Math 170A Notes

Introduction to Numerical Analysis: Linear Algebra

Winter 2023

Taught by Professor Shuang Liu

Table of Contents

1 Matrix Multiplication (Section 1.1)

Consider an $n \times m$ matrix, or a matrix with n rows and m columns:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix}.$$

The entries of A might be real or complex numbers although, for now, we'll assume they're real. Suppose we have an m-tuple (or vector) of real numbers:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}.$$

1.1 Multiplying Matrix by Vector

Suppose A is an $n \times \boxed{m}$ matrix and $\mathbf{x}$ is a vector with m elements (i.e., a $\boxed{m} \times 1$ matrix). Let's suppose we wanted to find $A\mathbf{x}$. There are two ways we can do this.

1.1.1 Solving as a Single Component

We can solve for

$$A\mathbf{x} = \mathbf{b},$$

where

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

Here,

$$b_i = a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{im}x_m = \sum_{j=1}^{m} a_{ij}x_j.$$

(Example.) Consider

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

We note that

$$A\mathbf{x} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

where

$$b_1 = a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = 1(7) + 2(8) + 3(9) = 50$$

and

$$b_2 = a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = 4(7) + 5(8) + 6(9) = 122.$$

We can write some code to perform this operation for us:

b $\leftarrow$ **0**
for $i = 1, \ldots, n$ **do**
  for $i = 1, \ldots, m$ **do**

$b_i \leftarrow b_i + a_{ij}x_j$
end for
end for

Note that the j-loop accumulates the inner product b_i.

1.1.2 Solving as a Formula

We can also solve for $A\mathbf{x} = \mathbf{b}$ by considering the following:

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} x_1 + \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{n2} \end{bmatrix} x_2 + \ldots + \begin{bmatrix} a_{1m} \\ a_{2m} \\ \vdots \\ a_{nm} \end{bmatrix} x_m.$$

This shows that $\mathbf{b}$ is a linear combination of the columns of A.

(Example.) Consider

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

Using this approach, we now have

$$\begin{bmatrix} 1 \\ 4 \end{bmatrix} 7 + \begin{bmatrix} 2 \\ 5 \end{bmatrix} 8 + \begin{bmatrix} 3 \\ 6 \end{bmatrix} 9 = \begin{bmatrix} 7 \\ 28 \end{bmatrix} + \begin{bmatrix} 16 \\ 40 \end{bmatrix} + \begin{bmatrix} 27 \\ 54 \end{bmatrix} = \begin{bmatrix} 50 \\ 122 \end{bmatrix}.$$

Proposition. *If $\mathbf{b} = A\mathbf{x}$, then $\mathbf{b}$ is a linear combination of the columns of A. If we let A_j denote the jth column of A, we have*

$$\mathbf{b} = \sum_{j=1}^{m} A_j x_j.$$

We can also express this as pseudocode:

$\mathbf{b} \leftarrow \mathbf{0}$
for $j = 1, \ldots, m$ **do**
$\mathbf{b} \leftarrow \mathbf{b} + A_j x_j$
end for

If we use a loop to perform each vector operation, the code becomes:

$\mathbf{b} \leftarrow \mathbf{0}$
for $j = 1, \ldots, m$ **do**
for $i = 1, \ldots, n$ **do**
$b_i \leftarrow b_i + a_{ij}x_j$
end for
end for

Notice how the loops for rows and columns are interchangeable.

1.2 Flop Counts

We note that real numbers are normally stored in computers in a floating-point format. The arithmetic operations that a computer performs on these numbers are called *floating-point operations*, or *flops*. So,

$$b_i \leftarrow b_i + a_{ij}x_j$$

involves two flops: one floating-point multiply and one floating-point add. **Essentially**, we would like to count the number of operations on real numbers, or the number of addition, subtraction, multiplication, and division operations, within our program.

(Example.) Consider

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}.$$

Let's compute the inner product[a]:

$$\begin{aligned} \langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v}^T \cdot \mathbf{w} &= \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \\ &= v_1 \cdot w_1 + v_2 \cdot w_2 + \ldots + v_n \cdot w_n \\ &= \sum_{i=1}^{n} v_i w_i. \end{aligned}$$

There are $n-1$ addition operations and n multiplication operations, for a total of $2n-1$ flop count.

[a] A generalization of the dot product

1.2.1 Big-O Notation

We note that $\mathbf{v}^T\mathbf{w}$, where $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ needs $2n$ flops. This is the same thing as saying that the operation takes $\mathcal{O}(n)$ time as $n \mapsto \infty$. This means that we can disregard the implied constant, which is 2 in this case. The reason why we can do this is because, as n grows larger, the constant doesn't really matter all that much.

For instance, if n is large, then $\mathcal{O}(n)$ is faster than $\mathcal{O}(n^2)$, and $\mathcal{O}(n^2)$ is faster than $\mathcal{O}(n^3)$.

2 Systems of Linear Equations (Section 1.2)

This section is mainly going to be reviewing systems of linear equations.

2.1 Nonsingularity and Uniqueness of Solutions

Suppose we have a system of n linear equations and n unknowns

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n. \end{aligned} \tag{1}$$

We're given the coefficients a_{ij} and b_i, and we want to find $x_1, \ldots, x_n$ that satisfies the equations. Generally, it's tedious to write (1) over and over again, so we might write it as a single matrix equation

$$A\mathbf{x} = \mathbf{b}, \tag{2}$$

where

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

In other words, we are given A and $\mathbf{b}$ and must solve for $\mathbf{x}$. $A \in \mathbb{R}^{n\times n}$ is an $n \times n$ matrix, also known as a *square matrix.*

Note that (2) has a unique solution if and only if the matrix A is nonsingular.

Theorem 2.1

Let A be a square matrix. The following six conditions are equivalent; that is, if any one holds, they all hold:

(a) A^{-1} exists.

(b) There is no nonzero $\mathbf{y}$ such that $A\mathbf{y} = \mathbf{0}$.

(c) The columns of A are linearly independent.

(d) The rows of A are linearly independent.

(e) $\det(A) \neq 0$.

(f) Given any vector b, there is exactly one vector $\mathbf{x}$ such that $A\mathbf{x} = \mathbf{b}$.

Remark: Existence and uniqueness of $\mathbf{x}$ only depends on A and not on $\mathbf{b}$.
To briefly review the inverse of a matrix, we should note that the $n \times n$ **identity matrix** is denoted by I, and is the unique matrix such that

$$AI = IA = A$$

for all $A \in \mathbb{R}^{n\times x}$.The identity matrix has `1`'s on its main diagonal and `0`'s everywhere else. For example, the 3×3 identity matrix has the form

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

Given a matrix A, if there is a matrix B such that $AB = BA = I$, the B is called the inverse of A and is denoted A^{-1}. *Not every matrix will have an inverse.*

In any case, if the conditions of Theorem 2.1 hold, A is said to be nonsingular or invertible. If the conditions do *not* hold, then A is said to be singular or noninvertible. In this case, (2) will have no solution or infinitely many solutions.

Now, we'll focus on when A is nonsingular. In this case, the unique solution of 2 can be obtained by multiplying both sides by A^{-1}. We can see this from how

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

While this is guaranteed to solve the problem, doing so is a bad idea as it's very error-prone. It's usually a better idea to solve $A\mathbf{x} = \mathbf{b}$. It should also be noted that computing A^{-1} is computationally expensive.

Note: We can use MATLAB to solve for $\mathbf{x}$ by using

```
x = A \ b
```

2.2 Numerical (Approximate) Solution of Differential Equations

In this section, we'll focus on ordinary differential equations (ODE). These contain one or more functions of one independent variable and the derivatives of those functions. For example, consider the initial value system

$$\begin{cases} u'(x) + au(x) = f(x) \\ u(0) = 0 \end{cases} \tag{3}$$

with $x \in [0, 1]$. We have two functions, $f(x)$ and $u(x)$, derivative $u'(x)$, and one independent variable x. Suppose a fixed $a \in \mathbb{R}$ and $f : \mathbb{R} \mapsto \mathbb{R}$. Our goal is to find a function u satisfying the system (3).

