# Cálculo

Uma Abordagem Computacional

Gustavo Mirapalheta

2022-02-07

# Sumário

# Capítulo 1

# Introduction

*Repetitia matter scientia est.* * Repetition is the mother of science*. From its earliest beginnings, science has had at its core the idea of the repetition of experiments. This repetition was responsible for the correct accumulation of knowledge and the growing confidence that people in general, scientists or not, placed in its teachings. With mathematics it could be no different. It is repetition, sometimes to the point of exhaustion, that makes the student confident in his abilities and consequently capable of correctly applying his knowledge to solve new problems.

For a long time, repetition was limited to the minimal algebraic ability that was required of students. This steep learning curve made the study of mathematics and its applications an arduous task. Minimizing the effects of this curve would make the process of teaching science and engineering as a whole more efficient. Until recently few resources existed to assist in this task, and those that did exist were of a very basic nature (slide rules, mechanical calculators, etc.). However, starting in the 1970s electronic calculators appeared, which greatly accelerated the repetitive part of solving numerical exercises, making the student able to verify by himself the correctness of his results. With the rapid spread of personal computers from the 1990s on, other uses were found for computational resources, including in the realm of algebraic problem solving.

As we enter the third decade of the 21st century, there is a wide range of software available, much of it free, that frees the student from the burden of repetitive calculation, be it numerical or algebraic. The teacher can now concentrate his energy on designing problems that focus on thinking, on imagination, without worrying about the needs of calculation, be it numerical or algebraic. The initial effort to perform numerical or algebraic calculations is now replaced by the need to know these tools thoroughly. Thus, designing textbooks based on the premise of using such software will certainly bring great benefits to students in terms of speed of learning, and to teachers in terms of the scope that can be given to the

teaching of math and science in a given period of time.

This was the premise that guided the writing of this text. The idea here is to teach the various applications of higher mathematics to quantitative problem solving, from the beginning using numerical and/or algebraic solving tools. The methodology used is focused on exercises with a minimum of theory that is sometimes presented along with the resolution of the examples themselves. This is expected to make learning more intuitive, dynamic and effective, freeing up more time for teaching problems that make a difference in the development of modeling and quantitative solving skills.

Knowing that the use of computational resources will be key to the success of the task we set out to tackle, we need to choose a preferred tool. There are many software programs on the market with a wide range of features, but in the last few years one tool has established itself, both in academia and in industry, either for numerical or algebraic problem solving: Python. Thus, the course will rely heavily on this language and the `numpy` and `sympy` packages for problem solving. Besides allowing the rapid absorption of concepts through the computational approach, the use of these tools has recently become a market requirement in the training of students. If the student absorbs little or nothing of the theoretical concepts of higher mathematics, training in the `Python` environment will certainly be of unparalleled benefit when the student seeks a professional placement in the job market.

It is known that without modern technology the welfare society would not be possible. This technology became a reality because our ancestors changed the way they saw the world, and started to do science. And modern science, from its beginnings in the European Renaissance in the 16th century, was built on a mathematical foundation. Mathematics is therefore essential to the modern world. Ways that bring more and more students into the exciting world of science and technological challenges of the 21st century will in this way be very useful for the development of the whole society.

Gustavo Corrêa Mirapalheta

São Paulo, March 31, 2022

# Capítulo 2

# Introduction to the Python Language

The Python language will be used extensively in this book for teaching mathematics. So our work begins with a basic understanding of it and the programming environment that will be used, Google Colab. This chapter can also be used independently in a programming language course.

## 2.1 The Google Colab Environment

Before starting our work with the Python language it is important to define and configure the environment in which it will be used. In the case of the examples and exercises in this text we recommend the use of the Google Colab environment.

Google's cloud environment is available at `http://colab.research.google.com`. To access the environment, create a Gmail account and use the user of that account to log into colab. The Google Colab environment uses the concept of Jupyter Python Notebooks. A Python Notebook is a file divided into cells. Each cell is a Python command block (a small program) that can be executed with `Ctrl+Enter`.

To start programming in Colab you must follow the procedure described below:

1. Create a Gmail account. If you already have one, log into it.
2. Access the Google Colab system on the site `http://colab.research.google.com`
3. Create a Jupyter Notebook
4. Edit its name to `class01.ipynb` `Class notes from` and its name.

5. Insert a text cell of type `Heading` into the file with the phrase: `First Python Class`. For python to interpret it as a title cell you need to precede the text with a hashtag `#`
6. Reposition the title cell to above the text cell.
7. Insert below the title cell a `Subheading` cell in the file with the phrase: `Access to Google Colab Environment`. For python to interpret it as a subheading cell you need to prepend the text with two hashtags
8. Insert a code cell below the text cell.
9. Insert in it the python print command: `print` with the sentence `Good morning. Welcome to your first Python class`. Then execute the code by pressing `Ctrl+Enter`. The text must be enclosed in parentheses and double quotes as shown below.

```
print("Good morning. Welcome to your first Python class")
```

```
Good morning. Welcome to your first Python class
```

## 2.2 Basic Mathematical Expressions

Python can be used like a calculator, in calculating the result of arithmetic expressions. The results can be displayed by using the `print` function

Addition, subtraction and multiplication are performed with the usual operators:

```
print(2+2)
```

```
4
```

```
print(5-3)
```

```
2
```

```
print(3*5)
```

```
15
```

Unlike most environments, where exponentiation is achieved with ^, in Python it is performed with the double star (`**`), as can be seen below:

```
print(2**3)
```

```
8
```

The integer part of a division can be obtained with `//`

```
b = 22 // 8
print(b)
```

```
2
```

The remainder of a division is obtained with the % operator

```
c = 22 % 8
print(c)
```

```
6
```

To change the precedence of operators we use parentheses:

```
d = (5 - 1) * ( (7 + 1) / (3 - 1) )
print(d)
```

```
16.0
```

### 2.2.1 Exercises

Calculate using Python:

1. The remainder of the division of 5 by 2.
2. $5^4$, $(-5)^4$, $-(5^4)$, $5^{-4}$, $\frac{5^4}{5^3}$, $\left(\frac{5}{3}\right)^{-2}$, $64^{-\frac{3}{4}}$
3. $5432.\sqrt{23}$. Hint: Raise to 0.5 to get the square root.
4. The average of the numbers 730, 629 and 647.
5. Enter the value $4.445.10^8$ in scientific notation. Note: in Python, scientific notation is represented by the lower case letter `e`.
6. The integer part of $-22.3$. Tip: Use the `int` function.
7. Round $-22.3$ to zero decimal places. Tip: Use the `round` function.
8. The absolute value of -32. Hint: Use the function `abs`.
9. Calculate $\pi.ln(\sqrt{42})$. Tip: Use the functions: `pi`, `log` and `sqrt` from the `numpy` package by loading it via the command: `import numpy as np`.

## 2.3 Variables

Instead of using the values directly as numbers in an expression they can be stored in a variable. This variable can then be used in an expression to replace the original number. This has the advantage that you can change the values used in the calculation without having to rewrite the expression.

```
a = 22/8
print(a)
```

```
2.75
```

Rules for naming variables:

1. Must be one word only (i.e. no spaces)
2. Must contain only letters, numbers and the underscore
3. Cannot start with a number

### 2.3.1 Integers and Floating-Point Numbers

The most common variable types are integers and floats (floating point numbers, that is decimals with a comma). In Python the decimal separator is the dot. The type of the variable can be confirmed using the `type()` function.

```
a = 5
print(type(a))
```

```
<class 'int'>
```

```
b = 3.14
print(type(b))
```

```
<class 'float'>
```

#### 2.3.1.1 Exercise

Compute the result of the following expressions by using variables in the Python language:

1. If $b = 3$ and $c = 4$ and $a = \sqrt{b^2 + 2.b.c + c^2}$, what is the value of $a$?
2. In the previous expression, what is the type of the variables $a$, $b$ and $c$?
3. What will be the value of the variable `a` if we first do `a=1` and then `a=a+1`?

### 2.3.2 Strings

Variables can contain text values. In this case the variable's type is called "string

```
a = "This is text"
print(a)
```

```
This is text
```

Two or more strings can be concatenated by "adding" them

```
a = "Eduardo"
b = "Luiza"
print(a + b)
```

```
EduardoLuiza
```

In addition they can also be repeated by "multiplying" them

```
c = "Thiago"
print(3*c)
```

```
ThiagoThiagoThiago
```

#### 2.3.2.1 Exercise

Perform the following operations by using variables of type `string` in the Python language.

1. Store the string `Hello World` in a variable.
2. Join the strings `Eduardo` and `Maria` into a single string.
3. Repeat the string `Carlos` five times.
4. Suppose `school = 'Harvard'`, what is the result of `print(school[2])`?
5. Determine the number of characters that make up the string "I know how to program in Python". Tip: Use the `len()` function.
6. The following string, in the Python language, will display an error message: `teslaQuote = 'I don't care that they stole my idea. I care that they don't have any of their own'`. How do we adjust it so that we don't have another error?
7. Suppose `a = 'Car'`, `b = 'Sports Car'`, what is the result of `a < b` and `len(a)<len(b)`?

### 2.3.3 Booleans

Boolean data types: The values that a variable of type Boolean can take are either `True` or `False` (in general we think of `True` as `1` and `False` as `0`). Please note the following: Although it is possible to use `True` as `1` and `False` as `0` the values that Python will store in the variable will be either `True` or `False`, and these values are NOT text. They are boolean. Booleans arise as the result of operations that evaluate the validity or otherwise of an expression. For example, the expression $3 < 4$ will result in True.

```
print(3 < 4)
```

```
True
```

Typical operations for two Boolean variables are `and` (&), `or` (|) and `not` (!). These are also known as logical operations.

An `and` (&) between two conditions will result in True only if both conditions are True. For example, the operation `(3<4) & (10<5)` will result in `True`.

```
print((3 < 4) & ( 10 < 15))
```

```
True
```

On the other hand, a `or` (|) will result in True if at least one of the conditions is True. In the following example, the operation `(3 < 4) | (10 > 17)` will result in `True`.

```
print((3 < 4) | (10 > 17))
```

```
True
```

A `not` (`!`) in turn reverses the value of the boolean (`True` becomes `False` and `False` becomes `True`).

The most common tests, which when performed between any two variables, give a boolean as the result are: equal `==`, greater than or equal to `>=`, less than or equal to `<=` or not equal to `!=`

#### 2.3.3.1 Exercise

Calculate the following results by using Boolean variables in the Python language:

1. `42 == '42'`, `42 == 42.0`, `42.0 == 0042.000`
2. Suppose `C=41`, calculate: a) `C == 40`, b) `C != 40 and C < 41`, c) `C != 40 or C < 41`, d) `not C == 40`, `not C > 40`, e) `C <= 41`

3. Calculate: (a) `not False`, (b) `True and False`, (c) `False or True`, (d) `False or False`, (e) `True and True and False`, (f) `False == 0`, (g) `True == 0`, (h) `True == 1`

4. Make `a=1/947*947.0` and `b=1`. Then calculate `a==b`. Is the result the one you expected? This exercise shows that the equality comparison between `floats` can lead to errors due to rounding performed by the machine when storing a comma value. To compare two floats it must be determined whether the absolute difference between them is smaller than an arbitrary small tolerance value chosen by the user.

### 2.3.4 Variable conversion

A numeric variable can be converted to text, and a string that only contains numbers can be converted to a number. To convert a number into a string use the function `str()`. To convert a string that only contains numbers into an integer use `int()`. If the numbers contain the decimal point, the variable can be converted to a number using `float()`.

```
a = "4"
print(a, type(a))
```

```
4 <class 'str'>
```

```
b = int(a)
print(b, type(b))
```

```
4 <class 'int'>
```

```
a = "3.14"
print(a, type(a))
```

```
3.14 <class 'str'>
```

```
b = float(a)
print(b, type(b))
```

```
3.14 <class 'float'>
```

## 2.4 Programs

A program is a set of commands that the computer executes in a sequential, step-by-step manner. An example can be seen below, where we have a program that presents itself and tells us the number of characters that make up a given name.

```
print("Hello World")
```

```
Hello World
```

```
myName = "Gustavo"

print("It was nice to meet you" + myName)
```

```
It was nice to meet youGustavo
```

```
print("Your name has " + str(len(myName)) + " characters")
```

```
Your name has 7 characters
```

The execution of a program is strictly sequential. Therefore, the order of the instructions is fundamental for the correct execution of the code. In the following example, an error was included on purpose. Determine the error, change the code, and run the program. What is the result?

```
try:
  C=A+B
  A=2
  B=3
  print(C)
except:
  print("Variables A and B not defined")
```

```
Variables A and B not defined
```

### 2.4.1 User Interaction

The program can interact with the user through the `input` function. The code below, if executed, will ask the user to enter a value, store it in the variable `a` and print a response message.

```
a = input("Enter a value")
print("The value you entered was: ", a)
```

### 2.4.2 Error Management

Another interesting feature is error handling. To prevent the program from stopping if an error occurs, you can use the pair `try` and `except` (i.e. exception handling). Note the use of the following pair:

```
a = 4
b = 0

try:
  c = a/b
except:
  c = "Error. Division by Zero."

print(c)
```

```
Error. Division by Zero.
```

Another example of using the error catch and avoidance features of the pair **try: and except:** would be to extract the digits of a string. For example extracting the numbers 2 and 3 from the string "2 cats and 3 dogs." To do this we use `try` to try to convert each element to integer and insert the result into the list. When it gives an error, we use `except: pass` to get out of the error and continue the loop.

```
a = "2 cats and 3 dogs"
list1 = []
for i in list(a):
  try:
    n = int(i)
    list1.append(n)
  except:
    pass

print(list1)
```

```
[2, 3]
```

### 2.4.3 Exercises

Develop the following programs in Python:

1. Write a program that asks the user to enter three numbers and then prints the average of the three.
2. Write a program that asks for the amount spent in a restaurant, the percentage of service, and the tip. The program should then print in four lines, the total of the meal, the service charge, the tip and the total of the bill.
3. Write a program that asks the user to enter: a) the first name, b) the last name, c) the number of hours worked in consulting, d) the hourly

rate paid by the customer, e) the percentage of the hourly rate that the employee receives as a salary, f) the number of dependents. The program should then calculate and print the employee's full name and total salary. Consider that each dependent adds 145 reais to the salary and that the employee's fixed salary is 2,800 reais.

4. Write a program that asks the user to enter the quantity, the price without taxes and the tax percentage (in the "inside"style) of each of two types of parts. The program should print the total sales price with taxes for each of the part types.

5. Write a program that asks the user to enter the number of sales made in the month, the average value of these sales, the commission percentage, the fixed commission value per sale and the fixed salary. The program should then calculate the employee's gross salary and print the result.

6. Write a program that chooses a number at random between 1 and 20. Use the `np.random.randint(1, 20)` function. The user will then have 7 attempts to guess the number. The program should ask the user to enter an attempt and if it is greater than the number it should indicate `Very high estimate`, otherwise it should indicate `Very high estimate`. If the user gets it right, the program should indicate `Correct Estimate` and then inform how many tries were necessary for the user to choose the right number. If the user doesn't get the right number in 7 tries, the program will stop the game and tell the user what was the correct number.

## 2.5 Conditional Deviation

The sequence of operations in a program can be changed through a conditional offset (also called a decision), which is implemented by the commands `if else`.

In the example below, we have a comparison with the value of the variable `name`. From the result of this comparison a segment of code will be executed or not. Other decisions (`ifs`) can be mixed together, as in this example.

```
name = "Maria"
password = "cod"

if name == "Maria":
  print("Hi Maria")
  if password == "cod":
    print("Access allowed.")
  else:
    print("Wrong password.")
```

```
Hi Maria
Access allowed.
```

In this next example a variant of the `if else` pair is presented, the `elif` which stands for `else if`. If the comparison in the main `if` results in a `False`, the program moves on to the `elif` where another comparison (decision) will be made, until we reach the final `else`.

```
name = 'Rafaella'
age = 9

if name == 'Rafaella':
  print('Hi, Rafaella.')
elif age < 12:
  print('You are not Rafaella, kid!')
elif age > 200:
  text1 = 'Unlike you, Rafaella is not eternal'
  text1 = text1 + ' immortal being'
  print(text1)
elif age > 100:
  print('Grandma, you are not Rafaella')
```

```
Hi, Rafaella.
```

### 2.5.1 Exercises

Develop the following programs by using conditional detour in the Python language:

1. Suppose the password for accessing a system is "BigGame", ask the user to enter a password. If the password entered is correct print "Access authorized", otherwise print "Access denied".

2. Create a program that receives two numbers `a` and `b`, and performs the following actions: (a) prints the larger of the two, (b) if `b` is greater than `a + 100` it prints `b is much larger than a`, (c) if both are even it should print `Both are even`, otherwise it should print "At least one number is odd", and d) if the two numbers are multiples of each other it should print the messages "The numbers are multiples of each other"or "The numbers are not multiples of each other".

3. Create a program that receives an integer and three real values, `a`, `b` and `c`. If the integer is negative the program should print the three values in descending order. If the integer is greater than 0 the program should print the three values in ascending order. If the integer is equal to zero the program should print the largest value in the middle of the other two, starting with the smallest value. In addition, if the difference between `a` and `b` is less than `c` in absolute terms the program should print numbers closer together than `c`, otherwise the program should print numbers further apart than `c`.

4. Create a program that asks the user if he wants to convert Celsius to Farenheit or the other way around. The user after indicating the desired conversion should enter a number and get as answer the temperature converted to the new scale.

5. Create a program that receives a number and prints out if a is greater than or equal to 18, prints out if a>=16 or otherwise.

6. Ask the user to enter his weight and height. Then calculate and print the $IMC$ value and category, where $IMC = weight/height^2$, and the healthy weight range. The $IMC$ category is defined by the following ranges: a) $<$ 16: Severe thinness, b) 16 to $<$ 17: Moderate thinness, c) 17 to $<$ 18.5: Mild thinness, d) 18.5 to $<$ 25: Healthy, e) 25 to $<$ 30: Overweight, f) 30 to $<$ 35: Grade I obesity, g) 35 to $<$ 40: Grade II (severe) obesity, h) $>=$ 40: Grade III (morbid) obesity

7. A year is leap, that is, it has 366 days, if it is divisible by 4. There are, however, exceptions: If it is divisible by 100 then it is not leap. However, if it is divisible by 400 it is leap. Write a program that determines whether a year is leap or not.

8. Write a program that asks the user to enter the start and end times of a game with the hours and minutes separated by colons. The program should then calculate the length of the game in hours and minutes. The game can start one day and end the next. The maximum duration is 2 hours and 30 minutes.

## 2.6 Repeat

### 2.6.1 Loops for

The most common type of change in the sequence of execution of a program is the loop. The simplest way to implement a loop is through the `for` command. In this case a piece of code will be repeated as many times as desired (i.e. as many times as is indicated in the loop definition). See below.

```
for i in range(12, 18, 2):
  print(i)
```

```
12
14
16
```

### 2.6.2 Loops while

Another type of flow control is the conditional repeat of an instruction. One way to implement a conditional loop is through the `while` command. The program exits the loop when the `while` test condition is no longer `True`.

```
spam = 0
while spam < 5:
  print('Hello everyone.')
  spam = spam + 1
```

```
Hello everyone.
Hello everyone.
Hello everyone.
Hello everyone.
Hello everyone.
```

### 2.6.3 Exit: break

The output of a loop can be forced using the `break` command

```
while True:
  print('Enter your name.')
  name = 'Gustavo'
  # name = input()
  if name == 'Gustavo':
    break
```

```
Enter your name.
```

```
print('Thank you!')
```

```
Thank you!
```

### 2.6.4 Return: continue

Just as the output can be forced with `break` the program can be made to execute the next operation of the loop by `continue`.

```
while True:
  print('What is your name?')
  name = 'Jack'
  #name = input()
  if name != 'Jack':
    continue
  text = 'Hi, Jack. What is the password?'
  text = text + '(Hint: A fish.)'
  print(text)
  #password = input()
  password = 'cod'
```

```
  if password == 'cod':
    break
```

```
What is your name?
Hi, Jack. What is the password?(Hint: A fish.)
```

```
print('Authorized Access')
```

```
Authorized Access
```

### 2.6.5 End: exit

Forced termination of a program (whether or not it is in a loop). The termination is done with the `exit` function from the `sys` package. Note how to use the function and load the package in the following example:

```
import sys

while True:
  print("Type exit to exit")
  response = "exit"
  # response = input()
  if response == "exit":
    sys.exit()
  print("You have entered " + response + ".")
```

### 2.6.6 Exercises

Solve the following exercises using the Python language's repetition features.

1. Print all numbers : a) from 0 to 9 (inclusive), every 2, b) in descending order, from 99 to 0, every 2, c) from the sequence: 8, 11, 14, 17, 20, . . . (d) from the sequence: 100, 98, 96, . . . 4, 2, and e) that belong to the Fibonacci series between 0 and 50.

2. Determine the numbers: a) that are divisible by 7 and multiples of 5 between 1500 and 2700, b) between 1000 and 3000 which when divided by 11 give 7 as a result, c) between 10 and 100 which are not divisible by 7. When a number divisible by 7 is found, the program should stop processing. Hint: use `break`.

3. Write a program that takes an integer positive number n and returns: a) the n-th term of the series: -1 1 7 9 15 17 23 25 31 33 39 etc..., b) the sum of the numbers from 1 to 10 (inclusive), c) the factorial of n, d) the multiplication table (from 1 to 10) for it.

4. Write a program that takes two values and returns all the odd ones between the lesser and the greater of them.

5. Write a program that asks the user to enter a string and a letter. Use a while loop to determine whether or not the letter is present in the string. If it is, print the position of the first occurrence of the letter in the string. Rework the program using a for loop and the break function.

6. Write a program that stores the password 'brazil2100' in a variable. The user when running the program should be asked to enter the password. If he makes the correct entry in one of the first three attempts, the program should print "Access authorized". Otherwise the program should print "Access denied".

7. Write a program that will play "Paper", "Rock", "Scissors"with the user. Who wins 3 rounds in a row will be the winner of the game.

8. Write a program that implements the following algorithm for calculating the square root of a number. Suppose you want to calculate the square root of 20. Start with 10. Calculate $(10 + 20/10)/2 = 6$. Use the previous result to calculate $(6 + 20/6)/2 = 14/3$. Apply the process to the fraction $14/3$ and calculate a new estimate for the root of 20. Stop the algorithm when the difference between two estimates is less than $10^{-10}$. The algorithm however should perform these calculations only with fractions, without the use of decimals, including the comparison of the two estimates that will stop the calculation.

## 2.7 List

Lists and tuples are sets of numbers. Lists are created with square brackets and tuples with parentheses. The main difference is that the elements of a tuple, once created, cannot be modified. In this text we will concentrate on the study of lists.

```
a = (1,2,3,4)
print("This is a tuple: ", a)
```

```
This is a tuple:  (1, 2, 3, 4)
```

```
b = [1,2,3,4]
print("This is a list1: ", b)
```

```
This is a list1:  [1, 2, 3, 4]
```

### 2.7.1 Access to elements

Access to the elements of a list is done by its index, which must be enclosed in square brackets after the list name. See below. Note that the numbering of addresses in Python starts at 0.

```
# First element of the list
a = [1, 2, 3]
a[0]
```

```
1
```

```
# Third value from list b
b = ["Albeto", "Bernardo", "Carlos", "Eugenio"]
b[2]
```

```
'Carlos'
```

```
# New value in the third element of the list b
b[2] = "Thiago"
b[2]
```

```
'Thiago'
```

```
# Data type of the third list element c
c = ["Good morning", 3.1415, True, None, 42]
type(c[2])
```

```
<class 'bool'>
```

An element at a given position can also be obtained using the `.index` method

```
# Position of the name Lucas in the list names
names = ["Richard", "James", "Luke", "Jane", \
         "Tereza", "Maria"]
names.index("Luke")+1
```

```
3
```

### 2.7.2 Lists of lists

The elements of a list can be another list. Accessing list elements within a list is done with double brackets, as seen below:

```
list1 = [["Carlos", "Bernardo"],[10,20,30,40,50]]

print(list1[0])
```

```
['Carlos', 'Bernardo']
```

```
print(list1[1])
```

```
[10, 20, 30, 40, 50]
```

```
print(list1[0][1])
```

```
Bernardo
```

```
print(list1[1][4])
```

```
50
```

### 2.7.3 Concatenation

To concatenate two lists you simply "sum" them (actually join them), as can be seen below:

```
[1, 2, 3] + ["A", "B", "C"]
```

```
[1, 2, 3, 'A', 'B', 'C']
```

### 2.7.4 Insertion

Insertion at the end of a list can be done by concatenation (we make the element a list by wrapping it in brackets and then concatenate this list into the original list)

```
list1 = ["A", "B", "C"]
element = ["D"]
list1 = list1 + element
list1
```

```
['A', 'B', 'C', 'D']
```

Another way to perform an element insertion at the end of a list is to use the `.append` method

```
list1 = ["A", "B", "C"]
list1.append("D")
list1
```

```
['A', 'B', 'C', 'D']
```

The insertion at a specific position is done by the `insert` method, described below: `nomedalist.insert(position, value)`. Suppose we want the following list of names at the end: `names = ['Richard', 'James', 'Luke', 'Joan', 'Tereza', 'Mary']`. However, at the moment the list names consists of the following elements: `names = ['Ricardo', 'Tiago', 'Tereza', 'Maria']`. We must then insert `Lucas` and `Joana` in the third and fourth positions of the final list. Remember that in python the numbering starts at 0, so the third position is the position with index equal to 2.

```
names = ["Ricardo", "Tiago", "Tereza", "Maria"]
print(names)
```

```
['Ricardo', 'Tiago', 'Tereza', 'Maria']
```

```
names.insert(2, "Lucas")
print(names)
```

```
['Ricardo', 'Tiago', 'Lucas', 'Tereza', 'Maria']
```

```
names.insert(3, "Jane")
print(names)
```

```
['Ricardo', 'Tiago', 'Lucas', 'Jane', 'Tereza', 'Maria']
```

### 2.7.5 Replication

A list can have its elements replicated `n` times by "multiplying" the list by n. Look at an example in the code below:

```
["X", "Y", "Z"] * 3
```

```
['X', 'Y', 'Z', 'X', 'Y', 'Z', 'X', 'Y', 'Z']
```

### 2.7.6 Delete

Deleting values, on the other hand, is done by the `del` command.

```
names = ["Ricardo", "Tiago", "Lucas", "Joana", \
         "Tereza", "Maria"]
print(names)
```

```
['Ricardo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']
```

```
del names[2]
print(names)
```

```
['Ricardo', 'Tiago', 'Joana', 'Tereza', 'Maria']
```

```
del names[2]
print(names)
```

```
['Ricardo', 'Tiago', 'Tereza', 'Maria']
```

### 2.7.7 Sorting

Finally, sorting the elements of a list can be done by the `.sort` method

```
# List of names sorted alphabetically
names = ["Richard", "James", "Luke", "Jane", \
         "Tereza", "Maria"]
names.sort()
names
```

```
['James', 'Jane', 'Luke', 'Maria', 'Richard', 'Tereza']
```

### 2.7.8 Slices

The length (number of elements) of the list is obtained with the `len` function.

```
list1 = ["A", "B", "C", "D", "E", "F", "G", "H", "I"]
n = len(list1)
print("The list has", n, "elements")
```

```
The list has 9 elements
```

As already mentioned, you can get a specific element from a list by looking at its address. We must remember that addresses in a list start at 0 and go to $n-1$ where $n$ is the number of elements in the list.

```
list1 = ["A", "B", "C", "D", "E", "F", "G", "H", "I"]
list1[0]
```

```
'A'
```

To get a portion of the elements of a list (a *slice* of the list) we pass an address range $i:j$. This will result in elements of index $i$ through $j-1$. For example, to get the elements of index 1 through 4 we pass `list1[1:5]`:

```
list1 = ["A", "B", "C", "D", "E", "F", "G", "H", "I"]
list1[1:5]
```

```
['B', 'C', 'D', 'E']
```

To get from the element of index 1 to the element of index 5, but every two positions we do:

```
list1 = ["A", "B", "C", "D", "E", "F", "G", "H", "I"]
list1[1:6:2]
```

```
['B', 'D', 'F']
```

If one of the indices is omitted it will be considered the index 0 or the index of the last element or if it is the increase step it will be considered 1. So, to get from the element of index 0 (beginning of the list) to the element of index 4 we can do:

```
list1 = ["A", "B", "C", "D", "E", "F", "G", "H", "I"]
list1[0:5], list1[:5]
```

```
(['A', 'B', 'C', 'D', 'E'], ['A', 'B', 'C', 'D', 'E'])
```

On the other hand, to get from the element of index 3 to the last element in the list we do:

```
list1 = ["A", "B", "C", "D", "E", "F", "G", "H", "I"]
list1[3:9]
```

```
['D', 'E', 'F', 'G', 'H', 'I']
```

or

```
list1[3:]
```

```
['D', 'E', 'F', 'G', 'H', 'I']
```

One of the most unique features in Python lists is the negative numbering system. To understand this system you have to imagine that each element in the list has two addresses. One is positive, starting from the first element (element of index 0) and moving from left to right. The other is negative, starting from the last element (element of index -1) and moving from right to left. To understand suppose the following list1:

```
list1 = ["A", "B", "C", "D", "E", "F"]
list1
```

```
['A', 'B', 'C', 'D', 'E', 'F']
```

Its elements will have the following indices:

| Element | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| Positive Index | 0 | 1 | 2 | 3 | 4 | 5 |
| Negative Index | -6 | -5 | -4 | -3 | -2 | -1 |

The indices are to be understood independently. What matters is the starting and ending element itself and how we want to go from one to the other, either from left to right (with positive step) or from right to left (with negative step).

