LINEAR ALGEBRA ESSENTIALS

FOR DATA SCIENCE

SYLVAIN BONNOT

Sylvain Bonnot

Linear Algebra Essentials for Data Science

FIRST EDITION

Leanpub

PUBLISHED BY LEANPUB

TUFTE-LATEX.GOOGLECODE.COM

First printing, May 2024

Preface

Linearity is a concept that appears everywhere in our daily lives. Consider this simple example: when you order 3 falafels and 2 salads at a restaurant, you expect to pay the total as a sum of individual prices:

$$\text{total} = \text{price}(\ 3 \text{ falafels+2 salads}) = 3 * \text{price(falafel)} + 2 * \text{price(salad)}$$

This equation shows that the total price is a *linear function* of the quantities of falafels and salads. More formally, a linear function f is a mapping from one set X (where you can add objects and multiply them by scalars [1]) to another set Y preserving addition and scalar multiplication:

[1] A *scalar* is just a number, like -3.12.

$$\begin{aligned} f(x+y) &= f(x)+f(y) && \text{"Addition is preserved"} \\ f(\alpha x) &= \alpha f(x) && \text{"Multiplication by a scalar is preserved"} \end{aligned}$$

Of course not all functions are linear. High school students sometimes think that the function $f(x) = x^2$ is linear, only to realize that it is not:

$$f(x+y) = (x+y)^2 = x^2 + 2xy + y^2 \neq x^2 + y^2 = f(x) + f(y)$$

Linear algebra is in essence the study of *linear functions*[2].

[2] We use also the expressions *linear maps* or *linear transformations*

The importance of linear maps lies in a fundamental principle:

> **General principle**
>
> "At small scales everything is linear"

In other words, when you observe closely with a microscope the graph of a function [3] $y = f(x)$ near a point with coordinates $(a, f(a))$, you will see that it looks like a line. A small change in x results in a proportional change in y. This approximation explains the widespread applicability of linear functions and highlights why

[3] The function must be *regular* enough: the technical word is *differentiable*. But most physical phenomena are represented by differentiable functions

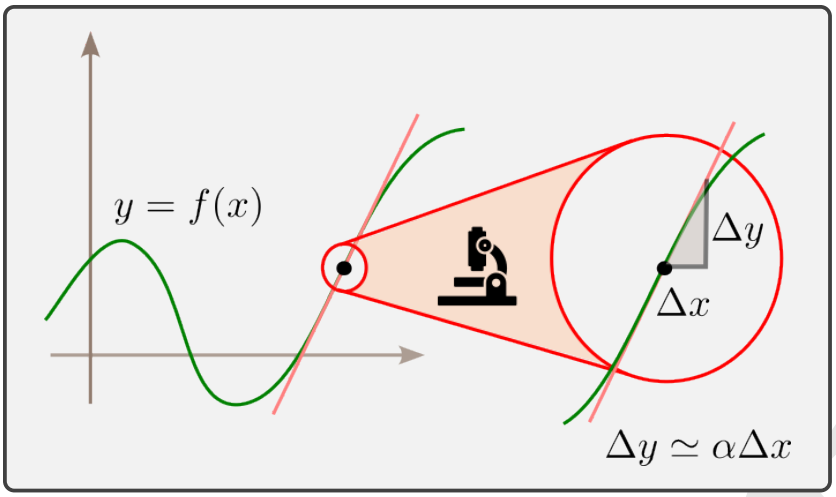


Figure 1: At the microscopic level, everything looks linear!

studying linear algebra is crucial—it serves as a *first-order approximation* to calculus [4].

To define linear maps, we first need to understand vector spaces, where addition and scalar multiplication are defined. The first part of this book will introduce these vector spaces and their properties.

Who should read this book This book is designed for data scientists, students, and professionals seeking to deepen their understanding of linear algebra. Whether you are a beginner or looking to refresh your knowledge, this book will provide the essentials you need to apply linear algebra effectively in your work.

[4] Calculus is the general study of functions

Part I

Vector spaces

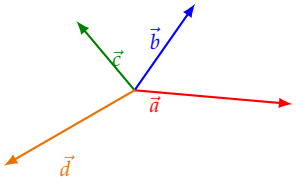
Vectors

Early release chapter

- ✓ Traditional vectors
- ✓ Abstract vectors
- ✓ Examples of vector spaces
- ✓ Vector spaces in pure Python
- ✓ Vectors with Numpy
- ✓ Subspaces
- ✗ Linear independence
- ✗ Systems of linear equations

Traditional vectors

When we first encounter vectors they are introduced as little arrows anchored at the origin in the plane or in space. In physics they are used to represent forces acting on points: the arrow points towards a particular direction (as if the force was pulling into that direction) and the size of the arrow indicates a magnitude (the "strength" of the pull).



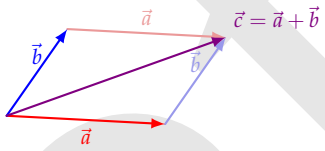
Notation Usually we write vectors with a little arrow above them, like this : $\vec{v}$ (read as "the vector v"), or sometimes simply in boldface **v**.

Addition of vectors Given a pair of vectors, one can create a third one: the sum of them.

$$\vec{c} = \vec{a} + \vec{b}$$

From high school, you might remember that the sum $\vec{c}$ of these two vectors is obtained by drawing a parallelogram based on $\vec{a}$ and $\vec{b}$ and defining the sum as the diagonal leaving the origin. The reason for that choice comes from physics: if $\vec{a}$ and $\vec{b}$ are two forces applied to the origin, they act together as a single resultant force which is exactly $\vec{c}$.

Alternatively, you might pick $\vec{b}$, drag it on the plane (while keeping it pointing into the same direction at all times) until its anchor coincides with the tip of $\vec{a}$. The sum is then the vector from the origin to the tip of the dragged vector.



Multiplication by a scalar [5] A vector can be *rescaled* by multiplying it by a number. Given a vector $\vec{v}$ and a scalar α, we can create a new vector written $\alpha\vec{v}$ as follows:

- **Case 1:** if $\alpha >= 0$ then $\alpha\vec{v}$ is a vector pointing in the same direction as $\vec{v}$ but whose length is α times the length of v.
- **Case 2:** if $\alpha < 0$ then $\alpha\vec{v}$ is a vector pointing in the opposite direction of $\vec{v}$ but whose length is $|\alpha|$ times the length of v. The notation $|\alpha|$ means "the absolute value [6] of α"

[5] a *scalar* is simply a number, like 3.1 if you want. Mathematicians like to use greek letters for them, like α (read "alpha") for example

[6] The absolute value of a number is just the number without its sign: $|-3.14| = 3.14$ and $|3.14| = 3.14$

Quite often in this book, I will illustrate new notions with code blocks. You will recognize them easily, as they will all look like this:

Absolute value The absolute value function is given in python by:

```
# Function to calculate the absolute value
def absolute_value(num):
    # Return the absolute value of the input
    if num >=0:
        return num
    else:
        return -num
# Example usage
result = absolute_value(-5)
print("The absolute value is:", result)
```

Properties of the addition It turns out that this simple *parallelogram rule* of addition has some nice properties. We list them because they will appear again and again in other situations.

