Differential Geometry

Lecture Notes

Dr. Silvio Fanzon

14 Oct 2023

Academic Year 2023/24

Department of Mathematics

University of Hull

Table of contents

Welcome

These are the Lecture Notes of **Differential Geometry 661955** for T1 2023/24 at the University of Hull. We will study curves and surfaces in $\mathbb{R}^3$. I will follow these lecture notes during the course. If you have any question or find any typo, please email me at

S.Fanzon@hull.ac.uk

Up to date information about the course, Tutorials and Homework will be published on the University of Hull **Canvas Website**

canvas.hull.ac.uk/courses/67594

and on the **Course Webpage** hosted on my website

silviofanzon.com/blog/2023/Differential-Geometry

Digital Notes

Digital version of these notes available at

silviofanzon.com/2023-Differential-Geometry-Notes

Readings

The main textbook of the course is Pressley [5]. Other interesting readings are the books by do Carmo [2] and Bär [1]. I will assume some knowledge from Analysis and Linear Algebra. A good place to revise these topics are the books by Zorich [6, 7].

Visualization

It is important to visualize the geometrical objects and concepts we are going to talk about in this course. I will show basic Python code to plot curves and surfaces. This part of the course is **`not required`** for the final examination. If you want to have fun plotting with Pyhton, I recommend installation through Anaconda or Miniconda. The actual coding can then be done through Jupyter Notebook. Good references for scientific Python programming are [3, 4].
If you do not want to mess around with Python, you can still visualize pretty much everything we will do in this course using the excellent online 3D grapher tool CalcPlot3D. To understand how it works, please refer to the help manual or to the short video introduction.

! You are not expected to purchase any of the above books. These lecture notes will cover 100% of the topics you are expected to known in order to excel in the final exam.

1 Curves

We all have in mind examples of curves. These are, intuitively speaking, 1D objects in the 2D or 3D space. For example in two dimensions one could think of a straight line, a hyperbole or a circle. These can be all described by an equation in the x and y coordinates: respectively

$$y = 2x + 1\,, \quad y = e^x\,, \quad x^2 + y^2 = 1\,.$$

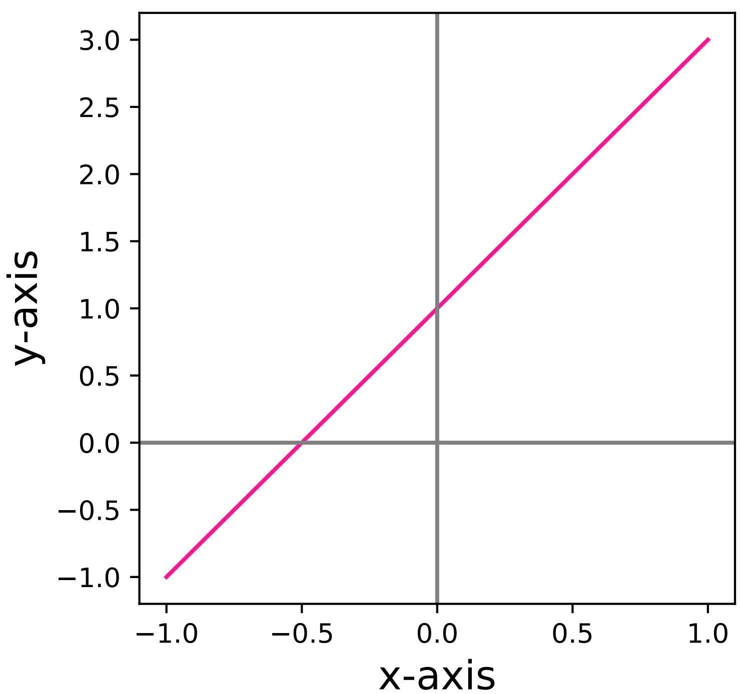


Figure 1.1: Plotting straight line $y = 2x + 1$

Goal

The aim of this course is to study curves by differentiating them.

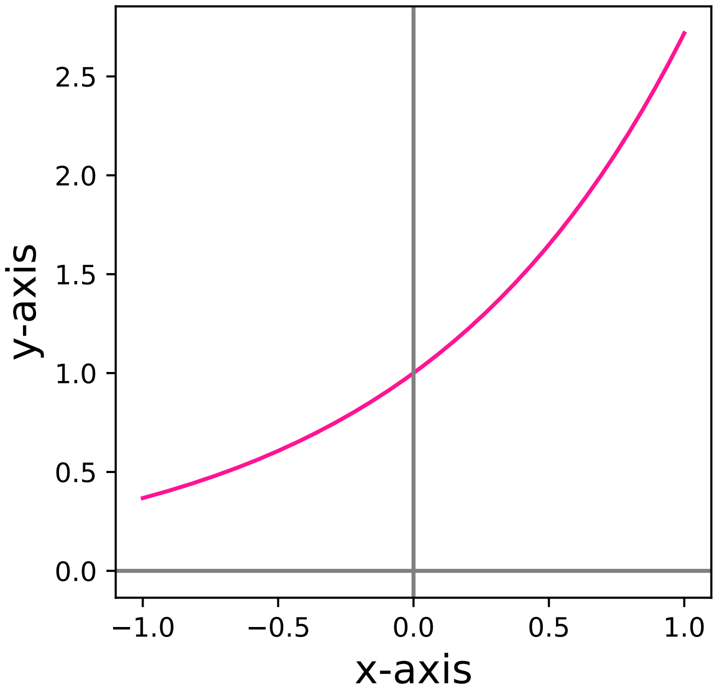


Figure 1.2: Plot of hyperbole $y = e^x$

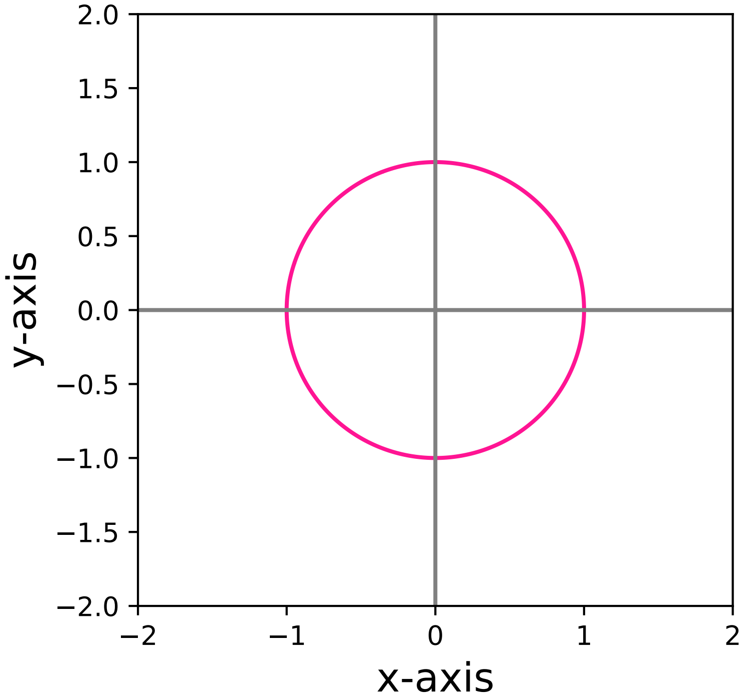


Figure 1.3: Plot of unit circle of equation $x^2 + y^2 = 1$

Question

In what sense do we differentiate the above curves?

It is clear that we need a way to mathematically describe the curves. One way of doing it is by means of Cartesian equations. This means that the curve is described as the set of points $(x, y) \in \mathbb{R}^2$ where the equation

$$f(x, y) = c\,,$$

is satisfied, where

$$f : \mathbb{R}^2 \to \mathbb{R}\,.$$

is some given function, and

$$c \in \mathbb{R}$$

some given value. In other words, the curve is identified with the subset of $\mathbb{R}^2$ given by

$$C = \{(x, y) \in \mathbb{R}^2 : \; f(x, y) = c\}\,.$$

For example, in the case of the straight line, we would have

$$f(x, y) = y - 2x\,, \quad c = 1\,.$$

while for the circle

$$f(x, y) = x^2 + y^2 \;, c = 1\,.$$

But what about for example a helix in 3 dimensions? It would be more difficult to find an equation of the form

$$f(x, y, z) = 0$$

to describe such object.

Problem

We need a unified way to describe curves.

1.1 Parametrized curves

Rather than Cartesian equations, a more useful way of thinking about curves is viewing them as the *path traced out by a moving point*. If $\gamma(t)$ represents the position a point in $\mathbb{R}^n$ at time t, the whole curve can be identified by the function

$$\gamma : \mathbb{R} \to \mathbb{R}^n\,, \;\; \gamma = \gamma(t)\,.$$

This motivates the following definition of **parametrized curve**, which will be our **main** definition of curve.

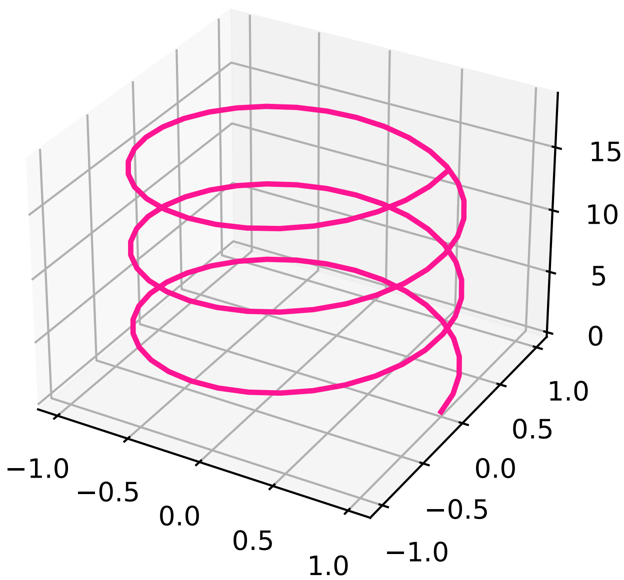


Figure 1.4: Plot of a 3D Helix

Definition 1.1: Parametrized curve

A **parametrized curve** in $\mathbb{R}^n$ is a function

$$\gamma : (a,b) \to \mathbb{R}^n .$$

where

$$-\infty \leq a < b \leq \infty .$$

A few remarks:

- The symbol (a,b) denotes an **open** interval

$$(a,b) = \{t \in \mathbb{R} : \ a < t < b\} .$$

- The requirement that

$$-\infty \leq a < b \leq \infty$$

  means that the interval (a,b) is possibly unbounded.
- For each $t \in (a,b)$ the quantity $\gamma(t)$ is a vector in $\mathbb{R}^n$.
- The **components** of $\gamma(t)$ are denoted by

$$\gamma(t) = (\gamma_1(t), \dots, \gamma_n(t)) ,$$

  where the components are functions

$$\gamma_i : (a,b) \to \mathbb{R} ,$$

  for all $i = 1, \dots, n$.

1.2 Parametrizing cartesian curves

At the start we said that examples of curves in $\mathbb{R}^2$ were the straight line, the hyperbole and the circle, with equations

$$y = 2x + 1\,, \quad y = e^x\,, \quad x^2 + y^2 = 1\,.$$

We saw that these can be represented by Cartesian equations

$$f(x, y) = c$$

for some function $f : \mathbb{R}^2 \to \mathbb{R}$ and value $c \in \mathbb{R}$. Curves that can be represented in this way are called **level curves**. Let us give a precise definition.

Definition 1.2: Level curve

A **level curve** in $\mathbb{R}^n$ is a set $C \subset \mathbb{R}^n$ which can be described as

$$C = \{(x_1, \ldots, x_n) \in \mathbb{R}^n \, : \, f(x_1, \ldots, x_n) = c\}$$

for some given function

$$f : \mathbb{R}^n \to \mathbb{R}$$

and value

$$c \in \mathbb{R}\,.$$

We now want to represent level curves by means of parametrizations.

Definition 1.3

Suppose given a level curve $C \subset \mathbb{R}^n$. We say that a curve

$$\gamma : (a, b) \to \mathbb{R}^n$$

parametrizes C if

$$C = \{(\gamma_1(t), \ldots, \gamma_n(t)) \, : \, t \in (a, b)\}\,.$$

Question

Can we **represent** the level curves we saw above by means of a parametrization γ?

The answer is YES, as shown in the following examples.

Example 1.4: Parametrizing the straight line

The straight line

$$y = 2x + 1$$

is a **level curve** with

$$C = \{(x, y) \in \mathbb{R}^2 \,:\, f(x, y) = c\} \,,$$

where

$$f(x, y) := y - 2x \,, \quad c := 1 \,.$$

How do we represent C as a **parametrized curve** γ? We know that the curve is 2D, therefore we need to find a function

$$\gamma \,:\, (a, b) \to \mathbb{R}^2$$

with componenets

$$\gamma(t) = (\gamma_1(t), \gamma_2(t)) \,.$$

The curve γ needs to be chosen so that it parametrizes the set C, in the sense that

$$C = \{(\gamma_1(t), \gamma_2(t)) \,:\, t \in (a, b)\} \,. \tag{1.1}$$

Thus we need to have

$$(x, y) = (\gamma_1, \gamma_2) \,. \tag{1.2}$$

How do we define such γ? Note that the points (x, y) in C satisfy

$$(x, y) \in C \iff y = 2x + 1 \,.$$

Therefore, using (1.2), we have that

$$\gamma_1 = x \,, \quad \gamma_2 = y = 2x + 1$$

from which we deduce that γ must satisfy

$$\gamma_2(t) = 2\gamma_1(t) + 1 \tag{1.3}$$

for all $t \in (a, b)$. We can then choose

$$\gamma_1(t) := t \,,$$

and from (1.3) we deduce that

$$\gamma_2(t) = 2t + 1 \,.$$

This choice of γ works:

$$\begin{aligned} C &= \{(x, 2x + 1) \,:\, x \in \mathbb{R}\} && (1.4) \\ &= \{(t, 2t + 1) \,:\, -\infty < t < \infty\} && (1.5) \\ &= \{(\gamma_1(t), \gamma_2(t)) \,:\, -\infty < t < \infty\} \,, && (1.6) \end{aligned}$$

where in the second line we just swapped the symbol x with the symbol t. In this case we have to choose the time interval as

$$(a, b) = (-\infty, \infty) \,.$$

In this way γ satisfies (1.1) and we have successfully parametrized the straight line C.

Remark 1.5: Parametrization is not unique

Let us consider again the straight line

$$C = \{(x, y) \in \mathbb{R}^2 \, : \, 2x + 1 = y\} \, .$$

We saw that $\gamma \colon (-\infty, \infty) \to \mathbb{R}^2$ defined by

$$\gamma(t) := (t, 2t + 1)$$

is a parametrization of C. But of course any γ satisfying

$$\gamma_2(t) = 2\gamma_1(t) + 1$$

would yield a parametrization of C. For example one could choose

$$\gamma_1(t) = 2t \, , \quad \gamma_2(t) = 2\gamma_1(t) + 1 = 4t + 1 \, .$$

In general, any time rescaling would work: the curve γ defined by

$$\gamma_1(t) = nt \, , \quad \gamma_2(t) = 2\gamma_1(t) + 1 = 2nt + 1$$

parametrizes C for all $n \in \mathbb{N}$. Hence there are **infinitely many** parametrizations of C.

Example 1.6: Parametrizing the circle

The circle C is described by all the points $(x, y) \in \mathbb{R}^2$ such that

$$x^2 + y^2 = 1 \, .$$

Therefore if we want to find a curve

$$\gamma = (\gamma_1, \gamma_2)$$

which parametrizes C, this has to satisfy

$$\gamma_1(t)^2 + \gamma_2(t)^2 = 1 \tag{1.7}$$

for all $t \in (a, b)$.
How to find such curve? We could proceed as in the previous example, and set

$$\gamma_1(t) := t \, .$$

Then (1.7) implies

$$\gamma_2(t) = \sqrt{1 - t^2} \, ,$$

from which we also deduce that

$$-1 \leq t \leq 1$$

are the only admissible values of t. However this curve does not represent the full circle C, but only the upper half, as seen in the plot below.
Simlarly, another solution to (1.7) would be γ with

$$\gamma_1(t) = t\,, \quad \gamma_2(t) = -\sqrt{1-t^2}\,,$$

for $t \in [-1, 1]$. However this choice does not parametrize the full circle C either, but only the bottom half, as seen in the plot below.
How to represent the whole circle? Recall the trigonometric identity

$$\cos(t)^2 + \sin(t)^2 = 1$$

for all $t \in \mathbb{R}$. This suggests to choose γ as

$$\gamma_1(t) := \cos(t)\,, \quad \gamma_2(t) := \sin(t)$$

for $t \in [0, 2\pi)$. This way γ satisfies (1.7), and actually parametrizes C, as shown below.
Note the following:

- If we had chosen $t \in [0, 4\pi]$ then γ would have covered C twice.
- If we had chosen $t \in [0, \pi]$, then γ would have covered the upper semi-circle
- If we had chosen $t \in [\pi, 2\pi]$, then γ would have covered the lower semi-circle
- Similarly, we can choose $t \in [\pi/6, \pi/2]$ to cover just a portion of C, as shown below.

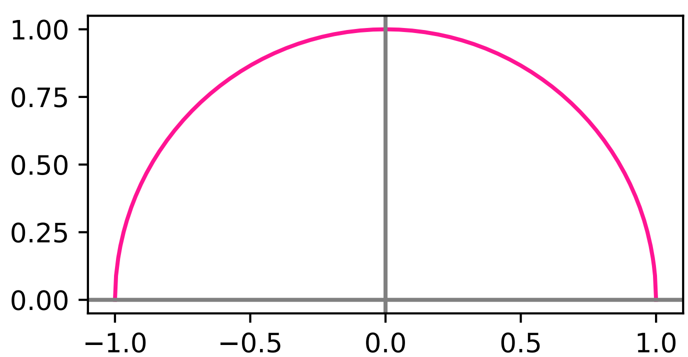


Figure 1.5: Upper semi-circle

Finally we are also able to give a mathematical description of the 3D Helix.

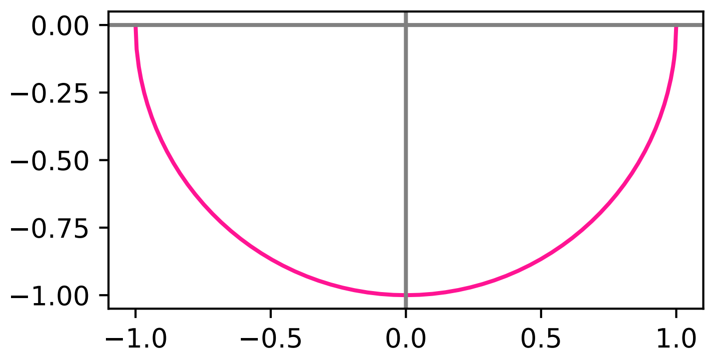


Figure 1.6: Lower semi-circle

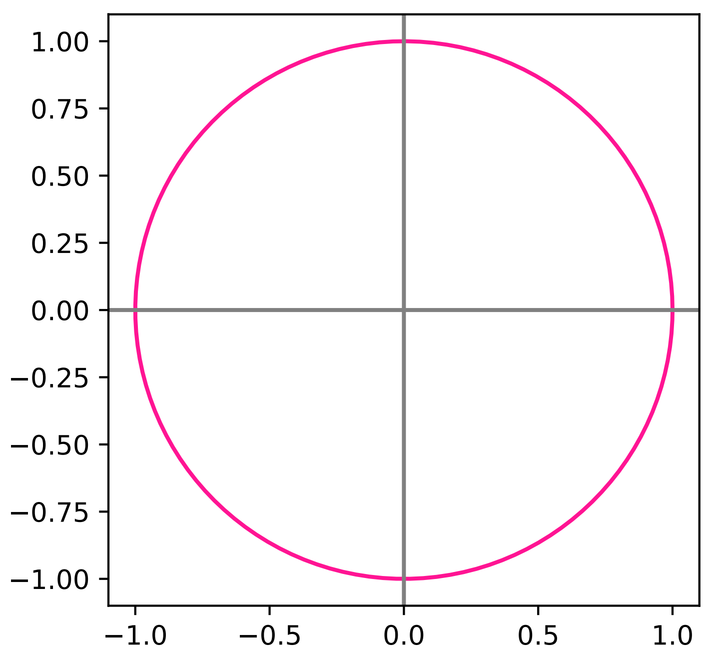


Figure 1.7: Lower semi-circle

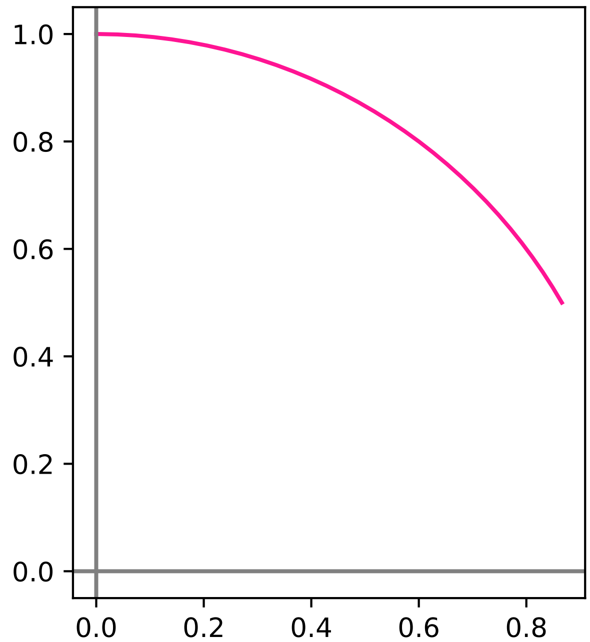


Figure 1.8: Plotting a portion of C

Example 1.7: Parametrizing the helix

The Helix plotted above can be parametrized by

$$\gamma : (-\infty, \infty) \to \mathbb{R}^3$$

defined by

$$\gamma_1(t) = \cos(t)\,, \;\; \gamma_2(t) = \sin(t)\,, \;\; \gamma_3(t) = t\,.$$

The above equations are in line with our intuition: the helix can be drawn by *tracing a circle while at the same time lifting the pencil.*

2 Curves in Python

2.1 Curves in 2D

Suppose we want to plot the parabola $y = t^2$ for t in the interval $[-3, 3]$. In our language, this is the two-dimensional curve

$$\gamma(t) = (t, t^2)\,, \quad t \in [-3, 3]\,.$$

The two Python libraries we use to plot γ are **numpy** and **matplotlib**. In short, **numpy** handles multi-dimensional arrays and matrices, and can perform high-level mathematical functions on them. For any question you may have about numpy, answers can be found in the searchable documentation available here. Instead **matplotlib** is a plotting library, with documentation here. Python libraries need to be imported every time you want to use them. In our case we will import:

```
import numpy as np
import matplotlib.pyplot as plt
```

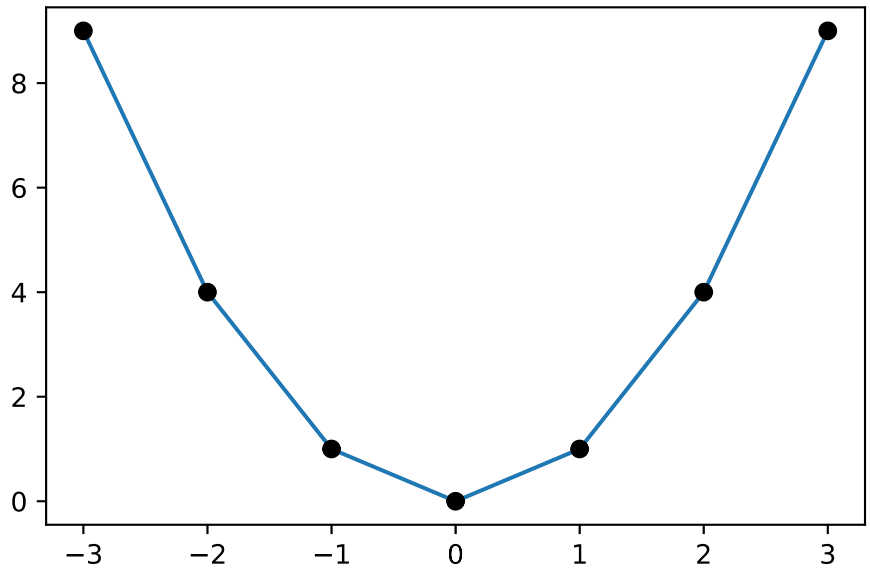
The above imports **numpy** and the module **pyplot** from **matplotlib**, and renames them to `np` and `plt`, respectively. These shorthands are standard in the literature, and they make code much more readable. The function for plotting 2D graphs is called `plot(x,y)` and is contained in `plt`. As the syntax suggests, `plot` takes as arguments two arrays

$$x = [x_1, \dots, x_n]\,, \quad y = [y_1, \dots, y_n]\,.$$

As output it produces a graph which is the linear interpolation of the points (x_i, y_i) in $\mathbb{R}^2$, that is, consecutive points (x_i, y_i) and (x_{i+1}, y_{i+1}) are connected by a segment. Using `plot`, we can graph the curve $\gamma(t) = (t, t^2)$ like so:

```
# Code for plotting gamma

import numpy as np
import matplotlib.pyplot as plt

# Generating array t
t = np.array([-3,-2,-1,0,1,2,3])

# Computing array f
f = t**2
```

```
# Plotting the curve
plt.plot(t,f)

# Plotting dots
plt.plot(t,f,"ko")

# Showing the plot
plt.show()
```



Let us comment the above code. The variable `t` is a numpy array containing the ordered values

$$t = [-3, -2, -1, 0, 1, 2, 3] \,. \tag{2.1}$$

This array is then squared entry-by-entry via the operation $t\,{*}{*}2$ and saved in the new numpy array `f`, that is,

$$f = [9, 4, 1, 0, 1, 4, 9] \,.$$

The arrays `t` and `f` are then passed to `plot(t,f)`, which produces the above linear interpolation, with `t` on the *x-axis* and `f` on the *y-axis*. The command `plot(t,f,'ko')` instead plots a black dot at each point (t_i, f_i). The latter is clearly not needed to obtain a plot, and it was only included to highlight the fact that `plot` is actually producing a linear interpolation between points. Finally `plt.show()` displays the figure in the user window[1].
Of course one can refine the plot so that it resembles the continuous curve $\gamma(t) = (t, t^2)$ that we all have in mind. This is achieved by generating a numpy array `t` with a finer stepsize, invoking the function `np.linspace(a,b,n)`. Such call will return a numpy array which contains `n` evenly spaced points, starts

[1] The command `plt.show()` can be omitted if working in Jupyter Notebook, as it is called by default.