For example, we can obtain the slice with the elements `[B, C, D]` in the following ways, all of which are obtained with positive stepping:

Using only the positive addresses

```
list1 = ["A", "B", "C", "D", "E", "F"]
list1[1:4:1]
```

```
['B', 'C', 'D']
```

Using only negative addresses

```
list1 = ["A", "B", "C", "D", "E", "F"]
list1[-5:-2:1]
```

```
['B', 'C', 'D']
```

Mixing the addresses, but always counting up

```
list1 = ["A", "B", "C", "D", "E", "F"]
list1[-5:4:1]
```

```
['B', 'C', 'D']
```

```
list1 = ["A", "B", "C", "D", "E", "F"]
list1[1:-2:1]
```

```
['B', 'C', 'D']
```

By counting descending we invert the sequence of elements. For example, we could have reversed the order of the list by doing:

```
list1 = ["A", "B", "C", "D", "E", "F"]
list1[5:-7:-1]
```

```
['F', 'E', 'D', 'C', 'B', 'A']
```

Or:

```
list1 = ["A", "B", "C", "D", "E", "F"]
list1[-1:-7:-1]
```

```
['F', 'E', 'D', 'C', 'B', 'A']
```

```
list1 = ["A", "B", "C", "D", "E", "F"]
list1[-1::-1]
```

```
['F', 'E', 'D', 'C', 'B', 'A']
```

```
list1 = ["A", "B", "C", "D", "E", "F"]
list1[::-1]
```

```
['F', 'E', 'D', 'C', 'B', 'A']
```

Or we could have obtained the slice [“D,” “C,” “B”] by counting down and passing the following addresses, start and end:

```
list1 = ["A", "B", "C", "D", "E", "F"]
list1[3:0:-1]
```

```
['D', 'C', 'B']
```

Since the addresses 0 and -1 represent distinct elements, if we want to go from the element of index 3 (in this case “D”) to the first element (in this case “A”) we can proceed as follows:

```
list1 = ["A", "B", "C", "D", "E", "F"]
list1[3::-1]
```

```
['D', 'C', 'B', 'A']
```

If we want to put a number between the two points, we cannot put -1 so that the count stops one earlier (at index 0). To do this we have to think that negative indexes continue to the right. Since the first element in this list has negative index -6, we must put -7 in the second position.

```
list1 = ["A", "B", "C", "D", "E", "F"]
list1[3:-7:-1]
```

```
['D', 'C', 'B', 'A']
```

Lest you forget, lists have two, independent numbering systems. Each element has two indices, one starting at 0 from the left (first element) and one starting at -1 from the right (last element).

### 2.7.9 Loops

To begin, notice the way the elements of a list are obtained when it is put into a Python loop. Each element is obtained as a variable. If we want to concatenate it into another list, it must be made into a list first (by enclosing it in square brackets). In the case of the `.append` method this is not necessary because .append assumes that an individual variable will be passed to be added to a list.

```
students1 = ["Richard", "James", "Luke"]
students2 = ["Joana", "Tereza", "Maria"]

names1 = []
names2 = []

#Insertion by concatenation
for name in students1:
  names1 = names1 + [name]
print(names1)

#Insertion by .append() method
```

```
['Richard', 'James', 'Luke']
```

```
for name in students2:
  names2.append(name)
print(names2)
```

```
['Joana', 'Tereza', 'Maria']
```

### 2.7.10 Comprehension

A list comprehension (*list comprehension*) is a way to access its elements one by one, just like in a loop. Take the following example:

```
names = ["Ricardo", "Tiago", "Lucas", "Joana", \
         "Tereza", "Maria"]
[str(name) for name in names]
```

```
['Ricardo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']
```

```
types = [str(name) for name in names]
for value in types:
  print(value)
```

```
Ricardo
Tiago
Lucas
Joana
Tereza
Maria
```

### 2.7.11 Enumeration

Another form is list enumeration. The difference here is that you can simultaneously control the index and the value of the list element being used in a loop. Notice the following:

```
list1 = ["A", "B", "C", "D", "E"]

for index, value in enumerate(list1):
  print(index, value)
```

```
0 A
1 B
2 C
3 D
4 E
```

### 2.7.12 Operator in

To determine whether or not an element is in the list we can use the `in` operator which returns `True` or `False` if the element is or is not in the list, respectively.

```
names = ["Ricardo", "Tiago", "Lucas", "Joana", \
         "Tereza", "Maria"]
student = "Tereza"

print(student in names)
```

```
True
```

### 2.7.13 zips of Lists

We can also group two lists together and run through them in parallel using a `zip`. A `zip` is a grouping of any objects, list or not. Below you can see the effect on a loop when creating a `zip` from two lists.

```
list1 = [10,20,30,40,50,60]
list2 = ["A", "B", "C", "D", "E", "F"]

for i,j in zip(list1,list2):
  print(i, j)
```

```
10 A
20 B
30 C
40 D
50 E
60 F
```

### 2.7.14 sets

Finally we have `set`s (sets). A set is used (for example) to find the unique values in a list.

```
list1 = [1,2,3,1,2,3,1,2,3,1,2,3]
set1 = set(list1)
print(set1)
```

```
{1, 2, 3}
```

Or a `set` can be used to determine the elements that belong to two lists. Suppose for example the lists x=[1,3,6,78,35,55] and y=[12,24,35,24,88,120,155,3]. The following program determines the common elements between `x` and `y`.

```
x=[1,3,6,78,35,55]
y=[12,24,35,24,88,120,155,3]

x1 = list(set(x))
y1 = list(set(y))

z = []
for i in x1:
  if i in y1:
    z.append(i)

print(z)
```

```
[3, 35]
```

The previous problem could easily be solved by using the intersection operator '\&, as we can see below:

```
x=[1,3,6,78,35,55]
y=[12,24,35,24,88,120,155,3]

x1 = set(x)
y1 = set(y)

#Intersection operator. Works only with "sets
z = x1 & y1
print(z)
```

```
{35, 3}
```

If we want to determine which elements belong to which list, we can use the union operator "|" (see below).

```
x=[1,3,6,78,35,55]
y=[12,24,35,24,88,120,155,3]

x1 = set(x)
y1 = set(y)

# Join operator. Only works with "sets".
z = x1 | y1
print(z)
```

```
{1, 3, 35, 6, 88, 12, 78, 55, 24, 155, 120}
```

### 2.7.15 Designation

If we make a = 4 and b = a, by changing the value of a, we will not change the value of b. This is called value passing. In this case we are dealing with scalar (dimensionless) variables. Notice the following.

```
a = 4
b = a
print(b)
```

```
4
```

```
a = 5
print(b)
```

```
4
```

With lists something different happens. The values of the elements are passed by reference. So if we create a list `names1 = ['Richard', 'James', 'Luke', 'Jane', 'Theresa', 'Mary']` and then make another list (`names2` for example) equal to `names1`, the change of an element in `names1` will be reflected in `names2`. This is called reference passing. Notice the following.

```
names1 = ["Richard", "James", "Luke", "Joan", \
          "Tereza", "Maria"]
names2 = names1
print(names2)
```

```
['Richard', 'James', 'Luke', 'Joan', 'Tereza', 'Maria']
```

```
names1[0] = "Gustavo"
print(names2)
```

```
['Gustavo', 'James', 'Luke', 'Joan', 'Tereza', 'Maria']
```

However, the name of the list itself is passed by value! This means that if we create a list `names1 = ['Richard', 'James', 'Luke', 'Jane', 'Theresa', 'Mary']` and then make `names2 = names1`, if we change the entire list, `names1` to another value set, `names2` will not change. Note the following:

```
names1 = ["Richard", "James", "Luke", "Jane", \
          "Tereza", "Maria"]
names2 = names1
print(names2)
```

```
['Richard', 'James', 'Luke', 'Jane', 'Tereza', 'Maria']
```

```
names1 = ['Andrew']
print(names2)
```

```
['Richard', 'James', 'Luke', 'Jane', 'Tereza', 'Maria']
```

If you want to create "unconnected" lists, that is, whose values have been passed by "value," you must use the function `copy` from the module of the same name. If you are copying a list that has other lists, use the function `deepcopy` (also from the module `copy`)

```
import copy as cp
names1 = ["Richard", "James", "Luke", "Jane", \
          "Tereza", "Maria"]

names2 = cp.copy(names1)
print(names2)
```

```
['Richard', 'James', 'Luke', 'Jane', 'Tereza', 'Maria']
```

```
names[1] = "Christine"
print(names2)
```

```
['Richard', 'James', 'Luke', 'Jane', 'Tereza', 'Maria']
```

This type of data passing where the elements of a list are passed by reference and the name of the list by value is called passing by assignment.

### 2.7.16 Exercises

Solve the following problems by using lists in the Python language:

1. Suppose the list `a=[-10,5,6,-1,-3,4,7,-12,9,-2]`. Determine how many elements of `a` are negative

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]
sum([i<0 for i in a])
```

```
5
```

2. Suppose the list `a=[-10,5,6,-1,-3,4,7,-12,9,-2]`. Create a list with the positions of the negative elements

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]
list1 = []
for i, value in enumerate(a):
  if value<0:
    list1.append(i)
print(list1)
```

```
[0, 3, 4, 7, 9]
```

3. Assume the list `a=[-10,5,6,-1,-3,4,7,-12,9,-2]`. Replace all negative values with 1 in the original list and present the resulting list.

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]
for i, value in enumerate(a):
  if value<0:
    a[i] = 1
print(a)
```

```
[1, 5, 6, 1, 1, 4, 7, 1, 9, 1]
```

4. Suppose the list `a=[-10,5,6,-1,-3,4,7,-12,9,-2]`. Determine the largest element in the list and its position.

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]
list1 = []
for i, value in enumerate(a):
  if value == max(a):
    list1.append(i)
print(max(a), list1)
```

```
9 [8]
```

5. Assume the list `a=[-10,5,6,-1,-3,4,7,-12,9,-2]`. sort its values in ascending order and present the new sorted list.

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]
a.sort()
print(a)
```

```
[-12, -10, -3, -2, -1, 4, 5, 6, 7, 9]
```

6. Assume the list `a=[-10,5,6,-1,-3,4,7,-12,9,-2]`. Invert the order of its elements and display the result.

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]
print(a[::-1])
```

```
[-2, 9, -12, 7, 4, -3, -1, 6, 5, -10]
```

7. Suppose the list `a=[-10,5,6,-1,-3,4,7,-12,9,-2]`. Swap the elements of even position with those of odd position. Note that in python the first element (therefore an odd position element) has index zero (therefore an even index)

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]

for indice, value in enumerate(a[:-1]):
  if indice % 2 == 0:
    a1 = a[index]
    a2 = a[index+1]
    a[index] = a2
    a[index+1] = a1

print(a)
```

```
[-10, 5, 6, -1, 4, -3, 7, -12, 9, -2]
```

8. Suppose the lists `a = [-10,5,6,-1,-3,4,7,-12,9,-2]` and `b = [-10,5,6,-1,-3,4,7,-12,9]`. Exchange the elements above the median position of the list with the elements below. For example if the list has 10 elements, the program should swap the first with the sixth, the second with the seventh and so on.

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]

n = len(a)

if n % 2 == 0:
  below = a[:int(n/2)]
  middle = []
  up = a[int(n/2):]
else:
  below = a[:int(n/2)]
  middle = a[int(n/2)]
  up = a[int(n/2)+1:]

a = up + middle + below
print(a)
```

```
[4, 7, -12, 9, -2, -10, 5, 6, -1, -3]
```

```
b = [-10,5,6,-1,-3,4,7,-12,9]

n = len(b)

if n % 2 == 0:
  below = b[:int(n/2)]
  middle = []
  up = b[int(n/2):]
else:
  below = b[:int(n/2)]
  middle = [b[int(n/2)]]
  up = b[int(n/2)+1:]

b = up + middle + below
print(b)
```

```
[4, 7, -12, 9, -3, -10, 5, 6, -1]
```

9. Suppose the list `a = [-10,5,6,-1,-3,4,7,-12,9,-2]`. Multiply all its elements by the number 3 and output the new list.

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]
print([i*3 for i in a])
```

```
[-30, 15, 18, -3, -9, 12, 21, -36, 27, -6]
```

10. Write a program that receives a list with 20 values. These values must first be sorted in ascending order. Then the program is to receive 10 more values. Each of these values should be inserted at the correct position in the list in order to keep the list sorted with increasing values. Present the resulting list of 30 numbers in ascending order at the end.
11. Write a program that reads a list with 20 elements and then calculates for each distinct value in the original list the number of times it appears repeated.

## 2.8 Functions

One of the most important programming concepts in Python is that of function. A function allows you to encapsulate a portion of code by avoiding the need to repeat commands every time an operation needs to be performed.

You can either use Python's native functions (such as `print`) or create your own. For example the following function takes a number as a parameter and returns a text value saying whether it is greater or less than 5. Note that the function does not print the result, it only passes a text value to the main program.

```
def greaterthan5(x):
  if x>5:
    a = "The number is greater than 5"
  else:
    a = "The number is less than or equal to 5"

  return(a)

c = 10
b = greaterthan5(c)
print(b)
```

```
The number is greater than 5
```

### 2.8.1 Scope

Functions create another important programming concept in Python which is the Scope of the variable. The scope is where the value of the variable is valid, and can be local (only in the function that created it) or global (valid for all functions in the environment). In the following example we will create a variable that will be global `x` and a variable that will be local `y`. Note that since `x` is created in the global scope, it "exists" in the scope inside the function.

```
def function1():
  y = 5
  print('This is the value of y in the local scope of function1:',y,"\n")
  print('This is the value of x in the local scope of function1:',x,"\n")
  return

x = 10
print('This is the value of x in the global scope:', x, "\n")
```

```
This is the value of x in the global scope: 10
```

```
function1()
```

```
This is the value of y in the local scope of function1: 5

This is the value of x in the local scope of function1: 10
```

However, if we try to print outside of `function1` the value of y (a variable that was created inside the scope of `function1`), we get an error. Notice below:

```
def function1():
  y = 5
  print('This is the value of y in the local scope of function1:',y,"\n")
  return

function1()
```

```
This is the value of y in the local scope of function1: 5
```

```
try:
  print('This is the value of y in the global scope:', y, "\n")
except:
  print('y does not exist in the global scope')
```

```
This is the value of y in the global scope: [12, 24, 35, 24, 88, 120, 155, 3
```

If you want a variable created inside a function to be accessible outside the function (in the `global` scope) it must be defined as such inside the function, as can be seen below. This however is not considered good programming practice and should be avoided whenever possible.

```
def function1():
  global y
  y = 5
  print('This is the value of y in the local scope of function1:')
```

```
  print(y)
  return

function1()
```

```
This is the value of y in the local scope of function1:
5
```

```
print('This is the value of y in the global scope:')
```

```
This is the value of y in the global scope:
```

```
print(y)
```

```
5
```

The most usual way to pass values from the global scope to the scope of a function is to use arguments in the function definition. Notice the following:

```
def function1(y):
  print('This is the value of y in the local scope of function1:')
  print(y)
  return

x = 10
function1(x)
```

```
This is the value of y in the local scope of function1:
10
```

Another important point is that if we have a variable x in the global scope and create another variable of the same name in the local scope of a function, the variable x in the global scope will not change.

```
def function1():
  x = 7
  print('This is the value of x in the local scope of function1:')
  print(x)
  return

x = 10
print('This is the value of x in the global scope,')
```

```
This is the value of x in the global scope,
```

```
print('Before executing function1:',x,"\n")
```

```
Before executing function1: 10
```

```
function1()
```

```
This is the value of x in the local scope of function1:
7
```

```
print('This is the value of x in the global scope,')
```

```
This is the value of x in the global scope,
```

```
print('After executing function1:',x, "\n")
```

```
After executing function1: 10
```

### 2.8.2 Designation, II

Remember that with lists a slightly different process occurs, called passing by designation. Recall that in passing by designation the name of the list is passed by value. So if we create a list, pass it to a function and inside the function change the value assigned to the name entirely this will have no effect on the global scope. See the following

```
list1 = [1,2,3,4,5]

def function1(value):
  print('value in local scope of function1:',value)
  value = 3
  print('value in local scope of function1:',value)

print('list in the global scope, before function1:')
```

```
list in the global scope, before function1:
```

```
print(list1)
```

```
[1, 2, 3, 4, 5]
```

```
function1(list1)
```

```
value in local scope of function1: [1, 2, 3, 4, 5]
value in local scope of function1: 3
```

```
print('list in the global scope, after function1:')
```

```
list in the global scope, after function1:
```

```
print(list1)
```

```
[1, 2, 3, 4, 5]
```

But let's remember that the elements of a list are passed by reference. So if instead of changing the value of the whole list in the function, we change one of its elements, this change will be reflected in the global scope. Notice that we will change the value of one of the elements of the argument passed as a parameter inside the function and this change will be reflected in the global scope.

```
list1 = [1,2,3,4,5]

def function1(value):
  print('value in the local scope of function1:',value)
  value[0] = -3
  print('value in the local scope of function1:',value)

print('list in global scope, before function1:')
```

```
list in global scope, before function1:
```

```
print(list1)
```

```
[1, 2, 3, 4, 5]
```

```
function1(list1)
```

```
value in the local scope of function1: [1, 2, 3, 4, 5]
value in the local scope of function1: [-3, 2, 3, 4, 5]
```

```
print('list in the global scope, after function1:')
```

```
list in the global scope, after function1:
```

```
print(list1)
```

```
[-3, 2, 3, 4, 5]
```

As noted earlier, this combination of passing the object itself by reference and its name by value is called passing by designation.

### 2.8.3 Exercises

Solve the following exercises using the user-defined functions feature of the Python language

1. Write a function that receives an integer and returns: a) the square of it, b) all cubes from 0 to n, c) its absolute value, d) its factorial, including 0. If a negative number, fractional number or string is given, a specific error message should be given in response.
2. Write a function that takes two numbers (a,b), returns the larger of them and one of the following messages: a) "a is the larger", b) "a equals b"or c) "b is the larger".
3. Write a function that converts the temperature in Celsius to Fahrenheit (C = 9/5F + 32)
4. Write a function that takes the area of a circle and returns the length of its circumference
5. Write a function that takes a number and determines whether it is equal when read from left to right or right to left.

## 2.9 Strings, II

The treatment of variables with text values (strings) is extremely sophisticated in the Python language, which has numerous functions, as we will see.

### 2.9.1 List Conversion

The most interesting feature is the ability to manipulate a string (`Python` for example) as a list whose elements are each of the original characters in the string.

```
name = "Gustavo"
print(name)
```

```
Gustavo
```

```
list1 = list(name)
print(list1)
```

```
['G', 'u', 's', 't', 'a', 'v', 'o']
```

### 2.9.2 Reconversion into String

Similarly, a list consisting of characters can be turned into a single text variable. To do this we use the `.join` method.

```
name = "Gustavo"
list1 = list(name)
print(list1)
```

```
['G', 'u', 's', 't', 'a', 'v', 'o']
```

```
new_name = "".join(list1)
print(new_name)
```

```
Gustavo
```

### 2.9.3 Case Detection

The `.islower` and `isupper` methods allow you to determine whether all characters in a string are formed by lowercase or uppercase.

```
name = "gustavo"
print(name.islower())
```

```
True
```

```
name = "GUSTAVO"
print(name.isupper())
```

```
True
```

### 2.9.4 Cash Conversion

Similarly the `.lower()\tthl{ and }.upper()` methods allow you to convert all characters in a string into their lower or upper case equivalents.

```
name = "gustavo"
print(name.upper())
```

```
GUSTAVO
```

```
name = "GUSTAVO"
print(name.lower())
```

```
gustavo
```

### 2.9.5 Exercises

Develop string manipulation functions in Python that:

1. Receives a string and changes all spaces to underscore. Use as an example string: `Artificial Intelligence and Machine Learning`.

```
name = "Artificial Intelligence and Machine Learning"
name = list(name)
for index, value in enumerate(name):
  if value == " ":
    name[index] = "_"

name = "".join(name)
print(name)
```

```
Artificial_Intelligence_and_Machine_Learning
```

2. Takes a string and replaces the periods with commas. Example: `tthl32,054.23. Expected result: `32,054.23`

3. Receives a string and: a) removes spaces and b) exchanges the letter `l` for `1`. Example: `Gustavo Correa Mirapalheta`. Expected result: `GustavoCorreaMirapa1heta`

4. Receive a string and change all characters present in `list1 = [' ','.','-',';']` to "_"and change the letter case to lowercase. Use as example string `Intelligence.Artificial e-Machine;Learning`.

```
name = 'Intelligence.Artificial e-Machine;Learning'
list1 = [" ",".","-",";"]

def exchange(name, list1):
  name = list(name.lower())
  for index, value in enumerate(name):
    if value in list1:
      name[index] = "_"
  name = "".join(name)
  return(name)

exchange(name, list1)
```

```
'intelligence_artificial_e_machine_learning'
```

5. Receives a string and exchanges all characters present in `list1 = [' ','.','-',';']` by the values of the same position in `list2 = ['_','_','_']` and passes their case to lowercase. Use as example string `Intelligence.Artificial e-Machine;Learning`

```
name = 'Intelligence.Artificial.e-Machine;Learning'
list1 = [" ",".","-",";"]
list2 = ["_","_","_","_"]

def exchange(name, list1, list2):
  name = list(name.lower())
  for index, value in enumerate(name):
    if value in list1:
      name[index] = list2[list1.index(value)]
  name = "".join(name)
  return(name)

print(exchange(name, list1, list2))
```

```
intelligence_artificial_e_machine_learning
```

6. Takes a string and changes all lowercase to uppercase and vice versa.
7. Takes a string and a number and returns all characters occurring in multiple positions of the number.
8. Takes a string and a character and returns all positions in the string where the character occurs.
9. Takes two strings and determines how many of the same characters at the same position are in both.
10. Takes two strings and determines whether they differ by only one character.

## 2.10 Dictionaries

A dictionary is a structure in which data appears in pairs called "key-value" (or "key-value"). There is no predetermined sequence of values in dictionaries (unlike lists). However we can list the set of keys, of values and of key-value pairs with the methods `.keys`, `values` and `items` respectively.

```
# Creation of the dictionary
a = {"Logic":8, "Finance":6, "Marketing":10, \
     "Mathematics":7, "Elective": "None"}
```

```
# Insert key-value pair
a["Immersion"]=True
```

```
# Value for the key "Logica"
a["Logic"]
```

```
8
```

```
# List of keys from the dictionary
for key in list(a.keys()):
  print(key)
```

```
Logic
Finance
Marketing
Mathematics
Elective
Immersion
```

```
# List of dictionary values
for value in list(a.values()):
  print(value)
```

```
8
6
10
7
None
True
```

```
# List of key,value pairs
for pair in list(a.items()):
  print(pair)
```

```
('Logic', 8)
('Finance', 6)
('Marketing', 10)
('Mathematics', 7)
('Elective', 'None')
('Immersion', True)
```

### 2.10.1 Value Search

Retrieving a value from its key in a dictionary can be done by the `.get()` method. The `get` method requires two values, the search `key` in the dictionary and the value that should be returned if the `key` is not found.