1. **Zero vector**: the zero vector $\vec{0}$ is an arrow of zero length (leaving the origin and pointing to the origin itself). Adding it to another vector $\vec{v}$ leaves $\vec{v}$ unchanged [7]:

[7] Do you see why? try to add to $\vec{a}$ a very tiny vector $\vec{b}$ and see what happens to the parallelogram

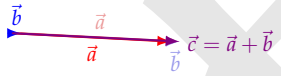
$$\vec{0} + \vec{v} = \vec{v} + \vec{0} = \vec{v}$$

2. **Commutativity** We say that the addition is *commutative* when the sum does not depend on the order of the vectors [8]:

$$\vec{v} + \vec{w} = \vec{w} + \vec{v}$$

3. **Associativity** When we add more than two vectors, we can group them as we wish to make intermediate sums:

$$\vec{u} + (\vec{v} + \vec{w}) = (\vec{u} + \vec{v}) + \vec{w}$$



[8] Why is that? observe that the parallelogram rule does not mention anything about the order of the vectors...

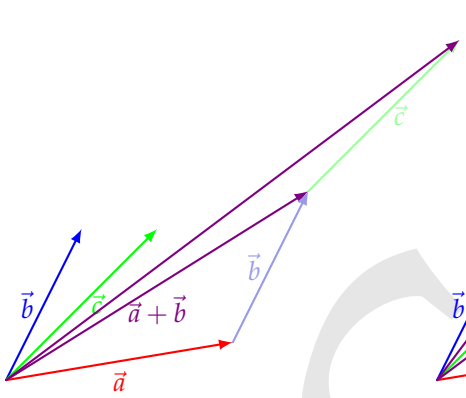


Figure 2: $(\vec{a} + \vec{b}) + \vec{c}$

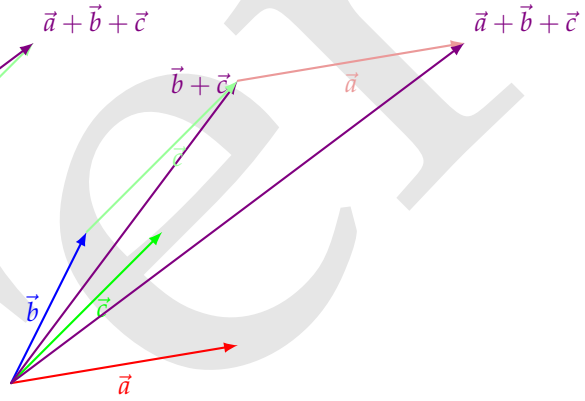


Figure 3: $\vec{a} + (\vec{b} + \vec{c})$

Why do we care about associativity? Without associativity, we could simply not make sense of the sum $\vec{a} + \vec{b} + \vec{c}$. Indeed, how can you tell if it means $(\vec{a} + \vec{b}) + \vec{c}$ or $(\vec{a} + \vec{b}) + \vec{c}$? [9] If we want to add say, twenty vectors, you can easily imagine the nightmare it would be...

[9] For a sum with n terms, the number C_n of possible ways to compute the sum grows very fast: $C_n \simeq \frac{4^n}{n^{3/2}\sqrt{\pi}}$...

Vectors in space One an also define vectors in 3D space (again thinking in terms of arrows anchored at the origin). They can be rescaled (that is, "multiplied by a scalar") and there is also an addition for them:

Coordinate description of vectors Vectors in the plane can be described by their coordinates: from the tip of $\vec{v}$ draw a vertical line until it hits the x axis, record the position of the intersection as a, then do the same along the horizontal. The pair of coordinates [10] (a, b) describes completely the vector $\vec{v}$.

[10] Usually called *cartesian* coordinates, from the name of their inventor René Descartes.

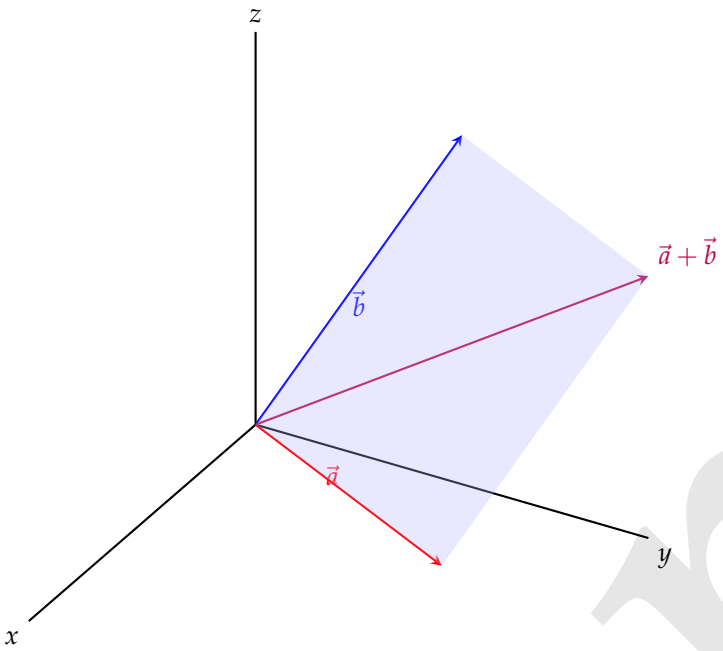


Figure 4: Vector addition in 3D space

Using coordinates, the multiplication by a scalar takes a simple form: simply multiply each coordinate by the same scalar

$$\alpha(a,b) = (\alpha a, \alpha b)$$

And the addition is also simple: add the first coordinates together then add the second coordinates together

$$(a,b) + (c,d) = (a+b, c+d)$$

Most of the time we prefer to write such vectors in column form:

$$\vec{v} = \begin{bmatrix} a \\ b \end{bmatrix}$$

Transpose of a vector For typographical reasons, we sometimes prefer to write a vector horizontally:

$$\vec{w} = \begin{bmatrix} a & b \end{bmatrix}$$

To go from one to the other, we use the *transpose* operator $\top$, that reverses the orientation:

$$\begin{bmatrix} a \\ b \end{bmatrix}^\top = \begin{bmatrix} a & b \end{bmatrix}$$

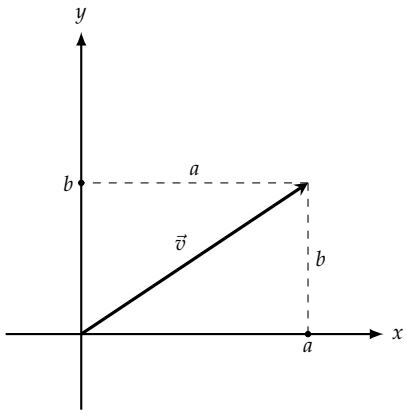


Figure 5: Illustration of a vector $\vec{v}$ with coordinates (a,b).

$$\begin{bmatrix} a & b \end{bmatrix}^\top = \begin{bmatrix} a \\ b \end{bmatrix}$$

The collection of all such column vectors with two entries is written $\mathbb{R}^2$. The symbol $\mathbb{R}$ denotes the set of all real numbers (think "float" numbers, like 3.14 or -12.75). We could write $\mathbb{R}^1$ as well, but that exponent 1 is not used. [11]

If you want column vectors with three entries, you get $\mathbb{R}^3$. You can go higher and define $\mathbb{R}^5$ if you wish.

How would you define addition in $\mathbb{R}^5$ and multiplication by a scalar? You would simply copy paste the definitions used for $\mathbb{R}^2$:

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} + \begin{bmatrix} f \\ g \\ h \\ i \\ j \end{bmatrix} = \begin{bmatrix} a+f \\ b+g \\ c+h \\ d+i \\ e+j \end{bmatrix} \quad \textbf{Addition of two vectors}$$

$$\alpha \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} \alpha a \\ \alpha b \\ \alpha c \\ \alpha d \\ \alpha e \end{bmatrix} \quad \textbf{Multiplication by a scalar}$$

The purpose of that chapter was simply to build upon earlier knowledge that you have, showing that these little arrows do behave nicely, following some patterns that will be generalized in the next chapter. This is perhaps what mathematicians do best: find patterns in the real world, extract those and generalize.