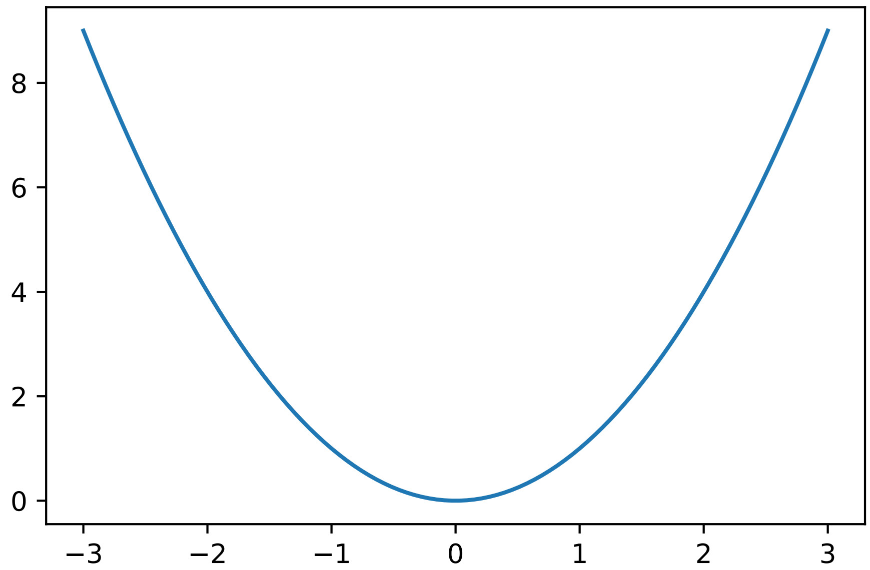
at `a`, and ends in `b`. For example `np.linspace(-3,3,7)` returns our original array `t` at 2.1, as shown below

```
# Displaying output of np.linspace

import numpy as np

# Generates array t by dividing interval
# (-3,3) in 7 parts
t = np.linspace(-3,3, 7)

# Prints array t
print("t =", t)
```

```
t = [-3. -2. -1.  0.  1.  2.  3.]
```

In order to have a more refined plot of γ, we just need to increase n.

```
# Plotting gamma with finer step-size

import numpy as np
import matplotlib.pyplot as plt

# Generates array t by dividing interval
# (-3,3) in 100 parts
t = np.linspace(-3,3, 100)

# Computes f
f = t**2

# Plotting
plt.plot(t,f)
plt.show()
```



We now want to plot a parametric curve $\gamma\colon (a,b) \to \mathbb{R}^2$ with

$$\gamma(t) = (x(t), y(t)) \,.$$

Clearly we need to modify the above code. The variable `t` will still be a numpy array produced by `linspace`. We then need to introduce the arrays `x` and `y` which ecode the first and second components of γ, respectively.

```
import numpy as np
import matplotlib.pyplot as plt

# Divides time interval (a,b) in n parts
# and saves output to numpy array t
t = np.linspace(a, b, n)

# Computes gamma from given functions x(y) and y(t)
x = x(t)
y = y(t)

# Plots the curve
plt.plot(x,y)

# Shows the plot
plt.show()
```

We use the above code to plot the 2D curve known as the Fermat's spiral

$$\gamma(t) = (\sqrt{t}\cos(t), \sqrt{t}\sin(t)) \quad \text{for} \quad t \in [0,50] \,. \tag{2.2}$$

```
# Plotting Fermat's spiral

import numpy as np
import matplotlib.pyplot as plt

# Divides time interval (0,50) in 500 parts
t = np.linspace(0, 50, 500)

# Computes Fermat's Spiral
x = np.sqrt(t) * np.cos(t)
y = np.sqrt(t) * np.sin(t)

# Plots the Spiral
plt.plot(x,y)
plt.show()
```

Before displaying the output of the above code, a few comments are in order. The array `t` has size 500, due to the behavior of `linspace`. You can also fact check this information by printing `np.size(t)`, which is the numpy function that returns the size of an array. We then use the numpy function `np.sqrt` to compute the square root of the array `t`. The outcome is still an array with the same size of `t`, that is,

$$t = [t_1, \dots, t_n] \quad \Longrightarrow \quad \sqrt{t} = [\sqrt{t_1}, \dots, \sqrt{t_n}] \,.$$

Similary, the call `np.cos(t)` returns the array

$$\cos(t) = [\cos(t_1), \dots, \cos(t_n)] \,.$$

The two arrays `np.sqrt(t)` and `np.cos(t)` are then multiplied, term-by-term, and saved in the array `x`. The array `y` is computed similarly. The command `plt.plot(x,y)` then yields the graph of the Fermat's spiral:

The above plots can be styled a bit. For example we can give a title to the plot, label the axes, plot the spiral by means of green dots, and add a plot legend, as coded below:

```
# Adding some style

import numpy as np
import matplotlib.pyplot as plt

# Computing Spiral
t = np.linspace(0, 50, 500)
x = np.sqrt(t) * np.cos(t)
y = np.sqrt(t) * np.sin(t)

# Generating figure
```

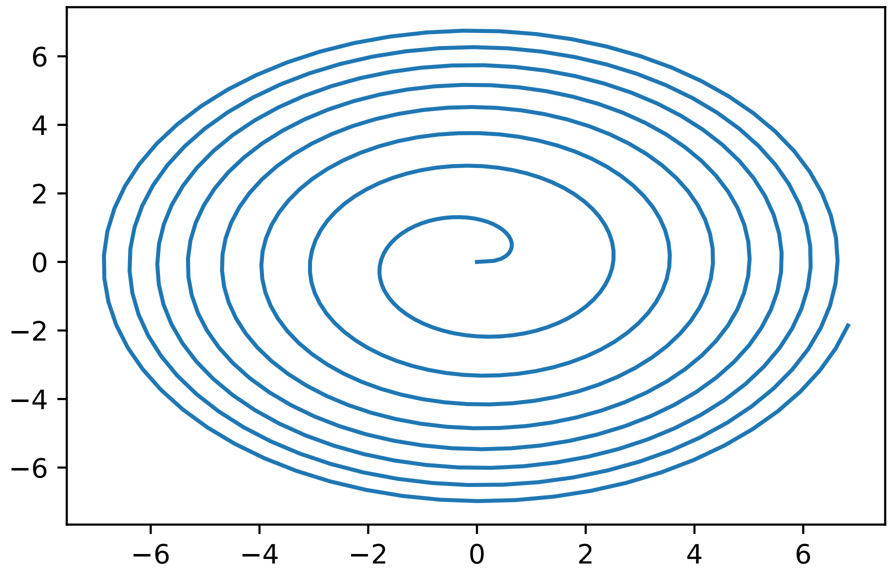


Figure 2.1: Fermat's spiral

```
plt.figure(1, figsize = (4,4))

# Plotting the Spiral with some options
plt.plot(x, y, '--', color = 'deeppink', linewidth = 1.5, label = 'Spiral')

# Adding grid
plt.grid(True, color = 'lightgray')

# Adding title
plt.title("Fermat's spiral for t between 0 and 50")

# Adding axes labels
plt.xlabel("x-axis", fontsize = 15)
plt.ylabel("y-axis", fontsize = 15)

# Showing plot legend
plt.legend()

# Show the plot
plt.show()
```

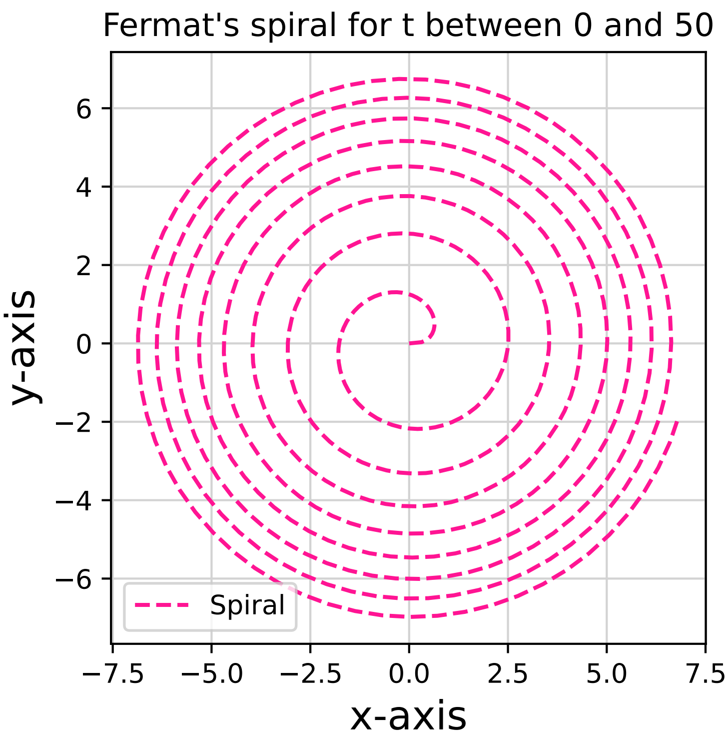


Figure 2.2: Adding a bit of style

Let us go over the novel part of the above code:

- `plt.figure()`: This command generates a figure object. If you are planning on plotting just one figure at a time, then this command is optional: a figure object is generated implicitly when calling `plt.plot`. Otherwise, if working with `n` figures, you need to generate a figure object with `plt.figure(i)` for each `i` between `1` and `n`. The number `i` uniquely identifies the i-th figure: whenever you call `plt.figure(i)`, Python knows that the next commands will refer to the i-th figure. In our case we only have one figure, so we have used the identifier `1`. The second argument `figsize = (a,b)` in `plt.figure()` specifies the size of `figure 1` in inches. In this case we generated a figure 4 x 4 inches.
- `plt.plot`: This is plotting the arrays `x` and `y`, as usual. However we are adding a few aestethic touches: the curve is plotted in *dashed* style with `--`, in *deep pink* color and with a line width of 1.5. Finally this plot is labelled *Spiral*.
- `plt.grid`: This enables a grid in *light gray* color.
- `plt.title`: This gives a title to the figure, displayed on top.
- `plt.xlabel` and `plt.ylabel`: These assign labels to the axes, with font size 15 points.
- `plt.legend()`: This plots the legend, with all the labels assigned in the `plt.plot` call. In this case the only label is *Spiral*.

Matplotlib styles

There are countless plot types and options you can specify in **matplotlib**, see for example the Matplotlib Gallery. Of course there is no need to remember every single command: a quick Google search can do wonders.

Generating arrays

There are several ways of generating evenly spaced arrays in Python. For example the function `np.arange(a,b,s)` returns an array with values within the half-open interval $[a, b)$, with spacing between values given by `s`. For example

```
import numpy as np

t = np.arange(0,1, 0.2)
print("t =",t)
```

```
t = [0.  0.2 0.4 0.6 0.8]
```

2.2 Implicit curves 2D

A curve γ in $\mathbb{R}^2$ can also be defined as the set of points $(x, y) \in \mathbb{R}^2$ satisfying

$$f(x, y) = 0$$

for some given $f \colon \mathbb{R}^2 \to \mathbb{R}$. For example let us plot the curve γ implicitly defined by

$$f(x, y) = (3x^2 - y^2)^2 \, y^2 - (x^2 + y^2)^4$$

for $-1 \leq x, y \leq 1$. First, we need a way to generate a grid in $\mathbb{R}^2$ so that we can evaluate f on such grid. To illustrate how to do this, let us generate a grid of spacing 1 in the 2D square $[0, 4]^2$. The goal is to obtain the 5 x 5 matrix of coordinates

$$A = \begin{pmatrix} (0,0) & (1,0) & (2,0) & (3,0) & (4,0) \\ (0,1) & (1,1) & (2,1) & (3,1) & (4,1) \\ (0,2) & (1,2) & (2,2) & (2,3) & (2,4) \\ (0,3) & (1,3) & (2,3) & (3,3) & (3,4) \\ (0,4) & (1,4) & (2,4) & (3,4) & (4,4) \end{pmatrix}$$

which corresponds to the grid of points

To achieve this, first generate `x` and `y` coordinates using

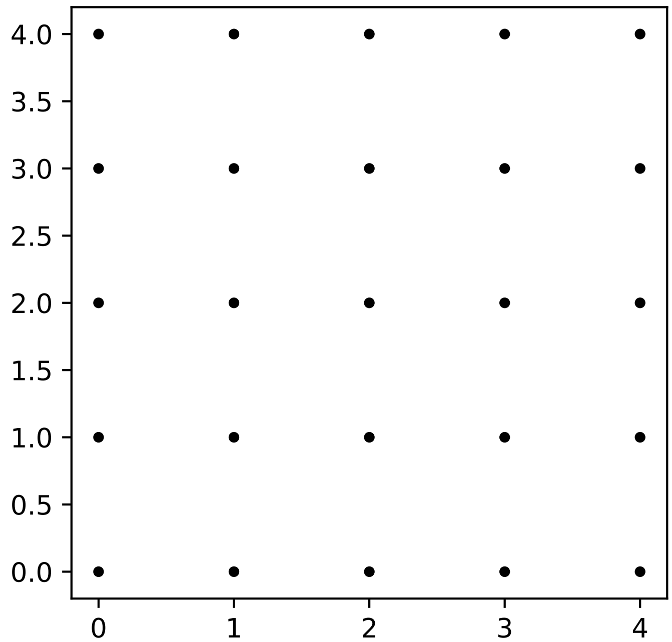


Figure 2.3: The 5 x 5 grid corresponding to the matrix A

```
x = np.linspace(0, 4, 5)
y = np.linspace(0, 4, 5)
```

This generates coordinates

$$x = [0, 1, 2, 3, 4] \,, \quad y = [0, 1, 2, 3, 4] \,.$$

We then need to obtain two matrices X and Y: one for the x coordinates in A, and one for the y coordinates in A. This can be achieved with the code

```
X[0,0] = 0
X[0,1] = 1
X[0,2] = 2
X[0,3] = 3
X[0,4] = 4
X[1,0] = 0
X[1,1] = 1
...
x[4,3] = 3
x[4,4] = 4
```

and similarly for Y. The output would be the two matrices X and Y

$$X = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{pmatrix}$$

If now we plot X against Y via the command

```
plt.plot(X, Y, 'k.')
```

we obtain Figure 2.3. In the above command the style `'k.'` represents black dots. This procedure would be impossible with large vectors. Thankfully there is a function in numpy doing exactly what we need: `np.meshgrid`.

```
# Demonstrating np.meshgrid

import numpy as np

# Generating x and y coordinates
xlist = np.linspace(0, 4, 5)
ylist = np.linspace(0, 4, 5)

# Generating grid X, Y
X, Y = np.meshgrid(xlist, ylist)

# Printing the matrices X and Y
# np.array2string is only needed to align outputs
print('X =', np.array2string(X, prefix='X= '))
print('\n')
print('Y =', np.array2string(Y, prefix='Y= '))
```

```
X = [[0. 1. 2. 3. 4.]
    [0. 1. 2. 3. 4.]
    [0. 1. 2. 3. 4.]
    [0. 1. 2. 3. 4.]
    [0. 1. 2. 3. 4.]]


Y = [[0. 0. 0. 0. 0.]
    [1. 1. 1. 1. 1.]
    [2. 2. 2. 2. 2.]
    [3. 3. 3. 3. 3.]
```

```
 [4. 4. 4. 4. 4.]]
```

Now that we have our grid, we can evaluate the function f on it. This is simply done with the command

```
Z =((3*(X**2) - Y**2)**2)*(Y**2) - (X**2 + Y**2)**4
```

This will return the matrix Z containing the values $f(x_i, y_i)$ for all (x_i, y_i) in the grid $[X, Y]$. We are now interested in plotting the points in the grid $[X, Y]$ for which Z is zero. This is achieved with the command

```
plt.contour(X, Y, Z, [0])
```

Putting the above observations together, we have the code for plotting the curve $f = 0$ for $-1 \leq x, y \leq 1$.

```
# Plotting f=0

import numpy as np
import matplotlib.pyplot as plt

# Generates coordinates and grid
xlist = np.linspace(-1, 1, 5000)
ylist = np.linspace(-1, 1, 5000)
X, Y = np.meshgrid(xlist, ylist)

# Computes f
Z =((3*(X**2) - Y**2)**2)*(Y**2) - (X**2 + Y**2)**4

# Creates figure object
plt.figure(figsize = (4,4))

# Plots level set Z = 0
plt.contour(X, Y, Z, [0])

# Set axes labels
plt.xlabel("x-axis", fontsize = 15)
plt.ylabel("y-axis", fontsize = 15)

# Shows plot
plt.show()
```

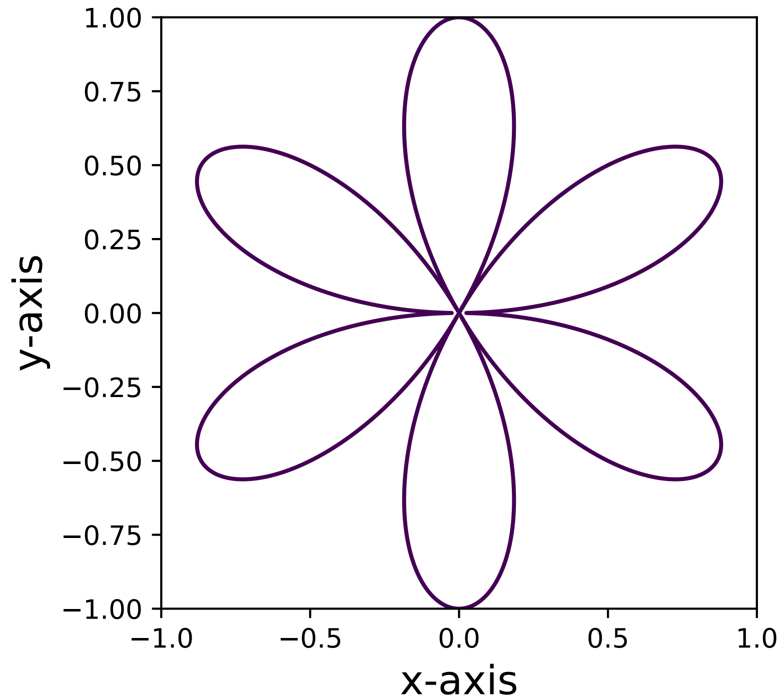


Figure 2.4: Plot of the curve defined by f=0

2.3 Curves in 3D

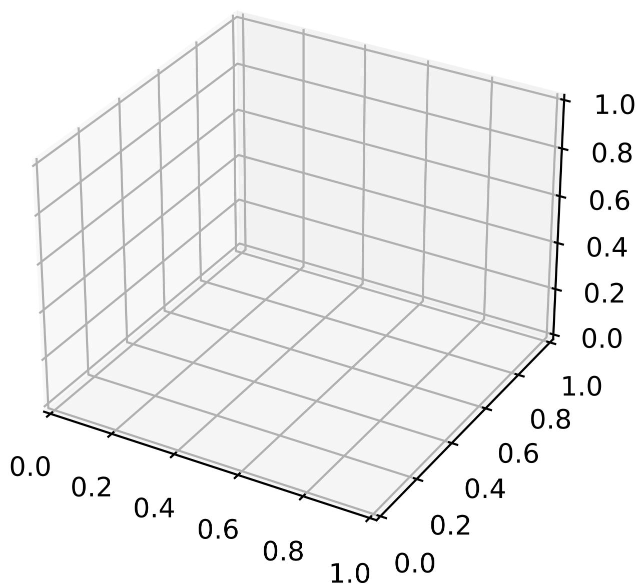
Plotting in 3D with matplotlib requires the `mplot3d` toolkit, see here for documentation. Therefore our first lines will always be

```
# Packages for 3D plots

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
```

We can now generate empty 3D axes

```
# Generates and plots empty 3D axes

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

# Creates figure object
```

```
fig = plt.figure(figsize = (4,4))

# Creates 3D axes object
ax = plt.axes(projection = '3d')

# Shows the plot
plt.show()
```



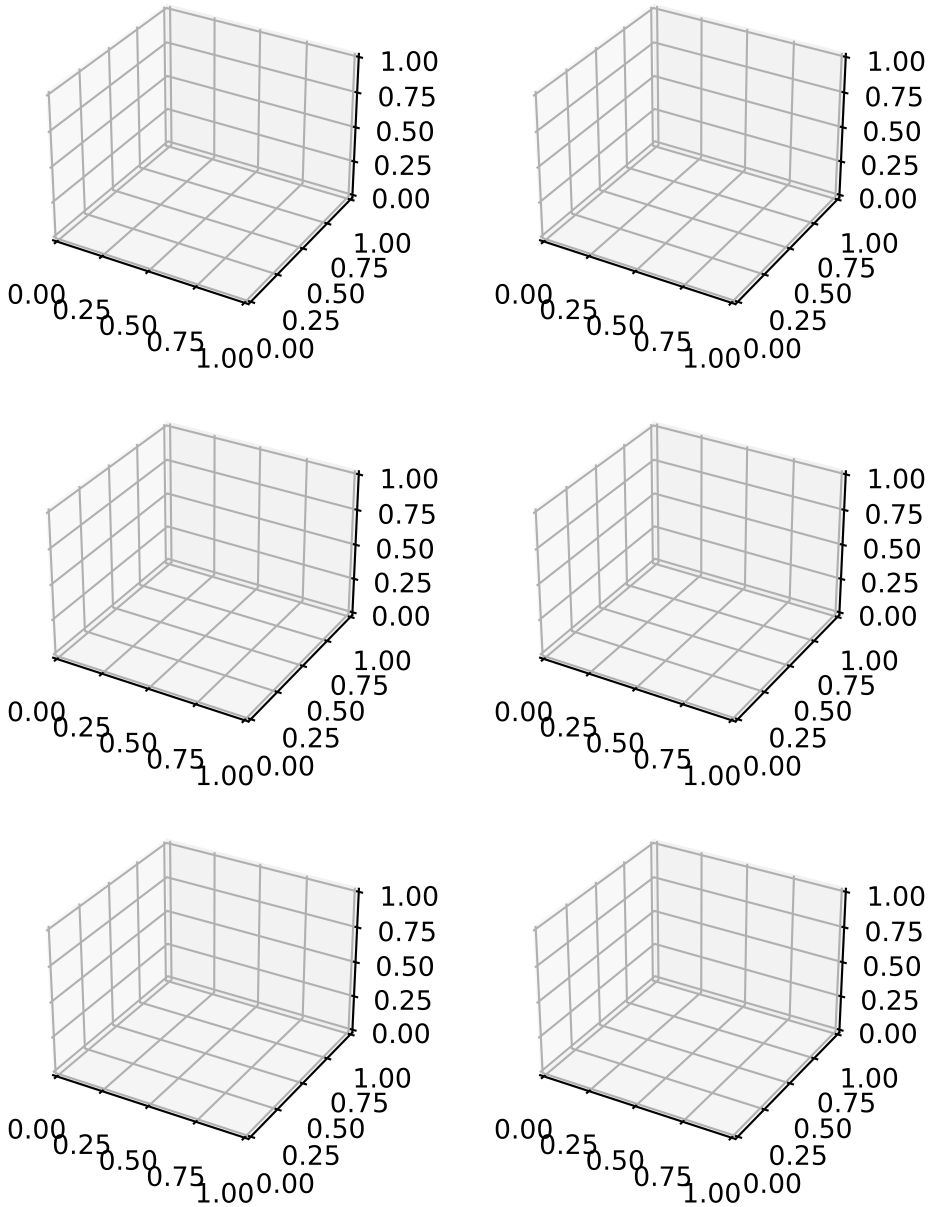
In the above code `fig` is a figure object, while `ax` is an axes object. In practice, the figure object contains the axes objects, and the actual plot information will be contained in axes. If you want multiple plots in the figure container, you should use the command

```
ax = fig.add_subplot(nrows = m, ncols = n, pos = k)
```

This generates an axes object `ax` in position `k` with respect to a `m x n` grid of plots in the container figure. For example we can create a 3 x 2 grid of empty 3D axes as follows

```
# Generates 3 x 2 empty 3D axes

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

# Creates container figure object
```

```
fig = plt.figure(figsize = (6,8))

# Creates 6 empty 3D axes objects
ax1 = fig.add_subplot(3, 2, 1, projection = '3d')
ax2 = fig.add_subplot(3, 2, 2, projection = '3d')
ax3 = fig.add_subplot(3, 2, 3, projection = '3d')
ax4 = fig.add_subplot(3, 2, 4, projection = '3d')
ax5 = fig.add_subplot(3, 2, 5, projection = '3d')
ax6 = fig.add_subplot(3, 2, 6, projection = '3d')

# Shows the plot
plt.show()
```



We are now ready to plot a 3D parametric curve $\gamma \colon (a,b) \to \mathbb{R}^3$ of the form

$$\gamma(t) = (x(t), y(t), z(t))$$

with the code

```
# Code to plot 3D curve

import numpy as np
import matplotlib.pyplot as plt
```

```
from mpl_toolkits import mplot3d

# Generates figure and 3D axes
fig = plt.figure(figsize = (size1,size2))
ax = plt.axes(projection = '3d')

# Plots grid
ax.grid(True)

# Divides time interval (a,b)
# into n parts and saves them in array t
t = np.linspace(a, b, n)

# Computes the curve gamma on array t
# for given functions x(t), y(t), z(t)
x = x(t)
y = y(t)
z = z(t)

# Plots gamma
ax.plot3D(x, y, z)

# Setting title for plot
ax.set_title('3D Plot of gamma')

# Setting axes labels
ax.set_xlabel('x', labelpad = 'p')
ax.set_ylabel('y', labelpad = 'p')
ax.set_zlabel('z', labelpad = 'p')

# Shows the plot
plt.show()
```

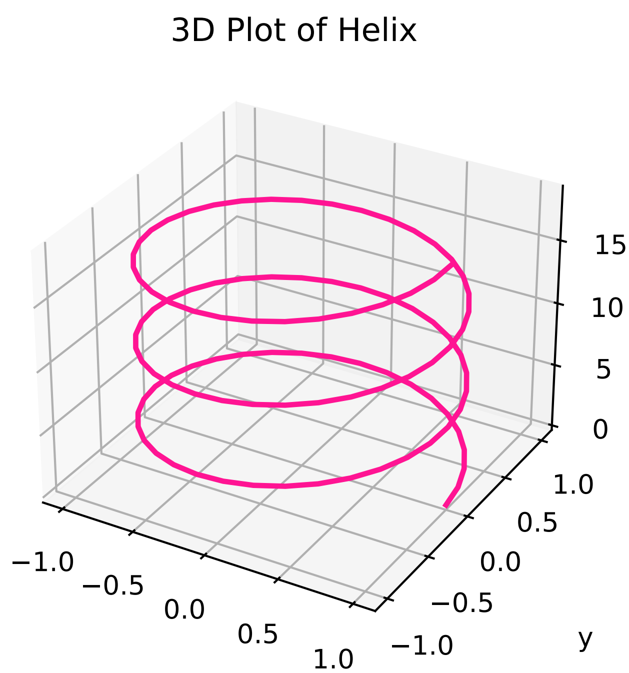
For example we can use the above code to plot the Helix

$$x(t) = \cos(t)\,, \quad y(t) = \sin(t)\,, \quad z(t) = t \tag{2.3}$$

for $t \in [0, 6\pi]$.

```
# Plotting 3D Helix

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
```

```python
# Generates figure and 3D axes
fig = plt.figure(figsize = (4,4))
ax = plt.axes(projection = '3d')

# Plots grid
ax.grid(True)

# Divides time interval (0,6pi) in 100 parts
t = np.linspace(0, 6*np.pi, 100)

# Computes Helix
x = np.cos(t)
y = np.sin(t)
z = t

# Plots Helix - We added some styling
ax.plot3D(x, y, z, color = "deeppink", linewidth = 2)

# Setting title for plot
ax.set_title('3D Plot of Helix')

# Setting axes labels
ax.set_xlabel('x', labelpad = 20)
ax.set_ylabel('y', labelpad = 20)
ax.set_zlabel('z', labelpad = 20)

# Shows the plot
plt.show()
```



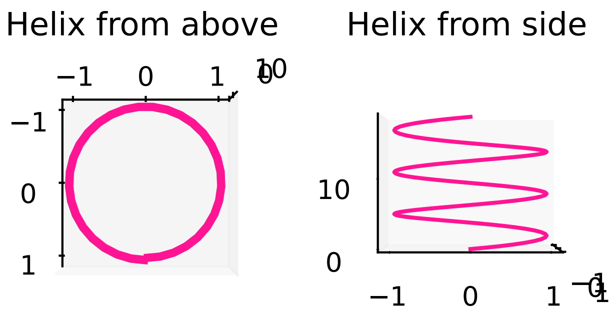
We can also change the viewing angle for a 3D plot store in `ax`. This is done via

```
ax.view_init(elev = e, azim = a)
```

which displays the 3D axes with an elevation angle `elev` of `e` degrees and an azimuthal angle `azim` of `a` degrees. In other words, the 3D plot will be rotated by `e` degrees above the xy-plane and by `a` degrees around the z-axis. For example, let us plot the helix with 2 viewing angles. Note that we generate 2 sets of axes with the `add_subplot` command discussed above.

```
# Plotting 3D Helix

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

# Generates figure object
fig = plt.figure(figsize = (4,4))

# Generates 2 sets of 3D axes
ax1 = fig.add_subplot(1, 2, 1, projection = '3d')
ax2 = fig.add_subplot(1, 2, 2, projection = '3d')

# We will not show a grid this time
ax1.grid(False)
```

```
ax2.grid(False)

# Divides time interval (0,6pi) in 100 parts
t = np.linspace(0, 6*np.pi, 100)

# Computes Helix
x = np.cos(t)
y = np.sin(t)
z = t

# Plots Helix on both axes
ax1.plot3D(x, y, z, color = "deeppink", linewidth = 1.5)
ax2.plot3D(x, y, z, color = "deeppink", linewidth = 1.5)

# Setting title for plots
ax1.set_title('Helix from above')
ax2.set_title('Helix from side')

# Changing viewing angle of ax1
# View from above has elev = 90 and azim = 0
ax1.view_init(elev = 90, azim = 0)

# Changing viewing angle of ax2
# View from side has elev = 0 and azim = 0
ax2.view_init(elev = 0, azim = 0)

# Shows the plot
plt.show()
```



2.4 Interactive plots

`Matplotlib` produces beautiful static plots; however it lacks built in interactivity. For this reason I would also like to show you how to plot curves with `Plotly`, a very popular Python graphic library which has built in interactivity. Documentation for `Plotly` and lots of examples can be found here.