```
a = {"Logic":8, "Finance":6, "Marketing":10, \
     "Mathematics":7, "Elective": "None"}
a["Immersion"]=True
```

```
a.get("Finance",False)
```

```
6
```

```
a.get("Microeconomics",False)
```

```
False
```

The `setdefault` method allows you to assign a value to a given key if it does not already exist in the dictionary

```
a = {"Logic":8, "Finance":6, "Marketing":10, \
     "Mathematics":7, "Elective": "None"}
a["Immersion"]=True

a.setdefault("Microeconomics", True)
```

```
True
```

```
a.get("Microeconomics", 0)
```

```
True
```

An example of applying the setdefault() method can be seen below in counting the different characters in a text

```
message = 'It was a bright cold day in April, and '
message = message + 'the clocks were striking thirteen.'

count = {} #Create an empty dictionary

for character in message:
  a = count.setdefault(character, 0)
  count[character] = count[character] + 1

i = 1
for par in count.items():
  if i // 5 == i /5:
    print(par)
  else:
    print(par, end=" ")
  i = i+1
```

```
('I', 1) ('t', 6) (' ', 13) ('w', 2) ('a', 4)
('s', 3) ('b', 1) ('r', 5) ('i', 6) ('g', 2)
('h', 3) ('c', 3) ('o', 2) ('l', 3) ('d', 3)
('y', 1) ('n', 4) ('A', 1) ('p', 1) (',', 1)
('e', 5) ('k', 2) ('.', 1)
```

```
print("Cell executed ok")
```

```
Cell executed ok
```

### 2.10.2 Exercises

Solve the following exercises using dictionaries in the Python language.

1. Write a dictionary with the following key-value pairs: a) `'banana': 4`, b) `'apple': 2`, c) `'orange': 1.5`, d) `'pear': 3`

2. Suppose the dictionary: `'acoes= { 'BBDC4':'Bradesco', 'ITUB4':'Itau Unibanco', 'VALE3':'Vale do Rio Doce'}`. Add to the dictionary BBAS3, Banco do Brasil and delete from the dictionary BBDC4, Bradesco.

3. Write a code to get the maximum and minimum value of a dictionary. Assuming `myDict = {'x':500, 'w':5103 , 'y':5874, 'z': 560}`, the expected result would be `{'Max value': 5874, 'Min value' : 500}`

4. Write the following program in two ways: a) using loops and b) using dictionary comprehension. The program should receive a number n and from it produce a dictionary with all numbers from 1 to n (inclusive) as keys and n squared as values. At the end the dictionary should be printed. For example, if n is 8, the following dictionary should be printed: `{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}`. Tip: An empty dictionary can be created with `dictionary = {}`. Inserting a new key can be done with `dictionary[key]=value`

5. Write a code to create a dictionary with the count of each letter in a text. Suppose: text = `Fundacao Getulio Vargas`. Expected result: `' ': 2, 'F': 1, 'G': 1, 'V': 1, 'a': 3, 'd': 1, 'e': 1, 'g': 1, 'i': 1, 'l': 1, 'n': 1, 'o': 2, 'r': 1, 's': 1, 't': 1, 'u': 1, 'ã': 1, 'ç': 1, 'ú': 1`

6. Create a dictionary with the names of the months of the year as keys and the number of days in each month as values. From this dictionary create a function that checks only the first three letters of the month provided as parameter and returns the number of days in this month.

7. A simple encryption method is that of letter substitution. For example a can be replaced by e, b by a, and so on. The only care needed in this case is to keep the substitutions unique. Create a program that takes two strings

and determines if they can be used as the basis of a substitution code. For example `house` and `zero` would not work - we would not know how to transform the a (to and or the?) in either the outgoing or the incoming. If they can, they should be stored in a dictionary as a `key:value` pair.

8. In identifying patterns in DNA we start with known basic sequences such as: `L = ['AATAATAC', 'CATAATCA', 'AAATTCTA', 'AATACTAT', 'ACATATTA']` and we try to identify which basic sequence a sample with known and unknown values would fit into, for example AAAAA*. In this case this sample fits into the first and fourth sequences. One way to determine which basic sequences a sample fits into is to use a dictionary where the keys are the indexes with the known values in the basic sequences and the values are the characters. Create a program that uses a dictionary-like structure to determine which basic strings a user-supplied sample fits into.

9. Create a program that converts Roman numbers into decimal numbers. The basic conversion keys are: `M=1000, D=500, C=100, L=50, X=10, V=5, I=1`. Also include combinations like `IV=4, XL=40`, etc... Use a dictionary structure to form the `key:value` pairs.

10. Create a program in which the user enters a string and the computer lists all the words that can be formed with the letters of this string.

11. Set up a board and register system for a Tic Tac Toe

```
# Empty Board
print("Data Structure")
theBoard = {'top-L': " ", 'top-M': " ", 'top-R': " ",
            'mid-L': " ", 'mid-M': " ", 'mid-R': " ",
            'low-L': " ", 'low-M': " ", 'low-R': " "}
print(theBoard)
print()

print("Board empty")
def printBoard(board):
  print(board['top-L'] + '|' + board['top-M'] + '|' + board['top-R'])
  print('-+-+-')
  print(board['mid-L'] + '|' + board['mid-M'] + '|' + board['mid-R'])
  print('-+-+-')
  print(board['low-L'] + '|' + board['low-M'] + '|' + board['low-R'])

printBoard(theBoard)
print()

print("Board with a registered move")
theBoard = {'top-L': "O", 'top-M': "O", 'top-R': "O",
            'mid-L': "X", 'mid-M': "X", 'mid-R': " ",
            'low-L': " ", 'low-M': " ", 'low-R': "X"}
```

```
# Move Register
theBoard = {'top-L': " ", 'top-M': " ", 'top-R': " ",
            'mid-L': " ", 'mid-M': " ", 'mid-R': " ",
            'low-L': " ", 'low-M': " ", 'low-R': " "}

def printBoard(board):
  print(board['top-L'] + '|' + board['top-M'] + '|' + board['top-R'])
  print('-+-+-')
  print(board['mid-L'] + '|' + board['mid-M'] + '|' + board['mid-R'])
  print('-+-+-')
  print(board['low-L'] + '|' + board['low-M'] + '|' + board['low-R'])

turn = "X"

for i in range(9):
  printBoard(theBoard)
  print("Turn for " + turn + '. Move on which space?')
  move = input()
  theBoard[move] = turn
  if turn == "X":
    turn = "O"
  else:
    turn = "X"

printBoard(theBoard)
```

## 2.11 Classes

One of the most sophisticated programming features is that of Object Orientation. In it we create a `class` (family) of objects with specific characteristics. During the execution of the program we instantiate the class and create specific objects (variables) that are "born" with all the properties defined by the class.

To create a class we must pay attention to a number of essential rules: a) Start by defining the name of the class with the command `classNameName`, b) All the methods that the class will contain will be defined by `def NameOf Method(self, parameters):`, c) The first method will always be defined by: \tthl{def ___init___(self, parˆametros):}, d) A parameter (x for example) used in the creation of the class must be renamed to `self.x = x`, so that it can be used by the other methods of the class.

The following are examples of class creation.

1. Create a class that has two methods: a) `getString`: to get a string and b) `printString`: to print the string in uppercase

```
class uppercase:
  def __init__(self, text1="null text"):
    self.text1 = text1

  def getString(self,text2):
    self.text1 = text2

  def printString(self):
    print(self.text1.upper())

test = uppercase()
test.printString()
```

```
NULL TEXT
```

```
test.getString("This is a test")
test.printString()
```

```
THIS IS A TEST
```

2. Create a class named Rectangle, in which, when an object is instantiated, it will receive width and length values. The class must have an Area method, which will calculate the area of the object.

```
class Rectangle:
  def __init__(self, width, length):
    self.width = width
    self.length= length

  def Area(self):
    area = self.width * self.length
    return(area)

obj = Rectangle(4,3)
print(obj.Area())
```

```
12
```

3. Create a class, which when created is given the number N. When the `.generate` method is used, a number x must be passed in. The class will then return a list of all numbers between 0 and x (inclusive) that are divisible by N.

```
class divisibleByN:
  def __init__(self, N):
```

```
    self.N = N

  def generate(self, x):
    list1 = []
    for i in range(x+1):
      if i % self.N == 0:
        list1.append(i)
    return(list1)

g = divisibleByN(7)
print(g.generate(50))
```

```
[0, 7, 14, 21, 28, 35, 42, 49]
```

### 2.11.1 Exercises

1. Create a class called `Investment` with main and interest fields. There should be a method called `futureValue` that will return the value of the investment after n years with compound interest. There should also be a method `future_value` so that the printout of the object returns the principal and interest value of the object.

2. Create a class called `Product`. The class should have fields called `name`, `amount` and `price` which will indicate the name of the product, the number of items in stock and its list price. There should be a method `askPrice` which will receive the number of items to be purchased and return the purchase cost, when up to 10 items are purchased. A discount of 10

3. Write a class called `.ManagePassword`. The class should have a list called `AntiquePasswords` that will contain all the previous passwords of the user. The last item in the list is the user's current password. There should be a method called `.getPassword` that will return the current password and a method called `.setPassword` that will change the user's password. The method `.definePassword` should change the password only if the new password is different from all the user's previous passwords. Finally, there should also be a method called `.thisCorrect` that will take a string and return a boolean `True` or `False`, depending on whether the string is equal to the current password or not.

4. Write a class called Time, whose only field is time in seconds. It should have a method called `.convertsToMinutes` that takes a number of seconds and returns a sequence of minutes and seconds in the format minutes:seconds. There should also be a method called `.convertsToHours` which will take a number of seconds and return a sequence of hours, minutes, and seconds in hours:minutes:seconds format.

5. Write a class called `.PlayWord`. It should have a field that contains a list of words. The user of the class should pass the list of words they want to use with the class. There should be the following methods:
   (a) `.wordsExtension` - returns a list of all words with the specified number of characters in extension.
   (b) `.beginsWith(x)` - returns a list of all words that begin with x.
   (c) `.endsCom(x)` - returns a list of all words that end with x.
   (d) `.Palindromes()` - returns a list of all existing palindromes in the instantiation list of the class.
   (e) `.L-only(L)` - returns a list of words that contain only the letters in L
   (f) `.without(L)` - returns a list of words that do not contain any letters in L
6. Create a class to simulate moves of a card game with the following rules: each player receives half of the cards from a deck chosen at random. Each turn, each player presents the card at the top of his card pile. The player with the highest card receives both cards and places them at the bottom of the pile. If there is a tie, the two cards are eliminated from the game. The game ends when one player runs out of cards.
7. Write a class called `StonePaperScissors` that implements the logic of the game of the same name. In it the user plays against the computer for a certain number of rounds. The class must have a field for how many rounds the game will have, the number of the current round and the number of wins each player has. There should be methods for getting the computer's choice and for finding the winner of a round.

## 2.12 Recursion

Recursion makes it possible for a function to call itself. This feature provides elegant solutions to a wide range of problems.

Below we have the factorial of a number solved by recursion. To implement recursion, the function returns 1 if n is 1. If n is greater than 1 the function should calculate and return the result variable using the formula: `result = n * factorial(n-1)`

```
def factorial(n):
  if n == 1:
    return(1)
  else:
    result = n*factorial(n-1)
    return(result)
```

```
factorial(5)
```

```
120
```

### 2.12.1 Exercises

Solve the following exercises using recursion. Some of the exercises require the use of `numpy` arrays. In all of them you are asked to develop a function to:

1. Determine whether a number $m$ is prime.
2. Calculate $X^Y$
3. Calculate the term of order n of the Fibonacci series (1,1,2,3,5,8,13,21,34,...). Do not try to calculate terms above order 20, as the process can be very time consuming.
4. Sort the elements of a one-dimensional array.
5. Find a value $x$ in a two-dimensional array, where the value $x$ is to be found in the range of positions $i, k$.
6. Find $x$ in the address range $k$ to $n$ in the one-dimensional array $v$, returning its position.
7. Invert a one-dimensional array.
8. Get the smallest value in the $k$ to $l$ address range of a one-dimensional array
9. Add up the last $k$ elements of a one-dimensional array, returning the sum of them.
10. Take a square array $[n, n]$ and a number $k$. The function should return the original array with the values of the part with addresses greater than or equal to k transposed.
11. Implement the so-called Ackerman function. You are asked to test this function with small values of m and n. This function is defined as follows: a) $n + 1$ if $m = 0$, b) $A(m, n) = A(n - 1, 1)$ if $m \neq 0$ and $n = 0$, c) $A(m - 1, A(m, n - 1)$ if $m \neq 0$ and $n \neq 0$

## 2.13 General Exercises

1. Write a program that, from a sequence of comma-separated numbers coming from the keyboard as a single string, generates a list. Suppose the input string is `34,67,55,33,12,98`. Hint: To transform a string into a list by separating the elements by a comma use `list1 = variable.split(sep=',')`.

2. Using both the list comprehension feature and the `map` function, create a program that calculates and prints the integer part of the result of the following formula: $Q = \sqrt{\frac{2.C.D}{H}}$, where C is 50 and H is 30. D is the variable whose value should be passed to your program via a comma-separated sequence of numbers. For this exercise suppose that D = `100,150,180`. In this case the result will be `18,22,24`.

```
# Resolution by list comprehension
def calculate(D):
  C = 50
  H = 30
  x = int((2*C*D/H)**(0.5))
  return(x)

def separate(D):
  Di = D.split(',')
  Di = [int(i) for i in Di]
  return(Di)

def joins(D):
  cont = str(D[0])
  for i in D[1:]:
    cont = cont + ',' + str(i)
  return(cont)

D = separate('100,150,180')
R = [calculate(i) for i in D]
J = joins(R)
print(J)
```

```
18,22,24
```

```
# Resolution by map
def calc(D):
  C = 50
  H = 30
  x = int((2*C*D/H)**(0.5))
  return(x)

R0 = '100,150,180'
R1 = R0.split(',')
R2 = map(int,R1)
R3 = map(calc,R2)
R4 = map(str,R3)
R5 = ','.join(R4)
print(R5)
```

```
18,22,24
```

3. Write a program that takes as input a sequence of words separated by spaces and returns as output the words separated by spaces, without repetition and sorted alphabetically. For this exercise suppose that the input sequence is: `hello world and practice makes perfect and hello world again`. Hint: The command `set(list1)` returns a set with the unique values in the list. To make the result of set another list, do `list(set(list1))`. Use the `.split(',')` method on the string to separate the string by the comma, generating a list. Use `.sort()` on the list to sort its elements.

4. Using both regular functions (created with `def`) and `lambda` functions write a program that receives a sequence of four binary numbers, separated by a comma, and returns a sequence also separated by a comma, only of those that are divisible by 5. Assume for this exercise that the input sequence is: `0100,0011,1010,1001`. Hint: converting a string of numbers to integers in base 2 can be done with `int(string,2)`. Also remember that `map(function, list)` applies the function to each list element and returns the result as an object of type `map`. To make the result of map a list do: `list(map(function, list))`. A `lambda` function is a small function implemented on a single line. For example: `f = lambda x : x**2` implements a function named `f` that receives a number `x` and returns the square of `x`.

5. Using the functions lambda, filter and map create a program that will calculate the square of every odd number in a sequence of comma-separated numbers. Use the sequence of numbers 0 through 9 as input values for this exercise.

```
ehimpar = lambda x : x%2!=0

odd = list(filter(ehimpar, range(10)))

quad = lambda x : x**2

squares = list(map(quad, odd))

print(squares)
```

```
[1, 9, 25, 49, 81]
```

6. Write a program that will receive a string of strings representing Deposits (D) or Withdrawals (W) and the respective values of a checking account. Assuming that the starting balance of the account is 0, calculate the ending

balance of the account. For this exercise assume that the entry values are: `['D 100', 'W 200', 'D 300', 'D 300', 'W 200', 'D 100']`

7. Create a program to check the validity of a set of passwords. The program should accept a sequence of passwords separated by commas and print as answer only those that meet the following set of criteria: a) at least: a lower case letter, a number, an upper case letter and a character between `[$#]`, b) minimum length: 6 characters and c) maximum length: 12 characters. For testing, use as input values the string: 'ABd12341,a F1#,2w3E*,2We3345'. In this set, the only password that meets all the criteria is `ABd12341`

Solution: The program is shown below. To implement it, we used the package `re` that provides the use of `regex` (regular expressions) in Python.

```
def testpassword(x):
  import re
  criterion1 = not not re.search('[a-z]',x)
  criterion2 = not not re.search('[0-9]',x)
  criterion3 = not not re.search('[A-Z]',x)
  criterion4 = not not re.search("[$#@]",x)

  n = len(x)
  criterion5 = n ≥ 6
  criterion6 = n ≤ 12

  validity = criterion1 and criterion2 and criterion3
  validity = validity and criterion4 and criterion5
  validity = validity and criterion6

  return(validity)

passwords = 'ABd1234@1,a F1#,2w3E*,2We3345'
passwords = passwords.split(',')
result = filter(testpassword, passwords)
print(list(result))
```

```
['ABd1234@1']
```

8. Using a) just loops, b) the `sorted(list1)` function and c) the `sorted(list1)` and `lambda` functions, create a program to sort the tuples formed by the variables (name, age, score) in ascending order, where name is a string, age and score are numbers. Use as example values, the strings: a) `Tom,19,80`, b) `John,20,90`, c) `Jony,17,91`, d) `Jony,17,93` and e) `Json,21,85`.

Solution with loops only

```
# Original list
```

```
list1 = ['Tom,19,80', 'John,20,90', 'Jony,17,91']
list1 = list1 + ['Jony,17,93', 'Json,21,85']
for value in list1:
  print(value)
```

```
Tom,19,80
John,20,90
Jony,17,91
Jony,17,93
Json,21,85
```

```
# largestFirst element
def biggestL1(list1):
  L = 0
  for value in list1:
    x = value.split(',')
    n = len(x[0])
    if n>L:
      L=n
  return(L)

L = biggestL1(list1); L
```

```
4
```

```
# New list
def insert1C(list1, C=" ", L=0):
  for index, value in enumerate(list1):
    x = value.split(',')
    n = len(x[0])
    if L>n:
      x[0] = x[0] + (L-n)*C
    new_value = ','.join(x)
    list1.pop(index)
    list1.append(new_value)
  return(list1)

list2 = insert1C(list1, " ", L=4);
for value in list2:
  print(value)
```

```
John,20,90
Jony,17,93
Tom ,19,80
Json,21,85
Jony,17,91
```

```
# Sorted List
list2.sort()
for value in list2:
  print(value)
```

```
John ,20 ,90
Jony ,17 ,91
Jony ,17 ,93
Json ,21 ,85
Tom  ,19 ,80
```

Solution with `sorted(list1)`

```
#Original list
list1 = ['Tom ,19 ,80', 'John ,20 ,90', 'Jony ,17 ,93']
list1 = list1 + ['Jony ,17 ,91', 'Json ,21 ,85']

for value in list1:
  print(value)
```

```
Tom ,19 ,80
John ,20 ,90
Jony ,17 ,93
Jony ,17 ,91
Json ,21 ,85
```

```
# List with split elements
list2 = [value.split(',') for value in list1]
for value in list2:
  print(value)
```

```
['Tom ', '19 ', '80 ']
['John ', '20 ', '90 ']
['Jony ', '17 ', '93 ']
['Jony ', '17 ', '91 ']
['Json ', '21 ', '85 ']
```

```
# Sorted list
def generate_keys(value):
  key1 = value[0]
  key2 = value[1]
  key3 = value[2]
  return(key1, key2, key3)

list3 = sorted(list2, key = generate_keys)
```

```
for value in list3:
  print(value)
```

```
['John', '20', '90']
['Jony', '17', '91']
['Jony', '17', '93']
['Json', '21', '85']
['Tom', '19', '80']
```

Solution with `sorted(list1)` and `lambda` functions

```
#``Original list
list1 = ['Tom,19,80', 'John,20,90', 'Jony,17,93']
list1 = list1 + ['Jony,17,91', 'Json,21,85']
for value in list1:
  print(value)
```

```
Tom,19,80
John,20,90
Jony,17,93
Jony,17,91
Json,21,85
```

```
# Sorted list
list2 = sorted(list1, key = lambda x:(x[0],x[1],x[2]) )
for value in list2:
  print(value)
```

```
John,20,90
Jony,17,93
Jony,17,91
Json,21,85
Tom,19,80
```

9. A robot moves in a plane starting at the point $(0, 0)$ and following the directions UP, DOWN, LEFT and RIGHT, in the number of steps indicated by a number next to the direction. Create a program that takes an arbitrary sequence of moves and calculates the final distance between the point (0,0) and the robot's stop position. If the result is a number with a comma, display the nearest integer. For this example, consider the sequence: `['UP 5', 'DOWN 3', 'LEFT 3', 'RIGHT 2']`

10. Write a program to calculate the frequency of words in a text. The results should be presented in alphabetical order. Use as input the string: `New to Python or choosing between Python 2 and Python 3? Read books on Python 2 and Python 3 in both cases.`. Tip: To count the number of times

a character appears in a character list use the `list.count(character)` method

```
text1 = "Beginner in Python or choosing between Python 2 "
text1 = text1 + "what about Python 3? Read books on Python 2 and "
text1 = text1 + "Python 3 in both cases."

list1 = list(text1)
characters = sorted(list(set(text1)))
dictionary1 = {}

a = [dictionary1.update({i:list1.count(i)}) for i in characters]

i = 0
for entry in dictionary1.items():
  i = i + 1
  if i//5 == i/5:
    print(entry, "\n")
  else:
    print(entry, end=" ")
```

```
(' ', 22) ('.', 1) ('2', 2) ('3', 2) ('?', 1)

('B', 1) ('P', 5) ('R', 1) ('a', 5) ('b', 4)

('c', 2) ('d', 2) ('e', 7) ('g', 2) ('h', 8)

('i', 4) ('k', 1) ('n', 13) ('o', 13) ('r', 2)

('s', 4) ('t', 9) ('u', 1) ('w', 2) ('y', 5)
```

11. Given a list `[12,24,35,24,88,120,155,88,120,155]`, write a program that removes all duplicate values and prints the resulting list, with the values in the original order. Tip: To invert the order of elements in a list, use the function `reversed(list1)`

```
x = [12,24,35,24,88,120,155,88,120,155,88,-10]

def limpadup(list1):
  list2 = list(reversed(list1))

  for value in list2:
    cont_val = list2.count(value)
    if cont_val > 1:
      list2.remove(value)

  list3 = list(reversed(list2))
```

```
  return(list3)

print(limpadup(x))
```

```
[12, 24, 35, 88, 120, 155, -10]
```

12. Given an integer N, print a Rangoli alphabet of size N. Rangoli alphabets are an Indian art form, based on creating parameters. Two examples of Rangoli alphabets are given below:

Of size 3

```
# n + (n-1)*3 = 4n-3
# 5 + (5-1)*3 = 17
# 3 + (3-1)*3 = 9

def line(n):
  l = 4*n-3
  x = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m"]
  x = x + ["n", "o", "p", "q", "r", "s", "t", "u", "v", "y", "x", "w", "z"]
  list2 = x[:n][::-1]

  def draws(i):
    list1 = list2[:i]
    if i > 1:
      list1 = list1 + list(reversed(list2[:i-1]))
    text1 = '-'.join(list1)
    n_ = int((l-len(text1))/2)
    text2 = '-'*n_
    print(text2+text1+text2)

  for i in range(1,n+1):
    draws(i)

  for i in range(n-1,0,-1):
    draws(i)

line(3)
```

```
----c----
--c-b-c--
c-b-a-b-c
--c-b-c--
----c----
```

of size 5

```
# n + (n-1)*3 = 4n-3
# 5 + (5-1)*3 = 17
# 3 + (3-1)*3 = 9

def line(n):
  l = 4*n-3
  x = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m"]
  x = x + ["n", "o", "p", "q", "r", "s", "t", "u", "v", "y", "x", "w", "z"]
  list2 = x[:n][::-1]

  def draws(i):
    list1 = list2[:i]
    if i > 1:
      list1 = list1 + list(reversed(list2[:i-1]))
    text1 = '-'.join(list1)
    n_ = int((l-len(text1))/2)
    text2 = '-'*n_
    print(text2+text1+text2)

  for i in range(1,n+1):
    draws(i)

  for i in range(n-1,0,-1):
    draws(i)

line(5)
```

```
--------e--------
------e-d-e------
----e-d-c-d-e----
--e-d-c-b-c-d-e--
e-d-c-b-a-b-c-d-e
--e-d-c-b-c-d-e--
----e-d-c-d-e----
------e-d-e------
--------e--------
```

```
# n + (n-1)*3 = 4n-3
# 5 + (5-1)*3 = 17
# 3 + (3-1)*3 = 9

def line(n):
  l = 4*n-3
  x = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m"]
  x = x + ["n", "o", "p", "q", "r", "s", "t", "u", "v", "y", "x", "w", "z"]
```

```
  list2 = x[:n][::-1]

  def draws(i):
    list1 = list2[:i]
    if i > 1:
      list1 = list1 + list(reversed(list2[:i-1]))
    text1 = '-'.join(list1)
    n_ = int((l-len(text1))/2)
    text2 = '-'*n_
    print(text2+text1+text2)

  for i in range(1,n+1):
    draws(i)

  for i in range(n-1,0,-1):
    draws(i)

line(7)
```

```
------------g------------
----------g-f-g----------
--------g-f-e-f-g--------
------g-f-e-d-e-f-g------
----g-f-e-d-c-d-e-f-g----
--g-f-e-d-c-b-c-d-e-f-g--
g-f-e-d-c-b-a-b-c-d-e-f-g
--g-f-e-d-c-b-c-d-e-f-g--
----g-f-e-d-c-d-e-f-g----
------g-f-e-d-e-f-g------
--------g-f-e-f-g--------
----------g-f-g----------
------------g------------
```

13. Using only the `numpy` random number generator, `np.randint`, write a program to generate a random number between 10 and 150 (inclusive), which must be divisible by 5 and 7.

```
import numpy as np

list1 = []
for i in range(10,150+1,1):
  if ((i%5==0) and (i%7==0)):
    list1.append(i)

n = len(list1)
index = np.random.randint(n)
```

```
print(list1[index])
```

```
35
```

14. Using understanding of list and the use of the `np.random.choice` function of `numpy` write a program to generate a random number between 10 and 150 (inclusive), which must be divisible by 5 and 7.

```
import numpy as np

def testadiv(x):
  result = ((x%5==0) and (x%7==0))
  return(result)

list1 = [i for i in range(10,150+1) if testadiv(i)]
print('The list of elements eh = ',list1)
```

```
The list of elements eh =  [35, 70, 105, 140]
```

```
draw = np.random.choice(list1)
print('The random choice was = ', draw)
```

```
The random choice was =  70
```

15. Using understanding of list and the use of the `np.random.choice` function of `numpy` write a program, on a single line, to generate a random number between 10 and 150 (inclusive), which must be divisible by 5 and 7.

```
print(np.random.choice([i for i in range(10,150+1) if i % 35 == 0]))
```

```
35
```

16. A number is called a perfect number if it is equal to the sum of all its divisors, not including the number itself. For example, 6 is a perfect number because the divisors of 6 are 1, 2, 3, 6, and $6 = 1 + 2 + 3$. As another example, 28 is a perfect number because its divisors are 1, 2, 4, 7, 14, 28 and $28 = 1 + 2 + 4 + 7 + 14$. However, 15 is not a perfect number because its divisors are 1, 3, 5, 15 and $15\ 6 = 1 + 3 + 5$. Write a program that finds all four perfect numbers that are less than 10000.

17. A pallidomic number is a number that is the same if read from left to right and from right to left. If we add a number to its reverse side, it is sometimes possible to generate a pallindrome. For example $241 + 142 = 383$. The process can be repeated until we reach the desired result, for example $84 + 48 = 132$. $132 + 231 = 363$. Write a program that will generate palindromic numbers by applying the process described here in up to twenty repetitions of the process.

18. Write a program to determine with how many zeros the factorial of 1000 ends.

19. Implement a word search in a dictionary using the tree search method. In this method you are given a word to search for in a list and a list of words in alphabetical order. Start by determining whether the word you are looking for is the middle word in the list. If it is, return ok. If it isn't, determine whether the word you are looking for is to the left or right of the middle word in the list. Suppose it is in the left half. Then search for the word in the middle to the part where it will be and compare it with the word you have. If it is the word you are looking for, return ok. Otherwise repeat the process until you have two words in sequence in the original list. If neither of them is the word you are looking for, return not found.

20. Write a function that takes an integer number up to the trillionths and returns the number in full (1243 should return one thousand, two hundred and forty-three)

21. Create a function that receives a list of numbers and a number. Then it should determine the number in the list which, being smaller than the number passed as argument, is closest to it.

# Capítulo 3

# Linear Algebra

Linear Algebra is the part of mathematics that deals with matrices and vectors. Recently its importance in the teaching and study of mathematics has expanded, since computers preferentially use matrix structures to perform highly complex calculations. In the case of the Python language the `numpy` package provides a wide range of features, making it an excellent teaching tool. The following are theory and exercises in Linear Algebra using the extensive features of `numpy`.

## 3.1 Matrices

A matrix is a table of numbers. For our purposes a matrix is an array `numpy` with only two axes. Each axis can contain $n$ dimensions. For example the following matrix is a 2-axis object, with the first axis (the rows) consisting of 2 dimensions and the second axis (the columns) consisting of 3 dimensions. For reasons that will be explained throughout the text, you should avoid using the term "Dimension" instead of "Axis."

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

Next we see how to create the above array as a `numpy` array:

```
import numpy as np
A = np.array([[1,2,3],[2,1,4]])
A
```

```
array([[1, 2, 3],
       [2, 1, 4]])
```

## 3.2 Matrix Transpose

The transpose of a matrix is obtained by exchanging the values of its rows for those of its columns and is denoted as $A^T$. So the transpose of a matrix $[3, 2]$ will be a matrix $[2, 3]$. In python it can be calculated by the `.T` method.