Exercise 1: Vector Addition

Problem: Given vectors $\mathbf{a} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, find the vector $\mathbf{c} = \mathbf{a} + \mathbf{b}$.

Solution:

$$\mathbf{c} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3+1 \\ 4+2 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \end{bmatrix}$$

[11] We use the notation $\mathbb{N}$ for the non-negative integers $0, 1, 2, \ldots$, the notation $\mathbb{Z}$ for the integers $\ldots, -2, -1, 0, 1, 2, 3, \ldots$. The set of fractions is denoted by $\mathbb{Q}$.

Exercise 2: Scalar Multiplication

Problem: Multiply the vector $\mathbf{a} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$ by the scalar $\alpha = 3$.

Solution:

$$\alpha\mathbf{a} = 3 \times \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 9 \\ 12 \end{bmatrix}$$

Exercise 3: Linear Combination

Problem: For vectors $\mathbf{a} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$, and scalars $\alpha = 2$ and $\beta = 3$, compute the linear combination $\mathbf{c} = \alpha\mathbf{a} + \beta\mathbf{b}$.

Solution:

$$\mathbf{c} = 2 \times \begin{bmatrix} 1 \\ 2 \end{bmatrix} + 3 \times \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2+6 \\ 4+9 \end{bmatrix} = \begin{bmatrix} 8 \\ 13 \end{bmatrix}$$

Vector spaces

Abstract vectors

Mathematicians like to abstract things whenever they get the chance. So they defined *a vector space* as a collection of objects that can be added together, and multiplied by a scalar. We have already seen some examples of *vector spaces*[12]: $\mathbb{R}$ (we can add two numbers, and certainly multiply a scalar), $\mathbb{R}^2, \mathbb{R}^3, \mathbb{R}^5, \ldots$

When you think about it, the addition $+$ is a function f (that you could call "sum") that takes a pair of vectors $(\vec{u}, \vec{v})$ and returns a new vector called their sum. In other words

$$(\vec{u}, \vec{v}) \mapsto f(\vec{u}, \vec{v}) = \vec{u} + \vec{v}$$

The notation $\vec{u} + \vec{v}$ is more common than $f(\vec{u}, \vec{v})$ in that context [13]. The set of all pairs $(\vec{u}, \vec{v})$ of elements in V is denoted by $V \times V$. Mathematicians use the notation:

$$f : V \times V \to V$$

to say that f is a function from the set of pairs of elements of V to V. Using this notation, the scalar multiplication is a function

$$g : \mathbb{R} \times V \to V$$

since it takes a pair (number, vector) and returns a vector.

[12] Some of them can be visualized, but $\mathbb{R}^5$ cannot for example

[13] Some programming languages, like Haskell, allow a *prefix* notation for the addition: $a + b$ can also be written as $(+)ab$

Definition of a vector space

A set V is called a vector space over $\mathbb{R}$ if it has a binary operation $V \times V \rightarrow V$, usually denoted as addition $(\vec{u}, \vec{v}) \rightarrow \vec{u} + \vec{v}$, and another binary operation $\mathbb{R} \times V \rightarrow V$, usually denoted as multiplication $(\alpha, \vec{u}) \rightarrow \alpha\vec{u}$, satisfying the following properties:
Addition properties

1. **Commutativity:** For all $\vec{v}, \vec{w} \in V$,

$$\vec{v} + \vec{w} = \vec{w} + \vec{v}$$

2. **Associativity:** For all $\vec{u}, \vec{v}, \vec{w} \in V$,

$$(\vec{u} + \vec{v}) + \vec{w} = \vec{u} + (\vec{v} + \vec{w})$$

3. **Zero vector:** There exists an element $\vec{0} \in V$ such that

$$\vec{v} + \vec{0} = \vec{v}$$

for all $\vec{v} \in V$.

4. **Additive Inverse:** For every $\vec{v} \in V$, there exists $\vec{w} \in V$ such that

$$\vec{v} + \vec{w} = \vec{0}$$

Scalar Multiplication properties

1. **Distributivity of scalar multiplication with respect to vector addition :** For all $\alpha \in \mathbb{R}$ and $\vec{v}, \vec{w} \in V$,

$$\alpha(\vec{v} + \vec{w}) = \alpha\vec{v} + \alpha\vec{w}$$

2. **Distributivity of scalar multiplication with respect to field addition:** For all $\alpha, \beta \in \mathbb{R}$ and $\vec{v} \in V$,

$$(\alpha + \beta)\vec{v} = \alpha\vec{v} + \beta\vec{v}$$

3. **Associativity of scalar multiplication:** For all $\alpha, \beta \in \mathbb{R}$ and $\vec{v} \in V$,

$$\alpha(\beta\vec{v}) = (\alpha\beta)\vec{v}$$

4. **Identity element of scalar multiplication :** one has

$$1\vec{v} = \vec{v}$$

for all $\vec{v} \in V$.

It's quite a long list, but I assure you that all these properties are very natural, once you remember that they originate from the properties of the good old vectors that you already know. Here are some examples:

- **Distributivity of scalar multiplication with respect to vector addition** : think about the geometry, if you rescale the two vectors, the parallelogram built on them is automatically rescaled as well!

- **Associativity of scalar multiplication:** If I amplify a vector by a factor of 2 and then by a factor of 3, the net result is an amplification by a factor of 6.

> **About the scalars**
>
> In this book, most of the vector spaces we will encounter will be *over* $\mathbb{R}$, meaning that the scalars are real numbers (think *floats*). But we will also encounter vector spaces over $\mathbb{C}$, the *complex numbers*. Of course, mathematicians love to generalize, so they defined vector spaces over a general *field*. What is a field then? Well, it's just a set that has an addition, a multiplication, where every non-zero element has a multiplicative inverse.

Examples of vector spaces

The zero space

I will use this one as a warm up: we consider the set that contains only one element $\{\vec{0}\}$ with the properties that $\vec{0} + \vec{0} = \vec{0}$ and that for every scalar α one has $\alpha\vec{0} = \vec{0}$.

An exotic space

I want to give an example [14]showing how abstract the notion of vector space can be, and how far the *addition* can be from standard addition. Call V the set of all positive real numbers. Define the *addition* on V to be the standard product (!) of two positive numbers. Define the scalar multiplication to be the exponentiation: in other words [15],

$$\alpha x := x^{\alpha}$$

It is a good exercise to convince yourself that this space V is indeed a vector space! On the way to do it, you will see many familiar formulas appearing, such as $x^{a+b} = x^a.x^b$ for example ... The solution of that exercise is given at the end of the chapter.

[14] There is a keyword explaining what is happening in this example: the *logarithm* function transforms this exotic addition into the usual addition.

[15] I will use the notation $A := B$ to mean "A is equal to B by definition.

Spaces of functions

Let S be an arbitrary set. Then the set of all functions f from S to $\mathbb{R}$ forms a vector space [16].