2.4.1 2D Plots

Say we want to plot the 2D curve $\gamma\colon (a,b) \to \mathbb{R}^2$ parametrized by

$$\gamma(t) = (x(t), y(t)) \,.$$

The `Plotly` module needed is called `graph_objects`, usually imported as `go`. The function for line plots is called `Scatter`. For documentation and examples see link. The code for plotting γ is as follows.

```
# Plotting gamma 2D

# Import libraries
import numpy as np
import plotly.graph_objects as go

# Compute times grid by dividing (a,b) in
# n equal parts
t = np.linspace(a, b, n)

# Compute the parametric curve gamma
# for given functions x(t) and y(t)
x = x(t)
y = y(t)

# Create empty figure object and saves
# it in the variable "fig"
fig = go.Figure()

# Create the line plot object
data = go.Scatter(x = x, y = y, mode = 'lines', name = 'gamma')

# Add "data" plot to the figure "fig"
fig.add_trace(data)

# Display the figure
fig.show()
```

Some comments about the functions called above:

- `go.Figure`: generates an empty Plotly figure
- `go.Scatter`: generates the actual plot. By default a scatter plot is produced. To obtain linear interpolation of the points, set `mode = 'lines'`. You can also label the plot with `name = "string"`
- `add_trace`: adds a plot to a figure
- `show`: displays a figure

As an example, let us plot the Fermat's Spiral defined at 2.2. Compared to the above code, we also add a bit of styling.

```
# Plotting Fermat's Spiral

# Import libraries
import numpy as np
import plotly.graph_objects as go

# Compute times grid by dividing (0,50) in
# 500 equal parts
t = np.linspace(0, 50, 500)

# Computes Fermat's Spiral
x = np.sqrt(t) * np.cos(t)
y = np.sqrt(t) * np.sin(t)

# Create empty figure object and saves
# it in the variable "fig"
fig = go.Figure()

# Create the line plot object
data = go.Scatter(x = x, y = y, mode = 'lines', name = 'gamma')

# Add "data" plot to the figure "fig"
fig.add_trace(data)

# Here we start with the styling options
# First we set a figure title
fig.update_layout(title_text = "Plotting Fermat's Spiral with Plotly")

# Adjust figure size
fig.update_layout(autosize = False, width = 600, height = 600)

# Change background canvas color
fig.update_layout(paper_bgcolor = "snow")
```

```
# Axes styling: adding title and ticks positions
fig.update_layout(
xaxis=dict(
        title_text="X-axis Title",
        titlefont=dict(size=20),
        tickvals=[-6,-4,-2,0,2,4,6],
        ),

yaxis=dict(
        title_text="Y-axis Title",
        titlefont=dict(size=20),
        tickvals=[-6,-4,-2,0,2,4,6],
        )
)

# Display the figure
fig.show()
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): text/html
```

The above code generates an image that cannot be rendered in pdf. To see the output, please click here for the digital version of these notes. Note that the style customizations could be listed in a single call of the function `update_layout`. There are also pretty buit-in themes available, see here. The layout can be specified with the command

```
fig.update_layout(template = template_name)
```

where `template_name` can be `"plotly"`, `"plotly_white"`, `"plotly_dark"`, `"ggplot2"`, `"seaborn"`, `"simple_white"`".

2.4.2 3D Plots

We now want to plot a 3D curve $\gamma\colon (a,b) \to \mathbb{R}^3$ parametrized by

$$\gamma(t) = (x(t), y(t), z(t)) \,.$$

Again we use the `Plotly` module `graph_objects`, imported as `go`. The function for 3D line plots is called `Scatter3d`, and documentation and examples can be found at link. The code for plotting γ is as follows.

```
# Plotting gamma 3D

# Import libraries
import numpy as np
import plotly.graph_objects as go

# Compute times grid by dividing (a,b) in
# n equal parts
t = np.linspace(a, b, n)

# Compute the parametric curve gamma
# for given functions x(t), y(t), z(t)
x = x(t)
y = y(t)
z = z(t)

# Create empty figure object and saves
# it in the variable "fig"
fig = go.Figure()

# Create the line plot object
data = go.Scatter3d(x = x, y = y, z = z, mode = 'lines', name = 'gamma')

# Add "data" plot to the figure "fig"
fig.add_trace(data)

# Display the figure
fig.show()
```

The functions `go.Figure`, `add_trace` and `show` appearing above are described in the previous Section. The new addition is `go.Scatter3d`, which generates a 3D scatter plot of the points stored in the array `[x,y,z]`. Setting `mode = 'lines'` results in a linear interpolation of such points. As before, the curve can be labeled by setting `name = "string"`.

As an example, we plot the 3D Helix defined at 2.3. We also add some styling. We can also use the same pre-defined templates described for `go.Scatter` in the previous section, see here for official documentation.

```
# Plotting 3D Helix

# Import libraries
import numpy as np
import plotly.graph_objects as go
```

```python
# Divides time interval (0,6pi) in 100 parts
t = np.linspace(0, 6*np.pi, 100)

# Computes Helix
x = np.cos(t)
y = np.sin(t)
z = t

# Create empty figure object and saves
# it in the variable "fig"
fig = go.Figure()

# Create the line plot object
# We add options for the line width and color
data = go.Scatter3d(
    x = x, y = y, z = z,
    mode = 'lines', name = 'gamma',
    line = dict(width = 10, color = "darkblue")
    )

# Add "data" plot to the figure "fig"
fig.add_trace(data)

# Here we start with the styling options
# First we set a figure title
fig.update_layout(title_text = "Plotting 3D Helix with Plotly")

# Adjust figure size
fig.update_layout(
    autosize = False,
    width = 600,
    height = 600
    )

# Set pre-defined template
fig.update_layout(template = "seaborn")

# Options for curve line style


# Display the figure
fig.show()
```

```
Unable to display output for mime type(s): text/html
```

The above code generates an image that cannot be rendered in pdf. To see the output, please click here for the digital version of these notes. Once again, the style customizations could be listed in a single call of the function `update_layout`.

3 Parametrized curves

Let us recall the definition of **parametrized curve**.

Definition 3.1: Parametrized curve

A **parametrized curve** in $\mathbb{R}^n$ is a function

$$\gamma : (a,b) \to \mathbb{R}^n .$$

where

$$(a,b) = \{t \in \mathbb{R} : \ a < t < b\} ,$$

with

$$-\infty \leq a < b \leq \infty .$$

The **components** of $\gamma(t) \in \mathbb{R}^n$ are denoted by

$$\gamma(t) = (\gamma_1(t), \dots, \gamma_n(t)) ,$$

where the components are functions

$$\gamma_i : (a,b) \to \mathbb{R} ,$$

for all $i = 1, \dots, n$.

As we already mentioned, the aim of the course is to study curves by **differentiating** them. Let us see what that means for curves.

Definition 3.2: Smooth functions

A scalar function $f : (a,b) \to \mathbb{R}$ is called **smooth** if the derivative

$$\frac{d^n f}{dt^n}$$

exists for all $n \geq 1$ and $t \in (a,b)$.

We will denote the first and second derivatives of f as follows:

$$\dot{f} := \frac{df}{dt}\,, \quad \ddot{f} := \frac{d^2 f}{dt^2}\,.$$

Example 3.3

The function $f(x) = x^4$ is smooth, with

$$\frac{df}{dt} = 4x^3, \ \frac{d^2 f}{dt^2} = 12x^2\,, \tag{3.1}$$

$$\frac{d^3 f}{dt^3} = 24x, \ \frac{d^4 f}{dt^4} = 24\,, \tag{3.2}$$

$$\frac{d^n f}{dt^n} = 0 \ \text{ for all } \ n \geq 5\,. \tag{3.3}$$

Other examples smooth functions are polynomials, as well as

$$f(t) = \cos(t), \ f(t) = \sin(t)\,, \ f(t) = e^t\,.$$

Definition 3.4

Let $\gamma : (a, b) \to \mathbb{R}^n$ with

$$\gamma(t) = (\gamma_1(t), \dots, \gamma_n(t))$$

be a parametrized curve. We say that γ is **smooth** if the components

$$\gamma_i : (a, b) \to \mathbb{R}$$

are smooth for all $i = 1, \dots, n$. The derivatives of γ are

$$\frac{d^k \gamma}{dt^k} := \left(\frac{d^k \gamma_1}{dt^k}, \dots, \frac{d^k \gamma_n}{dt^k} \right)$$

for all $k \in \mathbb{N}$. As a shorthand, we will denote the first derivative of γ as

$$\dot{\gamma} := \frac{d\gamma}{dt} = \left(\frac{d\gamma_1}{dt}, \dots, \frac{d\gamma_n}{dt} \right)$$

and the second by

$$\ddot{\gamma} := \frac{d^2\gamma}{dt^2} = \left(\frac{d^2\gamma_1}{dt^2}, \dots, \frac{d^2\gamma_n}{dt^2} \right)\,.$$

In Figure 3.1 we skectch a smooth and a non-smooth curve. Notice that the curve on the right is smooth, except for the point x.

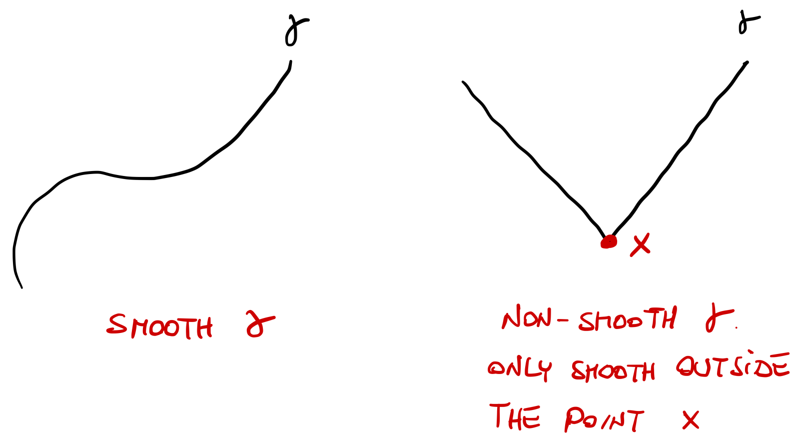


Figure 3.1: Example of smooth and non-smooth curves

We will work under the following assumption.

Assumption

All the parametrized curves in this lecture notes are assumed to be **smooth**.

Example 3.5

The circle

$$\gamma(t) = (\cos(t), \sin(t))$$

is a smooth parametrized curve, since both $\cos(t)$ and $\sin(t)$ are smooth functions. We have

$$\dot{\gamma} = (-\sin(t), \cos(t)) \,.$$

For example the derivative of γ at the point $(0,1)$ is given by

$$\dot{\gamma}(\pi/2) = (-\sin(\pi/2), \cos(\pi/2)) = (-1, 0) \,.$$

The plot of the circle and the derivative vector at $(-1,0)$ can be seen in Figure 3.2.

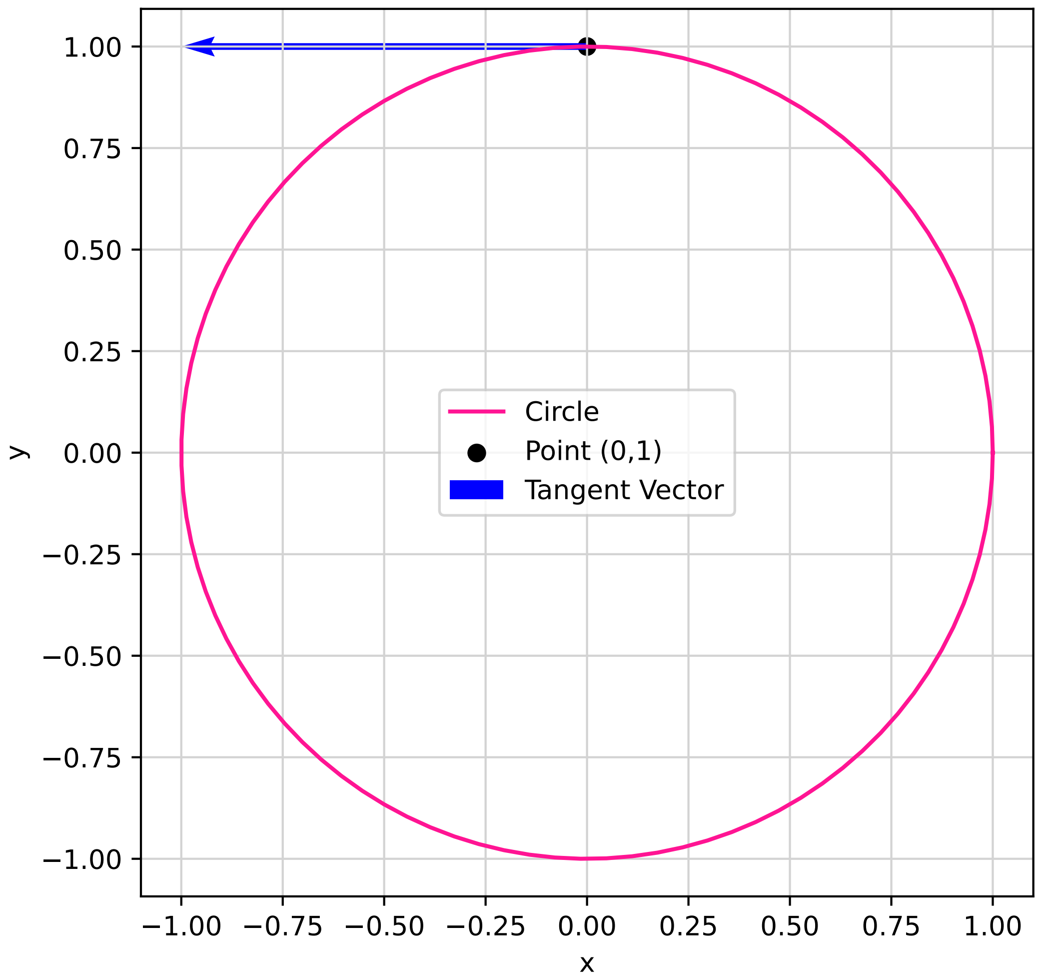


Figure 3.2: Plot of Circle and Tangent Vector at $(0, 1)$

3.1 Tangent vectors

Looking at Figure 3.2, it seems like the vector

$$\dot{\gamma}(\pi/2) = (-1, 0)$$

is **tangent** to the circle at the point

$$\gamma(\pi/2) = (0, 1)\,.$$

Is this a coincidence? Not that all. Let us look at the definition of derivative at a point:

$$\dot{\gamma}(t) := \lim_{\delta \to 0} \frac{\gamma(t+\delta) - \gamma(t)}{\delta}\,.$$

If we just look at the quantity

$$\frac{\gamma(t+\delta) - \gamma(t)}{\delta}$$

for non-negative δ, we see that this vector is parallel to the chord joining $\gamma(t)$ to $\gamma(t+\delta)$, as shown in Figure 3.3 below. As $\delta \to 0$, the length of the chord tends to zero. However the **direction** of the chord becomes **parallel** to that of the tangent vector of the curve γ at $\gamma(t)$. Since

$$\frac{\gamma(t+\delta) - \gamma(t)}{\delta} \to \dot{\gamma}(t)$$

as $\delta \to 0$, we see that $\dot{\gamma}(t)$ is **parallel** to the tangent of γ at $\gamma(t)$, as showin in Figure 3.3.

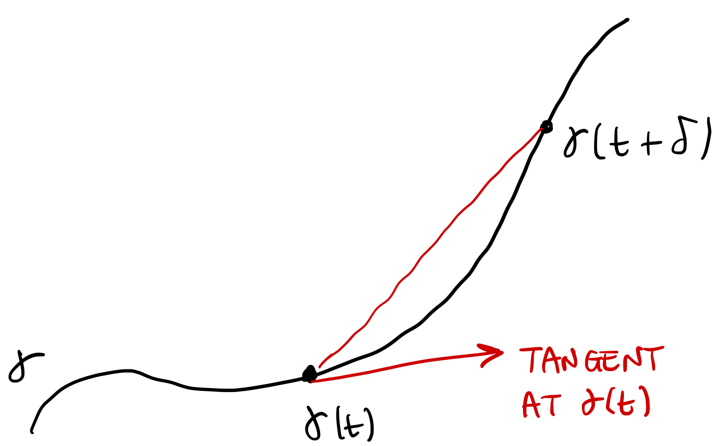


Figure 3.3: Approximating the tangent vector

The above remark motivates the following definition.

Definition 3.6: Tangent vector

Let $\gamma : (a,b) \to \mathbb{R}^n$ be a parametrized curve. The tangent vector to γ at the point $\gamma(t)$ is defined as

$$\tau := \dot\gamma(t)\,.$$

Example 3.7: Tangent vector to helix

The helix is described by the parametric curve

$$\gamma : \mathbb{R} \to \mathbb{R}^3$$

with

$$\gamma_1(t) = \cos(t)\,,\ \ \gamma_2(t) = \sin(t)\,,\ \ \gamma_3(t) = t.$$

This is plotted in Figure 3.4 below. The tangent vector at point $\gamma(t)$ is given by

$$\dot\gamma(t) = (-\sin(t), \cos(t), 1)\,.$$

For example in Figure 3.4 we plot the tangent vector at time $t = \pi/2$, that is,

$$\dot\gamma(\pi/2) = (-1, 0, \pi/2)\,.$$

The above looks very similar to the tangent vector to the circle. Except that there is a z component, and that component is constant and equal to 1. Intuitively this means that the helix is *lifting* from the plane xy with constant speed with respect to the z-axis. We will soon give a name to this concept.

Remark 3.8: Avoiding potential ambiguities

Sometimes it will happen that a curve self intersects, meaning that there are two time instants t_1 and t_2 and a point $p \in \mathbb{R}^n$ such that

$$p = \gamma(t_1) = \gamma(t_2)\,.$$

In this case there is ambiguity in talking about the tangent vector at the point p: in principle there are two tangent vectors $\dot\gamma(t_1)$ and $\dot\gamma(t_2)$, and it could happen that

$$\dot\gamma(t_1) \neq \dot\gamma(t_1)\,.$$

Thus the concept of tangent at p is not well-defined. We need then to be more precise and talk about tangent at a certain **time-step** t, rather than at some **point** p. We however do not amend Definition 1.6, but you should keep this potential ambiguity in mind.

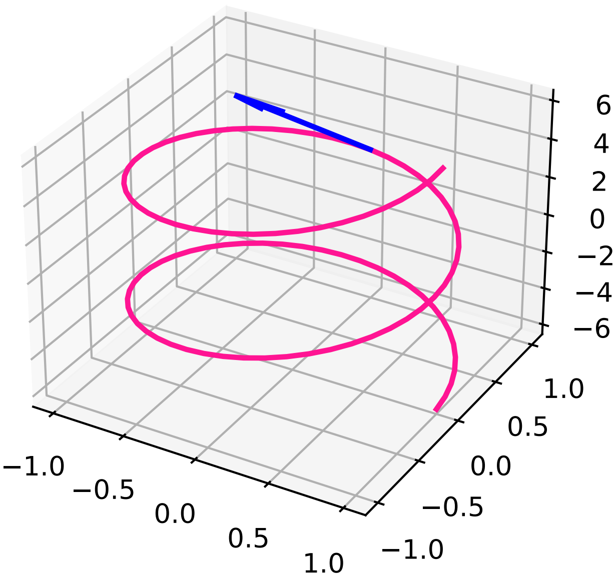


Figure 3.4: Plot of Helix with tangent vector

Example 3.9: The Lemniscate, a self intersecting curve

For example consider $\gamma : [0, 2\pi] \to \mathbb{R}^2$ defined as

$$\gamma_1(t) = \sin(t)\,, \;\; \gamma_2(t) = \sin(t)\cos(t)\,.$$

Such curve is called **Lemniscate**, see Wikipedia page, and is plotted in Figure 3.5 below. The orgin $(0,0)$ is a point of self-intersection, meaning that

$$\gamma(0) = \gamma(\pi) = (0,0)\,.$$

The tangent vector at point $\gamma(t)$ is given by

$$\dot{\gamma}(t) = (\cos(t), \cos^2(t) - \sin^2(t))$$

and therefore we have two tangents at $(0,0)$, that is,

$$\tau_1 = \dot{\gamma}(0) = (1,1)\,, \;\; \tau_2 = \dot{\gamma}(\pi) = (-1,1)\,.$$

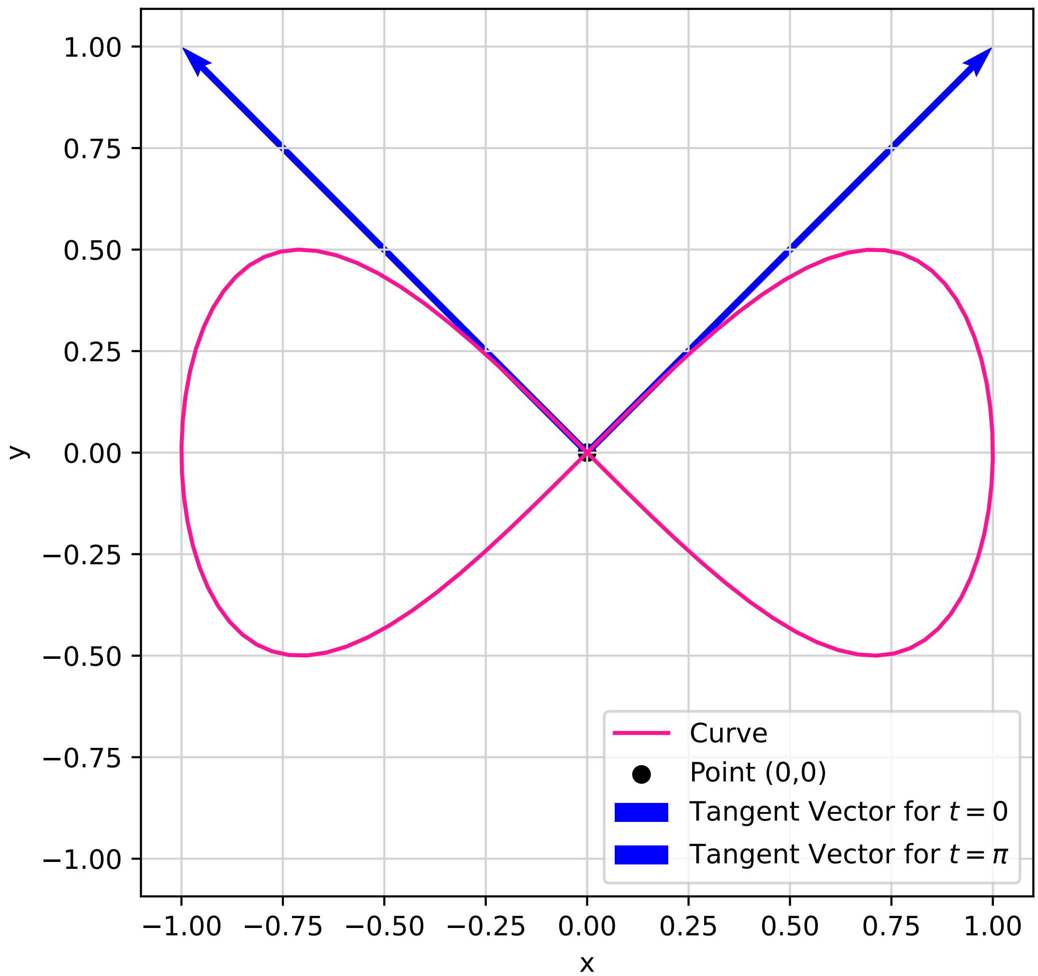


Figure 3.5: The Lemniscate curve

3.2 Length of curves

For a vector $v \in \mathbb{R}^n$ with components

$$v = (v_1, \dots, v_n),$$

its **length** is defined by

$$\|v\| := \sqrt{\sum_{i=1}^{n} v_i^2}\,.$$

The above is just an extension of the Pythagoras theorem to $\mathbb{R}^n$, and the length of v is computed from the origin.

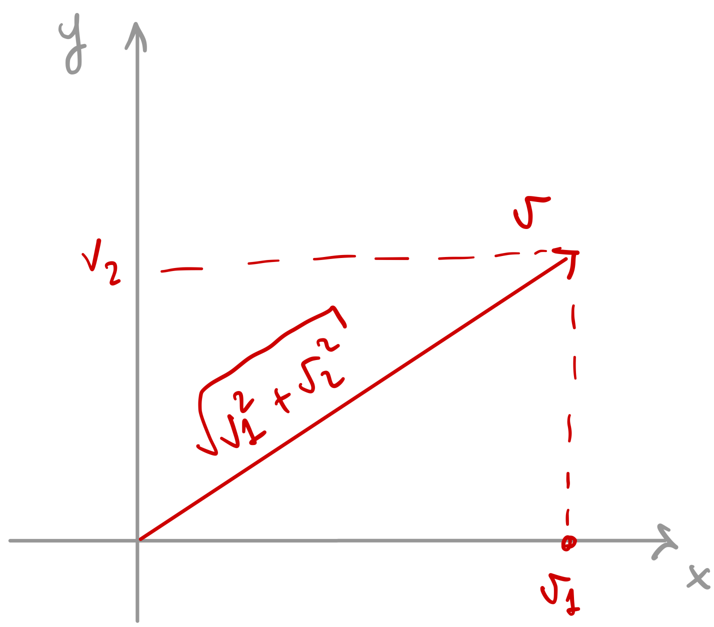


Figure 3.6: Interpretation of $\|v\|$ in $\mathbb{R}^2$

If we have a second vector $u \in \mathbb{R}^n$, then the quantity

$$\|u - v\| := \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2}$$

measures the length of the difference between u and v.

We would like to define the concept of **length** of a curve. Intuitively, one could proceed by approximation as in the figure below.

In formulae, this means choosing some time instants

$$t_0, \dots, t_m \in (a, b)\,.$$

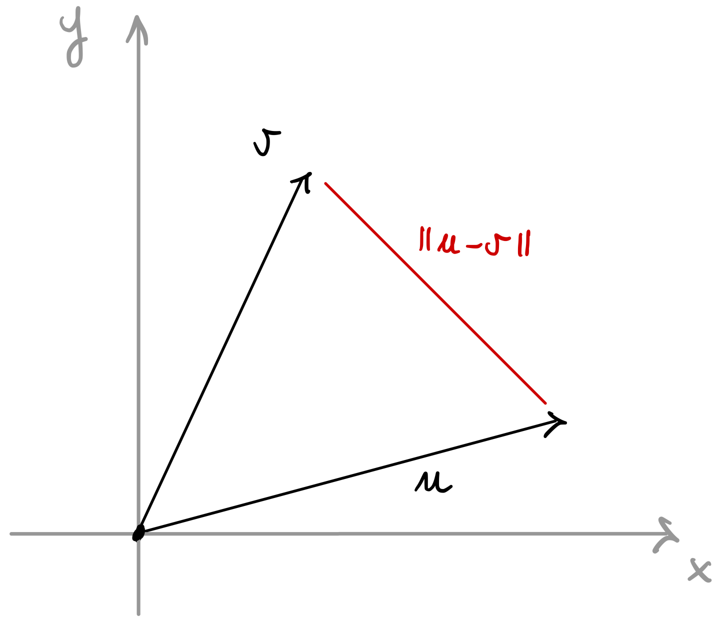


Figure 3.7: Interpretation of $\|u - v\|$ in $\mathbb{R}^2$

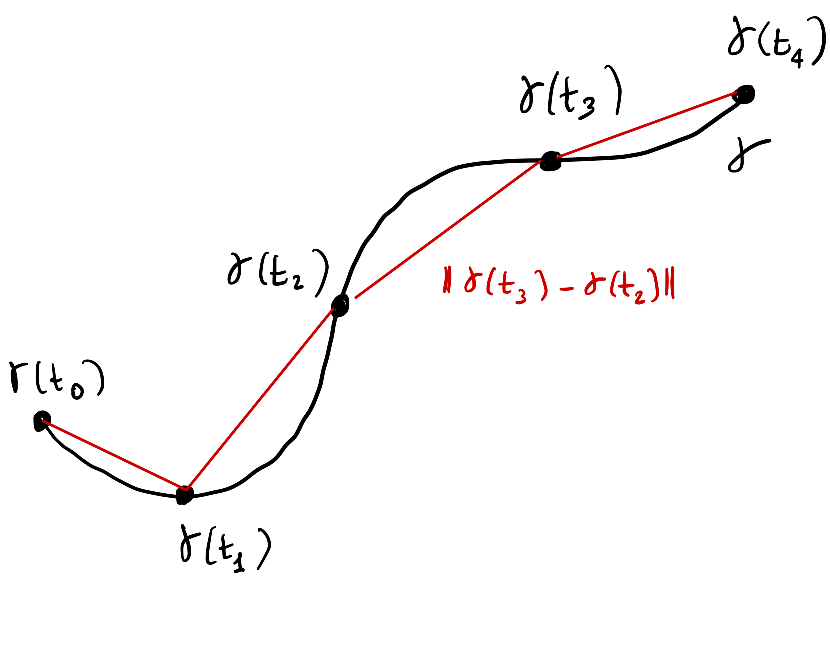


Figure 3.8: Approximating the length of γ

The length of the segment connecting $\gamma(t_{i-1})$ to $\gamma(t_i)$ is given by

$$\|\gamma(t_i) - \gamma(t_{i-1})\| \,.$$

Thus

$$L(\gamma) \approx \sum_{i=1}^{m} \|\gamma(t_i) - \gamma(t_{i-1})\| \,. \tag{3.4}$$

Intuitively, if we increase the number of points t_i, the quantity on the RHS of (3.4) should approximate $L(\gamma)$ better and better. Let us make this precise.

