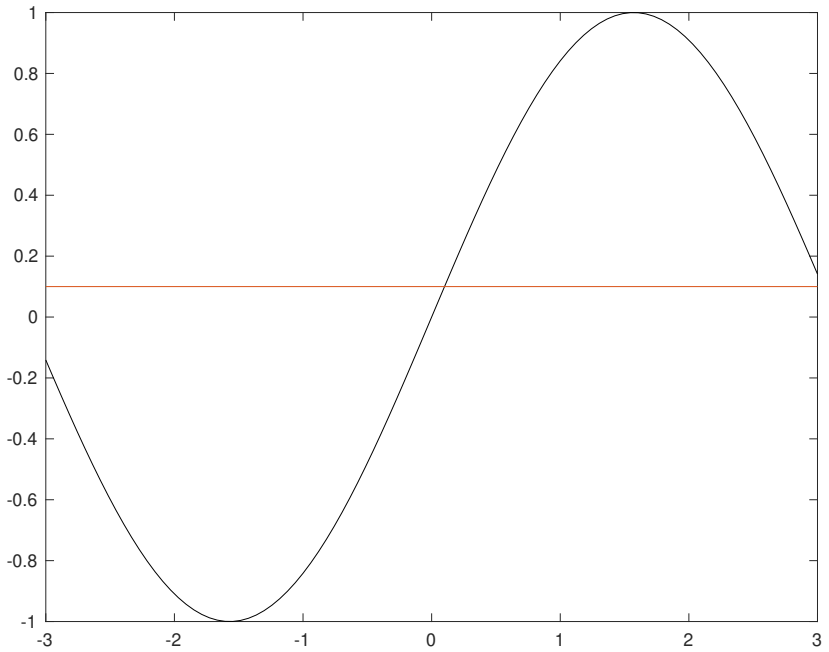


Figure 2: A demonstration of the intermediate value theorem for $f(x) = \sin(x)$.

This theorem states that both the maximum and minimum values of $f(x)$ on a closed interval $[a, b]$ must be attained within the interval (at points c_2 and c_1). A more involved theorem is the following:

Theorem 0.10. *Suppose $f \in C[a, b]$ is n-times differentiable on (a, b). If $f(x) = 0$ at $n + 1$ distinct numbers $a \leq x_0 < x_1 < \cdots < x_n \leq b$, then a number $c \in (x_0, x_n)$, (and hence in (a, b)) exists with $f^{(n)}(c) = 0$.*

Consider the fourth degree polynomial $f(x) = 8x^4 - 8x^2 + 1$:

```
f = @(x) 8*x.^4-8*x.^2+1; % has 4 zeros on (-1,1)
dddf = @(x) 8*4*3*2*x; % must have 1 zero on (-1,1)
x = linspace(-1,1,100);
plot(x,dddf(x))
```

The plot that results is shown in Figure 3

Theorem 0.11 (Taylor's Theorem). *Suppose $f \in C^n[a, b]$, and that $f^{(n+1)}$ exists on $[a, b]$, and $x_0 \in [a, b]$. For every $x \in [a, b]$, there exists a number $\xi(x)$ between x_0 and x with*

$$f(x) = P_n(x) + R_n(x),$$

where

$$P_n(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n,$$

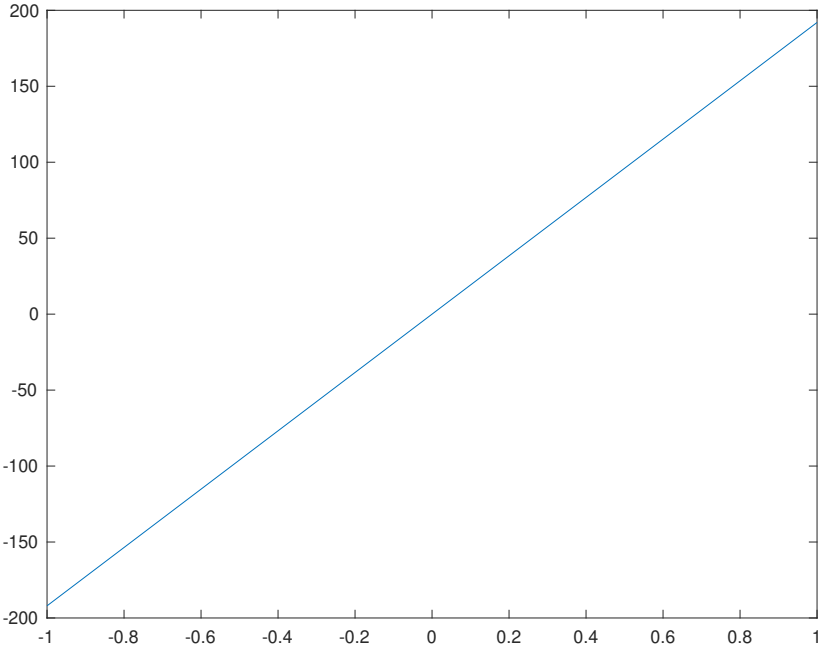


Figure 3: A demonstration of Theorem 0.10 with $f(x) = 8x^4 - 8x^2 + 1$.

and

$$R_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)^{n+1}.$$

0.1.2 ▪ Demonstrations from integral calculus

Definition 0.12. *The Riemann integral of the function f defined on the interval $[a, b]$ is the following limit (if it exists):*

$$\int_a^b f(x)dx = \lim_{\max \Delta x_i \to 0} \sum_{i=1}^{n} f(\bar{x}_i)\Delta x_i,$$

where the numbers $x_0, x_1, \ldots, x_n$ satisfy $a = x_0 \leq x_1 \leq \cdots \leq x_n = b$, $\Delta x_i = x_i - x_{i-1}$ for $i = 1, 2, \ldots, n$. And $\bar{x}_i$ is an arbitrary point in the interval $[x_{i-1}, x_i]$.

Let's choose the points x_i to be evenly spaced: $x_i = a + i\frac{b-a}{n}$ and $\bar{x}_i = x_i$. Then we have $\Delta x_i = \frac{b-a}{n}$ and

$$\int_a^b f(x)dx = \lim_{n \to \infty} \frac{b-a}{n} \sum_{i=1}^{n} f(x_i).$$

```
1 f = @(x) exp(x);
2 n = 10; a = -1; b = 1;
3 x = linspace(a,b,n+1); % create n + 1 points
```

```
x = x(2:end); % take the last n of these points
est = (b-a)/n*sum(f(x)) % evaluate f at these points and add them up
actual = exp(b)-exp(a) % the actual value
abs(est-actual)
```

```
est =

   2.593272082493666


actual =

   2.350402387287603


ans =

   0.242869695206063
```

Now choose and $\bar{x}_i = \dfrac{x_i + x_{i-1}}{2}$ to be the midpoint. We still have $\Delta x_i = \frac{b-a}{n}$ and

$$\int_a^b f(x)dx = \lim_{n\to\infty} \frac{b-a}{n} \sum_{i=1}^{n} f(\bar{x}_i).$$

```
f = @(x) exp(x);
n = 10; a = -1; b = 1;
x = linspace(a,b,n+1); % create n + 1 points
x = x(2:end); % take the last n of these points
x = x - (b-a)/(2*n); % shift to the midpoint
est = (b-a)/n*sum(f(x)) % evaluate f at these points and add them up
actual = exp(b)-exp(a) % the actual value
abs(est-actual)
```

```
est =

   2.346489615388305


actual =

   2.350402387287603


ans =

   0.003912771899298
```

Theorem 0.13 (Weighted Mean Value Theorem). *Suppose $f \in C[a,b]$, the Riemann integral of g exists on $[a,b]$, and $g(x)$ does not change sign on $[a,b]$. Then there*

exists a number c *in* (a, b) *with*

$$\int_a^b f(x)g(x)dx = f(c)\int_a^b g(x)dx.$$

Part I

Numerical linear algebra

Chapter 1
Systems of equations

Before we start combining mathematics with programming in MATLAB we need to learn some of the mathematics!

1.1 ▪ A linear system of equations

Consider the following basic problem of fitting a line to data:

Problem 1.1. *Find the line that passes through the coordinates* $(0, 1)$ *and* $(-1, 2)$.

To solve this problem we start with our general form for a line

$$y = f(x) = ax + b,$$

and then enforce the conditions $f(0) = 1$ and $f(-1) = 2$. This gives

$$\begin{aligned} a \cdot 0 + b &= 1, \\ a \cdot (-1) + b &= 2. \end{aligned}$$

This is a *linear system* of two equations for the two unknowns a, b. Below we will make the term linear precise and we will understand when we can solve such a system. We will also extend our understanding to systems of n equations for n unknowns. And even go further and generalize this to systems m linear equations with n unknowns!

Back to the problem at hand, we see that the first equation tells us $b = 1$ and the second equation gives $-a + 1 = 2$, or $a = -1$. So the line

$$y = -x + 1,$$

passes through the coordinates $(0, 1)$ and $(-1, 2)$.

1.1.1 ▪ Vectors

A vector can be understood as an $n \times 1$ array of numbers:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}.$$

We will also use the shortened notation $\mathbf{v} = (v_j)_{j=1}^n$ to represent the same vector. We refer to the v_j's as the entries, or components, of $\mathbf{v}$. Unless otherwise stated, for a vector $\mathbf{v}$, v_j will refer to its entries. Sometimes we will encounter *row vectors*. A row vector is a $1 \times n$ array of numbers.

Some more notation before we proceed:

- $\mathbb{R}$ and $\mathbb{C}$ refer to all real and complex numbers[1], respectively.
- To state that a is a real number, we write $a \in \mathbb{R}$ (and similarly $a \in \mathbb{C}$) if a is complex).
- $\mathbf{v}$ is a vector with n entries, all being real numbers, we say $v \in \mathbb{R}^n$ (and similarly $\mathbf{v} \in \mathbb{C}^n$) if $\mathbf{v}$ has complex entries).
- We use the notation $\sum_{j=1}^n v_j = v_1 + v_2 + \cdots + v_n$.

Vectors represent an extension of our usual real (or complex) number system and so we should give some rules for addition, subtraction, and multiplication.

Definition 1.2. *We use the following rules for manipulating vectors* $\mathbf{u} = (u_j)_{j=1}^n, \mathbf{v} = (v_j)_{j=1}^n$ *in* $\mathbb{R}^n$*:*

(1) $\mathbf{u} + \mathbf{v} = (u_j + v_j)_{j=1}^n$,

(2) $\mathbf{u} - \mathbf{v} = (u_j - v_j)_{j=1}^n$,

(3) $a\mathbf{v} = (av_j)_{j=1}^n$ *for* $a \in \mathbb{R}$*, and*

(4) $\mathbf{u} = \mathbf{v}$ *if and only if* $\mathbf{u} - \mathbf{v} = \mathbf{0}$ *where* $\mathbf{0} \in \mathbb{R}^n$ *is the vector of all zeros.*

These rules state that everything just happens entry-by-entry. The last statement is equivalent to $u_j = v_j$ for every j. Note that enforcing that a vector with n entries is zero is actually enforcing n conditions! We also see the convenience of the notation introduced above.

The rules we are used to for scalar multiplication and addition carry over:

Theorem 1.3. *Suppose* $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ *and* $a, b \in \mathbb{R}$*. Then*

- $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$ *(Associative, vector addition)*
- $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$ *(Commutative, vector addition)*
- $a(\mathbf{u} + \mathbf{v}) = (a\mathbf{u}) + (a\mathbf{v})$ *(Distributive, vector addition)*
- $(a + b)\mathbf{u} = a\mathbf{u} + b\mathbf{u}$ *(Distributive, scalar addition)*
- $\mathbf{u} + \mathbf{0} = \mathbf{u}$ *(Additive identity)*
- $\mathbf{u} + (-\mathbf{u}) = \mathbf{0}$ *(Additive inverse)*

Example 1.4. Let $\mathbf{v} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{w} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. Then

[1] A complex number z is equal to $z = x + \mathrm{i}\, y$ where $x, y \in \mathbb{R}$ and $\mathrm{i} = \sqrt{-1}$. While these numbers may feel strange and artificial, they are extremely helpful in simplifying computations, especially on a computer.

- $\mathbf{v} + \mathbf{w} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$, and
- $2\mathbf{w} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$.

Then Problem 1.1, a system of two equations, can be summarized using one *vector equation*. To see how to do this first note that

$$\begin{bmatrix} a \cdot 0 \\ a \cdot (-1) \end{bmatrix} = a \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} b \\ b \end{bmatrix} = b \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

So then we need to enforce that

$$a \begin{bmatrix} 0 \\ -1 \end{bmatrix} + b \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

This conversion of the problem does not actually help us solve it! We need to introduce matrices before this conversion really becomes useful.

1.1.2 ▪ Matrices

A matrix can be understood as an $m \times n$ array of numbers

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & \vdots \\ \vdots & & \ddots & \\ a_{m1} & \cdots & & a_{mn} \end{bmatrix}.$$

And just like the case of vectors we use the notation $A = (a_{ij})_{\substack{1 \le i \le m \\ 1 \le j \le n}}$ to quickly refer to the matrix. In this index notation we always take the top index (i in this case) to refer to the rows of the matrix and the bottom index (j) to refer to columns. Unless otherwise stated, for a matrix A, a_{ij} will refer to its entries.

Example 1.5. Write down the matrix A given by

$$A = \left(\frac{1}{i+j} \right)_{\substack{1 \le i \le 3 \\ 1 \le j \le 4}}.$$

The matrix A is then given by

$$A = \begin{bmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}.$$

Similar to vectors, if A is an $m \times n$ matrix of real numbers, we write $A \in \mathbb{R}^{m \times n}$ (similarly $A \in \mathbb{C}^{m \times n}$ if A has complex entries). And we have an analogous set of rules.

Definition 1.6. *We use the following rules for manipulating matrices* $A = (a_{ij})_{\substack{1 \le i \le m \\ 1 \le j \le n}}, B = (b_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le m}}$ *in* $\mathbb{R}^{m \times n}$*:*

(1) $A + B = (a_{ij} + b_{ij})_{\substack{1\le i\le m,\\ 1\le j\le n}}$,

(2) $A - B = (a_{ij} - b_{ij})_{\substack{1\le i\le m,\\ 1\le j\le n}}$,

(3) $cA = (ca_{ij})_{\substack{1\le i\le m\\ 1\le j\le n}}$ *for* $c \in \mathbb{R}$, *and*

(4) $A = B$ *if and only if* $A - B = \mathbf{0}$ *where* $\mathbf{0} \in \mathbb{R}^{m\times n}$ *is the matrix of all zeros.*

As with vectors, addition, multiplication and subtraction are simply taken entry-by-entry. Note that matrix sizes must be the same for them to added or subtracted.

Example 1.7. Let $A = \begin{bmatrix} 1 & 2 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & -1 \end{bmatrix}$. Then

$$A + B = \begin{bmatrix} 2 & 2 & 0 \\ -2 & 0 & 0 \end{bmatrix},$$
$$3A = \begin{bmatrix} 3 & 6 & 0 \\ -3 & 0 & 3 \end{bmatrix}.$$

Matrix-vector multiplication

We now return to this equation

$$a \begin{bmatrix} 0 \\ -1 \end{bmatrix} + b \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

The first equality can be taken as a definition of the matrix-vector product and it motivates the general definition below. But before we discuss this in greater detail we introduce some notation for columns of a matrix. Suppose $A \in \mathbb{R}^{m\times n}$. The vectors $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n \in \mathbb{R}^m$ are the columns of A, i.e.,

$$A = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{bmatrix}.$$

Definition 1.8. *Suppose* $A \in \mathbb{R}^{m\times n}$ *and* $\mathbf{v} \in \mathbb{R}^n$ *then we define*

$$A\mathbf{v} = \sum_{j=1}^{n} v_j \mathbf{a}_j.$$

In other words, the matrix-vector product $A\mathbf{v}$ gives a weighted sum of the columns of A.

Example 1.9.

$$\begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = (-1) \begin{bmatrix} 0 \\ -1 \end{bmatrix} + (1) \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

But this is not typically how one computes this by hand. By hand, most students find it easier to use the rule

Theorem 1.10. *For $A \in \mathbb{R}^{m\times n}$ and $v \in \mathbb{R}^n$*

$$A\mathbf{v} = \left(\sum_{j=1}^{n} a_{ij}v_j\right)_{i=1}^{m}.$$

Example 1.11. To compute

$$\mathbf{w} = \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 2 & 3 \end{bmatrix}}_{2\times 3} \underbrace{\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}}_{3\times 1}$$

we first note that the "inner dimensions" match and therefore we have a well-defined product. The dimension of the output is calculated by

$$(2 \times 3) \times (3 \times 1) \longrightarrow (2\times \cancel{3}) \times (\cancel{3} \times 1) \longrightarrow 2 \times 1.$$

So, $\mathbf{w} \in \mathbb{R}^2$. By Theorem 1.10 we see that

$$w_1 = 1 \cdot 1 + 0 \cdot 0 + (-1) \cdot (-1) = 2.$$

This is computed by multiplying the first row of the matrix times the vector and adding. Similarly, the second entry is given by

$$w_2 = 1 \cdot 1 + 2 \cdot 0 + 3 \cdot (-1) = -2.$$

We conclude that $\mathbf{w} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$. Let's now go back and verify that this gives the same as our definition:

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = 1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 0 \begin{bmatrix} 0 \\ 2 \end{bmatrix} + (-1) \begin{bmatrix} -1 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}.$$

Computing a matrix-vector product via Theorem 1.10 is simpler by hand because you can work entry-by-entry and you do not have to think about all of the columns at once. But for a computer, thinking about all the columns is actually a simpler way of doing things! This highlights something interesting that we'll encounter again, many times, in this book:

- Something that is hard for human computation may be "easy" for a computer.

1.1.3 ▪ Matrix-matrix product

While it is not immediately clear why we will ever need to develop a way to multiply two matrices, we define it here. It turns out this is absolutely critical!

Definition 1.12. *Suppose $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{\ell\times m}$. Then*

$$BA = \begin{bmatrix} B\mathbf{a}_1 & B\mathbf{a}_2 & \cdots & B\mathbf{a}_n \end{bmatrix}.$$

Note that

$$\underbrace{B}_{\ell\times m}\underbrace{A}_{m\times n},$$

so the output should be

$$(\ell \times m) \times (m \times n) \longrightarrow (\ell \times \not{m}) \times (\not{m} \times n) \longrightarrow \ell \times n.$$

This is confirmed by noting that $B\mathbf{a}_j \in \mathbb{R}^\ell$ and we have n of these vectors. To actually compute the matrix-matrix product by hand, it is often easiest to fill out each column of the resulting matrix at a time, by doing a matrix-vector product.

Example 1.13.

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ -2 & 6 \end{bmatrix}$$

The summary of the operations is given below.

Theorem 1.14. *Suppose A, B, C are matrices of the appropriate dimensions[2] so that the following combinations are defined. Suppose, $a, b \in \mathbb{R}$. Then*

- $A + B = B + A$ *(Commutative, matrix addition)*
- $(A + B) + C = A + (B + C)$ *(Associative, matrix addition)*
- $A + \mathbf{0} = A$ *(Additive identity)*
- $A + (-A) = \mathbf{0}$ *(Additive inverse)*
- $c(dA) = (cd)A$ *(Associative, scalar multiplication)*
- $c(A + B) = (cA) + (cB)$ *(Distributive, scalar multiplication) item* $(c + d)A = (cA) + (dA)$ *(Distributive, scalar multiplication)*
- $(AB)C = A(BC)$ *(Associative, matrix multiplication)*
- $A(B + C) = (AB) + (AC)$ *(Distributive, matrix multiplication)*
- $(B + C)A = (BA) + (CA)$ *(Distributive, matrix multiplication)*
- $c(AB) = A(cB) = (cA)B$ *(Compatibility)*
- $IA = A = AI$ *(Multiplicative identity)*
- $A\mathbf{0} = \mathbf{0} = \mathbf{0}A$ *(Zero matrix)*

[2]The dimensions may vary from line to line.

1.2 ▪ Creating and using vectors and matrices in Matlab

This section is the first example of what will become common in this text. Right after some new mathematical concepts are introduce, we realize them using MATLAB code.

1.2.1 ▪ Vectors

Here is the creation of a row vector `v` and a (column) vector `w`.

```
1  v = [1,2,3]; % row vector
2  w = [1;2;3]; % column vector
3  v
4  w
```

```
v =

     1     2     3


w =

     1
     2
     3
```

Now, let us create these vectors and then grab specific entries or groups of entries. The last command here add up all the entries in a vector. The output that follows gives the result from lines 3-6.

```
1  v = [1,2,3]; % row vector
2  w = [1;2;3]; % column vector
3  w(1)
4  v(1:2)
5  w(2:3)
6  sum(w)
```

```
ans =

     1


ans =

     1     2


ans =

     2
     3
```

```
ans =

     6
```

1.2.2 ▪ Matrices

Now, we will create a matrix by manually specifying every entry. MATLAB uses the semicolon ; as a line break.

```
M = [1,2,3,4;5,6,7,8;9,10,11,12;12,14,15,16]
M(1,4) % get the (1,4) element
M(2:3,2:3)% get the "middle" block of the matrix
M(3,:) % Get row 3
sum(M)
```

```
M =

     1     2     3     4
     5     6     7     8
     9    10    11    12
    12    14    15    16


ans =

     4


ans =

     6     7
    10    11


ans =

     9    10    11    12


ans =

    27    32    36    40
```

For the last example in this section we confirm the multiplication in Example 1.13.

```
A = [1,0,-1; 1,2,3];
B = [1,1; 0,1; -1,1];
A*B
```

```
ans =

     2     0
    -2     6
```

1.3 ▪ Regular Gaussian elimination

Before we begin our discussion of Gaussian elimination we define some important concepts

Definition 1.15.

- *The diagonal of a matrix* $A \in \mathbb{R}^{m\times n}$ *is the entries* a_{ii} *for* $i = 1, 2, \ldots, \min\{m, n\}$.
- *A matrix* L *is said to be* lower triangular *if* $\ell_{ij} = 0$ *for all* $i < j$.
- *A matrix* U *is said to be* upper triangular *if* $u_{ij} = 0$ *for all* $i > j$.
- *An upper/lower-triangular matrix is said to be* special *if all the diagonal entries are ones.*

We use an example to illustrate (regular) Gaussian elimination. Consider the linear system of equations

$$\begin{aligned} x + 4y + z &= 2, \\ 2x + 12y + z &= 7, \\ x + 2y + 4z &= 3. \end{aligned} \tag{1.1}$$

We know that this system is equivalent to the vector equation

$$\begin{bmatrix} 1 & 4 & 1 \\ 2 & 12 & 1 \\ 1 & 2 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 7 \\ 3 \end{bmatrix}.$$

This last system is represented in shorthand by the *augmented matrix*

$$\left[\begin{array}{ccc|c} 1 & 4 & 1 & 2 \\ 2 & 12 & 1 & 7 \\ 1 & 2 & 4 & 2 \end{array}\right].$$

$$\begin{aligned} x + 4y + z &= 2 \\ 2x + 12y + z &= 7 \\ x + 2y + 4z &= 3 \end{aligned} \qquad \Leftrightarrow \qquad \left[\begin{array}{ccc|c} 1 & 4 & 1 & 2 \\ 2 & 12 & 1 & 7 \\ 1 & 2 & 4 & 2 \end{array}\right]$$

Now, on the left, we multiply the first equation by -2 and add it to the second equation

$$\begin{aligned} (-2)\cdot(x + 4y + z &= 2) \\ + \quad 2x + 12y + z &= 7 \\ \hline 0x + 4y - z &= 3 \end{aligned}$$

We write the system out and convert it to our augmented matrix

$$\begin{aligned} x + 4y + z &= 2 \\ 0x + 4y - z &= 3 \\ x + 2y + 4z &= 3 \end{aligned} \qquad \Leftrightarrow \qquad \left[\begin{array}{ccc|c} 1 & 4 & 1 & 2 \\ 0 & 4 & -1 & 3 \\ 1 & 2 & 4 & 2 \end{array}\right]$$

Then, the important observation is that the new augmented matrix can be found from the old by taking $(-2) \cdot$ (row one) + (row two) and replacing row two with this: $-2R_1 + R_2 \to R_2$. We call this an *elementary row operation.* Now, we eliminate x from the last equation:

$$\begin{aligned} &(-1) \cdot (x + 4y + z = 2) \\ + \quad &x + 2y + 4z = 3 \\ \hline &0x - 2y + 3z = 1 \end{aligned}$$

$$\begin{aligned} x + 4y + z &= 2 \\ 0x + 4y - z &= 3 \\ 0x - 2y + 3z &= 1 \end{aligned} \qquad \Leftrightarrow \qquad \left[\begin{array}{ccc|c} 1 & 4 & 1 & 2 \\ 0 & 4 & -1 & 3 \\ 0 & -2 & 3 & 1 \end{array}\right]$$

The last step of (regular) Gaussian elimination for this system is to eliminate y from the last equation

$$\begin{aligned} &(1/2) \cdot (4y - z = 3) \\ + \quad &- 2y + 3z = 1 \\ \hline &0y + 5/2z = 5/2 \end{aligned}$$

$$\begin{aligned} x + 4y + z &= 2 \\ 0x + 4y - z &= 3 \\ 0x + 0y + \frac{5}{2}z &= \frac{5}{2} \end{aligned} \qquad \Leftrightarrow \qquad \left[\begin{array}{ccc|c} 1 & 4 & 1 & 2 \\ 0 & 4 & -1 & 3 \\ 0 & 0 & \frac{5}{2} & \frac{5}{2} \end{array}\right]$$

This is the last step of Gaussian elimination.

Definition 1.16. *An elementary row operation of type 1 is of the form $aR_j + R_i \to R_i$ for $i \neq j$.*

Definition 1.17. *Regular Gaussian elimination is the process by which a matrix (square or rectangular) is reduced to upper triangular form using only row operations of type I where $i > j$.*

This definition states that in regular Gaussian elimination we can use only rows above to introduce zeros.

Algorithm 1: Regular Gaussian elimination `rGE`

Result: An upper triangular matrix, or failure
Input: $A \in \mathbb{R}^{n \times n+j}$, $j \geq 0$
begin
  for $j = 1$ *to* $n-1$ **do**
    if $a_{jj} = 0$ **then**
      A is not regular;
      return *failure*;
    end
    for $i = j+1$ *to* n **do**
      set $\ell_{ij} = a_{ij}/a_{jj}$;
      perform $-\ell_{ij}R_j + R_i \to R_i$ *on* A;
    end
  end
end

We now need to understand why introducing zeros in the matrix, i.e. elimination, is beneficial. For the system

$$\begin{aligned} x + 4y + z &= 2 \\ 0x + 4y - z &= 3 \\ 0x + 0y + \frac{5}{2}z &= \frac{5}{2} \end{aligned}$$

we know that the solutions of this are the same as the solutions of the original system (1.1). We first determine z:

$$\frac{5}{2}z = \frac{5}{2} \Rightarrow z = 1,$$

then determine y

$$4y - z = 3 \Rightarrow y = 1,$$

and, lastly, determine x

$$x + 4y + z = 2 \Rightarrow x = -3.$$

This process of working backward up the matrix is referred to as *back substitution*. To properly describe this process algorithmically, consider breaking up the final augmented matrix

$$\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 4 & 1 & 2 \\ 0 & 4 & -1 & 3 \\ 0 & 0 & \frac{5}{2} & \frac{5}{2} \end{array}\right].$$

Then the following algorithm is back substitution.

Algorithm 2: Back substitution `Backsub`

Result: The solution of $U\mathbf{x} = \mathbf{c}$
Input: An augmented matrix $\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right]$, $U \in \mathbb{R}^{n\times n}$ upper triangular and $u_{jj} \neq 0$ for all j
begin
  set $x_n = c_n/u_{nn}$;
  for $i = n-1$ *to* 1 *with increment* -1 **do**
    set $x_i = \frac{1}{u_{ii}}\left(c_i - \sum_{j=i+1}^{n} u_{ij}x_j\right)$;
  end
end

Remark 1.18. *We return to the example linear system from Problem 1.1, now written in augmented form,*

$$\left[\begin{array}{cc|c} 0 & 1 & 1 \\ -1 & 1 & 2 \end{array}\right].$$

It is clear that regular Gaussian elimination will immediately fail on this matrix. We will introduce more row elementary row operations soon to deal with this system.

So, we now give a name to matrices for which regular Gaussian elimination succeeds.

Definition 1.19. *A matrix A is called regular if regular Gaussian elimination succeeds in transforming it to an upper-triangular matrix with non-zero diagonal entries.*

1.4 ▪ The LU factorization

Now consider the matrix product (by divine inspiration!)

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -\frac{1}{2} & 1 \end{bmatrix}\begin{bmatrix} 1 & 4 & 1 \\ 0 & 4 & -1 \\ 0 & 0 & \frac{5}{2} \end{bmatrix} = \begin{bmatrix} 1 & 4 & 1 \\ 2 & 12 & 1 \\ 1 & 2 & 4 \end{bmatrix} = A.$$

This is called an LU factorization of the matrix A. Note that the first matrix in the product is lower triangular with ones on the diagonal. How is this useful? To really understand this we need to discuss matrix inverses.

Definition 1.20 (Matrix inverse). *The inverse of an $n \times n$ square matrix A is another matrix B such that*

$$BA = I = I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}.$$

We refer to $I = I_n$ as the $n \times n$ identity matrix and we denote $B = A^{-1}$.

Example 1.21. For a 1×1 matrix $A = \begin{bmatrix} a \end{bmatrix}$, where $a \in \mathbb{R}$, it is clear that $A^{-1} = \begin{bmatrix} 1/a \end{bmatrix}$. But note that for

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix},$$

it is no longer clear what A^{-1} is!

Multiplication by the identity matrix does not change the vector.

Proposition 1.22. *For $\mathbf{v} \in \mathbb{R}^n$, $I_n \mathbf{v} = \mathbf{v}$.*

So, suppose $A = LU$ and we want to solve

$$A\mathbf{x} = \mathbf{b} \tag{1.2}$$

for a vector $\mathbf{b}$. Substitute,

$$LU\mathbf{x} = \mathbf{b}.$$

Multiply by L^{-1}

$$L^{-1}LU\mathbf{x} = U\mathbf{x} = L^{-1}\mathbf{b}.$$

So, we have the system $U\mathbf{x} = L^{-1}\mathbf{b}$ and we can now multiply by U^{-1}

$$U^{-1}U\mathbf{x} = \mathbf{x} = U^{-1}L^{-1}\mathbf{b}.$$

This entirely oversimplifies this whole procedure. How do we compute L^{-1}, U^{-1}? And then we have to multiply by them? This is something that is accomplished all at once.