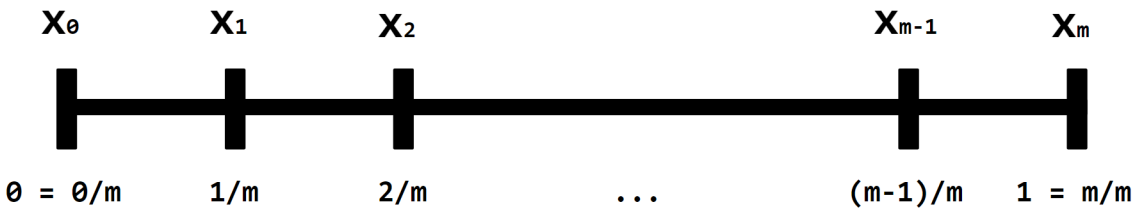
1. First, approximate u'.

$$u'(x) = \lim_{h \mapsto 0} \frac{u(x+h) - u(x)}{h} = \lim_{h \mapsto 0} \frac{u(x) - u(x-h)}{h} = \lim_{h \mapsto 0} \frac{u(x+h) - u(x-h)}{2h}.$$

   We want to pick a very small h to do this evaluation. That way, for a small enough h (close to 0), we can do

$$u'(x) \approx \frac{u(x+h) - u(x)}{h} \approx \frac{u(x) - u(x-h)}{h} \approx \frac{u(x+h) - u(x-h)}{2h} \tag{4}$$

2. To choose a small h, we can divide the given interval, which in this case is $[0, 1]$, into m subintervals. Pick a (possibly large) $m \in \mathbb{N}$, and then subdivide the interval into m equal subintervals of length $h = \frac{1}{m}$.



   The subdivision points of the intervals would be $x_i = \frac{i}{m}$ for $i = 0, 1, 2, \ldots, m$. That way, we can find u on those points; that is, we can find

$$u\left(x_i = \frac{i}{m}\right).$$

   With this in mind, the approximation formula from (4) can be rewritten as such:

$$u'(x_i) \approx \frac{u(x_i + h) - u(x_i)}{h} \approx \frac{u(x_i) - u(x_i - h)}{h} \approx \frac{u(x_i + h) - u(x_i - h)}{2h},$$

   or just

$$u'(x_i) \approx \frac{u(x_{i+1}) - u(x_i)}{h} \approx \frac{u(x_i) - u(x_{i-1})}{h} \approx \frac{u(x_{i+1}) - u(x_{i-1})}{2h}.$$

   The approximation used depends on the information you are given.

3. Now, we can set up a linear system. We want to find $u(x_i)$ for all $i \in [0, m]$.

   - For $i = 0$, it's clear that

$$u(x_i) = u(x_0) = u(0) = 0$$

     from the given condition.

- For $i = 1$, we can use the approximation

$$u'(x_i) \approx \frac{u(x_i) - u(x_{i-1})}{h}$$

to get

$$\begin{aligned}
&\frac{u(x_1) - u(x_0)}{h} + au(x_1) = f(x_1) \\
\implies &\frac{u(x_1) - u(0)}{\frac{1}{m}} + au(x_1) = f(x_1) \\
\implies &mu(x_1) + au(x_1) = f(x_1) \\
\implies &mu\left(\frac{1}{m}\right) + au\left(\frac{1}{m}\right) = f\left(\frac{1}{m}\right) \\
\implies &u\left(\frac{1}{m}\right)(m + a) = f\left(\frac{1}{m}\right) \\
\implies &u\left(\frac{1}{m}\right) = \frac{f\left(\frac{1}{m}\right)}{m + a}
\end{aligned}$$

- For $i = 2$, we can use the approximation

$$u'(x_i) \approx \frac{u(x_i) - u(x_{i-1})}{h}$$

to get

$$\begin{aligned}
&\frac{u(x_2) - u(x_1)}{h} + au(x_2) = f(x_2) \\
\implies &\frac{u(x_2) - u(x_1)}{\frac{1}{m}} + au(x_2) = f(x_2) \\
\implies &m\left(u(x_2) - u(x_1)\right) + au(x_2) = f(x_2) \\
\implies &mu(x_2) - mu(x_1) + au(x_2) = f(x_2) \\
\implies &u(x_2)(m + a) - mu(x_1) = f(x_2) \\
\implies &u(x_2)(m + a) = mu(x_1) + f(x_2) \\
\implies &u(x_2) = \frac{mu(x_1) + f(x_2)}{m + a} \\
\implies &u\left(\frac{2}{m}\right) = \frac{mu\left(\frac{1}{m}\right) + f\left(\frac{2}{m}\right)}{m + a}.
\end{aligned}$$

Note that we found $u\left(\frac{1}{m}\right)$ in the previous step.

- For $i = m$, we have

$$\begin{aligned}
&\frac{u(x_m) - u(x_{m-1})}{h} + au(x_m) = f(x_m) \\
\implies &\frac{u(1) - u(x_{m-1})}{\frac{1}{m}} + au(1) = f(1) \\
\implies &m(u(1) - u(x_{m-1})) + au(1) = f(1) \\
\implies &mu(1) - mu(x_{m-1}) + au(1) = f(1) \\
\implies &u(1)(m + a) - mu(x_{m-1}) = f(1) \\
\implies &u(1)(m + a) = mu(x_{m-1}) + f(1) \\
\implies &u(1) = \frac{mu(x_{m-1}) + f(1)}{m + a} \\
\implies &u(1) = \frac{mu(\frac{m-1}{m}) + f(1)}{m + a}.
\end{aligned}$$

With this all in mind, we want to be able to generate a linear system of the form

$$\begin{cases} u\left(\frac{1}{m}\right) = \frac{f\left(\frac{1}{m}\right)}{m+a} \\ u\left(\frac{2}{m}\right) = \frac{mu\left(\frac{1}{m}\right)+f\left(\frac{2}{m}\right)}{m+a} \\ \vdots \\ u(1) = \frac{mu(\frac{m-1}{m})+f(1)}{m+a} \end{cases}$$

$$\implies \begin{cases} (m+a)u\left(\frac{1}{m}\right) = f\left(\frac{1}{m}\right) \\ (m+a)u\left(\frac{2}{m}\right) = mu\left(\frac{1}{m}\right) + f\left(\frac{2}{m}\right) \\ \vdots \\ (m+a)u(1) = mu(\frac{m-1}{m}) + f(1) \end{cases}$$

$$\implies \begin{cases} (m+a)u\left(\frac{1}{m}\right) = f\left(\frac{1}{m}\right) \\ -mu\left(\frac{1}{m}\right) + (m+a)u\left(\frac{2}{m}\right) = f\left(\frac{2}{m}\right) \\ \vdots \\ -mu(\frac{m-1}{m}) + (m+a)u(1) = f(1) \end{cases}.$$

Representing it with matrices, we have

$$\begin{bmatrix} m+a & 0 & 0 & \dots & 0 \\ -m & m+a & 0 & \dots & 0 \\ 0 & -m & m+a & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & m+a \end{bmatrix} \begin{bmatrix} u(x_1) \\ u(x_2) \\ u(x_3) \\ \vdots \\ u(x_m) \end{bmatrix} = \begin{bmatrix} f\left(\frac{1}{m}\right) \\ f\left(\frac{2}{m}\right) \\ f\left(\frac{3}{m}\right) \\ \vdots \\ f\left(\frac{m}{m}\right) \end{bmatrix}.$$

Remarks:

- Note that if you choose a different approximation, you will get a different left-hand and right-hand side.
- A larger m gives a better approximation, but the resulting linear system is larger as well.