Definition 3.10: Partition

Let (a, b) be an interval. A partition $\mathcal{P}$ of $[a, b]$ is a vector of time instants

$$\mathcal{P} = (t_0, \dots, t_k) \in [a, b]^{m+1}$$

with

$$t_0 = a < t_1 < \dots < t_{m-1} < t_m = b\,.$$

If $\mathcal{P}$ is a partition of $[a, b]$, we define its maximum length as

$$\|\mathcal{P}\| := \max_{1 \le i \le m} |t_i - t_{i-1}|\,.$$

Note that $\|\mathcal{P}\|$ measures how fine the partition $\mathcal{P}$ is.

Definition 3.11: Length of approximating polygonal curve

Suppose $\gamma \colon (a, b) \to \mathbb{R}^n$ is a parametrized curve and $\mathcal{P}$ a partition of $[a, b]$. We define the length of the polygonal curve connecting the points

$$\gamma(t_0)\,, \;\; \gamma(t_1)\,, \;\; \dots\,, \;\; \gamma(t_m)$$

as

$$L(\gamma, \mathcal{P}) := \sum_{i=1}^{m} \|\gamma(t_i) - \gamma(t_{i-1})\| \,.$$

If $\|\mathcal{P}\|$ becomes smaller and smaller, that is, the partition $\mathcal{P}$ is finer and finer, it is reasonable to say that

$$L(\gamma, \mathcal{P})$$

is approximating the length of γ. We take this as definition of length.

Definition 3.12: Rectifiable curve and length

Suppose $\gamma \colon (a,b) \to \mathbb{R}^n$ is a parametrized curve. We say that γ is **rectifiable** if the limit

$$L(\gamma) = \lim_{\|P\| \to 0} L(\gamma, \mathcal{P})$$

exists finite. In such case we call $L(\gamma)$ the **length** of γ.

This definition definitely corresponds to our geometrical intuition of length of a curve.

Question 3.13

How do we use such definition in practice to compute the length of a given curve γ?

Thankfully, when γ is smooth, the length $L(\gamma)$ can be characterized in terms of $\dot\gamma$. Indeed, when δ is small, then the quantity

$$\|\gamma(t+\delta) - \gamma(t)\|$$

is approximating the length of γ between $\gamma(t)$ and $\gamma(t+\delta)$. Multiplying and dividing by δ we obtain

$$\frac{\|\gamma(t+\delta) - \gamma(t)\|}{\delta} \, \delta$$

which for small δ is close to

$$\|\dot\gamma(t)\| \, \delta \, .$$

We can now divide the time interval (a,b) in steps $t_0, \dots, t_m$ with $|t_i - t_{i-1}| < \delta$ and obtain

$$\begin{aligned} \|\gamma(t_i) - \gamma(t_{i-1})\| &= \frac{\|\gamma(t_i) - \gamma(t_{i-1})\|}{|t_i - t_{i-1}|} |t_i - t_{i-1}| && (3.5) \\ &\approx \|\dot\gamma(t_i)\| \, \delta && (3.6) \end{aligned}$$

since δ is small. Therefore

$$L(\gamma) \approx \sum_{i=1}^m \|\gamma(t_i) - \gamma(t_{i-1})\| \approx \sum_{i=1}^m \|\dot\gamma(t_i)\| \, \delta \, .$$

The RHS is a Riemann sum, therefore

$$L(\gamma) \approx \int_a^b \|\dot\gamma(t)\| \, dt \, .$$

The above argument can be made rigorous, as we see in the next theorem.

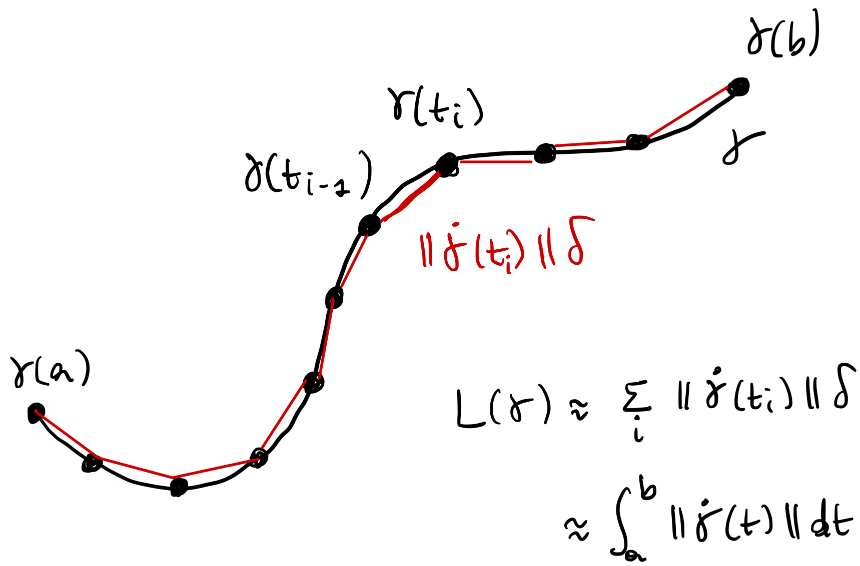


Figure 3.9: Approximating $L(\gamma)$ via $\dot\gamma$

Theorem 3.14: Characterizing the length of γ

Assume $\gamma : [a, b] \to \mathbb{R}^n$ is a parametrized curve, with $[a, b]$ bounded. Then γ is rectifiable and

$$L(\gamma) = \int_a^b \|\dot\gamma(t)\| \, dt \,. \tag{3.7}$$

Proof

Step 1. The integral in (3.7) is bounded.
Since γ is smooth, in particular $\dot\gamma$ is continuous. Since $[a, b]$ is bounded, then $\dot\gamma$ is bounded, that is

$$\sup_{t\in[a,b]} \|\dot\gamma(t)\| \le C$$

for some constant $C \ge 0$. Therefore

$$\int_a^b \|\dot\gamma(t)\| \, dt \le C(b - a) < \infty \,.$$

Step 2. Writing (3.7) as limit.
Recalling that

$$L(\gamma) = \lim_{\|\mathcal{P}\|\to 0} L(\gamma, \mathcal{P}) \,,$$

whenever the limit is finite, in order to show (3.7) we then need to prove

$$L(\gamma, \mathcal{P}) \to \int_a^b \|\dot{\gamma}(t)\| \, dt$$

as $\|\mathcal{P}\| \to 0$. Showing the above means proving that: for every $\varepsilon > 0$ there exists a $\delta > 0$ such that, if $\mathcal{P}$ is a partition of $[a, b]$ such that $\|\mathcal{P}\| < \delta$, then

$$\left| \int_a^b \|\dot{\gamma}(t)\| \, dt - L(\gamma, \mathcal{P}) \right| < \varepsilon \, . \tag{3.8}$$

Step 3. First estimate in (3.8).
This first estimate is easy, and only relies on the Fundamental Theorem of Calculus. To be more precise, we will show that each polygonal has shorter length than $\int_a^b \|\dot{\gamma}(t)\| \, dt$. To this end, take an arbitrary partition $\mathcal{P} = (t_0, \dots, t_m)$ of $[a, b]$. Then for each $i = 1, \dots, m$ we have

$$\|\gamma(t_i) - \gamma(t_{i-1})\| = \left\| \int_{t_{i-1}}^{t_i} \dot{\gamma}(t) \, dt \right\| \leq \int_{t_{i-1}}^{t_i} \|\dot{\gamma}(t)\| \, dt$$

where we used the Fundamental Theorem of calculus, and usual integral properties. Therefore by definition

$$L(\gamma, \mathcal{P}) = \sum_{i=1}^{m} \|\gamma(t_i) - \gamma(t_{i-1})\| \tag{3.9}$$

$$\leq \sum_{i=1}^{m} \int_{t_{i-1}}^{t_i} \|\dot{\gamma}(t)\| \, dt \tag{3.10}$$

$$= \int_a^b \|\dot{\gamma}(t)\| \, dt \, . \tag{3.11}$$

We have then shown

$$L(\gamma, \mathcal{P}) \leq \int_a^b \|\dot{\gamma}(t)\| \, dt \tag{3.12}$$

for all partitions $\mathcal{P}$.
Step 4. Second estimate in (3.8).
The second estimate is more delicate. We need to carefully construct a polygonal so that its length is close to $\int_a^b \|\dot{\gamma}\| \, dt$. This will be possible by uniform continuity of $\dot{\gamma}$. Indeed, note that $\dot{\gamma}$ is continuous on the compact set $[a, b]$. Therefore it is uniformly continuous by the Heine-Borel Theorem. Fix $\varepsilon > 0$. By uniform continuity of $\dot{\gamma}$ there exists $\delta > 0$ such that

$$|t - s| < \delta \implies \|\dot{\gamma}(t) - \dot{\gamma}(s)\| < \frac{\varepsilon}{b - a} \, . \tag{3.13}$$

for all $t, s \in [a, b]$. Let $\mathcal{P} = (t_0, \dots, t_m)$ be a partition of $[a, b]$ with $\|\mathcal{P}\| < \delta$. Recall that

$$\|\mathcal{P}\| = \max_{i=1,\dots,m} |t_i - t_{i-1}| \,.$$

Therefore the condition $\|\mathcal{P}\| < \delta$ implies

$$|t_i - t_{i-1}| < \delta \tag{3.14}$$

for each $i = 1, \dots, m$. For all $i = 1, \dots, m$ and $s \in [t_{i-1}, t_i]$ we have

$$\begin{align}
\gamma(t_i) - \gamma(t_{i-1}) &= \int_{t_{i-1}}^{t_i} \dot\gamma(t)\, dt \tag{3.15}\\
&= \int_{t_{i-1}}^{t_i} \dot\gamma(s) + (\dot\gamma(t) - \dot\gamma(s))\, dt \tag{3.16}\\
&= (t_i - t_{i-1})\dot\gamma(s) + \int_{t_{i-1}}^{t_i} (\dot\gamma(t) - \dot\gamma(s))\, dt \tag{3.17}
\end{align}$$

Therefore

$$\|\gamma(t_i) - \gamma(t_{i-1})\| = \left\| (t_i - t_{i-1})\dot\gamma(s) + \int_{t_{i-1}}^{t_i} (\dot\gamma(t) - \dot\gamma(s))\, dt \right\| \tag{3.18}$$

We can now use the reverse triangle inequality

$$|\|x\| - \|y\|| \le \|x - y\| \,,$$

for all $x, y \in \mathbb{R}^n$, which implies

$$\|x + y\| = \|x - (-y)\| \ge \|x\| - \|y\|$$

for all $x, y \in \mathbb{R}^n$. Applying the above to (3.18) we get

$$\|\gamma(t_i) - \gamma(t_{i-1})\| \ge (t_i - t_{i-1}) \|\dot\gamma(s)\| - \left\| \int_{t_{i-1}}^{t_i} (\dot\gamma(t) - \dot\gamma(s))\, dt \right\| \tag{3.19}$$

By standard properties of integral we also have

$$\left\| \int_{t_{i-1}}^{t_i} (\dot\gamma(t) - \dot\gamma(s))\, dt \right\| \le \int_{t_{i-1}}^{t_i} \|\dot\gamma(t) - \dot\gamma(s)\| \, dt \,,$$

so that (3.19) implies

$$\|\gamma(t_i) - \gamma(t_{i-1})\| \ge (t_i - t_{i-1}) \|\dot\gamma(s)\| - \int_{t_{i-1}}^{t_i} \|\dot\gamma(t) - \dot\gamma(s)\| \, dt \,. \tag{3.20}$$

Since $t, s \in [t_{i-1}, t_i]$, then

$$|t - s| \leq |t_i - t_{i-1}| < \delta$$

where the last inequality follows by (3.14). Thus by uniform continuity (3.13) we get

$$\|\dot{\gamma}(t) - \dot{\gamma}(s)\| < \frac{\varepsilon}{b-a} .$$

We can therefore further estimate (3.20) and obtain

$$\|\gamma(t_i) - \gamma(t_{i-1})\| \geq (t_i - t_{i-1})\, \|\dot{\gamma}(s)\| - \int_{t_{i-1}}^{t_i} \|\dot{\gamma}(t) - \dot{\gamma}(s)\| \, dt \tag{3.21}$$

$$\geq (t_i - t_{i-1})\, \|\dot{\gamma}(s)\| - (t_i - t_{i-1}) \frac{\varepsilon}{b-a} \, dt . \tag{3.22}$$

Dividing the above by $t_i - t_{i-1}$ we get

$$\frac{\|\gamma(t_i) - \gamma(t_{i-1})\|}{t_i - t_{i-1}} \geq \|\dot{\gamma}(s)\| - \frac{\varepsilon}{b-a} .$$

Integrating the above over s in the interval $[t_{i-1}, t_i]$ we get

$$\|\gamma(t_i) - \gamma(t_{i-1})\| \geq \int_{t_{i-1}}^{t_i} \|\dot{\gamma}(s)\| \, ds - \frac{\varepsilon}{b-a} (t_i - t_{i-1}) .$$

Summing over $i = 1, \dots, m$ we get

$$L(\mathcal{P}, \gamma) \geq \int_a^b \|\dot{\gamma}(s)\| \, ds - \varepsilon \tag{3.23}$$

since

$$\sum_{i=1}^m (t_i - t_{i-1}) = t_m - t_0 = b - a .$$

Conclusion.
Putting together (3.12) and (3.23) we get

$$\int_a^b \|\dot{\gamma}(s)\| \, ds - \varepsilon \leq L(\mathcal{P}, \gamma) \leq \int_a^b \|\dot{\gamma}(s)\| \, ds$$

which implies (3.8), concluding the proof.

Thanks to the above theorem we have now a way to compute $L(\gamma)$. Let us check that we have given a meaningful definition of length by computing $L(\gamma)$ on known examples.

Example 3.15: Length of Circle

The circle of radius R is parametrized by $\gamma : [0, 2\pi] \to \mathbb{R}^2$ defined by

$$\gamma(t) = (R\cos(t), R\sin(t)) .$$

Then

$$\dot\gamma(t) = (-R\sin(t), R\cos(t))$$

and

$$\begin{aligned} \|\dot\gamma(t)\| &= \sqrt{\dot\gamma_1^2(t) + \dot\gamma_2^2(t)} && (3.24) \\ &= R\sqrt{\sin^2(t) + \cos^2(t)} = R . && (3.25) \end{aligned}$$

Therefore

$$L(\gamma) = \int_0^{2\pi} \|\dot\gamma(t)\| \, dt = \int_0^{2\pi} R \, dt = 2\pi R$$

as expected.

Example 3.16: Length of helix

Let us consider one full turn of the Helix of radius R and rise H. This is parametrized by

$$\gamma(t) = (R\cos(t), R\sin(t), Ht)$$

for $t \in [0, 2\pi]$. Then

$$\dot\gamma(t) = (-R\sin(t), R\cos(t), H) ,$$

and

$$\begin{aligned} \|\dot\gamma(t)\| &= \sqrt{\dot\gamma_1^2 + \dot\gamma_2^2 + \dot\gamma_3^2} && (3.26) \\ &= \sqrt{R^2\sin^2(t) + R^2\cos^2(t) + H^2} = \sqrt{R^2 + H^2} . && (3.27) \end{aligned}$$

Therefore

$$L(\gamma) = \int_0^{2\pi} \|\dot\gamma(t)\| \, dt = 2\pi\sqrt{R^2 + H^2} .$$

3.3 Arc-length

We have just shown in Theorem 1.14 that the length of a regular curve $\gamma : [a, b] \to \mathbb{R}^n$ with $[a, b]$ bounded is given by

$$L(\gamma) = \int_a^b \|\dot{\gamma}(t)\| \, dt \, .$$

Using this formula, we introduce the notion of length of a portion of γ.

Definition 3.17: Arc-length

Let $\gamma : (a, b) \to \mathbb{R}^n$ be a curve, with (a, b) possibly unbounded. We define the **arc-length** of γ starting at the point $\gamma(t_0)$ as the function $s \colon \mathbb{R} \to \mathbb{R}$ defined by

$$s(t) := \int_{t_0}^t \|\dot{\gamma}(\tau)\| \, d\tau \, .$$

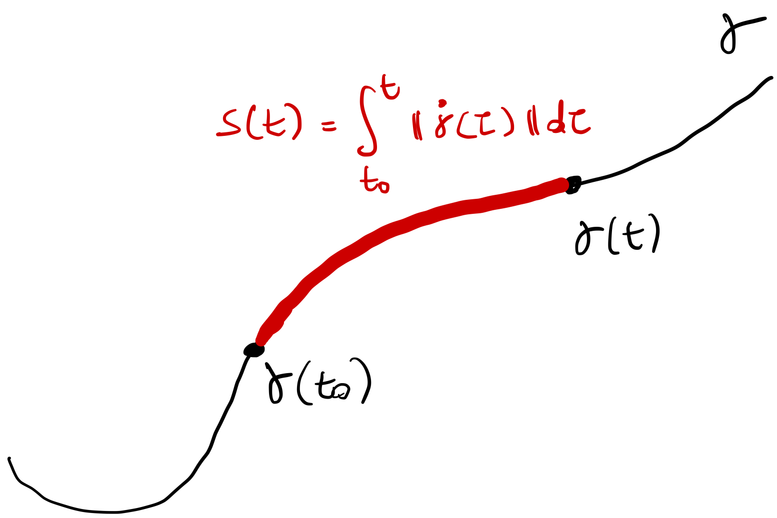


Figure 3.10: Arc-length of γ starting at $\gamma(t_0)$

Remark 3.18

A few remarks:

- Arc-length is well-defined

Indeed, γ is smooth, and so $\dot\gamma$ is continuous. WLOG assume $t \geq t_0$. Then

$$s(t) = \int_{t_0}^{t} \|\dot\gamma(\tau)\| \, d\tau \leq (t - t_0) \max_{\tau \in [t_0, t]} \|\dot\gamma(\tau)\| < \infty \,.$$

- We always have
$$s(t_0) = 0 \,.$$
- We have
$$t > t_0 \implies s(t) \geq 0$$
and
$$t < t_0 \implies s(t) \leq 0 \,.$$
- Choosing a different starting point changes the arc-length by a **constant**:

For example define $\tilde{s}$ as the arc-length starting from $\tilde{t}_0$

$$\tilde{s}(t) := \int_{\tilde{t}_0}^{t} \|\dot\gamma(\tau)\| \, d\tau \,.$$

Then by the properties of integral

$$s(t) = \int_{t_0}^{t} \|\dot\gamma(\tau)\| \, d\tau \tag{3.28}$$

$$= \int_{t_0}^{\tilde{t}_0} \|\dot\gamma(\tau)\| \, d\tau + \int_{\tilde{t}_0}^{t} \|\dot\gamma(\tau)\| \, d\tau \tag{3.29}$$

$$= \int_{t_0}^{\tilde{t}_0} \|\dot\gamma(\tau)\| \, d\tau + \tilde{s}(t) \,. \tag{3.30}$$

Hence

$$s = c + \tilde{s}$$

with

$$c := \int_{t_0}^{\tilde{t}_0} \|\dot\gamma(\tau)\| \, d\tau \,.$$

Note that c is the arc-length of γ between the starting points $\gamma(t_0)$ and $\gamma(\tilde{t}_0)$.

- The arc-length is a differentiable function, with

$$\dot{s}(t) = \frac{d}{dt} \int_{t_0}^{t} \|\dot\gamma(\tau)\| \, d\tau = \|\dot\gamma(t)\| \,.$$

Since $\dot\gamma$ is continuous, the above follows by the Fundamental Theorem of Calculus.

Example 3.19: Circle

The circle of radius R is parametrized by $\gamma : [0, 2\pi] \to \mathbb{R}^2$ defined by

$$\gamma(t) = (R\cos(t), R\sin(t)) .$$

Then

$$\dot\gamma(t) = (-R\sin(t), R\cos(t)) , \quad \|\dot\gamma(t)\| = R .$$

Therefore, for any fixed $t_0 \in [0, 2\pi]$ we have

$$s(t) = \int_{t_0}^{t} \|\dot\gamma(\tau)\| \, d\tau = \int_{t_0}^{t} R \, d\tau = (t - t_0)R .$$

In particular we see that $\dot s = R$ is constant.

Example 3.20: Logarithmic spiral

The Logarithmic spiral is defined by $\gamma : [0, 2\pi] \to \mathbb{R}^2$ with

$$\gamma(t) = (e^{kt}\cos(t), e^{kt}\sin(t)) ,$$

where $k \in \mathbb{R}$, $k \neq 0$, is called the **growth factor**. Then

$$\dot\gamma_1(t) = e^{kt}(k\cos(t) - \sin(t))$$

$$\dot\gamma_2(t) = e^{kt}(k\sin(t) + \cos(t))$$

and so, after some calculations,

$$\|\dot\gamma(t)\|^2 = \dot\gamma_1^2 + \dot\gamma_2^2 = (k^2 + 1)e^{2kt} .$$

The arc-length starting from t_0 is

$$\begin{aligned} s(t) &= \int_{t_0}^{t} \|\dot\gamma(\tau)\| \, d\tau && (3.31) \\ &= \sqrt{k^2+1} \int_{t_0}^{t} e^{k\tau} \, d\tau && (3.32) \\ &= \frac{\sqrt{k^2+1}}{k} (e^{kt} - e^{kt_0}) . && (3.33) \end{aligned}$$

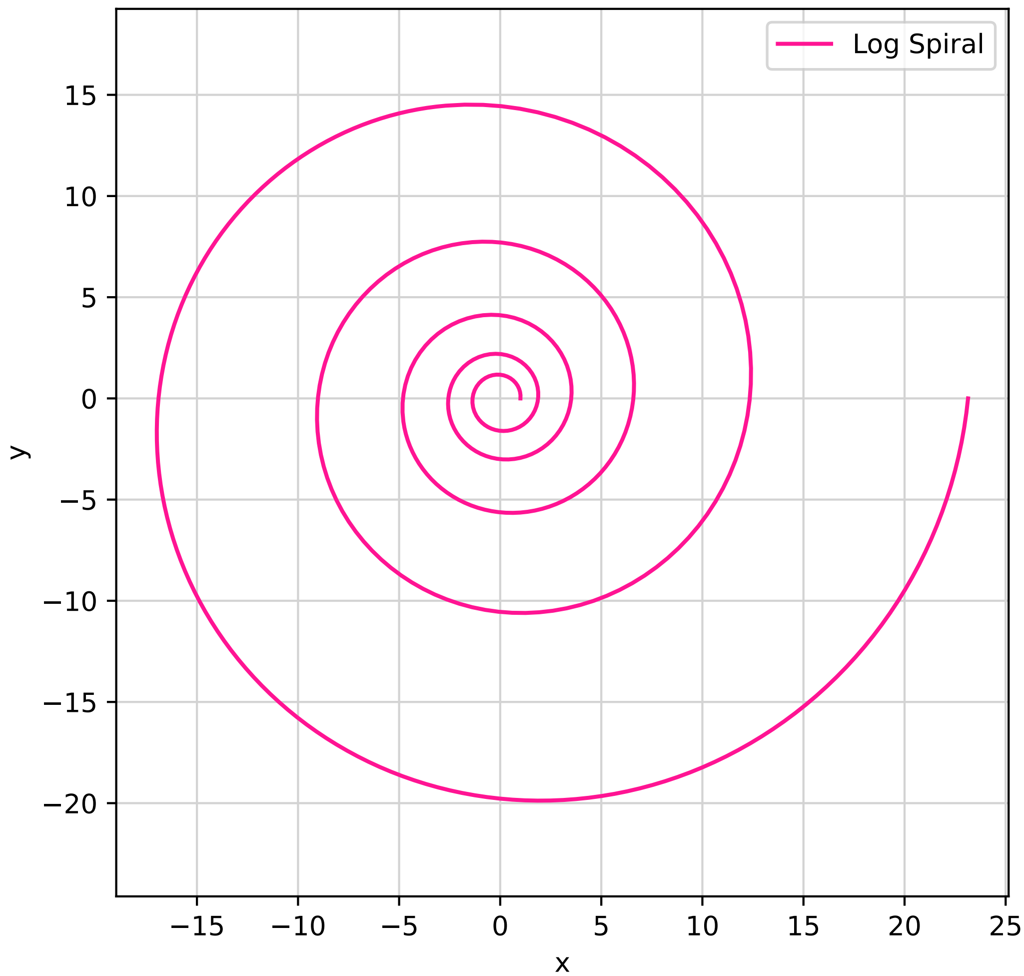


Figure 3.11: Plot of Logarithmic Spiral with $k = 0.1$

3.4 Scalar product in $\mathbb{R}^n$

Let us start by defining the scalar product in $\mathbb{R}^2$.

Definition 3.21: Scalar product in $\mathbb{R}^2$

Let $u, v \in \mathbb{R}^2$ and denote by $\theta \in [0, \pi]$ the angle formed by u and v. The *scalar product* between u and v is defined by

$$u \cdot v := |u||v| \cos(\theta) .$$

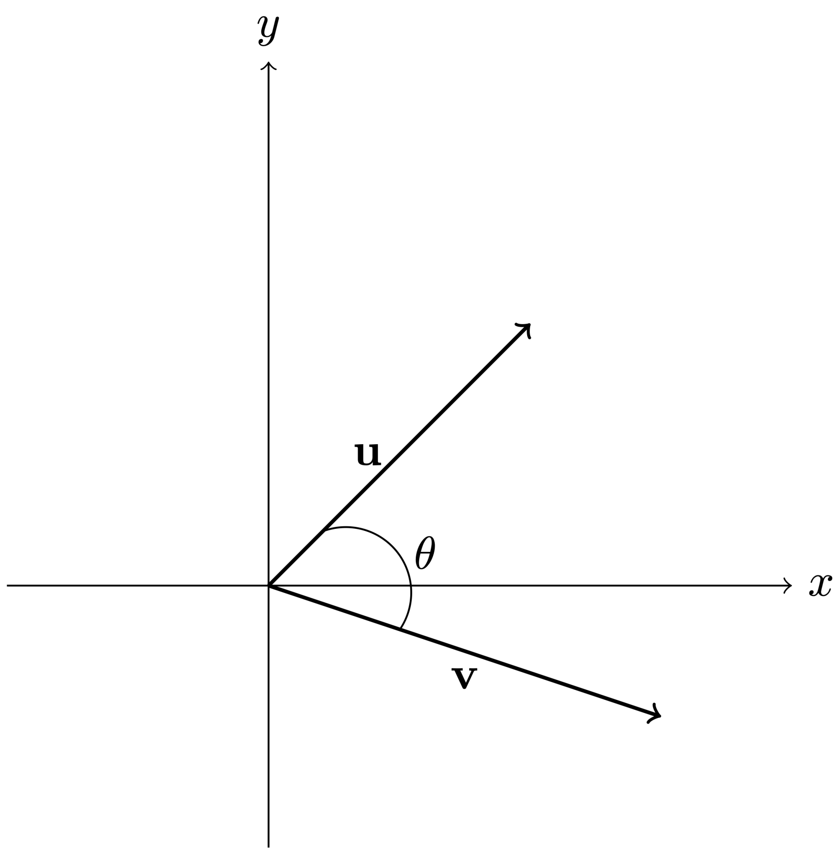


Figure 3.12: Vectors u and v in $\mathbb{R}^2$ forming angle θ

Remark 3.22

The scalar product is maximized for $\theta = 0$, for which we have

$$u \cdot v = |u||v| \cos(\theta) = |u||v| .$$

It is instead minimized for $\theta = \pi$, for which

$$u \cdot v = |u||v| \cos(\theta) = -|u||v| \,.$$

Definition 3.23: Orthogonal vectors

Let $u, v \in \mathbb{R}^2$. If

$$u \cdot v = 0$$

we say that u and v are **orthogonal**.