Theorem 1.23. *Suppose $U \in \mathbb{R}^{n \times n}$ is upper triangular with non-zero diagonal entries. Then computing $U^{-1}\mathbf{c}$ gives the exact same result as solving $\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right]$ with back substitution.*

This implies that (1) we never need to actually find out what U^{-1} is and (2) we already have an algorithm for handling this computation.

For this procedure to be successful we need to have an analogous procedure to handle L. Since L is lower-triangular, we can solve with forward substitution.

Algorithm 3: Forward substitution `Forsub`.

Result: The solution of $L\mathbf{c} = \mathbf{b}$
Input: An augmented matrix $\left[\begin{array}{c|c} L & \mathbf{b} \end{array}\right]$, $L \in \mathbb{R}^{n \times n}$ lower triangular and $\ell_{jj} = 1$ for all j
begin
  set $c_1 = b_1$;
  for $i = 2$ *to* n **do**
    set $c_i = b_i - \sum_{j=1}^{i-1} \ell_{ij} c_j$;
  end
end

Theorem 1.24. *Suppose $L \in \mathbb{R}^{n \times n}$ is lower triangular with ones on the diagonal.*

Then computing $L^{-1}\mathbf{b}$ gives the exact same result as solving $\left[\begin{array}{c|c} L & \mathbf{b} \end{array}\right]$ with forward substitution.

With this in hand we can now describe the use of the LU factorization — if we have it.

Algorithm 4: Solve a system using an LU factorization

Result: The solution of $LU\mathbf{x} = \mathbf{b}$
Input: $U \in \mathbb{R}^{n\times n}$ upper triangular and $u_{jj} \neq 0$ for all j and a special lower-triangular matrix $L \in \mathbb{R}^{n\times n}$
begin
  set $\mathbf{c} =$ `Forsub`$(\left[\begin{array}{c|c} L & \mathbf{b} \end{array}\right])$;
  set $\mathbf{x} =$ `Backsub`$(\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right])$;
end

Before describing how to compute the LU factorization, we pause to note that when we applied regular Gaussian elimination to the augmented system $\left[\begin{array}{c|c} A & \mathbf{b} \end{array}\right]$ and reduced it to $\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right]$, the forward substitution step is not needed (indeed, it is actually encoded within regular Gaussian elimination!).

Algorithm 5: Solve a system using regular Gaussian elimination

Result: The solution of $A\mathbf{x} = \mathbf{b}$
Input: $A \in \mathbb{R}^{n\times n}$, regular.
begin
  set $\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right] =$ `rGE`$(\left[\begin{array}{c|c} A & \mathbf{b} \end{array}\right])$;
  set $\mathbf{x} =$ `Backsub`$(\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right])$;
end

1.5 ▪ Computing the LU factorization

If we look back at `rGE` as defined in Algorithm 1 we see that quantities ℓ_{ij} are defined in the process. These are exactly the (strictly) lower-triangular entries of L. We will work with an example to see that this is indeed the case. Note that in Algorithm 1, ℓ_{ij} is defined but $-\ell_{ij}$ is used in the row operation. So, we collect ℓ_{ij}'s that are used in (1.1):

$$\begin{aligned}(i,j) = (2,1) &\Rightarrow \ell_{21} = 2,\\ (i,j) = (3,1) &\Rightarrow \ell_{21} = 1,\\ (i,j) = (3,2) &\Rightarrow \ell_{32} = -1/2.\end{aligned}$$

One can then check that

$$\begin{bmatrix} 1 & 4 & 1 \\ 2 & 12 & 1 \\ 1 & 2 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -1/2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 1 \\ 0 & 4 & -1 \\ 0 & 0 & \frac{5}{2} \end{bmatrix}.$$

But this is a long way from actually establishing that this is how we choose the entries. To see this is the case we introduce *elementary matrices*.

Definition 1.25. *An elementary matrix is formed by applying an elementary row operation to the identity matrix.*

Example 1.26. For the row operation $aR_1 + R_2 \to R_2$ on a 3×3 matrix we have

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{aR_1+R_2\to R_2} \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = E.$$

Here E is the elementary matrix associated to the row operation $aR_1 + R_2 \to R_2$ on a 3×3.

Next consider our example matrix, for which we do the initial row operation $(-2)R_1 + R_2 \to R_2$

$$\begin{bmatrix} 1 & 4 & 1 \\ 2 & 12 & 1 \\ 1 & 2 & 4 \end{bmatrix} \xrightarrow{(-2)R_1+R_2\to R_2} \begin{bmatrix} 1 & 4 & 1 \\ 0 & 4 & -1 \\ 1 & 2 & 4 \end{bmatrix}.$$

Then consider multiplication by the associated elementary matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 1 \\ 2 & 12 & 1 \\ 1 & 2 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 1 \\ 0 & 4 & -1 \\ 1 & 2 & 4 \end{bmatrix}.$$

So, we see that this elementary matrix applies the row operation! Continuing through all three row operations, we find

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \end{bmatrix}}_{E_3} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}}_{E_2} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{E_1} \underbrace{\begin{bmatrix} 1 & 4 & 1 \\ 2 & 12 & 1 \\ 1 & 2 & 4 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 4 & 1 \\ 0 & 4 & -1 \\ 0 & 0 & \frac{5}{2} \end{bmatrix}}_{U}. \tag{1.3}$$

1.5.1 ▪ The inverse of an elementary matrix of type 1

An elementary row operation of type 1, $aR_j + R_i \to R_i$ does not change R_j. To reverse, or undo, this operation we should add $-aR_j$ to R_i. Or, in other words, the elementary row operation $-aR_j + R_i \to R_i$ is inverse to $aR_j + R_i \to R_i$ and this then implies that, for example,

$$\begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -a & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which is easily verified by seeing that

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

This discussion shows us that

$$E_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad E_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad E_3^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{bmatrix}.$$

At this point we can work abstractly with (1.3) to solve for an expression for A

$$\begin{aligned} E_3E_2E_1A &= U, \\ E_3^{-1}E_3E_2E_1A &= E_3^{-1}U, \\ E_2E_1A &= E_3^{-1}U, \\ E_2^{-1}E_2E_1A &= E_2^{-1}E_3^{-1}U, \\ E_1A &= E_2^{-1}E_3^{-1}U, \\ E_1^{-1}E_1A &= E_1^{-1}E_2^{-1}E_3^{-1}U, \\ A &= \underbrace{E_1^{-1}E_2^{-1}E_3^{-1}}_{L}U. \end{aligned}$$

In principle, computing the matrix L requires multiplying three 3×3 matrices. If we were working with a 4×4 matrix we'd be looking at multiplying six 4×4 matrices! This is too much work (for us or for a computer) and that is part of the magic of Gaussian elimination:

$$E_1^{-1}E_2^{-1}E_3^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -\frac{1}{2} & 1 \end{bmatrix}.$$

It is important to note that this is not how matrix-matrix multiplication works in general, but because of the specific structure of these matrices we can just fill in entrys to compute the product of the inverses!

Algorithm 6: LU factorization `LU`

Result: L and U such that $LU = A$, or failure
Input: $A \in \mathbb{R}^{n \times n+j}$, $j \geq 0$
begin
  set $L = I_m$;
  set $U = A$;
  for $j = 1$ *to* $n - 1$ **do**
    if $u_{jj} = 0$ **then**
      A is not regular;
      return *failure*;
    end
    for $i = j + 1$ *to* n **do**
      set $\ell_{ij} = u_{ij}/u_{jj}$;
      perform $-\ell_{ij}R_j + R_i \to R_i$ *on* U;
    end
  end
end

Chapter 2
Pivoting

2.1 ▪ Pivoting

In terms of solving linear systems, the previous sections give us all the elementary tools we need to deal with regular matrices. But what if the matrix is not regular? We go back to Problem 1.1 to find a matrix that is not regular but one that we were able to solve. A new elementary row operation is introduced.

Definition 2.1. *An elementary row operation of type 2, denoted $R_i \Leftrightarrow R_j$, corresponds to the interchange of rows i and j.*

Example 2.2.

$$\left[\begin{array}{cc|c} 0 & 1 & 1 \\ -1 & 1 & 2 \end{array}\right] \xrightarrow{R_1 \leftrightarrow R_2} \left[\begin{array}{cc|c} -1 & 1 & 2 \\ 0 & 1 & 1 \end{array}\right]$$

And to understand that a row interchange in an augmented matrix will not change the solution of the associated linear systems of equations we just note that it only corresponds to reordering the equations

$$\begin{array}{l} 0a + b = 1 \\ -a + b = 2 \end{array} \quad \Leftrightarrow \quad \left[\begin{array}{cc|c} 0 & 1 & 1 \\ -1 & 1 & 2 \end{array}\right]$$

$$\begin{array}{l} -a + b = 2 \\ 0a + b = 1 \end{array} \quad \Leftrightarrow \quad \left[\begin{array}{cc|c} -1 & 1 & 2 \\ 0 & 1 & 1 \end{array}\right]$$

Definition 2.3. *A permutation matrix is the product of any number of elementary matrices associated to row operations of type 2.*

Example 2.4. For a 2×2 matrix

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \xrightarrow{R_1 \leftrightarrow R_2} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

With the addition of row operations of type 2 we expand the class of matrices that can be reduced to upper-triangular form.

Definition 2.5. *A square matrix is called* nonsingular *if it can be reduced to upper-triangular form with non-zero diagonal entries, using elementary row operations of type 1 & 2.*

The following is self-explanatory, but we include it to be clear.

Definition 2.6. *A square matrix is called* singular *if it not nonsingular.*

We have argued that type 1 and type 2 row operations do not change the solution(s) of a linear system. And we know that back substitution succeeds. This implies that we can always find a solution of a linear system $A\mathbf{x} = \mathbf{b}$ for any choice of $\mathbf{b}$ if A is nonsingular. And since we find only one solution we have proved part (2) of the following:

Theorem 2.7.

(1) If $n \times n$ linear system $A\mathbf{x} = \mathbf{b}$ has a unique solution for every choice of right-hand side $\mathbf{b}$ then A is nonsingular.

(2) If $A \in \mathbb{R}^{n\times n}$ is nonsingular then $A\mathbf{x} = \mathbf{b}$ for every choice of right-hand side vector $\mathbf{b}$.

We will prove part (1) below which shows that a matrix A being nonsingular is the *de facto* condition for being able to solve a $n \times n$ linear system.

Let us now demonstrate the process of solving a linear system with a nonsingular matrix. Consider

$$\begin{aligned} 2y + 3z + w &= 1, \\ x + 3y + z + w &= -1, \\ x - y - 5z + w &= 0, \\ x + y + z + w &= 0. \end{aligned}$$

This is converted to the augmented matrix

$$\left[\begin{array}{cccc|c} 0 & 2 & 3 & 1 & 1 \\ 1 & 3 & 1 & 1 & -1 \\ 1 & -1 & -5 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{array}\right]$$

Then we proceed with Gaussian elimination

$$\left[\begin{array}{cccc|c} 0 & 2 & 3 & 1 & 1 \\ 1 & 3 & 1 & 1 & -1 \\ 1 & -1 & -5 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{array}\right] \xrightarrow{R_1 \leftrightarrow R_2} \left[\begin{array}{cccc|c} 1 & 3 & 1 & 1 & -1 \\ 0 & 2 & 3 & 1 & 1 \\ 1 & -1 & -5 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{array}\right]$$

$$\left[\begin{array}{cccc|c} 1 & 3 & 1 & 1 & -1 \\ 0 & 2 & 3 & 1 & 1 \\ 1 & -1 & -5 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{array}\right] \xrightarrow{-R_1+R_3 \to R_3} \left[\begin{array}{cccc|c} 1 & 3 & 1 & 1 & -1 \\ 0 & 2 & 3 & 1 & 1 \\ 0 & -4 & -6 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{array}\right]$$

$$\left[\begin{array}{cccc|c} 1 & 3 & 1 & 1 & -1 \\ 0 & 2 & 3 & 1 & 1 \\ 0 & -4 & -6 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{array}\right] \xrightarrow{-R_1+R_4 \to R_4} \left[\begin{array}{cccc|c} 1 & 3 & 1 & 1 & -1 \\ 0 & 2 & 3 & 1 & 1 \\ 0 & -4 & -6 & 0 & 1 \\ 0 & -2 & 0 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{cccc|c} 1 & 3 & 1 & 1 & -1 \\ 0 & 2 & 3 & 1 & 1 \\ 0 & -4 & -6 & 0 & 1 \\ 0 & -2 & 0 & 0 & 1 \end{array}\right] \xrightarrow{2R_2+R_3 \to R_3} \left[\begin{array}{cccc|c} 1 & 3 & 1 & 1 & -1 \\ 0 & 2 & 3 & 1 & 1 \\ 0 & 0 & 0 & 2 & 3 \\ 0 & -2 & 0 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{cccc|c} 1 & 3 & 1 & 1 & -1 \\ 0 & 2 & 3 & 1 & 1 \\ 0 & 0 & 0 & 2 & 3 \\ 0 & -2 & 0 & 0 & 1 \end{array}\right] \xrightarrow{R_2+R_4 \to R_4} \left[\begin{array}{cccc|c} 1 & 3 & 1 & 1 & -1 \\ 0 & 2 & 3 & 1 & 1 \\ 0 & 0 & 0 & 2 & 3 \\ 0 & 0 & 3 & 1 & 2 \end{array}\right].$$

We need to perform yet another row interchange

$$\left[\begin{array}{cccc|c} 1 & 3 & 1 & 1 & -1 \\ 0 & 2 & 3 & 1 & 1 \\ 0 & 0 & 0 & 2 & 3 \\ 0 & 0 & 3 & 1 & 2 \end{array}\right] \xrightarrow{R_3 \leftrightarrow R_4} \left[\begin{array}{cccc|c} 1 & 3 & 1 & 1 & -1 \\ 0 & 2 & 3 & 1 & 1 \\ 0 & 0 & 3 & 1 & 2 \\ 0 & 0 & 0 & 2 & 3 \end{array}\right],$$

and this finalizes the transformation to upper-triangular form. Then we proceed with back substitution:

$$\begin{aligned} 2w = 3 &\Rightarrow w = \frac{2}{3}, \\ 3z + w = 2 &\Rightarrow z = \frac{1}{6}, \\ 2y + 3z + w = 1 &\Rightarrow y = -\frac{1}{2}, \\ x + 3y + z + w = -1 &\Rightarrow x = -\frac{7}{6}. \end{aligned}$$

It should be clear that this is not a true "algorithm" at this point because there was an implicit choice made. In the first step, we swapped the first and second rows. We could have swapped the first and third rows instead. And even if we do not "need" to swap, we could anyway. The "simplest" algorithm is that when a zero is encountered that would make regular Gaussian elimination fail, we interchange the first row below that has a non-zero entry in that column. This is stated more precisely in the following algorithm.

Algorithm 7: nonsingular Gaussian elimination `GE`

Result: An upper triangular matrix, or failure
Input: $A \in \mathbb{R}^{n\times n+j},\ j \geq 0$
begin
  for $j = 1$ *to* $n-1$ **do**
    if $a_{kj} = 0$ *for all* $k \geq j$ **then**
      A is singular;
      return *failure*;
    end
    find *the smallest* k *such that* $a_{kj} \neq 0$ **then**
      perform $R_k \leftrightarrow R_j$ *on* A;
    end
    for $i = j+1$ *to* n **do**
      set $\ell_{ij} = a_{ij}/a_{jj}$;
      perform $-\ell_{ij}R_j + R_i \to R_i$ *on* A;
    end
  end
end

2.2 ▪ Pivoting strategies

The choice of which row to swap, and if to swap it, is called the pivoting strategy. The most commonly used method used is called *partial pivoting*. And to fully understand the need for this we present a simple numerical example.

Example 2.8. Consider the linear system

$$\begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 1-\epsilon \end{bmatrix}.$$

We can verify that the solution of this is given by

$$x = \frac{\epsilon}{\epsilon - 1}, \quad y = \frac{\epsilon - 1 - \epsilon^2}{\epsilon - 1}.$$

```
eps = 1e-8;
format long
A = [eps,1;1,1];
b = [1;1-eps];
U = [A,b];
U(2,:) = U(2,:) - (U(2,1)/U(1,1))*U(1,:);
x2 = U(2,3)/U(2,2);
x1 = 1/U(1,1)*(U(1,3) - U(1,2)*x2);

x1_real = eps/(eps-1);
x2_real = (eps-1-eps^2)/(eps-1);

err1 = x1-x1_real
err2 = x2-x2_real
```

```
err1 =

    -1.220446039250313e-08


err2 =

     0
```

We encounter an error that is significant! This is due to rounding errors — computers typically working with floating point arithmetic — due to inexact arithmetic.

But if we interchange the rows of the system first, we see significantly better rounding errors!

```
1  A = [1,1;eps,1];
2  b = [1-eps;1];
3  U = [A,b];
4  U(2,:) = U(2,:) - (U(2,1)/U(1,1))*U(1,:);
5  x2 = U(2,3)/U(2,2);
6  x1 = 1/U(1,1)*(U(1,3) - U(1,2)*x2);
7
8  err1 = x1-x1_real
9  err2 = x2-x2_real
```

```
err1 =

    -1.722921957828615e-16


err2 =

     0
```

The heuristics that produce the partial pivoting are that we want to maximize the diagonal entries in our upper-triangular matrix.

Algorithm 8: Gaussian elimination with partial pivoting `GEpp`

Result: An upper triangular matrix, or failure
Input: $A \in \mathbb{R}^{n \times n+j},\ j \geq 0$
begin
  for $j = 1$ *to* $n-1$ **do**
    if $a_{kj} = 0$ *for all* $k \geq j$ **then**
      A is singular;
      return *failure*;
    end
    find k *such that* $|a_{kj}| \geq |a_{\ell j}|$ *for all* $\ell \geq j$ **then**
      perform $R_k \leftrightarrow R_j$ *on* A;
    end
    for $i = j+1$ *to* n **do**
      set $\ell_{ij} = a_{ij}/a_{jj}$;
      perform $-\ell_{ij}R_j + R_i \to R_i$ *on* A;
    end
  end
end

2.3 ▪ The permuted LU factorization

Once pivoting is involved, we will no longer just factor $A = LU$, we will see that we actually end up factoring $PA = LU$ for a permutation matrix P. Let us see how this works with the matrix

$$\begin{bmatrix} 0 & 2 & 3 & 1 \\ 1 & 3 & 1 & 1 \\ 1 & -1 & -5 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

First, write

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{will become } P} \begin{bmatrix} 0 & 2 & 3 & 1 \\ 1 & 3 & 1 & 1 \\ 1 & -1 & -5 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{will become } L} \underbrace{\begin{bmatrix} 0 & 2 & 3 & 1 \\ 1 & 3 & 1 & 1 \\ 1 & -1 & -5 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}}_{\text{will become } U}.$$

And we need some facts to proceed:

(1) If P is the elementary matrix associated with $R_i \leftrightarrow R_j$ then $PP = I$

(2) If P is the elementary matrix associated with $R_i \leftrightarrow R_j$ then multiplication on the right, AP, interchanges columns i and j of A.

Let P_1 be the elementary matrix associated with $R_1 \leftrightarrow R_2$. Then because $P_1P_1 = I$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 3 & 1 \\ 1 & 3 & 1 & 1 \\ 1 & -1 & -5 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} P_1P_1 \begin{bmatrix} 0 & 2 & 3 & 1 \\ 1 & 3 & 1 & 1 \\ 1 & -1 & -5 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Then using fact (2)

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

But we want the first matrix on the right-hand side to be upper triangular. Multiplying both sides by P_1 get us there

$$P_1\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&2&3&1\\1&3&1&1\\1&-1&-5&1\\1&1&1&1\end{bmatrix}=P_1\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&3&1&1\\0&2&3&1\\1&-1&-5&1\\1&1&1&1\end{bmatrix},$$

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

This might seem like more work than is really necessary for this step but it is really what is needed to deal with what happens further along in the process. Now we can perform the row operations $-R_1+R_3\to R_3$ and $-R_1+R_4\to R_4$ in the last matrix, accounting for this in the first matrix on the right-hand side, exactly as in the case of the LU factorization:

$$\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&2&3&1\\1&3&1&1\\1&-1&-5&1\\1&1&1&1\end{bmatrix}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\1&0&1&0\\1&0&0&1\end{bmatrix}\begin{bmatrix}1&3&1&1\\0&2&3&1\\0&-4&-6&0\\0&-2&0&0\end{bmatrix}.$$

We proceed with the next two row operations

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

And to finish the upper triangularization, we need to interchange the last two rows. So let P_2 be the elementary matrix for $R_3\leftrightarrow R_4$, giving

$$\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&2&3&1\\1&3&1&1\\1&-1&-5&1\\1&1&1&1\end{bmatrix}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\1&-2&1&0\\1&-1&0&1\end{bmatrix}P_2P_2\begin{bmatrix}1&3&1&1\\0&2&3&1\\0&0&0&2\\0&0&3&1\end{bmatrix}.$$

Absorbing P_2 into neighboring matrices gives

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

Like in our first step, we have broken the lower-triangular structure of the first factor on the right-hand side. So, we solve this problem by multiplying through by P_2

$$P_2\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&2&3&1\\1&3&1&1\\1&-1&-5&1\\1&1&1&1\end{bmatrix} = P_2\begin{bmatrix}1&0&0&0\\0&1&0&0\\1&-2&0&1\\1&-1&1&0\end{bmatrix}\begin{bmatrix}1&3&1&1\\0&2&3&1\\0&0&3&1\\0&0&0&2\end{bmatrix},$$

$$\underbrace{\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}}_{P}\underbrace{\begin{bmatrix}0&2&3&1\\1&3&1&1\\1&-1&-5&1\\1&1&1&1\end{bmatrix}}_{A} = \underbrace{\begin{bmatrix}1&0&0&0\\0&1&0&0\\1&-1&1&0\\1&-2&0&1\end{bmatrix}}_{L}\underbrace{\begin{bmatrix}1&3&1&1\\0&2&3&1\\0&0&3&1\\0&0&0&2\end{bmatrix}}_{U}.$$

It takes quite a bit of work to prove that this procedure will always result in an LU product on the right. But once you do, a result of this fact is the following.

Theorem 2.9. *A matrix $A \in \mathbb{R}^{n\times n}$ is nonsingular if and only if it has a factorization $PA = LU$ where P is a permutation matrix, L is special lower triangular and U is upper triangular with non-zero diagonal entries.*

In code, and especially in MATLAB, it is unnecessary to construct P as a matrix. This permutation matrix P is actually captured by the following vector of integers

$$P = \begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix} \leftrightarrow p = \begin{bmatrix}2&1&4&3\end{bmatrix}$$

were the integer gives, for each column, the location of the non-zero entry. Note that MATLAB easily lets us apply permutations using this notation:

```
P = [0,1,0,0;
     1,0,0,0;
     0,0,0,1;
     0,0,1,0];
p = [2,1,4,3];
V = [1,1;2,2;3,3;4,4]
P*V
V(p,:)
```

```
V =

     1     1
     2     2
     3     3
     4     4


ans =
```

```
     2     2
     1     1
     4     4
     3     3


ans =

     2     2
     1     1
     4     4
     3     3
```

We see that constructing P and applying to a matrix gives the same result as simply indexing with p. The $PA = LU$ factorization is summarized below. While we will work with the transpose more later, we define the transpose for a vector here:

Definition 2.10. *The transpose of a vector*

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix},$$

is a row vector with the same entries

$$\mathbf{v}^T = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix}.$$

Conversely, the transpose of a row vector is a vector with the same entries

Algorithm 9: The $PA = LU$ factorization with partial pivoting `PLU`

Result: A permutation $\mathbf{p}$, a special lower-triangular matrix $L \in R^{n\times n}$ and a nonsingular matrix $U \in \mathbb{R}^{n\times n+j}$, or failure
Input: $A \in \mathbb{R}^{n\times n+j},\ j \geq 0$
set $U = A$;
set $L = I_n$;
set $\mathbf{p} = \begin{bmatrix}1 & 2 & \cdots & n\end{bmatrix}^T$;
begin
  for $j = 1$ *to* $n-1$ **do**
    if $u_{kj} = 0$ *for all* $k \geq j$ **then**
      A is singular;
      return *failure*;
    end
    find k *such that* $|u_{kj}| \geq |u_{\ell j}|$ *for all* $\ell \geq j$ **then**
      perform $R_k \leftrightarrow R_j$ *on* U;
      perform $R_k \leftrightarrow R_j$ *on* L;
      perform *a interchange of columns* k *and* j *of* L;
      perform $R_k \leftrightarrow R_j$ *on* $\mathbf{p}$;
    end
    for $i = j+1$ *to* n **do**
      set $\ell_{ij} = u_{ij}/u_{jj}$;
      perform $-\ell_{ij}R_j + R_i \to R_i$ *on* U;
    end
  end
end

2.4 ▪ Reducing memory requirements in $PA = LU$

Let us to understand how large a matrix we can actually compute with. At the writing of these notes, a high-end laptop will typically come with 16 gigabytes (GBs) of ram. While we will get more into the the details of this in Chapter A, a floating point number is represented using 64 bits — 64 0's and 1's. There are 8,589,934,592 bits in one GB. So, one GB can store 134,217,728 floating point numbers. This translates to 2,146,483,648 numbers that can be stored on such a laptop. A square $n \times n$ matrix has n^2 entries. So,

$$\sqrt{2,146,483,648} \approx 46,340.95\ldots.$$

So, if we ignore all of the memory requirements of an operating system we can store one 46340×46340 matrix in memory on the laptop. As Algorithm 9 is set up, we have to store 3 matrices! So the size of the matrix is cut by a factor of $\sqrt{3}$ (recall n^2 entries!). This would become even worse if the P matrix was not encoded as a row vector.

On this laptop Algorithm 9 can support a matrix of size at most 26754×26754. We can rework the algorithm to overwrite A as it proceeds so that the size of the matrix is not cut by $\sqrt{3}$. To describe this algorithm we introduce matrix indexing notation that is similar to MATLAB's.

Important

For a vector $\mathbf{v} = (v_j)_{j=1}^n$ we us $\mathbf{v}_{i:j}$ to denote the subvector of $\mathbf{v}$ consisting of entries

i through j. For a matrix $A = (a_{ij})_{\substack{1\le i\le m\\ 1\le j\le n}}$ we use

$$A_{i:j,k:\ell}$$

to refer to the submatrix of A consisting of rows i through j and columns k through ℓ if $i = j$ or $k = \ell$ we use $A_{i,k:\ell}$ or $A_{i:j,k}$, respectively.

Algorithm 10: The memory optimized $PA = LU$ factorization with partial pivoting `GEpp`.

Result: A $n \times (n+1)$ matrix and a permutation $\mathbf{p}$, or failure
Input: $A \in \mathbb{R}^{n\times n}$
set $\mathbf{p} = \begin{bmatrix}1 & 2 & \cdots & n\end{bmatrix}^T$;
begin
  for $j = 1$ *to* $n-1$ **do**
    if $a_{kj} = 0$ *for all* $k \ge j$ **then**
      A is singular;
      return *failure*;
    end
    find k *such that* $|a_{kj}| \ge |a_{\ell j}|$ *for all* $\ell \ge j$ **then**
      perform $R_k \leftrightarrow R_j$ *on* A;
      perform $R_k \leftrightarrow R_j$ *on* $\mathbf{p}$;
    end
    for $i = j+1$ *to* n **do**
      set $a_{ij} = a_{ij}/a_{jj}$;
      perform $-a_{ij}R_j + R_i \to R_i$ *on* $A_{1:n,j+1:n}$;
    end
  end
end

This algorithm cleverly exploits the fact that every entry of A in which Gaussian elimination eliminates an entry corresponds to an entry of L that is filled in. Furthermore, the entries of L should not be involved in any sort of row operation going forward, so the row operations are just performed on the last $n - j$ columns. We demonstrate how the output is used with some simple MATLAB code. Suppose we apply this algorithm to a matrix `A`:

```
1  A = randn(7)
```

```
A =

    1.6360   -0.0549    1.8089    0.9421    0.4128   -0.1318    0.4400
   -0.4251   -1.1187   -1.0799    0.3005   -0.9870    0.5954   -0.6169
    0.5894   -0.6264    0.1992   -0.3731    0.7596    1.0468    0.2748
   -0.0628    0.2495   -1.5210    0.8155   -0.6572   -0.1980    0.6011
   -2.0220   -0.9930   -0.7236    0.7989   -0.6039    0.3277    0.0923
   -0.9821    0.9750   -0.5933    0.1202    0.1769   -0.2383    1.7298
    0.6125   -0.6407    0.4013    0.5712   -0.3075    0.2296   -0.6086
```

And then suppose `A` is modified and outputted, along with `p`, by Algorithm 10:

```
[Anew,p] = GEpp(A)
```

```
Anew =

   -2.0220   -0.9930   -0.7236    0.7989   -0.6039    0.3277    0.0923
    0.4857    1.4573   -0.2418   -0.2678    0.4703   -0.3975    1.6850
    0.0311    0.1924   -1.4520    0.8422   -0.7289   -0.1317    0.2741
   -0.8091   -0.5890   -0.7444    2.0577   -0.3415   -0.1988    1.7112
   -0.2915   -0.6285    0.1127   -0.1961    0.8943    0.8684    1.6653
    0.2102   -0.6244    0.7429   -0.3209   -0.1503    0.4428    1.0118
   -0.3029   -0.6461   -0.0179    0.3184   -0.1016    0.4997   -0.3683


p =

     5
     6
     4
     1
     3
     2
     7
```

The factors L and U can be reconstructed simply[3].

```
L = eye(length(Anew)) + tril(Anew,-1)
U = triu(Anew)
```

```
L =

    1.0000         0         0         0         0         0         0
    0.4857    1.0000         0         0         0         0         0
    0.0311    0.1924    1.0000         0         0         0         0
   -0.8091   -0.5890   -0.7444    1.0000         0         0         0
   -0.2915   -0.6285    0.1127   -0.1961    1.0000         0         0
    0.2102   -0.6244    0.7429   -0.3209   -0.1503    1.0000         0
   -0.3029   -0.6461   -0.0179    0.3184   -0.1016    0.4997    1.0000


U =

   -2.0220   -0.9930   -0.7236    0.7989   -0.6039    0.3277    0.0923
         0    1.4573   -0.2418   -0.2678    0.4703   -0.3975    1.6850
         0         0   -1.4520    0.8422   -0.7289   -0.1317    0.2741
         0         0         0    2.0577   -0.3415   -0.1988    1.7112
         0         0         0         0    0.8943    0.8684    1.6653
         0         0         0         0         0    0.4428    1.0118
         0         0         0         0         0         0   -0.3683
```

[3]Note that this is not necessary to do either forward substitution or backward substitution!