```
import numpy as np
A = np.array([[1,2,3],[2,1,4]])
A.T
```

```
array([[1, 2],
       [2, 1],
       [3, 4]])
```

## 3.3 Matrix Sum

In order for two matrices $A$ and $B$ to be summed they must have the same size. The sum is done by adding the elements of the same position in each array. In the following code we sum the array $A$ with itself.

```
import numpy as np
A = np.array([[1,2,3],[2,1,4]])
A + A
```

```
array([[2, 4, 6],
       [4, 2, 8]])
```

## 3.4 Matrix Product

The matrix product $A.B$ is calculated by multiplying the value of each row in $A$ by the value of each column in $B$. Thus, in order for it to be calculated, the first matrix must have a number of columns equal to the number of rows in the second matrix. Furthermore, the resulting matrix will have the number of lines of the first matrix and the number of columns of the second matrix. Next the matrix $A^T.A$ is calculated whichs will be a $[3, 3]$ matrix, since $A^T$ is a $[3, 2]$ matrix and $A$ is a $[2, 3]$ matrix. In `numpy` the product of two matrices is calculated using the `.dot` method

```
import numpy as np
A = np.array([[1,2,3],[2,1,4]])
A.T.dot(A)
```

```
array([[ 5,  4, 11],
       [ 4,  5, 10],
       [11, 10, 25]])
```

An important observation. The matrix product is not commutative. This can be easily seen by calculating the matrix products $A^T.A$ and $A.A^T$ which produce different results.

```
import numpy as np
A = np.array([[1,2,3],[2,1,4]])
A.dot(A.T)
```

```
array([[14, 16],
       [16, 21]])
```

## 3.5 Identity matrix

In multiplication we have the number 1 as the neutral element. This means that if we multiply any number by the number 1, we will get as a result the original number itself. In linear algebra there is the concept of Identity Matrix. The identity matrix is such that $A.I = I.A = A$.

For matrices $[3, 3]$ the identity matrix has the form:

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

To generate an identity matrix we use the function `np.eye`, indicating as parameter the desired size. Below we generate an identity matrix $[3, 3]$

```
import numpy as np
np.eye(3)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Below we show that multiplying any matrix by the identity matrix produces a matrix equal to the original matrix.

```
import numpy as np
A = np.array([[1,2,3],[2,1,4],[3,4,1]])
I3 = np.eye(3)
A.dot(I3)
```

```
array([[1., 2., 3.],
       [2., 1., 4.],
       [3., 4., 1.]])
```

## 3.6 Symmetric Matrix

A matrix is considered symmetric when it is equal to its transpose. Prove that the matrix $K_4$ (below) is a symmetric matrix.

$$K_4 = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

To confirm that the matrices are equal we will create a boolean array (via `==`) and then check that all elements of the same position are equal (via the `.all` method)

```
import numpy as np
K4 = np.array([[2,-1,0,0],[-1,2,-1,0],[0,-1,2,-1],[0,0,-1,2]])
(K4 == K4.T).all()
```

```
True
```

## 3.7 Inversion of Matrices

In number multiplication, if we multiply a number (3 for example) by its inverse ($\frac{1}{3}$, also represented by $3^{-1}$ we get as a result the neutral element, the number 1. In matrix algebra we have a similar concept, for square matrices. We say that a matrix $A^{-1}$ is the inverse of the matrix $A$ if $A^{-1}.A = A.A^{-1} = I$, that is: if we multiply a matrix by its inverse we get as result the identity matrix. In `numpy` the inversion of matrices is obtained with the function `np.linalg.inv`.

```
import numpy as np
A = np.array([[1,2,3],[2,1,4],[3,4,1]])
A
```

```
array([[1, 2, 3],
       [2, 1, 4],
       [3, 4, 1]])
```

```
Ainv = np.linalg.inv(A)
Ainv
```

```
array([[-0.75,  0.5 ,  0.25],
       [ 0.5 , -0.4 ,  0.1 ],
       [ 0.25,  0.1 , -0.15]])
```

We confirm that the matrix `Ainv` is the inverse of the matrix `A` by calculating the product of the two and noting that it equals the identity matrix $[2, 2]$ (within an appropriate numerical margin of error)

```
np.round(A.dot(Ainv),2)
```

```
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0., -0.,  1.]])
```

## 3.8 Resolution of linear systems

Given a given linear system, $A.\vec{x} = \vec{B}$ one can obtain the vector $\vec{x}$ of the solutions $x_1$, $x_2$, $x_3$, ...$x_n$ for this system by performing the operation $\vec{x} = A^{-1}.B$ as can be seen in the following listing. As an example, let's solve the system of equations 3.1:

$$\begin{array}{rrrcr} x & +2y & -z & = & 2 \\ 2x & -y & +3z & = & 9 \\ 3x & +3y & -2z & = & 3 \end{array} \tag{3.1}$$

through `numpy`

```
import numpy as np
A = np.array([[1, 2, -1],[2, -1, 3],[3, 3, -2]])
A
```

```
array([[ 1,  2, -1],
       [ 2, -1,  3],
       [ 3,  3, -2]])
```

```
B = np.array([[2, 9, 3]]).T
B
```

```
array([[2],
       [9],
       [3]])
```

```
x = np.linalg.inv(A).dot(B)
x
```

```
array([[1.],
       [2.],
       [3.]])
```

We confirm that the solution found is correct by showing that $A.\vec{x} = B$

```
A.dot(x)
```

```
array([[2.],
       [9.],
       [3.]])
```

To solve indeterminate linear systems the most practical way is to use `sympy`. We can with this package solve homogeneous and non-homogeneous indeterminate systems. For example, the system of equations 3.2, which contains more variables than equations:

$$\begin{aligned} 2x_1 + 3x_2 - 5x_3 &= -10 \\ x_1 - 3x_2 + 8x_3 &= +85 \end{aligned} \tag{3.2}$$

To do this we proceed as follows:

```
import sympy as sp
x1, x2, x3 = sp.symbols("x1 x2 x3")
Eq1 = 2*x1 + 3*x2 -5*x3 + 10
Eq2 = x1 - 3*x2 +8*x3 - 85
sp.solve([Eq1, Eq2],[x1, x2])
```

```
{x1: 25 - x3, x2: 7*x3/3 - 20}
```

## 3.9 Complex numbers

The main advantage of the `numpy` functions is that they allow the insertion and calculation of matrices with complex numbers. This feature is essential when we are working with Quantum Computation exercises. The use of complex numbers in Python (and especially in `numpy`) is done transparently, with complex numbers being treated as any real.

For example the imaginary number $j = \sqrt{-1}$ is represented in Python with `1j`. A real with representation in the complex numbers is created by adding to the number itself the `value 0j`. This value will be of type `complex`.

```
a = -1 + 0j
a, type(a)
```

```
((-1+0j), <class 'complex'>)
```

Below we calculate $sqrt{-1} = j$, plus $j^3$ and $j^{15}$

```
import numpy as np
np.sqrt(-1+0j), (1j)**3, (1j)**15
```

```
(1j, (-0-1j), (-0-1j))
```

The real, imaginary, conjugate, modulus and angle parts of a complex number are obtained using the methods `.real`, `.imag`, `.conjugate` and the functions `np.abs` and `np.angle`. The following calculates such values for the number $c = 3 + 4j$

```
import numpy as np
c = 3+4j
c, c.real, c.imag, c.conjugate(), np.abs(c), np.angle(c)
```

```
((3+4j), 3.0, 4.0, (3-4j), 5.0, 0.9272952180016122)
```

## 3.10 Hermitian matrix

The Hermitian matrix is obtained by transposing the elements of the original matrix and transforming the values into their conjugate complexes (i.e. reversing the sign of the imaginary part). In the code below we create a matrix A with complex values and multiply it by its corresponding Hermitian matrix.

```
import numpy as np
A = np.array([[1+1j, 1-2j],[2+1j, 2-1j]])
A
```

```
array([[1.+1.j, 1.-2.j],
       [2.+1.j, 2.-1.j]])
```

```
Ah = A.conj().T
Ah
```

```
array([[1.-1.j, 2.-1.j],
       [1.+2.j, 2.+1.j]])
```

```
A.dot(Ah)
```

```
array([[ 7.+0.j,  7.-2.j],
       [ 7.+2.j, 10.+0.j]])
```

## 3.11 Roots of a Complex

For example, the cubic roots of $c = (1 + j)$ can be expressed as the solution of the equation $x^3 = 1 + j$. In other words we can think of it as wanting to find the roots of the polynomial $x^3 + 0.x^2 + 0.x - (1 + j)$. In numpy, the roots of a polynomial can be calculated from its coefficients by the function `np.roots`.

```
import numpy as np
# We create a list with the coefficients of the polynomial
coef = [1,0,0,-(1+1j)]

# Pass this list on to np.roots
np.round(np.roots(coef),3)
```

```
array([-0.794+0.794j, -0.291-1.084j,  1.084+0.291j])
```

Python also has a symbolic math module (called `sympy`). When we are interested in analyzing a problem from an algebraic point of view, this module provides interesting features. In the following example we will prove that, given two complex numbers $c_1$ and $c_2$, we have the following relationship for their modules: $|c_1||c_2| \geq |c_1.c_2|$

```
import sympy as sp
a, b, c, d = sp.symbols("a b c d", real=True)
c1 = a + b*1j
c2 = c + d*1j
sp.Abs(c1)*sp.Abs(c2)
```

```
sqrt(a**2 + 1.0*b**2)*sqrt(c**2 + 1.0*d**2)
```

The above expression is clearly the same as the expression below.

```
sp.Abs(c1*c2)
```

```
sqrt(a**2 + 1.0*b**2)*sqrt(c**2 + 1.0*d**2)
```

## 3.12 Unitary Matrices

A square matrix composed of complex numbers is unitary if its conjugate transpose $U^*$ is also its inverse. Next we will prove that the matrix

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{j}{\sqrt{2}} & \frac{j}{\sqrt{2}} & 0 \\ 0 & 0 & j \end{bmatrix}$$

is unitary.

```
import numpy as np
U = np.array([[1/np.sqrt(2), 1/np.sqrt(2),0],
              [-1j/np.sqrt(2),1j/np.sqrt(2),0],
              [0,0,1j]])
Ud = U.conj().T
np.round(Ud,3)
```

```
array([[ 0.707-0.j   , -0.   +0.707j,  0.   -0.j   ],
       [ 0.707-0.j   ,  0.   -0.707j,  0.   -0.j   ],
       [ 0.   -0.j   ,  0.   -0.j   ,  0.   -1.j   ]])
```

## 3.13 Hadamard product

The Hadarmd product of two matrices is calculated by doing the multiplication element by element. By `numpy` this operation is done by the operator `*` Next we calculate the Hadamard product of the matrix

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{j}{\sqrt{2}} & \frac{j}{\sqrt{2}} & 0 \\ 0 & 0 & j \end{bmatrix}$$

with itself.

```
import numpy as np
U = np.array([[1/np.sqrt(2), 1/np.sqrt(2),0],
              [-1j/np.sqrt(2),1j/np.sqrt(2),0],
              [0,0,1j]])
Uhad = U*U
np.round(Uhad,3)
```

```
array([[ 0.5+0.j,  0.5+0.j,  0. +0.j],
       [-0.5+0.j, -0.5+0.j,  0. +0.j],
       [ 0. +0.j,  0. +0.j, -1. +0.j]])
```

## 3.14 Tensor Product

The tensor product on `numpy` is calculated by the function `np.kron`. Next we calculate the tensor product of the matrices

$$A = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \text{ and } C = \begin{bmatrix} 3 \\ 4 \end{bmatrix} \tag{3.3}$$

```
import numpy as np
a = np.array([[1],[2]])
c = np.array([[3],[4]])
np.kron(a,c)
```

```
array([[3],
       [4],
       [6],
       [8]])
```

And then we calculate the tensor product of the matrices

$$X = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix} \text{ and } Y = \begin{bmatrix} 0 & -j \\ j & 0 \end{bmatrix} \tag{3.4}$$

```
import numpy as np
X = np.array([[0,1],[1,0]])
Y = np.array([[0,-1j],[1j,0]])
np.kron(X,Y)
```

```
array([[0.+0.j, 0.-0.j, 0.+0.j, 0.-1.j],
       [0.+0.j, 0.+0.j, 0.+1.j, 0.+0.j],
       [0.+0.j, 0.-1.j, 0.+0.j, 0.-0.j],
       [0.+1.j, 0.+0.j, 0.+0.j, 0.+0.j]])
```

## 3.15 Autovalues and eigenvalues

We can divide Linear Algebra into two major areas : the problems of systems of equations and the problems of eigenvalues and eigenvectors. The basic idea of an eigenvalue and an eigenvector may seem abstract at first glance, however few ideas have had a number of applications in physics, engineering, statistics and mathematics itself like the concept of eigenvalue and eigenvector.

In its conception the idea of an eigenvalue and eigenvector comes from the solution of the following problem : knowing that the product of a square matrix $\mathbf{A}$ by a vector $text\vec{b}fX$ results in another vector $text\vec{b}fZ$, find the vector $text\vec{b}fX$ such that the product $\mathbf{A}\{vec\mathbf{X}$ is parallel to $\vec{\mathbf{X}}$ itself, which in other words means : $\mathbf{A}.\vec{\mathbf{X}} = \lambda.\vec{\mathbf{X}}$. The vector that satisfies this relation is called the *eigenvector* of the matrix $\mathbf{A}$ and the proportionality factor $\lambda$ is called the *autovalue* of the matrix $\mathbf{A}$, associated to the eigenvector $text\vec{b}fX$

### 3.15.1 Geometric representation

{

Before solving the problem and presenting the calculation methodology we will present the geometric representation of an eigenvector. To do this we must remember that the multiplication of a matrix **A** by a vector $text\vec{b}fX$ produces another vector $text\vec{b}fY$. We can therefore represent this operation in a table by a matrix multiplication. By making both vectors (input and output) a function of the angle $\theta$ formed by the components of the input vector we can draw the circularization of the input vector and the output vector. While the input vector will describe a circle, the output vector will describe an ellipse.

```
import numpy as np
A = np.array([[1., 1.],[0.,1.]]); A
```

```
array([[1., 1.],
       [0., 1.]])
```

```
xi = lambda theta: [np.cos(theta*np.pi/180), np.sin(theta*np.pi/180)]
xi(45)
```

```
[0.7071067811865476, 0.7071067811865475]
```

```
xo = lambda theta:list(A.dot(xi(theta)))
xo(45)
```

```
[1.414213562373095, 0.7071067811865475]
```

```
import pandas as pd
thetas = np.linspace(0,360,361)
xis = np.array(list(map(xi, thetas)))
xis[:3]
```

```
array([[1.        , 0.        ],
       [0.9998477 , 0.01745241],
       [0.99939083, 0.0348995 ]])
```

```
import pandas as pd
thetas = np.linspace(0,360,361)
xos = np.array(list(map(xo, thetas)))
xos[:3]
```

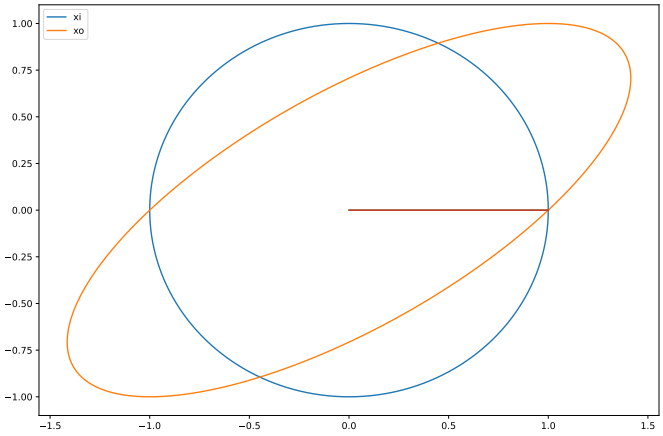
```
array([[1.        , 0.        ],
       [1.0173001 , 0.01745241],
       [1.03429032, 0.0348995 ]])
```

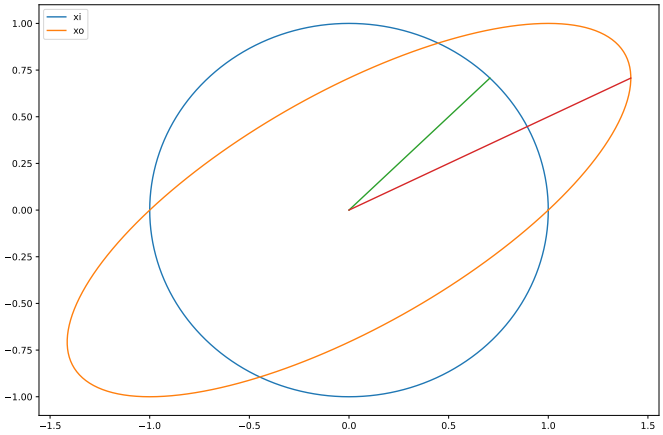
```
data = pd.DataFrame({'theta': thetas})
data['xi_x'] = xis[:,0]
data['xi_y'] = xis[:,1]
data['xo_x'] = xos[:,0]
data['xo_y'] = xos[:,1]
data.head()
```

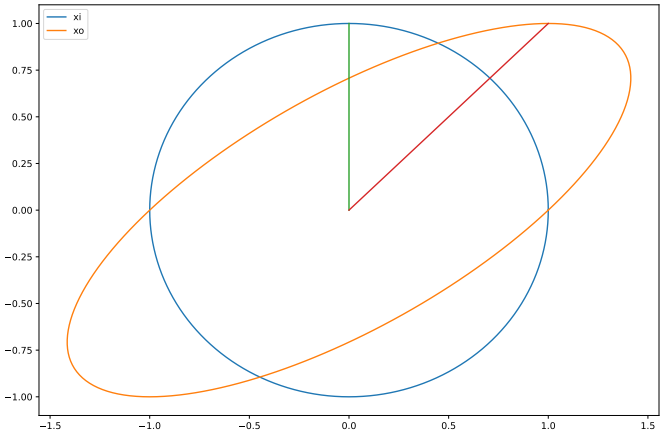
| | theta | xi_x | xi_y | xo_x | xo_y |
|---|---|---|---|---|---|
| 0 | 0.0 | 1.000000 | 0.000000 | 1.000000 | 0.000000 |
| 1 | 1.0 | 0.999848 | 0.017452 | 1.017300 | 0.017452 |
| 2 | 2.0 | 0.999391 | 0.034899 | 1.034290 | 0.034899 |
| 3 | 3.0 | 0.998630 | 0.052336 | 1.050965 | 0.052336 |
| 4 | 4.0 | 0.997564 | 0.069756 | 1.067321 | 0.069756 |

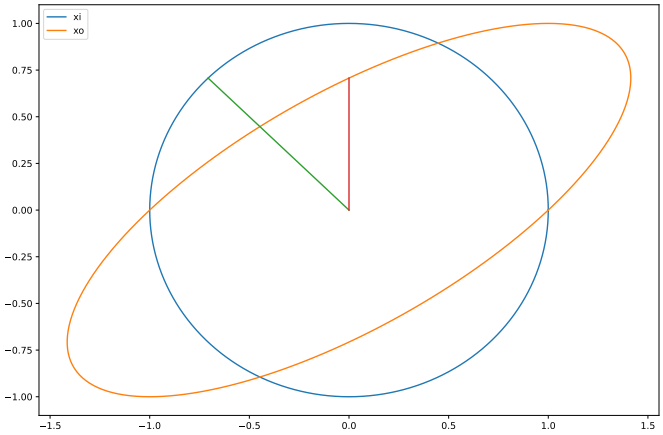
Now observe the effect of varying theta, on the representation of the input and output vectors. See that as theta increases, first the output vector is parallel, then it lags behind, then the delay decreases, and finally it becomes parallel to the input vector again. The values of theta to which the vectors are aligned, are the values of $x_i$ which are the eigenvectors of the matrix $A$. The size ratio between the two vectors ($\frac{x_o}{x_i}$) is the eigenvalue.

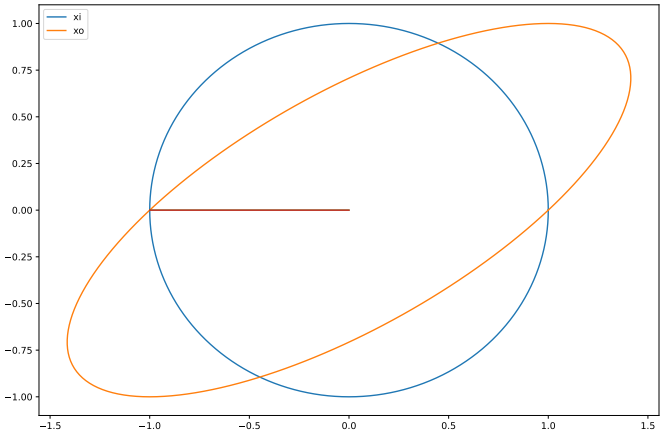
```
from matplotlib import pyplot as plt
for th in np.linspace(0,360,9):
  g = plt.plot(data['xi_x'], data['xi_y'], label = 'xi')
  g = plt.plot(data['xo_x'], data['xo_y'], label = 'xo')
  g = plt.plot([0, data['xi_x'][th]], [0, data['xi_y'][th]])
  g = plt.plot([0, data['xo_x'][th]], [0, data['xo_y'][th]])
  g = plt.legend();
  plt.show()
```

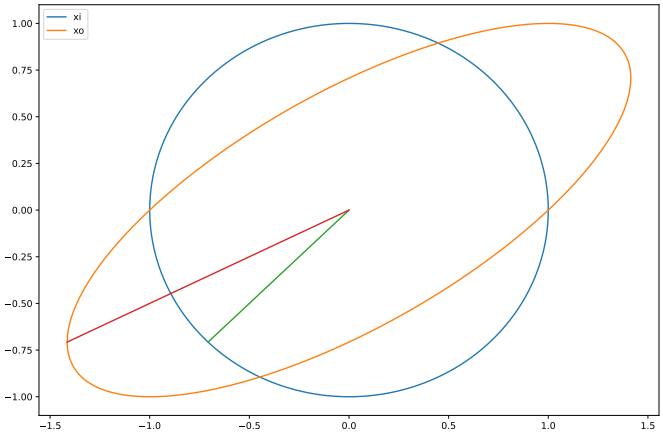

xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

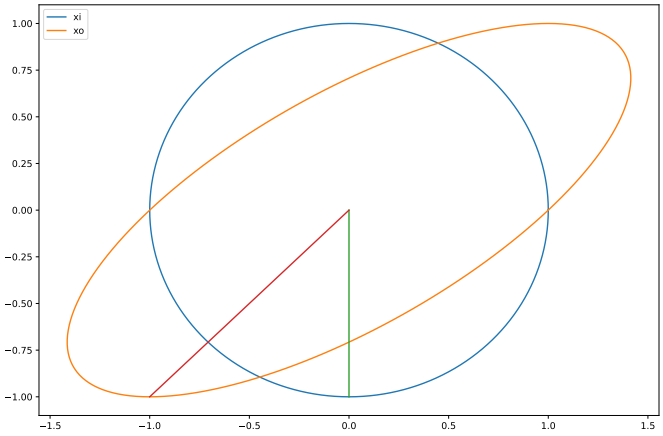

xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

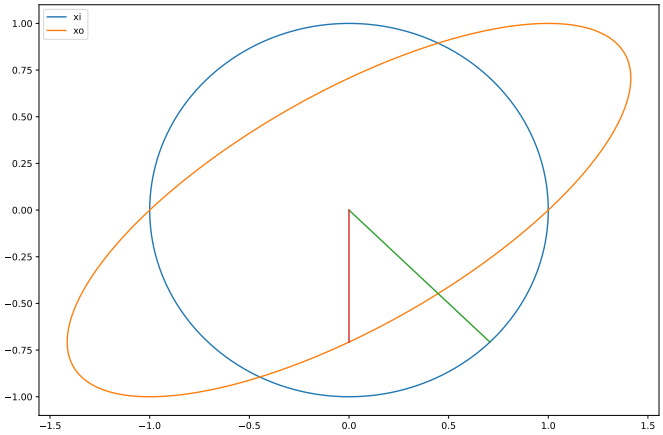

xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

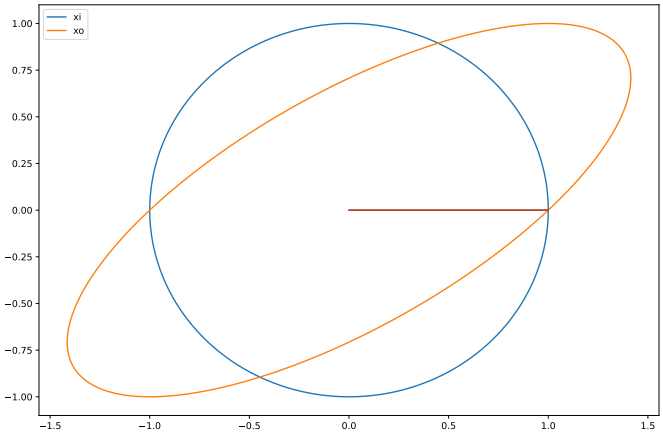

xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5




### 3.15.2 Calculation Methodology

{calculation_autovalues}

The 3.15.1 section described the geometric meaning of an eigenvalue and an eigenvector. In this step we will present the calculation methodology. The problem is divided into two parts : first the eigenvalues are calculated and from each eigenvalue the associated eigenvector is calculated.