Proposition 3.24: Bilinearity and symmetry of scalar product

Let $u, v, w \in \mathbb{R}^2$ and $\lambda \in \mathbb{R}$. Then

- **Symmetry**: $u \cdot v = v \cdot u$
- **Bilinearity**: It holds

$$\lambda(u \cdot v) = (\lambda u) \cdot v = u \cdot (\lambda v) \,,$$

$$u \cdot (v + w) = u \cdot v + u \cdot w \,.$$

We leave the proof to the reader. The above proposition is saying that the scalar product is **bilinear** and **symmetric**.

Proposition 3.25: Scalar products written wrt euclidean coordinates

Denote by

$$e_1 = (1, 0) \,, \quad e_2 = (0, 1)$$

the euclidean basis of $\mathbb{R}^2$. Let $u, v \in \mathbb{R}^2$ and denote by

$$u = (u_1, u_2) = u_1 e_1 + u_2 e_2$$

$$v = (v_1, v_2) = v_1 e_1 + v_2 e_2$$

their coordinates with respect to e_1, e_2. Then

$$u \cdot v = u_1 v_2 + u_2 v_2 \,.$$

Proof

Note that

$$e_1 \cdot e_1 = 1\,, \quad e_2 \cdot e_2 = 1\,, \quad e_1 \cdot e_2 = e_2 \cdot e_1 = 0\,.$$

Using the bilinearity of scalar product we have

$$\begin{aligned} u \cdot v &= (u_1 e_1 + u_2 e_2) \cdot (v_1 e_1 + v_2 e_2) && (3.34)\\ &= u_1 v_1 e_1 \cdot e_1 + u_1 v_2 e_1 \cdot e_2 + u_2 v_1 e_2 \cdot e_1 + u_2 v_2 e_2 \cdot e_2 && (3.35)\\ &= u_1 v_1 + u_2 v_2\,. && (3.36) \end{aligned}$$

The above proposition provides a way to generalize of the scalar product to $\mathbb{R}^n$..

Definition 3.26: Scalar product in $\mathbb{R}^n$

Let $u, v \in \mathbb{R}^n$ and denote their coordinates by

$$u = (u_1, \dots, u_n)\,, \quad u = (v_1, \dots, v_n)\,.$$

We define the scalar product between u and v by

$$u \cdot v := \sum_{i=1}^{n} u_i v_i\,.$$

With the above definition we still have that the scalar product is bilinear and symmetric, as detailed in the following proposition:

Proposition 3.27: Bilinearity and symmetry of scalar product in $\mathbb{R}^n$

Let $u, v, w \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$. Then

- **Symmetry**: $u \cdot v = v \cdot u$
- **Bilinearity**: It holds

$$\lambda(u \cdot v) = (\lambda u) \cdot v = u \cdot (\lambda v)\,,$$
$$u \cdot (v + w) = u \cdot v + u \cdot w\,.$$

The proof of the above proposition is an easy check, and is left to the reader for exercise.

Definition 3.28

Let $u, v \in \mathbb{R}^n$. We say that u and v are **orthogonal** if

$$u \cdot v = 0\,.$$

Proposition 3.29: Differentiating scalar product

Let $\gamma, \eta : (a,b) \to \mathbb{R}^n$ be parametrized curves. Then the scalar map

$$\gamma \cdot \eta : (a,b) \to \mathbb{R}$$

is smooth, and

$$\frac{d}{dt}(\gamma \cdot \eta) = \dot{\gamma} \cdot \eta + \gamma \cdot \dot{\eta}$$

for all $t \in (a,b)$.

Proof

Denote by

$$\gamma = (\gamma_1, \dots, \gamma_n)\,, \quad \eta = (\eta_1, \dots, \eta_n)$$

the coordinates of γ and η. Clearly the map

$$t \mapsto \gamma \cdot \eta = \sum_{i=1}^{n} \gamma_i \eta_i$$

is smooth, being sum and product of smooth functions.
Concerning the formula, by definition of scalar product and linearity of the derivative we have

$$\begin{aligned}
\frac{d}{dt}(\gamma \cdot \eta) &= \frac{d}{dt}\left(\sum_{i=1}^{n} \gamma_i \eta_i\right) && (3.37)\\
&= \sum_{i=1}^{n} \frac{d}{dt}(\gamma_i \eta_i) && (3.38)\\
&= \sum_{i=1}^{n} \dot{\gamma}_i \eta_i + \gamma_i \dot{\eta}_i && (3.39)\\
&= \dot{\gamma} \cdot \eta + \gamma \cdot \dot{\eta}\,, && (3.40)
\end{aligned}$$

where in the second to last equality we used the product rule of differentiation.

3.5 Speed of a curve

Given a curve γ we defined the **tangent** vector at $\gamma(t)$ to be

$$\dot{\gamma}(t)\,.$$

The tangent vector measures the change of direction of the curve. Therefore the magnitude of $\dot{\gamma}$ can be interpreted as the **speed** of the curve.

Definition 3.30

Let $\gamma : (a,b) \to \mathbb{R}^n$ be a curve. We define the speed of γ at the point $\gamma(t)$ by

$$\|\dot{\gamma}(t)\| \,.$$

We say that γ is a **unit-speed** curve if

$$\|\dot{\gamma}(t)\| = 1 \,, \quad \forall\, t \in (a,b) \,.$$

Remark 3.31

The derivative of the arc-length s gives the speed of γ:

$$s(t) := \int_{t_0}^{t} \|\dot{\gamma}(\tau)\| \, d\tau \quad \Longrightarrow \quad \dot{s}(t) = \|\dot{\gamma}(t)\| \,.$$

The reason why we introduce unit speed curves is because they make calculations easy. This is essentially because of the next proposition.

Proposition 3.32

Let $\gamma : (a,b) \to \mathbb{R}^n$ be a unit speed curve. Then

$$\dot{\gamma} \cdot \ddot{\gamma} = 0$$

for all $t \in (a,b)$.

Proof

Let us consider the identity

$$\dot{\gamma}(t) \cdot \dot{\gamma}(t) = \sum_{i=1}^{n} \dot{\gamma}_i^2(t) = \|\dot{\gamma}(t)\|^2 \,. \tag{3.41}$$

Since γ is unit speed we have

$$\|\dot{\gamma}(t)\|^2 = 1 \quad \forall\, t \in (a,b) \,.$$

and therefore

$$\frac{d}{dt}\left(\|\dot{\gamma}(t)\|^2\right) = 0 \quad \forall\, t \in (a,b)\,. \tag{3.42}$$

We can differentiate the LHS of (3.41) to get

$$\frac{d}{dt}(\dot{\gamma}\cdot\dot{\gamma}) = \ddot{\gamma}\cdot\dot{\gamma} + \dot{\gamma}\cdot\ddot{\gamma} = 2\dot{\gamma}\cdot\ddot{\gamma}\,. \tag{3.43}$$

where we used Proposition 1.27 and symmetry of the scalar product. Differentiating (3.41) and using (3.42)-(3.43) we conclude

$$2\dot{\gamma}\cdot\ddot{\gamma} = 0 \quad \forall\, t \in (a,b)\,.$$

Remark 3.33

Proposition 3.32 is saying that if γ is unit speed, then its tangent vector $\dot{\gamma}$ is always orthogonal to the second derivative $\ddot{\gamma}$. This will be very useful in the future.

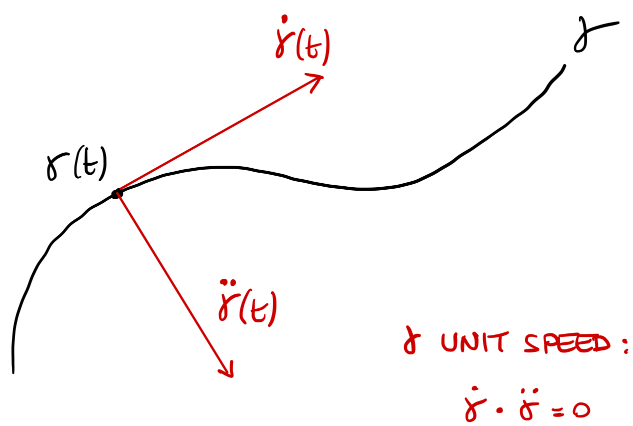


Figure 3.13: If γ is unit speed then $\dot{\gamma}$ and $\ddot{\gamma}$ are orthogonal

3.6 Reparametrization

As we have observed in the Examples of Chapter 1, there is in general no unique way to parametrize a curve. However we would like to understand when two parametrizations are related. In other words, we want to clarify the concept of **equivalence** of two parametrizations.

Definition 3.34: Diffeomorphism

Let $\phi : (a,b) \to (\tilde{a}, \tilde{b})$. We say that ϕ is a **diffeomorphism** if the following conditions are satisfied:

1. ϕ is invertible, with inverse $\phi^{-1} : (\tilde{a}, \tilde{b}) \to (a,b)$. Thus

$$\phi^{-1} \circ \phi = \phi \circ \phi^{-1} = \mathrm{Id}\,,$$

   where $\mathrm{Id} \colon \mathbb{R} \to \mathbb{R}$ is the identity map on $\mathbb{R}$, that is,

$$\mathrm{Id}(t) = t\,, \quad \forall\, t \in \mathbb{R}\,.$$

2. ϕ is smooth,
3. ϕ^{-1} is smooth.

Definition 3.35: Reparametrization

Let $\gamma : (a,b) \to \mathbb{R}^n$ be a parametrized curve. A **reparametrization** of γ is another parametrized curve $\tilde{\gamma} : (\tilde{a}, \tilde{b}) \to \mathbb{R}^n$ such that

$$\tilde{\gamma}(t) = \gamma(\phi(t)) \quad \forall\, t \in (\tilde{a}, \tilde{b})\,, \tag{3.44}$$

where

$$\phi \colon (\tilde{a}, \tilde{b}) \to (a,b)$$

is a diffeomerphism. We call both ϕ and ϕ^{-1} **reparametrization maps**.

Remark 3.36

A comment about the above definition. Given a parametrized curve γ, this identifies a 1D shape $\Gamma \subset \mathbb{R}^n$. A reparametrization $\tilde{\gamma}$ is just an equivalent way to describe Γ. For γ and $\tilde{\gamma}$ to be reparametrizations of each other, there must exist a smooth rule ϕ to switch from one to another, according to formula (3.44)

Example 3.37: Change of orientation

The map $\phi \colon (\tilde{a}, \tilde{b}) \to (a,b)$ defined by

$$\phi(t) := -t$$

is a diffeomoprhism. The inverse of ϕ is given by $\phi^{-1} \colon (a,b) \to (\tilde{a}, \tilde{b})$ defined by

$$\phi^{-1}(t) = -t\,.$$

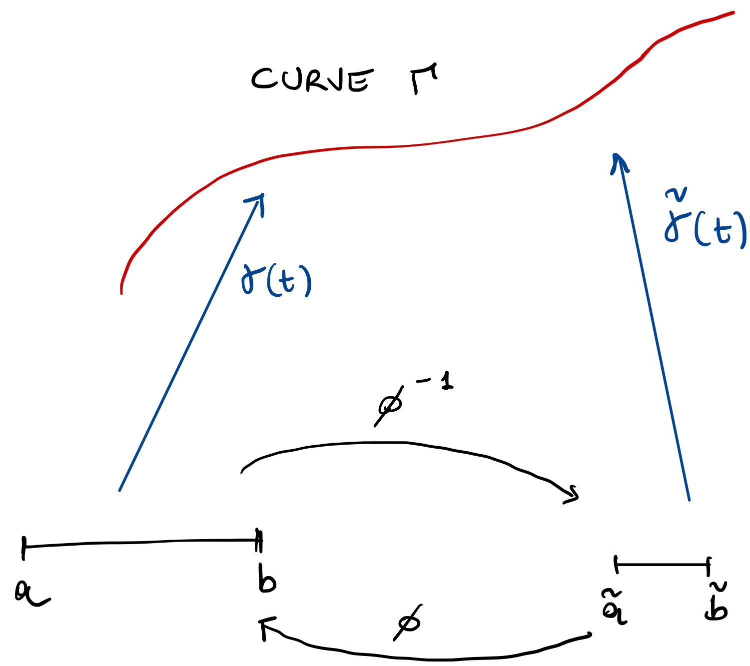


Figure 3.14: Sketch of 1D shaper parametrized by γ and $\tilde{\gamma}$

Note that ϕ can be used to **reverse the orientation** of a curve.

Example 3.38: Reversing orientation of circle

Consider the unit circle parametrized as usual by $\gamma : [0, 2\pi] \to \mathbb{R}^2$ defined as

$$\gamma(t) := (\cos(t), \sin(t)) .$$

To reverse the orientation we can reparametrize γ by using the diffeomorphism

$$\phi(t) := -t .$$

This way we obtain $\tilde{\gamma} := \gamma \circ \phi : [0, 2\pi] \to [0, 2\pi]$,

$$\tilde{\gamma}(t) = \gamma(\phi(t)) \tag{3.45}$$
$$= (\cos(-t), \sin(-t)) \tag{3.46}$$
$$= (\cos(t), -\sin(t)) , \tag{3.47}$$

where in the last identity we used the properties of cos and sin. Notice that in this way, for example,

$$\gamma(\pi/2) = (0, 1) , \quad \gamma(\pi/2) = (0, -1) .$$

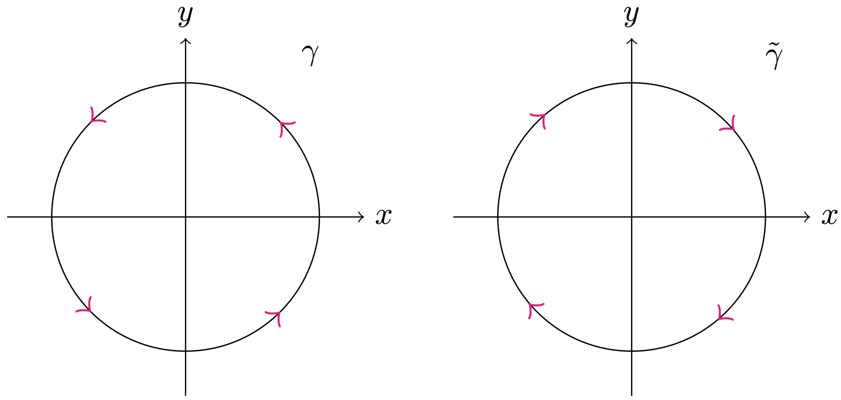


Figure 3.15: Unit circle with usual parametrization γ, and with reversed orientation $\tilde{\gamma}$

Example 3.39: Change of speed

Let $k > 0$. The map $\phi \colon (\tilde{a}, \tilde{b}) \to (a,b)$ defined by

$$\phi(t) := kt$$

is a diffeomoprhism. The inverse of ϕ is given by $\phi^{-1} \colon (a,b) \to (\tilde{a}, \tilde{b})$ defined by

$$\phi^{-1}(t) = \frac{t}{k}\,.$$

Note that ϕ can be used to **change the speed** of a curve:

- If $k > 1$ the speed increases ,
- If $0 < k < 1$ the speed decreases.

Example 3.40: Doubling the speed of Lemniscate

Recall the Lemniscate

$$\gamma(t) := (\sin(t), \sin(t)\cos(t))\,, \quad t \in [0, 2\pi]\,.$$

We can double the speed of the Lemniscate by using the Using the diffeomorphism

$$\phi(t) := 2t\,.$$

This way we obtain $\tilde{\gamma} := \gamma \circ \phi \, : [0,\pi] \to [0, 2\pi]$ with

$$\tilde{\gamma}(t) = \gamma(\phi(t)) = (\sin(2t), \sin(2t)\cos(2t))\,.$$

In this case we have that

$$\dot{\tilde{\gamma}}(t) = 2\dot{\gamma}(\phi(t))\,.$$

The above follows by chain rule. Indeed, $\dot{\phi} = 2$, so that

$$\dot{\tilde{\gamma}} = \frac{d}{dt}\left(\gamma(\phi(t))\right) = \dot{\phi}(t)\dot{\gamma}(\phi(t)) = 2\dot{\gamma}(\phi(t))\,.$$

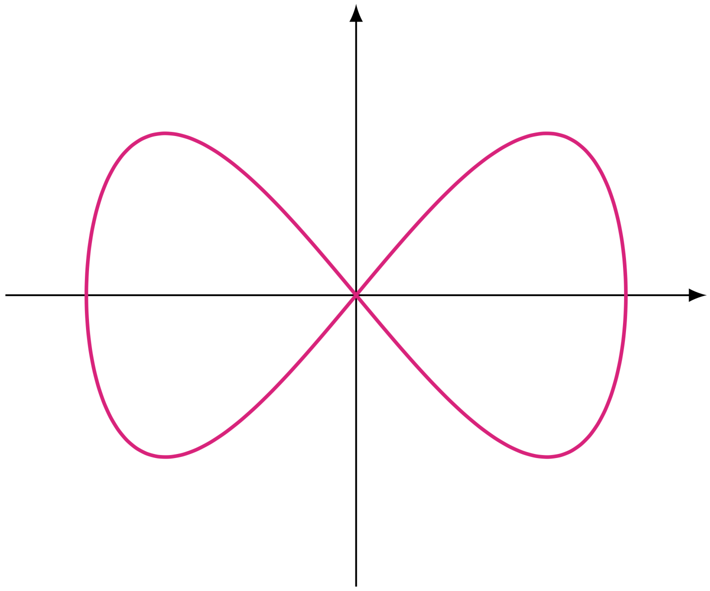


Figure 3.16: Lemniscate curve

Important

The main reason we are interested in reparametrizations is because we want to parametrize curves by **arc-lenght**: This means that, for a curve γ, we want to find a reparametrization $\tilde{\gamma}$ such that $\tilde{\gamma}$ is unit speed:

$$\left\|\dot{\tilde{\gamma}}\right\| = 1 \,, \quad \forall t \in (a,b) \,.$$

We will see that this is not always possible.

Definition 3.41: Regular points

Let $\gamma : (a,b) \to \mathbb{R}^n$ be a parametrized curve. We say that:

- $\gamma(t_0)$ is a **regular point** if
$$\dot{\gamma}(t_0) \neq 0 \,.$$
- A point $\gamma(t_0)$ is **singular** if it is not regular.
- The curve γ is regular if every point of γ is regular, that is,
$$\dot{\gamma}(t) \neq 0 \,, \quad \forall \, t \in (a,b) \,.$$

Note that when $\dot{\gamma}(t_0) = 0$, this means the curve is *stopping* at time t_0. Before making an example, let us prove a useful lemma about diffeomorphisms.

Lemma 3.42

Let $\phi : (a,b) \to (\tilde{a}, \tilde{b})$ be a diffeomorphism. Then

$$\dot{\phi}(t) \neq 0 \quad \forall\, t \in (a,b)\,.$$

Proof

We know that ϕ is smooth with smooth inverse

$$\psi := \phi^{-1} \,:\, (\tilde{a}, \tilde{b}) \to (a,b)\,.$$

In particular it holds

$$\psi(\phi(t)) = t\,, \quad \forall\, t \in (a,b)\,.$$

We can differentiate both sides of the above expression to get

$$\frac{d}{dt}\left(\psi(\phi(t))\right) = 1\,. \tag{3.48}$$

We can differentiate the LHS by chain rule

$$\frac{d}{dt}\left(\psi(\phi(t))\right) = \dot{\psi}(\phi(t))\,\dot{\phi}(t)\,.$$

From (3.48) we then get

$$\dot{\psi}(\phi(t))\,\dot{\phi}(t) = 1\,, \quad \forall\, t \in (a,b)\,.$$

Since on the LHS we have a product, this means that none of the LHS terms vanishes, so that

$$\dot{\phi}(t) \neq 0\,, \quad \forall\, t \in (a,b)\,.$$

Example 3.43: A curve with one singular point

Consider the parabola

$$\Gamma := \{(x,y) \in \mathbb{R}^2 \,:\, y = x^2,\, -1 \leq x \leq 1\}\,.$$

This can be parametrized in two ways by $\gamma, \eta : [-1,1] \to \mathbb{R}^2$ defined as

$$\gamma(t) = (t, t^2)\,, \quad \eta(t) = (t^3, t^6)\,.$$

We will see that the above parametrizations are **not** equivalent. This is intuitively clear, since the change of variables map should be

$$\phi(t) = t^3\,.$$

This is smooth and invertible, with inverse

$$\phi^{-1}(t) = \sqrt[3]{x}\,.$$

However ϕ^{-1} is not smooth at $t = 0$, and thus ϕ is not a diffeomorphism. Alternatively we could have just noticed that

$$\dot{\phi}(t) = 3t^2 \quad \implies \quad \dot{\phi}(0) = 0 \,,$$

and therefore ϕ cannot be a diffeomorphism due to Lemma 3.42.
Let us look at the derivatives:

$$\dot{\gamma}(t) = (1, 2t) \,, \quad \dot{\eta}(t) = (3t^2, 6t^5) \,.$$

We notice a difference:

- γ is a regular parametrization,
- $\eta(t)$ is regular only for $t \neq 0$.

Indeed if we animate the plots of the above parametrizations, we see that:

- The point $\gamma(t)$ moves with constant horizontal speed
- The point $\eta(t)$ is decelerating for $t < 0$, it stops at $t = 0$, and then accelerates again for $t > 0$.

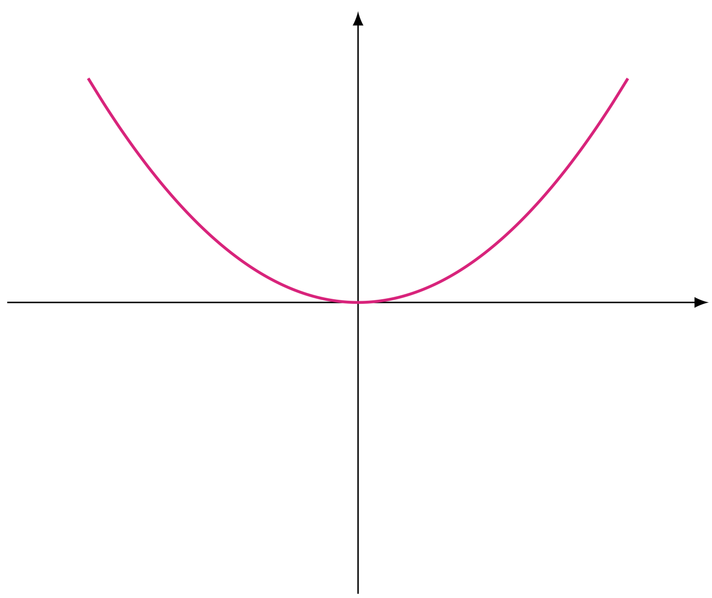


Figure 3.17: Parabola Γ

Proposition 3.44: Regularity is invariant for reparametrization

Let $\gamma : (a, b) \to \mathbb{R}^n$ be a parametrized curve and suppose that γ is regular, that is,

$$\dot{\gamma}(t) \neq 0 \,, \quad \forall \, t \in (a, b) \,.$$

Then every reparametrization of γ is also regular.

Proof

Let $\tilde{\gamma} : (\tilde{a}, \tilde{b}) \to \mathbb{R}^n$ be a reparametrization of γ. Then there exist $\phi : (\tilde{a}, \tilde{b}) \to (a, b)$ diffeomorphism such that

$$\tilde{\gamma}(t) = \gamma(\phi(t)) \,, \quad \forall\, t \in (\tilde{a}, \tilde{b}) \,.$$

By the chain rule we have

$$\dot{\tilde{\gamma}}(t) = \frac{d}{dt} (\gamma(\phi(t))) = \dot{\gamma}(\phi(t))\dot{\phi}(t) \,.$$

Therefore

$$\dot{\tilde{\gamma}}(t) \neq 0 \quad \iff \quad \dot{\gamma}(\phi(t))\dot{\phi}(t) \neq 0 \,. \tag{3.49}$$

But we are assuming that γ is regular, so that

$$\dot{\gamma}(\phi(t)) \neq 0 \,, \quad \forall\, t \in (\tilde{a}, \tilde{b}) \,.$$

Thus (3.49) is equivalent to

$$\dot{\tilde{\gamma}}(t) \neq 0 \quad \iff \quad \dot{\phi}(t) \neq 0 \,. \tag{3.50}$$

Since ϕ is a diffeomorphism, by Lemma 3.42 we have that

$$\dot{\phi}(t) \neq 0 \,, \quad \forall\, t \in (\tilde{a}, \tilde{b}) \,.$$

By (3.50) we conclude that

$$\dot{\tilde{\gamma}}(t) \neq 0 \,, \quad \forall\, t \in (\tilde{a}, \tilde{b}) \,,$$

proving that $\tilde{\gamma}$ is regular.

Example 3.45

Let us go back to the parabola

$$\Gamma := \{(x, y) \in \mathbb{R}^2 : \; y = x^2, \, -1 \leq x \leq 1\} \,,$$

with the two parametrizations $\gamma, \eta : [-1, 1] \to \mathbb{R}^2$ with

$$\gamma(t) = (t, t^2) \,, \quad \eta(t) = (t^3, t^6) \,.$$

We have that

$$\dot{\gamma}(t) = (1, 2t) \,, \quad \dot{\eta}(t) = (3t^2, 6t^5) \,.$$

Therefore

- γ is a regular parametrization,
- $\eta(t)$ is regular only for $t \neq 0$.

Proposition 3.44 implies that η is **NOT** a reparametrization of γ.

Definition 3.46: Unit speed reparametrization

Let γ be a parametrized curve. A **unit speed reparametrization** of γ is a reparametrization $\tilde{\gamma}$ such that $\tilde{\gamma}$ is unit speed.

The next theorem states that a curve is regular if and only if it has a unit speed reparametrization. For the proof, it is crucial to recall the definition of arc-length of a curve $\gamma \colon (a, b) \to \mathbb{R}^n$, which is given by

$$s(t) := \int_{t_0}^{t} \|\dot{\gamma}(\tau)\| \, d\tau \,,$$

for some arbitrary $t_0 \in (a, b)$ fixed. Indeed, we will see that for ϕ regular the unit speed parametrization map can be taken as

$$\phi = s^{-1} \,.$$

Theorem 3.47: Existence of unit speed reparametrization

Let γ be a parametrized curve. They are equivalent:

- γ is regular,
- γ has a unit speed reparametrization.