Lastly, we can check to make sure, using `p`.

```
A(p,:) - L*U
```

```
ans =

   1.0e-15 *

         0         0         0         0         0         0         0
         0         0         0   -0.0139   -0.0278         0         0
         0         0         0         0         0         0         0
         0    0.1180         0   -0.3331   -0.0555         0   -0.1110
         0    0.1110   -0.0278         0         0         0   -0.1665
         0         0   -0.2220   -0.0555         0         0   -0.1110
         0         0    0.0555         0         0         0         0
```

This last snippet tells us that to compute PA we evaluate `A(p,:)`.

2.5 ▪ Using `GEpp`

We now describe how use the output of `GEpp` to solve a linear system. The benefit of the storage both L and U in one matrix is that if you think closely about how back and forward substitution algorithms work, you will notice that

$$\texttt{Forsub([L,b])} = \texttt{Forsub([Anew,b])}$$

for any vector `b` and similarly

$$\texttt{Backsub([U,c])} = \texttt{Backsub([Anew,c])}.$$

So, the solution procedure is given below.

```
A = randn(7);
b = ones(7,1);
[Anew,p] = GEpp(A);
c = Forsub([Anew,b(p)]);
x = Backsub([Anew,c])
```

```
x =

    1.1908
    1.6383
   -0.2902
   -0.0049
    0.0773
    1.7849
   -0.0917
```

We test this against the output of MATLAB's built in function to solve linear systems.

```
x_real = A \ b; % Matlab's notation for solve Ax=b
x_real - x % test our solution
```

```
ans =

   1.0e-15 *

    0.4441
    0.2220
   -0.1665
    0.0893
   -0.0971
    0.4441
   -0.0833
```

2.6 ▪ Gauss-Jordan elimination

Gauss-Jordan elimination is the process by which an augmented matrix is first reduced to upper-triangular form, and then the left-most square submatrix is reduced to the identity matrix. In order to accomplish this, we need our last row operation type.

Definition 2.11. *An elementary row operation of type 3, denoted $aR_i \to R_i$, corresponds to multiplication of row i by a non-zero scalar a.*

We now demonstrate Gauss-Jordan elimination on a 3×3 matrix, where we augment the matrix with the identity matrix. Note that the textbook reserves the term Gauss-Jordan elimination for the case when the matrix is augmented like this with the identity matrix. I will use this term more generally.

$$\left[\begin{array}{ccc|ccc} 1 & 4 & 1 & 1 & 0 & 0 \\ 2 & 12 & 1 & 0 & 1 & 0 \\ 1 & 2 & 4 & 0 & 0 & 1 \end{array}\right] \xrightarrow{-2R_1+R_2\to R_2} \left[\begin{array}{ccc|ccc} 1 & 4 & 1 & 1 & 0 & 0 \\ 0 & 4 & -1 & -2 & 1 & 0 \\ 1 & 2 & 4 & 0 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc} 1 & 4 & 1 & 1 & 0 & 0 \\ 0 & 4 & -1 & -2 & 1 & 0 \\ 1 & 2 & 4 & 0 & 0 & 1 \end{array}\right] \xrightarrow{-R_1+R_3\to R_3} \left[\begin{array}{ccc|ccc} 1 & 4 & 1 & 1 & 0 & 0 \\ 0 & 4 & -1 & -2 & 1 & 0 \\ 0 & -2 & 3 & -1 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc} 1 & 4 & 1 & 1 & 0 & 0 \\ 0 & 4 & -1 & -2 & 1 & 0 \\ 0 & -2 & 3 & -1 & 0 & 1 \end{array}\right] \xrightarrow{\frac{1}{2}R_2+R_3\to R_3} \left[\begin{array}{ccc|ccc} 1 & 4 & 1 & 1 & 0 & 0 \\ 0 & 4 & -1 & -2 & 1 & 0 \\ 0 & 0 & \frac{5}{2} & -2 & \frac{1}{2} & 1 \end{array}\right]$$

Then, we divide the last row by the value on the diagonal

$$\left[\begin{array}{ccc|ccc} 1 & 4 & 1 & 1 & 0 & 0 \\ 0 & 4 & -1 & -2 & 1 & 0 \\ 0 & 0 & \frac{5}{2} & -2 & \frac{1}{2} & 1 \end{array}\right] \xrightarrow{\frac{2}{5}R_3\to R_3} \left[\begin{array}{ccc|ccc} 1 & 4 & 1 & 1 & 0 & 0 \\ 0 & 4 & -1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -\frac{4}{5} & \frac{1}{5} & \frac{2}{5} \end{array}\right]$$

The $(3,3)$ entry is then used to eliminate entries above the diagonal

$$\left[\begin{array}{ccc|ccc} 1 & 4 & 1 & 1 & 0 & 0 \\ 0 & 4 & -1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -\frac{4}{5} & \frac{1}{5} & \frac{2}{5} \end{array}\right] \xrightarrow{R_3+R_2\to R_2} \left[\begin{array}{ccc|ccc} 1 & 4 & 1 & 1 & 0 & 0 \\ 0 & 4 & 0 & -\frac{14}{5} & \frac{6}{5} & \frac{2}{5} \\ 0 & 0 & 1 & -\frac{4}{5} & \frac{1}{5} & \frac{2}{5} \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc} 1 & 4 & 1 & 1 & 0 & 0 \\ 0 & 4 & 0 & -\frac{14}{5} & \frac{6}{5} & \frac{2}{5} \\ 0 & 0 & 1 & -\frac{4}{5} & \frac{1}{5} & \frac{2}{5} \end{array}\right] \xrightarrow{-R_3+R_1\to R_1} \left[\begin{array}{ccc|ccc} 1 & 4 & 0 & \frac{9}{5} & -\frac{1}{5} & -\frac{2}{5} \\ 0 & 4 & 0 & -\frac{14}{5} & \frac{6}{5} & \frac{2}{5} \\ 0 & 0 & 1 & -\frac{2}{5} & \frac{1}{5} & \frac{2}{5} \end{array}\right].$$

Right now, we could use row 2 to eliminate the $(1,2)$ entry. But we will follow the more algorithmic method of first dividing then then eliminating. So, we divide the second row by 4.

$$\left[\begin{array}{ccc|ccc} 1 & 4 & 0 & \frac{9}{5} & -\frac{1}{5} & -\frac{2}{5} \\ 0 & 4 & 0 & -\frac{14}{5} & \frac{6}{5} & \frac{2}{5} \\ 0 & 0 & 1 & -\frac{2}{5} & \frac{1}{5} & \frac{2}{5} \end{array}\right] \xrightarrow{\frac{1}{4}R_2 \to R_2} \left[\begin{array}{ccc|ccc} 1 & 4 & 0 & \frac{9}{5} & -\frac{1}{5} & -\frac{2}{5} \\ 0 & 1 & 0 & -\frac{7}{10} & \frac{3}{10} & \frac{1}{10} \\ 0 & 0 & 1 & -\frac{2}{5} & \frac{1}{5} & \frac{2}{5} \end{array}\right].$$

The last step of Gauss-Jordan elimination is to eliminate the $(2,1)$ entry

$$\left[\begin{array}{ccc|ccc} 1 & 4 & 0 & \frac{9}{5} & -\frac{1}{5} & -\frac{2}{5} \\ 0 & 1 & 0 & -\frac{7}{10} & \frac{3}{10} & \frac{1}{10} \\ 0 & 0 & 1 & -\frac{2}{5} & \frac{1}{5} & \frac{2}{5} \end{array}\right] \xrightarrow{-4R_2+R_1 \to R_1} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{23}{5} & -\frac{7}{5} & -\frac{4}{5} \\ 0 & 1 & 0 & -\frac{7}{10} & \frac{3}{10} & \frac{1}{10} \\ 0 & 0 & 1 & -\frac{2}{5} & \frac{1}{5} & \frac{2}{5} \end{array}\right].$$

We say that Gauss-Jordan elimination succeeds if the left block becomes the identity matrix.

Theorem 2.12.

- *Gauss-Jordan elimination succeeds on* $\left[\begin{array}{c|c} A & I \end{array}\right]$ *for* $A \in \mathbb{R}^{n\times n}$ *if and only if* A *is nonsingular.*
- *If Gauss-Jordan elimination succeeds on* $\left[\begin{array}{c|c} A & I \end{array}\right]$ *it produces* $\left[\begin{array}{c|c} I & B \end{array}\right]$ *where* $BA = I$*, or* $B = A^{-1}$*.*

The first part of this theorem follows from the fact that a matrix is nonsingular if and only if we can reduce it to upper-triangular form, with non-zero diagional entries, with type 1 and 2 row operations. Then we can divide by the diagonal entries and elimination what is above the diagonal. The second part follows from the fact that all row operations can be expressed as elementary matrices. Let $E_1, E_2, \ldots, E_k$ be the row operations that are sequentially performed:

$$E_k E_{k-1} \cdots E_1 \left[\begin{array}{c|c} A & I \end{array}\right] = \left[\begin{array}{c|c} I & B \end{array}\right].$$

Then it follows that both $E_k E_{k-1} \cdots E_1 A = I$ and $E_k E_{k-1} \cdots E_1 = B$.

To compute the inverse of a nonsingular matrix $A \in \mathbb{R}^{n\times n}$, apply Gauss–Jordan elimination to $\left[\begin{array}{c|c} A & I \end{array}\right]$ and take the augmented matrix!

It also follows from this that if A is nonsingular then applying Gauss-Jordan elimination is effectively the same as computing

2.7 ▪ Using the inverse matrix

Suppose that we use Gauss–Jordan elimination to compute the inverse A^{-1} for a nonsingular matrix A. Then if we want to solve

$$\begin{aligned} A\mathbf{x} &= \mathbf{b}, \\ \underbrace{A^{-1}A}_{I}\mathbf{x} &= A^{-1}\mathbf{b}, \\ \mathbf{x} &= A^{-1}\mathbf{b}. \end{aligned}$$

Definition 2.13. *Let $\mathbf{e}_j$ be the j column of the identity matrix I.*

In the previous definition, the dimension of the vector will be inferred from context. Returning to the context Theorem 2.7(1), we see that if we can solve $A\mathbf{x} = \mathbf{b}$ for every choice of $\mathbf{b}$, we can then, in particular, solve

$$A\mathbf{x}_j = \mathbf{e}_j, \quad \text{for} \quad j = 1, 2, \ldots, n, \tag{2.1}$$

To solve all of these systems at once, we could use the augmented matrix

$$\left[\begin{array}{c|cccc} A & \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_n \end{array}\right],$$

put A in upper triangular form using row operations of type 1 and 2, and the preform back substitution. It can be seen that this gives the same result as Gauss-Jordan elimination — x_j is the jth column of the inverse A^{-1}! This tells us the following.

Theorem 2.14. *For a square nonsingular matrix A,*

$$A^{-1}A = I = AA^{-1}.$$

Note that this does not prove Theorem 2.7(1)! We will do so in Section 2.9.

2.8 ▪ Complexity theory

At this point, one should be wondering why we waited so long to find a method to compute the inverse of a matrix, spending all this time on the $PA = LU$ factorization, if all we want to do is solve $A\mathbf{x} = \mathbf{b}$.

Why compute $PA = LU$ instead of A^{-1}?

The answer to this question we need to discuss the complexity of the operations we have described. We do this using FLOPs (floating point operations). We count addition, multiplication, subtraction and division as a FLOP.

Example 2.15 (Matrix-vector multiplication). The multiplication $A\mathbf{v}$, $A \in \mathbb{R}^{n\times n}$, $\mathbf{v} \in \mathbb{R}$ requires n^2 multiplications and $n(n-1) = n^2 - n$ additions.

So, if we know the inverse of matrix, we need to perform $2n^2 - n$ FLOPs. We now compare this with forward and back substitution

Example 2.16 (Forward substitution). Suppose L is a special lower-triangular matrix ($\ell_j j = 1$ for all j). Forward substituion to solve $L\mathbf{c} = \mathbf{b}$ requires $\frac{n^2-n}{2}$ additions and $\frac{n^2-n}{2}$ multiplications.

Example 2.17 (Back substitution). Suppose U is an upper-triangular matrix with non-zero diagonal entries. Back substituion to solve $U\mathbf{x} = \mathbf{c}$ requires $\frac{n^2-n}{2}$ additions and $\frac{n^2+n}{2}$ multiplications/divisions.

From these we conclude that it requires $2n^2 - n$ FLOPs to solve $A\mathbf{x} = \mathbf{b}$ with the $PA = LU$ factorization.

This is the same amount of "work" that is required to use A^{-1}. From a computational perspective, having the $PA = LU$ factorization is just as good as having the actual inverse!

But then we just make the clear observation that computing the inverse takes a lot more work than computing the $PA = LU$ factorization — twice as much, actually — and so in practice the inverse of a matrix is almost never calculated.

2.9 ▪ Singular matrices and determinants

We first discuss what happens when we perform row operations on a singular matrix. If $A \in \mathbb{R}^{n\times n}$ is singular then at some point we must encounter a situation where row interchanges do not allow us to put a non-zero entry on the diagonal of a matrix. That means the matrix U at this intermediate stage must be of the form

$$\begin{bmatrix} u_{11} & u_{1,2} & \cdots & \cdots & & & & \cdots & u_{1,n} \\ 0 & u_{22} & u_{2,3} & \cdots & & & & \cdots & u_{2,n} \\ 0 & 0 & \ddots & & & & & & \vdots \\ \vdots & \vdots & & & \ddots & & & & \\ 0 & 0 & \cdots & 0 & u_{k,k} & u_{k,k+1} & \cdots & \cdots & u_{k,n} \\ 0 & 0 & \cdots & 0 & 0 & \boxed{0} & u_{k+1,k+2} & \cdots & u_{k+1,n} \\ 0 & 0 & \cdots & 0 & 0 & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & u_{n,k+2} & \cdots & u_{n,n} \end{bmatrix}.$$

The zero that is highlighted is on the diagonal. We can continue to apply row operations on the lower right block:

$$\begin{bmatrix} u_{k+1,k+2} & \cdots & u_{k+1,n} \\ \vdots & & \vdots \\ u_{n,k+2} & \cdots & u_{n,n} \end{bmatrix}.$$

But we note that his has more rows than columns! And if we put this into upper-triangular form, there must be a zero in the (n, n) entry.

Proposition 2.18. *A singular matrix A can be reduced to upper triangular form with elementary row operations of type 1 and type 2 and the resulting upper-triangular matrix must have a row of all zeros.*

Consider the matrix

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

we wish to check if it is nonsingular. First, suppose $a \neq 0$. Then we do no need to interchange rows. We perform the one needed row operation

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \xrightarrow{R_2 - \frac{c}{a}R_1 \to R_2} \begin{bmatrix} a & b \\ 0 & d - \frac{c}{a}b \end{bmatrix}$$

Then the matrix is nonsingular if $d - \frac{c}{a}b \neq 0$ or since $a \neq 0$ this is equivalent to $ad - bc \neq 0$. Similarly, if $a = 0$, then either $d = 0$ and the matrix is singular, or if $d \neq 0$, one can

interchange rows and do the reduction. The same condition $ad-bc \neq 0$ for nonsingularity follows! This scalar quantity $ad-bc$ is the determinant of a general 2×2 matrix and we will see that it is nonzero if and only if the matrix is nonsingular. But first, we work on the general definition for the determinant of an $n \times n$ matrix.

Theorem 2.19. *Associated with every square matrix, there exists a uniquely defined scalar quantity, known as its* determinant *and denoted* $\det A$*, that obeys the following axioms:*

(1) Row operations of type 1 do not change the determinant.

(2) A row operation of type 2 changes the determinant by a factor of -1*.*

(3) A row operation of type 3 with a scalar γ *multiplies the determinant by the same factor* γ*. Note that* $\gamma = 0$ *is allowed here.*

(4) The determinant of an upper-triangular matrix U *is equal to the product of the diagonal entries:* $\det U = u_{11}u_{22}\cdots u_{nn}$*.*

We have some immediate consequences:

- $\det I = 1$
- Suppose that A has a row of all zeros, say row i. Then axiom (3) states that

$$A \xrightarrow{\gamma R_i \to R_i} \tilde{A},$$
$$\det A \xrightarrow{\gamma R_i \to R_i} \det \tilde{A} = \gamma \det A.$$

  But then since row i is a row of all zeros, $\tilde{A} = A$, and we choose $\gamma = 0$ to see that $\det A = 0$ if A has a row of all zeros!

- If $A = LU$ then $\det A = \det U$. This follows from axiom (1).
- If $PA = LU$ then $\det A = (-1)^k \det U$ where k is the number of row interchanges that are performed.
- $\det A = 0$ if and only if A is singular. This follows from Proposition 2.18.

We continue with some more involved properties.

Proposition 2.20. *Suppose* $A, B \in \mathbb{R}^{n\times}$ *then*

$$\det(AB) = \det A \det B.$$

Proof. First, suppose A is nonsingular. Then Gauss-Jordan elimination shows that we can express A as a product of elementary matrices of type 1,2 & 3, $A = E_k E_{k-1} \cdots E_1$.

We can then see that

$$\begin{aligned} I &\xrightarrow{R_i+aR_j\to R_i} E_1, \\ 1=\det I &\xrightarrow{R_i+aR_j\to R_i} \det E_1 = 1, \\ I &\xrightarrow{R_i\leftrightarrow R_j, j\neq i} \det E_2, \\ 1=\det I &\xrightarrow{R_i\leftrightarrow R_j, j\neq i} \det E_2 = -1, \\ I &\xrightarrow{aR_i\to R_i} \det E_3, \\ 1=\det I &\xrightarrow{aR_i\to R_i} \det E_3 = a. \end{aligned}$$

So if we perform the row operations associated with $E_1, E_2, \ldots, E_k$ to the identity matrix to form A, we see that

$$\det A = \det E_k \det E_{k-1} \cdots \det E_1.$$

Then if we consider AB, we can interpret it as applying the associated row operations that combine to create A, it follows that

$$\det(AB) = \det(E_k E_{k-1} \cdots E_1 B) = \det A \det B.$$

Proposition 2.21. *Suppose A is nonsingular, then* $\det A^{-1} = \dfrac{1}{\det A}$.

Now, to finally prove Theorem 2.7(1). If $A\mathbf{x} = \mathbf{b}$ can be solved for every choice then in reviewing (2.1), we see that in defining

$$X = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots x_n \end{bmatrix} \tag{2.2}$$

we have

$$AX = I \Rightarrow \det A \det X = 1.$$

This implies that $\det A \neq$ and therefore A is nonsingular.

We now illustrate the computation of determinants.

Example 2.22. Compute the determinant of

$$A = \begin{bmatrix} 1 & 4 & 1 \\ 2 & 12 & 1 \\ 1 & 2 & 4 \end{bmatrix}.$$

In reviewing the calculations in Section 2.6 (or in the previous chapter) we see that this matrix is reduced to upper triangular form

$$U = \begin{bmatrix} 1 & 4 & 1 \\ 0 & 4 & -1 \\ 0 & 0 & \frac{5}{2} \end{bmatrix},$$

using only row operations of type 1. Thus $\det A = \det U = (1)(4)(5/2) = 10$.

Example 2.23. Compute the determinant of

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

In reviewing the calculations in Section 2.1 we use row operations of type 1 and 2 to get to

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

Since we used two interchanges we find

$$\det A = (-1)^2 \det U = (1)(2)(3)(2) = 12.$$

Note that we did not use row operations of type 3 in these preceding examples. We can indeed do that, but often that complicates things because scaling used has to be accounted for.

Remark 2.24. *You may have encountered other methods of computing determinants. These methods actually require more computation! But they are typically easier to perform by hand when variables are present in the matrix under consideration. When we introduce eigenvalues we will need to introduce other methods to evaluate determinants.*

And now we encounter the first important use of the transpose of a matrix so we introduce it here:

Definition 2.25. *Let $A \in \mathbb{R}^{m\times n}$ then the transpose of A, denoted $A^T \in \mathbb{R}^{n\times m}$, is given by*

$$A^T = (a_{ji})_{\substack{1\leq i\leq n \\ 1\leq j\leq m}}.$$

Example 2.26. If

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

then

$$A^T = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

We now give some properties of the determinant.

Proposition 2.27. *Suppose $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{m\times p}$ then*

(1) $(AB)^T = B^T A^T$ *and*

(2) if $m = n$, $\det A = \det A^T$.

We can see that the second property follows from the first. Suppose A is nonsingular $PA = LU$. Set $B = PA$ and we then have that

$$\det B = \det U = u_{11} u_{22} \cdots u_{nn}.$$

Then $B^T = U^T L^T$. Define

$$D = \begin{bmatrix} u_{11} & & & \\ & u_{22} & & \\ & & \ddots & \\ & & & u_{nn} \end{bmatrix}.$$

It follows that $U^T D^{-1}$ is special lower triangular and DL^T upper triangular with $u_{11}, u_{22}, \ldots, u_{nn}$ on the diagonal. Thus

$$\det B^T = u_{11} u_{22} \cdots u_{nn} = \det B.$$

Lastly, we have $\det P \det A = \det B = \det B^T = \det A^T \det P^T$. Here, since P is a permutation matrix, P a product of some number, say k, of elementary permutation matrices:

$$P = P_1 P_2 \cdots P_k, \quad \det P_j = -1.$$

Then it is evident that if P_j is an elementary permutation matrix then so is P_j^T, and, in fact, $P_j^T = P_j$. Therefore $\det P = (-1)^j = \det P^T$ and we conclude that $\det A = \det A^T$. Now if $\det A^T = 0$, i.e., if A^T is singular, then so is A. Because if $\det A \neq 0$ the the argument just given would establish that $\det A^T \neq 0$!

One use of the transpose, as far as determinants go, is the following that applied to both upper- and lower-triangular matrices.

Corollary 2.28. *Suppose that* $A \in \mathbb{R}^{n \times n}$ *is a triangular (either upper or lower) matrix. Then* $\det A = a_{11} a_{22} \cdots a_{nn}$.

Chapter 3

Iterative methods

3.1 ▪ Simple scalar iteration

Suppose that $|m| < 1$ and consider the iteration

$$x_0 = 0$$
$$x_{k+1} = mx_k + c.$$

If $\lim_{k\to\infty} x_k$ exists, say, $\lim_{k\to\infty} x_k = x$ then

$$x = mx + c.$$

So, then $x = \frac{c}{1-m}$! Notice here that $|m| < 1$ is sufficient to ensure that this equation makes sense. Let's take a moment to show that this limit exists.

$$\begin{aligned}
x_0 &= 0 \\
x_1 &= c \\
x_2 &= mc + c \\
x_3 &= m^2c + mc + c \\
x_4 &= m^3c + m^2c + mb + c \\
&\vdots \\
x_k &= m^{k-1}c + m^{k-2}c + \cdots + mc + c.
\end{aligned}$$

Then

$$x_k = \frac{1-m^k}{1-m}c \overset{k\to\infty}{\longrightarrow} \frac{c}{1-m}.$$

3.2 ▪ Matrix iteration

We can attempt to do the same steps with a matrix M, and a vector $\mathbf{b}$. But what is an analogous condition to $|m| < 1$? Well, if $|m| < 1$ if and only if $m^k \to 0$ as $k \to \infty$. And that is exactly what is needed in the above proof.

Definition 3.1. *An $n \times n$ matrix M is* convergent *if*

$$\lim_{k\to\infty} (M^k)_{ij} = 0,$$

for all $1 \leq i, j \leq n$.

So, suppose that M is convergent and consider the iteration

$$\mathbf{x}^{(0)} = \mathbf{0},$$
$$\mathbf{x}^{(k+1)} = M\mathbf{x}^{(k)} + \mathbf{c}.$$

Again suppose that $\lim_{k\to\infty} \mathbf{x}^{(k)} = \mathbf{x}$ for any $\mathbf{b}$. Indeed, this will hold if the matrix is convergent. Then we have

$$\mathbf{x} = M\mathbf{x} + \mathbf{c}$$

or, equivalently,

$$(I - M)\mathbf{x} = \mathbf{c}.$$

So, we can then solve $(I-M)\mathbf{x} = \mathbf{c}$ for any choice of $\mathbf{c}$ and $(I-M)$ must be nonsingular. We summarize this in the following theorem.

Theorem 3.2. *If M is convergent then $\lim_{k\to\infty} \mathbf{x}^{(k)} = \mathbf{x}$ and $(I - M)\mathbf{x} = \mathbf{c}$.*

Proof. To come!

We now illustrate this.

```
1 n = 10;
2 M = .3*(rand(n,n)-.5);
3 A = eye(n) - M;
4 c = rand(n,1);
5 y = zeros(n,1);
6 format long
7
8 y = M*y+c
```

```
y =

   0.100750511921236
   0.507848830829537
   0.585609125701878
   0.762887095910741
   0.082962649110544
   0.661596193082714
   0.516979014706213
   0.171048017525447
   0.938557864331842
   0.590483177142572
```

```
y = M*y+c
```

```
y =

   0.077727154976476
   0.363806457179105
   0.557369511603846
   0.826806199241477
   0.255422901714552
   0.654633360078442
   0.700934934901354
   0.112068576471026
   1.177176246248708
   0.583426758737004
```

```
y = M*y+c
```

```
y =

   0.082842695914941
   0.294356706610472
   0.540812014758675
   0.870628718117166
   0.267063702748974
   0.615778654434824
   0.709714248726689
   0.127461627133065
   1.174512522775538
   0.615045985569920
```

```
y = M*y+c
```

```
y =

   0.079412489145411
   0.281314175491463
   0.541724244583188
   0.868548783529157
   0.269134760081902
   0.607213140925380
   0.711363265789150
   0.141891164485511
   1.165261731102694
   0.619361991943737
```

Then we can check how close we are to a solution:

```
A*y-c
```

```
ans =

   0.001218592337001
  -0.000229910292611
  -0.000121057628151
   0.002162997640271
  -0.000325319008234
   0.001295842952326
  -0.000896903506264
  -0.003470421629003
   0.006721034684748
   0.000220713676622
```

3.3 ▪ Jacobi's Algorithm

Consider the linear system

$$A\mathbf{x} = \mathbf{c}, \quad A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Since A has no non-zero entries off the diagonal we say that A is a diagonal matrix. We apply the simple iteration from the previous example, with disasterous results!

```
A = diag([1,2,3,4,5]);
M = eye(5) - A;
c = [1,1,1,1,1]';
y = zeros(5,1);
format long

for i = 1:5
    y = A*y + c
end
```

```
y =

     1
     1
     1
     1
     1


y =

     2
```

```
     3
     4
     5
     6


y =

     3
     7
    13
    21
    31


y =

     4
    15
    40
    85
   156


y =

     5
    31
   121
   341
   781
```

This is clearly not converging, but the solution is easily obtained by row operations of type 3:

$$\mathbf{x} = \begin{bmatrix} 1 \\ 1/2 \\ 1/3 \\ 1/4 \\ 1/5 \end{bmatrix}.$$

Any "good" algorithm should work on a nonsingular diagonal matrix and Jacobi's algorithm is a modification of the simple iteration that fixes this issue. To describe this algorithm, we decompose $A \in \mathbb{R}^{n\times n}$ is a simple manner:

$$A = L + D + U.$$

Here L is the strictly lower-triangular of A, D is just the diagonal of A and U is the strictly upper-triangular part of A. Note that here, L, U do not have any relation to the

LU factorization. Our goal is to solve

$$\begin{aligned} A\mathbf{x} &= \mathbf{b} \\ (L+D+U)\mathbf{x} &= \mathbf{b} \\ D^{-1}(L+D+U)\mathbf{x} &= D^{-1}\mathbf{b} \\ \mathbf{x} + D^{-1}(L+U)\mathbf{x} &= D^{-1}\mathbf{b}. \end{aligned}$$

This last equation is of the form:

$$(I-M)\mathbf{x} = \mathbf{c}, \quad M = -D^{-1}(L+U), \quad \mathbf{c} = D^{-1}\mathbf{b}.$$

Then Jacobi's algorithm is just the simple iteration applied to this equation:

$$\begin{aligned} \mathbf{x}^{(0)} &= \mathbf{0}, \\ \mathbf{x}^{(k+1)} &= -D^{-1}(L+U)\mathbf{x}^{(k)} + D^{-1}\mathbf{b}, \quad k = 0, 1, 2, \ldots. \end{aligned}$$

If we write out the entries of the iteration we see

$$x_i^{(k+1)} = \frac{1}{d_{ii}} \left(b_i - \sum_{j=1}^{i-1} \ell_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} u_{ij} x_j^{(k)} \right), \quad i = 1, \ldots, n.$$

Here we use the convention that $\sum_{j=i}^{k} = 0$ if $i > k$.

Algorithm 11: Jacobi's algorithm.

Result: An approximate solution of $A\mathbf{x} = \mathbf{b}$ or failure
Input: $A \in \mathbb{R}^{n\times n}$ with non-zero diagonal entries, $\mathbf{b} \in \mathbb{R}^n$, an error tolerance ϵ and a maximum number of steps $K > 0$
decompose $A = L + D + U$;
set $\mathbf{y} = \mathbf{0}$, $\mathbf{z} = \mathbf{0}$;
begin
  for $k = 1$ *to* K **do**
    for $i = 1$ *to* n **do**
      set $z_i = \frac{1}{d_{ii}} \left(b_i - \sum_{j=1}^{i-1} \ell_{ij} y_j - \sum_{j=i+1}^{n} u_{ij} y_j \right)$;
    end
    if $\max_{i=1}^{n} |y_i - z_i| < \epsilon$ **then**
      return $\mathbf{z}$;
    end
    set $\mathbf{y} = \mathbf{z}$;
  end
  return *Failed to converge*;
end

This algorithm might appear very complicated, but the MATLAB implementation is actually quite simple.

```
A = diag([1,2,3,4,5]) + .1*randn(5);
L = tril(A,-1);
U = triu(A,+1);
LpU = L + U; % L + U
D = diag(A);  % this is just a vector
b = [1,1,1,1,1]';
y = zeros(5,1);
```

and then one iteration is given by:

```
% compute y = -D^{-1}(L + U)y + D^{-1}b
y = LpU*y;  % does y = (L+U)*y
y = (b-y)./D % does y = D^{-1}(y- b)
```

```
y =

   1.121717334469232
   0.525299224393024
   0.348722648710539
   0.238094431031120
   0.203743873727282
```

The full algorithm is then given by

```
K = 20; eps = 1e-5;
y = zeros(5,1);
for i = 1:K
    z = LpU*y;  % does y = (L+U)*y
    z = (b-z)./D; % does y = D^{-1}(y- b)
    if max(abs(y-z)) < eps
        y = z
        break;
    end
    y = z;
end
i
y
```

```
i =

     6


y =

   1.205185617349236
   0.609134473234669
   0.335089112402153
   0.299768194236575
   0.200559560326893
```

We then check the error

```
A*y-b
```

```
ans =

   1.0e-05 *
```

```
 0.040172683357653
 0.084002826383767
-0.005611569531272
 0.127622150825069
-0.006015806019999
```

3.4 ▪ The Gauss-Seidel algorithm

We can improve on Jacobi's algorithm while still using the $A = L+D+U$ decomposition. Jacobi's algorithm leveraged the fact that we can easily invert a diagonal matrix, or rather, solving $D\mathbf{x} = \mathbf{b}$ when D is a diagonal matrix. But we know that back substitution allows us to solve $(U + D)\mathbf{x} = \mathbf{b}$ because $U + D$ is an upper-triangular matrix. Gauss-Seidel uses this.

$$\begin{aligned}\mathbf{x}^{(0)} &= \mathbf{0},\\ \mathbf{x}^{(k+1)} &= -(U+D)^{-1}L\mathbf{x}^{(k)} + (U+D)^{-1}\mathbf{b}, \quad k = 0, 1, 2, \ldots.\end{aligned}$$

If we write out the entries of the iteration, it looks very similar to Jacobi's algorithm but the iteration indices on the right-hand side are modified.

$$x_i^{(k+1)} = \frac{1}{d_{ii}}\left(b_i - \sum_{j=1}^{i-1} \ell_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} u_{ij}x_j^{(k+1)}\right), \quad i = n, n-1, \ldots, 1.$$

We again use the convention that $\sum_{j=i}^{k} = 0$ if $i > k$. Note that while the right-hand side depends on the iteration $k + 1$, which we are in the process of determining, it only depends on values that have been determined, provided that $i = n, n-1, n-2, \ldots$.