Ultimately, the goal here is to solve the linear system $A\mathbf{x} = \mathbf{b}$ to find $u(x_1), u(x_2), \dots, u(x_m)$. This is the approximation to u.

2.3 Solving Diagonal Systems

Let's start with the simplest kind of linear system to solve.

$$\begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 \\ 0 & a_{22} & 0 & \dots & 0 \\ 0 & 0 & a_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}.$$

Writing this as a system of linear equations gives

$$\begin{cases} a_{11}x_1 + 0x_2 + \dots + 0x_n = b_1 \\ 0x_1 + a_{22}x_2 + \dots + 0x_n = b_2 \\ \vdots \\ 0x_1 + 0x_2 + \dots + a_{nn}x_n = b_n \end{cases} \implies \begin{cases} a_{11}x_1 = b_1 \\ a_{22}x_2 = b_2 \\ \vdots \\ a_{nn}x_n = b_n \end{cases}.$$

The solution is just

$$x_1 = \frac{b_1}{a_{11}} \qquad x_2 = \frac{b_2}{a_{22}} \qquad \dots \qquad x_n = \frac{b_n}{a_{nn}}.$$

We can write an algorithm to solve this as well.

x $\leftarrow$ **0**
for $i = 1, \dots, n$ **do**
  $x_i \leftarrow \frac{\mathbf{b}_i}{A_{ii}}$
end for

From this, it follows that the flop count is just n, and its Big-O is $\mathcal{O}(n)$.

3 Triangular Systems (Section 1.3)

Generally, it's common practice to reduce general systems down to a triangular form, generally through a process known as Gaussian elimination. They are also easy to solve, and can be solved inexpensively.

3.1 Lower and Upper Triangular Matrices

We say that $L \in \mathbb{R}^{n\times n}$ is a **lower triangular** matrix if $\ell_{ij} = 0$ whenever $i < j$. Thus, a lower triangular matrix has the form

$$A = \begin{bmatrix} \ell_{11} & 0 & 0 & 0 & 0 & \dots & 0 \\ \ell_{21} & \ell_{22} & 0 & 0 & 0 & \dots & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} & 0 & 0 & \dots & 0 \\ \ell_{41} & \ell_{42} & \ell_{43} & \ell_{44} & 0 & \dots & 0 \\ \ell_{51} & \ell_{52} & \ell_{53} & \ell_{54} & \ell_{55} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \ell_{n4} & \ell_{n5} & \dots & \ell_{nn} \end{bmatrix}.$$

Similarly, we say that $U \in \mathbb{R}^{n\times n}$ is an **upper triangular** matrix if $u_{ij} = 0$ whenever $i > j$; they have the form

$$A = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & u_{24} & u_{25} & \dots & u_{2n} \\ 0 & 0 & u_{33} & u_{34} & u_{35} & \dots & u_{3n} \\ 0 & 0 & 0 & u_{44} & u_{45} & \dots & u_{4n} \\ 0 & 0 & 0 & 0 & u_{55} & \dots & u_{5n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix}.$$

A **triangular matrix** is one that is either upper or lower triangular.

(Example.) Consider the following matrix

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

This is a **lower triangular matrix**.

Remarks:

- We can have 0's in the places where there are normally nonzero numbers. This does not violate the definition of a lower- or upper-triangular matrix.
- Because of this, we can say that a diagonal matrix is both a lower- and upper-triangular matrix.

3.2 Uniqueness of Solution

When is there a unique system to $L\mathbf{x} = \mathbf{b}$ or $U\mathbf{x} = \mathbf{b}$?

> A triangular system (lower or upper) has a unique solution if and only if *all diagonal entries* are non-zero.

In fact, as long as the *determinant* of the triangular matrix A is not 0, there will be a unique solution. With triangular matrices, computing the determinant is easy: we just need to multiply all the elements on the *diagonal* together. More technically, as long as

$$\det(A) = a_{11} \cdot a_{22} \cdot a_{33} \cdot \ldots \cdot a_{nn} \neq 0,$$

then $A\mathbf{x} = \mathbf{b}$ will be a unique solution.

3.3 Solving Lower Triangular Systems: Forward Substitution

Suppose we want to solve the following system

$$\begin{bmatrix} \ell_{11} & 0 & 0 & \ldots & 0 \\ \ell_{21} & \ell_{22} & 0 & \ldots & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \ldots & \ell_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}.$$

Notice that

$$\ell_{11}x_1 = b_1 \implies x_1 = \frac{b_1}{\ell_{11}}.$$

Also notice that

$$\ell_{21}x_1 + \ell_{22}x_2 = b_2 \implies \ell_{22}x_2 = b_2 - \ell_{21}x_1 \implies x_2 = \frac{b_2 - \ell_{21}x_1}{\ell_{22}}.$$

And then notice that

$$\ell_{31}x_1 + \ell_{32}x_2 + \ell_{33}x_3 = b_3 \implies \ell_{33}x_3 = b_3 - \ell_{31}x_1 - \ell_{32}x_2 \implies x_3 = \frac{b_3 - \ell_{31}x_1 - \ell_{32}x_2}{\ell_{33}}.$$

Notice how we started off with a simple linear equation, which gave us the answer for x_1, and then we can use x_1 to find x_2 easily in the next equation, and so on. This algorithm is known as **forward substitution**. For $i = 1, \ldots, n$, it makes use of the formula

$$\boxed{x_i = \frac{b_i - \ell_{i1}x_1 - \ell_{i2}x_2 - \ldots - \ell_{i,i-1}x_{i-1}}{\ell_{ii}} = \frac{b_i - \sum_{j=1}^{i-1} \ell_{ij}x_j}{\ell_{ii}}.}$$

Roughly speaking, the algorithm looks like the following:

for $i = 1, \ldots, n$ **do**
  for $j = 1, \ldots, i-1$ **do**
    $x_i \leftarrow x_i - \ell_{ij}x_j$
  end for
  if $g_{ii} = 0$ **then**
    Set Error Flag, Exit
  end if
  $x_i \leftarrow x_i/\ell_{ii}$
end for

3.4 Solving Upper Triangular Systems: Backwards Substitution

Suppose we have an upper triangular matrix U, vector $\mathbf{x}$, and $\mathbf{b}$ like so:

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ 0 & 0 & \dots & u_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & u_{nn} \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}.$$

Suppose we want to solve for $\mathbf{x}$ in $U\mathbf{x} = \mathbf{b}$. First, notice how

$$\begin{cases} u_{nn}x_n = b_n \\ u_{n-1,n-1}x_{n-1} + u_{n-1,n}x_n = b_{n-1} \\ u_{n-2,n-2}x_{n-2} + u_{n-2,n-1}x_{n-1} + u_{n-2,n}x_n = b_{n-2} \\ \vdots \end{cases}.$$

Solving the equations above gives

$$\begin{cases} x_n = \frac{b_n}{u_{nn}} \\ x_{n-1} = \frac{b_{n-1} - u_{n-1,n}x_n}{u_{n-1}} \\ x_{n-2} = \frac{b_{n-2} - u_{n-2,n-1}x_{n-1} - u_{n-2,n}x_n}{u_{n-2}} \\ \vdots \end{cases}$$

Generalizing this, we have

$$u_{ii}x_i + u_{i,i+1}x_{i+1} + \ldots + u_{in}x_n = b_i.$$

Solving for x_i yields

$$\boxed{x_i = \frac{b_i - (u_{i,i+1}x_{i+1} + \ldots + u_{i,n}x_n)}{u_{ii}} = \frac{b_i - \sum_{j=i+1}^{n} u_{ij}x_j}{u_{ii}}},$$

which is known as **backwards substitution**.