Proof

Step 1. Direct implication.
Assume $\gamma : (a, b) \to \mathbb{R}^n$ is regular, that is,

$$\dot{\gamma}(t) \neq 0 \,, \quad \forall \, t \in (a, b) \,.$$

Let $s : (a, b) \to \mathbb{R}$ be the arc-length of γ starting at any point $t_0 \in (a, b)$. By the Fundamental Theorem of Calculus we have

$$\dot{s}(t) = \|\dot{\gamma}(t)\| \tag{3.51}$$

so that

$$\dot{s}(t) > 0 \,, \quad \forall \, t \in (a, b) \,.$$

Since s is a scalar function, the above condition and the Inverse Function Theorem guarantee the existsence of a smooth inverse

$$s^{-1} : (\tilde{a}, \tilde{b}) \to (a, b)$$

for some $\tilde{\alpha} < \tilde{\beta}$. Define the reparametrization map ϕ as

$$\phi := s^{-1}$$

and the corresponding reparametrization of γ given by the curve

$$\tilde{\gamma} : (\tilde{a}, \tilde{b}) \to \mathbb{R}^n , \quad \tilde{\gamma} := \gamma \circ \phi \, .$$

We claim that $\tilde{\gamma}$ is unit speed. Indeed, by definition

$$\tilde{\gamma} := \gamma \circ \phi \qquad \Longrightarrow \qquad \gamma = \tilde{\gamma} \circ \phi^{-1} = \tilde{\gamma} \circ s \, ,$$

or in other words

$$\gamma(t) = \tilde{\gamma}(s(t)) \, , \quad \forall t \in (a, b) \, .$$

Differentiating the above expression and using the chain rule we get

$$\dot{\gamma}(t) = \dot{\tilde{\gamma}}(s(t)) \, \dot{s}(t) = \dot{\tilde{\gamma}}(s(t)) \, \|\dot{\gamma}(t)\|$$

where in the last equality we used (3.51). Taking the absolute value of the above yileds

$$\|\dot{\gamma}(t)\| = \left\|\dot{\tilde{\gamma}}(s(t))\right\| \, \|\dot{\gamma}(t)\| \, . \tag{3.52}$$

Since γ is regular, we have

$$\|\dot{\gamma}(t)\| \neq 0 \, , \quad \forall \, t \in (a, b) \, .$$

Therefore we can divide (3.52) by $\|\dot{\gamma}(t)\|$ and obtain

$$\left\|\dot{\tilde{\gamma}}(s(t))\right\| = 1 \, , \quad \forall \, t \in (a, b) \, .$$

By invertibility of s, the above holds if and only if

$$\left\|\dot{\tilde{\gamma}}(t)\right\| = 1 \, , \quad \forall \, t \in (\tilde{a}, \tilde{b}) \, ,$$

showing that $\tilde{\gamma}$ is a unit speed reparametrization of γ.
Step 2. Reverse implication.
Suppose there exists a unit speed reparametrization of γ denoted by

$$\tilde{\gamma} : (\tilde{a}, \tilde{b}) \to \mathbb{R}^n , \quad \tilde{\gamma} = \gamma \circ \phi$$

for some reparametrization map $\phi : (\tilde{a}, \tilde{b}) \to (a, b)$. Differentiating $\tilde{\gamma} = \gamma \circ \phi$ and using the chain rule we get

$$\dot{\tilde{\gamma}}(t) = \dot{\gamma}(\phi(t)) \, \dot{\phi}(t) \, .$$

Taking the norm

$$\left\|\dot{\tilde{\gamma}}(t)\right\| = \|\dot{\gamma}(\phi(t))\| \, |\dot{\phi}(t)| \, .$$

Since $\tilde{\gamma}$ is unit speed we obtain

$$\|\dot{\gamma}(\phi(t))\| \, |\dot{\phi}(t)| = 1 \, , \quad \forall \, t \in (\tilde{a}, \tilde{b}) \, . \tag{3.53}$$

Since ϕ is a diffeomorphism from $(\tilde{a}, \tilde{b})$ into (a, b), Lemma 3.42 guarantees that

$$\dot{\phi}(t) \neq 0 \, , \quad \forall \, t \in (a, b) \, .$$

In particular (3.53) implies

$$\dot{\gamma}(\phi(t)) \neq 0 \, , \quad \forall \, t \in (\tilde{a}, \tilde{b}) \, .$$

As ϕ is invertible, we also have

$$\dot{\gamma}(t) \neq 0 \, , \quad \forall \, t \in (a, b) \, ,$$

proving that γ is regular.

The proof of Theorem 3.47 told us that, if γ is regular, then

$$\tilde{\gamma} = \gamma \circ s^{-1}$$

is a unit speed reparametrization of γ. In the next proposition we show that the arc-length s is essentially the only unit-speed reparametrization of a regular curve.

Proposition 3.48: Arc-length and unit speed reparametrization

Let $\gamma : (a, b) \to \mathbb{R}^n$ be a regular curve. Let $\tilde{\gamma} : (\tilde{a}, \tilde{b}) \to \mathbb{R}^n$ be reparametrization of γ, so that

$$\gamma(t) = \tilde{\gamma}(\phi(t)), \quad \forall \, t \in (a, b) \, .$$

for some diffeomorphism $\phi : (a, b) \to (\tilde{a}, \tilde{b})$. Denote by

$$s(t) := \int_{t_0}^{t} \|\dot{\gamma}(\tau)\| \, d\tau \, , \quad t \in (a, b)$$

the arc-length of γ starting at any point $t_0 \in (a, b)$. We have:

1. If $\tilde{\gamma}$ is unit speed, then there exists $c \in \mathbb{R}$ such that
$$\phi(t) = \pm s(t) + c \, , \quad \forall \, t \in (a, b) \, . \tag{3.54}$$
2. If ϕ is given by (3.54) for some $c \in \mathbb{R}$, then $\tilde{\gamma}$ is unit speed.

Proof

Step 1. First Point.
First note that a unit speed reparametrization $\tilde{\gamma}$ of γ exists by Theorem 3.47, since γ is assumed to be regular. Thus assume $\tilde{\gamma}$ is unit speed reparametrization of γ. By differentiating both sides of

$$\gamma(t) = \tilde{\gamma}(\phi(t)), \quad \forall\, t \in (a,b)\,,$$

we obtain

$$\dot{\gamma}(t) = \frac{d}{dt}\tilde{\gamma}(\phi(t)) = \dot{\tilde{\gamma}}(\phi(t))\,\dot{\phi}(t)\,.$$

Taking the norms we then have

$$\begin{aligned} \|\dot{\gamma}(t)\| &= \left\|\dot{\tilde{\gamma}}(\phi(t))\,\dot{\phi}(t)\right\| && (3.55)\\ &= \left\|\dot{\tilde{\gamma}}(\phi(t))\right\|\,|\dot{\phi}(t)| && (3.56)\\ &= |\dot{\phi}(t)|\,, && (3.57) \end{aligned}$$

where in the last equality we used that $\tilde{\gamma}$ is unit speed, and so

$$\left\|\dot{\tilde{\gamma}}\right\| \equiv 1\,.$$

To summarize, so far we have proven that

$$\|\dot{\gamma}(t)\| = |\dot{\phi}(t)|\,, \quad \forall\, t \in (a,b)\,.$$

Therefore

$$s(t) = \int_{t_0}^{t} \|\dot{\gamma}(\tau)\|\, d\tau = \int_{t_0}^{t} |\dot{\phi}(\tau)|\, d\tau\,.$$

By the Fundamental Theorem of Calculus we get

$$\dot{s}(t) = |\dot{\phi}(t)|$$

and therefore

$$\phi = \pm s + c$$

for some $c \in \mathbb{R}$, concluding the proof.
Step 2. Second Point.
Suppose that

$$\phi := \pm s + c$$

for some $c \in \mathbb{R}$, so that $\phi : (a,b) \to (\tilde{a}, \tilde{b})$. We have

$$\dot{\phi}(t) = \pm \dot{s}(t) = \pm\,\|\dot{\gamma}(t)\| \neq 0 \qquad (3.58)$$

where the last term is non-zero since γ is regular. Therefore, due to the Inverse Function Theorem, ϕ is invertible with smooth inverse. This proves that $\tilde{\gamma}$ defined by

$$\tilde{\gamma} := \gamma \circ \psi\,, \quad \psi := \phi^{-1}\,,$$

is a reparametrization of γ. In particular

$$\gamma = \tilde{\gamma} \circ \phi \,.$$

Differentiating the above, and recalling (3.58), we get

$$\dot{\gamma}(t) = \dot{\tilde{\gamma}}(\phi(t))\,\dot{\phi}(t) = \dot{\tilde{\gamma}}(\phi(t))\,\left(\pm \left\|\dot{\gamma}(t)\right\|\right) \,.$$

Taking the absolute value of the above yields

$$\left\|\dot{\gamma}(t)\right\| = \left\|\dot{\tilde{\gamma}}(\phi(t))\right\| \left\|\dot{\gamma}(t)\right\| \,.$$

Since γ is regular we can divide by $\left\|\dot{\gamma}(t)\right\|$ to get

$$\left\|\dot{\tilde{\gamma}}(\phi(t))\right\| = 1 \quad \forall\, t \in (a,b) \,.$$

Since ϕ is invertible, the above is equivalent to

$$\left\|\dot{\tilde{\gamma}}(t)\right\| = 1 \quad \forall\, t \in (\tilde{a},\tilde{b}) \,,$$

proving that $\tilde{\gamma}$ is a unit speed reparametrization.

Remark 3.49

Let γ be regular. The above proposition tells us that they are equivalent:

1. Computing a unit speed reparametrization of γ,
2. Computing s the arc-length of γ.

In some cases however, unit speed reparametrization and arc-length are impossible to characterize in terms of elementary functions, even for very simple curves.

Example 3.50: Twisted cubic

Define the **twisted cubic** $\gamma : \mathbb{R} \to \mathbb{R}^3$ by

$$\gamma(t) = (t, t^2, t^3) \,.$$

Therefore

$$\dot{\gamma}(t) = (1, 2t, 3t^2) \,,$$

so that

$$\dot{\gamma}(t) \neq 0 \,, \quad \forall \, t \in \mathbb{R} \,,$$

meaning that γ is regular. In particular we have

$$\|\dot{\gamma}(t)\| = \sqrt{1 + 4t^2 + 9t^4}$$

so that the arc-length of γ is

$$s(t) = \int_{t_0}^{t} \sqrt{1 + 4\tau^2 + 9\tau^4} \, d\tau \,.$$

Since γ is regular, by Proposition 3.48 we know that γ admits a unit speed reparametrization $\tilde{\gamma}$ such that

$$\gamma = \tilde{\gamma} \circ \phi$$

with the diffeomorphism ϕ given by

$$\phi(t) = \pm s(t) + c = \pm \int_{t_0}^{t} \sqrt{1 + 4\tau^2 + 9\tau^4} \, d\tau + c$$

for some $c \in \mathbb{R}$. It can be shown that the above integral does not have a closed form in terms of elementary functions. Therefore the unit speed parametrization $\tilde{\gamma}$ cannot be computed explicitly.

3.7 Closed curves

So far we have seen examples of:

- Curves which are infinite, or **unbounded**. This is for example the parabola

$$\gamma(t) := (t, t^2) \,, \quad \forall \, t \in \mathbb{R} \,,$$

- Curves which are finite and have end-points, such as the semi-circle

$$\gamma(t) := (\cos(t), \sin(t)) \,, \quad \forall \, t \in [0, \pi] \,,$$

- Curves which form **loops**, such as the circle

$$\gamma(t) := (\cos(t), \sin(t)) \,, \quad \forall \, t \in [0, 2\pi] \,.$$

However there are examples of curves which are in between the above types.

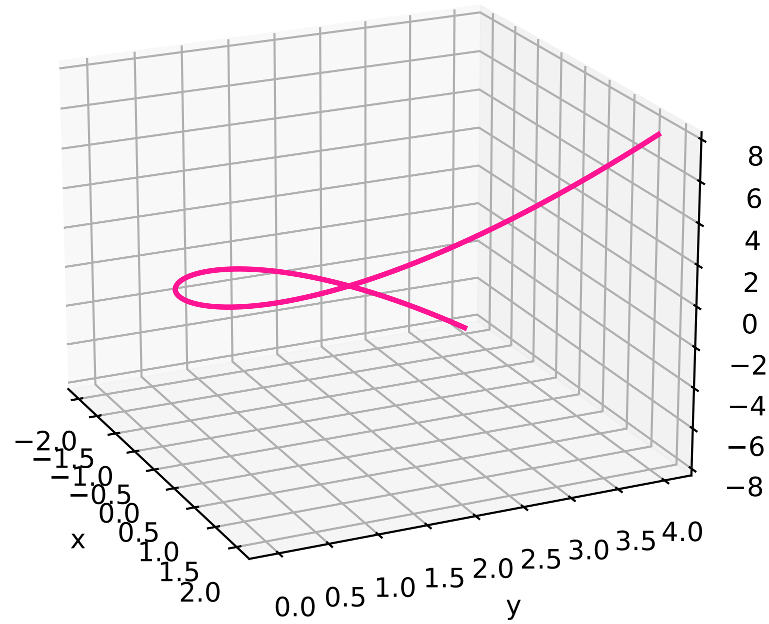


Figure 3.18: Plot of Twisted Cubic for t between -2 and 2

Example 3.51

For example consider the curve $\gamma : \mathbb{R} \to \mathbb{R}^2$

$$\gamma(t) := (t^2 - 1, t^3 - t) \quad \forall\, t \in \mathbb{R}\,.$$

This curve has two main properties:

- γ is unbounded: If define $\tilde{\gamma}$ as the restriction of γ to the time interval $[1, \infty)$, then $\tilde{\gamma}$ is unbounded. A point which starts at $\gamma(1) = (0,0)$ goes towards infinity.
- γ contains a loop: If we define $\tilde{\gamma}$ as the restriction of γ to the time interval $[-1, 1]$, then $\tilde{\gamma}$ is a closed loop starting at $\gamma(-1) = (0,0)$ and returnning at $\gamma(1) = (0,0)$.

The aim of this section is to make precise the concept of **looping curve**. To do that, we need to define **periodic curves**.

Definition 3.52: Periodic curve

Let $\gamma : \mathbb{R} \to \mathbb{R}^n$ be a parametrized curve, and let $T \in \mathbb{R}$. We say that γ is **T-periodic** if

$$\gamma(t) = \gamma(t + T)\,, \quad \forall\, t \in \mathbb{R}\,.$$

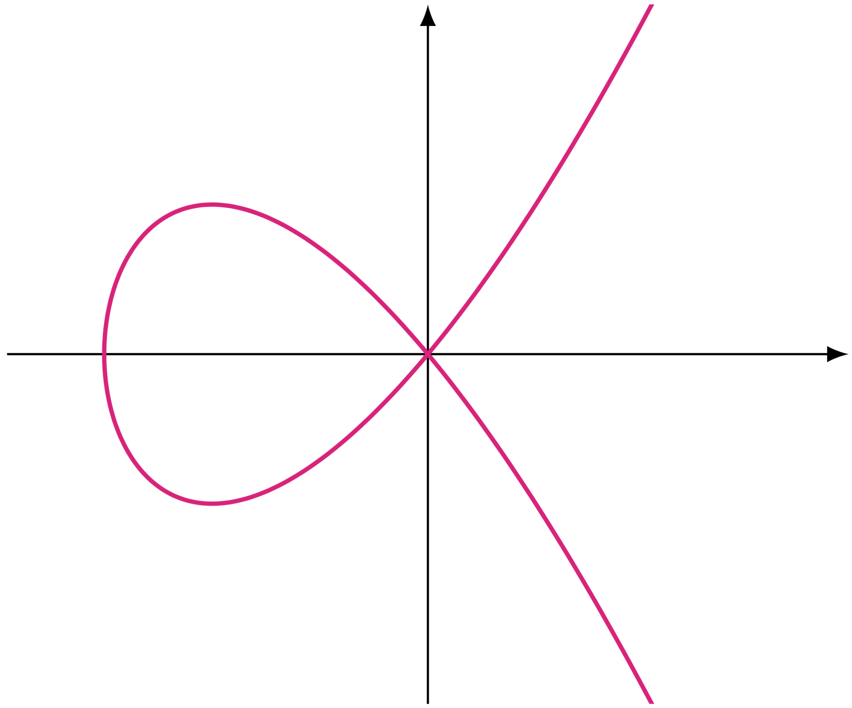


Figure 3.19: Plot of curve $\gamma(t) = (t^2 - 1, t^3 - 1)$ for $t \in [-2, 2]$

Note that every curve is 0-periodic. Therefore to define a closed curve we need to rule out this case.

Definition 3.53: Closed curve

Let $\gamma : \mathbb{R} \to \mathbb{R}^n$ be a parametrized curve. We say that γ is **closed** if:

- γ is not constant,
- γ is T-periodic for some $T \neq 0$.

Remark 3.54

We have the following basic facts:

1. If γ is T-periodic, then a point moving around γ returns to its starting point after time T.

   This is exactly the definition of T-periodicity. Indeed let $p = \gamma(a)$ be the point in question, then
   $$\gamma(a + T) = \gamma(a) = p$$
   by periodicity. Thus γ returns to p after time T.

2. If γ is T-periodic, then γ is determined by its restriction to any interval of length $|T|$.

3. Conversely, suppose that $\gamma : [a, b] \to \mathbb{R}^n$ satisfies
   $$\gamma(a) = \gamma(b) , \quad \frac{d^k\gamma}{dt^k}(a) = \frac{d^k\gamma}{dt^k}(b)$$

for all $k \in \mathbb{N}$. Set

$$T := b - a \, .$$

Then γ can be extended to a T-periodic curve $\tilde{\gamma} : \mathbb{R} \to \mathbb{R}^n$ defined by

$$\tilde{\gamma}(t) := \gamma(\tilde{t}) \, , \quad \tilde{t} := t - \left\lfloor \frac{t-a}{b-a} \right\rfloor (b-a) \, , \quad \forall \, t \in \mathbb{R} \, .$$

The above means that $\tilde{\gamma}(t)$ is defined by $\gamma(\tilde{t})$ where $\tilde{t}$ is the unique point in $[a, b]$ such that

$$t = \tilde{t} + k(b-a)$$

with $k \in \mathbb{Z}$ defined by

$$k := \left\lfloor \frac{t-a}{b-a} \right\rfloor ,$$

see figure below. In this way $\tilde{\gamma}$ is T-periodic.

4. If γ is T-periodic, then it is also $(-T)$-periodic.

   Because if γ is T-periodic then

   $$\gamma(t) = \gamma((t-T)+T) = \gamma(t-T)$$

   where in the first equality we used the trivial identity $t = (t-T)+T$, while in the second equality we used T-periodicity of γ.

5. If γ is T-periodic for some $T \neq 0$, then it is T-periodic for some $T > 0$.

   This is an immediate consequence of Point 4.

6. If γ is T-periodic the γ is (kT)-periodic, for all $k \in \mathbb{Z}$.

   By point 4 we can assume WLOG that $k \geq 0$. We proceed by induction:

   - The statement is true for $k = 1$, since γ is T-periodic.
   - Assume now that γ is kT-periodic. Then

   $$\begin{aligned} \gamma(t + (k+1)T) &= \gamma((t+T) + kT) && (3.59) \\ &= \gamma(t+T) \qquad \text{(by } kT\text{-periodicity)} && (3.60) \\ &= \gamma(t) \qquad \text{(by } T\text{-periodicity)} && (3.61) \end{aligned}$$

   showing that γ is $(k+1)T$-periodic.

   By induction we conclude that γ is (kT)-periodic for all $k \in \mathbb{N}$.

7. If γ is T_1-periodic and T_2-periodic then γ is $(k_1 T_1 + k_2 T_2)$-periodic, for all $k_1, k_2 \in \mathbb{Z}$.

By Point 6 we know that γ is $k_1 T_1$-periodic and $k_2 T_2$-periodic. Set $T := k_1 T_1 + k_2 T_2$. We have

$$
\begin{aligned}
\gamma(t+T) &= \gamma((t + k_1 T_1) + k_2 T_2) && && (3.62)\\
&= \gamma(t + k_1 T_1) && \text{(by } k_2T_2\text{-periodicity)} && (3.63)\\
&= \gamma(t) && \text{(by } k_1T_1\text{-periodicity)} && (3.64)
\end{aligned}
$$

showing that γ is $(k_1 T_1 + k_2 T_2)$-periodic.

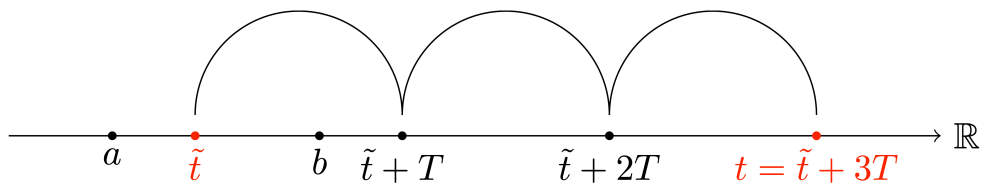


Figure 3.20: The points $t \in \mathbb{R}$ and $\tilde{t} \in [a, b]$ from Point 3 in Remark 3.54. In this skecth $t = \tilde{t} + 3T$, with $T = b - a$.

Definition 3.55

Let γ be a closed curve. The **period** of γ is the smallest $T > 0$ such that γ is T-periodic, that is

$$
\text{Period of } \gamma := \min\{T : \ T > 0\,,\ \gamma \ \text{ is T-periodic}\}\,.
$$

We need to show that the above definition is well-posed, i.e., that there exists such smallest $T > 0$.

Proposition 3.56

Let γ be a closed curve. Then there exists a smallest $T > 0$ such that γ is T-periodic. In other words, the set

$$
S := \{T : \ T > 0\,,\ \gamma \ \text{ is T-periodic}\}\,.
$$

admits positive minumum

$$
P = \min S\,, \quad P > 0\,.
$$

Proof

We make 2 observations about the set S:

- Since γ is closed, we have that γ is T-periodic for some $T \neq 0$. By Remark 3.54 Point 5, we know that T can be chosen such that $T > 0$. Therefore

$$S \neq \emptyset \,.$$

- S is bounded below by 0. This is by definition of S.

Thus, by the Axiom of Completeness of the Real Numbers, the set S admits an infimum

$$P = \inf S \,.$$

The proof is concluded if we show that:
Claim. We have

$$P = \min S \,.$$

This is equivalent to saying that

$$P \in S \,.$$

Proof of claim.
To see that $P \in S$ we need to show that

1. γ is P-periodic,
2. $P > 0$.

Since P is the infimum of S, there exists an infimizing sequence $\{T_n\}_{n\in\mathbb{N}} \subset S$ such that

$$T_n \to P \,.$$

WLOG we can choose T_n decreasing, that is, such that

$$T_1 > T_2 > ... > T_n > ... > 0 \,.$$

Proof of Point 1. As $T_n \in S$, we have that γ is T_n-periodic. Then

$$\gamma(t + T_n) = \gamma(t) \,, \quad \forall \, t \in \mathbb{R} \,, \; n \in \mathbb{N} \,.$$

Since $T_n \to P$, we can take the limit as $n \to \infty$ and use the continuity of γ to obtain

$$\gamma(t) = \lim_{n\to\infty} \gamma(t + T_n) = \gamma(t + P) \,, \quad \forall \, t \in \mathbb{R} \,,$$

showing that γ is P-periodic.
Proof of Point 2. Suppose by contradiction that

$$P = 0 \,.$$

Fix $t \in \mathbb{R}$. Since $T_n > 0$, we can find unique

$$t_n \in [0, T_n] \,, \quad k_n \in \mathbb{Z} \,,$$

such that

$$t = t_n + k_n T_n \,,$$

as shown in the figure below. Indeed, it is sufficient to define

$$k_n := \left\lfloor \frac{t}{T_n} \right\rfloor \in \mathbb{Z} \,, \quad t_n := t - k_n T_n \,.$$

Since $T_n \in S$, we know that γ is T_n-periodic. Remark 3.54 Point 6 implies that γ is also $k_n T_n$-periodic, since $k_n \in \mathbb{Z}$. Thus

$$\begin{aligned} \gamma(t) &= \gamma(t_n + k_n T_n) && \text{(definition of } t_n) && (3.65) \\ &= \gamma(t_n) && \text{(by } k_n T_n\text{-periodicity)} \,. && (3.66) \end{aligned}$$

Therefore

$$\gamma(t) = \gamma(t_n) \,, \quad \forall \, n \in \mathbb{N} \,. \tag{3.67}$$

Also notice that

$$0 \leq t_n \leq T_n \,, \quad \forall \, n \in \mathbb{N} \,.$$

by construction. Since $T_n \to 0$, by the Squeeze Theorem we conclude that

$$t_n \to 0 \quad \text{as } \; n \to \infty \,.$$

Using the continuity of γ, we can pass to the limit in (3.67) and obtain

$$\gamma(t) = \lim_{n \to \infty} \gamma(t_n) = \gamma(0) \,.$$

Since $t \in \mathbb{R}$ was arbitrary, we have shown that

$$\gamma(t) = \gamma(0) \,, \quad \forall \, t \in \mathbb{R} \,.$$

Therefore γ is constant. This is a contradiction, as we were assuming that γ is closed, and, in particular, not constant.

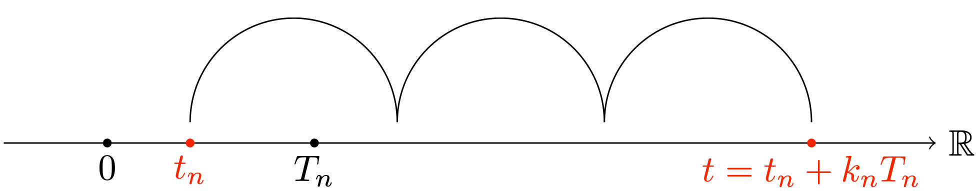


Figure 3.21: For each $t \in \mathbb{R}$ there exist unique $k_n \in \mathbb{Z}$ and $\tilde{t}_n \in [0, T_n]$ such that $t = \tilde{t} + k_n T_n$. In this skecth $k_n = 3$.

Example 3.57

Some examples of closed curves:

- The circumference
$$\gamma(t) = (\cos(t), \sin(t))\,, \quad t \in \mathbb{R}$$
is not costant and is 2π-periodic. Thus γ is closed. The period of γ is 2π.

- The Lemniscate
$$\gamma(t) = (\sin(t), \sin(t)\cos(t))\,, \quad t \in \mathbb{R}$$
is not costant and is 2π-periodic. Thus γ is closed. The period of γ is 2π.

- Consider again the curve from Example 3.51
$$\gamma(t) := (t^2 - 1, t^3 - t)\,, \quad t \in \mathbb{R}\,.$$
According to our definition, γ is not periodic. Therefore γ is not closed. However there is a point of **self-intersection** on γ, namely
$$p := (0,0)\,,$$
for which we have
$$p = \gamma(-1) = \gamma(1)\,.$$

The last curve in the above example motivates the definition of **self-intersecting** curve.

Definition 3.58: Self-intersecting curve

Let $\gamma : \mathbb{R} \to \mathbb{R}^n$ be a parametrized curve. We say that γ is **self-intersecting** at a point p on the curve if

1. There exist two times $a \neq b$ such that
$$p = \gamma(a) = \gamma(b)\,,$$
2. If γ is closed with period T, then $b - a$ is not an integer multiple of T.

Remark 3.59

The second condition in the above definition is important: if we did not require it, then any closed curve would be self-intersecting. Indeed consider a closed curve $\gamma : \mathbb{R} \to \mathbb{R}^n$ and let T be its period. Then by Point 6 in Remark 3.54 we have
$$\gamma(a) = \gamma(a + kT)\,, \quad \forall\ a \in \mathbb{R},\ k \in \mathbb{Z}\,.$$

Therefore every point $\gamma(a)$ would be of self-intersection. Point 2 in the above definition rules this example out. Indeed set $b := a + kT$, then

$$b - a = kT \,,$$

meaning that $b - a$ is an integer multiple of T.

Example 3.60

Let us go back to the curve of Example 3.51, that is,

$$\gamma(t) := (t^2 - 1, t^3 - t) \,, \quad t \in \mathbb{R} \,.$$

We have that γ is not periodic, and therefore not closed. However $p = (0, 0)$ is a point of **self-intersection** on γ, since we have

$$p = \gamma(-1) = \gamma(1) \,.$$

Example 3.61: The Limaçon

Define the parametrized curve $\gamma : \mathbb{R} \to \mathbb{R}^2$ by

$$\gamma(t) = ((1 + 2\cos(t))\cos(t), (1 + 2\cos(t))\sin(t)) \,, \quad \forall\, t \in \mathbb{R} \,.$$

Such curve, plotted bolow, is called limaçon (French for snail). This curve is non constant and 2π-periodic. Therefore it is closed. The period of γ is 2π. Moreover we have

$$\gamma(a) = \gamma(b) = (0, 0) \,.$$

with $a = 2\pi/3$ and $b = 4\pi/3$. Note that

$$b - a = \frac{4\pi}{3} - \frac{2\pi}{3} = \frac{2\pi}{3}$$

which is not an integer multiple of the period 2π. Therefore γ is **self-intersecting** at $(0, 0)$.