Algorithm 12: Gauss-Seidel algorithm.

Result: An approximate solution of $A\mathbf{x} = \mathbf{b}$ or failure
Input: $A \in \mathbb{R}^{n\times n}$ with non-zero diagonal entries, $\mathbf{b} \in \mathbb{R}^n$, an error tolerance ϵ and a maximum number of steps $K > 0$
decompose $A = L + D + U$;
set $\mathbf{y} = \mathbf{0}$, $\mathbf{z} = \mathbf{0}$;
begin
  for $k = 1$ *to* K **do**
    for $i = n$ *to* 1 *with steps of* -1 **do**
      set $\mathbf{z} = \mathbf{y}$;
      set $z_i = \frac{1}{d_{ii}}\left(b_i - \sum_{j=1}^{i-1} \ell_{ij}z_j - \sum_{j=i+1}^{n} u_{ij}z_j\right)$;
    end
    if $\max_{i=1}^{n} |y_i - z_i| < \epsilon$ **then**
      return $\mathbf{z}$;
    end
    set $\mathbf{y} = \mathbf{z}$;
  end
  return *Failed to converge*;
end

When we implement this in MATLAB we use matrix products and `Backsub` to simplify the calculations.

```
format long
A = diag([1,2,3,4,5]) + .1*randn(5);
UpD = triu(A);
L = tril(A,-1);
b = [1,1,1,1,1]';
y = zeros(5,1);
```

And one iteration is given by the following.

```
y = L*y;
y = Backsub([UpD,b-y])
```

```
y =

   1.013662166771282
   0.466425148512682
   0.333718136696037
   0.258290303921671
   0.202366035768882
```

The convergence is evident after just a couple of iterations.

```
y = L*y;
y = Backsub([UpD,b-y])
```

```
y =

   1.006409315749220
   0.435189514755413
   0.343010156594949
   0.238502700950548
   0.220848441572283
```

```
y = L*y;
y = Backsub([UpD,b-y])
```

```
y =

   1.006164937363233
   0.435365413217977
   0.342461609754539
   0.238489278211545
   0.221465576386166
```

```
A*y-b
```

```
ans =

   1.0e-04 *

                  0
  -0.153900509562988
  -0.067695517094180
   0.038155862203126
   0.086718538330199
```

3.4.1 ▪ Comparision with Jacobi's algorithm

We use the following code to compare the performance of Jacobi's algorithm to that of the Gauss-Seidel algorithm.

```
A = diag([1,2,3,4,5]) + .1*randn(5);
L = tril(A,-1);
U = triu(A,+1);
D = diag(A);
LpU = L + U; % L + U
UpD = triu(A);
b = [1,1,1,1,1]';

y_gs = zeros(5,1);
y_j = zeros(5,1);
x = A\b;  % solves Ax = b

T = 10;
err_gs = zeros(1,T);
err_j = zeros(1,T);

for i = 1:T
    y_j = LpU*y_j;
    y_j = (b-y_j)./D ;
    y_gs = L*y_gs;
    y_gs = Backsub([UpD,b-y_gs]);
    err_gs(i) = max(abs(y_gs-x));
    err_j(i) = max(abs(y_j-x));
end
hold off
semilogy(1:T,err_gs,'r')
hold on
grid on
semilogy(1:T,err_j,'b')
legend('Gauss-Seidel','Jacobi')
xlabel('Iteration number')
```

The output is displayed in Figure 3.1

3.5 ▪ Convergence

We have given examples when Jacobi's algorithm and the Gauss-Seidel algorithm work. But understanding when these algorithms work in general is important. Given $A \in \mathbb{R}^{n\times n}$ with non-zero diagonal entries decompose $A = L + D + U$ as before and define

$$M_{\mathrm{J}} = -D^{-1}(L+U), \quad M_{\mathrm{GS}} = -(D+U)^{-1}L.$$

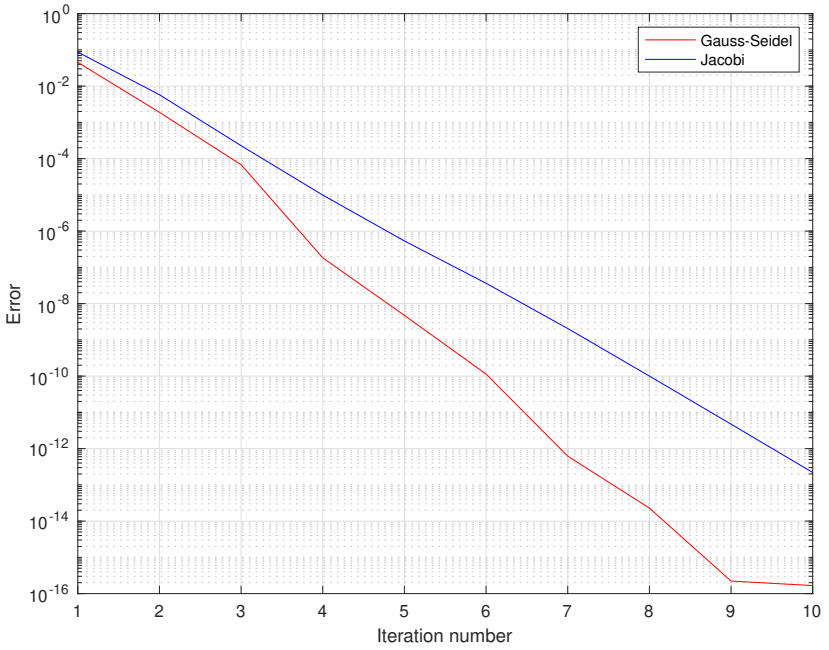


Figure 3.1: A comparision of the errors in produced by the Jacobi's algorithm and the Gauss-Seidel algorithm

It follows that if M_{J} is a convergent matrix then Jacobi's algorithm converges, and similarly, if M_{GS} is a convergent matrix then the Gauss-Seidel algorithm converges. But these conditions are really difficult to check! The next definition will help us give a condition that is simpler to check.

Definition 3.3. *$A \in \mathbb{R}^{n\times n}$ is said to be diagonally dominant if*

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, \quad \textit{for} \quad 1 \leq i \leq n.$$

A diagonally dominant matrix is a matrix whose diagonal entries, in absolute value, are larger than the sum of the absolute of the other entries in the row.

Example 3.4. The following matrix is diagonally dominant

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

while the following matrix is not

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

Theorem 3.5. *Jacobi's algorithm and the Gauss-Seidel algorithm converge if A is diagonally dominant.*

Note that the algorithm may still converge if the matrix is not diagonally dominant, but it is just not guaranteed by this theorem.

Chapter 4

Vector spaces

The set $\mathbb{R}^n$ of all vectors with n entries is an important example of a *vector space*. The abstract definition of a vector space is given below and it follows immediately that $\mathbb{R}^n$ is a vector space.

Definition 4.1. *A vector space (over $\mathbb{R}$) V is a set equipped with two operations*

- *Addition: for $\mathbf{v}, \mathbf{w} \in V$, $\mathbf{v} + \mathbf{w} \in V$,*
- *Scalar Multiplication: for $c, d \in \mathbb{R}$ and $\mathbf{v} \in V$, $c\mathbf{v} \in V$.*

These operations must satisfy the axioms we saw in Chapter 1:

- $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$ *(Associative, vector addition)*
- $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$ *(Commutative, vector addition)*
- $\mathbf{u} + \mathbf{0} = \mathbf{u}$ *(Additive identity)*
- $\mathbf{u} + (-\mathbf{u}) = \mathbf{0}$ *(Additive inverse)*
- $a(\mathbf{u} + \mathbf{v}) = (a\mathbf{u}) + (a\mathbf{v})$ *(Distributive, vector addition)*
- $(a + b)\mathbf{u} = a\mathbf{u} + b\mathbf{u}$ *(Distributive, scalar addition)*
- $(cd)\mathbf{u} = c(d\mathbf{u})$ *(Commutative, scalar multiplication)*
- $1\mathbf{u} = \mathbf{u}$ *(Multiplicative identity)*

Some immediate consequences:

- $0\mathbf{v} = \mathbf{0} \in V$,
- $c\mathbf{0} = \mathbf{0}$,
- $(-1)\mathbf{v} = -\mathbf{v}$, and
- if $c\mathbf{v} = \mathbf{0}$ then either $c = 0$, $\mathbf{v} = \mathbf{0}$ or both.

The following two statements give some motivation for this abstraction.

Definition 4.2. *Let $\mathbb{P}_n = \mathbb{P}_n(\mathbb{R})$ be the set of all polynomials in one variable of degree less than or equal to n with real coefficients, that is,*

$$\mathbb{P}_n = \{p : p(x) = c_0 + c_1x + c_2x^2 + \cdots + c_nx^n \text{ where } \mathbf{c} \in \mathbb{R}^{n+1}\}.$$

Proposition 4.3. *$\mathbb{P}_n$ is a vector space.*

Example 4.4. Consider $n = 2$, then $\mathbb{P}_2$ is the set of all polynomials of the form $p(x) = a_0 + a_1x + a_2x^2$. Let q be another polynomial of this form, $q(x) = b_0 + b_1x + b_2x^2$. Then

$$p(x) + q(x) = c_0 + c_1x + c_2x^2 = (a_0 + b_0) + (a_1 + b_1)x + (a_2 + b_2)x^2,$$
$$\mathbf{c} = \mathbf{a} + \mathbf{b} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}.$$

And also, for $d \in \mathbb{R}$,

$$d\,p(x) = c_0 + c_1x + c_2x^2 = (da_0) + (da_1)x + (da_2)x^2,$$
$$\mathbf{c} = d\mathbf{a} = d \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}.$$

The space $\mathbb{P}_n$ will be used extensively in Part II of these notes (whenever that gets written...).

The vector space we will consider almost exclusively is $\mathbb{R}^n$. But special subsets of $\mathbb{R}^n$ are also themselves vector spaces and we will want to introduce language to identify them.

Definition 4.5. *A* subspace *of a vector space V is a subset $W \subset V$ that is also a vector space, inheriting addition and scalar multiplication from V.*

Note that since W is itself a vector space then it must contain the zero element! The following gives us two things to check if we want to tell if a subset $W \subset V$ is indeed a subspace.

Theorem 4.6. *A non-empty subset $W \subset V$ of a vector space is a subspace if and only if*

- *for every $\mathbf{v}, \mathbf{w} \in W$, $\mathbf{v} + \mathbf{w} \in W$, and*
- *for every $\mathbf{v} \in W$ and every $c \in \mathbb{R}$, $c\mathbf{v} \in W$.*

Example 4.7. $\mathbb{R}^n$ is a vector space for any n.

Example 4.8. Consider

$$W = \left\{ \mathbf{v} \in \mathbb{R}^3 : v_1 + v_2 + v_3 = 0 \right\}.$$

This is the set of all vectors in $\mathbb{R}^3$ where the entries sum to zero. Then we claim that W is a subspace of $\mathbb{R}^3$. For $\mathbf{v}, \mathbf{w} \in W$

$$v_1 + v_2 + v_3 = 0, \quad w_1 + w_2 + w_3 = 0.$$

The vector $\mathbf{u} = \mathbf{v} + \mathbf{w}$ has entries

$$u_1 = v_1 + w_1, \quad u_2 = v_2 + w_3, \quad u_3 = v_3 + w_3.$$

Then note that

$$u_1 + u_2 + u_3 = \underbrace{(v_1 + v_2 + v_3)}_{=0} + \underbrace{(w_1 + w_2 + w_3)}_{=0} = 0.$$

This implies that $\mathbf{u} \in W$. Now consider $\mathbf{u} = c\mathbf{v}$ for $c \in \mathbb{R}$. Then $\mathbf{u}$ has entries

$$u_1 = cv_1, \quad u_2 = cv_2, \quad u_3 = cv_3$$

and

$$u_1 + u_2 + u_3 = c\underbrace{(v_1 + v_2 + v_3)}_{=0} = 0.$$

This implies that, again, $\mathbf{u} \in W$. By Proposition 4.6, W is a subspace of $\mathbb{R}^3$.

Example 4.9. Consider

$$W = \left\{ \mathbf{v} \in \mathbb{R}^3 : v_1 + v_2 + v_3 = 1 \right\}.$$

This is the set of all vectors in $\mathbb{R}^3$ where the entries sum to one. This is not a subspace because $\mathbf{0} \notin W$.

4.1 ▪ Span and linear independence

The vectors $\mathbf{e}_j \in \mathbb{R}^n$ are of critical importance because it is clear that any vector $\mathbf{v} \in \mathbb{R}^n$ can be written as the sum

$$\mathbf{v} = v_1\mathbf{e}_1 + v_2\mathbf{e}_2 + \cdots + v_n\mathbf{e}_n = \sum_{j=1}^{n} v_j\mathbf{e}_j.$$

When we simply speak about the "entries" in a vector in $\mathbb{R}^n$, we are implicitly using this fact! And these vectors have an additional property that is important. In $\mathbb{R}^2$, for example, we usually take $\mathbf{e}_1$ to be the vector that points in the direction of the positive x-axis and $\mathbf{e}_2$ to be the vector that points in the direction of the positive y-axis. So, $\mathbf{e}_1$ and $\mathbf{e}_2$ are perpendicular. This property will play a dominant role in the next chapter.

Finding an analogous construction for the subspace

$$W = \{\mathbf{v} \in \mathbb{R}^3 : v_1 + v_2 + v_3 = 0\},$$

is much less clear. We now work through definitions and consequences of those definitions to find such a construction.

Definition 4.10. *Let $\mathbf{v}_1, \dots, \mathbf{v}_k$ be elements of a vector space V. A sum of the form*

$$c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots c_k\mathbf{v}_k = \sum_{j=1}^{k} c_j\mathbf{v}_j,$$

where $c_j \in \mathbb{R}$ for every j is called a linear combination of the vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$.

Definition 4.11. *The set W of all linear combinations of vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_k\} \subset V$ is called the* span *of the vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ and is denoted*

$$W = \text{span}\{\mathbf{v}_1, \dots, \mathbf{v}_k\}.$$

Example 4.12. The span of the vectors $\{\mathbf{e}_2, \mathbf{e}_3\} \subset \mathbb{R}^3$ is given by

$$\{\mathbf{v} \in \mathbb{R}^3 : v_1 = 0\}.$$

4.2 ▪ Linear independence

It may happen that for some vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_k\} \subset V$ we have that

$$W = \text{span}\{\mathbf{v}_1, \dots, \mathbf{v}_k\} = \text{span}\{\mathbf{v}_1, \dots, \mathbf{v}_{k-1}\},$$

in which case we see that the addition of the vector $\mathbf{v}_k$ does not enrich the span, it is redundant. Then consider the (almost trivial) linear combination

$$\mathbf{w} = \mathbf{v}_k.$$

Then because $w \in \text{span}\{\mathbf{v}_1, \dots, \mathbf{v}_{k-1}\}$ we know that it must be expressible as a linear combination of these vectors (without v_k)

$$\mathbf{w} = \mathbf{v}_k = c_1\mathbf{v}_1 + \cdots + c_{k-1}\mathbf{v}_{k-1}.$$

So, we see that $\mathbf{v}_k$ can be written in terms of the other vectors! This can be rewritten as

$$c_1\mathbf{v}_1 + \cdots + c_{k-1}\mathbf{v}_{k-1} + c_k\mathbf{v}_k = \mathbf{0},$$

where $c_k = -1$. This is now directly related to the following definition.

Definition 4.13. *A set of vector space elements $\{\mathbf{v}_1, \dots, \mathbf{v}_k\} \subset V$ are said to be linearly dependent if there exists scalars $c_1, c_2, \dots, c_k$, at least one of which is nonzero, such that*

$$c_1\mathbf{v}_1 + \cdots + c_k\mathbf{v}_k = \mathbf{0}.$$

If such a set is not linearly dependent then the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are said to be linearly independent.

Example 4.14. The set $\{\mathbf{0}\} \subset \mathbb{R}^n$ is linearly dependent because $c_1\mathbf{0} = \mathbf{0}$ for any c_1. Furthermore, if $\mathbf{0}$ is in a set of vector space elements then that set is linearly dependent.

Example 4.15. The set

$$\left\{ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right\}$$

is linearly dependent because

$$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \mathbf{0}.$$

Example 4.16. The set

$$\left\{ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right\}$$

is linearly independent because the only way that

$$c_1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + c_2 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \mathbf{0}$$

is if $c_1 = c_2 = 0$.

We completed these two examples by inspection. But there is a general procedure for determining if a set of vectors is linearly dependent or independent. Consider the set

$$\left\{ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right\}.$$

Analyzing the linear dependence/independence amounts to understanding solutions of

$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \mathbf{0}.$$

This is called a *homogeneous linear system*. And when we consider the associated augmented matrix, we do not write in the column of zeros because row operations will never alter it. Reducing this matrix to upper-triangular form gives

$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \xrightarrow{-R_1+R_2 \to R_2} = \begin{bmatrix} \boxed{1} & 0 & 1 \\ 0 & \boxed{1} & -1 \\ 0 & 0 & 0 \end{bmatrix}.$$

The last equation reads $c_2 - c_3 = 0$, or $c_2 = c_3$. The first equation reads $c_1 = -c_3$. So, if we choose $c_3 = -1$, we find $c_1 = 1$, $c_2 = -1$ which is the linear combination used in Example 4.15. Because we could choose c_3 to be anything, we say that c_3 is a *free variable*. The formal definition of a free variable can be found in the next section.

4.3 ▪ Rectangular linear systems

To effectively determine if a collection of vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\} \subset \mathbb{R}^n$ is linearly dependent, or not, we will need to develop a deeper understanding of the reduction of rectangular matrices to upper-triangular form. This is something we have encountered with augmented matrices — both for just solving $A\mathbf{x} = \mathbf{b}$ and for computing A^{-1} using Gauss-Jordan elimination. But in those settings we always had more columns than rows. We can also have more rows than columns.

Definition 4.17. *A matrix $U \in \mathbb{R}^{m\times n}$ is said to be in* row echelon form *if it has the following structure*

$$U = \begin{bmatrix}
0 & \boxed{\times} & * & \cdots & * & * & \cdots & * & * & \cdots & \cdots & * & * & * & \cdots & * \\
0 & 0 & 0 & \cdots & 0 & \boxed{\times} & \cdots & * & * & \cdots & \cdots & * & * & * & \cdots & * \\
0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \boxed{\times} & \cdots & \cdots & * & * & * & \cdots & * \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & & \ddots & & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & \cdots & 0 & \boxed{\times} & * & \cdots & * \\
0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & \cdots & 0
\end{bmatrix}$$

Here all the $\boxed{\times}$ entries represent leading non-zero entries and are called pivots. *And $*$ represents arbitrary zero or non-zero entries. In mathematical terms this can be stated as:*

(1) Suppose u_{ij} is the first nonzero entry in row i and $u_{k\ell}$ is the first non-zero entry in row k, then $i < k$ implies $j < \ell$.

(2) If r is the number of non-zero rows of U, then the last $m - r$ rows of U must all be zero.

Suppose U is in row echelon form. Here are some immediate consequences:

- If first r rows of U are not identically zero, then each row must have a pivot.
- Not every column of U needs to have a pivot.

Example 4.18. The following matrix

$$U = \begin{bmatrix}
0 & \boxed{3} & 1 & 4 & -2 & 0 \\
0 & 0 & 0 & 0 & \boxed{2} & 1 \\
0 & 0 & 0 & 0 & 0 & \boxed{-3} \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

is in row echelon form. The boxed entries give the pivots.

Proposition 4.19. *Every matrix can be reduced to row echelon form by row operations of type 1 and 2.*

Definition 4.20. *Suppose the linear system $A\mathbf{x} = \mathbf{0}$ is reduced to row echelon form. Then:*

(1) If column j contains a pivot then we say that x_j is a pivot variable.

(2) If column j does not contain a pivot then we say that x_j is a free variable.

One more definition will be necessary.

Definition 4.21. *A linear system $A\mathbf{x} = \mathbf{b}$ is said to be* compatible *if it has at least one solution.*

The following is immediate and summarizes some observations made previously.

Theorem 4.22. *Suppose $A \in \mathbb{R}^{m\times n}$*

(1) The columns of A are linearly dependent if and only if there is a non-zero solution of $A\mathbf{x} = \mathbf{0}$.

(2) The columns of A are linearly independent if and only if the only solution of $A\mathbf{c} = \mathbf{0}$ is $\mathbf{x} = \mathbf{0}$

(3) A vector $\mathbf{b}$ lies in the span of the columns of A if and only $A\mathbf{x} = \mathbf{b}$ is compatible, i.e., it has at least one solution.

(4) Suppose the linear system $A\mathbf{x} = \mathbf{0}$ is reduced to row echelon form. Then the columns of A are linearly independent if and only if there are no free variables, i.e., the number of pivots is equal to the number of columns.

(5) Suppose the linear system $\left[\begin{array}{c|c} A & \mathbf{b} \end{array}\right]$ is reduced to row echelon form $\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right]$. Then the system is compatible if and only if every row either (1) contains a pivot of U or (2) is identically zero.

(6) The linear system $A\mathbf{x} = \mathbf{b}$ is compatible for every choice of vector $\mathbf{b}$ if and only if A has a pivot in every row.

Part (5) of the previous theorem helps us address the following question: What if we fewer equations than unknowns? Consider the system

$$\begin{aligned} x + y &= a \\ x - y &= b \\ 2x + y &= c. \end{aligned}$$

And suppose that a, b, c are given and we want to know if we can solve for x and y — but with three conditions and two degrees of freedom, we expect potential issues. We

convert to an agumented matrix and start performing row operations to put the matrix in row echelon form:

$$\left[\begin{array}{cc|c} 1 & 1 & a \\ 1 & -1 & b \\ 2 & 1 & c \end{array}\right] \xrightarrow{-R_1+R_2\to R_2} \left[\begin{array}{cc|c} 1 & 1 & a \\ 0 & -2 & b-a \\ 2 & 1 & c \end{array}\right]$$

$$\left[\begin{array}{cc|c} 1 & 1 & a \\ 0 & -2 & b-a \\ 2 & 1 & c \end{array}\right] \xrightarrow{-2R_1+R_3\to R_3} \left[\begin{array}{cc|c} 1 & 1 & a \\ 0 & -2 & b-a \\ 0 & -1 & c-2a \end{array}\right]$$

$$\left[\begin{array}{cc|c} 1 & 1 & a \\ 0 & -2 & b-a \\ 0 & -1 & c-2a \end{array}\right] \xrightarrow{-\frac{1}{2}R_2+R_3\to R_3} \left[\begin{array}{cc|c} 1 & 1 & a \\ 0 & -2 & b-a \\ 0 & 0 & c-\frac{1}{2}b-\frac{3}{2}a \end{array}\right]$$

The last row gives rise to the equation:

$$0x + 0y = c - \frac{1}{2}b - \frac{3}{2}a,$$

and if we hope to solve the system, we better have $0 = c - \frac{1}{2}b - \frac{3}{2}a$! Another way that one can interpret this is that if we do not have a pivot in every row of the coefficient matrix then we may not be able to solve for every choice of right-hand side vector.

In solving $A\mathbf{x} = \mathbf{b}$ we reduce $\left[\begin{array}{c|c} A & \mathbf{b} \end{array}\right]$ to row echelon form $\left[\begin{array}{c|c} U & \mathbf{c} \end{array}\right]$. We see that:

- Of the rows of U that are identically zero, the corresponding entries of $\mathbf{c}$ may also vanish, in which case the system is compatible. Otherwise the system in incompatible.
- If the system is compatible, then there may be free variables. If there are free variables then any choice of these gives a solution — infinite many solutions.
- If the system is compatible and there are no free variables then there is only one solution.

Theorem 4.23. *A system $A\mathbf{x} = \mathbf{b}$ of m linear equations in n unknowns ($A \in \mathbb{R}^{m\times n}$) has either (i) exactly one solution, (ii) infinitely many solutions, or (iii) no solution.*

4.3.1 ▪ Rank

Another important calculation is the following. Let $A \in \mathbb{R}^{m\times n}$. Suppose A is put into row echelon forms U and $\tilde{U}$ using different row operations of type 1 and 2. Then we know that we have two nonsingular matrices E and $\tilde{E}$, built out of elementary matrices, such that

$$EA = U, \quad \tilde{E}A = \tilde{U}.$$

Now suppose that U has more pivots that $\tilde{U}$, and specifically consider a situation like:

$$E\begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 & \mathbf{a}_4 & \mathbf{a}_5 & \mathbf{a}_6\end{bmatrix} = \begin{bmatrix} 0 & \boxed{3} & 1 & 4 & -2 & 0 \\ 0 & 0 & 0 & 0 & \boxed{2} & 1 \\ 0 & 0 & 0 & 0 & 0 & \boxed{-3} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

It then follows that

$$E\begin{bmatrix}\mathbf{a}_2 & \mathbf{a}_5 & \mathbf{a}_6\end{bmatrix} = \begin{bmatrix}\boxed{3} & -2 & 0\\ 0 & \boxed{2} & 1\\ 0 & 0 & \boxed{-3}\\ 0 & 0 & 0\end{bmatrix}.$$

Now, let's try to find a solution of

$$\begin{bmatrix}\mathbf{a}_2 & \mathbf{a}_5 & \mathbf{a}_6\end{bmatrix}\begin{bmatrix}c_1\\ c_2\\ c_3\end{bmatrix} = \mathbf{0}. \tag{4.1}$$

Multiplying through by E gives

$$\begin{bmatrix}\boxed{3} & -2 & 0\\ 0 & \boxed{2} & 1\\ 0 & 0 & \boxed{-3}\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}c_1\\ c_2\\ c_3\end{bmatrix} = \mathbf{0}.$$

Back substitution gives that $c_1 = c_2 = c_3$ and therefore $\{\mathbf{a}_2, \mathbf{a}_5, \mathbf{a}_6\}$ are linearly independent! Indeed, this also follows from Theorem 4.22.

Then since it is assumed that $\tilde{U}$ has fewer pivots

$$\tilde{E}\begin{bmatrix}\mathbf{a}_2 & \mathbf{a}_5 & \mathbf{a}_6\end{bmatrix} = \begin{bmatrix}* & * & *\\ * & * & *\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}.$$

But because we can have at most two pivots, Theorem 4.22 then implies that $\{\mathbf{a}_2, \mathbf{a}_5, \mathbf{a}_6\}$ are linearly dependent! This is a contradiction and the it must be the case that U and $\tilde{U}$ have the same number of pivots:

Proposition 4.24. *If square matrix $A \in \mathbb{R}^{n\times n}$ is reduced to row echelon form then the resulting matrix always has the same number of pivots.*

This proposition tells that the following is well-defined.

Definition 4.25. *The* rank *of a matrix is the number of pivots. Rank of A is denoted* rank A .

4.4 ▪ Basis and dimension

As we have seen, we can write any vector $\mathbf{v} \in \mathbb{R}^n$ as a linear combination of the vectors $\{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$:

$$\mathbf{v} = \sum_{j=1}^{n} v_j \mathbf{e}_j.$$

But how should we discuss such a decomposition for a subset $W \subset \mathbb{R}^n$? We use the notion of basis.

Definition 4.26. *A* basis *of a vector space V is a finite collection of elements $\{\mathbf{v}_1, \dots, \mathbf{v}_n\} \subset V$ that*

(a) spans V, and

(b) is linearly independent.

Example 4.27. The set $\{\mathbf{e}_1, \dots, \mathbf{e}_n\}$ is a basis for $\mathbb{R}^n$.

Theorem 4.28. *Every basis of $\mathbb{R}^n$ consists of exactly n vectors. Furthermore, a set of n vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_n\} \subset \mathbb{R}^n$ is a basis if and only if the $n \times n$ matrix*

$$V = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{bmatrix}$$

is nonsingular, i.e. $\operatorname{rank} A = n$.

This theorem is a special case of a more general fact that expands our results beyond $\mathbb{R}^n$.

Theorem 4.29. *Suppose a vector space V has a basis $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ for some integer n. Then every other basis of V consists of n vectors. In this case we write* $\dim V = n$, *i.e., V is of dimensions n.*

Proof. Suppose $\mathbf{w}_1, \dots, \mathbf{w}_m$ is another basis with $m > n$. A defining property of a basis is that it spans the vector space. This means that for each $i = 1, 2, \dots, m$

$$\mathbf{w}_i = \sum_{i=1}^{n} a_{ij} \mathbf{v}_i,$$

for some choice of coefficients a_{ij}. Then consider the linear combination

$$c_1\mathbf{w}_1 + c_2\mathbf{w}_2 + \cdots + c_m\mathbf{w}_m = \sum_{j=1}^{m}\sum_{i=1}^{n} c_j a_{ij} \mathbf{v}_i = \sum_{i=1}^{n} \left(\sum_{j=1}^{m} c_j a_{ij} \right) \mathbf{v}_i.$$

And we compare the sum $\sum_{j=1}^{m} c_j a_{ij}$ with Theorem 1.10. This tells us that solving

$$0 = \sum_{j=1}^{m} c_j a_{ij}, \quad i = 1, 2, \dots n, \tag{4.2}$$

is the same as

$$A\mathbf{c} = \mathbf{0}, \quad A = (a_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le m}}.$$

Since A has more columns than rows, it must have a column with no pivot, and therefore the must be free variables. So we can satisfy (4.2), implying that $\{\mathbf{w}_1, \dots, \mathbf{w}_m\}$ is linearly dependent. This shows that two bases cannot have a different number of vectors.