From the algebraic point of view the resolution is obtained as follows :

$$\begin{aligned} \mathbf{A}.\vec{\mathbf{X}} &= \lambda.\vec{\mathbf{X}} \\ \mathbf{A}.\vec{\mathbf{X}} - \lambda.\vec{\mathbf{X}} &= \vec{0} \\ \mathbf{A}.\vec{\mathbf{X}} - \lambda.\mathbf{I}.\vec{\mathbf{X}} &= \vec{0} \\ [\mathbf{A} - \lambda.\mathbf{I}]\,.\vec{\mathbf{X}} &= \vec{0} \end{aligned} \tag{3.5}$$

From the equation 3.5 it can be seen that the eigenvalues come from the calculation of the determinant of the matrix

$$[\mathbf{A} - \lambda.\mathbf{I}] \tag{3.6}$$

The equation 3.6 will be a polynomial equation of degree equal to the order of the square matrix **A** and will therefore have the number of roots equal to its order. After calculating the eigenvalues each eigenvalue must be substituted into 3.5, which will each generate an indeterminate system. The solution (the solutions in fact) of each system will make up each associated eigenvector of the matrix **A**. This technique will become clearer in the example (solved verbatim) presented in the 3.15.3

### 3.15.3 Calculation Example

{

Let be the matrix $\mathbf{A} = \begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix}$. We will compute the eigenvalues and eigenvectors of $\mathbf{A}$ by replicating the sequence of steps described in equation 3.5. To do so we do :

$$\begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{3.7}$$

$$\begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \lambda . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \vec{0} \tag{3.8}$$

$$\begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \lambda . \mathbf{I} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \vec{0} \tag{3.9}$$

$$\left[ \begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix} - \lambda . \mathbf{I} \right] . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \vec{0} \tag{3.10}$$

$$\left[ \begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix} - \lambda . \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right] . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \vec{0} \tag{3.11}$$

$$\left[ \begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \right] . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \vec{0} \tag{3.12}$$

$$\begin{bmatrix} 1-\lambda & -5 \\ -5 & 1-\lambda \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.13}$$

Now to calculate $\lambda$ we need to solve the equation 3.14, that is, calculate $\lambda$ so that the determinant is 0.

$$\begin{vmatrix} 1-\lambda & -5 \\ -5 & 1-\lambda \end{vmatrix} = 0 \tag{3.14}$$

$$(1-\lambda).(1-\lambda) - 25 = 0 \tag{3.15}$$

$$(1-\lambda)^2 = 25 \tag{3.16}$$

$$1 - \lambda = \pm 5 \tag{3.17}$$

$$\lambda_1 = -4\lambda_2 = 6 \tag{3.18}$$

To sum up, for the calculation of the eigenvalues of a square matrix **A** it is enough to decrease $\lambda$ on the main diagonal and equal to 0 the determinant of the resulting matrix. Another detail that needs to be mentioned is that the equation 3.15 most of the time needs to be expanded to form a polynomial and then the resulting polynomial equation can be solved. In this case the expansion would take the following form:

$$\lambda^2 - 2\,\lambda - 24 = 0 \tag{3.19}$$

The equation 3.19 is called the *characteristic equation* of the matrix **A**. Now we must return the equation 3.11 and substitute each value of $\lambda$, solving the resulting systems and calculating the corresponding eigenvectors. In this case for $\lambda_1 = -4$ we will have :

$$\begin{bmatrix} 1-(-4) & -5 \\ -5 & 1-(-4) \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.20}$$

$$\begin{bmatrix} 5 & -5 \\ -5 & 5 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.21}$$

$$\begin{aligned} 5x_1 - 5x_2 &= 0 \\ -5x_1 + 5x_2 &= 0 \end{aligned} \tag{3.22}$$

The system represented in the equations 3.22 is indeterminate (which was expected since we calculated the value of $\lambda$ such that the determinant of the coefficient matrix of the system had value equal to 0). This way we will calculate a relation between $x_1$ and $x_2$ which will give us the line, that is, the direction, in which the eigenvector is located. In this case we will have :

$$\begin{aligned} 5x_1 - 5x_2 &= 0 \\ 5x_1 &= 5x_2 \\ x_1 &= x_2 \end{aligned} \tag{3.23}$$

This makes the eigenvector have the form shown in equation 3.24

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} .x_1 \tag{3.24}$$

Considering that $x_1$ is arbitrary we can represent it by the letter $k$. The eigenvector is then represented by

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} .k \tag{3.25}$$

Among all possible eigenvectors of the **A** matrix, associated with the eigenvalue $\lambda_1 = -4$ it is common to use a unit eigenvector, that is, with modulus equal to 1. This process is called normalizing the vector. Thus, normalizing the vector represented in equation 3.25 we have :

$$\sqrt{k^2 + k^2} = 1\sqrt{2}.k^2 = 1k = \frac{1}{\sqrt{2}} \tag{3.26}$$

As such the first pair of eigenvalue $\lambda_1$ and normalized eigenvector $e_1$ of the matrix **A** will be :

$$e_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \text{ com } \lambda_1 = -4 \tag{3.27}$$

Repeating the sequence of calculations presented between 3.14 and 3.27 for the eigenvalue $\lambda_2 = 6$ we will have the second pair of eigenvalue and normalized eigenvector of the matrix **A** which will be :

$$e_2 = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \text{ com } \lambda_2 = 6 \tag{3.28}$$

## 3.16 Calculation via numpy

The calculation of eigenvalues and eigenvectors is done by the function `numpy.linalg.eig`. As expected, this function returns both real and complex eigenvalues. For simplicity, suppose the matrix A (below), which because it is symmetric contains only real eigenvalues.

```
import numpy as np
A = np.array([[1,2,3],[2,4,1],[3,1,5]]); A
```

```
array([[1, 2, 3],
       [2, 4, 1],
       [3, 1, 5]])
```

The function `numpy.linalg.eig` returns both the eigenvalues and the associated eigenvectors. The eigenvalues can be seen below.

```
vals, vecs = np.linalg.eig(A);
vals
```

```
array([-0.97344343,  7.58959802,  3.38384541])
```

And below we have the eigenvectors arranged in the columns of the array.

```
vecs
```

```
array([[-0.87716982,  0.47994164, -0.015137  ],
       [ 0.2733656 ,  0.47320619, -0.83746472],
       [ 0.39477127,  0.73873671,  0.54628172]])
```

Notice the effect of multiplying the matrix A by one of the eigenvectors and then dividing each component by the associated eigenvalue. The resulting vector is equal to the initial vector.

```
A.dot(vecs[:,0])/vals[0]
```

```
array([-0.87716982,  0.2733656 ,  0.39477127])
```

## 3.17 Exercises

1. Perform the following operations with complex numbers, giving the result in both rectangular and polar form:
   (a) $(2-4j)+(3+4j)$
   (b) $(2-4j).(3+5j).(3-2j).(-2-3j)^*$
   (c) $\left(\frac{3-2j}{3+2j}\right)'$
   (d) $\left(100\sqrt{2}\,(1-j)\right)^{1/4}$
   (e) $\frac{-8+3\,j}{-2+4\,j}$
2. Calculate the value of the functions of complex variable $f(z)$ at the points $z = a + b, j$ given
   (a) $z^2 + j$ em $z = 1 + j$
   (b) $|z|^2 + j$ em $z = 1 + j$
   (c) $f^*(z)$ se $f(z) = z^2 + 2\,z\,z^* + j$ em $z = 1 + 2\,j$
   (d) $f(z) = 1 - z + |z|^2$ em $z = 1 + 2\,j$
3. Draw the graph of $f(z) = z + \frac{1}{z}$ for z ranging from $0 + 0\,j$ to $10 + 10\,j$
4. Compute the matrix operations shown below:

(a) Let $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 2 & 3 & 0 \\ -1 & 2 & 5 \end{bmatrix}$, calculate $\mathbf{A} + \mathbf{B}$ e $\mathbf{A} - \mathbf{B}$

(b) Let $\mathbf{A} = \begin{bmatrix} 1 & -2 \\ 2 & 3 \end{bmatrix}$, $k_1 = 5$ e $k_2 = -3 + 2j$, calculate $k_1\, A$ e $k_2\, A$.

(c) Produce a 3x3 Identity Matrix

(d) Let the matrix $\mathbf{A} = \begin{bmatrix} 1+2j & j \\ 3 & 2-3j \end{bmatrix}$, calculate $\mathbf{A}^{*T}$ i.e. the transpose of the conjugate of $\mathbf{A}$

5. Is $L(\overrightarrow{u}) = \begin{bmatrix} -5u_1 + 4u_2 + 5 \\ +3u_1 - 6u_2 \end{bmatrix}$ transformation linear? Prove your assertion by testing both linearity conditions.

6. Verify whether the transformation defined by $L([u_1, u_2, u_3]) = [u_1 + u_2 - u_3, u_1 + 2u_3, u_1 - 2u_2 + 4]$ is linear or not by testing BOTH linearity conditions. Does this transformation have a canonical matrix? If yes, present the same, if no explain why.

7. Verify whether or not a transformation that has the following as its results could be defined as linear, explaining why your answer

   (a) L(u) = [9 ; 8], L[v] = [5 ; 4] and L[u+v] = [ 14 ; 12 ]

   (b) L(u) = [9 ; 8], L[3u] = [27 ; 34]

8. Calculate the value of K so that the system below is impossible.

$$\begin{gathered} X + 3Y + 4Z = -1 \\ +Y + KZ = -3 \\ 2X + 2Z = -2 \end{gathered}$$

9. Find the largest value of X that makes the determinant of the matrix below equal to 0

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ x & 1 & 2 & 3 \\ x^2 & 1 & 4 & 9 \\ x^3 & 1 & 8 & 27 \end{bmatrix}$$

10. Be X1, X2, X3 the solutions of the system represented by the matrices A and B below, where X1, X2, and X3 are arranged in GROWING order. What is the value of the expression X1+X2-X3?

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 3 & 2 \end{bmatrix}, B = \begin{bmatrix} 3 \\ 8 \\ 11 \end{bmatrix}$$

11. Calculate the eigenvalues and eigenvectors of the matrix $A = \begin{bmatrix} 3 & -1 \\ -1 & 5 \end{bmatrix}$ (use 2 places after the comma as numerical precision if necessary)

12. Can the matrix below represent a covariance matrix? Justify your answer

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

13. Find the equation that determines the eigenvalues, (also called and characteristic equation), the eigenvalues and the normalized eigenvectors (i.e., with modulus equal to 1) of the following matrix:

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

14. Consider the set of eigenvalues and associated normalized eigenvectors shown in table 3.1. Determine how many factors will be required to represent at least 90% of the original matrix and present this matrix from the spectral analysis formula.

Tabela 3.1: Table of eigenvalues and eigenvectors

| eigenvalues | eigenvector_1 | eigenvector_2 | eigenvector_3 | eigenvector_4 |
|---|---|---|---|---|
| 70.778687 | -0.053753 | 0.857547 | -0.405648 | -0.311726 |
| 8.673544 | -0.009352 | 0.361164 | 0.049184 | 0.931157 |
| 1.802103 | -0.314252 | 0.328895 | 0.872815 | -0.176826 |
| 0.265425 | 0.947771 | 0.161251 | 0.266878 | -0.067122 |

15. Consider the set of associated normalized eigenvalues and eigenvectors available in the table 3.2. Determine how many factors will be required to represent at least 90% of the original matrix and present this matrix from the spectral analysis formula.

Tabela 3.2: Table of eigenvalues and eigenvectors

| eigenvalues | eigenvector_1 | eigenvector_2 | eigenvector_3 | eigenvector_4 | eigenvector_5 |
|---|---|---|---|---|---|
| 21.71 | 0.76 | 0.01 | -0.18 | -0.21 | -0.59 |
| 9.62 | 0.44 | -0.37 | -0.13 | -0.36 | 0.73 |
| 5.35 | 0.46 | 0.26 | 0.67 | 0.47 | 0.21 |
| 4.15 | 0.03 | 0.89 | -0.23 | -0.30 | 0.24 |
| 0.97 | 0.16 | 0.03 | -0.66 | 0.72 | 0.15 |

16. A dairy is going to mix two types of milk: one that has 1

17. A car fuel has 10

18. A simple macroeconomic model consists of three equations: a) Consumption $C = \alpha_0 + \alpha_1 Y + \alpha_2 T$, b) Investment $I = \beta_0 + \beta_1 Y + \beta_2 K$ and Income $Y = C + I + G$ where $T$ is taxes, $K$ is the Capital stock and $G$ is government spending. This model could be written in matrix form as described in 3.29. Give a necessary condition for such a system to have unique solution. Ans: for example the coefficient matrix $[3, 3]$ on the left must be invertible, that is have nonzero determinant.

$$\begin{bmatrix} 1 & 0 & -\alpha_1 \\ 0 & 1 & -\beta_1 \\ -1 & -1 & 1 \end{bmatrix} . \begin{bmatrix} C \\ I \\ Y \end{bmatrix} = \begin{bmatrix} \alpha_0 + \alpha_2 T \\ \beta_0 + \beta_2 K \\ G \end{bmatrix} \tag{3.29}$$

19. Give an example where the linear combination of three non-null vectors of $R^4$ produces the null vector. Confirm the result by creating with these three vectors a matrix A and multiplying it by an appropriate x-vector to generate the null vector. Also enter the dimensions of A, x and the null vector.

```
import numpy as np
import sympy as sp
x1 = np.array([1,1,1,1]).reshape(-1,1)
x2 = np.array([2,3,4,5]).reshape(-1,1)
x3 = np.array([3,4,5,6]).reshape(-1,1)
A = np.concatenate([x1,x2,x3],axis=1)
A
```

```
array([[1, 2, 3],
       [1, 3, 4],
       [1, 4, 5],
       [1, 5, 6]])
```

```
x = np.array([1,1,-1]).reshape(-1,1)
x
```

```
array([[ 1],
       [ 1],
       [-1]])
```

```
R = A.dot(x)
R
```

```
array([[0],
       [0],
```

```
        [0],
        [0]])
```

20. Suppose a matrix with vectors in $R^4$ where there are two columns where each is a linear combination of two other columns. Based on this statement find two distinct solutions x and y for Az=0 (i.e. vectors x and y such that Ax=0 and Ay=0).

21. Suppose a matrix of ones (3x3). Find two independent vectors that are solution of Az=0. Is there a 3rd independent vector? No! The space of the columns of A has dimension 1. The dimension of A is 3. Therefore the null space of A has dimension 2. Having dimension 2 we have only two independent vectors that can be the solution of A.z = 0.

22. The vectors $v = (1, 1, 0)$ and $w = (0, 1, 1)$ form a plane in $R^3$. Find a vector z perpendicular to this plane. Solution: If this vector is called $z$ we will have $(c\vec{v} + d\vec{w}).z = 0$. Then $c = -z_1$, $d = -z_3$ and $c + d = -z_2$. Then the vector z perpendicular to the plane formed by $v$ and $w$ will have the form $z_1 + z_3 = z_2$. An example would be $z = (1, 2, 1)$

# Capítulo 4

# Linear Algebra via Python Numpy

Let's now learn more about `numpy`, Python's library for manipulating arrays. Understand an array `numpy` as a matrix stored in the computer's memory. First we will set up the environment, as described in the following commands:

```
import numpy as np
# Number of decimal places for printing results
np.set_printoptions(precision=3)

# Whether to suppress or keep scientific notation
np.set_printoptions(suppress=False)

# Maximum number of elements to be printed
np.set_printoptions(threshold=1000)

# Version of numpy in use
print(np.__version__)
```

```
1.21.2
```

To begin we create a numpy array similar to the array described below:

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix} \tag{4.1}$$

We can create this array explicitly by indicating each number that makes it up:

```
import numpy as np
a = np.array([[0,1,2,3],[4,5,6,7],[8,9,10,11]])
a
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

Since the numbers are in ascending order, from 0 through 11 we can also create it automatically with `np.arange` (which will generate a one-dimensional `numpy` array also called a vector) and then reformat it (via the `.reshape` method):

```
import numpy as np
a = np.arange(0,12).reshape(3,4)
a
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

## 4.1 Axes and Dimensions

An important first observation. The array:

```
import numpy as np
a = np.arange(0,12).reshape(3,4)
a
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

has 2 axes, the first axis has 3 dimensions and the second axis has 4 dimensions. We therefore say that it is `2D [3,4]`. Always read the capital D as **axis** and avoid reading the D as `dimension`. The word dimension specifies how many addresses we have on each axis. Another example: a 3D array [5,1,7] is a three axis object, the first axis with five dimensions, the second axis with one dimension and the third axis with seven dimensions.

Using the name "axis" and avoiding the use of the word "dimension" for capital D ends the ambiguity between vectors in programming and vectors in mathematics. In mathematics a vector is a matrix with n rows and only one column. So in mathematics, a vector is an object with two axes (2D). In programming a vector is an object with only one axis (1D therefore) and n dimensions. The reader

should pay attention to the context and remember that "vector" in mathematics is a different object than "vector" in computer science.

That said, indexing, that is the numbering of the addresses on each axis in `numpy` arrays, works in a similar way to that of lists. In a way, the `numpy` arrays can be understood as "lists with greater computational functionality." The most important of these is the possibility to perform a mathematical operation directly on each element of the array without the use of loops or list comprehension. For example, suppose you want to add 3 to each of the elements of the above array. To do this, simply add 3 to the array, as you can see below:

```
import numpy as np
a = np.arange(0,12).reshape(3,4)
a + 3
```

```
array([[ 3,  4,  5,  6],
       [ 7,  8,  9, 10],
       [11, 12, 13, 14]])
```

Or for example determine which element is equal to 5. To perform such an operation simply compare the array to 5 using `==`. This will generate a sequence of `True` and `False` with the value `True` at the positions where the element value is 5.

```
import numpy as np
a = np.arange(0,12).reshape(3,4)
a == 5
```

```
array([[False, False, False, False],
       [False,  True, False, False],
       [False, False, False, False]])
```

Regarding operations with Booleans, the operators not: `~`, and:`&` and ou:`|` can be applied normally on arrays, as for example below when we calculate $(not(A == 5))\ and\ (not(A == 6))$.

```
import numpy as np
a = np.arange(0,12).reshape(3,4)
(~(a==5))&(~(a==6))
```

```
array([[ True,  True,  True,  True],
       [ True, False, False,  True],
       [ True,  True,  True,  True]])
```

Other important methods are `.any(axis=number)` and `.all(axis=number)`. When applied to an array of booleans they check whether at least one (any) or all (all) elements of an array (of logical elements) along an axis contain the value

`True`. Notice the following. We create the logical array above and save it in the variable `b`. If we apply on `b` the method `.all(axis=0)` we get as response an array of one axis (1D) with 4 positions (i.e. four dimensions) one for each column of `b`. The value of this result will be `True` if all elements of the column are `True`.

```
import numpy as np
a = np.arange(12).reshape(3,4)
b = (~(a==5))&(~(a==6))
b.all(axis=0)
```

```
array([ True, False, False,  True])
```

Or if we want to evaluate which rows have any of their elements equal to `True`, we will apply the `.any(axis=1)` method, obtaining as a result a one-axis (1D) array of three dimensions, one for each row, as can be seen below.

```
import numpy as np
a = np.arange(12).reshape(3,4)
b = (~(a==5))&(~(a==6))
b.any(axis=1)
```

```
array([ True,  True,  True])
```

Another important observation, especially for users used to the calculation logic of R or Excel. In Python when we pass the parameter `axis=0` we mean that the calculation will be done **along* the 0 axis, so the results will be obtained per column! This is the opposite of what happens in R where we must specify the axis on which we want the results! So remember, in Python we specify the axis along which the operation will be performed, not the axis along which we want the results. Experienced R users in particular take a long time to build up the "instinct" to know when to specify `axis=0` or `axis=1`, because for them the logic works the other way around.

Like everything in computing there are advantages and disadvantages to both approaches. R's logic (specifying the axis on which we want to "see" the results) is more "visual" but suffers from one flaw. It works well only for two dimensions when we can "see" where the results will appear (along the row axis or along the column axis). But how to "see" the result in 3, 4, 5 dimensions? Suppose a 4-axis array with 10 dimensions on each axis. Each element of it would be specified for example as: `a[1,5,3,4]`. How do you visualize the axis along which the results will appear? What does it mean to ask for the results to be displayed "along" the third axis (`axis=2`)?

Now think of the way Python "thinks." If you say `a.all(axis=2)` (third axis) it will (in this example) analyze for each trio of values (in the other 3 axes) whether all elements `a[a0,a1,0,a3]` through `a[a0,a1,9,a3]` are `True` and will return

you as answer a new now three-axis array. Complicated? Yes, but at least this way you can manipulate objects of more than two axes, while the other way, if the number of axes is greater than two, "who knows?"

Continuing our study of `numpy` features, it is interesting to mention the following functions, which will be very useful in solving the exercises in this chapter:

1. The `.ravel()` method "disassembles"an array into 1D format (with only one axis). This allows you to locate in an array with more axes the element of n-th position.

2. Random numbers can be generated by the functions `np.random.randint(smallest, largest, quantity)`, `np.random.uniform(smallest, largest, quantity)` and `np.random.normal(mean, variance, quantity)`, where the parameter `quantity` is the desired number of samples. The result will be a 1D array (of only one axis) which can be reformatted using the `.reshape` method. As an example, to create a two-axis (2D) array of dimensions `[4,5]` consisting of 20 samples from an integer distribution between 1 and 6 we do `np.random.randint(1,7,20).reshape(4,5)`

3. To determine the common elements of two arrays (of the same number of axes and shapes) we can use a combination of the function `set` (which provides the unique elements of an object) and the logical operator `and` which in Python is `&` by doing: `set(x) & set(y)`. Or we can use the function `np.intersect1D(x,y)`. To determine all the distinct elements in the two arrays (their union) we do: `set(list(set(x)) + list(set(y)))`

4. The position of an element (from its value, for example the largest value in a single-axis (1D) array) can be determined as follows: `z[z==np.max(z)]`. Note that `z==np.max(z)` will compare each value of `z` with the largest value in the array. This will generate a sequence of `True` and `False`. When we apply this boolean sequence to the addresses in the `z` array we get only the values for which we have `True` in the boolean sequence.

5. When we want to get the position of an element in an array of more than one axis (2D for example) we use the function `np.unravel_index`. Suppose you want to determine the position of the 10th element, but in an array of two axes, in the format `[5,5]`. For this we would do `np.unravel_index(9, (5,5))` and get as answer `(1,4)` i.e. 2nd row, 4th column.

6. The logic for finding the position of a value in an array presented in detail in 4 and 5 is also implemented directly through the function `np.where`

7. Creating a 3D (3-axis) array of dimensions `[3,4,5]` means in practice creating 3 different arrays, each of dimensions `4,5`. To observe such an effect execute `np.arange(60).reshape(3,4,5)`

## 4.2 Structured Arrays

A structured array is an array in which every position in the array is another array. For example, look below for a data type (dtype) that describes a color as four unsigned bytes (RGBA)

```
import numpy as np
color = np.dtype([("r", np.ubyte, (1,)),
                  ("g", np.ubyte, (1,)),
                  ("b", np.ubyte, (1,)),
                  ("a", np.ubyte, (1,))])

print(color)
```

```
[('r', 'u1', (1,)), ('g', 'u1', (1,)), ('b', 'u1', (1,)), ('a', 'u1', (1,))]
```

### 4.2.1 1st Example

As a first example, suppose we wish to create a structured array in which the coordinates $x$ and $y$ cover the area $[0, 1], [0, 1]$. As stated a structured array is one in which each element of the array is another array, tuple or list. The following array is of the unstructured type.

```
import numpy as np
z = np.zeros([3,3], dtype=float)
print(z)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

The following array is of the structured type, but each element has a unique internal structure.

```
import numpy as np
z = np.zeros([3,3], dtype=[("x",float)])
print(z)
```

```
[[(0.,) (0.,) (0.,)]
 [(0.,) (0.,) (0.,)]
 [(0.,) (0.,) (0.,)]]
```

We can load the internal structure of the array via its specific label.

```
print(z["x"])
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

```
z["x"] = 1
print(z)
```

```
[[(1.,) (1.,) (1.,)]
 [(1.,) (1.,) (1.,)]
 [(1.,) (1.,) (1.,)]]
```

Now let's create an array with two internal structures. Notice that at this point we are creating an array of pairs of 0s. The array of the 1st element of each pair is accessed as `x`, the data type is float. The array of the 2nd element of each pair is accessed as `y`, the data type is also float.

```
z = np.zeros([3,3], dtype=[("x",float),("y",float)])
print(z)
```

```
[[(0., 0.) (0., 0.) (0., 0.)]
 [(0., 0.) (0., 0.) (0., 0.)]
 [(0., 0.) (0., 0.) (0., 0.)]]
```

Now let's modify the internal structures of the array. First the structure whose label is called `x`.

```
z["x"]=1
print(z)
```

```
[[(1., 0.) (1., 0.) (1., 0.)]
 [(1., 0.) (1., 0.) (1., 0.)]
 [(1., 0.) (1., 0.) (1., 0.)]]
```

And now the structure with the label `y`

```
z["y"]=2
print(z)
```

```
[[(1., 2.) (1., 2.) (1., 2.)]
 [(1., 2.) (1., 2.) (1., 2.)]
 [(1., 2.) (1., 2.) (1., 2.)]]
```

Notice that each element of the `z` array is a structure with two values. For example `z[0,0]`

```
print(z[0,0])
```

```
(1., 2.)
```

In turn the element `(0,0)` of the `x` structure of the `z` array is a single value and equal to `1`.

```
print(z["x"][0,0])
```

```
1.0
```

We can then insert the desired elements into each of the "sub-arrays." First we create the structured array with two "parallel" sub-arrays.

```
z = np.zeros([3,3], dtype = [("x",float),("y",float)])
print(z)
```

```
[[(0., 0.) (0., 0.) (0., 0.)]
 [(0., 0.) (0., 0.) (0., 0.)]
 [(0., 0.) (0., 0.) (0., 0.)]]
```

We create a 1D array, which for purposes of working with 2D arrays is considered a row array.

```
x = np.linspace(0,2,3)
print(x)
```

```
[0. 1. 2.]
```

The sub-array `z['x']` is a 2D array, $(5, 5)$. For the purposes of inserting the values the 1D array `x` is taken as $[1, 5]$ and will be aligned on top of the array `z['x']`. Then the values will be matched row by row.

```
z['x'] = x
print(z)
```

```
[[(0., 0.) (1., 0.) (2., 0.)]
 [(0., 0.) (1., 0.) (2., 0.)]
 [(0., 0.) (1., 0.) (2., 0.)]]
```

```
print()
```

We change the shape of the array x to a 2D array, $[3, 1]$. For comparison purposes with the `z['y']` array, which is $[3, 3]$, it will be aligned "upright" next to the `z['y']` array and then have the values matched column by column.

```
z['y'] = x.reshape(3,1)
print(z)
```

```
[[(0., 0.) (1., 0.) (2., 0.)]
 [(0., 1.) (1., 1.) (2., 1.)]
 [(0., 2.) (1., 2.) (2., 2.)]]
```

The following form "stacks" 1D arrays via `np.vstack()`.

```
a = np.linspace(0,2,3)
print(a)
```

```
[0. 1. 2.]
```

```
print(np.vstack([a,a,a]))
```

```
[[0. 1. 2.]
 [0. 1. 2.]
 [0. 1. 2.]]
```

```
print(np.linspace(0,2,3).reshape(3,1))
```

```
[[0.]
 [1.]
 [2.]]
```

```
print(np.vstack([a,a,a]).transpose())
```

```
[[0. 0. 0.]
 [1. 1. 1.]
 [2. 2. 2.]]
```

```
a = np.linspace(0,1,3)
print(a)
```

```
[0.  0.5 1. ]
```

```
print()
```

```
z = np.zeros( [3,3] , dtype = [("x",float), \
                               ("y",float)] )
print(z)
```

```
[[(0., 0.) (0., 0.) (0., 0.)]
 [(0., 0.) (0., 0.) (0., 0.)]
 [(0., 0.) (0., 0.) (0., 0.)]]
```

```
print()
```

Next, for reasons of print space we will "stack" only three copies of the 1D array `a`.

```
z["x"], z["y"] = [np.vstack([a,a,a]), \
                  np.vstack([a,a,a]).transpose()]
print(z)
```

```
[[(0. , 0. ) (0.5, 0. ) (1. , 0. )]
 [(0. , 0.5) (0.5, 0.5) (1. , 0.5)]
 [(0. , 1. ) (0.5, 1. ) (1. , 1. )]]
```

This is the most efficient way. It automatically generates two independent, transposed arrays. One with equal elements in the rows and one with equal elements in the columns. We can use this system to avoid having to assign values by transposing `np.linspace` explicitly.

```
z = np.meshgrid(np.linspace(0,1,3),np.linspace(0,1,3))
print(z[0], "\n", z[1])
```

```
[[0.  0.5 1. ]
 [0.  0.5 1. ]
 [0.  0.5 1. ]]
 [[0.  0.  0. ]
 [0.5 0.5 0.5]
 [1.  1.  1. ]]
```

We will then pass each of the results of `np.meshgrid` to each of the sub arrays that form the structured array `z`.