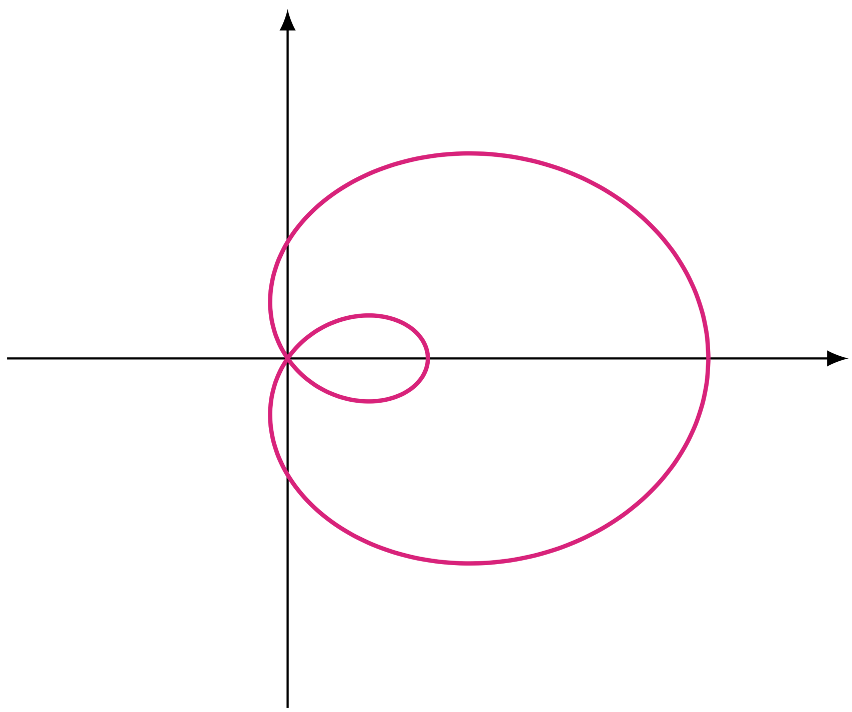


Figure 3.22: Limaçon curve

4 Curvature and Torsion

We have seen how to describe curves and reparametrized them. Now we want to look at local properties of curves:

- How much does a curve twist?
- How much does a curve bend?

We will measure two quantities:

- **Curvature**: measures how much a curve γ deviates from a straight line.
- **Torsion**: measures how much a curve γ fails to lie on a plane.

For example a 2D spiral is curved, but still lies in a plane. Instead the Helix both deviates from a straight line and *pulls away* from any fixed plane.

4.1 Curvature

We start with an informal discussion. Suppose γ is a straight line

$$\gamma(t) = \mathbf{a} + t\mathbf{v}$$

with $\mathbf{a}, \mathbf{v} \in \mathbb{R}^3$. The tangent vector to γ is constant

$$\dot{\gamma}(t) = \mathbf{v}\,.$$

Whatever the definition of curvature will be, it has to hold that γ has zero curvature in this case. If we further derive the tangent vector, we obtain

$$\ddot{\gamma}(t) = 0\,.$$

Thus $\ddot{\gamma}$ seems to be a good candidate for the definition of curvature of γ at the point $\gamma(t)$.

Suppose now that γ is a curve in $\mathbb{R}^2$ with unit speed. We have proven that in this case

$$\dot{\gamma} \cdot \ddot{\gamma} = 0\,,$$

that is, the vector $\ddot{\gamma}$ is orthogonal to the tangent $\dot{\gamma}$ at all times. Now let $\mathbf{n}(t)$ be the unit vector orthogonal to $\dot{\gamma}(t)$ at the point $\gamma(t)$. The amount that the curve γ deviates from its tangent at $\gamma(t)$ after time t_0 is

$$(\gamma(t + t_0) - \gamma(t)) \cdot \mathbf{n}(t)\,, \tag{4.1}$$

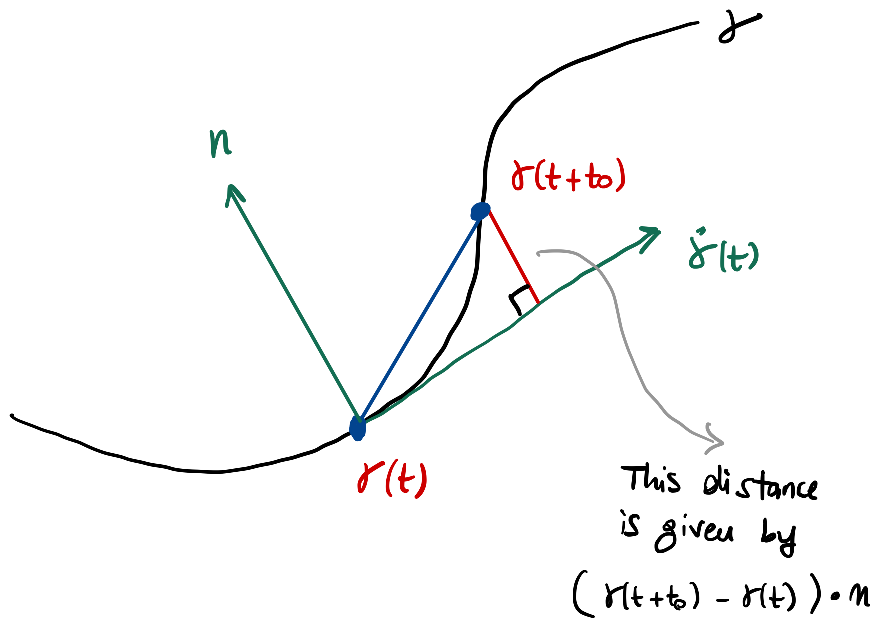


Figure 4.1: Amount that γ deviates from tangent is $(\gamma(t+t_0)-\gamma(t))\cdot\mathbf{n}(t)$

as seen in the figure below.

Equation (4.1) is what we take as measure of curvature. Since

$$\dot\gamma(t)\cdot\ddot\gamma(t)=0 \quad \text{and} \quad \dot\gamma(t)\cdot\mathbf{n}(t)=0\,,$$

we conclude that $\ddot\gamma(t)$ is parallel to $\mathbf{n}(t)$. Since $\mathbf{n}(t)$ is a unit vector, there exists a scalar $\kappa(t)$ such that

$$\ddot\gamma(t)=\kappa(t)\,\mathbf{n}(t)\,.$$

Note that, since $\mathbf{n}$ is unitary, we have

$$\kappa(t)=\|\ddot\gamma(t)\|$$

Now, approximate γ at t with its second order Taylor polynomial:

$$\gamma(t+t_0)=\gamma(t)+\dot\gamma(t)t_0+\frac{\ddot\gamma(t)}{2}t_0^2+o(t_0)$$

with the remainder $o(t_0)$ is such that

$$\lim_{t_0\to 0}\frac{o(t_0)}{t_0^2}=0\,.$$

Therefore, forgetting about the remainder,

$$\gamma(t+t_0)-\gamma(t)\approx\dot\gamma(t)t_0+\frac{\ddot\gamma(t)}{2}t_0^2\,.$$

Multiplying by $\mathbf{n}(t)$ we get

$$(\gamma(t+t_0)-\gamma(t))\cdot\mathbf{n}(t)\approx\dot{\gamma}(t)\cdot\mathbf{n}(t)t_0+\frac{\ddot{\gamma}(t)\cdot\mathbf{n}(t)}{2}t_0^2\,.$$

Recalling that

$$\dot{\gamma}(t)\cdot\mathbf{n}(t)=0\,,\quad \ddot{\gamma}(t)\cdot\mathbf{n}(t)=\kappa(t)\,,$$

we then obtain

$$(\gamma(t+t_0)-\gamma(t))\cdot\mathbf{n}(t)\approx\frac{1}{2}\,\kappa(t)\,t_0^2$$

Important

The amount that γ deviates from a straight line is proportional to

$$\kappa(t)=\|\ddot{\gamma}(t)\|\ .$$

We take this as definition of curvature for a general unit speed curve in $\mathbb{R}^n$.

Definition 4.1

Let $\gamma:(a,b)\to\mathbb{R}^n$ be a unit speed curve. The **curvature** of γ at $\gamma(t)$ is

$$\kappa^\gamma(t):=\|\ddot{\gamma}(t)\|\ .$$

Note that $\kappa(t)$ is a function of time. Therefore the curvature of γ can change from point to point.

We now define curvature for curves which are regular, but not necessarily unit speed.

Definition 4.2

Let $\gamma:(a,b)\to\mathbb{R}^n$ be a regular. The **curvature** of γ at $\gamma(t)$ is

$$\kappa^\gamma(t):=\left\|\ddot{\tilde{\gamma}}(\phi(t))\right\|\,,\quad \forall\,t\in(a,b)\,,$$

where $\tilde{\gamma}$ is a unit speed reparametrization of γ, with $\gamma=\tilde{\gamma}\circ\phi$.

Remark 4.3

The above definition is well posed:

- Since γ is regular, there exist a unit speed reparametrization $\tilde{\gamma}$ of γ.
- If $\hat{\gamma}$ is another unit speed reaprametrization of γ, with $\gamma=\hat{\gamma}\circ\hat{\phi}$, then

$$\kappa^\gamma(t)=\left\|\ddot{\hat{\gamma}}(\hat{\phi}(t))\right\|\,,$$

showing that there is no ambiguity in the definition of κ^γ.

Indeed, since $\tilde\gamma$ and $\hat\gamma$ are both reparametrizations of γ, then

$$\gamma(t) = \tilde\gamma(\tilde\phi(t))\,, \quad \gamma(t) = \hat\gamma(\hat\phi(t))$$

for some diffeomorphisms $\tilde\phi, \hat\phi$. Hence

$$\tilde\gamma(t) = \hat\gamma(\phi(t))\,, \quad \phi := \hat\phi \circ (\tilde\phi)^{-1}\,, \tag{4.2}$$

where ϕ is a diffeomorphism, since it is composition of diffeomorphisms. Differentiating (4.2) we get

$$\dot{\tilde\gamma}(t) = \dot{\hat\gamma}(\phi(t))\dot\phi(t)\,. \tag{4.3}$$

Taking the norms of the above, and recalling that $\tilde\gamma$ and $\hat\gamma$ are unit speed, we get

$$|\dot\phi(t)| = 1\,, \quad \forall\, t\,. \tag{4.4}$$

Since ϕ is a diffeomorphism, we already know that $|\dot\phi| \neq 0$. As $\dot\phi$ is continuous, this means that the sign of $\dot\phi$ is constant. Thus (4.4) implies

$$\dot\phi(t) \equiv 1 \quad \text{or} \quad \dot\phi(t) \equiv -1\,.$$

In both cases, we have

$$\ddot\phi \equiv 0\,.$$

Differentiating (4.3) we then obtain

$$\begin{align}
\ddot{\tilde\gamma}(t) &= \ddot{\hat\gamma}(\phi(t))\dot\phi^2(t) + \dot{\hat\gamma}(\phi(t))\ddot\phi(t) \tag{4.5}\\
&= \ddot{\hat\gamma}(\phi(t))\dot\phi^2(t)\,. \tag{4.6}
\end{align}$$

Taking the norms and using again that $|\dot\phi| \equiv 1$, we get that

$$\left\|\ddot{\tilde\gamma}(t)\right\| = \left\|\ddot{\hat\gamma}(\phi(t))\right\|\,.$$

Recalling that $\phi = \hat\phi \circ (\tilde\phi)^{-1}$ we get

$$\left\|\ddot{\tilde\gamma}(\tilde\phi(t))\right\| = \left\|\ddot{\hat\gamma}(\hat\phi(t))\right\|\,, \quad \forall\, t \in (a,b)\,.$$

Therefore

$$\kappa^\gamma(t) = \left\|\ddot{\tilde\gamma}(\tilde\phi(t))\right\| = \left\|\ddot{\hat\gamma}(\hat\phi(t))\right\|\,.$$

Remark 4.4: Methods for computing curvature

In summary, the curvature of a regular curve

$$\gamma : (a,b) \to \mathbb{R}^n$$

is defined via unit speed reparametrizations of γ. To compute κ we do the following:

- We find a unit speed reparametrization $\tilde{\gamma}$ of the regular curve γ
- This can be done by computing s the arc-length of γ, and then defining

$$\tilde{\gamma} := \gamma \circ \psi \,, \quad \psi := s^{-1}$$

- Then we compute
$$\kappa^{\tilde{\gamma}}(t) = \left\| \ddot{\tilde{\gamma}} \right\| (t)$$
- We obtain the curvature of γ by
$$\kappa^{\gamma}(t) = \kappa^{\tilde{\gamma}}(t)$$

When γ is regular and has values in $\mathbb{R}^3$, there is a way to compute κ without reparametrizing. To do this, we will need the notion of **cross product**, or **vector product**. We will see this in the following sections.

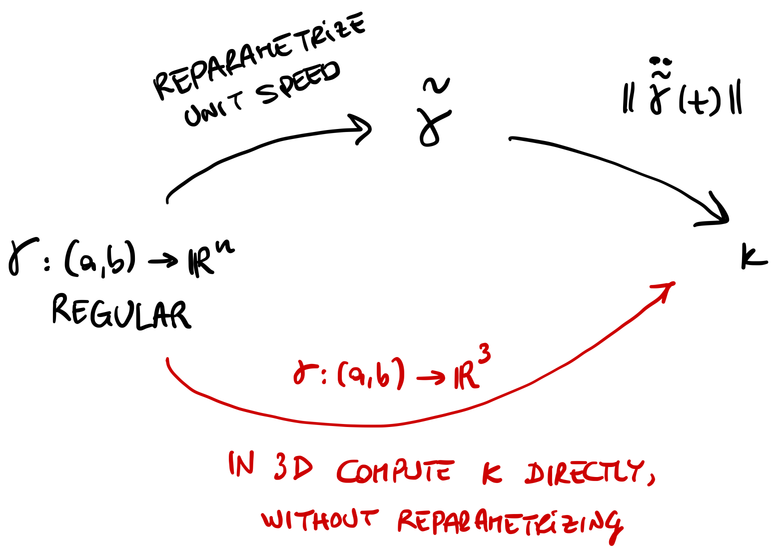


Figure 4.2: Procedure for computing curvature κ

We conclude with an example where we compute κ using reparametrizations.

Example 4.5

Consider the circle of radius $R > 0$:

$$\gamma(t) = (R\cos(t), R\sin(t)) , \quad t \in [0, 2\pi] .$$

To compute the curvature of γ we need to find a unit speed reparametrization. We have shown that:

$$\gamma \text{ regular} \quad \Longrightarrow \quad \phi = s^{-1} \text{ unit speed reparametrization}$$

where s is the arc length of γ:

$$s(t) := \int_{t_0}^{t} \|\dot\gamma(\tau)\| \, d\tau .$$

In our case

$$\dot\gamma(t) = (-R\sin(t), R\cos(t)) \quad \Longrightarrow \quad \|\dot\gamma(t)\| = R$$

and so γ is regular. The arc length starting at $t_0 = 0$ is

$$s(t) = \int_0^t R d\tau = tR .$$

The inverse of s is

$$\phi(t) := s^{-1}(t) = \frac{t}{R} .$$

Therefore a unit speed reparametrization of γ is

$$\tilde\gamma := \gamma \circ \phi$$

which reads

$$\tilde\gamma(t) := \left(R\cos\left(\frac{t}{R}\right), R\sin\left(\frac{t}{R}\right) \right) .$$

We have

$$\dot{\tilde\gamma}(t) = \left(-\sin\left(\frac{t}{R}\right), \cos\left(\frac{t}{R}\right) \right) \tag{4.7}$$

$$\ddot{\tilde\gamma}(t) = \left(-\frac{1}{R}\cos\left(\frac{t}{R}\right), -\frac{1}{R}\sin\left(\frac{t}{R}\right) \right) \tag{4.8}$$

Therefore the curvature of γ is

$$\kappa(t) = \left\|\ddot{\tilde\gamma}(t)\right\| = \frac{1}{R} .$$

In this case $\kappa(t)$ is constant. The curvature also tells us that the smaller the circle, the higher the curvature. For a large circle, like the Earth, the curvature is barely noticeable.

4.2 Vector product in $\mathbb{R}^3$

The discussion in this section follows [2]. We start by defining **orientation** for a vector space.

Definition 4.6: Same orientation

Consider two ordered basis of $\mathbb{R}^3$

$$b = (b_1, b_2, b_3)\,, \quad \tilde{b} = (\tilde{b}_1, \tilde{b}_2, \tilde{b}_3)\,.$$

We say that b and $\tilde{b}$ have the same orientation if the matrix of change of basis has positive determinant.

When two basis b and $\tilde{b}$ have the same orientation, we write

$$b \sim \tilde{b}\,.$$

The above is clearly an equivalence relation on the set of ordered basis. Therefore the set of ordered basis of $\mathbb{R}^3$ can be decomposed into equivalence classes. Since the determinant of the matrix of change of basis can only be positive or negative, there are only two equivalence classes.

Definition 4.7: Orientation

The two equivalence classes determined by $\sim$ on the set of ordered basis are called **orientations**.

Definition 4.8: Positive orientation

Consider the standard basis of $\mathbb{R}^3$

$$e = (e_1, e_2, e_3)$$

where we set

$$e_1 = (1, 0, 0)\,, \quad e_2 = (0, 1, 0)\,, \quad e_3 = (0, 0, 1)\,.$$

Then:

- The orientation corresponding to e is called **positive orientation** of $\mathbb{R}^3$.
- The orientation corresponding to the other equivalence class is called **negative orientation** of $\mathbb{R}^3$.

For a basis b of $\mathbb{R}^3$ we say that:

- b is a **positive basis** if it belongs to the class of e.
- b is a **negative basis** if it does not belong to the class of e.

Example 4.9

Since we are dealing with ordered basis, the order in which vectors appear is fundamental. For example, we defined the equivalence class of

$$e = (e_1, e_2, e_3)\,,$$

to be the positive orientation of $\mathbb{R}^3$. In particular e is a positive basis.
Consider instead

$$\tilde{e} = (e_2, e_1, e_3)\,.$$

The matrix of change of variables between $\tilde{e}$ and e is

$$(e_2|e_1|e_3) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and the latter has negative determinant. Thus $\tilde{e}$ does not belong to the class of e, and is therefore a negative basis.

We are now ready to define the vector product in $\mathbb{R}^3$.

Definition 4.10: Vector product in $\mathbb{R}^3$

Let $u, v \in \mathbb{R}^3$. The vector product of u and v is the unique vector

$$u \times v \in \mathbb{R}^3$$

which satisfies the property:

$$(u \times v) \cdot w = \begin{vmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{vmatrix}\,, \quad \forall\, w \in \mathbb{R}^3\,. \tag{4.9}$$

Here $|a_{ij}|$ denotes the determinant of the matrix (a_{ij}), and

$$u = \sum_{i=1}^{3} u_i e_i\,, \quad v = \sum_{i=1}^{3} v_i e_i\,, \quad w = \sum_{i=1}^{3} w_i e_i\,,$$

with (e_1, e_2, e_3) standard basis of $\mathbb{R}^3$.

The following proposition gives an explicit formula for computing $u \times v$.

Proposition 4.11

Let $u, v \in \mathbb{R}^3$. Then

$$u \times v = \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix} e_1 - \begin{vmatrix} u_1 & u_3 \\ v_1 & v_3 \end{vmatrix} e_2 + \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix} e_3 \,. \tag{4.10}$$

Proof

Denote by $(u \times v)_i$ the i-th component of $u \times v$ with respect to the standard basis, that is,

$$u \times v = \sum_{i=1}^{3} (u \times v)_i \, e_i \,.$$

We can use (4.9) with $w = e_1$ to obtain

$$(u \times v) \cdot e_1 = \begin{vmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ 1 & 0 & 0 \end{vmatrix} = \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix}$$

where we used the Laplace expansion for computing the determinant of the 3×3 matrix. As the standard basis is orthonormal, by bilinearity of the scalar product we get

$$(u \times v) \cdot e_1 = \sum_{i=1}^{3} (u \times v)_i \, e_i \cdot e_1 = (u \times v)_i \,.$$

Therefore we have shown

$$(u \times v)_1 = \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix} \,.$$

Similarly we obtain

$$(u \times v)_2 = \begin{vmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ 0 & 1 & 0 \end{vmatrix} = - \begin{vmatrix} u_1 & u_3 \\ v_1 & v_3 \end{vmatrix}$$

and

$$(u \times v)_3 = \begin{vmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix} \,,$$

from which we conclude.

Proposition 4.12

The vector product in $\mathbb{R}^3$ satisfies the following properties: For all $u, v \in \mathbb{R}^3$

1. $u \times v = -v \times u$
2. $u \times v = 0$ if and only if u and v are linearly dependent
3. $(u \times v) \cdot u = 0$, $(u \times v) \cdot v = 0$
4. Moreover for all $w \in \mathbb{R}^3$, $a, b \in \mathbb{R}$

$$(au + bw) \times v = au \times v + bw \times w$$

The proof, which is based on the properties of determinants, is omitted.

Remark 4.13: Geometric interpretation of vector product

Let $u, v \in \mathbb{R}^3$ be linearly independent. We make some observations:

- Property 3 in Proposition 4.12 says that

$$(u \times v) \cdot u = 0\,, \quad (u \times v) \cdot v = 0\,.$$

  Therefore $u \times v$ is orthogonal to both u and v.

- In particular $u \times v$ is orthogonal to the plane generated by u and v.
- Since u and v are linearly independent, Property 2 in Proposition 4.12 says that

$$u \times v \neq 0$$

- Therefore we have

$$(u \times v) \cdot (u \times v) = \|u \times v\|^2 > 0$$

- On the other hand, using the definition of $u \times v$ with $w = v \times w$ yields

$$(u \times v) \cdot (u \times v) = \begin{vmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ (u \times v)_1 & (u \times v)_2 & (u \times v)_3 \end{vmatrix}$$

- Therefore the determinant of the matrix

$$(u|v|u \times v)$$

  is positive. This shows that

$$(u, v, u \times v)$$

  is a **positive basis** of $\mathbb{R}^3$.

- For all $u, v, x, y \in \mathbb{R}^3$ it holds

$$(u \times v) \cdot (x \times y) = \begin{vmatrix} u \cdot x & v \cdot x \\ u \cdot y & v \cdot y \end{vmatrix} . \tag{4.11}$$

  Indeed, one can check that the above formula holds for the standard vectors e_i, and thus the general formula follows by linearity.

- Using (4.11) we get

$$\begin{aligned} \|u \times v\|^2 = (u \times v) \cdot (u \times v) &= \begin{vmatrix} u \cdot u & v \cdot u \\ u \cdot v & v \cdot v \end{vmatrix} && (4.12) \\ &= \|u\|^2 \|v\|^2 - |u \cdot v|^2 && (4.13) \\ &= \|u\|^2 \|v\|^2 - \|u\|^2 \|v\|^2 \cos^2(\theta) && (4.14) \\ &= \|u\|^2 \|v\|^2 (1 - \cos^2(\theta)) && (4.15) \\ &= \|u\|^2 \|v\|^2 \sin^2(\theta) && (4.16) \\ &= A^2 && (4.17) \end{aligned}$$

  where A is the area of the parallelogram with sides u and v.

Let us summarize the above remark.

Remark 4.14: Summary: Properties of $u \times v$

Let $u, v \in \mathbb{R}^3$ be linearly independent. Then

- $u \times v$ is orthogonal to the plane spanned by u, v
- $\|u \times v\|$ is equal to the area of the parallelogram with sides u, v
- $u \times v$ is such that

$$(u, v, u \times v)$$

  is a positive basis of $\mathbb{R}^3$.

We conclude with a useful proposition for differentiating the cross product of curves in $\mathbb{R}^3$.

Proposition 4.15

Suppose $\gamma, \eta : (a, b) \to \mathbb{R}^3$ are parametrized curves. Then the curve

$$\gamma \times \eta : (a, b) \to \mathbb{R}^3$$

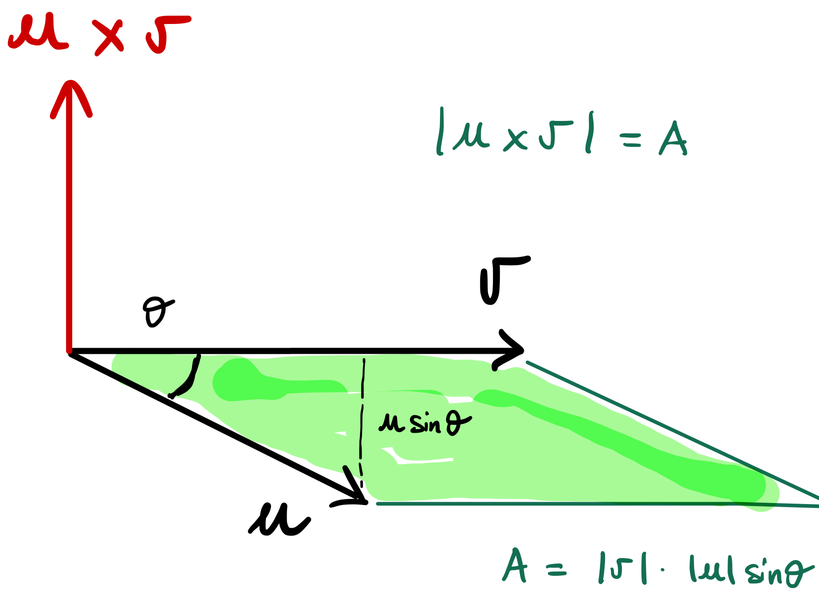


Figure 4.3: For u, v linearly independent, $u \times v$ is orthogonal to the plane generated by u, v. Moreover $|u \times v|$ is the area of the parallelogram with sides u, v, and $(u, v, u \times v)$ is a positive basis of $\mathbb{R}^3$

is smooth, and

$$\frac{d}{dt}(\gamma \times \eta) = \dot{\gamma} \times \eta + \gamma \times \dot{\eta} .$$

The proof is omitted. It follows immediately from formula (4.10).

4.3 Curvature formula in $\mathbb{R}^3$

Given a unit speed curve

$$\gamma : (a, b) \to \mathbb{R}^n$$

we defined its curvature as

$$\kappa(t) = \|\ddot{\gamma}(t)\| .$$

If γ is not unit speed then the curvature is not defined. However, when γ is regular, then we can find a unit-speed reparametrization $\tilde{\gamma}$ of γ, and compute κ as

$$\kappa(t) = \left\|\ddot{\tilde{\gamma}}(t)\right\| .$$

If γ is a regular curve in $\mathbb{R}^3$, there is a way to compute κ without passing through $\tilde{\gamma}$. The formula for computing κ is as follows.

Proposition 4.16

Let $\gamma \colon (a,b) \to \mathbb{R}^3$ be a regular curve. Then the curvature $\kappa(t)$ of γ at $\gamma(t)$ is given by

$$\kappa(t) = \frac{\left\|\ddot{\gamma} \times \dot{\gamma}\right\|}{\left\|\dot{\gamma}\right\|^3} . \tag{4.18}$$

We delay the proof of the above Proposition, as this will get easier when the **Frenet frame** is introduced. For a proof which does not make use of the Frenet frame, see the proof of Proposition 2.1.2 in [5].

For now we use (4.18) the above proposition to compute the curvature on specific curves.

Example 4.17

Consider the straight line

$$\gamma(t) = \mathbf{a} + t\mathbf{v}$$

for some $\mathbf{a}, \mathbf{v} \in \mathbb{R}^3$ fixed, with $\mathbf{v} \neq 0$. Then

$$\dot{\gamma}(t) = \mathbf{v} \,, \quad \ddot{\gamma}(t) = 0 \,.$$

Therefore

$$\left\|\dot{\gamma}(t)\right\| = \left\|\mathbf{v}\right\| \neq 0$$

showing that γ is regular. We have

$$\ddot{\gamma} \times \dot{\gamma} = 0 \times \mathbf{v} = 0 \,.$$

Therefore the curvature is

$$\kappa = \frac{\left\|\ddot{\gamma} \times \dot{\gamma}\right\|}{\left\|\dot{\gamma}\right\|^3} = 0 \,,$$

as expected.