This theorem implies, and indeed it is a direct consequence of the proof, that if V has a basis of n vectors then every collection of m vectors where $m > n$ must be linearly dependent! The most important way to remember a basis is through the following theorem.

Theorem 4.30. *Suppose V is an n-dimensional vector space. Then*

(1) Every set of more than n elements of V is linearly dependent.

(2) No set of fewer then n elements spans V.

(3) A set of n elements forms a basis if and only if it spans V.

(4) A set of n elements forms a basis if and only if it is linearly independent.

Other consequences can be stated which are really a rephrasing of what has already been stated.

Proposition 4.31. *If $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ are elements of V and are linearly independent then $k \leq \dim V$.*

Proposition 4.32. *If $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ are elements of V and and span V then $k \geq \dim V$.*

The following is an important lemma that really gives the change of coordinates from a vector $\mathbf{x} \in V$ to a vector $\mathbf{c} \in \mathbb{R}^n$.

Lemma 4.33. *The elements $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ form a basis for V if and only if every $\mathbf{x} \in V$ can be written uniquely as a linear combination of the basis elements:*

$$\mathbf{x} = c_1\mathbf{v}_1 + \cdots + \mathbf{c}_k\mathbf{v}_k = \sum_{i=1}^{k} c_i\mathbf{v}_i.$$

Proof. Suppose that $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ form a basis for V. Then because this must span V we have that $\mathbf{x}$ can be written as a linear combination of the basis vectors. Suppose it can be written in two ways:

$$\sum_{i=1}^{k} d_i\mathbf{v}_i = \mathbf{x} = \sum_{i=1}^{k} c_i\mathbf{v}_i.$$

Then

$$\mathbf{0} = \sum_{i=1}^{k} (d_i - c_i)\mathbf{v}_i,$$

and linear independence of the basis implies that $d_i = c_i$ for each i. Now, conversely, suppose that $\mathbf{x}$ can be written uniquely as a linear combination of the vectors. This implies that $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ spans V. If the vectors were not linearly independent then the linear combination would not be unique. Indeed, if the vectors are not linearly

independent then

$$\mathbf{0} = \sum_{i=1}^{k} \hat{c}_i \mathbf{v}_i,$$

for some choice of c_i's, not all zero. Then

$$\mathbf{x} = \sum_{i=1}^{k} c_i \mathbf{v}_i \Rightarrow \mathbf{x} + \mathbf{0} = \sum_{i=1}^{k} (c_i + \hat{c}_i)\mathbf{v}_i,$$

showing that the linear combination cannot be unique.

4.5 ▪ The fundamental matrix subspaces

There are four fundamental subspaces associated to any matrix. But in this course we will only consider two of them, the kernel and the image.

Definition 4.34. *The* image *of $A \in \mathbb{R}^{m\times n}$ is the subspace* $\operatorname{img} A \subset \mathbb{R}^m$ *spanned by its columns. The* kernel *of A is the subspace* $\ker A \subset \mathbb{R}^n$ *defined by*

$$\ker A = \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} = \mathbf{0}\} \subset \mathbb{R}^n.$$

We do need to first confirm that $\ker A$ is indeed a subspace.

Proposition 4.35. *If* $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \subset \ker A$ *then*

$$c_1\mathbf{x}_1 + \cdots + c_k\mathbf{x}_k \in \ker A.$$

Proof. We just note because $A(\mathbf{x} + \mathbf{y}) = A\mathbf{x} + A\mathbf{y}$ and $A(c\mathbf{x}) = cA\mathbf{x}$

$$A(c_1\mathbf{x}_1 + \cdots + c_k\mathbf{x}_k) = c_1A\mathbf{x}_1 + \cdots + c_kA\mathbf{x}_k = c_1\mathbf{0} + \cdots + c_k\mathbf{0} = \mathbf{0}.$$

This proposition is enough to show that $\ker A$ is indeed a subspace of $\mathbb{R}^n$.

Example 4.36. Find a basis for $\ker A$:

$$A = \begin{bmatrix} 1 & -1 & 3 & 2 \\ 2 & 4 & 1 & 1 \\ 5 & 1 & 10 & 7 \end{bmatrix}.$$

Since the kernel of A consists of all solutions of $A\mathbf{x} = \mathbf{0}$ we need not augment A by anything and just put it in row echelon form.

$$\begin{bmatrix} 1 & -1 & 3 & 2 \\ 2 & 4 & 1 & 1 \\ 5 & 1 & 10 & 7 \end{bmatrix} \xrightarrow[-5R_1 + R_3 \to R_3]{-2R_1 + R_2 \to R_2} \begin{bmatrix} 1 & -1 & 3 & 2 \\ 0 & 6 & -5 & -3 \\ 0 & 6 & -5 & -3 \end{bmatrix}.$$

Then

$$\begin{bmatrix} 1 & -1 & 3 & 2 \\ 0 & 6 & -5 & -3 \\ 0 & 6 & -5 & -3 \end{bmatrix} \xrightarrow{-R_2+R_3\to R_3} \begin{bmatrix} 1 & -1 & 3 & 2 \\ 0 & 6 & -5 & -3 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Then we see that x_3, x_4 are free variables because columns 3 and 4 do not contain a pivot. The the system becomes,

$$6x_2 - 5x_2 - 3x_4 = 0 \implies x_2 = \frac{5}{6}x_3 + \frac{1}{2}x_4,$$

and

$$x_1 - x_2 + 3x_3 + 2x_4 \implies x_1 = x_2 - 3x_3 - 4x_4 = -\frac{13}{6}x_3 - \frac{7}{2}x_4.$$

Then the general solution is written solely in terms of the free variables

$$\mathbf{x} = \begin{bmatrix} -\frac{13}{6}x_3 - \frac{7}{2}x_4 \\ \frac{5}{6}x_3 + \frac{1}{2}x_4 \\ x_3 \\ x_4 \end{bmatrix} = x_3 \begin{bmatrix} -\frac{13}{6} \\ \frac{5}{6} \\ 1 \\ 0 \end{bmatrix} + x_4 \begin{bmatrix} -\frac{7}{2} \\ \frac{1}{2} \\ 0 \\ 1 \end{bmatrix}.$$

Then a basis for $\ker A$ is given by

$$\left\{ \begin{bmatrix} -\frac{13}{6} \\ \frac{5}{6} \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -\frac{7}{2} \\ \frac{1}{2} \\ 0 \\ 1 \end{bmatrix} \right\}.$$

The following proposition guarantees that the vectors found by the procedure in the previous exercise are always linearly independent.

Proposition 4.37. *Suppose $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\} \subset \mathbb{R}^n$ satisfy the following property: Each vector $\mathbf{v}_j$ has an non-zero entry such that all the other $\mathbf{v}_i$, $i \neq j$, vanish in that entry. Then $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ is linearly independent.*

Example 4.38. This proposition tells us that

$$\left\{ \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \right\}.$$

is linearly independent because (1) the first vector is non-zero in the first entry and the other are zero, (2) the second vector is non-zero in the second entry and the other are zero, and (3) the third vector is non-zero in the last entry and the others are zero.

We summarize the most important relationship between kernel and image in the following theorem.

Theorem 4.39. $\ker A = \{\mathbf{0}\}$ *if and only if* $A\mathbf{x} = \mathbf{b}$ *has a unique solution for every* $b \in \operatorname{img} A$.

But there are many equivalent characterizations.

Proposition 4.40. *Suppose* $A \in \mathbb{R}^{m\times n}$. *Then the following conditions are equivalent:*

(1) $\ker A = \{\mathbf{0}\}$, *i.e. the unique solution to* $A\mathbf{x} = \mathbf{0}$ *is* $\mathbf{x} = \mathbf{0}$.

(2) $\operatorname{rank} A = n$.

(3) The linear system $A\mathbf{x} = \mathbf{b}$ *has no free variables.*

(4) A has a pivot in every column.

(5) $A\mathbf{x} = \mathbf{b}$ *has a unique solution for every* $\mathbf{b} \in \operatorname{img} A$.

If, in addition, $m = n$ *then the following conditions are equivalent:*

(6) A is nonsingular.

(7) $\operatorname{rank} A = n$. *(pivot in every row, see Theorem 4.22(6) and Theorem 2.7(1))*

(8) $\ker A = \{\mathbf{0}\}$. *(pivot in every column, see Theorem 4.22(4))*

(9) $\operatorname{img} A = \mathbb{R}^n$. *(pivot in every row, see Theorem 4.22(6))*

It is instructive to see how Proposition 4.40(1) implies Proposition 4.40(5). Supposing Proposition 4.40(1) holds we take $b \in \operatorname{img} A$ which implies that we have at least one solution $\mathbf{x}_1$ of $A\mathbf{x}_1 = \mathbf{b}$. Then let $\mathbf{x}_2$ be another solution. So,

$$A\mathbf{x}_1 = \mathbf{b}, \quad A\mathbf{x}_2 = \mathbf{b}.$$

Then we find that

$$A(\mathbf{x}_1 - \mathbf{x}_2) = A\mathbf{x}_1 - A\mathbf{x}_2 = \mathbf{b} - \mathbf{b} = \mathbf{0}.$$

And then because $\mathbf{x}_1 - \mathbf{x}_2$ is a solution of $A\mathbf{x} = \mathbf{0}$, by Proposition 4.40(1), $\mathbf{x}_1 - \mathbf{x}_2 = \mathbf{0}$, or $\mathbf{x}_1 = \mathbf{x}_2$. So, the solution must be unique.

We end this chapter with a discussion of an important theorem and its implications for finding a basis of $\operatorname{img} A$.

Theorem 4.41 (Fundamental theorem of linear algebra). *Let* $A \in \mathbb{R}^{m\times n}$ *and let* r *be its rank. Then*

$$\dim \operatorname{img} A = \operatorname{rank} A = r \quad \textit{and} \quad \dim \ker A = n - r.$$

Example 4.42. Find a basis for $\operatorname{img} A$:

$$A = \begin{bmatrix} 1 & -1 & 3 & 2 \\ 2 & 4 & 1 & 1 \\ 5 & 1 & 10 & 7 \end{bmatrix}.$$

We saw that this matrix, when reduced to row echelon form is

$$A \to \begin{bmatrix} \boxed{1} & -1 & 3 & 2 \\ 0 & \boxed{6} & -5 & -3 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

This implies that $\dim \operatorname{img} A = 2$. Now, since we have a pivot in the first two columns, we claim that

$$\left\{ \begin{bmatrix} 1 \\ 2 \\ 5 \end{bmatrix}, \begin{bmatrix} -1 \\ 4 \\ 1 \end{bmatrix} \right\}$$

forms a basis for $\dim \operatorname{img} A$. Why? Because, by definition, these are two vectors in $\operatorname{img} A$ which we know is two-dimensional. Furthermore, when reducing the matrix

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

to row echelon form, we will find two pivots, implying linear independence — two linearly independent vectors in a two-dimensional subspace must be a basis!

Chapter 5

Inner products and normed vector spaces

In many applications, the linear system one encounters is not square. And Gaussian elimination, or even our iterative methods, do not work — typically — on such systems. In this course, our main use of *norms* and *inner products* is to make sense of generalized solutions of linear systems when there is (1) no true solution, or (2) too many (i.e., infinitely many) solutions.

5.1 ▪ Inner products and norms

The most basis inner product is the dot product: For $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$

$$\mathbf{v} \cdot \mathbf{w} = v_1 w_1 + v_2 w_2 + \cdots + v_n w_n.$$

Theorem 5.1. *On $\mathbb{R}^n$, the dot product obeys*

(1) Bilinearity: For $c, d \in \mathbb{R}$ and $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^n$

$$(c\mathbf{u} + d\mathbf{v}) \cdot \mathbf{w} = c(\mathbf{u} \cdot \mathbf{w}) + d(\mathbf{v} \cdot \mathbf{w}),$$
$$\mathbf{u} \cdot (c\mathbf{v} + d\mathbf{w}) = c(\mathbf{u} \cdot \mathbf{v}) + d(\mathbf{u} \cdot \mathbf{w}).$$

(2) Symmetry: For $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{u}.$$

(3) Positivity: For $\mathbf{v} \in \mathbb{R}^n$, $\mathbf{v} \cdot \mathbf{v} \geq 0$ and $\mathbf{v} \cdot \mathbf{v} = 0$ if and only if $\mathbf{v} = \mathbf{0}$.

The dot product produces a natural notion of length, called the 2-norm

$$\|\mathbf{v}\|_2 = \sqrt{\mathbf{v} \cdot \mathbf{v}} = \left(v_1^2 + v_2^2 + \cdots + v_n^2\right)^{1/2}.$$

But this is not the only notion of the length of a vector that one can use. For any $1 \leq p < \infty$ define the p-norm

$$\|\mathbf{v}\|_p = \left(|v_1|^p + |v_2|^p + \cdots + |v_n|^p\right)^{1/p}.$$

The most common cases are $p = 1, 2, \infty$ where

$$\|\mathbf{v}\|_\infty = \max_{1 \le i \le n} |v_i|,$$

gives the largest entry of the vector, in absolute value.

Other norms and inner products come up naturally throughout mathematics, physics, statistics and beyond. The textbook details this but we will large avoid any generalizations.

Definition 5.2. *A norm $\|\cdot\|$ on a vector space V is a scalar-valued function $\mathbf{v} \to \|\mathbf{v}\|$ that satisfies the following properties:*

(1) Positivity: For $\mathbf{v} \in V$, $\|\mathbf{v}\| \ge 0$ and $\|\mathbf{v}\| = 0$ if and only if $\mathbf{v} = \mathbf{0}$.

(2) Homogeneity: For $\mathbf{v} \in V$ and $c \in \mathbb{R}$, $\|c\mathbf{v}\| = |c|\|\mathbf{v}\|$.

(3) Triangle inequality: For $\mathbf{v}, \mathbf{w} \in V$, $\|\mathbf{v} + \mathbf{w}\| \le \|\mathbf{v}\| + \|\mathbf{w}\|$.

The norm $\|\mathbf{v}\|$ measures the size of a vector while the difference $\|\mathbf{v} - \mathbf{w}\|$ measures how close the two vectors are to each other. Alternatively, if we associate the vectors $\mathbf{v}$ and $\mathbf{w}$ with points in space, then $\|\mathbf{v} - \mathbf{w}\|$ gives the distance between these two points. In Figure 5.1 we demonstrate this. Matrices also have norms and we will encounter one

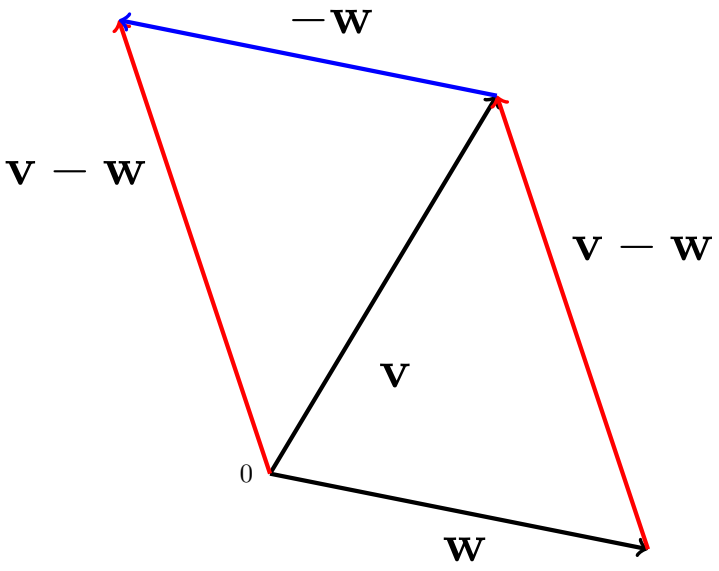


Figure 5.1: A demonstration that length of the vector $\mathbf{v} - \mathbf{w}$ gives the distance between the end points of $\mathbf{v}$ and $\mathbf{w}$.

such norm later in the course.

5.2 ▪ Solving general linear systems

When attempting to solve $A\mathbf{x} = \mathbf{b}$ when $A \in \mathbb{R}^{m \times n}$ we encounter three situations. We initially phrase them as contradictory situations and then describe how to make sense of

the problems.

(1) Solving $A\mathbf{x} = \mathbf{b}$ when there is no solution but there is a trivial kernel, i.e., $\operatorname{img} A \neq \mathbb{R}^m$ and $\ker A = \{\mathbf{0}\}$.

(2) Solving $A\mathbf{x} = \mathbf{b}$ when there are infinitely many solutions, i.e., $\mathbf{b} \in \operatorname{img} A$ but $\ker A \neq \{\mathbf{0}\}$.

(3) Solving $A\mathbf{x} = \mathbf{b}$ when $\mathbf{b} \notin \operatorname{img} A$ and $\ker A \neq \{\mathbf{0}\}$.

We will make sense of solving these problems in a particular way. It is important to note that there are many different ways to make sense of these problems. More specifically, we will use the 2-norm to make sense of these problems and one can, in principle, use any norm to make sense of things. The 2-norm makes the problems easier to solve but might not accomplish what one desires in a specific application. In the following sections we will discuss how to deal with (1), (2) and (3).

5.3 ▪ Solving $A\mathbf{x} = \mathbf{b}$ when there is no solution but $\ker A = \{\mathbf{0}\}$

Since we cannot find $\mathbf{x}$ such that $A\mathbf{x} = \mathbf{b}$, we can find $\mathbf{x}$ such that

$$\|A\mathbf{x} - \mathbf{b}\|_2$$

is as small as possible, i.e. $\|A\mathbf{x} - \mathbf{b}\|_2 < \|A\mathbf{y} - \mathbf{b}\|_2$ for $\mathbf{y} \neq \mathbf{x}$. One should also note that the assumptions made in this section imply that if $A \in \mathbb{R}^{m\times n}$ then $m > n$. If $n > m$ then there are more columns than rows and then there must be a column with out a pivot and therefore there is a free variable. This cannot happen, because we have assumed the kernel is trivial. Now if $n = m$, then the kernel being trivial implies that there is a pivot in every column and therefore in every row. But a pivot in every row implies that $\operatorname{img} A = \mathbb{R}^n$, and this contradicts our assumption that there is no solution of $A\mathbf{x} = \mathbf{b}$.

The following theorem shows us how to turn A (which is not square) into a nonsingular matrix and find the $\mathbf{x}$ that minimizes

$$\|A\mathbf{x} - \mathbf{b}\|_2$$

.

Theorem 5.3. *Suppose A is $m \times n$ and $\ker A = \{\mathbf{0}\}$. Then $A^TA \in \mathbb{R}^{n\times n}$ is nonsingular and if $\mathbf{x} = (A^TA)^{-1}A^T\mathbf{b}$ then*

$$\|A\mathbf{x} - \mathbf{b}\|_2 < \|A\mathbf{y} - \mathbf{b}\|_2,$$

for any $\mathbf{y} \neq \mathbf{x}$.

Proof. To show that A^TA is nonsingular it suffices to show that $\ker A^TA = \{\mathbf{0}\}$ as this establishes a pivot in every column. If $A^TA\mathbf{x} = \mathbf{0}$ then $\mathbf{x}^TA^TA\mathbf{x} = 0$. Then we note that $\mathbf{x}^TA^TA\mathbf{x} = (A\mathbf{x})^TA\mathbf{x} = \|A\mathbf{x}\|_2^2$. Since the assumption is that $\ker A = \{\mathbf{0}\}$, if $\|A\mathbf{x}\|_2 = 0$ then $\mathbf{x} = 0$ and $\ker A^TA = \{\mathbf{0}\}$.

Now, to understand the minimum, define

$$\mathcal{L}(\mathbf{y}) = \|A\mathbf{y} - \mathbf{b}\|_2^2.$$

By the definition of the 2-norm, and the bilinearity property of the dot product

$$\begin{aligned}\mathcal{L}(\mathbf{y}) &= (A\mathbf{y}-\mathbf{b})^T(A\mathbf{y}-\mathbf{b}),\\ &= \mathbf{y}^TA^TA\mathbf{y}-\mathbf{y}^TA^T\mathbf{b}-\mathbf{b}^TA\mathbf{y}+\mathbf{b}^T\mathbf{b}.\end{aligned}$$

To minimize this function we could compute its gradient and set it equal to zero. This is a completely valid approach. But here we will work with each term systematically. Let $\mathbf{y}=\mathbf{x}+\mathbf{z}$ where $\mathbf{x}=(A^TA)^{-1}A^T\mathbf{b}$. For the first term:

$$\begin{aligned}\mathbf{y}^TA^TA\mathbf{y} &= (\mathbf{x}+\mathbf{z})^TA^TA(\mathbf{x}+\mathbf{z})\\ &= (\mathbf{x}+\mathbf{z})^TA^TA((A^TA)^{-1}A^T\mathbf{b}+\mathbf{z})\\ &= (\mathbf{x}+\mathbf{z})^T(A^T\mathbf{b}+A^TA\mathbf{z})\\ &= \mathbf{x}^TA^T\mathbf{b}+\mathbf{z}^TA^T\mathbf{b}+\underbrace{\mathbf{x}^TA^TA}_{(A^TA\mathbf{x})^T}\mathbf{z}+\mathbf{z}^TA^TA\mathbf{z}\\ &= \mathbf{x}^TA^T\mathbf{b}+\mathbf{z}^TA^T\mathbf{b}+\mathbf{b}^TA\mathbf{z}+\mathbf{z}^TA^TA\mathbf{z}\end{aligned}$$

The for the middle two terms:

$$\begin{aligned}\mathbf{y}^TA^T\mathbf{b} &= (\mathbf{x}+\mathbf{z})^TA^T\mathbf{b} = \mathbf{x}^TA^T\mathbf{b}+\mathbf{z}^TA^T\mathbf{b},\\ \mathbf{b}^TA\mathbf{y} &= \mathbf{b}^TA(\mathbf{x}+\mathbf{z}) = \mathbf{b}^TA\mathbf{x}+\mathbf{b}^TA\mathbf{z}.\end{aligned}$$

Adding everything, there is a cancellation of three terms, giving

$$\mathcal{L}(\mathbf{y}) = \mathbf{z}^TA^TA\mathbf{z}+\mathbf{b}^T(\mathbf{b}-A\mathbf{x}).$$

Lastly, we note that $\mathbf{z}^TA^TA\mathbf{z} = \|A\mathbf{z}\|_2^2 \geq 0$ and is equal to zero if and only if $\mathbf{z}=0$. Thus $\mathbf{z}=0$ gives the global minimum, i.e. $\mathbf{y}=\mathbf{x}$.

This theorem states that the unique solution of

$$A^TA\mathbf{x} = A^T\mathbf{b},$$

is the unique minimizer of the function $\mathcal{L}(\mathbf{y}) = \|A\mathbf{y}-\mathbf{b}\|_2^2$.

Example 5.4. Find the least-square solution of

$$\begin{bmatrix}1 & 1\\ 0 & 1\\ 1 & 1\end{bmatrix}\mathbf{x} = \begin{bmatrix}1\\ 1\\ 0\end{bmatrix}.$$

First note that there is no solution! From this we need to compute the matrix A^TA and the right-hand side vector $A^T\mathbf{b}$:

$$A^TA = \begin{bmatrix}1 & 0 & 1\\ 1 & 1 & 1\end{bmatrix}\begin{bmatrix}1 & 1\\ 0 & 1\\ 1 & 1\end{bmatrix} = \begin{bmatrix}2 & 2\\ 2 & 3\end{bmatrix},$$

$$A^T\mathbf{b} = \begin{bmatrix}1 & 0 & 1\\ 1 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\ 1\\ 0\end{bmatrix} = \begin{bmatrix}1\\ 2\end{bmatrix}.$$

Then we need to solve

$$\left[\begin{array}{cc|c} 2 & 2 & 1\\ 2 & 3 & 2\end{array}\right] \Rightarrow \mathbf{x} = \begin{bmatrix}-\frac{1}{2}\\ 1\end{bmatrix}.$$

5.4 ▪ Orthogonality

In this section we will discuss how to use a generalize notion of two vectors being "perpendicular" to help us develop an efficient computation method to solve $A^TA\mathbf{x} = A^T\mathbf{b}$.

5.4.1 ▪ Block matrix notation

It will be convenient to divide a matrix into smaller submatrices and interpret the larger matrix as a matrix of matrices!

Example 5.5. Consider $A\mathbf{x}$ where $A \in \mathbb{R}^{4\times 4}$. We divide it up as

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ \hline a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right].$$

Then we partition $\mathbf{x}$ similarly

$$\mathbf{x} = \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ \hline x_4 \end{array}\right] = \left[\begin{array}{c} \mathbf{x}_1 \\ \hline \mathbf{x}_2 \end{array}\right].$$

And then

$$A\mathbf{x} = \left[\begin{array}{c} A_{11}\mathbf{x}_1 + A_{12}\mathbf{x}_2 \\ \hline A_{21}\mathbf{x}_1 + A_{22}\mathbf{x}_2 \end{array}\right].$$

We then use $\mathbf{0}$ to also stand for the matrix of all zeros whose dimension is inferred from context.

5.4.2 ▪ Orthogonal vectors

Definition 5.6. *Two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ are said to be orthogonal if $\mathbf{u} \cdot \mathbf{v} = 0$.*

Example 5.7. In Figure 5.2 we give two vectors $\mathbf{u} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ that are orthogonal:

$$\mathbf{u} \cdot \mathbf{v} = (1)(1) + (1)(-1) = 0.$$

Definition 5.8. *A square matrix Q is said to be orthogonal if $Q^TQ = I$.*

Below we will connect orthogonal matrices to the notion of othogonality for vectors.

We state some properties of orthogonal matrices.

Proposition 5.9. *Suppose $Q, U \in \mathbb{R}^{n\times n}$ are orthogonal. Then Q^T and QU are orthogonal matrices.*

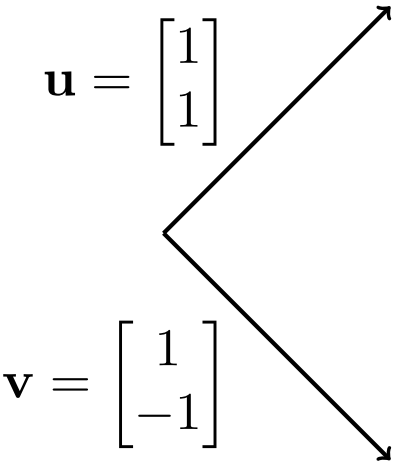


Figure 5.2: The vectors $\mathbf{u}$ and $\mathbf{v}$ as pictured are orthogonal.

A this point we can describe a (computationally) preferable way to solve the normal equations. Suppose we can find a factorization

$$\underbrace{A}_{m\times n} = \underbrace{Q}_{m\times m}\underbrace{\begin{bmatrix} \hat{R} \\ \hline \mathbf{0} \end{bmatrix}}_{m\times n}$$

where $\hat{R} \in \mathbb{R}^{n\times n}$ is upper-triangular with non-zero diagonal entries. Then we can compute

$$A^TA = \left[\begin{array}{c|c} \hat{R}^T & \mathbf{0} \end{array}\right] \underbrace{Q^TQ}_{I} \begin{bmatrix} \hat{R} \\ \hline \mathbf{0} \end{bmatrix} = \left[\begin{array}{c|c} \hat{R}^T & \mathbf{0} \end{array}\right] \begin{bmatrix} \hat{R} \\ \hline \mathbf{0} \end{bmatrix} = \hat{R}^T\hat{R}.$$

Then

$$A^T\mathbf{b} = \left[\begin{array}{c|c} \hat{R}^T & \mathbf{0} \end{array}\right] Q^T\mathbf{b} = \hat{R}^T \left[\begin{array}{c|c} I & \mathbf{0} \end{array}\right] Q^T\mathbf{b}.$$

With this notation, the normal equations become

$$\begin{aligned} A^TA\mathbf{x} &= A^T\mathbf{b} \\ \hat{R}^T\hat{R}\mathbf{x} &= \hat{R}^T \left[\begin{array}{c|c} I & \mathbf{0} \end{array}\right] Q^T\mathbf{b}. \end{aligned}$$

Then since $\hat{R}^T$ is nonsingular (it is lower-triangular with non-zero diagonal entries) we can multiply through by $(\hat{R}^T)^{-1}$ to find

$$\hat{R}\mathbf{x} = \left[\begin{array}{c|c} I & \mathbf{0} \end{array}\right] Q^T\mathbf{b}.$$

Provided we know Q and $\hat{R}$ we can then solve this with back substitution! In the next section we will discuss how to compute this factorization.

5.5 ▪ QR factorization

We begin with a definition.

Definition 5.10. *The QR factorization of a matrix $A \in \mathbb{R}^{m\times n}$, $\operatorname{rank} A = \min\{m, n\}$ is a factorization of the form*

$$A = QR, \quad Q^TQ = I, \quad R = \left[\begin{array}{c} \hat{R} \\ \hline \mathbf{0} \end{array}\right],$$

where $\hat{R}$ is $n \times n$ and upper triangular. Note that if $m = n$ then $R = \hat{R}$.

It is common to enforce that the diagonal entries of $\hat{R}$ should be positive. If $n = m$ this makes the decomposition unique. But we will not enforce positivity. Typically, we will have $m \geq n$, but the case $n = m + 1$ is reasonable to encounter when considering the augmented matrix $\left[\begin{array}{c|c} A & \mathbf{b} \end{array}\right]$.

Before we describe how to compute Q and $\hat{R}$ as in the previous section, we recall what happened with the LU factorization. For the LU factorization, supposing A is regular, we used elementary matrices, E_j to put A into upper-triangular form

$$A = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}, \quad E_1A = \begin{bmatrix} * & * & * \\ 0 & * & * \\ * & * & * \end{bmatrix},$$

$$E_2E_1A = \begin{bmatrix} \boxed{\times} & * & * \\ 0 & * & * \\ 0 & * & * \end{bmatrix}, \quad E_3E_2E_1A = \begin{bmatrix} \boxed{\times} & * & * \\ 0 & \boxed{\times} & * \\ 0 & 0 & \boxed{\times} \end{bmatrix}.$$

Where, as before $*$ represents arbitrary entries and $\times$ represents non-zero entries Then it turned out that because each of E_3, E_2, E_1 are special lower triangular

$$(E_3E_2E_1)^{-1} = E_1^{-1}E_2^{-1}E_3^{-1} = L$$

is also special lower triangular. A similar phenomenon will occur in reducing a matrix to upper-triangular form using orthogonal matrices. Suppose we can find Q_1, Q_2, Q_3 and Q_4 that act on A as follows

$$A = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}, \quad Q_1A = \begin{bmatrix} \boxed{\times} & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix},$$

$$Q_2Q_1A = \begin{bmatrix} \times & * & * & * \\ 0 & \times & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{bmatrix}, \quad Q_3Q_2Q_1A = \begin{bmatrix} \times & * & * & * \\ 0 & \times & * & * \\ 0 & 0 & \times & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & * \end{bmatrix},$$

$$Q_4Q_3Q_2Q_1A = \begin{bmatrix} \times & * & * & * \\ 0 & \times & * & * \\ 0 & 0 & \times & * \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{5.1}$$

Then we see that $(Q_4Q_3Q_2Q_1)^{-1} = Q_1^{-1}Q_2^{-1}Q_3^{-1}Q_4^{-1}$, but for an orthogonal matrix Q_j, because $Q_j^TQ_j = I$ we know that $Q_j^{-1} = Q_j^T$. This implies

$$Q = (Q_4Q_3Q_2Q_1)^{-1} = Q_1^TQ_2^TQ_3^TQ_4^T,$$

(which is easy to compute) so that $A = Q\begin{bmatrix}\hat{R}\\ \hline \mathbf{0}\end{bmatrix}$.