[16] Saying that a function is "from a set A to a set B actually means that the function takes an element x in A and returns an element $f(x)$ in B

A short intermede: set notation

Instead of saying with words

$$x \text{ is in } A$$

we can use the notation

$$x \in A.$$

Instead of saying: *the set of all x in A that are non negative*, we can write:

$$\{x \in A \mid x \geq 0\}$$

This idiom is the inspiration for the *list comprehension* notion in Python:

$$[x \text{ for } x \text{ in } A \text{ if } x >= 0]$$

- **Vector Addition**

  For any two functions f, g from S to $\mathbb{R}$, their sum $f + g$ is defined by

  $$(f + g)(s) = f(s) + g(s) \quad \text{for all } s \in S.$$

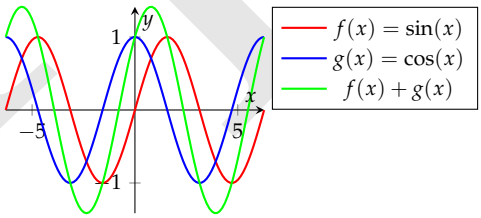
  This operation is commutative, associative, and has an identity element, which is the zero function 0 defined by $0(s) = 0$ for all $s \in S$.

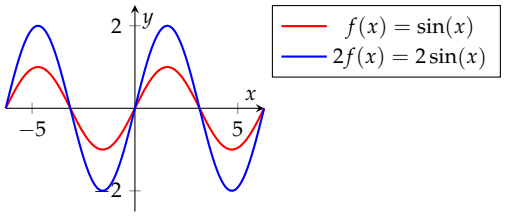
- **Scalar Multiplication**

  For any scalar $\alpha \in \mathbb{R}$ and any function f from S to $\mathbb{R}$, the scalar multiplication αf is defined by

  $$(\alpha f)(s) = \alpha \cdot f(s) \quad \text{for all } s \in S.$$

  This operation is distributive over both vector addition and addition in $\mathbb{R}$.





Polynomial functions

Polynomials of degree n or less form a vector space. Given:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0$$

$$q(x) = b_n x^n + b_{n-1} x^{n-1} + \ldots + b_0$$

Then:

$$p(x) + q(x) = (a_n + b_n)x^n + (a_{n-1} + b_{n-1})x^{n-1} + \ldots + (a_0 + b_0)$$

In other words, you add everything and then group together terms of equal degrees [17]. The scalar multiplication works as follows:

$$c \cdot p(x) = ca_n x^n + ca_{n-1} x^{n-1} + \ldots + ca_0$$

Clearly the result is also a polynomial of degree less than n or equal to n.

[17] For example:
$(3x^2+6x+1)+(x^2+2x-3) = 4x^2+8x-2$

Audio signal

An audio signal is a representation of sound, typically as an electrical voltage. In digital audio, this signal is encoded as a series of discrete numerical samples taken at a constant rate.

Nature of Sound Waves Sound is a mechanical wave of pressure and displacement through a medium such as air. The behavior of sound can be described by changes in pressure over time. When we digitize sound, we actually convert these changes in pressure into electrical signals (analog), and then into digital data that can be processed by computers.

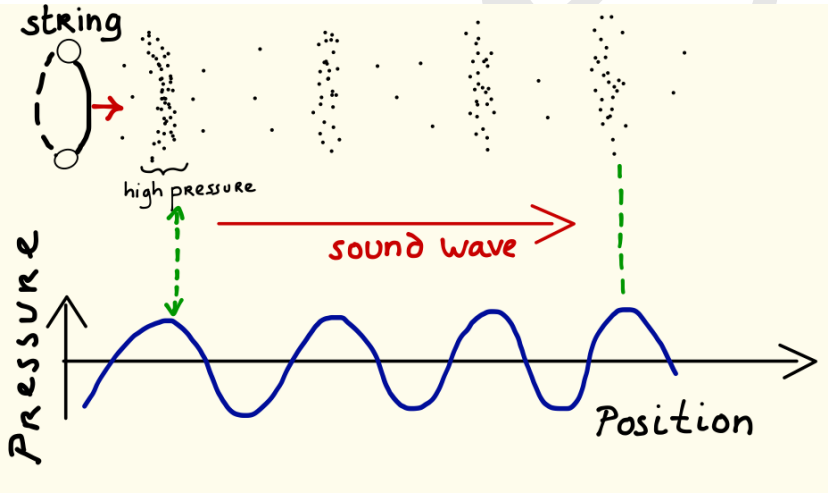
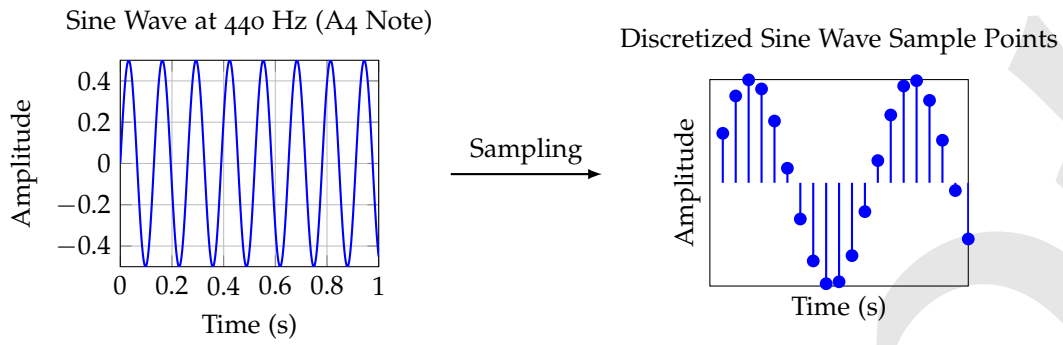


Figure 6: Sound as pressure changes

Sampling The process of converting an analog audio signal into a digital form involves sampling [18]the continuous signal at discrete intervals. The standard sample rate for CD-quality audio is 44.1 kHz, which means the audio is sampled 44,100 times per second. Each sample represents the audio wave's amplitude at that moment.

[18] Sampling means that you take a measurement at regular time intervals of the signal of interest and save the array of results.



Quantization Each sampled value is quantized into a numerical value. For CD-quality audio, this is typically done at 16-bit depth, which gives $2^{16} = 65536$ possible values for the amplitude. These values can range from -32,768 to 32,767, where zero represents no sound pressure difference.

The discretized signal is entirely described by a sequence of *samples* $(a_0, a_1, a_2, \ldots, a_N)$: they are the values of the signal at the consecutive timestamps $t_0, t_1, t_2, \ldots, t_N$. In other words, if the signal is given by the function $s(t)$ (t being the time):

$$a_0 = s(t_0), a_1 = s(t_1), \ldots, a_N = s(t_N)$$

If they are N such timestamps, the discretized signal is an element of the vector space $\mathbb{R}^N$ that we already encountered.

Superposition of Audio Signals Superposition in audio signals refers to the additive property where two or more sound waves pass through the same medium at the same time, causing the sound pressures to be added. For instance, if two audio signals (or waves) $f(t)$ and $g(t)$ are played together, their superposition results in a new audio signal $h(t)$:

$$h(t) = f(t) + g(t)$$

Here $h(t)$ is the combined sound that you hear. In the vector space of (discretized) audio signals, the addition corresponds to the superposition of sounds. The scalar multiplication in this context is called a *gain factor* (when larger than 1, otherwise it is an *attenuation*).

Not every space is a vector space

The set $\mathbb{R}^+$ of all positive numbers is closed under addition [19]. However it is not closed under scalar multiplication (multiply a vector by -1 for example will not produce a positive number).

[19] To be *closed under addition* means that by adding two elements of the particular set in consideration you get a result that also belongs to that set. Indeed, here $x > 0, y > 0$ implies that $x + y > 0$.

Dual of a vector space

Given a vector space V. The set $V^\star$ of all functions f from V to $\mathbb{R}$ such that $f(x+y) = f(x) + f(y)$ and $f(\alpha x) = \alpha f(x)$ for all x, y in V and any α in $\mathbb{R}$ is a vector space, called the *dual space* of V.