4 Gaussian Elimination and the LU Decomposition (Section 1.7)

We'll now consider the problem of *solving* a system of n linear equations in n unknowns

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

by Gaussian elimination. Here, we'll assume that A is invertible[1] and, as usual, $n \times n$. No special properties of A are assumed (e.g., not triangular, not symmetric, not positive definite.).

Strategy: We want to transform $A\mathbf{x} = \mathbf{b}$ into an *equivalent system* $U\mathbf{x} = \mathbf{y}$ with U being an upper triangular matrix. Then, we can use back substitution to obtain the solution.

4.1 Elementary Transformations

The following transformations can be performed on a system of linear equations, which will not change the solution. Note that we'll assume the use of matrices to represent the problem.

[1] There's a unique solution $\mathbf{x}$, go find it!

1. Add a multiple of one row to another row.
$$R_i \mapsto R_i + cR_j,$$
where c is the multiple.

2. Interchange two rows (also known as pivoting).
$$R_i \leftrightarrow R_j$$

3. Multiply a row by a non-zero scalar.
$$R_i \mapsto cR_i,$$
where c is the non-zero scalar.

These transformations will be applied to a system of the form $\begin{bmatrix} A & \mathbf{b} \end{bmatrix}$.

4.2 Applying Elementary Operations

For now, we'll talk about Gaussian Elimination (GE) without row interchanges (pivoting). For now, we'll assume that $a_{11} \neq 0$. We want to convert all entries under a_{11} to 0.

1. First, let's get rid of a_{21}. We can use the operation
$$R_2 \mapsto R_2 - \frac{a_{21}}{a_{11}} R_1$$
to do just this:
$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}}_{\mathbf{b}} \xrightarrow[R_2 \mapsto R_2 - \frac{a_{21}}{a_{11}} R_1]{\text{Type (1) operation.}} \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2^{(1)} \\ b_3 \\ \vdots \\ b_n \end{bmatrix}.$$
Note that, while we were able to get rid of a_{21}, the other entries in row 2 (a_{22}, a_{23}, and so on) were updated (hence $a_{22}^{(1)}$, $a_{23}^{(1)}$, and so on). We should also note that b_2 was updated, since each row operation affects the *entire* row, which means both the A and the $\mathbf{b}$.

2. Next, let's get rid of a_{31}. Very similarly to the previous step, we can do
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2^{(1)} \\ b_3 \\ \vdots \\ b_n \end{bmatrix} \xrightarrow[R_3 \mapsto R_3 - \frac{a_{31}}{a_{11}} R_1]{\text{Type (1) operation.}} \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & \dots & a_{3n}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(1)} \\ \vdots \\ b_n \end{bmatrix}$$

3. Suppose we want to get rid of a_{i1}. This is just
$$R_i \mapsto R_i - \frac{a_{i1}}{a_{11}} R_1$$
for $i = 3, \dots, n$. So, this gives us
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & \dots & a_{3n}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & a_{n3}^{(1)} & \dots & a_{nn}^{(1)} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(1)} \\ \vdots \\ b_n^{(1)} \end{bmatrix}.$$

Next, we want to get rid of $a_{32}^{(1)}$, $a_{42}^{(1)}$, and so on[2]. Let's assume that $a_{22}^{(1)} \neq 0$.

1. Let's get rid of a_{32}. To do this, we'll use the operation

$$R_3 \mapsto R_3 - \frac{a_{32}^{(1)}}{a_{22}^{(1)}} R_2.$$

This gives us

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & \dots & a_{3n}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & a_{n3}^{(1)} & \dots & a_{nn}^{(1)} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(1)} \\ \vdots \\ b_n^{(1)} \end{bmatrix} \xrightarrow[R_3 \mapsto R_3 - \frac{a_{32}^{(1)}}{a_{22}^{(1)}} R_2]{\text{Type (1) operation.}} \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \dots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(2)} \\ \vdots \\ b_n \end{bmatrix}.$$

2. Likewise, we can repeat this for a_{i2} using the operation

$$R_i \mapsto R_i - \frac{a_{i2}^{(1)}}{a_{22}^{(1)}} R_2$$

for $i = 4, \dots, n$. This gives us

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \dots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n3}^{(2)} & \dots & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix}.$$

We can continue this procedure until we produce an upper-triangular matrix. This upper-triangular matrix will look like

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \dots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn}^{(n-1)} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(2)} \\ \vdots \\ b_n^{(n-1)} \end{bmatrix}.$$

At the very end, we can use back substitution on the upper-triangular matrix.

Remark: We are dividing by a_{11}, $a_{22}^{(1)}$, $a_{33}^{(2)}$, and so on for a general matrix. In reality, *some of these entries could be 0.*

4.3 LU Decomposition

Theorem 4.1: LU Decomposition

Let A be an $n \times n$ matrix whose leading principal submatrices are all nonsingular. Then, A can be decomposed in exactly one way into a product

$$A = LU,$$

such that L is unit lower-triangular and U is upper triangular.

[2]We omit $a_{22}^{(1)}$ because, remember, our goal is to make our system into an upper-triangular system.

In other words,

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ \ell_{21} & 1 & 0 & \dots & 0 \\ \ell_{31} & \ell_{32} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \dots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix}.$$

(Exercise: LU Decomposition.) Let

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

Solve $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$.

$$\begin{bmatrix}1&3&4\\1&2&6\\3&5&7\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} \xrightarrow[R_2\mapsto R_2-R_1]{\text{Operation (1)}} \begin{bmatrix}1&3&4\\0&-1&2\\3&5&7\end{bmatrix}\begin{bmatrix}1\\0\\1\end{bmatrix}$$
$$\xrightarrow[R_3\mapsto R_3-3R_1]{\text{Operation (1)}} \begin{bmatrix}1&3&4\\0&-1&2\\0&-4&-5\end{bmatrix}\begin{bmatrix}1\\0\\-2\end{bmatrix}$$
$$\xrightarrow[R_3\mapsto R_3-4R_2]{\text{Operation (1)}} \begin{bmatrix}1&3&4\\0&-1&2\\0&0&-13\end{bmatrix}\begin{bmatrix}1\\0\\-2\end{bmatrix}$$

Notice that our matrix is in upper-triangular form. From here, we just need to solve

$$\begin{cases}-13x_3=-2\\-1x_2+2x_3=0\\x_1+3x_2+4x_3=1\end{cases},$$

which can be done via backwards substitution.

We performed three operations to get to the upper-triangular matrix. Each operation corresponds to a lower-triangular matrix. In particular, if we write out

$$\tilde{L_1}\tilde{L_2}\tilde{L_3}A=U,$$

where each of the L_i represents a lower-triangular matrix and corresponds to an operation, then $\tilde{L_3}$ represents the first operation, $R_2\mapsto R_2-R_1$, or $\tilde{L_3}=\begin{bmatrix}1&0&0\\-1&1&0\\0&0&1\end{bmatrix}$. Likewise, $\tilde{L_2}$ represents the second operation, $R_3\mapsto R_3-3R_1$, or $\tilde{L_2}=\begin{bmatrix}1&0&0\\0&1&0\\-3&0&1\end{bmatrix}$. Finally, $\tilde{L_1}$ represents the third operation, $R_3\mapsto R_3-4R_2$, or $\tilde{L_1}=\begin{bmatrix}1&0&0\\0&1&0\\0&-4&1\end{bmatrix}$. Notice how the positioning of the numbers corresponds to the entry that we tried to eliminate in A. For example, in L_2, we put -3 in position L_{21} because we eliminated the 3 from position A_{21} in the second operation. Therefore,

$$\begin{bmatrix}1&0&0\\0&1&0\\0&-4&1\end{bmatrix}\begin{bmatrix}1&0&0\\0&1&0\\-3&0&1\end{bmatrix}\begin{bmatrix}1&0&0\\-1&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&3&4\\1&2&6\\3&5&7\end{bmatrix}=\begin{bmatrix}1&3&4\\0&-1&2\\0&0&-13\end{bmatrix}$$