5.5.1 ▪ Computing the QR factorization

There are many ways to compute a QR decomposition. We use an approach that mirrors row reduction via elementary matrices.

Theorem 5.11. *Let $\mathbf{x} \in \mathbb{R}^n$. Define*

$$\mathbf{w} = \mathbf{x} + \begin{bmatrix}\operatorname{sign}(x_1)\|\mathbf{x}\|_2\\ 0\\ \vdots\\ 0\end{bmatrix}, \quad \operatorname{sign}(x) = \begin{cases}1 & x \geq 0,\\ -1 & x < 0.\end{cases}$$

Then $H = I - 2\frac{\mathbf{w}\mathbf{w}^T}{\|\mathbf{w}\|_2^2}$ satisfies

- *$H^TH = I$ (H is an orthogonal matrix),*
- $$H\mathbf{x} = \begin{bmatrix}-\operatorname{sign}(x_1)\|\mathbf{x}\|_2\\ 0\\ \vdots\\ 0\end{bmatrix}.$$

Definition 5.12. *The matrix H in the previous theorem is called a* Householder reflector *and we use $H(\mathbf{x})$ to refer to it. Also, if $\mathbf{x}$ happens to have only one entry (this will indeed come up), then $H(\mathbf{x}) = 1$. Lastly, define $H(\mathbf{0}) = I$.*

When we have correctly put the first few rows of a matrix into upper-triangular form we will not want to mess them up going forward. So, we need to develop a matrix that leaves the first row (or two, or three, etc.) alone while modifying the last rows.

Important

Consider a vector

$$\begin{bmatrix}a\\ x_1\\ \vdots\\ x_n\end{bmatrix} = \begin{bmatrix}a\\ \mathbf{x}\end{bmatrix}.$$

Then

$$\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & H(\mathbf{x}) \end{bmatrix} \begin{bmatrix} a \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} a \\ \pm\|\mathbf{x}\|_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Or, more generally, if we replace a with a vector $\mathbf{a}$,

$$\begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & H(\mathbf{x}) \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{a} \\ \pm\|\mathbf{x}\|_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

And

$$\begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & H(\mathbf{x}) \end{bmatrix}^T \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & H(\mathbf{x}) \end{bmatrix} = I.$$

From this discussion, it is now clear that we can preform the operations required to get a matrix into the form given in (5.1). We now give the QR factorization algorithm and we recall the submatrix indexing notation from Section 2.4.

Algorithm 13: QR Factorization `QR`

Result: An $m \times m$ orthogonal matrix Q and an $m \times n$ upper-triangular matrix R
Input: $A \in \mathbb{R}^{m \times n}$
begin
  set $Q = I$;
  set $R = A$;
  for $j = 1$ *to* n **do**
    set $H_0 = H(R_{j:m,j})$;
    set $R_{j:,j:} = H_0 R_{j:m,j:n}$;
    set $Q_{1:m,j:m} = Q_{1:m,j:m} H_0^T$;
  end
end

The algorithm is demonstrated now in MATLAB.

```
1  A = randn(5,3); R = A
```

```
R =

    0.5377   -1.3077   -1.3499
    1.8339   -0.4336    3.0349
   -2.2588    0.3426    0.7254
    0.8622    3.5784   -0.0631
    0.3188    2.7694    0.7147
```

The we proceed to introduce zeros in the first column.

```
x = R(1:end,1);
w = x; w(1) = x(1) + sign(x(1))*norm(x);
w = w/norm(w);
R(1:end,1:end) = R(1:end,1:end) - 2*w*(w'*R(1:end,1:end)) % using ...
    (1:end,1:end) is unnecessary
% but it helps to see the pattern
```

```
R =

   -3.0983   -0.5473   -1.0892
    0.0000   -0.0501    3.1664
   -0.0000   -0.1297    0.5635
    0.0000    3.7587   -0.0012
    0.0000    2.8361    0.7376
```

```
x = R(2:end,2);
w = x; w(1) = x(1) + sign(x(1))*norm(x);
w = w/norm(w);
R(2:end,2:end) = R(2:end,2:end) - 2*w*(w'*R(2:end,2:end))
```

```
R =

   -3.0983   -0.5473   -1.0892
    0.0000    4.7107    0.3939
   -0.0000    0.0000    0.4879
    0.0000   -0.0000    2.1877
    0.0000   -0.0000    2.3892
```

```
x = R(3:end,3);
w = x; w(1) = x(1) + sign(x(1))*norm(x);
w = w/norm(w);
R(3:end,3:end) = R(3:end,3:end) - 2*w*(w'*R(3:end,3:end))
```

```
R =

   -3.0983   -0.5473   -1.0892
    0.0000    4.7107    0.3939
   -0.0000    0.0000   -3.2760
    0.0000   -0.0000    0.0000
    0.0000   -0.0000    0.0000
```

The actual implementation of this algorithm is different because there is no reason to ever calculate the matrix H_0 at each step! To see why, consider the number of FLOPs required to compute the following for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

$$H_0 = H(\mathbf{x}), \quad H_0\mathbf{y}.$$

We enumerate all the components:

(1) Compute $\|\mathbf{x}\|_2$: $2n$ FLOPs (suppose the square root costs one FLOP).

(2) Compute $\mathbf{w}$: 1 FLOP.

(3) We note that $2\frac{\mathbf{w}\mathbf{w}^T}{\|\mathbf{w}\|_2^2} = \mathbf{v}\mathbf{v}^T$, $\mathbf{v} = \frac{\sqrt{2}}{\|\mathbf{w}\|_2}\mathbf{w}$, so to compute $\mathbf{v}$: $n+5$ FLOPs (this can be accomplished reusing some computation that is used to compute $\|\mathbf{x}\|_2$.

(4) Compute $\mathbf{v}\mathbf{v}^T$: n^2 FLOPs

(5) Compute $H_0 = I - \mathbf{v}\mathbf{v}^T$: n FLOPs

(6) Compute $H_0\mathbf{y}$: $n^2 - n$ FLOPs

All together this results in $2n^2 + 3n + 6$ FLOPs. If we decide not to compute H_0 we still have to complete steps (1)-(4) but can finish by directly computing $H_0\mathbf{y}$:

(5) Compute $H_0\mathbf{y} = \mathbf{y} - \mathbf{v} \underbrace{(\mathbf{v}^T\mathbf{y})}_{2n-1 \text{ FLOPs}}$: $4n - 1$ FLOPS

The final count is then $n^2 + 8n + 5$ FLOPS. As n increases, this is roughly half as many FLOPS that are needed if H_0 is constructed!

5.5.2 ▪ The QR factorization algorithm for the normal equations

The QR factorization algorithm has very similar properties to the LU factorization. Recall that LU, or more generally, $PA = LU$, is Gaussian elimination with bookkeeping. But if we want to solve $A\mathbf{x} = \mathbf{b}$ for one choice of right-hand side vector $\mathbf{b}$, we augment $\left[\begin{array}{c|c} A & \mathbf{b} \end{array}\right]$ and do Gaussian elimination, ignoring what L is. We can do the same with the QR factorization and we consider this schematically using the same notation that led to (5.1). Suppose

$$
\left[\begin{array}{c|c} A & \mathbf{b} \end{array}\right] = \left[\begin{array}{cccc|c} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{array}\right] \xrightarrow{Q_1} \left[\begin{array}{cccc|c} \times & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{array}\right]
$$

$$
\left[\begin{array}{cccc|c} \times & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{array}\right] \xrightarrow{Q_2} \left[\begin{array}{cccc|c} \times & * & * & * & * \\ 0 & \times & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{array}\right]
$$

$$
\left[\begin{array}{cccc|c} \times & * & * & * & * \\ 0 & \times & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{array}\right] \xrightarrow{Q_3} \left[\begin{array}{cccc|c} \times & * & * & * & * \\ 0 & \times & * & * & * \\ 0 & 0 & \times & * & * \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & * & * \end{array}\right]
$$

$$
\left[\begin{array}{cccc|c} \times & * & * & * & * \\ 0 & \times & * & * & * \\ 0 & 0 & \times & * & * \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & * & * \end{array}\right] \xrightarrow{Q_4} \left[\begin{array}{cccc|c} \times & * & * & * & * \\ 0 & \times & * & * & * \\ 0 & 0 & \times & * & * \\ 0 & 0 & 0 & \times & * \\ 0 & 0 & 0 & 0 & * \end{array}\right]
$$

Here $\xrightarrow{Q_j}$ is giving how the augmented matrix transforms after mulitiplication by Q_j. Now, let us look at this last matrix more closely, and divide it into block form

$$\left[\begin{array}{cccc|c} \times & * & * & * & * \\ 0 & \times & * & * & * \\ 0 & 0 & \times & * & * \\ 0 & 0 & 0 & \times & * \\ \hline 0 & 0 & 0 & 0 & * \end{array}\right] = \left[\begin{array}{c|c} \hat{R} & \mathbf{c} \\ \hline \mathbf{0} & \mathbf{r} \end{array}\right].$$

Now. $\hat{R}\mathbf{x} = \mathbf{c}$ can be solved with back substitution giving the solution of $A^TA\mathbf{x} = \mathbf{A}^T\mathbf{b}$. But what is happening with the last row? It turns out that $\|\mathbf{r}\|_2 = \|A\mathbf{x} - \mathbf{b}\|_2$, the 2-norm of what we throw away on the right-hand side is a measure of how close we are to finding a true solution — instead of a least-squares solution. On a computer, this is much forming A^TA, A^Tb and applying Gaussian elimination.

To summarize: Suppose $A \in \mathbb{R}^{m\times n}$, $m \geq n$, form $\left[\begin{array}{c|c} A & \mathbf{b} \end{array}\right]$, reduce it to upper-triangular form using Householder reflectors. Take the first n rows and solve using back substitution.

5.6 ▪ Gradient descent for the normal equations

We now describe an iterative method for solving $A^TA\mathbf{x} = A^T\mathbf{b}$ that uses the fact that $\mathbf{x}$ is the minimizer of the function

$$\mathcal{L}(\mathbf{y}) = \|A\mathbf{y} - \mathbf{b}\|_2^2.$$

Since $\mathcal{L}$ is a function of n variables (when $A \in \mathbb{R}^{m\times n}$) and it is difficult to visualize, we begin with an example with two variables.

Example 5.13. Consider $f : \mathbb{R}^2 \to \mathbb{R}$, say $f(x, y) = \frac{3}{2}x^2 + (y + 1)^2$. It is clear that this function is the sum of square and can therefore never take a negative value. So the global minimum occurs at $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$. But suppose we do not know this, and we guess a point, say $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$ and we want to figure out how to update our position. The gradient of f

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f}{\partial x}(x, y) \\ \frac{\partial f}{\partial y}(x, y) \end{bmatrix} = \begin{bmatrix} 3x \\ 2(y + 1) \end{bmatrix},$$

points in the direction of the greatest increase f. So, we want to move in the opposite direction, see Figure 5.3.

Our goal is to minimize

$$\mathcal{L}(\mathbf{y})\|A\mathbf{y} - \mathbf{b}\|_2^2 = (A\mathbf{y} - \mathbf{b})^T(A\mathbf{y} - \mathbf{b}),$$

starting from an initial point $\mathbf{y} = \mathbf{x}_0$, taking steps in the opposite direction of the gradient.

Proposition 5.14.

$$\nabla\mathcal{L}(\mathbf{y}) = 2(A^TA\mathbf{y} - A^T\mathbf{b}).$$

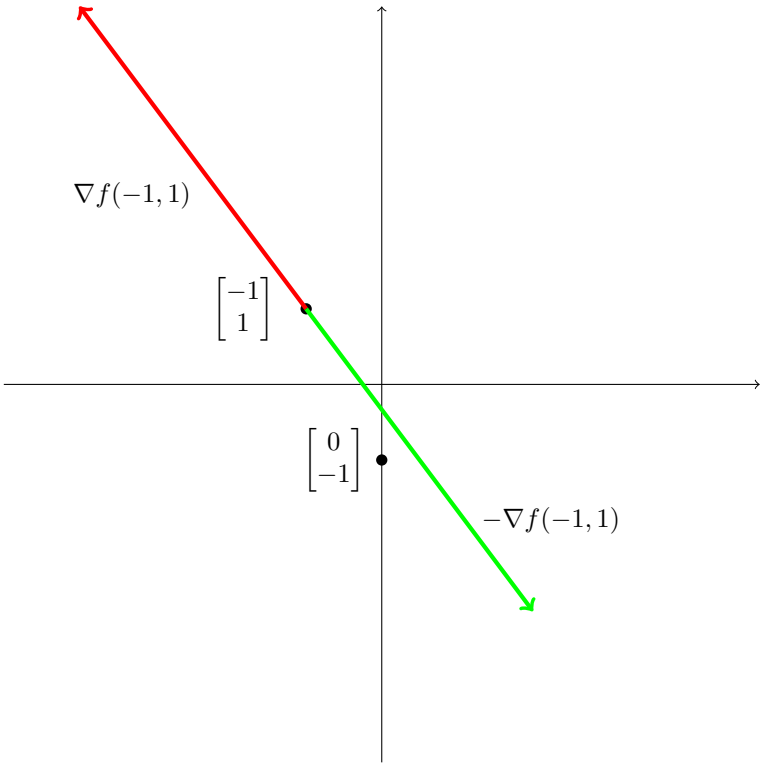


Figure 5.3: A demonstration that moving in the direction of $-\nabla f(-1,1) = \begin{bmatrix}3\\-4\end{bmatrix}$ moves the point $\begin{bmatrix}-1\\1\end{bmatrix}$ in the general direction of the global minimum at $\begin{bmatrix}0\\-1\end{bmatrix}$.

Proof. It follows that

$$\frac{\partial}{\partial y_j}(A\mathbf{y}-\mathbf{b}) = A\mathbf{e}_j.$$

The product rule then implies that

$$\begin{aligned}\frac{\partial}{\partial y_j}\mathcal{L}(\mathbf{y}) &= (A\mathbf{e}_j)^T(A\mathbf{y}-\mathbf{b}) + (A\mathbf{y}-\mathbf{b})^T A\mathbf{e}_j \\ &= 2(A\mathbf{e}_j)^T(A\mathbf{y}-\mathbf{b}).\end{aligned}$$

Here we used that $\mathbf{v}^T\mathbf{u} = \mathbf{u}^T\mathbf{v}$. Then

$$\frac{\partial}{\partial y_j}\mathcal{L}(\mathbf{y}) = \mathbf{e}_j^T 2(A^TA\mathbf{y} - A^T\mathbf{b}).$$

Then

$$\nabla\mathcal{L}(\mathbf{y}) = \begin{bmatrix} \frac{\partial}{\partial y_1}\mathcal{L}(\mathbf{y}) \\ \vdots \\ \frac{\partial}{\partial y_n}\mathcal{L}(\mathbf{y}) \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1^T 2(A^TA\mathbf{y} - A^T\mathbf{b}) \\ \vdots \\ \mathbf{e}_n^T 2(A^TA\mathbf{y} - A^T\mathbf{b}) \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \\ \vdots \\ \mathbf{e}_n^T \end{bmatrix}}_{I} 2(A^TA\mathbf{y} - A^T\mathbf{b})$$

$$= 2(A^TA\mathbf{y} - A^T\mathbf{b}).$$

We now give the gradient descent algorithm with step size h_k:

Algorithm 14: Gradient descent algorithm `GD` with step size h_k

Result: An approximation of a minimum of $\mathcal{L}(\mathbf{y}) = \|A\mathbf{y} - \mathbf{b}\|_2^2$
Input: $A \in \mathbb{R}^{m\times n}$, $m \geq n$, $b \in \mathbb{R}^m$, an initial guess $\mathbf{x}_0 \in \mathbb{R}^n$, a tolerance ϵ and a maximum iteration count K

```
begin
    for k = 1 to K do
        set x_k = x_{k-1} − (h_k/2) ∇L(x_{k-1})  (x_k = x_{k-1} − h_k(A^T A x_{k-1} − A^T b));
        if ‖x_k − x_{k-1}‖_2 < ε then
            return x_k;
        end
    end
end
```

This algorithm leaves open the question of what to choose for h_k. We can choose h_k in an appropriately optimal way. We start with the equation that gives the iteration

$$\mathbf{x}_k = \mathbf{x}_{k-1} - h_k(A^TA\mathbf{x}_{k-1} - A^T\mathbf{b}).$$

Let $\mathbf{x}$ be the true minimizer of $\mathcal{L}(\mathbf{y})$, i.e. $A^TA\mathbf{x} = A^T\mathbf{b}$. Subtracting from both sides,

$$\mathbf{x} - \mathbf{x}_k = \mathbf{x} - \mathbf{x}_{k-1} - h_k(A^TA\mathbf{x}_{k-1} - A^T\mathbf{b}).$$

If we define $\mathbf{r}_k = \mathbf{x} - \mathbf{x}_k$ we find that this whole iteration can be written in terms of $\mathbf{r}_k$:

$$\mathbf{r}_k = \mathbf{r}_{k-1} - h_kA^TA\mathbf{r}_{k-1}.$$

It is natural to choose h_k to minimize a norm of $\mathbf{r}_k$. If we tried to minimize the 2-norm we would consider

$$\begin{aligned}\mathbf{r}_k^T\mathbf{r}_k &= (\mathbf{r}_{k-1} - h_kA^TA\mathbf{r}_{k-1})^T(\mathbf{r}_{k-1} - h_kA^TA\mathbf{r}_{k-1}) \\ &= \mathbf{r}_{k-1}^T\mathbf{r}_{k-1} - 2h_k\mathbf{r}_{k-1}A^TA\mathbf{r}_k + h_k^2\mathbf{r}_{k-1}A^TAA^TA\mathbf{r}_{k-1}.\end{aligned}$$

But, this, while complicated, as a function of h_k is just a quadratic! So, we know the minimizer is found when

$$h_k = \frac{\mathbf{r}_{k-1}A^TA\mathbf{r}_{k-1}}{\mathbf{r}_{k-1}A^TAA^TA\mathbf{r}_{k-1}}.$$

But there is a critical problem here! We do not know $\mathbf{r}_k = \mathbf{x} - \mathbf{x}_k$ because we do not know $\mathbf{x}$. We do, however, know that $A^TA\mathbf{x} = A^T\mathbf{b}$. We return to the equation for $\mathbf{r}_k$, and multiply through by A and then minimize

$$\begin{aligned} A\mathbf{r}_k &= A\mathbf{r}_{k-1} - h_k AA^TA\mathbf{r}_{k-1}, \\ (A\mathbf{r}_k)^T(A\mathbf{r}_k) &= (A\mathbf{r}_{k-1} - h_k AA^TA\mathbf{r}_{k-1})^T(A\mathbf{r}_{k-1} - h_k AA^TA\mathbf{r}_{k-1}). \end{aligned} \tag{5.2}$$

Choosing h_k to minimize (5.2) gives

$$h_k = \frac{(A^TA\mathbf{r}_{k-1})^T(A^TA\mathbf{r}_{k-1})}{(AA^TA\mathbf{r}_{k-1})^T(AA^TA\mathbf{r}_{k-1})}.$$

But this needs to be rewritten to make it reasonable to work with. We see that

$$A^TA\mathbf{r}_{k-1} = A^T\mathbf{b} - A^TA\mathbf{x}_{k-1},$$

so, define $\mathbf{y}_k = A^T\mathbf{b} - A^TA\mathbf{x}_{k-1}$, and we discover

$$h_k = \frac{\mathbf{y}_k^T\mathbf{y}_k}{(A\mathbf{y}_k)^T(A\mathbf{y}_k)}.$$

We are led to a refinement of Algorithm 15.

Algorithm 15: Gradient descent algorithm `GDopt` with optimal step size h_k

Result: An approximation of a minimum of $\mathcal{L}(\mathbf{y}) = \|A\mathbf{y} - \mathbf{b}\|_2^2$
Input: $A \in \mathbb{R}^{m\times n}$, $m \geq n$, $b \in \mathbb{R}^m$, an initial guess $\mathbf{x}_0 \in \mathbb{R}^n$, a tolerance ϵ and a maximum iteration count K

```
begin
    for k = 1 to K do
        set y_k = A^T(b − Ax_{k−1});
        set h_k = (y_k^T y_k) / ((Ay_k)^T (Ay_k));
        set x_k = x_{k−1} + h_k y_k;
        if ‖h_k y_k‖_2 < ε then
            return x_k;
        end
    end
end
```

We do want to point out that this algorithm is rarely used to approximate solutions of the least-squares problem, but, it is an important algorithm that generalizes beyond functions that have the same form as $\mathcal{L}$. For such problems outside the realm of least squares, finding an optimal choice for h_k is elusive.

We now demonstrate the gradient descent algorithm with optimal step size in MATLAB. We first set up a least squares system in two variables:

```
A = randn(8,2)+1.4
b = 20*ones(8,1)
L = @(y1,y2)  norm(A*[y1;y2]-b)^2;
```

```
A =
```

```
    0.2529   -0.3115
    0.3311    1.2978
    0.5905    1.1586
   -1.5443    1.7192
    2.8384    1.7129
    1.7252    0.5351
    0.6451    1.3699
    2.7703    1.2351


b =

    20
    20
    20
    20
    20
    20
    20
    20
```

The function `L(y1,y2)` is the function we are seeking to minimize, i.e. `L`$= \mathcal{L}$. The following code plots the level curves of $\mathcal{L}$ along with an initial guess for the minimum. The resulting plot is shown in Figure 5.4a.

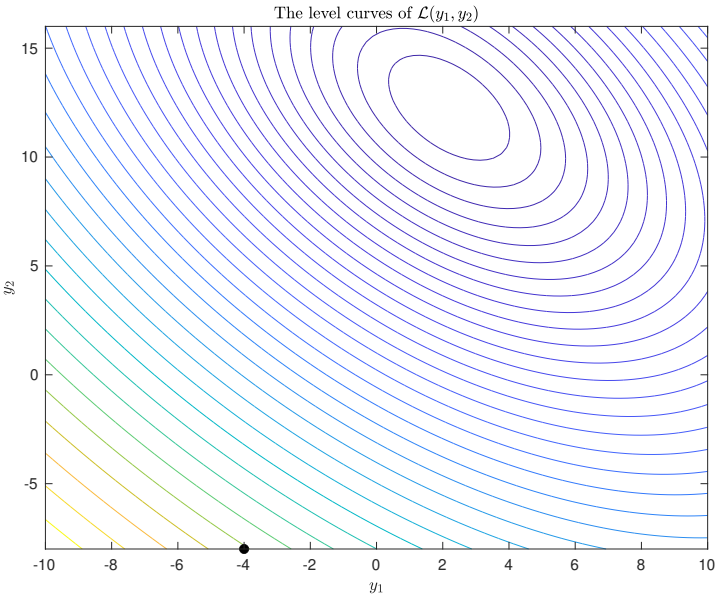
```
Y1 = linspace(-10,10,100);
Y2 = linspace(-8,16,100);
[Y1,Y2] = meshgrid(Y1,Y2);
Z = Y1;
for i = 1:size(Y1,1)
    for j = 1:size(Y1,2)
        Z(i,j) = L(Y1(i,j),Y2(i,j));
    end
end
low = norm(A*(A\b)-b)^2;
curves = exp(linspace(log(low)-.1,log(low)+5,60));

x = [-4;-8]; % initial guess
x1s = [x(1)];
x2s = [x(2)];
hold off
contour(Y1,Y2,Z,curves)
title('The level curves of $\mathcal L(y_1,y_2)$','Interpreter','latex')
xlabel('$y_1$','Interpreter','latex')
ylabel('$y_2$','Interpreter','latex')
hold on
plot(x1s,x2s,'ko','MarkerFaceColor',[0,0,0])
```

Then we perform five iterations of gradient descent:

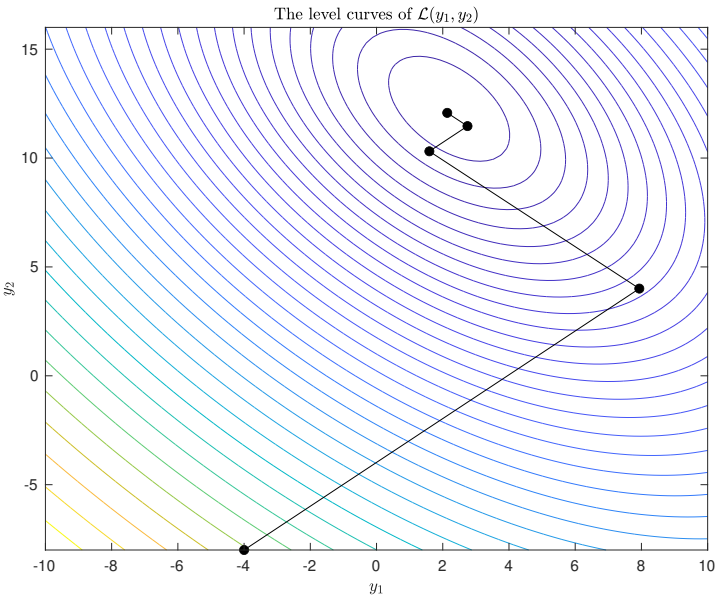
```
x = [-4;-8]; % initial guess
x1s = [x(1)];
x2s = [x(2)];
hold off
contour(Y1,Y2,Z,curves)
```

```
title('The level curves of $\mathcal L(y_1,y_2)$','Interpreter','latex')
xlabel('$y_1$','Interpreter','latex')
ylabel('$y_2$','Interpreter','latex')
hold on

for i = 1:4
    y = A'*(b - A*x);
    v = A*y;
    h = (y'*y)/(v'*v);
    x = x + h*y;
    x1s = [x1s,x(1)];
    x2s = [x2s,x(2)];
end
plot(x1s,x2s,'ko-','MarkerFaceColor',[0,0,0])
```

The resulting trajectory is show in Figure 5.4b.



(a) The level curves of $\mathcal{L}$ along with an initial guess.



(b) The first for iterations of gradient descent with the optimal step size.

Chapter 6

Tikhonov regularization

We now discuss solving $A\mathbf{x} = \mathbf{b}$ when there are infinitely many solutions. The fundamental question here is, if there are many solutions, how do we pick one?

6.1 ▪ Solving $A\mathbf{x} = \mathbf{b}$ when $\mathbf{b} \in \operatorname{img} A$, $\ker A \neq \{\mathbf{0}\}$

Suppose $A \in \mathbb{R}^{m\times n}$ with $m < n$. Thus A has more columns than rows. Since the number of pivots we have is at most the number of rows, we must have columns without any pivots — free variables. An therefore $\ker A \neq \{\mathbf{0}\}$. Note that this is not the only setting where $\mathbf{b} \in \operatorname{img} A$, $\ker A \neq \{\mathbf{0}\}$ can occur. For example, $A \in \mathbb{R}^{n\times n}$ could be singular but it could happen that $\mathbf{b} \in \operatorname{img} A$.

Example 6.1. Consider the three sets of matrices and vectors

(1)
$$A = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

(2)
$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}.$$

(3)
$$A = \begin{bmatrix} 1 & -1 \\ 0 & 0 \\ 1 & -1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix}.$$

In all these examples $\ker A \neq \{\mathbf{0}\}$ but $\mathbf{b} \in \operatorname{img} A$.

If $\mathbf{b} \in \operatorname{img} A$, we know that we have $\mathbf{x}$ such that $A\mathbf{x} = \mathbf{b}$. Now, let $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ be a basis for $\ker A$. Then it follows that

$$A(\mathbf{x} + c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + \cdots + c_k\mathbf{x}_k) = A\mathbf{x} = \mathbf{b}.$$

So, for every distinct choice of the scalars $\{c_j\}$ we find a new solution. And, for example,

$$c_1 = c_2 = c_3 = \cdots = c_k = 1,000,000,000,$$

is a valid choice but seems unlikely to be a relevant solution!

There are two immediate approaches to solving the non-uniqueness issue.

(1) Among all the possible solutions of $A\mathbf{x} = \mathbf{b}$, find the one with the smallest 2-norm.

(2) Find the vector $\mathbf{y}$ that minimizes

$$\mathcal{L}_\lambda(\mathbf{y}) = \|A\mathbf{y} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{y}\|_2^2, \quad \lambda > 0,$$

among all choices of $\mathbf{y} \in \mathbb{R}^n$. This is different than choice (1) and is called the *Tikhonov regularization* of the problem $A\mathbf{x} = \mathbf{b}$ and λ is called the regularlization parameter.

We focus on (2) here. In an optional section, we will cover (1). To solve (2) we first do the calculation

$$\left\| \begin{bmatrix} I \\ A \end{bmatrix} \mathbf{y} - \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix} \right\|_2^2 = \left\| \begin{bmatrix} \mathbf{y} \\ A\mathbf{y} - \mathbf{b} \end{bmatrix} \right\|_2^2 = \begin{bmatrix} \mathbf{y} \\ A\mathbf{y} - \mathbf{b} \end{bmatrix}^T \begin{bmatrix} \mathbf{y} \\ A\mathbf{y} - \mathbf{b} \end{bmatrix} = \mathbf{y}^T\mathbf{y} + (A\mathbf{y} - \mathbf{b})^T(A\mathbf{y} - \mathbf{b})$$
$$= \|\mathbf{y}\|_2^2 + \|A\mathbf{y} - \mathbf{b}\|_2^2.$$

We can then introduce the parameter $\lambda > 0$ into the picture by replacing I with $\sqrt{\lambda}I$ and it follows that

$$\mathcal{L}_\lambda(\mathbf{y}) = \left\| \begin{bmatrix} \sqrt{\lambda}I \\ A \end{bmatrix} \mathbf{y} - \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix} \right\|_2^2.$$

To find the minimizer of this, we just need to solve the normal equations associated to a regularized linear system! To find the minimum of $\mathcal{L}_\lambda$ we:

1. Replace A with $\begin{bmatrix} \sqrt{\lambda}I \\ A \end{bmatrix}$.
2. Replace $\mathbf{b}$ with $\begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix}$.
3. Solve the normal equations for the pair $\left(\begin{bmatrix} \sqrt{\lambda}I \\ A \end{bmatrix}, \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix} \right)$.