Product of vector spaces

Let V and W be vector spaces. The **product** of V and W, denoted $V \times W$, is defined as the set of all ordered pairs (v, w) where $v \in V$ and $w \in W$. The set $V \times W$ itself becomes a vector space with the following operations:

The addition in $V \times W$ is defined componentwise:

$$(v, w) + (v', w') = (v + v', w + w')$$

for all $(v, w), (v', w') \in V \times W$.

The scalar multiplication in $V \times W$ is also defined componentwise:

$$\alpha(v, w) = (\alpha v, \alpha w)$$

for all $\alpha \in \mathbb{R}$ and $(v, w) \in V \times W$.

The operations of addition and scalar multiplication in $V \times W$ satisfy all the axioms of a vector space, making $V \times W$ a vector space.

For example the space $\mathbb{R}^2$ we already encountered is simply the product $\mathbb{R} \times \mathbb{R}$ (hence the notation $\mathbb{R}^2$).

Vector spaces in pure Python

If we want to implement the notion of vector space in python, we need to go back to the root of "what is a vector space". As we saw, it's a set where an addition is defined (things like $a + b$ make sense), and a scalar multiplication exists (ie typing $3x$ makes sense).

You might think: lists in python look like vectors in $\mathbb{R}^n$, so we could try an addition like : $[1, 2, 3] + [0, 5, 6]$. Unfortunately it results in $[1, 2, 3, 0, 5, 6]$, not what we want... In python, when you write a sum $obj1 + obj2$ of two objects $obj1, obj2$, this actually means under the hood `obj1.__add__(obj2)`, meaning that a special method with double underscore `__add__` needs to be defined for the class of which $obj1$ is an instance. The same thing happens if you want to be able to write expressions like $3 * obj1$ (to represent scalar multiplication).

So in order to design a vector space, we can imagine a base class *VectorSpace* of which particular spaces (like $\mathbb{R}^n$) would be subclasses. However, we want every such subclass to necessarily implement

three methods: one addition, one scalar multiplication and one zero vector. In python, there is a mechanism to do that, using *abstract base classes* and *abstract methods* [20]

[20] By decorating a method with *@abstractmethod* we ensure that any subclass of VectorSpace must implement that method.

```python
from abc import ABC, abstractmethod

class VectorSpace(ABC):
    @abstractmethod
    def add(self, other):
        """
        Add two vectors in the vector space.
        """
        pass

    @abstractmethod
    def scalar_multiply(self, scalar):
        """
        Multiply this vector by a scalar.
        """
        pass

    @abstractmethod
    def zero_vector(self):
        """
        Return the zero vector of this vector space.
        """
        pass

    def __add__(self, other):
        return self.add(other)

    def __mul__(self, scalar):
        return self.scalar_multiply(scalar)

    def __rmul__(self, scalar):
        return self.scalar_multiply(scalar)
```

The "dunder methods" [21] `__add__` and `__mul__` allow us to use the familiar symbols $+$ and $*$ to represent the operations [22].

For concrete applications, the code I am giving here is not useful: linear algebra in Python is best done through the use of the Numpy library, that we will see later.

[21] "dunder" stands for *double underscore*.

[22] The technical name is "operator overloading", meaning that you give artificially one more meaning to the $+$ operator: now you can use it with your custom made VectorSpace class

```python
# Example implementation for R^n using lists
class Rn(VectorSpace):
    def __init__(self, elements):
        if not isinstance(elements, list):
            raise TypeError("Elements must be a list")
        self.elements = elements

    def add(self, other):
        if len(self.elements) != len(other.elements):
            raise ValueError("Both vectors must have
                              the same dimension")
        return Rn([self.elements[i] + other.elements[i]
                   for i in range(len(self.elements))])

    def scalar_multiply(self, scalar):
        try:
            multiplier = float(scalar)
        except TypeError:
            return NotImplemented
        return Rn([scalar * x for x in self.elements])

    def zero_vector(self):
        return Rn([0] * len(self.elements))

    def __repr__(self):
        return f"Rn({self.elements})"

# Example usage
v1 = Rn([1, 2, 3])
v2 = Rn([4, 5, 6])
v3 = v1 + v2
print(v3)  # Output: Rn([5, 7, 9])

v4 = 3 * v1
print(v4)  # Output: Rn([3, 6, 9])
```

Exercise 1

Problem: Call V the set of all positive real numbers. Define the *addition* on V to be the standard product of two positive numbers. Define the scalar multiplication to be the exponentiation: in other words

$$\alpha x := x^{\alpha}$$

Show that this space V is indeed a vector space.

Solution: Let us write the addition with a special symbol to avoid confusion: $x \oplus y$ means $x.y$. This addition is commutative and associative (because the product of two real numbers is). The 0 vector is the number 1:

$$x \oplus 1 = 1 \oplus x = 1.x = x$$

The distributivity of the "scalar multiplication" comes from:

$$\alpha.(x+y) = (x+y)^{\alpha} = x^{\alpha}.x^{\alpha} = (\alpha x) \oplus (\alpha y)$$

Finally we have

$$\alpha.(\beta x) = (x^{\beta})^{\alpha} = x^{\alpha.\beta} = (\alpha.\beta)x$$

Exercise 2: Dual space

Problem: Verify that the dual space $V^\star$ of a vector space V is a vector space.

Solution: Axiom 1: Closure under Addition If $f, g \in V^\star$, then $f+g$ defined by $(f+g)(v) = f(v) + g(v)$ must also be a linear functional (i.e., in $V^\star$). Since f and g are linear,

$$\begin{aligned}(f+g)(av+bw) &= f(av+bw) + g(av+bw) \\ &= af(v) + bf(w) + ag(v) + bg(w) \\ &= a(f(v)+g(v)) + b(f(w)+g(w)) \\ &= a(f+g)(v) + b(f+g)(w)\end{aligned}$$

for any $v, w \in V$ and $a, b \in \mathbb{F}$. Hence, $f+g$ is linear and $V^\star$ is closed under addition.

Axiom 2: Closure under Scalar Multiplication If $f \in V^\star$ and $\alpha \in \mathbb{R}$, then αf defined by $(\alpha f)(v) = \alpha \cdot f(v)$ must also be a linear functional. Since f is linear,

$$\begin{aligned}(\alpha f)(av+bw) &= \alpha \cdot f(av+bw) \\ &= \alpha \cdot (af(v) + bf(w)) \\ &= a(\alpha f(v)) + b(\alpha f(w)) \\ &= a(\alpha f)(v) + b(\alpha f)(w)\end{aligned}$$

for any $v, w \in V$ and $a, b \in \mathbb{R}$. Thus, αf is linear and $V^\star$ is closed under scalar multiplication.

Axiom 3: Additive Identity The zero functional 0 defined by $0(v) = 0$ for all $v \in V$ is in $V^\star$ since it is linear:

$$0(av+bw) = 0 = 0 + 0 = 0(v) + 0(w)$$

Thus, there exists an additive identity in $V^\star$.

Axiom 4: Additive Inverse For each $f \in V^\star$, the functional $-f$ defined by $(-f)(v) = -f(v)$ is in $V^\star$ and serves as the additive inverse of f because:

$$(f + (-f))(v) = f(v) + (-f)(v) = f(v) - f(v) = 0$$

Associativity, Commutativity of Addition, Distributivity Since addition and scalar multiplication in $V^\star$ are defined pointwise based on the corresponding operations in $\mathbb{R}$, they inherit the properties of associativity, commutativity of addition, and distributivity of scalar multiplication over vector addition and field addition from $\mathbb{R}$.