```
z = np.zeros((3,3), [("x",float),("y",float)])
z["x"], z["y"] = np.meshgrid(np.linspace(0,1,3),
                             np.linspace(0,1,3))
print(z)
```

```
[[(0. , 0. ) (0.5, 0. ) (1. , 0. )]
 [(0. , 0.5) (0.5, 0.5) (1. , 0.5)]
 [(0. , 1. ) (0.5, 1. ) (1. , 1. )]]
```

### 4.2.2 2nd Example

In this example we will create a 5x1 structured array representing a position $(x, y)$ and a color $(r, g, b)$.

First we create the array with the representation of $x$ and $y$

```
position = [("x", "float"),("y", "float")]
print(position)
```

```
[('x', 'float'), ('y', 'float')]
```

Then the array with the color representation

```
color = [("r", "float"),("g", "float"),("b", "float")]
print(color)
```

```
[('r', 'float'), ('g', 'float'), ('b', 'float')]
```

And then the array `z`, `[3,1]`, with the `datype=position,color`

```
z = np.zeros([3,1], dtype=[("p",position),("c",color)])
print(z)
```

```
[[((0., 0.), (0., 0., 0.))]
 [((0., 0.), (0., 0., 0.))]
 [((0., 0.), (0., 0., 0.))]]
```

Or straightforwardly:

```
Z = np.zeros([5,1], [ ("position", \
                       [ ("x", float, 1),
                         ("y", float, 1) ] ),
                      ("color",
                       [ ("r", float, 1),
                         ("g", float, 1),
                         ("b", float, 1) ] ) ] )
print(Z)
```

```
[[((0., 0.), (0., 0., 0.))]
 [((0., 0.), (0., 0., 0.))]
 [((0., 0.), (0., 0., 0.))]
 [((0., 0.), (0., 0., 0.))]
 [((0., 0.), (0., 0., 0.))]]
```

Let's interpret the previous result. Starting with an array of 5 elements.

```
z = np.zeros(5)
print(z)
```

```
[0. 0. 0. 0. 0.]
```

An array of 5 elements, structured in 2 sub-arrays of 2 elements. The first is called position and the second is called color. Different numeric types were used to make it easier to see.

```
z = np.zeros(5, dtype=[("position",np.int64), \
                       ("color",float)])
print(z)
```

```
[(0, 0.) (0, 0.) (0, 0.) (0, 0.) (0, 0.)]
```

Accessing the first sub-array:

```
print(z["position"])
```

```
[0 0 0 0 0]
```

Accessing the second sub-array:

```
print(z["color"])
```

```
[0. 0. 0. 0. 0.]
```

Continuing from the previous cell. We will now increase the complexity of the first sub-array. Each element of the first array will consist of 2 elements

```
import numpy as np
z = np.zeros(5, dtype=[("position", \
                        [("x",float), \
                         ("y",float)]), \
                         ("color",float)])
print(z)
```

```
[((0., 0.), 0.) ((0., 0.), 0.) ((0., 0.), 0.) ((0., 0.), 0.)
 ((0., 0.), 0.)]
```

```
print(z["position"])
```

```
[(0., 0.) (0., 0.) (0., 0.) (0., 0.) (0., 0.)]
```

We will now increase the complexity of the second sub-array. Each element of it will consist of three elements

```
z = np.zeros(5, dtype=[("position",[("x",float),
                                    ("y",float)]),
                       ("color",[("r",float),
                                 ("g",float),
                                 ("b",float)])])
print(z)
```

```
[((0., 0.), (0., 0., 0.)) ((0., 0.), (0., 0., 0.))
 ((0., 0.), (0., 0., 0.)) ((0., 0.), (0., 0., 0.))
 ((0., 0.), (0., 0., 0.))]
```

Let's include the third argument in each dtype

```
z = np.zeros(5, dtype=[("position",[("x",float,1),
                                    ("y",float,1)]),
                       ("color",[("r",float,1),
                                 ("g",float,1),
                                 ("b",float,1)])])
print(z)
```

```
[((0., 0.), (0., 0., 0.)) ((0., 0.), (0., 0., 0.))
 ((0., 0.), (0., 0., 0.)) ((0., 0.), (0., 0., 0.))
 ((0., 0.), (0., 0., 0.))]
```

### 4.2.3 3rd Example

In this example we will create a calculation grid for $z = x^2 + y^2$ and get all the values of `z` in the plane domain $x = [-2, 2]$ and $y = [-2, 2]$ divided into 5 points.

First let's generate a "zeroed" grid

```
w = np.zeros([5,5])
print(w)
```

```
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
```

Let's next produce the horizontal axis, with 11 points between -5 and 5

```
x = np.linspace(-2,2,5).reshape(1,-1)
print(x)
```

```
[[-2. -1.  0.  1.  2.]]
```

And now using the broadcast feature let's do `xg = x - w` and see the result

```
xg = x - w
print(xg)
```

```
[[-2. -1.  0.  1.  2.]
 [-2. -1.  0.  1.  2.]
 [-2. -1.  0.  1.  2.]
 [-2. -1.  0.  1.  2.]
 [-2. -1.  0.  1.  2.]]
```

Let's do something similar for `y`, but taking care to make it a vertical array first.

```
y = np.linspace(-2,2,5).reshape(-1,1)
print(y)
```

```
[[-2.]
 [-1.]
 [ 0.]
 [ 1.]
 [ 2.]]
```

Now notice the effect of doing `yg = y - w`

```
yg = y - w
print(yg)
```

```
[[-2. -2. -2. -2. -2.]
 [-1. -1. -1. -1. -1.]
 [ 0.  0.  0.  0.  0.]
 [ 1.  1.  1.  1.  1.]
 [ 2.  2.  2.  2.  2.]]
```

Let's now create the grid of point pairs `xg,yg` via a structured array named `g`

```
g = np.zeros((5,5), [("x",float),("y",float)])
g["x"]=xg
g["y"]=yg
print(g)
```

```
[[(-2., -2.) (-1., -2.) ( 0., -2.) ( 1., -2.) ( 2., -2.)]
 [(-2., -1.) (-1., -1.) ( 0., -1.) ( 1., -1.) ( 2., -1.)]
 [(-2.,  0.) (-1.,  0.) ( 0.,  0.) ( 1.,  0.) ( 2.,
0.)]
 [(-2.,  1.) (-1.,  1.) ( 0.,  1.) ( 1.,  1.) ( 2.,
1.)]
 [(-2.,  2.) (-1.,  2.) ( 0.,  2.) ( 1.,  2.) ( 2.,
2.)]]
```

From the structured array `g` we calculate the function $z = x^2 + y^2$

```
z = g["x"]**2 + g["y"]**2
print(z)
```

```
[[8. 5. 4. 5. 8.]
 [5. 2. 1. 2. 5.]
 [4. 1. 0. 1. 4.]
 [5. 2. 1. 2. 5.]
 [8. 5. 4. 5. 8.]]
```

## 4.3 Generators and Iterators

`Generators` and `Iterators` in Python are similar to the `while` and {for} loops with the difference that they allow the code to leave the loop and when returning, find it at the point where it was exited last time. As with the `while` loops a `Generator` keeps its state through local variables and similarly to the `for` loops or `Iterators` keep its state automatically (by a variable defined along with the `Iterator`). Let's look at some examples below.

### 4.3.1 1st Example

As a first example let's create a generator that outputs 10 whole numbers. We will use it together with the `np.fromiter` function to produce arrays with size, 1, 2, 3, 10. We will also see what will happen in cases where the size indicated in the count parameter equals both -1 and 11.

```
def generate():
  for x in range(10):
    yield(x)

z = np.fromiter(generate(), dtype="int64", count=1)
print(z)
```

```
[0]
```

```
z = np.fromiter(generate(), dtype="int64", count=2)
print(z)
```

```
[0 1]
```

```
z = np.fromiter(generate(), dtype="int64", count=3)
print(z)
```

```
[0 1 2]
```

```
z = np.fromiter(generate(), dtype="int64", count=10)
print(z)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

```
z = np.fromiter(generate(), dtype="int64", count=-1)
print(z)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

```
try:
  z = np.fromiter(generate(), dtype="int64", count=11)
  print(z)
except:
  print("Generator produced an error because it was out of range")
```

```
Generator produced an error because it was out of range
```

### 4.3.2 2nd Example

In this example we will use an iterator to calculate the sum of two arrays with formats `[1,3]` and `3,1`.

Take a look at the following solution:

```
A = np.arange(3).reshape(3,1)
B = np.arange(3).reshape(1,3)
it = np.nditer([A,B,None])
for x,y,z in it:
  z[...] = x + y

print(it.operands[2])
```

```
[[0 1 2]
 [1 2 3]
 [2 3 4]]
```

To understand the presented code, we look in the help of `numpy` for the definition of Iterator where we get the following example:

```
a = np.arange(6).reshape(2,3)
print(a)
```

```
[[0 1 2]
 [3 4 5]]
```

```
print()
```

```
for x in np.nditer(a):
  print(x, end=' ')
```

```
0 1 2 3 4 5
```

As can be seen the function `np.nditer()` scrolls through the array element by element. So let's take a look at the code provided above. To better identify A and B we will assign distinct values

```
A = np.arange(3).reshape(3,1)
print(A)
```

```
[[0]
 [1]
 [2]]
```

```
B = np.arange(10,13).reshape(1,3)
print(B)
```

```
[[10 11 12]]
```

```
for x in np.nditer(A):
  print(x,' ')
```

```
0
1
2
```

```
for x in np.nditer(B):
  print(x,' ')
```

```
10
11
12
```

See that the effect of `np.nditer([A,B])` is the same as that of a double loop

```
for x in np.nditer([A,B]):
  print(x,' ')
```

```
(array(0), array(10))
(array(0), array(11))
(array(0), array(12))
(array(1), array(10))
(array(1), array(11))
(array(1), array(12))
(array(2), array(10))
(array(2), array(11))
(array(2), array(12))
```

```
for x in np.nditer([A,B]):
  print(x[0], ' ')
```

```
0
0
0
1
1
1
2
2
2
```

```
for x in np.nditer([A,B]):
  print(x[1], ' ')
```

```
10
11
12
10
11
12
10
11
12
```

What is the function of None in the original iterator? We'll see that it is used to create an uninitialized variable

```
for x in np.nditer([A,B,None]):
  print(x,' ','shape =',np.array(x).shape)
```

```
(array(0), array(10), array(0))    shape = (3,)
(array(0), array(11), array(4602678819172646912))
shape = (3,)
```

```
(array(0), array(12), array(4607182418800017408))
shape = (3,)
(array(1), array(10), array(0))    shape = (3,)
(array(1), array(11), array(4602678819172646912))
shape = (3,)
(array(1), array(12), array(4607182418800017408))
shape = (3,)
(array(2), array(10), array(0))    shape = (3,)
(array(2), array(11), array(4602678819172646912))
shape = (3,)
(array(2), array(12), array(4607182418800017408))
shape = (3,)
```

Now, to recall, the command below creates the 2D array, A with the format `[3,1]`

```
A = np.arange(3).reshape(3,1)
print(A)
```

```
[[0]
 [1]
 [2]]
```

Creates the 2D array, with the format `B[1,3]`

```
B = np.arange(3).reshape(1,3)
print(B)
```

```
[[0 1 2]]
```

Generates an iteration object. This object in turn generates 3 values `A[i], B[i], None`

```
it = np.nditer([A,B,None])
```

Before the loop. The 3rd element of `[A,B,None]` is initialized with whatever value is currently in memory.

```
print(it.operands[0])
```

```
[[0]
 [1]
 [2]]
```

```
print()
```

```
print(it.operands[1])
```

```
[[0 1 2]]
```

```
print()
```

```
print(it.operands[2])
```

```
[[                   0                    0
0]
 [4602678819172646912 4602678819172646912 4602678819172646912]
 [4607182418800017408 4607182418800017408 4607182418800017408]]
```

```
print()
```

The loop is executed

```
for x,y,z in it:
  z[...] = x + y
```

After the loop

```
print(it.operands[0])
```

```
[[0]
 [1]
 [2]]
```

```
print()
```

```
print(it.operands[1])
```

```
[[0 1 2]]
```

```
print()
```

```
print(it.operands[2])
```

```
[[0 1 2]
 [1 2 3]
 [2 3 4]]
```

The `...` operator, also known as ellipsis, remains to be understood. Let's get back to the code. Notice what happens when we reset the values of `it` at the end of the code and then rerun it

```
A = np.arange(0,4,dtype=np.int32).reshape(4,1)
B = np.arange(0,4,dtype=np.int32).reshape(1,4)

it = np.nditer([A,B,None])

print(it.operands[2])
```

```
[[-1680603523 -1076628655           0           0]
 [-1009901430  1072548873           0           0]
 [    -7696581 -1077501381           0           0]
 [ 1255642279 -1078907160           0           0]]
```

```
print()
```

```
for x,y,z in it:
  z[...] = 3*x + y

print(it.operands[2])
```

```
[[ 0  1  2  3]
 [ 3  4  5  6]
 [ 6  7  8  9]
 [ 9 10 11 12]]
```

```
print()
```

```
it = np.nditer([A,B,np.zeros([4,4])])
```

In summary, the values of `it.operands[2]` are initialized with whatever value is available in memory, i.e., any value. Normally they are initialized with the last values assigned to that memory location

Notice now the effect of removing the ellipsis from `z`

```
A = np.arange(0,4,dtype=np.int32).reshape(4,1)
B = np.arange(0,4,dtype=np.int32).reshape(1,4)

it = np.nditer([A,B,None])

print(it.operands[2])
```

```
[[ 7  0 11  0]
 [15  0 19  0]
 [24  0 28  0]
 [32  0 36  0]]
```

```
print()
```

```
for x,y,z in it:
  z = 3*x + y

print(it.operands[2])
```

```
[[ 7  0 11  0]
 [15  0 19  0]
 [24  0 28  0]
 [32  0 36  0]]
```

```
print()
```

```
it = np.nditer([A,B,np.zeros([4,4])])
```

In short the values are lost

### 4.3.3 3rd Example

Let's now see how to implement the equivalent feature of `enumerate` on `numpy` arrays.

First let's recall an example of enumerate. Enumerate makes it easy to synchronize indexes and values in a loop that goes through all the elements of an array.

```
list1 = ["Algeria", "Brazil", "China",\
         "Spain", "Finland"]

for indice, value in enumerate(list1):
  print(indice, value)
```

```
0 Algeria
1 Brazil
2 China
3 Spain
4 Finland
```

Assuming that list is a numpy array we can do:

```
list1 = np.array(["Algeria", "Brazil", "China", \
                  "Spain", "Finland"])

for indice, value in np.ndenumerate(list1):
  print(indice, value)
```

```
(0,) Algeria
(1,) Brazil
(2,) China
(3,) Spain
(4,) Finland
```

Notice that the index came in the format of a numpy array dimension. Let's look at what happens if the numpy array is of type 2D. For convenience we will use an array with only numeric data.

```
z = np.arange(16).reshape(4,4)
print(z)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
```

We see from the example below that enumerate "goes down" only one level in the data structure (either a list or a numpy array) and enumerates the respective indexes and values

```
z = np.arange(16).reshape(4,4)
for index, value in enumerate(z):
  print(index, value)
```

```
0 [0 1 2 3]
1 [4 5 6 7]
2 [ 8  9 10 11]
3 [12 13 14 15]
```

We could have used a nested loop

```
z = np.arange(16).reshape(4,4)
for indice1, value1 in enumerate(z):
  for indice2, value2 in enumerate(value1):
    print(indice1, indice2, value2)
```

```
0 0 0
0 1 1
0 2 2
0 3 3
1 0 4
1 1 5
1 2 6
1 3 7
```

```
2 0 8
2 1 9
2 2 10
2 3 11
3 0 12
3 1 13
3 2 14
3 3 15
```

Or np.nditer(z):

```
z = np.arange(16).reshape(4,4)
for index, value in enumerate(np.nditer(z)):
  print(index, value)
```

```
0 0
1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
10 10
11 11
12 12
13 13
14 14
15 15
```

In both cases to retrieve the address in its original format we would have to use np.unravel_index.

```
z = np.arange(16).reshape(4,4)
for index, value in enumerate(np.nditer(z)):
  index2d = np.unravel_index(index,shape=[4,4])
  print(index2d, value)
```

```
(0, 0) 0
(0, 1) 1
(0, 2) 2
(0, 3) 3
(1, 0) 4
(1, 1) 5
```

```
(1, 2) 6
(1, 3) 7
(2, 0) 8
(2, 1) 9
(2, 2) 10
(2, 3) 11
(3, 0) 12
(3, 1) 13
(3, 2) 14
(3, 3) 15
```

Or simply np.ndenumerate(z), which has the advantage of preserving the original address, without the need for reconversion

```
z = np.arange(16).reshape(4,4)
for index, value in np.ndenumerate(z):
  print(index, value)
```

```
(0, 0) 0
(0, 1) 1
(0, 2) 2
(0, 3) 3
(1, 0) 4
(1, 1) 5
(1, 2) 6
(1, 3) 7
(2, 0) 8
(2, 1) 9
(2, 2) 10
(2, 3) 11
(3, 0) 12
(3, 1) 13
(3, 2) 14
(3, 3) 15
```

## 4.4 Exercises

1. Create a two-axis (2D) array with dimensions `[5,6]` formed by the numbers 0 through 29. In the even-indexed rows the numbers should be arranged in ascending order, and in the odd-indexed rows they should be arranged in descending order. Call this array `x`. Then transform this array into a single-axis array (1D) by calling this second array `y` and determine:
   (a) from `x`:
      i. all elements of the array

ii. insert the value at position 12

iii. the last value

iv. the first value

v. the second-to-last value

vi. the elements from the fourth to the tenth position

vii. all elements from position 7 onwards in positions with even numbers

viii. all elements in reverse order.

ix. the position of the element whose value is 9

x. the position of the element with the smallest value

xi. enter the position of all even elements. Use the remainder operator `enumerate` to exchange all pairs by -1

(b) from `y`:

i. the value in the 1st row and 2nd column

ii. the value in the 3rd row and the last position of the 3rd column

iii. how to change the value in position [0,0] to 12

iv. all values in the last row

v. all values in the last column

vi. the sum of the values in the second column

vii. the sum of the values in the first and third row

viii. all values between the second and third row and from the second to the last column

ix. the 2D position (on two axes, row and column) of the largest element

2. Create two arrays of random numbers. The first should be 1D and contain 10 values between -1 and 1 obtained from a uniform distribution. Call this array `x`. The second should have two axes (2D) of dimensions `[4,6]` and contain 24 values obtained from a normal distribution with mean 10 and standard deviation 2. Call this array `y`. From these two arrays determine:

(a) For the `x` array:

i. The position and value of the element in `x` that is closest to the value `c=0.5` in absolute terms. Hint: use `np.where`

ii. The position and value of all elements smaller than 0

iii. The positions of the five largest values. Tip: use the `.argsort()` method

iv. Round negative values to minus (-1.1 becomes -2) and positive values to plus (1.1 becomes 2). Hint: use `np.ceil` and `np.floor`

v. Determine the result of the operations `np.tile(x,2)` and `np.repeat(x,2)` `number`

(b) For the array `y`:

i. The position in two axes (2D) and the value of the element in `y` that is closest to the value `c=0.5` in absolute terms. Hint: use `np.where`

ii. If any of the columns of `y` is composed only of values less than 10.

iii. How many elements will be obtained by slice `y[[1,2,3],[1,2,3]]`? Answer: 3 (THREE!). The elements at positions `y[1,1]`, `y[2,2]` and `y[3,3]`. This shows that the slice `z[1:4,1:4]` is a bit more complex than two lists of numbers. The slice `z[1:4,1:4]` is actually all the combinations of row addresses `[1,2,3]` with all the respective combinations of column addresses `[1,2,3]`.

iv. How do you change all elements less than 10 by 9 and all elements greater than 10 by 11? Tip: `np.where(condition, value if true, value if false)`

v. The values of the last column in ascending order, changing it with such values.

vi. The values of the second row in descending order, changing it with such values

vii. The average of each column, from last to second column (that is in inverted order)

viii. The sum of each row, from second to last row

ix. How to sort the rows according to the values in the last column. Tip: use the method `.argsort()`

x. The positions of the five largest values. Hint: use the `.argsort()` method

xi. How to place the array `y` beside and on top of a copy of itself. Hint: use `np.r_` and hlnp.c_

xii. How to standardize (decrease from the mean and divide by the standard deviation) the values of `y` by column

3. Create a 2d 5x5 array with the following values

(a) 1 on the edges and 0 on the inside. Assign the values in the center of the array in a single statement

(b) 0 on the edges and the inner part filled with random integers between 1 and 9. Tip: use the `np.pad` function

(c) A random number between 1 and 9 in the center position of the array. Wrap it with 1s and wrap this array ([3,3]) with zeros.

4. From the 1D array `[1,2,3,4,5]` create:

   (a) Using the `np.vstack` function create a 2D array `[3,5]` in which the 1D array is "stacked"horizontally.

   (b) Using the `np.hstack` function create a 2D array `[5,4]` in which the 1D array is placed side by side vertically.

   (c) Repeat the previous exercises using the `np.concatenate` function. Note: `np.concatenate` requires two-axis (2D) arrays.

5. Create a 4x5 array, in which row values should be between 0 and 1 and equally spaced from each other.

6. From a 5x5 matrix of zeros, create a matrix with the values 1,2,3,...: (Hint: use `np.diag()`)

   (a) on the diagonal

   (b) below the diagonal

   (c) above the diagonal

   (d) on the first diagonal to the left of the main diagonal

   (e) on the second diagonal to the left of the main diagonal

7. Create an 8x8 matrix, with and without the use of loops, whose values will be given by the following operations (i is row number and j is column number) or by the given formats:

   (a) i+j

   (b) the remainder of (i+j) / 2

   (c) the pattern of a chessboard (white=0, black=1)

   (d) five equal rows, with the sequence values 0,2,4,...

   (e) five equal columns, with the values of the sequence 1,3,5,...

   (f) with the values of each row equal to i + 3

   (g) with the values of each column equal to j - 3

8. Divide the array `z = np.arange(10)` into three arrays. The middle array should go from position 3 to position 6. Use `np.split`.

9. Transform the array `x = [0,1,2,3,4]` into a 2D array `[n,1]` and the array `y = x + 0.5` into an array `[1,n]`. Compute the difference of each pair of values in the associated `[n,n]` array using: a) `np.vstack` and `np.hstack`, b) `np.concatenate` and c) the value broadcast rules of `numpy`.

## 4.5 Exercise II

1. Suppose 2 sets of points P0,P1 describing lines (2d) and a set of points P. How to calculate the distance from each point j (P[j]) to each line i (P0[i],P1[i])?

```
# Author: Italmassov Kuanysh

def distance(P0, P1, p):
    T = P1 - P0
    L = (T**2).sum(axis=1)
    U = -((P0[:,0]-p[...,0])*T[:,0] + \
          (P0[:,1]-p[...,1])*T[:,1]) / L
    U = U.reshape(len(U),1)
    D = P0 + U*T - p
    return np.sqrt((D**2).sum(axis=1))


P0 = np.random.uniform(-10, 10, (5,2))
P1 = np.random.uniform(-10,10,(5,2))
p = np.random.uniform(-10, 10, (5,2))
np.array([distance(P0,P1,p_i) for p_i in p])
```

```
array([[ 1.887,  0.124,  0.323,  5.385,  2.537],
       [ 9.152,  3.608,  5.63 , 17.152,  4.791],
       [10.584,  1.502,  9.097, 12.092,  0.178],
       [ 3.633,  9.207,  5.328, 14.934, 11.295],
       [ 0.153,  1.51 ,  2.247,  4.651,  0.827]])
```

2. Create a program that can add 1 to each element of a vector, indexed by a second vector. Take into account the case of repeated indexes.

```
# Author: Brett Olsen

Z = np.ones(10)
I = np.random.randint(0,len(Z),20)
Z += np.bincount(I, minlength=len(Z))
print(Z)

# Another solution
# Author: Bartosz Telenczuk
```

```
[2. 5. 3. 1. 2. 4. 5. 2. 3. 3.]
```

```
np.add.at(Z, I, 1)
print(Z)
```

```
[3. 9. 5. 1. 3. 7. 9. 3. 5. 5.]
```

3. How to accumulate the elements of a vector (x) into an array (F) based on a list of indices (I)?

```
# Author: Alan G Isaac

X = [1,2,3,4,5,6]
I = [1,3,9,3,4,1]
F = np.bincount(I,X)
print(F)
```

```
[0. 7. 0. 6. 5. 0. 0. 0. 0. 3.]
```

4. Suppose an image `(w,h,3)` of type `(dtype=ubyte)`. Calculate the number of unique colors of it. Hint: Calculate the result of 256*256 first. Otherwise numpy will make dtype F equal to "uint16"and this will cause an overfolw in the sequence.

```
# Author: Nadav Horesh

w,h = 16,16
I = np.random.randint(0,2,(h,w,3)).astype(np.ubyte)
F = I[...,0]*(256*256) + I[...,1]*256 +I[...,2]
n = len(np.unique(F))
print(n)
```

```
8
```

5. Suppose a four-dimensional array. Create a program to get the sum along the last two axes at once.

Solution by passing a tuple of axes (valid as of numpy 1.7.0).

```
A = np.random.randint(0,10,(3,4,3,4))
sum = A.sum(axis=(-2,-1))
print(sum)
```

```
[[48 69 45 46]
 [51 40 63 58]
 [66 51 53 49]]
```

Solution by condensing (flattening) the last two dimensions into a single dimension. This solution is useful for functions that do not yet accept tuples as arguments for creating and handling axes.

```
sum = A.reshape(A.shape[:-2] + (-1,)).sum(axis=-1)
print(sum)
```

```
[[48 69 45 46]
 [51 40 63 58]
 [66 51 53 49]]
```

6. Suppose a 1D vector D. Compute the average of subsets of the vector using a vector S of the same size that describes the indices of each subset.

Solution with numpy arrays only

```
# Author: Jaime Fernandez del Rio

D = np.random.uniform(0,1,100)
S = np.random.randint(0,10,100)
D_sums = np.bincount(S, weights=D)
D_counts = np.bincount(S)
D_means = D_sums / D_counts
print(np.round(D_means,2))
```

```
[0.55 0.44 0.48 0.45 0.72 0.54 0.49 0.46 0.4  0.56]
```

Solution using panda dataframes

```
import pandas as pd
print(pd.Series(D).groupby(S).mean())
```

```
0    0.552089
1    0.435813
2    0.477324
3    0.454061
4    0.718146
5    0.544538
6    0.494873
7    0.463283
8    0.399532
9    0.561716
dtype: float64
```

7. How to get the diagonal of a matrix product?

```
# Author: Mathieu Blondel

A = np.random.uniform(0,1,(5,5))
B = np.random.uniform(0,1,(5,5))
```

Slow version

```
np.diag(np.dot(A, B))
```

```
array([1.319, 1.092, 1.642, 1.103, 1.792])
```

Regular speed version

```
np.sum(A * B.T, axis=1)
```

```
array([1.319, 1.092, 1.642, 1.103, 1.792])
```

Faster version

```
np.einsum("ij,ji->i", A, B)
```

```
array([1.319, 1.092, 1.642, 1.103, 1.792])
```

8. Suppose the vector [1, 2, 3, 4, 5]. Create a new vector with 3 zeros in sequence interspersed between each value of the vector.

```
# Author: Warren Weckesser

Z = np.array([1,2,3,4,5])
nz = 3
Z0 = np.zeros(len(Z) + (len(Z)-1)*(nz))
Z0[::nz+1] = Z
print(Z0)
```

```
[1. 0. 0. 0. 2. 0. 0. 0. 3. 0. 0. 0. 4. 0. 0. 0. 5.]
```

9. Create a program that can swap two rows of position with each other in an array.