In any case, we know that

$$\underbrace{\tilde{L_1}\tilde{L_2}\tilde{L_3}}_{\tilde{L}}A=U.$$

Then,

$$\tilde{L}A=U \implies A=\tilde{L}^{-1}U=LU$$

where

$$L=\tilde{L}.$$

Remark: To see how each operation corresponds to a lower-triangular matrix, consider $R_2\mapsto R_2+5R_1$:

$$\begin{bmatrix}1&0&0\\5&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}a_{11}&a_{12}&a_{13}\\a_{21}&a_{22}&a_{23}\\a_{31}&a_{32}&a_{33}\end{bmatrix}=\begin{bmatrix}a_{11}&a_{12}&a_{13}\\5a_{11}+a_{21}&5a_{12}+a_{22}&5a_{13}+a_{23}\\a_{31}&a_{32}&a_{33}\end{bmatrix}.$$

Likewise, consider $R_3 \mapsto R_3 + 10R_2$:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 10 & 0 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 10a_{21} + a_{31} & 10a_{22} + a_{32} & a_{23} + a_{33} \end{bmatrix}.$$

5 Gaussian Elimination with Pivoting (Section 1.8)

In the previous section, we discussed Gaussian Elimination *without* row interchanges (i.e., pivoting). In this section, we will now permit row interchanges. Consider the following system,

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

Notice that this matrix has several properties:

- It's invertible ($\det(A) \neq 0$), therefore a unique solution exists.
- However, using Gaussian Elimination *without* row interchanges would fail.

However, we can resolve this by switching two rows, e.g., R1 and R3, to get

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

When deciding which two rows to switch, the rule is to use the row with the *first* entry containing the **largest absolute value**[3]. The idea is that we'll divide by first element in next step for Gaussian elimination.

5.1 Relation to the Permutation Matrix

Note taht a permutation matrix P is a matrix which only contains 0's and 1's such that each row and each column has exactly one entry equal to 1.

Example of Permutation Matrix	**<u>Not</u> a Permutation Matrix**
$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ This matrix has two 1's in the first row and last column.

How does the permutation matrix work in relation to row interchanges?

(Example.) Suppose we multiply a permutation matrix,

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

on the left.

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

This corresponds to switching R1 and R3.

[3]The reason why we choose the largest absolute value is because dividing by small numbers can cause errors, and these errors can accumulate.

(Example.) Suppose we multiply the same permutation matrix as in the previous example on the right. Then,

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

Here, this corresponds to switching C1 and C3.

5.2 PLU Decomposition

Gaussian Elimination with pivoting leads to a decomposition of the form $PA = LU$, also known as *PLU decomposition*, where

- P is a permutation matrix,
- L is a lower-triangular matrix, and
- U is an upper-triangular matrix. The PLU exists if A is invertible.

Remarks:

- The PLU exists if A is invertible.
- In MATLAB, we can run `[P, L, U] = lu(A)`, where

$$P \cdot A = L \cdot U \implies A = P^T \cdot L \cdot U.$$

5.2.1 Finding PLU Decomposition

We'll find the PLU decomposition through an example.

(Example.) Suppose we want to find the PLU decomposition of

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

- Step 1: First, let's switch R1 and R3.

$$P_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \qquad P_1 A = \begin{bmatrix} 2 & 2 & 5 \\ 1 & 3 & 4 \\ 0 & 4 & 1 \end{bmatrix}.$$

- Step 2: Next, let's perform the operation $R_2 \mapsto R_2 - \frac{1}{2}R_1$.

$$L_1 = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad L_1 P_1 A = L_1 \begin{bmatrix} 2 & 2 & 5 \\ 1 & 3 & 4 \\ 0 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 5 \\ 0 & 2 & \frac{3}{2} \\ 0 & 4 & 1 \end{bmatrix}.$$

- Step 3: Next, notice that **4** is a pivot (the 2 in the second row is smaller than 4). let's switch R_2 and R_3.

$$P_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad P_2 L_1 P_1 A = \begin{bmatrix} 2 & 2 & 5 \\ 0 & 4 & 1 \\ 0 & 2 & \frac{3}{2} \end{bmatrix}.$$

- Step 4: We can now perform the operation $R_3 \mapsto R_3 - \frac{1}{2}R_2$.

$$L_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{bmatrix}, \qquad L_2P_2L_1P_1A = \begin{bmatrix} 2 & 2 & 5 \\ 0 & 4 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Here, we find our upper-triangular matrix

$$U = L_2P_2L_1P_1A = \begin{bmatrix} 2 & 2 & 5 \\ 0 & 4 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Now that we found the upper-triangular matrix U, we need to find the decomposition. We don't know what the P or L matrices are. Notice that, with $L_2P_2L_1P_1$, we can do

$$P_2L_1 = \tilde{L_1}P_2.$$

Then,

$$L_2P_2L_1P_1 = \underbrace{L_2\tilde{L_1}}_{\tilde{L}}\overbrace{P_2P_1}^{P}.$$

Therefore, if

$$P_2L_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ -\frac{1}{2} & 1 & 0 \end{bmatrix},$$

then

$$\begin{aligned} P_2L_1 &= \tilde{L_1}P_2 \\ &\implies \tilde{L_1} = P_2L_1P_2^{-1} \\ &\implies \tilde{L_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ -\frac{1}{2} & 1 & 0 \end{bmatrix} P_2^{-1} \\ &\implies \tilde{L_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ -\frac{1}{2} & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \\ &\implies \tilde{L_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{2} & 0 & 1 \end{bmatrix}. \end{aligned}$$

Therefore, going back to

$$L_2P_2L_1P_1 = \underbrace{L_2\tilde{L_1}}_{\tilde{L}}\overbrace{P_2P_1}^{P},$$

we have

$$\tilde{L}PA = U \implies PA = \tilde{L}^{-1}U.$$

From this, we find

$$\tilde{L}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{bmatrix}.$$

Finally, defining $L = \tilde{L}^{-1}$ gives us

$$PA = LU.$$

6 Positive Definite Systems & Cholesky Decomposition (Section 1.4)

In the previous section, we learned about Gaussian Elimination with Pivoting (i.e., $A\mathbf{x} = \mathbf{b} \iff PA = LU$), which can be used to solve *any* system with invertible matrix A. However, for some matrix A with special properties, we can use a *better* way to solve it, mainly for computational efficiency.

One such class of matrices are **positive definite** ones. In particular, if A is $n \times n$, real, and satisfies the conditions

1. $A = A^T$ (i.e., symmetric), and
2. $\mathbf{x}^T A\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \langle x^T, Ax \rangle > 0$ for all nonzero $\mathbf{x} \in \mathbb{R}^n$,

then A is said to be **positive definite**.

6.1 Properties of Positive Definite Matrix

There are some useful properties of positive definite matrices.

Lemma 6.1

A positive definite matrix is invertible.

Theorem 6.1

Let M be an invertible $n \times n$ matrix. Then, $A = M^T M$ is positive definite.

Theorem 6.2: Cholesky Decomposition Theorem

Let A be positive definite. Then, there exists an upper-triangular matrix R such that

$$A = R^T R.$$

This is known as the Cholesky decomposition, and R is called the **Cholesky factor** of A. In addition, this decomposition is unique if we choose $r_{ii} > 0$ for $i = 1, 2, \dots, n$.