Example 4.18

Consider the Helix of radius $R > 0$ and rise $H > 0$

$$\gamma(t) = (R\cos(t), R\sin(t), Ht) \,, \quad t \in \mathbb{R} \,.$$

Then

$$\dot{\gamma}(t) = (-R\sin(t), R\cos(t), H) \tag{4.19}$$
$$\ddot{\gamma}(t) = (-R\cos(t), -R\sin(t), 0) \tag{4.20}$$

From this we deduce that

$$\|\dot{\gamma}(t)\| = \sqrt{R^2 + H^2}\,,$$

showing that γ is regular. Finally

$$\begin{aligned}
\ddot{\gamma} \times \dot{\gamma} &= \begin{vmatrix} \ddot{\gamma}_2 & \ddot{\gamma}_3 \\ \dot{\gamma}_2 & \dot{\gamma}_3 \end{vmatrix} e_1 - \begin{vmatrix} \ddot{\gamma}_1 & \ddot{\gamma}_3 \\ \dot{\gamma}_1 & \dot{\gamma}_3 \end{vmatrix} e_2 + \begin{vmatrix} \ddot{\gamma}_1 & \ddot{\gamma}_2 \\ \dot{\gamma}_1 & \dot{\gamma}_2 \end{vmatrix} e_3 && (4.21) \\
&= \begin{vmatrix} -R\sin(t) & 0 \\ R\cos(t) & H \end{vmatrix} e_1 - \begin{vmatrix} -R\cos(t) & 0 \\ -R\sin(t) & H \end{vmatrix} e_2 + \begin{vmatrix} -R\cos(t) & -R\sin(t) \\ -R\sin(t) & R\cos(t) \end{vmatrix} e_3 && (4.22) \\
&= \left(-RH\sin(t), RH\cos(t), -R^2\cos^2(t) - R^2\sin^2(t)\right) && (4.23) \\
&= (-RH\sin(t), RH\cos(t), -R^2) && (4.24)
\end{aligned}$$

and therefore

$$\|\ddot{\gamma} \times \dot{\gamma}\| = R\sqrt{R^2 + H^2}\,.$$

By the general formula we have

$$\kappa = \frac{\|\ddot{\gamma} \times \dot{\gamma}\|}{\|\dot{\gamma}\|^3} = \frac{R(R^2 + H^2)^{\frac{1}{2}}}{(R^2 + H^2)^{\frac{3}{2}}} = \frac{R}{R^2 + H^2}$$

We notice the following:

- If $H = 0$ then the Helix is just a circle of radius R. In this case the curvature is

  $$\kappa = \frac{1}{R}$$

  which agrees with the curvature computed for the circle of radius R.

- If $R = 0$ then the Helix is just parametrizing the z-axis. In this case the curvature is

  $$\kappa = 0\,,$$

  which agrees with the curvature of a straight line.

4.4 Signed curvature of plane curves

In this section we assume to have plane curves, that is, curves with values in $\mathbb{R}^2$. In this case we can give a geometric interpretation for the sign of the curvature. This cannot be done in higher dimension.

Definition 4.19

Let $\gamma : (a, b) \to \mathbb{R}^2$ be unit speed. We define the **signed unit normal** to γ at $\gamma(t)$ as the unit vector $\mathbf{n}(t)$ obtained by rotating $\dot\gamma(t)$ anti-clockwise by an angle of $\pi/2$.

Definition 4.20

Let $\gamma : (a, b) \to \mathbb{R}^2$ be unit speed. The **signed curvature** of γ at $\gamma(t)$ is the scalar $\kappa_s(t)$ such that

$$\ddot\gamma(t) = k_s(t)\mathbf{n}(t)$$

Remark 4.21

Notice that since $\mathbf{n}$ is a unit vector and γ is unit speed, then

$$|\kappa_s(t)| = \|\ddot\gamma(t)\| = \kappa(t)\,.$$

Thus the signed curvature is related to the curvature by

$$\kappa_s(t) = \pm\kappa(t)\,.$$

Remark 4.22

It can be shown that the signed curvature is the rate at which the tangent vector $\dot\gamma$ of the curve γ rotates. The signed curvature is:

- positive if $\dot\gamma$ is rotating anti-clockwise
- negative if $\dot\gamma$ is rotating clockwise

In other words,

- $k_s > 0$ means the curve is turning left,
- $k_s < 0$ means the curve is turning right.

A rigorous justification of the above statement is found in Proposition 2.2.3 in [5].

For curves which are not unit speed, we define the signed curvature as the signed curvature of the unit speed reparametrization.

Definition 4.23

Let $\gamma : (a,b) \to \mathbb{R}^2$ be regular and let $\tilde{\gamma}$ be a unit speed reparametrization of γ. The **signed curvature** of γ at $\gamma(t)$ is the scalar $\kappa_s(t)$ such that

$$\ddot{\tilde{\gamma}}(t) = k_s(t)\mathbf{n}(t)\,,$$

where $\mathbf{n}(t)$ is the unit vector obtained by rotating $\dot{\tilde{\gamma}}(t)$ anti-clockwise by an angle $\pi/2$.

The signed curvature completely characterizes plane curves, in the sense of the following theorem.

Theorem 4.24

Let $\phi : \mathbb{R} \to \mathbb{R}$ be smooth. Then:

1. There exists a parametrized curve $\gamma : \mathbb{R} \to \mathbb{R}^2$ such that its signed curvature κ_s satisfies

$$\kappa_s(t) = \phi(t)\,, \quad \forall\, t \in \mathbb{R}\,.$$

2. Suppose that $\tilde{\gamma} : \mathbb{R} \to \mathbb{R}^2$ is any curve such that its signed curvature $\tilde{\kappa}_s$ satisfies

$$\tilde{\kappa}_s(t) = \phi(t)\,, \quad \forall\, t \in \mathbb{R}\,.$$

   Then

$$\tilde{\gamma} = \gamma$$

   up to rotations and translations.

We do not prove the above theorem. For a proof, see Theorem 2.2.6 in [5].

4.5 Space curves

In this section we deal with **space curves**, that is, curves with values in $\mathbb{R}^3$. There are several issues compare to the plane case:

- A 3D counterpart of the signed curvature does not exist, since there is no notion of *turning left* or *turning right*.
- We have seen in the previous section that the signed curvature completely characterizes plane curves. In 3D however curvature is not enough to characterize curves: there exist γ and η space curves such that

$$\kappa^\gamma = \kappa^\eta\,, \quad \gamma \neq \eta\,,$$

that is, γ and η have same curvature but are different curves.

Example 4.25

Let γ be a circle of radius $R > 0$

$$\gamma(t) = (R\cos(t), R\sin(t), 0) ,$$

and η be a helix of radius $S > 0$ and rise $H > 0$

$$\eta(t) = (S\cos(t), S\sin(t), Ht) .$$

We have computed that

$$\kappa^\gamma = \frac{1}{R} , \quad \kappa^\eta = \frac{S}{\sqrt{S^2 + H^2}} .$$

5 Isoperimetric Inequality

6 Topology

Topology $\mathbb{R}$

7 Surfaces

Definition 7.1

A set $\mathcal{S} \subset \mathbb{R}^3$ is a **surface** if for every point $x \in \mathcal{S}$ there exist open sets $U \subset \mathbb{R}^2$, $V \subset \mathbb{R}^3$ such that

- $x \in V$,
- U is diffeomorphic to $V \cap \mathcal{S}$.

Further:

- A diffeomorphism of U into $V \cap \mathcal{S}$, denoted by

$$\sigma \, : U \to V \cap \mathcal{S}$$

  is called a **surface chart**.
- For each $i \in I$ suppose to have a surface chart

$$\sigma_i \, : U_i \to V_i \cap \mathcal{S} \, .$$

  We say that the family $\{\sigma_i\}_{i \in I}$ is an **atlas** of $\mathcal{S}$ if

$$\bigcup_{i \in I} (V_i \cap \mathcal{S}) = \mathcal{S} \, .$$

Note that a surface chart σ is a map from $\mathbb{R}^2$ into $\mathbb{R}^3$. Points in U will be denoted by the pair (u, v), while points in $\mathbb{R}^3$ by $x = (x_1, x_2, x_3)$.

Definition 7.2

Let $U \subset \mathbb{R}^2$ be open. A surface chart

$$\sigma = \sigma(u, v) \, : U \to \mathbb{R}^3$$

is called **regular** if the partial derivatives

$$\sigma_u(u, v) = \frac{d\sigma}{du}(u, v) \, , \quad \sigma_v(u, v) = \frac{d\sigma}{dv}(u, v)$$

are linearly independent vectors of $\mathbb{R}^3$ for all $(u, v) \in U$.

8 Surfaces in Python

8.1 Plots with Matplotlib

I will take for granted all the commands explained in Chapter 2. Suppose we want to plot a surface S which is defined by the parametric equations

$$x = x(u,v)\,, \quad y = y(u,v)\,, \quad z = z(u,v)$$

for $u \in (a,b)$ and $v \in (c,d)$. This can be done via the function called `plot_surface` contained in the mplot3d Toolkit. This function works as follows: first we generate a mesh-grid $[U,V]$ from the coordinates (u,v) via the command

```
[U, V] = np.meshgrid(u, v)
```

Then we compute the parametric surface on the mesh

```
x = x (U, V)
y = y (U, V)
z = z (U, V)
```

Finally we can plot the surface with the command

```
plt.plot_surface(x, y, z)
```

The complete code looks as follows.

```
# Plotting surface S

# Importing numpy, matplotlib and mplot3d
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

# Generates figure object of size m x n
fig = plt.figure(figsize = (m,n))
```

```
# Generates 3D axes
ax = plt.axes(projection = '3d')

# Shows axes grid
ax.grid(True)

# Generates coordinates u and v
# by dividing the interval (a,b) in n parts
# and the interval (c,d) in m parts
u = np.linspace(a, b, m)
v = np.linspace(c, d, n)

# Generates grid [U,V] from the coordinates u, v
U, V = np.meshgrid(u, v)

# Computes S given the functions x, y, z
# on the grid [U,V]
x = x(U,V)
y = y(U,V)
z = z(U,V)

# Plots the surface S
ax.plot_surface(x, y, z)

# Setting plot title
ax.set_title('The surface S')

# Setting axes labels
ax.set_xlabel('x', labelpad=10)
ax.set_ylabel('y', labelpad=10)
ax.set_zlabel('z', labelpad=10)

# Setting viewing angle
ax.view_init(elev = e, azim = a)

# Showing the plot
plt.show()
```

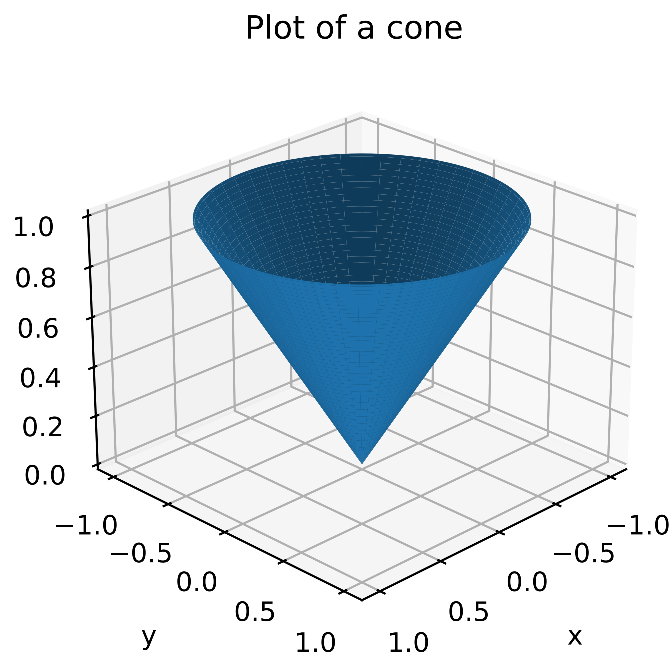
For example let us plot a cone described parametrically by:

$$x = u\cos(v)\,, \quad y = u\sin(v)\,, \quad z = u$$

for $u \in (0,1)$ and $v \in (0, 2\pi)$. We adapt the above code:

```
# Plotting a cone

# Importing numpy, matplotlib and mplot3d
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

# Generates figure object of size 4 x 4
fig = plt.figure(figsize = (4,4))

# Generates 3D axes
ax = plt.axes(projection = '3d')

# Shows axes grid
ax.grid(True)

# Generates coordinates u and v by dividing
# the intervals (0,1) and (0,2pi) in 100 parts
u = np.linspace(0, 1, 100)
v = np.linspace(0, 2*np.pi, 100)

# Generates grid [U,V] from the coordinates u, v
U, V = np.meshgrid(u, v)

# Computes the surface on grid [U,V]
x = U * np.cos(V)
y = U * np.sin(V)
z = U

# Plots the cone
ax.plot_surface(x, y, z)

# Setting plot title
ax.set_title('Plot of a cone')

# Setting axes labels
ax.set_xlabel('x', labelpad=10)
ax.set_ylabel('y', labelpad=10)
ax.set_zlabel('z', labelpad=10)

# Setting viewing angle
ax.view_init(elev = 25, azim = 45)
```

```
# Showing the plot
plt.show()
```



As discussed in Chapter 2, we can have multiple plots in the same figure. For example let us plot the torus viewed from 2 angles. The parametric equations are:

$$
\begin{aligned}
x &= (R + r\cos(u))\cos(v)\\
y &= (R + r\cos(u))\sin(v)\\
z &= r\sin(u)
\end{aligned}
$$

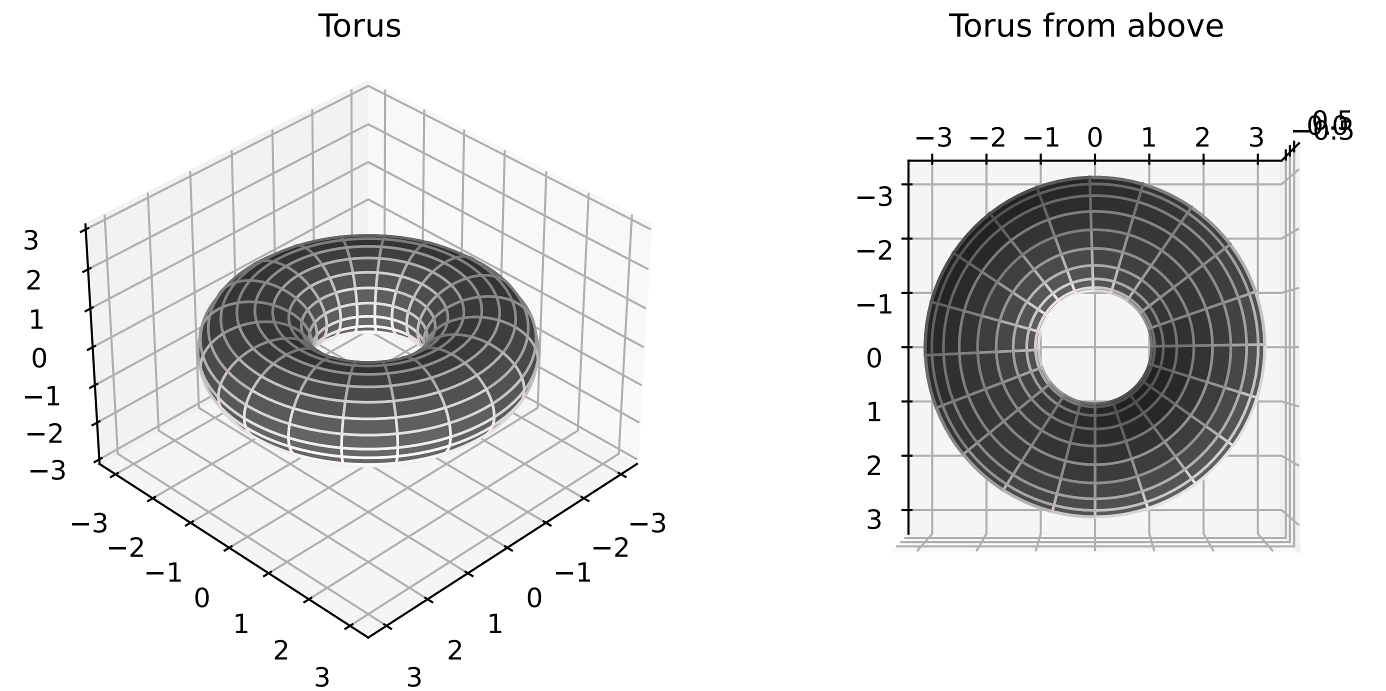
for $u, v \in (0, 2\pi)$ and with

- R distance from the center of the tube to the center of the torus
- r radius of the tube

```
# Plotting torus seen from 2 angles

# Importing numpy, matplotlib and mplot3d
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

# Generates figure object of size 9 x 5
fig = plt.figure(figsize = (9,5))
```

```
# Generates 2 sets of 3D axes
ax1 = fig.add_subplot(1, 2, 1, projection = '3d')
ax2 = fig.add_subplot(1, 2, 2, projection = '3d')

# Shows axes grid
ax1.grid(True)
ax2.grid(True)

# Generates coordinates u and v by dividing
# the interval (0,2pi) in 100 parts
u = np.linspace(0, 2*np.pi, 100)
v = np.linspace(0, 2*np.pi, 100)

# Generates grid [U,V] from the coordinates u, v
U, V = np.meshgrid(u, v)

# Computes the torus on grid [U,V]
# with radii r = 1 and R = 2
R = 2
r = 1

x = (R + r * np.cos(U)) * np.cos(V)
y = (R + r * np.cos(U)) * np.sin(V)
z = r * np.sin(U)

# Plots the torus on both axes
ax1.plot_surface(x, y, z, rstride = 5, cstride = 5, color = 'dimgray',
 ↪  edgecolors = 'snow')

ax2.plot_surface(x, y, z, rstride = 5, cstride = 5, color = 'dimgray',
 ↪  edgecolors = 'snow')

# Setting plot titles
ax1.set_title('Torus')
ax2.set_title('Torus from above')

# Setting range for z axis in ax1
ax1.set_zlim(-3,3)

# Setting viewing angles
ax1.view_init(elev = 35, azim = 45)
ax2.view_init(elev = 90, azim = 0)
```

```
# Showing the plot
plt.show()
```



Notice that we have added some customization to the `plot_surface` command. Namely, we have set the color of the figure with `color = 'dimgray'` and of the edges with `edgecolors = 'snow'`. Moreover the commands `rstride` and `cstride` set the number of *wires* you see in the plot. More precisely, they set by how much the data in the mesh $[U, V]$ is downsampled in each direction, where rstride sets the row direction, and cstride sets the column direction. On the torus this is a bit difficult to visualize, due to the fact that $[U, V]$ represents angular coordinates. To appreciate the effect, we can plot for example the paraboiloid

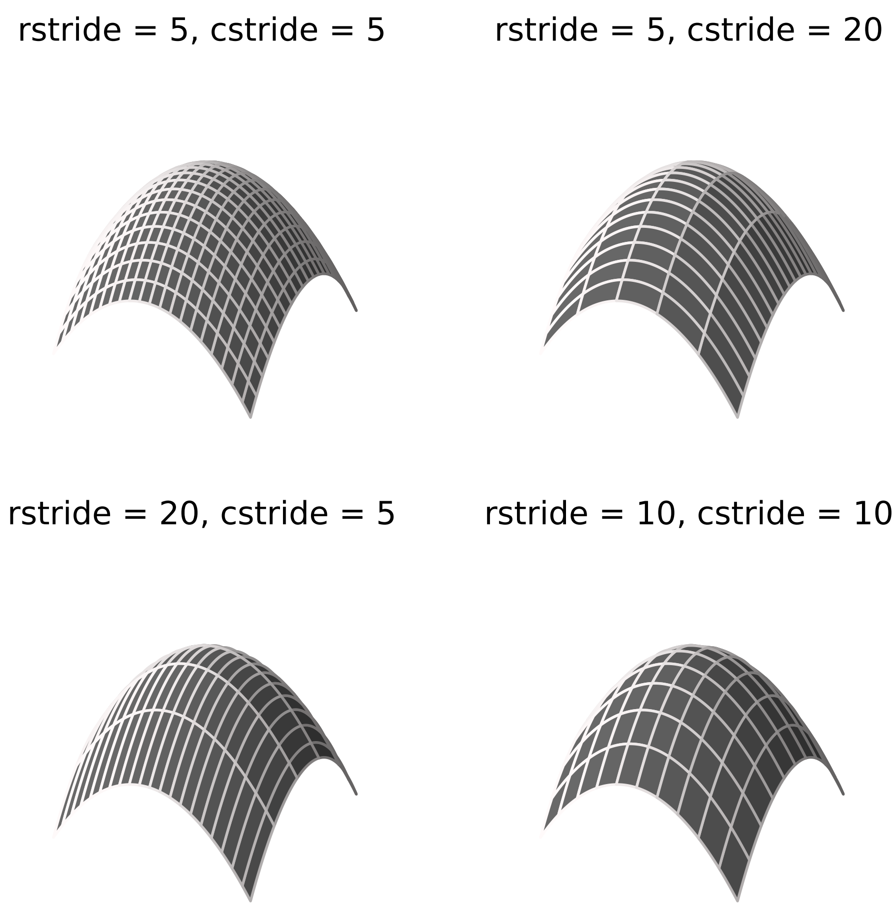
$$
\begin{aligned}
x &= u \\
y &= v \\
z &= -u^2 - v^2
\end{aligned}
$$

for $u, v \in [-1, 1]$.

```
# Showing the effect of rstride and cstride

# Importing numpy, matplotlib and mplot3d
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
```

```python
# Generates figure object of size 6 x 6
fig = plt.figure(figsize = (6,6))

# Generates 2 sets of 3D axes
ax1 = fig.add_subplot(2, 2, 1, projection = '3d')
ax2 = fig.add_subplot(2, 2, 2, projection = '3d')
ax3 = fig.add_subplot(2, 2, 3, projection = '3d')
ax4 = fig.add_subplot(2, 2, 4, projection = '3d')

# Generates coordinates u and v by dividing
# the interval (-1,1) in 100 parts
u = np.linspace(-1, 1, 100)
v = np.linspace(-1, 1, 100)

# Generates grid [U,V] from the coordinates u, v
U, V = np.meshgrid(u, v)

# Computes the paraboloid on grid [U,V]
x = U
y = V
z = - U**2 - V**2

# Plots the paraboloid on the 4 axes
# but with different stride settings
ax1.plot_surface(x, y, z, rstride = 5, cstride = 5, color = 'dimgray',
↪   edgecolors = 'snow')

ax2.plot_surface(x, y, z, rstride = 5, cstride = 20, color = 'dimgray',
↪   edgecolors = 'snow')

ax3.plot_surface(x, y, z, rstride = 20, cstride = 5, color = 'dimgray',
↪   edgecolors = 'snow')

ax4.plot_surface(x, y, z, rstride = 10, cstride = 10, color = 'dimgray',
↪   edgecolors = 'snow')

# Setting plot titles
ax1.set_title('rstride = 5, cstride = 5')
ax2.set_title('rstride = 5, cstride = 20')
ax3.set_title('rstride = 20, cstride = 5')
ax4.set_title('rstride = 10, cstride = 10')
```

```
# We do not plot axes, to get cleaner pictures
ax1.axis('off')
ax2.axis('off')
ax3.axis('off')
ax4.axis('off')

# Showing the plot
plt.show()
```



In this case our mesh is `100 x 100`, since `u` and `v` both have 100 components. Therefore setting `rstride` and `cstride` to 5 implies that each row and column of the mesh is sampled one time every 5 elements, for a total of

$$100/5 = 20$$

samples in each direction. This is why in the first picture you see a `20 x 20` grid. If instead one sets `rstride` and `cstride` to 10, then each row and column of the mesh is sampled one time every 10 elements, for a total of

$$100/10 = 10$$

samples in each direction. This is why in the fourth figure you see a `10x10` grid.

8.2 Plots with Plotly

As done in Section 2.4, we now see how to use `Plotly` to generate an interactive 3D plot of a surface. This can be done by means of functions contained in the `Plotly` module `graph_objects`, usually imported as `go`. Specifically, we will use the function `go.Surface`. The code will look similar to the one used to plot surfaces with `matplotlib`:

- generate meshgrid on which to compute the parametric surface,
- store such surface in the numpy array `[x,y,z]`,
- pass the array `[x,y,z]` to `go.Surface` to produce the plot.

The full code is below.

```
# Plotting a Torus with Plotly

# Import "numpy" and the "graph_objects" module from Plotly
import numpy as np
import plotly.graph_objects as go

# Generates coordinates u and v by dividing
# the interval (0,2pi) in 100 parts
u = np.linspace(0, 2*np.pi, 100)
v = np.linspace(0, 2*np.pi, 100)

# Generates grid [U,V] from the coordinates u, v
U, V = np.meshgrid(u, v)

# Computes the torus on grid [U,V]
# with radii r = 1 and R = 2
R = 2
r = 1

x = (R + r * np.cos(U)) * np.cos(V)
y = (R + r * np.cos(U)) * np.sin(V)
z = r * np.sin(U)

# Generate and empty figure object with Plotly
# and saves it to the variable called "fig"
fig = go.Figure()

# Plot the torus with go.Surface and store it
```

```
# in the variable "data". We also do now show the
# plot scale, and set the color map to "teal"
data = go.Surface(
    x = x , y = y, z = z,
    showscale = False,
    colorscale='teal'
    )

# Add the plot stored in "data" to the figure "fig"
# This is done with the command add_trace
fig.add_trace(data)

# Set the title of the figure in "fig"
fig.update_layout(title_text="Plotting a Torus with Plotly")

# Show the figure
fig.show()
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): text/html
```

The above code generates an image that cannot be rendered in pdf. To see the output, see the link to the digital version of these notes. To further customize your plots, you can check out the documentation of `go.Surface` at this link. For example, note that we have set the colormap to `teal`: for all the pretty colorscales available in Plotly, see this page.

One could go even fancier and use the tri-surf plots in `Plotly`. This is done with the function `create_trisurf` contained in the module `figure_factory` of `Plotly`, usually imported as `ff`. The documentation can be found here. We also need to import the Python library `scipy`, which we use to generate a *Delaunay triangulation* for our plot. Let us for example plot the torus.

```
# Plotting Torus with tri-surf

# Importing libraries
import numpy as np
import plotly.figure_factory as ff
from scipy.spatial import Delaunay

# Generates coordinates u and v by dividing
# the interval (0,2pi) in 100 parts
u = np.linspace(0, 2*np.pi, 20)
v = np.linspace(0, 2*np.pi, 20)
```

```python
# Generates grid [U,V] from the coordinates u, v
U, V = np.meshgrid(u, v)

# Collapse meshes to 1D array
# This is needed for create_trisurf
U = U.flatten()
V = V.flatten()

# Computes the torus on grid [U,V]
# with radii r = 1 and R = 2
R = 2
r = 1

x = (R + r * np.cos(U)) * np.cos(V)
y = (R + r * np.cos(U)) * np.sin(V)
z = r * np.sin(U)

# Generate Delaunay triangulation
points2D = np.vstack([U,V]).T
tri = Delaunay(points2D)
simplices = tri.simplices

# Plot the Torus
fig = ff.create_trisurf(
    x=x, y=y, z=z,
    colormap = "Portland",
    simplices=simplices,
    title="Torus with tri-surf",
    aspectratio=dict(x=1, y=1, z=0.3),
    show_colorbar = False
    )

# Adjust figure size
fig.update_layout(autosize = False, width = 700, height = 700)

# Show the figure
fig.show()
```

```
Unable to display output for mime type(s): text/html
```

Again, the above code generates an image that cannot be rendered in pdf. To see the output, see the link to the digital version of these notes.

License

Reuse

Citation

For attribution, please cite this work as:

> Fanzon, Silvio. (2023). *Lecture Notes on Differential Geometry.*
> https://www.silviofanzon.com/2023-Differential-Geometry-Notes/

BibTex citation:

```
@electronic{Danzon-Diff-Geom-2023,
    author = {Fanzon, Silvio},
    title = {Lecture Notes on Differential Geometry},
    url = {https://www.silviofanzon.com/2023-Differential-Geometry-Notes/},
    year = {2023}}
```

References

[1] C. Bär. *Elementary Differential Geometry*. Cambridge University Press, 2010.

[2] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Second Edition. Dover Books on Mathematics, 2017.

[3] R. Johansson. *Numerical Python. Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib*. Second Edition. Apress, 2019.

[4] Q. Kong, T. Siauw, and A. Bayen. *Python Programming and Numerical Methods*. Academic Press, 2020.

[5] A. Pressley. *Elementary Differential Geometry*. Second Edition. Springer, 2010.

[6] V. A. Zorich. *Mathematical Analysis I*. Second Edition. Springer, 2015.

[7] V. A. Zorich. *Mathematical Analysis II*. Second Edition. Springer, 2016.