```
# Author: Eelco Hoogendoorn

A = np.arange(25).reshape(5,5)
A[[0,1]] = A[[1,0]]
print(A)
```

```
[[ 5  6  7  8  9]
 [ 0  1  2  3  4]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
```

10. Suppose a set of 5 triples of values describing 5 triangles (which are connected by their vertices. Find the set of unique line segments that make up the set of triangles.

```
# Author: Nicolas P. Rougier

faces = np.random.randint(0,100,(5,3))
F = np.roll(faces.repeat(2,axis=1),-1,axis=1)
F = F.reshape(len(F)*3,2)
F = np.sort(F,axis=1)
G = F.view( dtype=[("p0",F.dtype),("p1",F.dtype)] )
G = np.unique(G)
print(G)
```

```
[(12, 41) (12, 50) (12, 88) (12, 90) (30, 76) (30, 96) (31, 65) (31, 84)
 (41, 90) (49, 61) (49, 94) (50, 88) (61, 94) (65, 84) (76, 96)]
```

## 4.6 Exercise III

1. Create dummy variables for each unique value in the following array. In English this operation is called calculating one-hot encodings.

```

np.random.seed(101)
arr = np.random.randint(1,4, size=6)
arr
#> array([2, 3, 2, 2, 1])
# Solution
```

```
array([2, 3, 2, 2, 2, 1])
```

```
def one_hot_encodings(arr):
  uniqs = np.unique(arr)
  out = np.zeros((arr.shape[0], uniqs.shape[0]))
  for i, k in enumerate(arr):
    out[i, k-1] = 1
  return out

print(one_hot_encodings(arr))
```

```
[[0. 1. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [1. 0. 0.]]
```

```
print("")

# Method 2:
```

```
print((arr[:, None] == np.unique(arr)).view(np.int8))
```

```
[[0 1 0]
 [0 0 1]
 [0 1 0]
 [0 1 0]
 [0 1 0]
 [1 0 0]]
```

2. Given a numeric array, calculate its rank.

```
np.random.seed(10)
a = np.random.randint(20, size=10)
print('Array: ', a)
# Solution
```

```
Array:  [ 9  4 15  0 17 16 17  8  9  0]
```

```
print(a.argsort().argsort())
```

```
[4 2 6 0 8 7 9 3 5 1]
```

```
print('Array: ', a)
```

```
Array:  [ 9  4 15  0 17 16 17  8  9  0]
```

3. Create a rank array of the same format as a given numeric array a.

```
np.random.seed(10)
a = np.random.randint(20, size=[2,5])
print(a)
# Solution
```

```
[[ 9  4 15  0 17]
 [16 17  8  9  0]]
```

```
print(a.ravel().argsort().argsort().reshape(a.shape))
```

```
[[4 2 6 0 8]
 [7 9 3 5 1]]
```

4. Calculate the maximum value in each row of an array.

```
np.random.seed(100)
a = np.random.randint(1,10, [5,3])
a
# Solution 1
```

```
array([[9, 9, 4],
       [8, 8, 1],
       [5, 3, 6],
       [3, 3, 3],
       [2, 1, 9]])
```

```
print(np.amax(a, axis=1))
# Solution 2
```

```
[9 8 6 3 9]
```

```
print(np.apply_along_axis(np.max, arr=a, axis=1))
```

```
[9 8 6 3 9]
```

5. Calculate the min-by-max for each row of a 2d array

```
np.random.seed(100)
a = np.random.randint(1,10, [5,3])
a
# Solution
```

```
array([[9, 9, 4],
       [8, 8, 1],
       [5, 3, 6],
       [3, 3, 3],
       [2, 1, 9]])
```

```
f = lambda x: np.min(x)/np.max(x)
print(np.apply_along_axis(f, arr=a, axis=1))
```

```
[0.444 0.125 0.5   1.    0.111]
```

6. Find the duplicate entries in a given numpy array and mark them as True. The first occurrence of each value should be set to False.

```
np.random.seed(100)
a = np.random.randint(0, 5, 10)
## Solution
out = np.full(a.shape[0], True)

## Finding the positions of unique elements
unique_positions = np.unique(a, return_index=True)[1]

# marking the positions as False
out[unique_positions] = False
print(out)
```

```
[False  True False  True False False  True  True
True  True]
```

7. Delete all nan values from 1D array

```
a = np.array([1,2,3,np.nan,5,6,7,np.nan])
print(a[~np.isnan(a)])
```

```
[1. 2. 3. 5. 6. 7.]
```

8. Find all local maxima in a 1D array. Local maximums in an array are defined as values that are surrounded by smaller numbers to the right and left.

```
a = np.array([1, 3, 7, 1, 2, 6, 0, 1])
doublediff = np.diff(np.sign(np.diff(a)))
peak_locations = np.where(doublediff == -2)[0] + 1
print(peak_locations)
```

```
[2 5]
```

9. Calculate the moving average of size 3 for a 1D array

```
# Solution
# Source: https://stackoverflow.com/questions/14313510/
# how-to-calculate-moving-average-using-numpy

def moving_average(a, n=3) :
```

```
  ret = np.cumsum(a, dtype=float)
  ret[n:] = ret[n:] - ret[:-n]
  return ret[n - 1:] / n

np.random.seed(100)
Z = np.random.randint(10, size=10)
print('array: ', Z)
```

```
array:  [8 8 3 7 7 0 4 2 5 2]
```

```
# method 1
moving_average(Z, n=3).round(1)
```

```
array([6.3, 6. , 5.7, 4.7, 3.7, 2. , 3.7, 3. ])
```

```
# method 2: # Source: AlanLRH!
# np.ones(3)/3 gives equal weights.
# Use np.ones(4)/4 for window size 4.
z1 = np.ones(3)/3
print('moving average:', np.convolve(Z, z1, mode="valid"))
```

```
moving average: [6.333 6.    5.667 4.667 3.667 2.
3.667 3.   ]
```

10. Given an array with a sequence of non-consecutive dates, make it continuous by inserting the missing dates into the array.

```

dates = np.arange(np.datetime64('2018-02-01'),
                  np.datetime64('2018-02-25'), 2)
print(dates)
```

```
['2018-02-01' '2018-02-03' '2018-02-05' '2018-02-07' '2018-02-09'
 '2018-02-11' '2018-02-13' '2018-02-15' '2018-02-17' '2018-02-19'
 '2018-02-21' '2018-02-23']
```

```
# Solution ---------------
filled_in = np.array([np.arange(date, (date+d))
            for date, d in zip(dates, np.diff(dates))])

filled_in = filled_in.reshape(-1)
```

```
# including the last day
output = np.hstack([filled_in, dates[-1]])
print(output)
```

```
['2018-02-01' '2018-02-02' '2018-02-03' '2018-02-04' '2018-02-05'
 '2018-02-06' '2018-02-07' '2018-02-08' '2018-02-09' '2018-02-10'
 '2018-02-11' '2018-02-12' '2018-02-13' '2018-02-14' '2018-02-15'
 '2018-02-16' '2018-02-17' '2018-02-18' '2018-02-19' '2018-02-20'
 '2018-02-21' '2018-02-22' '2018-02-23']
```

```
# Version with loops
out = []
for date, d in zip(dates, np.diff(dates)):
  out.append(np.arange(date, (date+d)))

filled_in = np.array(out).reshape(-1)

# including the last day
output = np.hstack([filled_in, dates[-1]])
print(output)
```

```
['2018-02-01' '2018-02-02' '2018-02-03' '2018-02-04' '2018-02-05'
 '2018-02-06' '2018-02-07' '2018-02-08' '2018-02-09' '2018-02-10'
 '2018-02-11' '2018-02-12' '2018-02-13' '2018-02-14' '2018-02-15'
 '2018-02-16' '2018-02-17' '2018-02-18' '2018-02-19' '2018-02-20'
 '2018-02-21' '2018-02-22' '2018-02-23']
```

11. Given a 1D array, generate a 2D array using the concept of strides, with a window length of 4 and strides of 2. The final result should look like `[ [0,1,2,3], [2,3,4,5], [4,5,6,7]...]`

```
def gen_strides(a, stride_len=5, window_len=5):
  n_strides = ((a.size-window_len)//stride_len) + 1
  sL = stride_len
  return np.array([a[s:(s+window_len)]
           for s in np.arange(0, n_strides*sL,sL)])

z = np.arange(15)
print(gen_strides(z, stride_len=2, window_len=4))
```

```
[[ 0  1  2  3]
 [ 2  3  4  5]
 [ 4  5  6  7]
 [ 6  7  8  9]
 [ 8  9 10 11]
 [10 11 12 13]]
```

12. Calculate by bootstrapping, 95

```
# Author: Jessica B. Hamrick

X = np.random.randn(100) # random 1D array
N = 1000 # number of bootstrap samples
idx = np.random.randint(0, X.size, (N, X.size))
means = X[idx].mean(axis=1)
confint = np.percentile(means, [2.5, 97.5])
print(confint)
```

```
[-0.268  0.131]
```

13. Calculate the counts of unique values in a 2D array, along its rows.

```
np.random.seed(100)
arr = np.random.randint(1,11,size=(6, 10))
print(arr)
```

```
[[ 9  9  4  8  8  1  5  3  6  3]
 [ 3  3  2  1  9  5  1 10  7  3]
 [ 5  2  6  4  5  5  4  8  2  2]
 [ 8  8  1  3 10 10  4  3  6  9]
 [ 2  1  8  7  3  1  9  3  6  2]
 [ 9  2  6  5  3  9  4  6  1 10]]
```

Solution

```
# Solution
def counts_of_all_values_rowwise(arr2d):
  # Unique values and its counts row wise
  temp = [np.unique(row, return_counts=True) for row in arr2d]
  num_counts_array = temp
  # Counts of all values row wise
  unicos = np.unique(arr2d)
  return([[int(b[a==i]) if i in a else 0 for i in unicos]
                              for a, b in num_counts_array])
# Print
print(np.arange(1,11))
```

```
[ 1  2  3  4  5  6  7  8  9 10]
```

```
print(np.array(counts_of_all_values_rowwise(arr)))
```

```
[[1 0 2 1 1 1 0 2 2 0]
 [2 1 3 0 1 0 1 0 1 1]
 [0 3 0 2 3 1 0 1 0 0]
```

```
 [1 0 2 1 0 1 0 2 1 2]
 [2 2 2 0 0 1 1 1 1 0]
 [1 1 1 1 1 2 0 0 2 1]]
```

Example 2

```
arr = np.array([np.array(list('bill clinton')),
                np.array(list("narendramodi")),
                np.array(list("jjayalalitha"))])
print(np.unique(arr))
```

```
[' ' 'a' 'b' 'c' 'd' 'e' 'h' 'i' 'j' 'l' 'm' 'n' 'o' 'r' 't' 'y']
```

```
print(np.array(counts_of_all_values_rowwise(arr)))
```

```
[[1 0 1 1 0 0 0 2 0 3 0 2 1 0 1 0]
 [0 2 0 0 2 1 0 1 0 0 1 2 1 2 0 0]
 [0 4 0 0 0 0 1 1 2 2 0 0 0 0 1 1]]
```

14. Suppose a wooden bar is cut at two random points. What is the probability that the three resulting pieces of the bar will produce a triangle?

```
import numpy as np

def triangle():
  L1 = np.random.uniform()
  x2 = np.random.uniform(L1,1)
  L2 = x2 - L1
  L3 = 1 - x2

  d = False
  if L1 < L2 + L3:
    if L2 < L1 + L3:
      if L3 < L1 + L2:
        if L1 > np.abs(L2 - L3):
          if L2 > np.abs(L1 - L3):
            if L3 > np.abs(L1 - L2):
              d = True
  return(d)

def prob(n):
  sum = 0
  for i in range(n):
    sum = sum + triangle()
  result = sum/n
  return(result)
```

```
prob(100000)
```

```
0.19278
```

15. Given an array c that is a *bincount*, how do you create an array A such that np.bincount(A) == C?

```
# Author: Jaime Fernandez del Rio

C = np.bincount([1,1,2,3,4,6])
A = np.repeat(np.arange(len(C)), C)
print(A)
```

```
[1 1 2 3 4 6]
```

16. How do I calculate averages from a moving data window that runs through an array?

```
# Author: Jaime Fernandez del Rio

def moving_average(a, n=3) :
    ret = np.cumsum(a, dtype=float)
    ret[n:] = ret[n:] - ret[:-n]
    return ret[n - 1:] / n
Z = np.arange(10)
print(moving_average(Z, n=3))
```

```
[1. 2. 3. 4. 5. 6. 7. 8.]
```

17. Assume a 1D array z. From it create a 2D array whose first row is `(Z[0],Z[1],Z[2])` and each subsequent row is shifted by one unit. The last row in this case should be `(Z[-3],Z[-2],Z[-1])`.

```
# Author: Joe Kington / Erik Rigtorp
from numpy.lib import stride_tricks

def rolling(a, window):
    shape = (a.size - window + 1, window)
    strides = (a.itemsize, a.itemsize)
    return stride_tricks.as_strided(a, shape=shape, \
                                    strides=strides)
Z = rolling(np.arange(10), 3)
print(Z)
```

```
[[0 1 2]
 [1 2 3]
 [2 3 4]
 [3 4 5]
 [4 5 6]
 [5 6 7]
 [6 7 8]
 [7 8 9]]
```

18. How to negate a boolean. How to change the sign of a given float type without copying (i.e. `inplace`)?

```
# Author: Nathaniel J. Smith

Z = np.random.randint(0,2,5)
print(np.logical_not(Z, out=Z))
```

```
[0 0 0 0 1]
```

```
Z = np.random.uniform(-1.0,1.0,5)
print(np.round(np.negative(Z, out=Z),4))
```

```
[ 0.458  0.074 -0.983  0.093  0.791]
```

19. Suppose an arbitrary array. Create a function that will extract a subset of it with a predefined format and center on a specific element. Add fixed values with 'fill' where necessary).

```
# Author: Nicolas Rougier

Z = np.random.randint(0,10,(5,5))
shape = (2,2)
fill  = 0
position = (1,1)

R = np.ones(shape, dtype=Z.dtype)*fill
P  = np.array(list(position)).astype(int)
Rs = np.array(list(R.shape)).astype(int)
Zs = np.array(list(Z.shape)).astype(int)

R_start = np.zeros((len(shape),)).astype(int)
R_stop  = np.array(list(shape)).astype(int)
Z_start = (P-Rs//2)
Z_stop  = (P+Rs//2)+Rs%2
```

```
R_start = (R_start - np.minimum(Z_start,0)).tolist()
Z_start = (np.maximum(Z_start,0)).tolist()

zmax = np.maximum(Z_stop-Zs,0)
R_stop = np.maximum(R_start, (R_stop - zmax))
R_stop = R_stop.tolist()
Z_stop = (np.minimum(Z_stop,Zs)).tolist()

Rzip = zip(R_start,R_stop)
Zzip = zip(Z_start,Z_stop)
r = [slice(start,stop) for start,stop in Rzip]
z = [slice(start,stop) for start,stop in Zzip]
R[r] = Z[z]
print(Z)
```

```
[[4 9 2 1 6]
 [4 2 3 7 9]
 [1 0 0 2 3]
 [4 6 0 3 2]
 [9 7 1 6 9]]
```

```
print(R)
```

```
[[4 9]
 [4 2]]
```

20. Suppose an array `Z = [1,2,3,4,5,6,7,8,9,10,11,12,13,14]`. Create an array `R = [[1,2,3,4], [2,3,4,5], [3,4,5,6], ..., [11,12,13,14]]`.

```
# Author: Stefan van der Walt

Z = np.arange(1,15,dtype=np.uint32)
R = stride_tricks.as_strided(Z,(11,4),(4,4))
print(R)
```

```
[[ 1  2  3  4]
 [ 2  3  4  5]
 [ 3  4  5  6]
 [ 4  5  6  7]
 [ 5  6  7  8]
 [ 6  7  8  9]
 [ 7  8  9 10]
 [ 8  9 10 11]
 [ 9 10 11 12]
 [10 11 12 13]
 [11 12 13 14]]
```

21. Get the first 3x3 continuous blocks of a 5x5 random matrix.

```
# Author: Chris Barker

Z = np.random.randint(0,5,(5,5))
n = 3
i = 1 + (Z.shape[0]-3)
j = 1 + (Z.shape[1]-3)
Z1 = Z.strides + Z.strides
s = (i, j, n, n)
C = stride_tricks.as_strided(Z, shape=s, strides=Z1)
print(C[0])
```

```
[[[0 4 4]
  [0 0 1]
  [1 0 3]]

 [[4 4 1]
  [0 1 3]
  [0 3 2]]

 [[4 1 2]
  [1 3 4]
  [3 2 2]]]
```

22. Create a subclass of the 2D arrays such that Z[i,j] == Z[j,i]. The functionality will only need to work for 2D arrays with value assignment via indices.

```
# Author: Eric O. Lebigot

class Symetric(np.ndarray):
  def __setitem__(self, index, value):
      i,j = index
      super(Symetric, self).__setitem__((i,j), value)
      super(Symetric, self).__setitem__((j,i), value)

def symetric(Z):
  Z1 = np.asarray(Z + Z.T - np.diag(Z.diagonal()))
  return Z1.view(Symetric)

S = symetric(np.random.randint(0,10,(5,5)))
S[2,3] = 42
print(S)
```

```
[[ 9 15 17  6 12]
 [15  5  2  6  9]
```

```
 [17  2  6 42  9]
 [ 6  6 42  5 10]
 [12  9  9 10  6]]
```

23. Suppose a set of matrices with shape (n,n) and a set of p vectors with shape (n,1). How to calculate the sum of p products of matrices at once, such that the result has shape (n,1).

The following function will work since: - M is (p,n,n) - V is (p,n,1) - Therefore, calculating the sum along the paired axes 0 and 0 (of M and V independently) and axes 2 and 1 ends up with a vector (n,1).

```
# Author: Stefan van der Walt

p, n = 5, 10
M = np.ones((p,n,n))
V = np.ones((p,n,1))
S = np.tensordot(M, V, axes=[[0, 2], [0, 1]])
print(S)
```

```
[[50.]
 [50.]
 [50.]
 [50.]
 [50.]
 [50.]
 [50.]
 [50.]
 [50.]
 [50.]]
```

24. Suppose a 16x16 array. How do you get the sum of the 4x4 blocks that make up the array?

```
# Author: Robert Kern

Z = np.ones((16,16))
k = 4

Z0 = np.arange(0, Z.shape[0], k)
Z1 = np.arange(0, Z.shape[1], k)
S = np.add.reduceat(np.add.reduceat(Z, Z0, axis=0,),
                                       Z1, axis=1)
print(S)
```

```
[[16. 16. 16. 16.]
 [16. 16. 16. 16.]
```

```
 [16. 16. 16. 16.]
 [16. 16. 16. 16.]]
```

25. Implement the Game of Life using only numpy arrays.

```
# Author: Nicolas Rougier

def iterate(Z):
  # count neighbours
  N = (Z[0:-2,0:-2] + Z[0:-2,1:-1] + Z[0:-2,2:] +
       Z[1:-1,0:-2] + Z[1:-1,2:] +
       Z[2: ,0:-2] + Z[2: ,1:-1] + Z[2: ,2:])

  # Apply rules
  birth = (N==3) & (Z[1:-1,1:-1]==0)
  survive = ((N==2) | (N==3)) & (Z[1:-1,1:-1]==1)
  Z[...] = 0
  Z[1:-1,1:-1][birth | survive] = 1
  return(Z)

Z = np.random.randint(0,2,(50,50))
for i in range(100): Z = iterate(Z)
print(Z)
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

26. Get the n largest values in an array.

```
Z = np.arange(10000)
np.random.shuffle(Z)
n = 5
```

Slow form

```
print (Z[np.argsort(Z)[-n:]])
```

```
[9995 9996 9997 9998 9999]
```

Fast form

```
print (Z[np.argpartition(-Z,n)[:n]])
```

```
[9999 9998 9997 9996 9995]
```

27. Given an arbitrary number of vectors, compute their Cartesian product, that is the combination of all vectors two by two.

```
# Author: Stefan Van der Walt

def cartesiano(arrays):
  arrays = [np.asarray(a) for a in arrays]
  shape = [len(x) for x in arrays]

  ix = np.indices(shape, dtype=int)
  ix = ix.reshape(len(arrays), -1).T

  for n, arr in enumerate(arrays):
      ix[:, n] = arrays[n][ix[:, n]]

  return(ix)

print(cartesiano(([1, 2, 3], [4, 5], [6, 7])))
```

```
[[1 4 6]
 [1 4 7]
 [1 5 6]
 [1 5 7]
 [2 4 6]
 [2 4 7]
 [2 5 6]
 [2 5 7]
 [3 4 6]
 [3 4 7]
 [3 5 6]
 [3 5 7]]
```

28. How to create an array of records (a record array) from an ordinary array?

```
Z = np.array([("Hello", 2.5, 3),
              ("World", 3.6, 2)])
R = np.core.records.fromarrays(Z.T,
                   names='col1, col2, col3',
                   formats = 'S8, f8, i8')
print(R)
```

```
[(b'Hello', 2.5, 3) (b'World', 3.6, 2)]
```

29. Suppose two arrays A and B of dimensions (8,3) and (2,2). Get the rows of A that contain elements present in every row of B, regardless of their order in B.

```
# Author: Gabe Schwartz

A = np.random.randint(0,5,(8,3))
B = np.random.randint(0,5,(2,2))

C = (A[..., np.newaxis, np.newaxis] == B)
rows = np.where(C.any((3,1)).all(1))[0]
print(rows)
```

```
[0 1 2 3 4]
```

30. Suppose a 10x3 matrix. Get the rows that contain only unique values.(e.g.[2,2,3])

    Solution for arrays of all types (including string arrays and record arrays).

```
# Author: Robert Kern

Z = np.random.randint(0,5,(10,3))
print(Z)
```

```
[[2 2 1]
 [3 1 2]
 [1 3 3]
 [0 3 3]
 [1 4 1]
 [3 0 0]
 [1 1 3]
 [4 1 2]
 [4 1 2]
 [4 0 4]]
```

```
E = np.all(Z[:,1:] == Z[:,:-1], axis=1)
U = Z[~E]
print(U)
```

```
[[2 2 1]
 [3 1 2]
 [1 3 3]
 [0 3 3]
 [1 4 1]
 [3 0 0]
 [1 1 3]
```

```
 [4 1 2]
 [4 1 2]
 [4 0 4]]
```

Solution for numeric arrays. This solution works for any number of columns in z

```
U = Z[Z[Z.max(axis=1) != Z.min(axis=1),:]]
print(U)
```

```
[[[1 3 3]
  [1 3 3]
  [3 1 2]]

 [[0 3 3]
  [3 1 2]
  [1 3 3]]

 [[3 1 2]
  [0 3 3]
  [0 3 3]]

 [[2 2 1]
  [0 3 3]
  [0 3 3]]

 [[3 1 2]
  [1 4 1]
  [3 1 2]]

 [[0 3 3]
  [2 2 1]
  [2 2 1]]

 [[3 1 2]
  [3 1 2]
  [0 3 3]]

 [[1 4 1]
  [3 1 2]
  [1 3 3]]

 [[1 4 1]
  [3 1 2]
  [1 3 3]]

 [[1 4 1]
```

```
  [2 2 1]
  [1 4 1]]]
```

31. Convert a vector of integers to a binary array representation.

```
# Author: Warren Weckesser

I = np.array([0, 1, 2, 3, 15, 16, 32, 64, 128])
B = (I.reshape(-1,1) & (2**np.arange(8)) != 0)
B = B.astype(int)
print(B[:,::-1])
```

```
[[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 1 1]
 [0 0 0 0 1 1 1 1]
 [0 0 0 1 0 0 0 0]
 [0 0 1 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]]
```

```
# Author: Daniel T. McDonald

I = np.array([0, 1, 2, 3, 15, 16, 32, 64, 128])
I = np.array(I, dtype=np.uint8)
print(np.unpackbits(I[:, np.newaxis], axis=1))
```

```
[[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 1 1]
 [0 0 0 0 1 1 1 1]
 [0 0 0 1 0 0 0 0]
 [0 0 1 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]]
```

32. Given a two-dimensional array, how to extract single rows from it?

```
# Author: Jaime Fernandez del Rio

Z = np.random.randint(0,2,(6,3))
Z1 = np.ascontiguousarray(Z)
Z2 = Z.dtype.itemsize * Z.shape[1]
T = Z1.view(np.dtype((np.void, Z2)))
```

```
_, idx = np.unique(T, return_index=True)
uZ = Z[idx]
print(uZ)
```

```
[[1 0 0]
 [1 0 1]
 [1 1 0]
 [1 1 1]]
```

33. Suppose two vectors A & B, write the equivalent einstein sum (or sum in tensor notation) for the numpy methods: .inner(), .outer(), .sum(), and .mul().