(Example: Cholesky Algorithm.) Suppose

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \quad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix},$$

so that

$$R^T = \begin{bmatrix} r_{11} & 0 & 0 \\ r_{21} & r_{22} & 0 \\ r_{31} & r_{32} & r_{33} \end{bmatrix}.$$

Then,

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} r_{11} & 0 & 0 \\ r_{21} & r_{22} & 0 \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix}$$
$$= \begin{bmatrix} r_{11}^2 & r_{11}r_{12} & r_{11}r_{13} \\ r_{11}r_{12} & r_{12}^2 + r_{22}^2 & r_{12}r_{13} + r_{22}r_{23} \\ r_{11}r_{13} & r_{13}r_{12} + r_{23}r_{22} & r_{13}^2 + r_{23}^2 + r_{33}^2 \end{bmatrix}$$

One thing to notice is that this matrix is symmetric. So, when finding the values of each element, we only need to find the values of specific elements (one side). We can find the value of each element in R. Starting with the first row, we have

- $r_{11}^2 = a_{11} \implies \boxed{r_{11} = \sqrt{a_{11}}}$.
- $r_{11}r_{12} = a_{12} \implies \boxed{r_{12} = \dfrac{a_{12}}{r_{11}}}$.
- $r_{11}r_{13} = a_{13} \implies \boxed{r_{13} = \dfrac{a_{13}}{r_{11}}}$.

For the second row, we have

- $r_{12}^2 + r_{22}^2 = a_{22} \implies \boxed{r_{22} = \sqrt{a_{22} - r_{12}^2}}$.
- $r_{12}r_{13} + r_{22}r_{23} = a_{23} \implies \boxed{r_{23} = \dfrac{a_{23} - r_{12}r_{13}}{r_{22}}}$.

For the third row, we have

- $r_{13}^2 + r_{23}^2 + r_{33}^2 = a_{33} \implies \boxed{r_{33} = \sqrt{a_{33} - r_{13}^2 - r_{23}^2}}$.

The above example is the Cholesky Algorithm for a 3×3 matrix. For the general case, suppose we have

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} r_{11} & 0 & 0 & \dots & 0 \\ r_{21} & r_{12} & 0 & \dots & 0 \\ r_{31} & r_{23} & r_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{1n} & r_{2n} & r_{3n} & \dots & r_{nn} \end{bmatrix}}_{R^T} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1n} \\ 0 & r_{22} & r_{23} & \dots & r_{2n} \\ 0 & 0 & r_{33} & \dots & r_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & r_{nn} \end{bmatrix}}_{R}.$$

Here,

$$a_{ij} = \begin{cases} \sum_{k=1}^{i} r_{ki}r_{kj} & i < j \\ \sum_{k=1}^{i} r_{ki}^2 & i = j \\ \sum_{k=1}^{j} r_{ki}r_{kj} & i > j \end{cases} \tag{5}$$

From this, we're able to solve r_{ii}:

$$r_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} r_{ki}^2}. \tag{6}$$

After solving for r_{ii}, we can find r_{ij}:

$$r_{ij} = \frac{\left(a_{ij} - \sum_{k=1}^{i-1} r_{ki}r_{kj}\right)}{r_{ii}}. \tag{7}$$

for $j = i+1, \ldots, n$, noting that for $j < i$, $r_{ij} = 0$.

In general, Cholesky is doing half of the LU decomposition work: $\mathcal{O}(n^3)$.

(Exercise.) Determine whether or not the following matrices have a Cholesky factorization; if they do, compute (by hand) the Cholesky factor R.

- $A = \begin{bmatrix} 1 & -2 & 0 \\ -2 & 13 & 6 \\ 0 & 6 & 5 \end{bmatrix}$

We know that if $R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix}$ is the Cholesky factor of A, then

$$r_{11} = \sqrt{1} = 1,$$
$$r_{12} = \frac{-2}{1} = -2,$$
$$r_{13} = \frac{0}{1} = 0,$$
$$r_{22} = \sqrt{13 - (-2)^2} = \sqrt{13 - 4} = \sqrt{9} = 3,$$
$$r_{23} = \frac{6 - (-2)(0)}{3} = \frac{6}{3} = 2,$$
$$r_{33} = \sqrt{5 - (0)^2 - (2)^2} = \sqrt{5 - 4} = \sqrt{1} = 1.$$

Therefore, it follows that the Cholesky factorization is

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

as desired.

- $B = \begin{bmatrix} 4 & -4 & 0 \\ -4 & 4 & 0 \\ 0 & 0 & 5 \end{bmatrix}$.

We note that

$$\det(B) = 4(4 \cdot 5 - 0) - (-4)(-4 \cdot 5 - 0) + 0 = 80 - 80 = 0.$$

From the contrapositive of the above theorem, because $\det(B) = 0$, it follows that B must not be singular and thus is not positive definite.

(Exercise.) Consider the following matrix,

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

Is it positive definite?

We can run through the Cholesky Algorithm.

$$r_{11} = \sqrt{1} = 1,$$

$$r_{12} = \frac{3}{1} = 3,$$

$$r_{13} = \frac{-5}{1} = -5,$$

$$r_{22} = \sqrt{0 - (1)^2}.$$

However, we cannot get a real number from r_{22}, so A is not positive definite.

7 Vectors and Matrix Norms (Section 2.1)

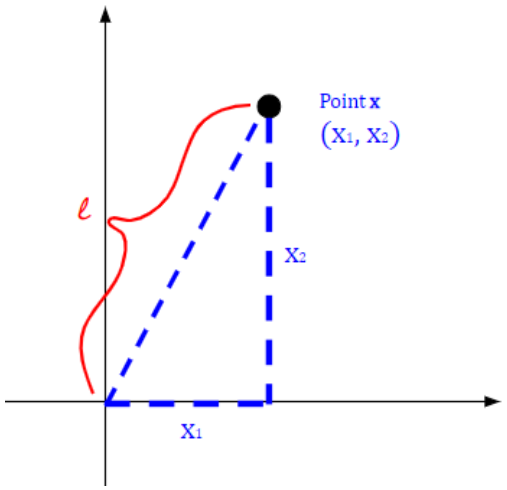
In numerical analysis, we want to find approximate solutions to problems (e.g., ODEs). Some things we want to know are

- How good the approximate solution is?
- How close is the approximate solution to the exact solution?

So, **norms** are a measure of length, or the measure of being close or far apart.

7.1 Vector Norms

The vector norm we're most familiar with is the one in $\mathbb{R}^2$, also known as the *2-norm*. These might look something like



For a point

$$\mathbf{x} = (x_1, x_2),$$

we can define the *2-norm* of a vector to be

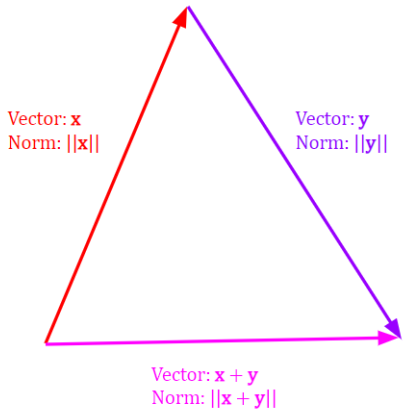
$$||\mathbf{x}||_2 = \sqrt{x_1^2 + x_2^2}.$$

Definition 7.1: Vector Norm

A **norm** of a vector (i.e., a **vector norm**) $\mathbf{x} \in \mathbb{R}^n$ is a real number $||\mathbf{x}||$ that is assigned to $\mathbf{x}$. For all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and all $c \in \mathbb{R}$, the following properties are satisfied:

1. Positive Definite Property: $||\mathbf{x}|| > 0$ for $\mathbf{x} \neq 0$ and $||\mathbf{0}|| = 0$.
2. Absolute Homogeneity: $||c\mathbf{x}|| = |c| \cdot ||\mathbf{x}||$.
3. Triangle Inequality: $||\mathbf{x} + \mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||$.