Vectors with Numpy

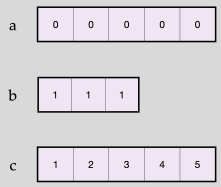
About this chapter

NumPy, which stands for *Numerical Python* stands at the root of an entire ecosystem of scientific libraries. Every single machine learning, data science or scientific computing library uses it in a crucial way. Its main object, the *NumPy array* provides a highly efficient replacement for Python lists and arrays that can be used whenever large datasets must be dealt with in a highly efficient manner. This chapter relates everything we have seen so far to the NumPy mindset.

Creating NumPy arrays

Make one-dimensional arrays The *numpy* library must be imported. Numpy has ready-made functions to create arrays filled with zeros, with ones, or populated from a standard python list.

```
import numpy as np

# array of zeroes
a = np.zeros(5)
print(a)
# [Out]: [0. 0. 0. 0. 0.]
b = np.ones(3)
print(b)
# [Out]: [1. 1. 1.]
c = np.array([1,2,3,4,5])
print(c)
# [Out]: [1 2 3 4 5]
```



Range array creation

```
# Create an array of 5 elements, evenly spaced between 10 and 50
linspace_array = np.linspace(10, 50, 5)
print("Linspace Array:", linspace_array)
# [Out]: Linspace Array: [10. 20. 30. 40. 50.]

# Create an array of integers from 1 to 10
range_array = np.arange(1, 11)
print("Range Array:", range_array)
# [Out]: Range Array: [ 1  2  3  4  5  6  7  8  9 10]
```

Basic array operations Addition and scalar multiplication are performed element-wise, and so is the application of a function to an array. If you want to apply a function to each element of a list in Python, you would typically use *map* or a *list comprehension*.

```
# Pure Python: doubling a list
my_list = [1,2,3,4,5]

def double(x):
    return x*2

# using map
map_list = list(map(double, my_list))
print(map_list)
#[Out]: [2 4 6 8 10]

# using list comprehension
list_comp = [double(x) for x in my_list]
print(list_comp)
#[Out]: [2 4 6 8 10]
```

Basic array operations

```
# Addition of two arrays
arr1 = np.array([1, 2, 3, 4, 5])

result_add = arr1 + np.arange(5, 10)
print("Addition:", result_add)
# [Out]: Addition: [ 6  8 10 12 14]

# Multiplication of elements by a scalar
result_mul = arr1 * 2
print("Multiplication by 2:", result_mul)
# [Out]: Multiplication by 2: [ 2  4  6  8 10]

# Sine function of each element
sine_values = np.sin(arr1)
print("Sine values:", sine_values)
#[Out]: Sine values: [ 0.84147  0.90929  0.14112 -0.7568  -0.95892]
```

Array slicing and indexing The slicing and indexing syntax is very close to the familiar syntax encountered for python lists.

Array slicing and indexing

```
# Access elements from index 1 to 3
arr1 = np.array([1, 2, 3, 4, 5])

sub_array = arr1[1:4]
print("Sub-array:", sub_array)
# [Out]: Sub-array: [2 3 4]

# Access elements at even index positions
even_index_elements = arr1[::2]
print("Elements at even indices:", even_index_elements)
# [Out]: Elements at even indices: [1 3 5]

# Modify elements using indices
arr1[0:3] = 9
print("Modified Array:", arr1)
# [Out]: Modified Array: [9 9 9 4 5]
```

Part II

Linear maps and matrices

Linearity

Linear maps

In the previous chapter we defined vector spaces. Now it is time to focus on the functions between such spaces, that "preserve the vector space structure": the so-called *linear maps*. What does this mean? [23]

Definition of a linear map

Let V and W be two vector spaces, and f a function from V to W. The function f is said to be *linear* if for all $\vec{v}, \vec{w}$ in V and for all α in $\mathbb{R}$ we have

$$\begin{aligned} f(\vec{v}+\vec{w}) &= f(\vec{v})+f(\vec{w}), \\ f(\alpha\vec{v}) &= \alpha f(\vec{v}). \end{aligned}$$

Non-examples Let's begin with examples of function that are **not** linear, just to convince ourselves that not all functions are linear. A function that is **not** linear is the quadratic function $f : \mathbb{R} \to \mathbb{R}$ given by $f(x) = x^2$. Remember the quadratic formula[24]:

$$f(x+y) = (x+y)^2 = x^2 + 2xy + y^2.$$

Examples

1. The functions $f : \mathbb{R} \to \mathbb{R}$ that are linear are exactly the ones of type $f(x) = \alpha x$ for some α in $\mathbb{R}$.
2. The functions $g : \mathbb{R}^2 \to \mathbb{R}$ of the form $f(x,y) = \alpha x + \beta y$ are linear[25].

[23] We can summarize "f is a function from V to W with the following notation:

$$f : V \to W$$

[24] If f were linear we would have the (false) identity:

$$f(x+y) = f(x) + f(y) = x^2 + y^2$$

[25] It would have been more correct to write

$$f\left(\begin{bmatrix} x \\ y \end{bmatrix}\right)$$

instead of $f(x,y)$ but that kind of abuse is common

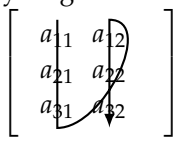
```python
import numpy as np

# Define vectors
v1 = np.array([2, 3])
v2 = np.array([1, 0])

# Define a matrix representing a linear transformation
# Example: A scaling transformation matrix
A = np.array([[2, 3]])

# Function to apply the linear transformation
def linear_transformation(A, v):
    return A.dot(v)

# Apply the transformation
transformed_v1 = linear_transformation(A, v1)
transformed_v2 = linear_transformation(A, v2)

# Check linearity
# Vector addition
vector_addition = v1 + v2
transformed_addition = linear_transformation(A,
                            vector_addition)
assert np.allclose(transformed_addition,
        transformed_v1 + transformed_v2)

# Scalar multiplication
scalar = 5
new_scalar_multiplication = linear_transformation(A,
                                        scalar * v1)
assert np.allclose(new_scalar_multiplication,
                   scalar * transformed_v1)
```

Matrices

Definition of a matrix

An $(n \times m)$ matrix is a rectangular array of numbers, with n rows and m columns. The set of all $n \times m$ matrices is denoted by $Mat(n,m)$.

Column vectors vs matrices A column vector of size n can be seen as a $n \times 1$ matrix. But if needed, one can also "unroll" an entire matrix into a column vector in $\mathbb{R}^{n \times m}$, by using a "snake" curve going down every single column of the matrix



yielding the vector [26] $\begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{12} & a_{22} & a_{32} \end{bmatrix}^T$

[26] Here I use the transpose, for typographical reasons, to avoid to write inline a very tall vector.

Addition of matrices Consider two 3×2 matrices A and B:

$$A = \begin{bmatrix} 1 & 3 \\ 4 & 5 \\ 7 & 9 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 4 \\ 0 & 1 \\ 3 & 3 \end{bmatrix}$$

The addition of A and B is given by:

$$A + B = \begin{bmatrix} 1+2 & 3+4 \\ 4+0 & 5+1 \\ 7+3 & 9+3 \end{bmatrix} = \begin{bmatrix} 3 & 7 \\ 4 & 6 \\ 10 & 12 \end{bmatrix}$$

Multiplication by a Scalar

Let $\lambda = 2$. The product of λ with matrix A is:

$$\lambda A = 2 \begin{bmatrix} 1 & 3 \\ 4 & 5 \\ 7 & 9 \end{bmatrix} = \begin{bmatrix} 2 \times 1 & 2 \times 3 \\ 2 \times 4 & 2 \times 5 \\ 2 \times 7 & 2 \times 9 \end{bmatrix} = \begin{bmatrix} 2 & 6 \\ 8 & 10 \\ 14 & 18 \end{bmatrix}$$

With these two operations, $Mat(n,m)$ becomes a vector space. But there is more to it: matrices of adequate shapes can be multiplied together.