Example 6.2. Consider solving $A\mathbf{x} = \mathbf{b}$ when

$$A = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

with regularization parameter $\lambda = 1$.

1.

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

2.
$$\begin{bmatrix}0\\1\end{bmatrix} \mapsto \begin{bmatrix}0\\0\\0\\0\\1\end{bmatrix}$$

3. To construct the normal equations
$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\1&1&-1\\1&0&1\end{bmatrix}^T \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\1&1&-1\\1&0&1\end{bmatrix} = \begin{bmatrix}1&0&0&1&1\\0&1&0&1&0\\0&0&1&-1&1\end{bmatrix} \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\1&1&-1\\1&0&1\end{bmatrix} = \begin{bmatrix}3&1&0\\1&2&-1\\0&-1&3\end{bmatrix}.$$
and
$$\begin{bmatrix}1&0&0&1&1\\0&1&0&1&0\\0&0&1&-1&1\end{bmatrix} \begin{bmatrix}0\\0\\0\\0\\1\end{bmatrix} = \begin{bmatrix}1\\0\\1\end{bmatrix}.$$
And we then need to consider the augmented system
$$\left[\begin{array}{ccc|c}3&1&0&1\\1&2&-1&0\\0&-1&3&1\end{array}\right]$$

This gives the solution
$$\mathbf{x} = \begin{bmatrix}\frac{1}{3}\\0\\\frac{1}{3}\end{bmatrix}.$$

It is important to note in the previous example that
$$A\mathbf{x} = \begin{bmatrix}1&1&-1\\1&0&1\end{bmatrix} \begin{bmatrix}\frac{1}{3}\\0\\\frac{1}{3}\end{bmatrix} = \begin{bmatrix}0\\\frac{2}{3}\end{bmatrix} \neq \begin{bmatrix}0\\1\end{bmatrix} = \mathbf{b}.$$

So, while $\mathbf{b} \in \operatorname{img} A$, the regularization of the problem produces a "solution" that is not a true solution: Minimizing $\mathcal{L}_\lambda(y) = \|A\mathbf{y} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{y}\|_2^2$ is different than minimizing each term individually!

6.1.1 ▪ The effect of λ

We use Example 6.2 to demonstrate how the choice of λ affects the solution:

- $\lambda = 1$ gives $\begin{bmatrix}\frac{1}{3}\\0\\\frac{1}{3}\end{bmatrix}$.

- $\lambda = 1/2$ gives $\begin{bmatrix} \frac{2}{5} \\ 0 \\ \frac{2}{5} \end{bmatrix}$.
- $\lambda = 1/4$ gives $\begin{bmatrix} \frac{4}{9} \\ 0 \\ \frac{4}{9} \end{bmatrix}$.
- $\lambda = 1/8$ gives $\begin{bmatrix} \frac{8}{17} \\ 0 \\ \frac{8}{17} \end{bmatrix}$.

Indeed, for general λ the augmented system is

$$\left[\begin{array}{ccc|c} \lambda+2 & 1 & 0 & 1 \\ 1 & \lambda+1 & -1 & 0 \\ 0 & -1 & \lambda+2 & 1 \end{array}\right] \Rightarrow \mathbf{x} = \begin{bmatrix} \frac{1}{2+\lambda} \\ 0 \\ \frac{1}{2+\lambda} \end{bmatrix}.$$

As $\lambda \to 0^+$ this approaches $\mathbf{x} = \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix}$. Now, we see that

$$A\mathbf{x} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \mathbf{b}.$$

If we set $\lambda = 0$ initially, we have infinitely many solutions! But we choose $\lambda > 0$ but small, we get a single "solution" that is nearly a solution of $A\mathbf{x} = \mathbf{b}$ and has norm $\|\mathbf{x}\|_2$ that is not too large.

6.2 ▪ Orthonormal bases

Suppose that $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ is a basis for a subspace $W \subset \mathbb{R}^n$. Form the matrix

$$X = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_k \end{bmatrix}.$$

We note that if R is a $k \times k$ invertible matrix then $\operatorname{img} X = \operatorname{span}\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\} = \operatorname{img} XR$. Why? Because

$$\begin{aligned} \mathbf{y} \in \operatorname{span}\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\} &\Leftrightarrow \mathbf{y} = X\mathbf{c} \text{ for some } \mathbf{c} \\ &\Leftrightarrow \mathbf{y} = XR(R^{-1}\mathbf{c}) \text{ for some } \mathbf{c} \\ &\Leftrightarrow \mathbf{y} \in \operatorname{img} XR. \end{aligned}$$

This implies that if we compute the QR decomposition of X

$$X = QR = Q \begin{bmatrix} \hat{R} \\ \hline \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_k \end{bmatrix} \hat{R},$$

and if $\hat{R}$ is invertible (it is if X has linearly independent columns) then

$$\operatorname{span}\{\mathbf{q}_1, \ldots \mathbf{q}_k\} = \operatorname{span}\{\mathbf{x}_1, \ldots \mathbf{x}_k\}.$$

Moreover, the fact that $Q^TQ = I$ implies something very important about the $\mathbf{q}_j$'s:

$$\begin{aligned} Q^TQ &= \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_n \end{bmatrix}^T \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_n \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{q}_1^T \\ \mathbf{q}_2^T \\ \vdots \\ \mathbf{q}_n^T \end{bmatrix} \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_n \end{bmatrix} = (\mathbf{q}_i^T \mathbf{q}_j)_{\substack{1 \le i \le n \\ 1 \le j \le n}} = I. \end{aligned}$$

Specifically,

$$\mathbf{q}_i^T \mathbf{q}_j = \begin{cases} 0 & i \neq j, \\ 1 & i = j. \end{cases}$$

This implies that $\{\mathbf{q}_1, \dots \mathbf{q}_k\}$ forms an *orthonormal basis* for $\operatorname{img} X$:

Definition 6.3. *An orthonormal basis for a subspace $W \subset \mathbb{R}^n$ is a basis $\{\mathbf{q}_1, \dots \mathbf{q}_k\}$ for W that satisfies*

$$\mathbf{q}_i^T \mathbf{q}_j = \begin{cases} 0 & i \neq j, \\ 1 & i = j. \end{cases}$$

And we have see that an effective method to compute an orthonormal basis is to:

(1) Find a basis $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k\}$.

(2) Compute the QR decomposition of $X = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_k \end{bmatrix}$

(3) Take the first k columns of Q.

6.3 ▪ Minimal norm solution (optional)

We go back to the first regularization idea.

(1) Among all the possible solutions of $A\mathbf{x} = \mathbf{b}$, find the one with the smallest 2-norm.

6.3.1 ▪ Finding one solution

Since we are assuming that $\mathbf{b} \in \operatorname{img} A$ we have at least one solution. To find one solution, reduce $\left[\begin{array}{c|c} A & b \end{array}\right]$ to row echelon form. If rank $A = r$ the whole system will be of the form:

$$\left[\begin{array}{ccccccccccccccccc|c} 0 & \boxed{\times} & * & \cdots & * & * & \cdots & * & * & \cdots & \cdots & * & * & * & \cdots & * & * \\ 0 & 0 & 0 & \cdots & 0 & \boxed{\times} & \cdots & * & * & \cdots & \cdots & * & * & * & \cdots & * & * \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \boxed{\times} & \cdots & \cdots & * & * & * & \cdots & * & * \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & & \ddots & & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & \cdots & 0 & \boxed{\times} & * & \cdots & * & * \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \end{array}\right]$$

If we remove all the columns associated to free variables, and remove the zero rows, we arrive at a solvable triangular system!

$$\left[\begin{array}{cccccc|c} \boxed{\times} & * & \cdots & & \cdots & * & * \\ 0 & \boxed{\times} & * & \cdots & \cdots & * & * \\ 0 & 0 & \boxed{\times} & * & \cdots & * & * \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \boxed{\times} & * & * \\ 0 & 0 & \cdots & \cdots & 0 & \boxed{\times} & * \end{array}\right]$$

Solving this system will give a vector $\mathbf{c}$ that gives a solution of $\left[\begin{array}{c|c} A & b \end{array}\right]$, when properly interpreted:

- Suppose $j_1, j_2, \ldots, j_r$ are the indices corresponding to the pivot columns of A.
- Define $\mathbf{x} = (x_i)_{i=1}^n$ via

$$x_i = \begin{cases} c_k & i = j_k, \\ 0 & \text{otherwise.} \end{cases}$$

This is just one process for finding a solution. Any other successful process will work.

6.3.2 ▪ Find an orthonormal basis for $\ker A$

The kernel of a matrix $A \in \mathbb{R}^{m\times n}$ is actually characterized by the set of all vectors that are orthogonal to the rows of A. So, if we perform a QR factorization of A^T: $A^T = QR$ then we see that

$$A = \underbrace{R^T}_{m\times n}\underbrace{Q^T}_{n\times n}.$$

In general, R will have some number of rows of all zeros, say $r_1, r_2, \ldots, r_p$. Then an orthonormal basis for $\ker A$ is given by

$$\{\mathbf{q}_{r_1}, \mathbf{q}_{r_2}, \ldots, \mathbf{q}_{r_p}\}.$$

Any solution of $A\mathbf{x} = \mathbf{b}$ can now be written in the form

$$\mathbf{x} - \sum_{j=1}^{p} c_j \mathbf{q}_{r_j} = \mathbf{x} + \tilde{Q}\mathbf{c},$$
$$\tilde{Q} = \begin{bmatrix} \mathbf{q}_{r_1} & \cdots & \mathbf{q}_{r_p} \end{bmatrix}.$$

6.3.3 ▪ Find the best c_j's

We now need to minimize

$$\mathcal{M}(\mathbf{c}) = \|\mathbf{x} - \tilde{Q}\mathbf{c}\|_2^2.$$

But we already know how to do this, by solving the normal equations, which in this case becomes very easy:

$$\underbrace{\tilde{Q}^T\tilde{Q}}\, I\mathbf{c} = \mathbf{c} = \tilde{Q}^T\mathbf{x}$$

So, the minimal norm solution is given by

$$\mathbf{x}_{\min} = \mathbf{x} - \tilde{Q}\tilde{Q}^T\mathbf{x}.$$

6.3.4 ▪ Improving the solution process

This process was inefficient because it relied first on Gaussian elimination and then second on a QR factorization. Indeed, one can use the $A = R^T Q^T$ factorization to find the initial solution $\mathbf{x}$. This is demonstrated in the following example.

Example 6.4. Find the solution of $A\mathbf{x} = \mathbf{b}$ such that $\|\mathbf{x}\|_2$ is minimized where

$$A = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

We first find a QR factorization of A

$$A^T = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ -1 & 1 \end{bmatrix} = QR = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & -\frac{2}{\sqrt{6}} \\ -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} \end{bmatrix} \begin{bmatrix} \sqrt{3} & 0 \\ 0 & \sqrt{2} \\ 0 & 0 \end{bmatrix},$$

This gives

$$A\mathbf{x} = \mathbf{b} \Rightarrow \begin{bmatrix} \sqrt{3} & 0 & 0 \\ 0 & \sqrt{2} & 0 \end{bmatrix} \underbrace{Q^T \mathbf{x}}_{\mathbf{z}} = \mathbf{b},$$

$$\left[\begin{array}{ccc|c} \sqrt{3} & 0 & 0 & 0 \\ 0 & \sqrt{2} & 0 & 1 \end{array}\right]$$

We have a free variable here and we choose it to be zero.

$$\mathbf{z} = \begin{bmatrix} 0 \\ 1/\sqrt{2} \\ 0 \end{bmatrix} \Rightarrow \mathbf{x} = Q\mathbf{z} = \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix}.$$

Then our minimal norm solution is given by

$$\mathbf{x}_{\min} = \mathbf{x} - \tilde{Q}\tilde{Q}^T\mathbf{x} = \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix} - \begin{bmatrix} \frac{1}{\sqrt{6}} \\ -\frac{2}{\sqrt{6}} \\ -\frac{1}{\sqrt{6}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix}.$$

This example shows us that by choosing the free variable in the augmented system to be zero allows us to directly solve for the minimal norm solution! And this gives us the same solution as the Tikhonov regularization method as $\lambda \to 0^+$.

6.4 ▪ An example application

Suppose your rock company produces:

- 20 tons of road base (RB),
- 10 tons of sand (S), and
- 10 tons of gravel (G)

each day. You have 4 categories of equipment (Truck, Shovel A, Shovel B, Mixer) and you are wondering if you can increase production by 50% by buying and selling equipment. But you do not want to restructure the company too much in the process. As a simplification, you estimate

- 1 Truck per 1 hour $\rightarrow \begin{bmatrix} 0 \text{ tons of RB} \\ 0 \text{ tons of S} \\ 1 \text{ ton of G} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
- 1 Shovel A per 1 hour $\rightarrow \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$
- 1 Mixer per 1 hour $\rightarrow \begin{bmatrix} 1 \\ -\frac{1}{2} \\ -\frac{1}{2} \end{bmatrix}$
- 1 Shovel B per 1 hour $\rightarrow \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$

We capture this information in the matrix

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

Currently,

$$A\mathbf{x} = \begin{bmatrix} 20 \\ 10 \\ 10 \end{bmatrix}.$$

Note: It is worthwhile to understand the units of A. For example, a_{11} has units of tons of RB per hour per Truck. The units of x_1 are Trucks $\times$ hours/day. We suppose that the number of hours worked per day is fixed. Our production goals state that we want

$$A\mathbf{x}_{\text{new}} = \begin{bmatrix} 30 \\ 20 \\ 20 \end{bmatrix},$$

implying

$$A(\mathbf{x}_{\text{new}} - \mathbf{x}) = \begin{bmatrix} 10 \\ 5 \\ 5 \end{bmatrix}$$

One approach to solving this problem is to use Tikhonov regularization: Minimize

$$\mathcal{L}_\lambda(\mathbf{y}) = \left\| A\mathbf{y} - \begin{bmatrix} 10 \\ 5 \\ 5 \end{bmatrix} \right\|_2^2 + \lambda \|\mathbf{y}\|_2^2.$$

And now λ becomes a design parameter. The larger it its the more we are emphasizing that we want $\mathbf{x} - \mathbf{x}_{\text{new}}$ to be small — that we do not want to restructure the business too

much. The smaller it is, the more we are enforcing that we want to hit our production goals. We solve this problem with MATLAB using different choices of λ.

First we set up the linear systems.

```
A = [0,0,1,0;
     0,2,-.5,1;
     1,1,-.5,1];
b = [10;5;5];
```

For $\lambda = 1$, we assemble the matrix and compute the R factor for the QR factorization of the augmented matrix

```
lam = 1;
An = [eye(4)*sqrt(lam); A]; bn = [zeros(4,1); b];
[¬,R] = House([An,bn])
```

```
R =

   -1.4142   -0.7071    0.3536   -0.7071   -3.5355
         0   -2.3452    0.5330   -1.0660   -5.3300
         0         0   -1.4460    0.1257   -6.2869
         0         0         0   -1.1610   -2.2470
         0         0         0         0   -8.0322
         0    0.0000         0         0   -0.0000
    0.0000         0         0         0   -0.0000
```

Then the difference $\mathbf{x}_{\text{new}} - \mathbf{x}$ is computed

```
dx = Backsub(R(1:4,:))
```

```
dx =

    1.4516
    2.4194
    4.5161
    1.9355
```

This is stating that one should buy 1-2 new Trucks, 2-3 new Shovel A's, 4-5 new Shovel B's and 2 new Mixers. And to see how close we get to our target:

```
A*dx
```

```
ans =

    4.5161
    4.5161
    3.5484
```

Our goal was $[10, 5, 5]^T$, so this is not particularly good. So, we probably want to decrease λ to try to better hit the production goal. With $\lambda = 0.1$:

```
lam = 0.1;
An = [eye(4)*sqrt(lam); A]; bn = [zeros(4,1); b];
[¬,R] = House([An,bn]);
dx = Backsub(R(1:4,:))
A*dx
```

```
dx =

    2.9221
    3.2004
    8.9518
    3.0612


ans =

    8.9518
    4.9861
    4.7078
```

With significantly more investment, we almost hit the production goal. So, figuring out which choice of λ gives you the best solution is probably a matter of budget.

6.5 ▪ Solving $A\mathbf{x} = \mathbf{b}$ when $\mathbf{b} \notin \operatorname{img} A$

The last topic we need to discuss concerning the solution of linear systems is that of solving $A\mathbf{x} = \mathbf{b}$ with there is no solution, and when there is not a unique solution of the least squares problem. Fortunately, Tikhonov regularization provides a solution for this without any extra effort. Just find the minimum of

$$\mathcal{L}_\lambda(\mathbf{y}) = \|A\mathbf{y} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{y}\|_2^2,$$

using the methods we have described. This will always work. But the trade-off is that we have to choose λ. This choice would typically be guided by an application.

An alternate approach would be to find, among all the vectors $\mathbf{x}$ that minimize

$$\|A\mathbf{y} - \mathbf{b}\|_2^2,$$

find the one that has the smallest 2-norm. In next chapter we will see how the singular value decomposition can be used to solve this problem.

Chapter 7

Eigenvalues and eigenvectors

Consider the following simplified scenario of an influenza outbreak. We divide the entire population into three classes:

- Susceptible (S),
- Infected (I), and
- Recovered/immune (R).

The susceptible class has either not been exposed to the virus or has been given the vaccine. The infected class will recover eventually but can infect others in the meantime. The recovered class has either had been exposed to the virus and has recovered or has been successfully vaccinated.

We hypothesize that we cannot change human human behavior/interaction but we can increase vaccinations. So, we study the effect of vaccination within the outbreak. Suppose that today is day zero and we capture the proportion of the population in each of the three class S,I and R in a vector

$$\mathbf{x}_0 = \begin{bmatrix} 0.9(\text{S}) \\ 0.09(\text{I}) \\ 0.01(\text{R}) \end{bmatrix}.$$

It is observed that each day 1 out of 1000 of the infected individuals recover. Each day, 1 out of every 200 susceptible individuals become infected[4]. Because of mutations in the virus, one out of every 10000 recovered individuals become susceptible again. Through vaccination, we are able to move some number of individuals in S directly into R. Class this fraction p, $0 \leq p \leq 1$. The evolution of the population from day k to day $k+1$ is given by

$$\mathbf{x}_{k+1} = M_p \mathbf{x}_k, \quad M_p = \begin{bmatrix} 1 - \frac{1}{200} - p & 0 & \frac{1}{10000} \\ \frac{1}{200} & 1 - \frac{1}{1000} & 0 \\ p & \frac{1}{1000} & 1 - \frac{1}{10000} \end{bmatrix}.$$

Note that the columns of the matrix are all positive and all sum to one. This is going preserve that the components of $\mathbf{x}_k$ are non-negative and $\|\mathbf{x}_k\|_1 = 1$.

[4]Note that this is not the best way to model this. If a larger faction of the population was infected then we would expect susceptible individuals to become infected at a higher rate. But this is nonlinear and modeling and analyzing the system with linear algebra along would not be possible.

Now suppose[5]

$$M_p = U\Lambda U^{-1}, \quad \Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}.$$

Then

$$\begin{aligned} \mathbf{x}_1 &= M_p\mathbf{x}_0, \\ \mathbf{x}_2 &= M_p\mathbf{x}_1 = M_pM_p\mathbf{x}_0, \\ \mathbf{x}_3 &= M_p\mathbf{x}_2 = M_pM_pM_p\mathbf{x}_0, \\ &\vdots \\ \mathbf{x}_k &= M_p^k\mathbf{x}_0. \end{aligned}$$

This leads us to examine powers of the matrix,

$$\begin{aligned} M_p &= U\Lambda U^{-1}, \\ M_p^2 &= U\Lambda U^{-1}U\Lambda U^{-1} = U\Lambda^2U^{-1}, \\ M_p^3 &= U\Lambda U^{-1}U\Lambda^2U^{-1} = U\Lambda^3U^{-1}, \\ &\vdots \\ M_p^3 &= U\Lambda^kU^{-1}. \end{aligned}$$

This implies that

$$\mathbf{x}_k = U\Lambda^kU^{-1}\mathbf{x}_0 = U\begin{bmatrix} \lambda_1^k & 0 & 0 \\ 0 & \lambda_2^k & 0 \\ 0 & 0 & \lambda_3^k \end{bmatrix}U^{-1}\mathbf{x}_0.$$

The numbers $\lambda_1, \lambda_2, \lambda_3$ are called *eigenvalues* and we will define them soon. They can be computed using MATLAB's `eig` command.

```
p = 0;
M = [1-1/200-p, 0, 1/10000;
      1/200,   1-1/1000,     0;
      p, 1/1000, 1-1/10000];
lams = eig(M)
```

```
lams =

    0.9950
    0.9989
    1.0000
```

From this we that $\lambda_1 \approx 0.9950$, $\lambda_2 \approx 0.9989$ and $\lambda_3 = 1$. Well, this does not prove that $\lambda_3 = 1$, it just gives a strong indication of it. Indeed a theorem called the *Perron-Frobenius theorem* does establish this. But we find that

$$\mathbf{x}_\infty = \lim_{k\to\infty}\mathbf{x}_k = \lim_{k\to\infty} U\begin{bmatrix} \lambda_1^k & 0 & 0 \\ 0 & \lambda_2^k & 0 \\ 0 & 0 & \lambda_3^k \end{bmatrix}U^{-1}\mathbf{x}_0 = U\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}U^{-1}\mathbf{x}_0$$

[5]We will see that this is indeed the case.

The matrix U here is the matrix of what are called eigenvectors. We can evaluate this last expression using MATLAB as follows. First, find U and Λ:

```
[U,Lambda] = eig(M)
```

```
U =

    0.6144   -0.0180    0.0199
   -0.7729    0.7159    0.0995
    0.1586   -0.6979    0.9948


Lambda =

    0.9950         0         0
         0    0.9989         0
         0         0    1.0000
```

Check that this is indeed the right factorization

```
M - U*Lambda*inv(U)
```

```
ans =

   1.0e-15 *

    0.4441   -0.1388   -0.2492
   -0.1154    0.3331   -0.1943
   -0.3331   -0.2229    0.2220
```

Then, compute $\mathbf{x}_\infty$:

```
x0 = [ .9;    % S
       .09;   % I
       .01 ]; % R
N = zeros(3);
N(3,3) = 1;
xinf = U*N*(U\x0)
```

```
xinf =

   0.017857142857175
   0.089285714286028
   0.892857142856604
```

We check that this is indeed (approximately) the same as taking a large number of iterations:

```
xk = x0;
for k = 1:100000
    xk = M*xk;
end
xk
```

```
xk =

   0.017857142857149
   0.089285714285757
   0.892857142857460
```

This construction allows us to quickly examine the effects of changing p.

```
p = 0.01;
M = [1-1/200-p, 0, 1/10000;
      1/200,   1-1/1000,     0;
      p, 1/1000, 1-1/10000];
x0 = [ .9;    % S
       .09;   % I
       .01 ]; % R
[U,Lambda] = eig(M);
N = Lambda;
for i = 1:3
    if N(i,i) < 1-1e-8 % Automatically determine which entries
       N(i,i) = 0;     % to zero out
    end
end
xinf = U*N*(U\x0)
```

```
xinf =

   0.006410256410244
   0.032051282051178
   0.961538461538756
```

And so, increasing p to $p = 0.1$ approximately cuts the number of infected individuals, in the long term, buy almost a factor of 3.

7.1 ▪ Eigenvalues and eigenvectors

We now give the definition of eigenvalues and eigenvectors.

Definition 7.1. *Let $A \in \mathbb{R}^{n\times n}$ be an $n\times n$ matrix. A scalar λ is called an* eigenvalue *of A if there is a non-zero vector $\mathbf{v}$, called an eigenvector, such that*

$$A\mathbf{v} = \lambda\mathbf{v}. \tag{7.1}$$

It is important that (7.1) can be rewritten as

$$A\mathbf{v} = \lambda\mathbf{v} \Leftrightarrow A\mathbf{v} = \lambda I\mathbf{v} \Leftrightarrow (A - \lambda I)\mathbf{v} = \mathbf{0}.$$

This immediately leads to the following theorem.

Theorem 7.2. *A scalar λ is an eigenvalue of $A \in \mathbb{R}^{n\times n}$ if and only if $(A - \lambda I)$ is singular. The corresponding eigenvectors are solutions to the eigenvalue equation $(A - \lambda I)\mathbf{v} = \mathbf{0}$.*

We will see below that $\det(A - \lambda I)$ is a degree n polynomial in λ if $A \in \mathbb{R}^{n\times n}$. And with that in mind we have the following definition.

Definition 7.3. *For $A \in \mathbb{R}^{n\times n}$, $p(\lambda) = \det(A - \lambda I)$ is called the* characteristic polynomial *and*

$$\det(A - \lambda I) = 0$$

is called the characteristic equation.

This the leads to a way to characterize eigenvalues themselves.

Corollary 7.4. *A scalar λ is an eigenvalue of the matrix $A \in \mathbb{R}^{n\times n}$ if and only if λ is a solution of the characteristic equation*

$$\det(A - \lambda I) = 0.$$

Example 7.5. Find the eigenvalues and eigenvectors of

$$A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}.$$

We first look at the characteristic equation

$$\det(A - \lambda I) = \det \begin{bmatrix} 3-\lambda & 1 \\ 1 & 3-\lambda \end{bmatrix} = (3-\lambda)^2 - 1 = 0$$

Note that we did not use an LU decomposition to compute this, instead we used

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

While there is a simpler method, we use the quadratic equation to find λ:

$$0 = (3-\lambda)^2 - 1 = \lambda^2 - 6\lambda + 8,$$
$$\lambda = \frac{6 \pm \sqrt{36 - 32}}{2} = 3 \pm 1 = 2, 4.$$

For the first eigenvector, $\lambda = 2$:

$$\left[\ (A - 2I) \mid \mathbf{0}\ \right] = \left[\begin{array}{cc|c} 1 & 1 & 0 \\ 1 & 1 & 0 \end{array}\right].$$

Row echelon form is simply

$$\left[\begin{array}{cc|c} 1 & 1 & 0 \\ 0 & 0 & 0 \end{array}\right].$$

This states $v_1 + v_2 = 0$, or $v_1 = -v_2$, and we can choose $v_2 = 1$, so that

$$\mathbf{v} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

is an eigenvector with eigenvalue $\lambda = 2$.

For the second eigenvector, $\lambda = 4$:

$$\left[\begin{array}{c|c} (A - 4I) & \mathbf{0} \end{array}\right] = \left[\begin{array}{cc|c} -1 & 1 & 0 \\ 1 & -1 & 0 \end{array}\right].$$

Row echelon form is simply

$$\left[\begin{array}{cc|c} -1 & 1 & 0 \\ 0 & 0 & 0 \end{array}\right].$$

This states $v_1 - v_2 = 0$, or $v_1 = -v_2$, and we can choose $v_2 = 1$, so that

$$\mathbf{v} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

is an eigenvector with eigenvalue $\lambda = 4$. To summarize:

- $A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$ has eigenvalue $\lambda = 2$ with eigenvector $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$.
- $A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$ has eigenvalue $\lambda = 4$ with eigenvector $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

7.2 ▪ Determinants revisited: Cofactor expansion

We now present an alternate method to compute the determinant of a matrix. Computing a determinant by means of the LU decomposition, i.e., by row operations, requires division and for this reason it less desirable to work with when the matrix under consideration contains a variable, as is the case with the characteristic equation $\det(A - \lambda I) = 0$.

7.2.1 ▪ 3×3 determinants

The key to understanding determinants of larger-than-2×2 matrices is to keep in mind the sign table

$$S = \begin{bmatrix} + & - & + & - & \cdots \\ - & + & - & - & \cdots \\ + & - & + & - & \cdots \\ - & + & - & - & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} = ((-1)^{i+j})_{\substack{1 \le i \le n \\ 1 \le j \le n}}.$$

So, in the 3×3 case, we consider

$$\begin{bmatrix} + & - & + \\ - & + & - \\ + & - & + \end{bmatrix}.$$

Then associated to the matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}.$$

is the matrix of cofactors

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

where c_{ij} is the determinant of the 2×2 matrix found by deleting row i and column j of A:

$$c_{11} = \det \begin{bmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{bmatrix}, \quad c_{23} = \det \begin{bmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{bmatrix}.$$

Then the important formula is

$$\begin{aligned} \det A &= s_{i1}a_{i1}c_{i1} + s_{i2}a_{i2}c_{i2} + s_{i3}a_{i3}c_{i3}, \\ &= s_{1j}a_{1j}c_{1j} + s_{2j}a_{2j}c_{2j} + s_{3j}a_{3j}c_{3j} \end{aligned}$$

This is called the *cofactor expansion* of the determinant and this formula states that it can performed down any column or across any row.

Example 7.6. Across the first row:

$$\det \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = a_{11} \det \begin{bmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{bmatrix} - a_{12} \begin{bmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{bmatrix} + a_{13} \begin{bmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix},$$

Down the second column:

$$\det \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = -a_{12} \det \begin{bmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{bmatrix} + a_{22} \det \begin{bmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{bmatrix} - a_{32} \det \begin{bmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{bmatrix}.$$

Example 7.7. Compute the determinant of

$$\begin{bmatrix} 1 & 0 & 1 \\ 2 & -1 & 1 \\ 3 & 0 & 1 \end{bmatrix}.$$

To make the computation easiest, by hand, you want to utilize as as many zeros as possible. So the cofactor expansion down the second column is the way to go. From the sign table, the (2,2) entry gets a +1, so we find

$$\det \begin{bmatrix} 1 & 0 & 1 \\ 2 & -1 & 1 \\ 3 & 0 & 1 \end{bmatrix} = (-1) \det \begin{bmatrix} 1 & 1 \\ 3 & 1 \end{bmatrix} = -(1 - 3) = 2.$$

But it should be noted that in most cases, beyond 3×3, using row echelon form to compute the determinant will be simpler by hand.

7.2.2 ▪ $n \times n$ determinants

In the case of the 3×3 determinant, we expressed it as a linear combination of 3 2×2 determinants. In general, an $n \times n$ determinant can be expressed as a linear combination of n $(n-1) \times (n-1)$ determinants. As before, given a matrix A, we define a matrix C of cofactors so that c_{ij} is the determinant of the matrix found by deleting row i and row j of the matrix A.