Remark: With regards to the third property, consider



Note that $||\mathbf{x} + \mathbf{y}|| = ||\mathbf{x}|| + ||\mathbf{y}||$ if $\mathbf{x}$ and $\mathbf{y}$ points at the same direction.

7.1.1 Common Norms

There are some common norms that we've seen before. As implied, they all satisfy the properties above.

- For $p \geq 1$, we define
$$||\mathbf{x}||_p = \sqrt[p]{|x_1|^p + |x_2|^p + \ldots + |x_n|^p}.$$
Note that some special cases are
$$||\mathbf{x}||_2 = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$$
and
$$||\mathbf{x}||_1 = |x_1| + |x_2| + \ldots + |x_n| = \sum_{i=1}^{n} |x_i|.$$
- The infinite norm is defined to be
$$||\mathbf{x}||_\infty = \max_{i=1,2,\ldots,n} |x_i|.$$
It should be noted that
$$\lim_{p \mapsto \infty} ||\mathbf{x}||_p = ||\mathbf{x}||_\infty.$$

(Exercise.) Consider the vector,
$$\mathbf{v} = \begin{bmatrix} 4 \\ 8 \\ 6 \end{bmatrix}.$$

- Compute $||\mathbf{v}||_1$.

$$||\mathbf{v}||_1 = |4| + |8| + |6| = 18.$$

- Compute $||\mathbf{v}||_2$.

$$||\mathbf{v}||_2 = \sqrt{4^2 + 8^2 + 6^2} = \sqrt{116}.$$

- Compute $||\mathbf{v}||_\infty$.

$$||\mathbf{v}||_\infty = \max_{i=1,2,3} |v_i| = \max\{|4|, |8|, |6|\} = 8.$$

7.2 Matrix Norms

We now want to consider norms for a matrix $A \in \mathbb{R}^{n\times n}$. There are two ways we can interpret matrix norms.

1. Interpret matrix as a vector. For example, suppose we have

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

Then, we can "convert" this matrix to a vector like so:

$$\mathbf{v} = \begin{bmatrix} -1 \\ 0 \\ 5 \\ 8 \\ 2 \\ 7 \\ -3 \\ 1 \\ 0 \end{bmatrix}.$$

Here, $\mathbf{v} \in \mathbb{R}^9$. Notice how the first column of A is the top three elements in $\mathbf{v}$, the second column of A is the middle three elements of $\mathbf{v}$, and the last column of A is the bottom three elements of $\mathbf{v}$.

2. We can also define the matrix as a linear operator. That is, for a function $L : \mathbb{R}^n \mapsto \mathbb{R}^n$, we have

$$L(\mathbf{x}) = A\mathbf{x}.$$

7.2.1 General Definition of Matrix Norms

Definition 7.2: Matrix Norm

A **matrix norm** assigns a real number $||A||$ to a matrix A. This should satisfy the following conditions for all $A, B \in \mathbb{R}^{n\times n}$ and $c \in \mathbb{R}$.

1. $||A|| > 0$ if $A \neq 0$, and $||0|| = 0$.
2. $||cA|| = |c| \cdot ||A||$.
3. $||A + B|| \leq ||A|| + ||B||$.
4. Submultiplicity: $||AB|| \leq ||A|| \cdot ||B||$.

Remark: Regarding submultiplicity, for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we have

$$|\langle x, y\rangle| \leq ||\mathbf{x}||_2 ||y||_2,$$

known as the Cauchy Schwarz inequality.

7.2.2 Vector Viewpoint

Going back to the vector viewpoint, let's suppose we have

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

Then,

$$\mathbf{v} = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \\ a_{12} \\ a_{22} \\ \vdots \\ a_{n2} \\ \vdots \\ a_{1n} \\ a_{2n} \\ \vdots \\ a_{nn} \end{bmatrix}.$$

The Frobenius norm of A is defined by

$$||A||_F = ||\mathbf{v}||_2 = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{\frac{1}{2}}.$$

7.2.3 Matrix Norm

Matrix p-norms are defined as follows:

$$||A||_p = \max_{\substack{\mathbf{x} \in \mathbb{R}^n \\ \mathbf{x} \neq \mathbf{0}}} \frac{||A\mathbf{x}||_p}{||\mathbf{x}||_p}.$$

This measures the *maximum stretch* the linear function $L(\mathbf{x}) = A\mathbf{x}$ can do to a vector (normalized by the length of the vector).

Some of the most important matrix p-norms are

- For $p = 1$,

$$||A||_1 = \max_{\mathbf{x} \neq \mathbf{0}} \frac{||A\mathbf{x}||_1}{||\mathbf{x}||_1} = \max_{j=1,2,\dots,n} \sum_{i=1}^{n} |a_{ij}|,$$

  the maximum L_1-norm of each column.

- For $p = \infty$,
$$||A||_\infty = \max_{\mathbf{x}\neq\mathbf{0}} \frac{||A\mathbf{x}||_\infty}{||\mathbf{x}||_\infty} = \max_{i=1,2,\ldots,n} \sum_{j=1}^{n} |a_{ij}|,$$
the maximum L_1-norm of each row.

- For $p = 2$:
$$||A||_2 = \max_{\mathbf{x}\neq\mathbf{0}} \frac{||A\mathbf{x}||_2}{||\mathbf{x}||_2} = \sigma_1,$$
the largest[4] singular value of matrix A.

Remark: Don't confuse $||A||_2$ and $||A||_F$. They are very different! Specifically,
$$||A||_2 = \max_{\mathbf{x}\neq\mathbf{0}} \frac{||A\mathbf{x}||_2}{||\mathbf{x}||_2}$$
while
$$||A||_F = ||\mathbf{v}||_2 = \left(\sum_{i,j} (a_{ij})^2 \right)^{\frac{1}{2}}.$$

(Exercise.) Consider the following matrix
$$A = \begin{bmatrix} 4 & 8 & 6 \\ 8 & 17 & 10 \\ 6 & 10 & 29 \end{bmatrix}.$$

- Compute $||A||_F$.

$$||A||_F = \sqrt{4^2 + 8^2 + 6^2 + 8^2 + 17^2 + 10^2 + 6^2 + 10^2 + 29^2} = \sqrt{1546}.$$

- Compute $||A||_1$.

We consider the sum of the absolute value of the entries in each column.

  - For the first column, we know that
$$\sum_{i=1}^{3} |a_{i1}| = |4| + |8| + |6| = 18.$$
  - For the second column,
$$\sum_{i=1}^{3} |a_{i2}| = |8| + |17| + |10| = 35.$$
  - For the third column,
$$\sum_{i=1}^{3} |a_{i3}| = |6| + |10| + |29| = 45.$$

Therefore,
$$||A||_1 = \max\{18, 35, 45\} = 45.$$

[4] This is related to SVD, which we will learn later.

- Compute $||A||_\infty$.

  We consider the sum of the absolute value of the entries in each row.