Product of matrices

Let us see the rule on an example of a product of two matrices A in $Mat(2,3)$ and B in $Mat(3,2)$. To compute the entry at position (i,j) in the resulting matrix AB, you need to pick the i-th row of A ($\begin{bmatrix} a_{i1} & a_{i2} & a_{i3} \end{bmatrix}$), the j-th column of B (namely $\begin{bmatrix} b_{1j} \\ b_{2j} \\ b_{3j} \end{bmatrix}$), compute [27] the dot product of these two vectors and place the result in position (i,j). On some example it looks as follows:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix}$$

The product of A and B is then:

$$AB = \begin{bmatrix} (1\cdot 7 + 2\cdot 9 + 3\cdot 11) & (1\cdot 8 + 2\cdot 10 + 3\cdot 12) \\ (4\cdot 7 + 5\cdot 9 + 6\cdot 11) & (4\cdot 8 + 5\cdot 10 + 6\cdot 12) \end{bmatrix}$$

In other words:

$$AB = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

It is important to note several things:

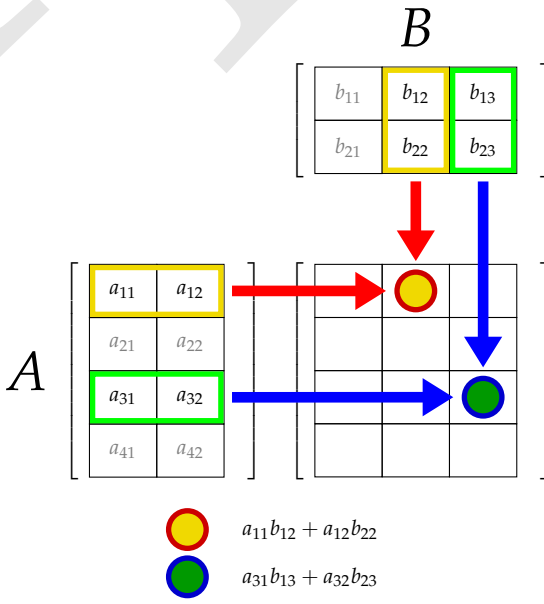
- the shapes of the matrices are: $(n \times m).(m) \times p) \rightarrow (n \times p)$. In other words the number of columns of the first matrix must be equal to the number of rows of the second one.

- the product is not commutative: if A is in $Mat(2,3)$, B in $Mat(3,4)$ then the product AB exists, but BA does not [28].

The general pattern to compute matrix products looks like that:

[27] The dot product will be seen later, but the formula is

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} . \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = a_1b_1 + a_2b_2 + a_3b_3$$



[28] Even when the shapes do match, the product is not commutative:

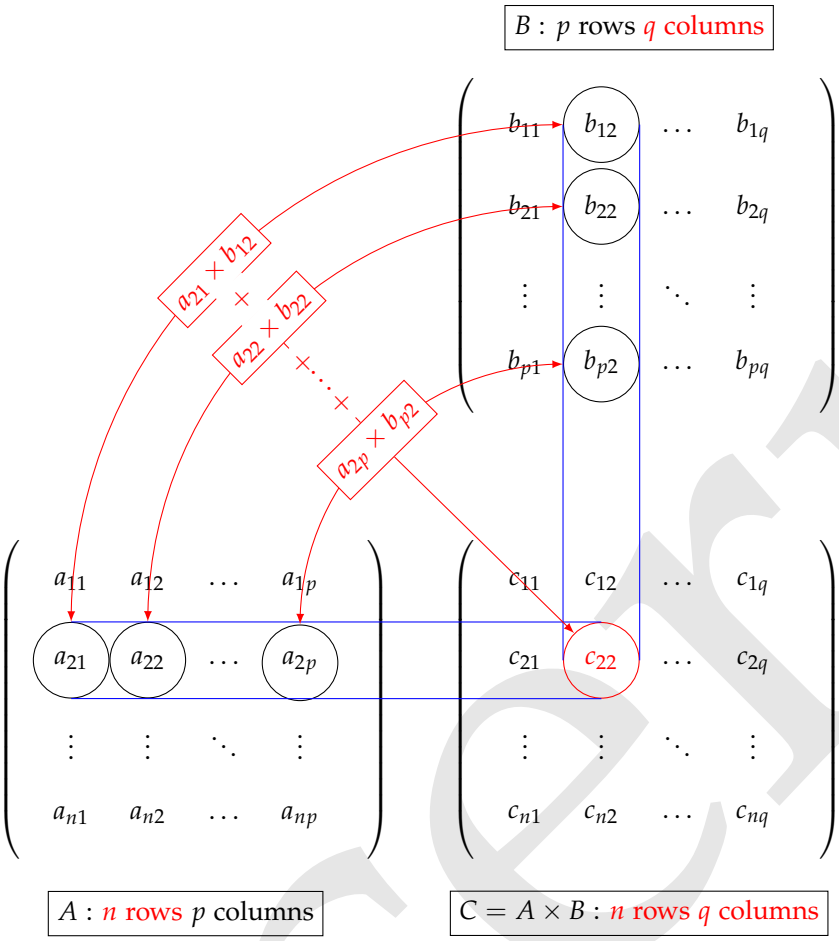
$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$$

Then the product AB is

$$AB = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 8 & 5 \\ 20 & 13 \end{bmatrix}$$

While

$$BA = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 13 & 20 \\ 5 & 8 \end{bmatrix}$$


B : p rows q columns
b_{11} b_{12} … b_{1q}
b_{21} b_{22} … b_{2q}
b_{p1} b_{p2} … b_{pq}
$a_{21} \times b_{12}$
+
$a_{22} \times b_{22}$
+ … +
$a_{2p} \times b_{p2}$
a_{11} a_{12} … a_{1p}
a_{21} a_{22} … a_{2p}
a_{n1} a_{n2} … a_{np}
c_{11} c_{12} … c_{1q}
c_{21} c_{22} … c_{2q}
c_{n1} c_{n2} … c_{nq}
A : n rows p columns
$C = A \times B$: n rows q columns

Part III

Matrix decompositions

SVD

The effect of a linear map

What does a linear map actually do? Let's see some examples of what it can and cannot do, by focussing on the case of the plane $\mathbb{R}^2$.

Large scale effect

A linear map acts the same everywhere, and at all possible scales. This consequence of the definition of the linearity can be observed by drawing a regular grid and observing its image by a linear map.

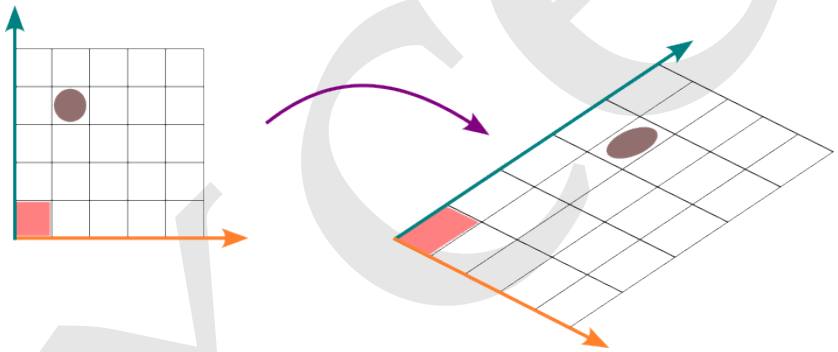


Figure 7: Large scale effect

What a linear map L does on a little square S anchored at the origin, it will do it on the same square translated by a large vector $\vec{V}$. Indeed, the image of the little square is $L(S)$. And the image of the translated [29]square is $L(S + \vec{V})$. But since L is linear, this is equal to $L(S) + L(\vec{V})$, in other words the image of the little square translated by $L(\vec{V})$.