Theorem 7.8. *The cofactor expansion of* $\det A$ *where* $A = (a_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le n}} \in \mathbb{R}^{n \times n}$ *is given by*

$$\begin{aligned} \det A &= s_{i1} a_{i1} c_{i1} + s_{i2} a_{i2} c_{i2} + \cdots + s_{in} a_{in} c_{in}, \\ &= s_{1j} a_{1j} c_{1j} + s_{2j} a_{2j} c_{2j} + \cdots + s_{nj} a_{nj} c_{nj} \end{aligned}$$

Example 7.9. Compute the determinant of

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

We notice that the second row has only two non-zero entries and we therefore use the cofactor expansion there

$$\det A = -(1) \det \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 2 & 0 & 1 \\ 0 & 1 & -1 & 1 \\ 2 & 1 & 1 & 1 \end{bmatrix} + (1) \det \begin{bmatrix} 2 & 0 & 1 & 1 \\ -1 & 2 & 0 & 1 \\ 0 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Then for each of these sub determinants we just make use of the cofactor expansion across the first row

$$\det \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 2 & 0 & 1 \\ 0 & 1 & -1 & 1 \\ 2 & 1 & 1 & 1 \end{bmatrix} = (1) \det \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix} - (1) \det \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & -1 \\ 2 & 1 & 1 \end{bmatrix}$$

$$\det \begin{bmatrix} 2 & 0 & 1 & 1 \\ -1 & 2 & 0 & 1 \\ 0 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = (2) \det \begin{bmatrix} 2 & 0 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 1 \end{bmatrix} + (1) \det \begin{bmatrix} -1 & 2 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} - (1) \det \begin{bmatrix} -1 & 2 & 0 \\ 0 & 1 & -1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Then we have to work on these 5 3×3 determinants

$$\det \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix} = (1) \det \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} - (1) \det \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} = 2,$$

$$\det \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & -1 \\ 2 & 1 & 1 \end{bmatrix} = (1) \det \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} - (-1) \det \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} = -2,$$

$$\det \begin{bmatrix} 2 & 0 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = (2) \det \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} + (1) \det \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} = -2,$$

$$\det \begin{bmatrix} -1 & 2 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = (1) \det \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} - (1) \det \begin{bmatrix} -1 & 2 \\ 1 & 1 \end{bmatrix} = 1,$$

$$\det \begin{bmatrix} -1 & 2 & 0 \\ 0 & 1 & -1 \\ 1 & 1 & 1 \end{bmatrix} = (1) \det \begin{bmatrix} -1 & 0 \\ 1 & 1 \end{bmatrix} - (-1) \det \begin{bmatrix} -1 & 2 \\ 1 & 1 \end{bmatrix} = -4.$$

This gives

$$\det \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 2 & 0 & 1 \\ 0 & 1 & -1 & 1 \\ 2 & 1 & 1 & 1 \end{bmatrix} = 4, \quad \det \begin{bmatrix} 2 & 0 & 1 & 1 \\ -1 & 2 & 0 & 1 \\ 0 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = 1.$$

and finally

$$\det A = -3$$

The reader is encouraged to recalculate $\det A$ but row operations and one immediately notices that the calculation is much easier.

7.3 ▪ More important aspects of eigenvalues

We begin with at theorem that summarizes important facts.

Theorem 7.10.

- *A is invertible if and only if $\lambda = 0$ is not an eigenvalue.*
- *A has at least one and at most n distinct eigenvalues.*
- *The eigenvalues of a triangular matrix are the entries on the diagonal.*

Example 7.11. Compute the eigenvalues and eigenvectors of

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

So, to compute the characteristic polynomial using a cofactor expansion across the first row

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

This is zero if $\lambda=1$ or if $\lambda^2=1$, or $\lambda=\pm 1$. So, we get $\lambda=1$ twice. We then compute the eigenvectors. For $\lambda=1$

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

After interchanging the first row with the last and reducing, we obtain

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

For an eigenvector $\mathbf{v}$ this just states that $v_2-v_3=0$, and v_1 and v_3 are free variables:

$$\mathbf{v}=\begin{bmatrix}v_1\\ v_3\\ v_3\end{bmatrix}=v_1\begin{bmatrix}1\\ 0\\ 0\end{bmatrix}+v_3\begin{bmatrix}0\\ 1\\ 1\end{bmatrix}.$$

In this case we find two linearly independent eigenvectors $\mathbf{v}_1=\begin{bmatrix}1\\ 0\\ 0\end{bmatrix}$ and $\mathbf{v}_2=\begin{bmatrix}0\\ 1\\ 1\end{bmatrix}$.

Then for $\lambda=-1$

$$A+I=\begin{bmatrix}2 & 0 & 0\\ 0 & 1 & 1\\ 0 & 1 & 1\end{bmatrix}.$$

Eliminating in the last row gives

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

An eigenvector $\mathbf{v}$ is determined by $2v_1=0$, and $v_2=-v_3$, or

$$\mathbf{v}=\begin{bmatrix}0\\ -v_3\\ v_3\end{bmatrix}=v_3\begin{bmatrix}0\\ -1\\ 1\end{bmatrix}.$$

And a third eigenvector for A is $\begin{bmatrix}0\\ -1\\ 1\end{bmatrix}$, associated to $\lambda=-1$.

The next issue we need to address is brought to the fore by the following example.

Example 7.12. Compute the eigenvalues and eigenvectors of

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

So, to compute the characteristic polynomial using a cofactor expansion across the first row

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

This is zero if $\lambda = 1$ or if $\lambda^2 = -1$, or $\lambda = \pm\sqrt{-1}$. So, we started with a real matrix, and we have now encountered our first complex number!

Definition 7.13. *A complex number is $z = x + \mathrm{i}\, y$, $x, y \in \mathbb{R}$, $\mathrm{i}^2 = -1$. We use $\mathbb{C}$ to denote the set of all complex numbers.*

We note that we can use all of the arithmetic operations that we are used to using on real number on complex numbers:

$$\begin{aligned} (x_1 + \mathrm{i}\, y_1) + (x_2 + \mathrm{i}\, y_2) &= (x_1 + x_2) + \mathrm{i}(y_1 + y_2) \in \mathbb{C}, \\ (x_1 + \mathrm{i}\, y_1)(x_2 + \mathrm{i}\, y_2) &= x_1 x_2 - y_1 y_2 + \mathrm{i}(y_1 x_2 + y_2 x_1) \in \mathbb{C}, \\ \frac{x_1 + \mathrm{i}\, y_1}{x_2 + \mathrm{i}\, y_2} &= \frac{x_1 + \mathrm{i}\, y_1}{x_2 + \mathrm{i}\, y_2} \frac{x_2 - \mathrm{i}\, y_2}{x_2 - \mathrm{i}\, y_2} = \frac{x_1 x_2 + y_1 y_2}{x_2^2 + y_2^2} + \mathrm{i}\, \frac{y_1 x_2 - y_2 x_1}{x_2^2 + y_2^2} \in \mathbb{C}, \\ |x_1 + \mathrm{i}\, y_1| &= \sqrt{(x_1 + \mathrm{i}\, y_1)(x_1 - \mathrm{i}\, y_1)} = \sqrt{x_1^2 + y_1^2}. \end{aligned}$$

In the above calculation we used $x_1 + \mathrm{i}\, y_1$ and $x_1 - \mathrm{i}\, y_1$. The latter is called the complex conjugate of the former. If $z = x + \mathrm{i}\, y$ the complex conjugate is denoted $\bar{z} = x - \mathrm{i}\, y$.

Definition 7.14. *An eigenvalue $\hat{\lambda}$ of a matrix $A \in \mathbb{R}^{n \times n}$ has multiplicity k if*

$$\det(A - \lambda I) = (\lambda - \hat{\lambda})^k p(\lambda),$$

where $p(\lambda)$ is a polynomial satisfying $p(\hat{\lambda}) \neq 0$.

With this in mind we have the following theorem.

Theorem 7.15. *A matrix $A \in \mathbb{R}^{n \times n}$ has exactly n, possibly complex, eigenvalues if each eigenvalue is counted according its multiplicity.*

The proof of this follows from the fundamental theorem of algebra, something we do not cover here. In general, eigenvalues are difficult to compute in the sense that roots of polynomials are tough to find — solving linear equations required rational number while solving a quadratic involves roots and polynomials beyond second order are even more difficult to handle.

The Gershgorin circle theorem is a relatively simple theorem to use that allows us to say something immediately about eigenvalues after inspecting a matrix. For some matrices it is effectively useless but it can be quite useful for others. Before we state the theorem we have a couple definitions.

For $a \in \mathbb{C}$ and $r \geq 0$ define

$$D = \{z \in \mathbb{C} : |z - a| \leq r\}.$$

It is worth taking some time to understand this set. Write $z = x + \mathrm{i}\, y$ and $a = \alpha + \mathrm{i}\, \beta$. Then

$$|z - a|^2 = (x - \alpha + \mathrm{i}(y - \beta))(x - \alpha - \mathrm{i}(y - \beta)) = (x - \alpha)^2 + (y - \beta)^2.$$

This is the distance squared between the two points (x, y) and (α, β). Geometrically, the set D is the set of all points in $\mathbb{C}$ that are a distance of at most r from $a = \alpha + \mathrm{i}\, \beta$ — a filled in circle, i.e., a disk! See Figure 7.1 for a demonstration of this.

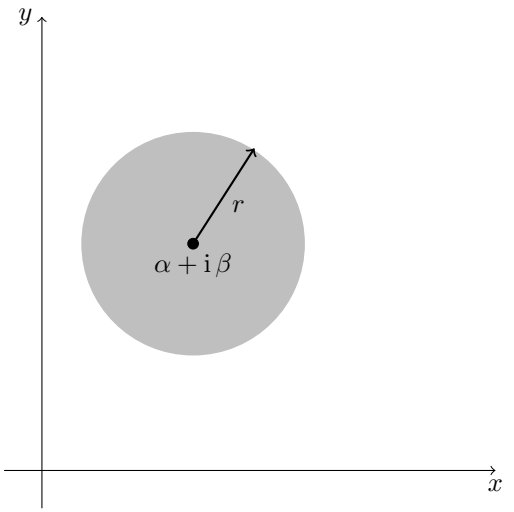


Figure 7.1: A disk D centered at $a = \alpha + \mathrm{i}\, \beta$ with radius r.

Definition 7.16. *Let $A \in \mathbb{R}^{n \times n}$ (or $\mathbb{C}^{n \times n}$). For each $i = 1, 2, \ldots, n$ define the* Gershgorin disks

$$D_i = \{z \in \mathbb{C} : |z - a_{ii}| \leq r_i\}, \quad r_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|.$$

The the Gershgorin domain *D_A is defined to be the union of all the disks:*

$$D_A = \bigcup_{i=1}^{n} D_i \subset \mathbb{C}.$$

Theorem 7.17 (Gershgorin Circle Theorem). *All the eigenvalues of A lie in its Gershgorin domain.*

Proof. Suppose λ is an eigenvalue of A:

$$A\mathbf{v} = \lambda\mathbf{v}, \quad \mathbf{v} \neq \mathbf{0}.$$

Suppose the ith entry of v is largest: $|v_i| = \|\mathbf{v}\|_\infty$. And since we can divide both sides of this equation scalars, we can suppose that $v_i = 1$. Looking at the ith row of this equation gives

$$\sum_{j=1}^{n} a_{ij}v_i = \lambda v_j,$$

$$a_{ii} + \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}v_j = \lambda.$$

This then implies that

$$|\lambda - a_{ii}| \leq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}v_j| \leq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|$$

because $|v_j| \leq 1$. This shows that every eigenvalue must be in at least one disk and the conclusion follows. .

Example 7.18. Consider the matrix

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{7.2}$$

The Gershgorin disk for this matrix are

$$D_1 = \{z \in \mathbb{C} : |z - 1| \leq 0\},$$
$$D_2 = \{z \in \mathbb{C} : |z - 0| \leq 1\},$$
$$D_3 = \{z \in \mathbb{C} : |z - 0| \leq 1\}.$$

The Gershgorin domain is shown in Figure 7.2. This gives us a rough idea of where the eigenvalues are but really does not tell us very much.

Example 7.19. Now consider the matrix

$$A = \begin{bmatrix} 2 & -1 & 0 \\ 1 & 4 & -1 \\ -1 & -1 & -3 \end{bmatrix}. \tag{7.3}$$

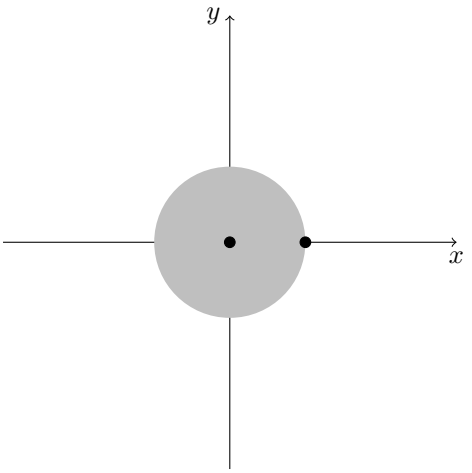


Figure 7.2: The Gershgorin domain for (7.2)

Now, the Gershgorin disks are given by

$$D_1 = \{z \in \mathbb{C} : |z - 2| \leq 1\},$$
$$D_2 = \{z \in \mathbb{C} : |z - 4| \leq 2\},$$
$$D_3 = \{z \in \mathbb{C} : |z + 3| \leq 2\}.$$

While these disks have a larger radius of that in the previous example, it gives us a good idea of where the eigenvalues can be.

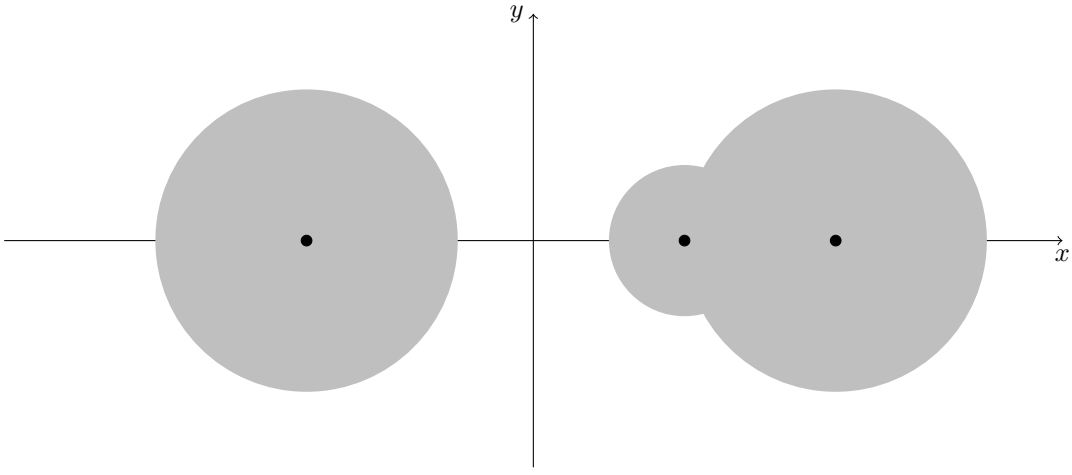


Figure 7.3: The Gershgorin domain for (7.3)

An application of the Gershgorin circle theorem gives us the following

Proposition 7.20. *If a matrix $A \in \mathbb{R}^{n\times n}$ is diagonally dominant then it is invertible.*

Proof. Recall that a matrix is invertible if it does not have zero for an eigenvalue.

Furthermore, A is diagonally dominant if

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|,$$

for every $i = 1, 2, \ldots, n$. This implies that $0 \notin D_A$ and zero cannot be an eigenvalue of A.

7.3.1 ▪ Eigenvalues and symmetric matrices

Suppose now that $A = A^T$. This has a dramatic impact on the possible eigenvalues for the matrix A.

Theorem 7.21. *Suppose $A \in \mathbb{R}^{n\times n}$ and $A = A^T$. Then*

(1) If λ is an eigenvalue of A then $\lambda \in \mathbb{R}$.

(2) If $\lambda_1 \neq \lambda_2$ are two eigenvalues of A with eigenvectors $\mathbf{v}_1, \mathbf{v}_2$ then $\mathbf{v}_1^T\mathbf{v}_2 = 0$, i.e., eigenvectors from different eigenvalues are orthgonal.

Proof. To prove (1) suppose that $A\mathbf{v} = \lambda\mathbf{v}$. Then since A is real if we take the complex conjugate we arrive at

$$A\overline{\mathbf{v}} = \bar{\lambda}\overline{\mathbf{v}}.$$

This then implies that $\bar{\lambda}$ is also an eigenvalue of A with eigenvector λ. Now, consider the two sequences of equalities

$$\mathbf{v}^T A\overline{\mathbf{v}} = \mathbf{v}^T\bar{\lambda}\overline{\mathbf{v}} = \bar{\lambda}\mathbf{v}^T\overline{\mathbf{v}},$$
$$\mathbf{v}^T A^T\overline{\mathbf{v}} = (A\mathbf{v}^T)\overline{\mathbf{v}} = \lambda\mathbf{v}^T\overline{\mathbf{v}}.$$

Then since

$$\mathbf{v}^T\overline{\mathbf{v}} = \sum_{j=1}^{n} |v_j|^2 \neq 0,$$

we can conclude that $\lambda = \bar{\lambda}$, i.e., $\lambda \in \mathbb{R}$. For (2), consider

$$A\mathbf{v}_1 = \lambda_1\mathbf{v}_1, \quad A\mathbf{v}_2 = \lambda_2\mathbf{v}_2.$$

Then we compute

$$\lambda_1(\mathbf{v}_1^T\mathbf{v}_2) = (A\mathbf{v}_1)^T\mathbf{v}_2 = \mathbf{v}_1A^T\mathbf{v}_2 = \mathbf{v}_1^TA\mathbf{v}_2 = \lambda_2(\mathbf{v}_1^T\mathbf{v}_2).$$

This implies that $0 = \lambda_1(\mathbf{v}_1^T\mathbf{v}_2) - \lambda_2(\mathbf{v}_1^T\mathbf{v}_2)$, implying that $\mathbf{v}_1^T\mathbf{v}_2 = 0$.

7.3.2 ▪ Spectral factorization

Definition 7.22. *A matrix $B \in \mathbb{R}^{n\times n}$ is* similar *to A if there exists an invertible matrix V such that*

$$AV = VB.$$

Theorem 7.23. *If A and B are similar then they have the same eigenvalues.*

Proof. It suffices to show that A and B have the same characteristic polynomial. Indeed

$$A - \lambda I = VBV^{-1} - \lambda VV^{-1} = V(B - \lambda I)V^{-1}$$

so that $\det(A - \lambda I) = \det V \det(B - \lambda I) \det V^{-1} = \det(B - \lambda I).s$

Definition 7.24. *$A \in \mathbb{R}^{n\times n}$ is diagonalizable if it is similar to a diagonal matrix:*

$$AV = VD, \quad D = \begin{bmatrix} \lambda_1 & & & & \\ & \lambda_2 & & & \\ & & \lambda_3 & & \\ & & & \ddots & \\ & & & & \lambda_n \end{bmatrix},$$

and the factorization $A = VDV^{-1}$ is called a spectral factorization of A.

This relation implies a lot about D and V: If we look at it column by column we find

$$\begin{aligned} AV &= A\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n\end{bmatrix} \\ &= \begin{bmatrix}A\mathbf{v}_1 & A\mathbf{v}_2 & \cdots & A\mathbf{v}_n\end{bmatrix} = \begin{bmatrix}\lambda_1\mathbf{v}_1 & \lambda_2\mathbf{v}_2 & \cdots & \lambda_n\mathbf{v}_n\end{bmatrix}. \end{aligned}$$

This implies that the columns of V are eigenvectors and the diagonal entries of D are eigenvalues (as we already knew from Theorem 7.23).

We demonstrate the spectral factorization with MATLAB. Compute V and D:

```
1  A = [1, 3; 4, 1]; % a non-symmetric matrix
2  [V,D] = eig(A)
```

```
V =

    0.6547   -0.6547
    0.7559    0.7559


D =

    4.4641         0
         0   -2.4641
```

And then check that this gives a spectral factorization:

```
1 V*D*inv(V)
```

```
ans =

   1.000000000000000   3.000000000000000
   4.000000000000000   1.000000000000000
```

Example 7.25. The reader is also invited to check, based on Example 7.11, that if

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 1 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

then

$$A = VDV^{-1}.$$

7.3.3 ▪ Computing eigenvalues

For deep mathematical reasons, the numerical computation of eigenvalues has to be iterative. We will focus on only one algorithms here. The confusingly-named QR eigenvalue algorithm is driven by the following observation. Suppose $A^{(0)} = QR$. Then consider $A^{(1)} = RQ$. Then

$$A^{(1)} = RQ = Q^T A^{(0)} Q.$$

This implies that the eigenvalues of $A^{(1)}$ are the same as those of $A^{(0)}$. We can continue. Factor $A^{(1)} = Q_2 R_2$ and define $A^{(2)} = R_1 Q_1$. The process can be continued and while we do not discuss the general conditions for convergence here, if the matrix A is symmetric, it will tend to a diagonal matrix.

Here we use the notation that $[Q, R] = \texttt{QR}(A)$ gives the QR factorization of a matrix A.

Algorithm 16: QR eigenvalue algorithm `QReig`

Result: An approximation $\lambda_1, \lambda_2, \ldots, \lambda_n$ to within ϵ of the eigenvalues of A, or failure

Input: $A \in \mathbb{R}^{m \times n}$, $A^T = A$, a maximum number of iterations K and an error tolerance ϵ

begin
  set $A^{(0)} = A$;
  for $k = 0$ *to* $K - 1$ **do**
    set $[Q, R] = \texttt{QR}(A^{(k)})$;
    set $A^{(k+1)} = RQ$;
    if $\sum_{\substack{j=1 \\ j \neq i}} |a_{ij}^{(k+1)}| < \epsilon$ *for each* $i = 1, 2, \ldots, n$ **then**
      return $\text{diag}(A^{(k+1)})$;
    end
  end
  return *Tolerance not met*;
end

The condition used in the if statement within this algorithm is checking if the radii of all the Gershgorin disks is less than ϵ. If so, this means that the diagonal entries of $A^{(k+1)}$ must each be within ϵ of a true eigenvalue of A! We can guarantee accuracy... or failure. The following theorem gives us confidence that the above process will work.

Theorem 7.26. *Suppose $A = A^T$, $A \in \mathbb{R}^{n\times n}$. Let $A^{(k)}$, $k = 0, 1, 2, \ldots$ be the iterates of the QR eigenvalue algorithm with $A^{(0)} = A$. Then*

$$\lim_{k\to\infty} A^{(k)} = D = \begin{bmatrix} \lambda_1 & & & & \\ & \lambda_2 & & & \\ & & \lambda_3 & & \\ & & & \ddots & \\ & & & & \lambda_n \end{bmatrix}.$$

Furthermore, if

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}^{(k)}| < \epsilon,$$

for every $i = 1, 2, \ldots, n$ then the diagonal entries of $A^{(k)}$ must each be within ϵ of a true eigenvalue of A.

We now demonstrate this algorithm in MATLAB. We start with the matrix from Example 7.5.

```
A = [3, 1; 1, 3]
[Q,R] = qr(A);
A = R*Q
[Q,R] = qr(A);
A = R*Q
[Q,R] = qr(A);
A = R*Q
```

```
A =

     3     1
     1     3


A =

    3.6000   -0.8000
   -0.8000    2.4000


A =

    3.8824    0.4706
    0.4706    2.1176
```

```
A =

    3.9692   -0.2462
   -0.2462    2.0308
```

This is already close to the eigenvalues which are $\lambda = 2, 4$. We run it for longer:

```
format long
A = [3, 1; 1, 3];
for i = 1:30
    [Q,R] = qr(A);
    A = R*Q;
end
A
```

```
A =

   4.000000000000000   0.000000001862646
   0.000000001862645   2.000000000000000
```

7.4 ▪ Singular value decomposition

While eigenvalues only exist for square matrices, all non-zero matrices have at least one singular value.

Theorem 7.27. *Suppose $A \in \mathbb{R}^{m\times n}$. Then there exists an $m \times m$ matrix U, an $m \times n$ matrix Σ and an $n \times n$ matrix V such that*

$$A = U\Sigma V^T$$

where U, V are orthogonal and Σ is diagonal with non-negative diagonal entries.

Definition 7.28. *The factorization $A = U\Sigma V^T$ is called a singular value decomposition (SVD) of A and the non-zero diagonal entries of Σ are called the singular values of A.*

Our convention for Σ is that it is of the form

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & & & \\ & \sigma_2 & & & & & \\ & & \ddots & & & & \\ & & & \sigma_k & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \\ 0 & \cdots & & & & \cdots & 0 \\ \vdots & & & & & & \vdots \\ 0 & \cdots & & & & \cdots & 0 \end{bmatrix}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0$. The following gives an important method to compute the singular values.

Theorem 7.29. *The singular value of A are the square roots of the non-zero eigenvalues of $A^T A$.*

Our two main uses for the SVD are given in the following two sections. In the third (optional) section we use the SVD to prove that limit as $\lambda \to 0^+$ of the solution of a Tikhonov regularized, underdetermined, linear system gives the minimal norm solution. In the last (optional) section we use the singular values to derive a bound on the convergence rate of gradient descent.

7.4.1 ▪ Condition numbers and rounding errors

Definition 7.30. *The condition number for an $m \times n$ matrix A is*

$$\operatorname{cond}(A) = \frac{\sigma_1}{\sigma_{\min(m,n)}},$$

provided that A has $\min(m, n)$ singular values. Otherwise the condition number is taken to be infinite.

Definition 7.31. *The 2-norm of a matrix is defined to be*

$$\|A\|_2 = \sigma_1.$$

An important application of the condition number is that it measure how many digits of accuracy you expect to lose when solving $A\mathbf{x} = \mathbf{b}$ via the relation

$$\underbrace{\frac{\|\tilde{\mathbf{x}} - \mathbf{x}\|_2}{\|\mathbf{x}\|_2}}_{\text{relative error}} \approx \underbrace{2.2 \times 10^{-16}}_{\text{machine precision}} \operatorname{cond}(A).$$

In the following MATLAB code we demonstrate that a condition number on the order of 10^{16} causes complete loss of accuracy when solving the normal equations for a Tikhonov regularized problem $A\mathbf{x} \approx \mathbf{b}$ with $\lambda = 10^{-16}$.

```
r = RandStream('mt19937ar','Seed',1234);
A = r.randn(6,10);
At = [1e-8*eye(10); A];
```

We choose $\mathbf{b}$ so that we know the exact solution of the enlarged system.

```
x_true = ones(10,1);
bt = At*x_true;
```

Now solve

```
[LU,P] = GEpp(At'*At);
x = applyP(P,At'*bt);
x = Forsub([tril(LU,-1) + eye(10),x]);
x = Backsub([triu(LU),x])
```

```
x =

    0.4157
   14.3963
   -3.5763
    3.8573
   10.3228
    3.7279
   -3.0940
   24.0897
  -10.7333
  -11.7064
```

If the solution was fully accurate the following should be approximately 2.2×10^{-16}

```
norm(x-x_true)/norm(x_true)
```

```
ans =

   10.7344
```

This loss of accuracy is due to the large condition number.

```
cond(At'*At)
```

```
ans =

   3.2503e+17
```

On the other hand, if we use Householder QR we encounter a smaller condition number and get correspondingly better results.

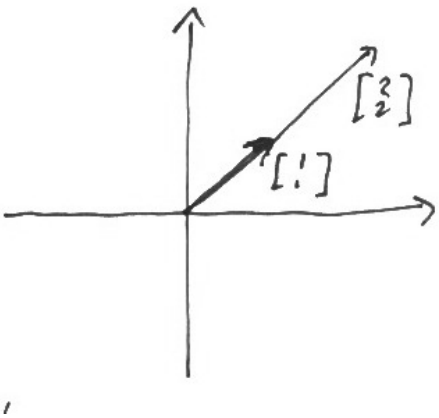
```
cond(At)
n = size(At,2);
[¬,R] = House([At,bt]);
x = Backsub(R(1:n,1:n+1));
norm(x-x_true)/norm(x_true)
```

```
ans =

   5.0409e+08


ans =

   3.2325e-08
```

7.4.2 ▪ Image compression

Consider the hand-drawn image of coordinate axes.

```
A = imread('demo.png');
A = rgb2gray(A);
imshow(A)
```



Since the black and white image is just a grid of pixes containing gray values, it is actually a matrix and we can compute the SVD of the image.

```
[U,S,V] = svd(double(A));
S(1:5,1:5)
```

```
ans =

   1.0e+04 *
```

```
8.4321         0         0         0         0
     0    0.2951         0         0         0
     0         0    0.2397         0         0
     0         0         0    0.2186         0
     0         0         0         0    0.2051
```

The first singular value is much larger the the second. So, we see what happens if we reconstruct the image using just the first singular value.

```
k = 1; % A = U S V^T
W = V';
[m,n] = size(A);
B =  U(:,1:k)*S(1:k,1:k)*W(1:k,:);
B = uint8(B);
imshow(B)
fprintf('Compression ratio = %.5f \n', (m*k+k+n*k)/(m*n))
```

```
Compression ratio = 0.00598
```

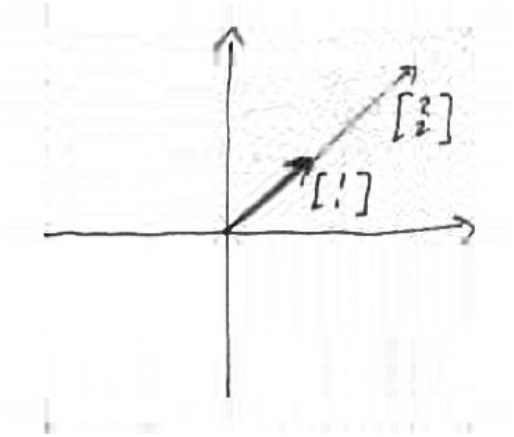
The computed compression ratio here is the ratio of storage required after throwing away all but one singular value (only retaining the first columns of U and V too) compared to the storage required to encode the original image. But we see that the form of the axes is actually captured with just one singular value! If we want to view more of the image we need to include more singular values.

```
k = 10; % A = U S V^T
W = V';
[m,n] = size(A);
B =  U(:,1:k)*S(1:k,1:k)*W(1:k,:);
B = uint8(B);
imshow(B)
fprintf('Compression ratio = %.5f \n', (m*k+k+n*k)/(m*n))
```

```
Compression ratio = 0.05977
```

```
k = 30; % A = U S V^T
W = V';
[m,n] = size(A);
B =  U(:,1:k)*S(1:k,1:k)*W(1:k,:);
B = uint8(B);
imshow(B)
fprintf('Compression ratio = %.5f \n', (m*k+k+n*k)/(m*n))
```

```
Compression ratio = 0.17932
```



7.4.3 ▪ Analysis of Tikhonov regularization

To come!

7.4.4 ▪ Convergence of gradient descent

To come!

Part II

Approximation theory

Chapter 8

Interpolation

To come!

Appendix A

Rounding errors

sddsfsdf

Bibliography

[OS18] P J Olver and C Shakiban, *Applied Linear Algebra*, vol. 30, Undergraduate Texts in Mathematics, no. 12, Springer International Publishing, Cham, 2018.