  - For the first row,
  $$\sum_{j=1}^{3} |a_{1j}| = |4| + |8| + |6| = 18.$$
  - For the second row,
  $$\sum_{j=1}^{3} |a_{2j}| = |8| + |17| + |10| = 35.$$
  - For the third row,
  $$\sum_{j=1}^{3} |a_{3j}| = |6| + |10| + |29| = 45.$$

  Therefore,
  $$||A||_\infty = \max\{18, 35, 45\} = 45.$$

8 The Discrete Least Squares Problem (Section 3.1)

Given points $(t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n)$, we want to find an *approximate function* to fit those points; that is,

$$p(t_i) = y_i$$

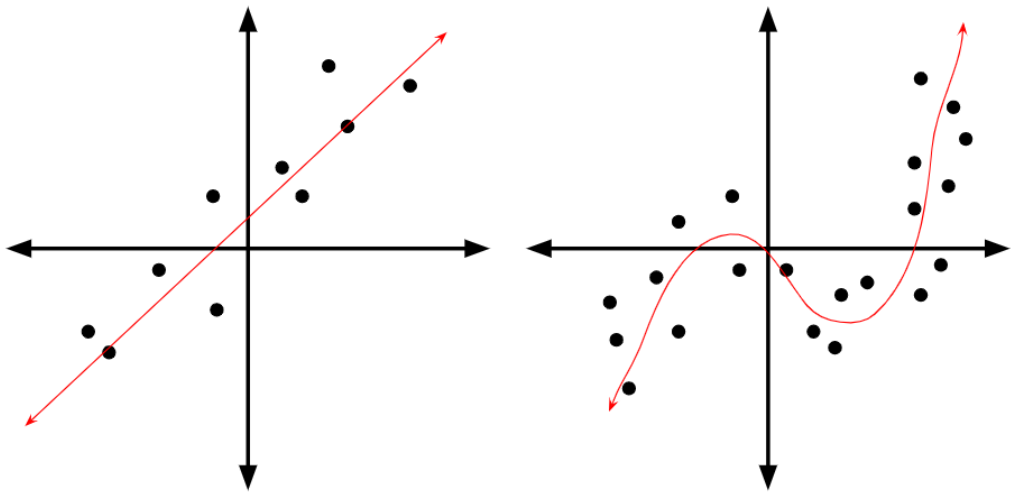
for $i = 1, 2, \ldots, n$. For example, suppose we want to find

$$p(t) = a_0 + a_1 t$$

or

$$p(t) = a_0 + a_1 t + a_2 t^2.$$

The idea is that we're just trying to find the *line of best fit* (a line with the smallest margin of error). We're not trying to find a line that passes through all the points, just the one of best fit.



The error, as mentioned, is defined by $r_i = y_i - p(t_i)$, where y_i is the exact value and $p(t_i)$ is the value of the function (the estimate). Then, the r_i's generate a vector known as the **residual** vector.

8.1 Problem Statement

Our goal is to find a $p(t) = a_0 + a_1 t$ such that $r \in \mathbb{R}^a$ is as small as possible. Hence, it turns into finding a_0, a_1 to minimize $||r||_2$.

8.1.1 Matrix Notation

Let's begin by converting

$$r_i = y_i - p(t_i) = y_i - (a_0 + a_1 t_i) = y_i - a_0 - a_1 t_i \quad i = 1, 2, \ldots, n$$

into matrix notation. This gives us

$$\underbrace{\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix}}_{\mathbf{r} \in \mathbb{R}^n} = \begin{bmatrix} y_1 - a_0 - a_1 t_1 \\ y_2 - a_0 - a_1 t_2 \\ \vdots \\ y_n - a_0 - a_1 t_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \begin{bmatrix} a_0 + a_1 t_1 \\ a_0 + a_1 t_2 \\ \vdots \\ a_0 + a_1 t_n \end{bmatrix} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{\mathbf{y} \in \mathbb{R}^n} - \underbrace{\begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_n \end{bmatrix}}_{A \in \mathbb{R}^{n \times m}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \end{bmatrix}}_{\mathbf{x} \in \mathbb{R}^m}$$

So, with the above matrix notation, also represented by

$$\mathbf{r} = \mathbf{y} - A\mathbf{x},$$

the goal is to find an $\mathbf{x} \in \mathbb{R}^m$ such that $||\mathbf{r}||_2$ is minimized. Essentially,

$$\min_{x \in \mathbb{R}^m} ||\mathbf{r}||_2 = \min_{x \in \mathbb{R}^m} ||\mathbf{y} - A\mathbf{x}||_2.$$

Here,

- n is the number of datapoints, and
- m is the number of unknowns from the function.

Remarks:

- A is not squared.
- $n > m$.

There will be no solutions[5] to $A\mathbf{x} = \mathbf{y}$. Instead, we want to find $\mathbf{x}$ that minimizes the overall error.

8.2 Other Basis Functions

Instead of linear functions $p(t) = a_0 + a_1 t$, we an try other types of functions. For example, consider polynomials, or

$$p(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_k t^k = \sum_{i=0}^{k} a_i t^i.$$

Here, $t^0 = 1$ and we have $(k+1)$ unknowns. The matrix formulation is given by

$$\underbrace{\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix}}_{r \in \mathbb{R}^n} = \begin{bmatrix} y_1 - p(t_1) \\ y_2 - p(t_2) \\ \vdots \\ y_n - p(t_n) \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} p(t_1) \\ p(t_2) \\ \vdots \\ p(t_n) \end{bmatrix} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{\mathbf{y} \in \mathbb{R}^n} - \underbrace{\begin{bmatrix} 1 & t_1 & t_1^2 & \ldots & t_1^k \\ 1 & t_2 & t_2^2 & \ldots & t_2^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & t_n^2 & \ldots & t_n^k \end{bmatrix}}_{A \in \mathbb{R}^{nn \times (k+1)}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix}}_{\mathbf{x} \in \mathbb{R}^{k+1}}.$$

Here, $\mathbf{r} = \mathbf{y} - A\mathbf{x}$ is called the *residual*. To solve this, we will use the QR decomposition.

[5] A has more rows than columns.

8.3 QR Decomposition

In essence, QR decomposition states that we can decompose a matrix A into two matrices, Q and R, such that

$$A = QR,$$

where Q is orthogonal and R is upper-triangular.

8.3.1 A Brief Review

Definition 8.1: Orthogonal Matrix

$Q \in \mathbb{R}^{n\times n}$ is called **orthogonal** if $Q^T = Q^{-1}$, i.e., $Q^TQ = QQ^T = I$.

(Example.) Consider the rotations in $\mathbb{R}^n$. Note that

$$Q = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad Q^T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

It follows that

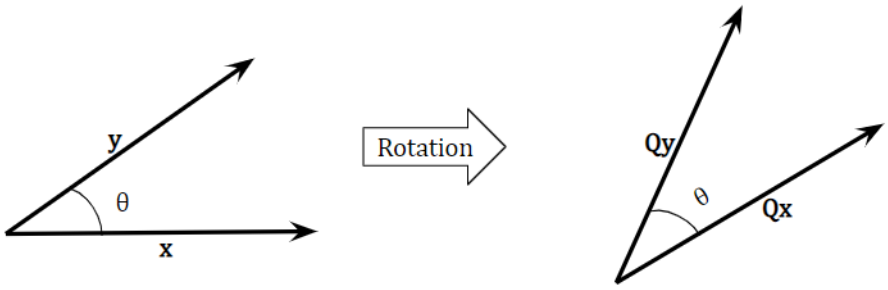
$$QQ^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I.$$

Theorem 8.1

If $Q \in \mathbb{R}^{n\times n}$ is an orthogonal matrix, then

1. $\langle Q\mathbf{x}, Q\mathbf{y}\rangle = \langle \mathbf{x}, \mathbf{y}\rangle$.
2. $||Q\mathbf{x}||_2 = ||\mathbf{x}||_2$.

It might be useful to consider this visualization:



Proof. We'll prove each part.

1. $\langle Q\mathbf{x}, Q\mathbf{y}\rangle = (Q\mathbf{x})^TQ\mathbf{y} = \mathbf{x}^TQ^TQ\mathbf{y} = \mathbf{x}^T\mathbf{y} = \langle \mathbf{x}, \mathbf{y}\rangle$.
2. $||Q\mathbf{x}||_2 = \sqrt{\langle Q\mathbf{x}, Q\mathbf{y}\rangle} = \sqrt{\langle \mathbf{x}, \mathbf{x}\rangle} = ||\mathbf{x}||_2$.

Thus, we're done. □