In particular a linear map will never perform a *local deformation*, where only a limited region of the space is affected by the map, while the rest remains unchanged.

[29] The notation $S + \vec{V}$ means actually : the set of all $\vec{x} + \vec{V}$ for $\vec{x}$ in S.

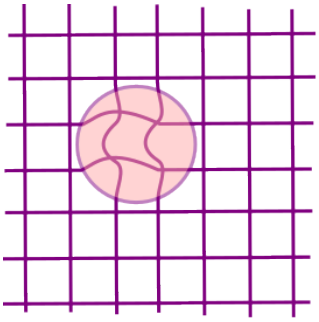


Figure 8: Linear maps cannot deform only locally the space

Convexity

Linearity imposes some strong constraints actually. For example it preserves *convexity*.

A map f *preserves the convexity* if the image[30] $f(C)$ of every convex set is also a convex set.

[30] Remember that the *image* of a set C by a function f is the set of all points $f(x)$ for all x in C. Using the set notation:

$$f(C) = \{f(x) | x \in C\}$$

Definition of a Convex Set

A subset C of a vector space V is called **convex** if for every pair of vectors $\vec{x}, \vec{y} \in C$ and every scalar t such that $0 \leq t \leq 1$, the following condition holds:

$$t\vec{x} + (1-t)\vec{y} \in C.$$

This means that for every two points in the set, the line segment connecting these points is also entirely contained within the set.

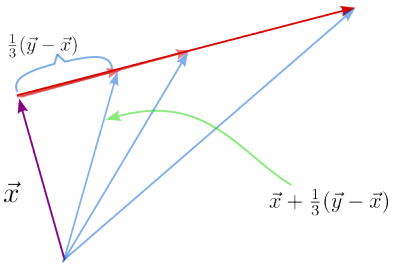


Figure 9: Segment joining $\vec{x}$ to $\vec{y}$

Stretching

Linear maps in $\mathbb{R}^2$ can stretch (or contract) horizontally :

$$\begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2x \\ y \end{bmatrix}$$

They can stretch (or contract) vertically:

$$\begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 1 & 0 \\ 0 & 1/3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y/3 \end{bmatrix}$$

They can stretch in one direction, and contract in another one at

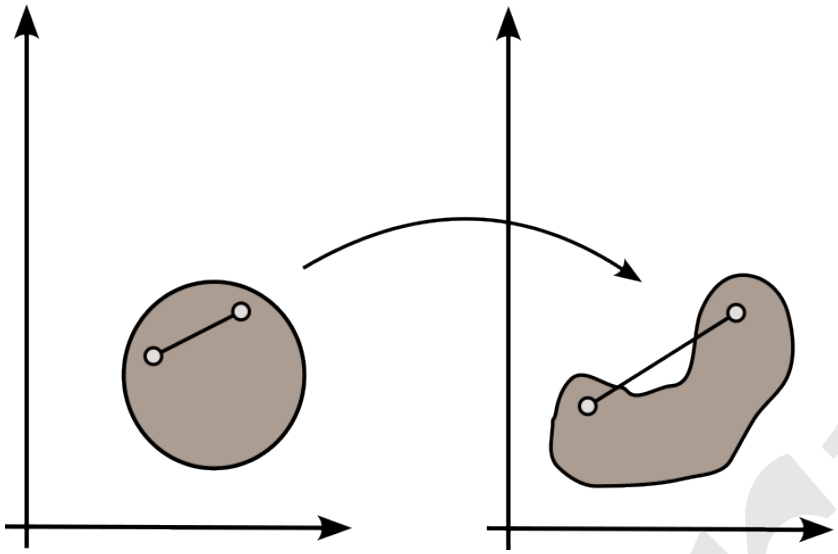


Figure 10: This map does not preserve convexity: the segment is not fully contained in $f(C)$

the same time:

$$\begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 4 & 0 \\ 0 & 1/3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4x \\ y/3 \end{bmatrix}$$

rotating

The case of dimension 2

The steps are as follows:

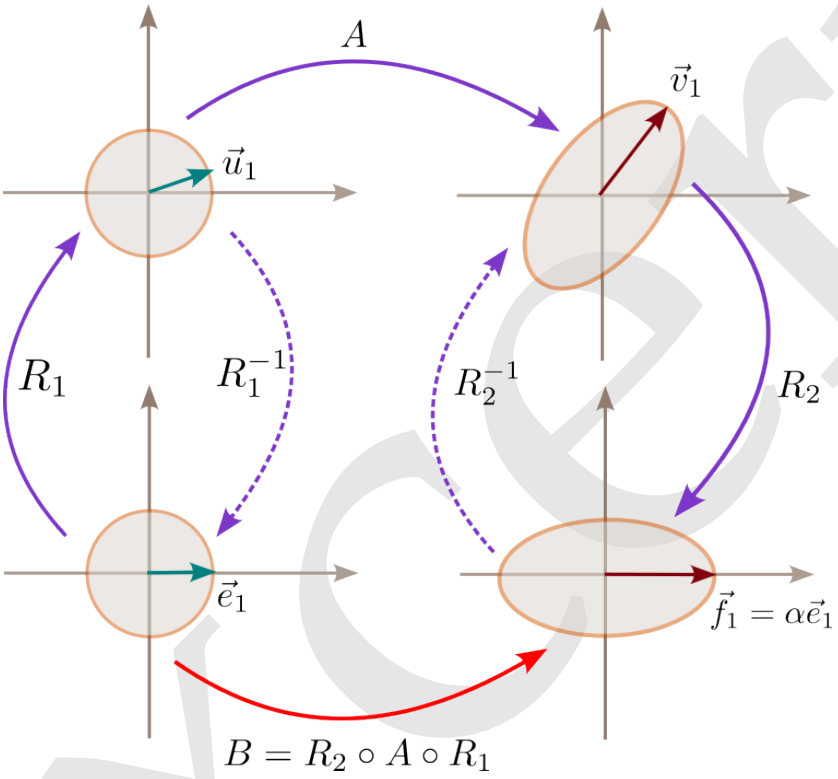
1. Find $\vec{u}_1$ such that $\vec{v}_1 := A(\vec{u}_1)$ has maximal norm among all vectors $A(\vec{u})$ where $\vec{u}$ is on the unit circle.
2. Find a rotation R_1 sending $\vec{e}_1 := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ to $\vec{u}_1$.
3. Find a rotation R_2 sending $\vec{v}_1$ to a multiple of $\vec{e}_1$.

Now, instead of studying the matrix A we will focus on $B := R_2 \circ A \circ R_1$. It is a 2 by 2 matrix whose first column can be written as

$$\begin{bmatrix} \alpha \\ 0 \end{bmatrix} \text{ because } B.\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

A priori the matrix B should look like

$$B = \begin{bmatrix} \alpha & b \\ 0 & c \end{bmatrix}$$


A
$\vec{u}_1$
$\vec{v}_1$
R_1
R_1^{-1}
R_2^{-1}
R_2
$\vec{e}_1$
$\vec{f}_1 = \alpha\vec{e}_1$
$B = R_2 \circ A \circ R_1$