    Read the reference: http://ajcr.net/Basic-guide-to-einsum/

```
# Author: Alex Riley

A = np.random.uniform(1,2,5)
B = np.random.uniform(1,2,5)

# np.sum(A)
print(np.einsum('i->', A))

# A * B
```

```
7.7761001287961315
```

```
print(np.einsum('i,i->i', A, B))

# np.inner(A, B)
```

```
[1.604 1.716 2.305 2.75  1.267]
```

```
print(np.einsum('i,i', A, B))

# np.outer(A, B)
```

```
9.642733771091653
```

```
print(np.einsum('i,j->ij', A, B))
```

```
[[1.604 1.683 1.886 2.766 1.675]
 [1.636 1.716 1.923 2.821 1.708]
 [1.961 2.057 2.305 3.381 2.047]
 [1.595 1.673 1.875 2.75  1.665]
 [1.214 1.274 1.427 2.093 1.267]]
```

34. From the path described by two vectors (X,Y), how to sample it using only equidistant samples?

```
# Author: Bas Swinckels

phi = np.arange(0, 10*np.pi, 0.1)
a = 1
x = a*phi*np.cos(phi)
y = a*phi*np.sin(phi)

# segment lengths
dr = (np.diff(x)**2 + np.diff(y)**2)**.5
r = np.zeros_like(x)

# integrate path
r[1:] = np.cumsum(dr)

# regular spaced path
r_int = np.linspace(0, r.max(), 200)

# integrate path
x_int = np.interp(r_int, r, x)
y_int = np.interp(r_int, r, y)
```

35. Given an integer n and a 2D array x, select from x the rows that can be interpreted as obtained from a multinomial distribution with n degrees of freedom, that is, the rows that contain only integers and which sum to n.

```
# Author: Evgeni Burovski

X = np.asarray([[1.0, 0.0, 3.0, 8.0],
                [2.0, 0.0, 1.0, 1.0],
                [1.5, 2.5, 1.0, 0.0]])
n = 4
M = np.logical_and.reduce(np.mod(X, 1) == 0, axis=-1)
M &= (X.sum(axis=-1) == n)
print(X[M])
```

```
[[2. 0. 1. 1.]]
```

36. Create a two-axis (2D) random vector with dimensions `[5,2]`, consisting of integer values from -9 to 9. Suppose that each row of the vector is the x,y coordinates of a point. Compute the matrix with the distances between all pairs of points.

```
np.random.seed(100)
z = np.random.randint(-9,9,(5,2))
```

```
x1 = z[:,0].reshape(-1,1); x2 = x1.T
y1 = z[:,1].reshape(-1,1); y2 = y1.T
print( np.sqrt((x1-x2)**2 + (y1-y2)**2) )
```

```
[[ 0.     12.042 10.63   6.083 12.53 ]
 [12.042  0.     10.296 13.928  5.099]
 [10.63  10.296  0.     16.125 14.56 ]
 [ 6.083 13.928 16.125  0.     12.   ]
 [12.53   5.099 14.56  12.     0.   ]]
```

37. From the sample iris.target dataset obtained with: `np.sort(np.random.choice(iris.target, size=15)` count the row numbers per categorical variable. For R users: we are implementing count by group_by.

```
from sklearn.datasets import load_iris
iris = load_iris()
np.random.seed(100)
sample = np.random.choice(iris.target, size=15)
species_small = np.sort(sample)
sp_sm = species_small
print(sp_sm)
```

```
[0 0 0 0 0 1 1 1 1 1 1 1 2 2 2]
```

```
print([i for val in np.unique(sp_sm)
         for i, grp in enumerate(sp_sm[sp_sm==val])])
```

```
[0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2]
```

38. Create group ids from a categorical variable, using the species_small dataset (below) as an example.

```
species = iris.target
species_small = np.sort(np.random.choice(species, size=15))
species_small
```

```
array([0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
# Solution:
uniques = np.unique(species_small)
output = [np.argwhere(uniques == s).tolist()[0][0] \
    for val in species_small
        for s in species_small[species_small==val]]
```

```
# Solution: For Loop version
output = []
uniqs = np.unique(species_small)
# uniq values in group
for val in uniqs:
  # each element in group
  for s in species_small[species_small==val]:
    # groupid
    groupid = np.argwhere(uniqs == s).tolist()[0][0]
    output.append(groupid)

print(output)
```

```
[0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2]
```

39. Calculate the average of a numeric column grouped by a categorical column in a 2D array

```
iris_2d = iris.data
iris_rsp = iris.target.reshape(-1,1)
iris_2d = np.concatenate([iris_2d, iris_rsp], axis=1)
names = ("sepallength", "sepalwidth", "petallength", \
         "petalwidth", "species")

# Solution
# sepalwidth
numeric_column = iris_2d[:, 1].astype("float")
# species
grouping_column = iris_2d[:, 4]
```

```
# Solution using list comprehension
gp_col = grouping_column
gp_unq = np.unique(grouping_column)
# gp_val is~ short for group_val
sol = [[gp_val, numeric_column[gp_col==gp_val].mean()]
                                for gp_val in gp_unq]
print(np.round(sol,3))
```

```
[[0.    3.428]
 [1.    2.77 ]
 [2.    2.974]]
```

```
# Solution using for loop
output = []
gp_col = grouping_column
```

```
for group_val in np.unique(grouping_column):
  media = numeric_column[gp_col==group_val].mean()
  output.append([group_val, media])

print(np.round(output,3))
```

```
[[0.    3.428]
 [1.    2.77 ]
 [2.    2.974]]
```

40. Create a random sampling of the iris.data dataset indicating which rows of it should be chosen, so that the amount of samples of the setosa species is twice the amount of the versicolor and virginica species.

```
np.random.seed(100)
np.set_printoptions(precision=3)
probs = np.r_[np.linspace(0, 0.500, num=50),
              np.linspace(0.501, .750, num=50),
              np.linspace(.751, 1.0, num=50)]
print(np.round(probs[:10]),2)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] 2
```

```
index = np.searchsorted(probs, np.random.random(150))
print(index[:10])
```

```
[ 59  28  42 119   1  12  84 115  14  65]
```

```
species_out = iris.target[index]
print(species_out[:10])
```

```
[1 0 0 2 0 0 1 2 0 1]
```

```
print(np.unique(species_out, return_counts=True))
```

```
(array([0, 1, 2]), array([77, 37, 36]))
```

41. Import an image available in a URL and convert it to a numpy array. In the solution a wikipedia URL is used.

```
from io import BytesIO
from PIL import Image
import PIL, requests

# Import data from the Internet
```

```
URL = 'https://upload.wikimedia.org/wikipedia/"
URL = URL + 'commons/8/8b/Denali_Mt_McKinley.jpg'
response = requests.get(URL)

# Reading the data as an image
I = Image.open(BytesIO(response.content))

# Size Adjustment
I = I.resize([150,150])

# Converting to a numpy array
arr = np.asarray(I)

# You can also do the inverse conversion
# (array -> image) and display the same.
im = PIL.Image.fromarray(np.uint8(arr))
#Image.Image.show(im)
```

# Capítulo 5

# Functions

This chapter will explore the basic calculus and graphing capabilities of Python, through the `numpy` and `sympy` packages and `matplotlib`. Function computation will involve practical examples and basic graph plotting using the `Python` language will be introduced.

## 5.1 Mathematical Models

A mathematical model is a numerical description of a real phenomenon. Examples of mathematical models would be: a) a description of the number of people in the population of a country over time, b) the quantity of products demanded by a group of customers, of products offered by a set of firms, and the amount of price at which both groups will be willing to execute a business transaction, and c) the rate of return on an investment at which we are willing to invest in it.

The process of creating a model involves the steps of: a) determining the variables to be studied, b) obtaining actual values for these variables, c) proposing a relationship between them, d) running the model, e) analyzing the results, and f) presenting the results obtained. In this text we will be mainly interested in the formulation of models and their execution both algebraically (manually) and numerically (by the computer).

## 5.2 Algebraic Representation of a Function

Functions are expressions in which operations are performed on the value of one variable (e.g. $x$) and then the value of another variable (e.g. $y$) is obtained by saying that $y$ is a function of $x$ and representing this relationship generically by $y = f(x)$.

The concept of mathematical model is closely related to that of function. In

both cases we perform mathematical operations on a set of given values in order to obtain another set of values. The elements that constitute a function $y = f(x)$ are:

1. An input variable (also called independent variable) usually represented by $x$
2. A set of mathematical operations to be performed on the input variable $x$, which are represented implicitly in the symbol $f()$
3. An output variable (or response variable, also called dependent variable), usually represented by $y$
4. The set of possible values for the input variable $x$ is called the function domain and the set of possible values for the output variable $y$ is called the function image.

## 5.3 Constant Function

The first function we will look at is the constant function, for example $y = f(x) = 4$. Note that in this case, for any value chosen for $x$ the value of the function does not change. The graphical representation of it is a horizontal line (i.e. parallel to the $x$-axis, see 5.1)

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-10,10,20)
y = x - x + 4
plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D object at 0x7f8697d422e0>]
```

```
plt.show()
```

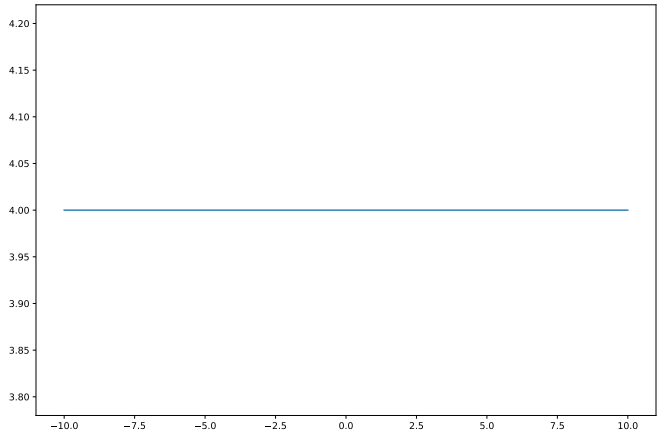



Figura 5.1: Graphical Representation of the Constant Function

## 5.4 Function of First Degree

The next function to look at is the first degree function. This function (actually this family of functions) is any relationship between $x$ and $y$ that can be represented by $y = m.x + p$ where $m$ and $p$ are two given numbers. $m$ is called the angular coefficient of the function and $p$ its intercept. Since the graphical representation of this type of function is always a straight line, $m$ and $p$ are also called the angular coefficient and intercept of the line $y = mx + p$.

If $m$ is positive the associated line will slope upwards. For example $y = 3x + 5$, the graphical representation of which can be seen in 5.2:

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-4,4,100)
y = 3*x+5
plt.plot(x,y);
plt.show();
```

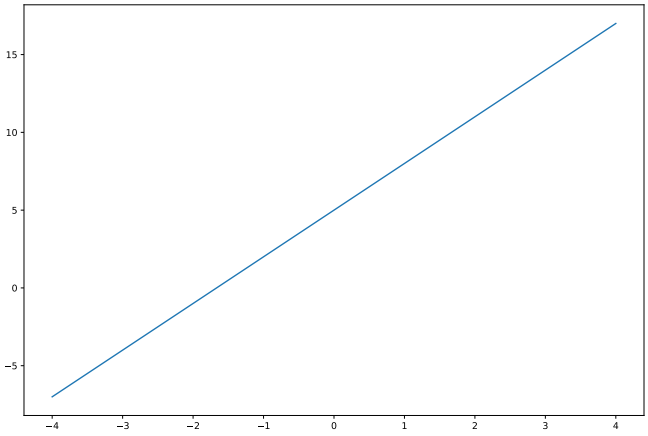



Figura 5.2: Graphical Representation of Function y = 3x+5

If $m$ is negative the line will slope down. For example $y = -3x + 5$ (graphical representation in 5.3)

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-4,4,100)
y = -3*x+5
plt.plot(x,y);
plt.show();
```

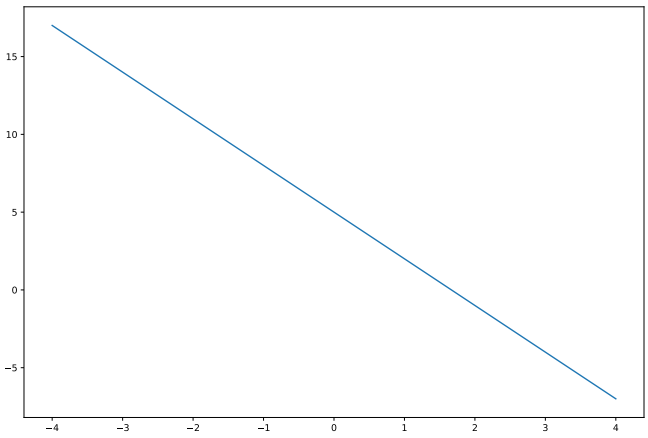



Figura 5.3: Graphical Representation of Function y = -3x+5

Note that if $m = 0$ the first degree function reduces to a constant function, where the value of the constant will be given by the value of the intercept, as for example in $y = 0.x + 4$, whose graph can be seen in 5.4.

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-4,4,100)
y = 0*x+4
plt.plot(x,y);
plt.show();
```

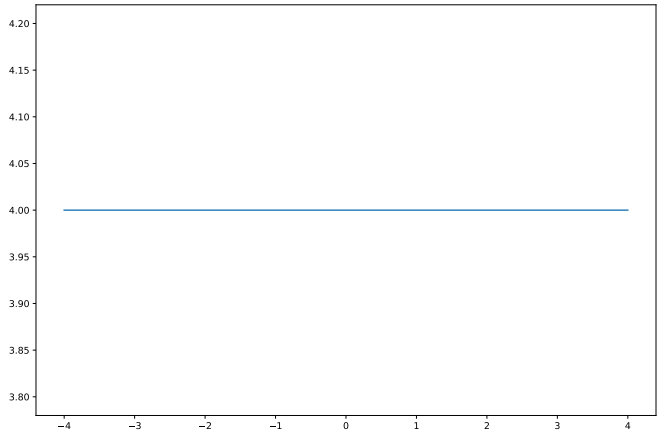



Figura 5.4: Graphical Representation of the Function y = 0.x+4

As a comparison measure below are presented two sets each with three straight lines of different angular coefficients, all with the intercept $p$ equal to 0 (zero). The first set (5.5) with three positive values for $m$ and the second (5.6) with three negative values of $m$. From them you can see that the larger the absolute value of $m$ the steeper the corresponding straight line.

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-3,3,100)
y1 = 0.5*x
y2 = 1*x
y3 = 2*x
plt.plot(x, y1);
plt.plot(x, y2);
plt.plot(x, y3);
plt.show();
```

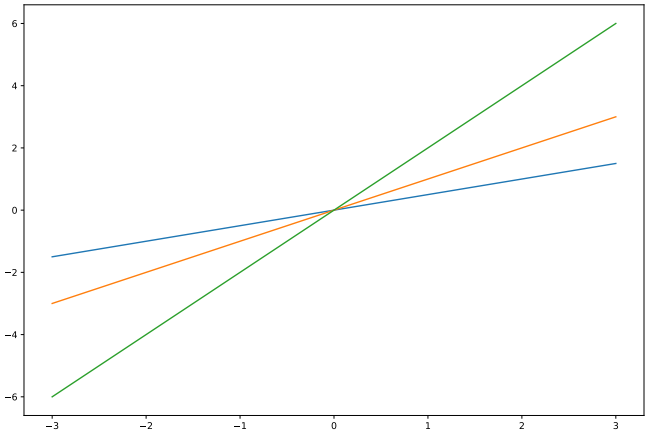



Figura 5.5: Three Straight Lines with different values of m, all positive (0.5, 1, 2)

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-3,3,100)
y1 = -0.5*x
y2 = -1*x
y3 = -2*x
plt.plot(x, y1);
plt.plot(x, y2);
plt.plot(x, y3);
plt.show();
```

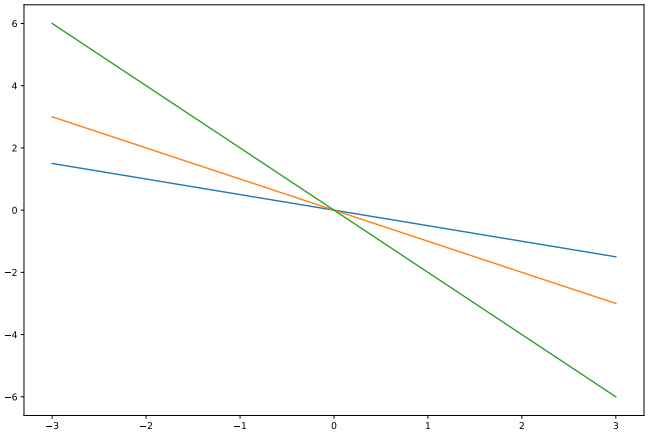



Figura 5.6: Three Straights with different values of m, all negative (-0.5, -1, -2)

Given two points belonging to the same first degree function $(x_0, y_0)$ and $(x_1, y_1)$ one can calculate both the angle coefficient $m$ and the intercept $p$. To calculate the angular coefficient $m$ we do: $m = \frac{y_1 - y_0}{x_1 - x_0}$ and $p = f(0)$.

## 5.5 Application in Economics: One Product Supply Function

In Economics we are interested in determining the relationship between the quantity that producers will be willing to supply from the unit price they can get for their product. From this information, consider the following example:

1. Find the first degree function, which relates the quantity offered by the producers and the price they will be able to obtain, knowing that when the quantity offered was 10 units the price obtained was 5 monetary units and when the quantity offered was 15 units the price obtained was 10 monetary units.

Resolution: We assume that the quantity is $y$ and the price is $x$. Therefore, from the statement we know that $y(5) = 10$ and $y(10) = 15$. Thus, $m = \frac{15-10}{10-5} = 1$. Knowing that $y = 1.x + p$ we calculate $p = 10 - 5 = 5$. Then $y = x + 5$

## 5.6 Application to Human Resources: Wage of a Worker

Salary is the remuneration received by a person for his or her work. This compensation can be fixed (independent of any other factor) or variable when it is dependent on a third factor, for example the number of overtime hours worked during the month. Based on this information solve the following example:

1. A worker has his salary composed of two parts: one fixed and one variable. The variable part is directly proportional to the number of overtime hours worked. Knowing that in the month in which the worker worked 12 overtime hours the remuneration was $R840.00 and in the month in which 20 overtime hours were worked the remuneration was $R1000.00, determine the linear function that relates the number of overtime hours to the worker's total wage, his fixed wage and the amount per overtime hour worked.

Resolution: $m = \frac{f(20)-f(12)}{20-12} = \frac{1000-840}{20-12} = 20$. So $y = m.x + p$, then $p = y - m.x = 840 - 20.12 = 600$. From this we conclude that $y = 20.x + 600$. The fixed salary is therefore \$600.00 and each overtime hour is remunerated with \$20.00.

## 5.7 Second Degree Function

A function of the second degree is any relation between $y$ and $x$ that is of the form: $y = a.x^2 + b.x + c$ where $a \neq 0$. The graph of such a function is a parabola. If $a > 0$ the parabola has upward concavity, as can be seen below for $y = x^2 + x + 1$ in the 5.7

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-3,3,100)
y = x**2 + x + 1
plt.plot(x,y);
plt.show();
```

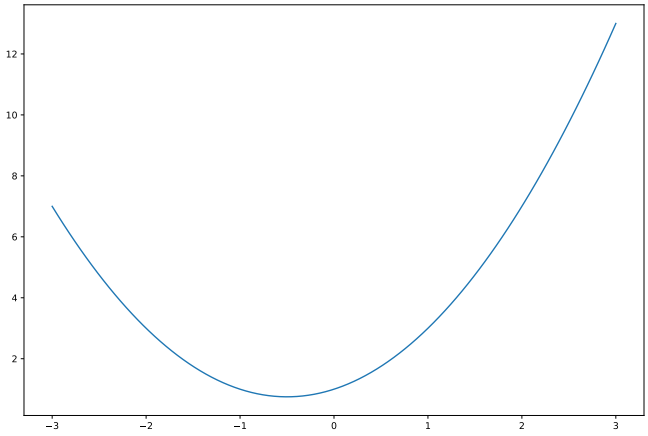



Figura 5.7: Graph of y = x2 + x + 1

If the value of $a$ is less than zero the parabola will have a downward concavity. In 5.8 we can see the graph for $y = -x^2 + x + 1$.

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-3,3,100)
y = -x**2 + x + 1
plt.plot(x,y);
plt.show();
```

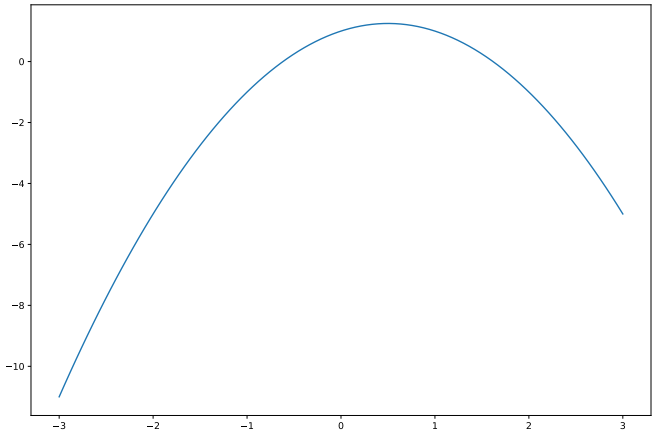



Figura 5.8: Graph of y = x2 + x + 1

The values of $x$ that make $y = 0$ are called second degree function roots. If $y = a.x^2 + b.x + c$ the roots can be calculated by Báskara's formula $x_r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. Depending on the value of $\Delta = b^2 - 4ac$ the roots can be both real and distinct (when $\Delta > 0$), real and equal (when $\Delta = 0$) or conjugate complex (when $\Delta < 0$).

Next we calculate the roots for $y = x^2 - 5x + 6$ using Báskara's formula.

```
import numpy as np
a = 1; b = -5; c = 6;
d = b**2 -4*a*c
x1 = (-b + d)/(2*a)
x2 = (-b - d)/(2*a)
x1, x2
```

```
(3.0, 2.0)
```

Finally, a parabola has an extreme point which can be a maximum of $y$ (if $a < 0$) or a minimum of $y$ (if $a > 0$). Starting from the expression for the parabola $y = ax^2 + bx + c$ this point is given by $(-\frac{b}{2a}, -\frac{\Delta}{4a})$

## 5.8 Application to Marketing: Past and Future Sales of a Product

Consider the following example:

1. An office supply firm has determined that the number of cell phones sold $x$ years from the current year is given, approximately, by the function $f(x) = 40 + 3x + 0.5\,x^2$, where $x = 0$ corresponds to the current year. Based on these data determine:
   (a) What represents $f(0)$
   (b) The number of mobile handsets expected to be sold in six years.
   (c) The values sold from two years ago to the current value, plus the values expected to be sold in two, four, and five years in the future.
   (d) Present these results in graphical form.

The value of the function at $x = 0$ represents the number of handsets sold in the current year. To obtain the number of telephone sets sold six years in the future one must calculate $f(6)$, whose value in this case is equal to: $f(6) = 40 + 3 * 6 + 0,5 * (6)^2$. Such a calculation is done immediately below:

```
def f(x):
  f = 40 + 3*x + 0.5*(x**2)
  return(f)

f(6)
```

```
76.0
```

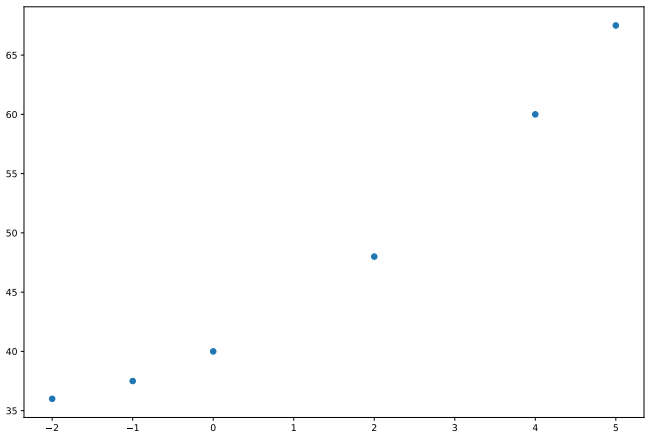
To draw the graph of a function it is necessary to calculate its values at different points on its domain. This requires performing a series of operations in sequence. Python has the facility to perform an operation repeatedly for multiple values of an input variable $x$ through the *numpy* library. The following uses this feature to calculate the value of the function $f(x) = x^3 - 5x^2 + 10x - 7$, at the points $f(-2)$, $f(-1)$, $f(0)$, $f(2)$, $f(4)$, and $f(5)$.

```
import numpy as np
x = np.array([-2, -1, 0, 2, 4, 5])
y = 40 + 3*x + 0.5*(x**2)
f(x)
```

```
array([36. , 37.5, 40. , 48. , 60. , 67.5])
```

Once the pairs of points $x$ and $y$ have been calculated it is possible to generate a graph from them as can be seen below.

```
import numpy as np
from matplotlib import pyplot as plt
x = np.array([-2, -1, 0, 2, 4, 5])
y = 40 + 3*x + 0.5*(x**2)
plt.scatter(x,y);
plt.show();
```




## 5.9 Application to General Management: Break-even Point and Maximum Profit

Second degree functions are useful because they serve to describe the behavior of a function that either increases or decreases from an independent variable. This is a typical behavior of the revenue and cost functions of a product. Due to this behavior, the concept of break-even point arises, which is defined as the quantity that must be sold in order for the company's total profit to be zero. This concept is used to solve the following example:

1. For marketing a certain product a retailer observed that the revenue from the sale of $q$ units of the same is given by the function $R = -3q^2 + 120q$ and the cost associated with its production is given by $C = +2q^2 + 20q + 375$. Based on these infomations ask:

   (a) Sketch the graphs of revenue and cost on the same system of axes, determining and indicating the break-even points.

   (b) Indicate on the graph in the previous item the quantities for which profit is positive

   (c) Obtain the profit function and sketch the graph, indicating the main points.

   (d) What quantity of watches must be sold for the profit to be maximum? What is the maximum profit?

   (e) For which traded quantity is the profit positive? Compare with the results obtained for the break-even point.

## 5.10 Application to Finance: Prices of a Stock Exchange Traded Stock

Often we do not have a precise algebraic expression to represent the pairs of $x$ and $y$ values of a function (such as in the previous example, where $y = 40+3x+0.5x^2$). In this case we can make use of a table with the pairs of points $x$ and $y$. An example can be seen below:

1. Suppose that the price of a stock over time can be represented by the function: $y = f(t)$ where $t$ are the months of the current year, from January ($t = 0$) to December ($t = 11$). The values of the function $y$ in this case are represented in the following table. Based on these values ask:
   (a) Present the graphical representation of the function.
   (b) Assuming the current month is $t = 2$, what is the value of the stock three months in the future?
   (c) What is the month in which the stock had its maximum value?
   (d) What is the month when the stock had its minimum value?
   (e) Which month had the smallest percentage increase in value? (in relation to the previous month)
   (f) Which month had the highest percentage increase in value? (in relation to the previous month)
   (g) What is the average value of the stock over the year?
   (h) Suppose an investor bought 100 shares of the company for a total of $370.00. In approximately which month do you think he executed this purchase transaction?

```
x = [0,1,2,3,4,5,6,7,8,9,10,11]
y = [5.0, 6.5, 7.8, 3.5, 4.8, 5.3]
y = y+[8.1, 9.5, 10.1, 7.5, 6.5, 9.9]
```

Tabela 5.1: Representation of a Function in Table Form

| x | y |
|---|---|
| 0 | 5.0 |
| 1 | 6.5 |
| 2 | 7.8 |
| 3 | 3.5 |
| 4 | 4.8 |
| 5 | 5.3 |
| 6 | 8.1 |
| 7 | 9.5 |

| x | y |
|---|---|
| 8 | 10.1 |
| 9 | 7.5 |
| 10 | 6.5 |
| 11 | 9.9 |

Resolution:

The graphical representation of the function can be seen in the figure 5.9:

```
import numpy as np
from matplotlib import pyplot as plt
t = [0,1,2,3,4,5,6,7,8,9,10,11]
p = [5.0, 6.5, 7.8, 3.5, 4.8, 5.3]
p = p + [8.1, 9.5, 10.1, 7.5, 6.5, 9.9]
plt.scatter(t,p);
plt.plot(t,p);
plt.show()
```

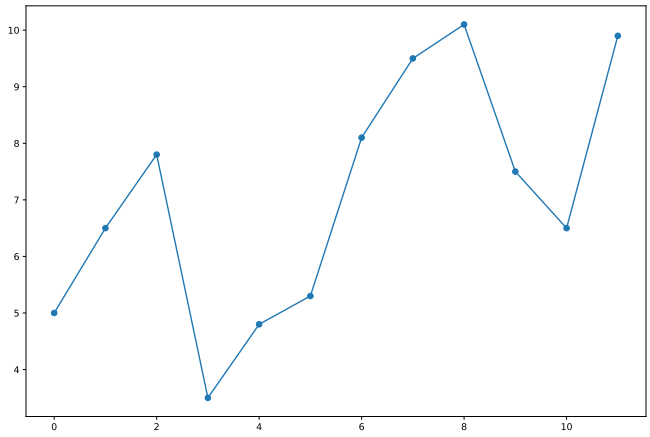



Figura 5.9: Graphical Representation, Price of a Stock Exchange-Traded Share Over Time

If the current month is $t = 2$, the month $t = 2 + 3 = 5$ will be three months in the future. For $t = 5$ we see from the table that $p = 5.3$. We can get this value directly from the $p$ array by `p[5]` as follows:

```
import numpy as np
```

```
from matplotlib import pyplot as plt
p = [5.0, 6.5, 7.8, 3.5, 4.8, 5.3]
p = p + [8.1, 9.5, 10.1, 7.5, 6.5, 9.9]
p[5]
```

```
5.3
```

To get the months in which the action had its maximum and minimum values used the `.argmax` and `argmin` methods (from `numpy` arrays), which provide the months $t = 8$ (which would be September) and $t = 3$ (which would be April):

```
import numpy as np
from matplotlib import pyplot as plt
p = [5.0, 6.5, 7.8, 3.5, 4.8, 5.3]
p = p + [8.1, 9.5, 10.1, 7.5, 6.5, 9.9]
p = np.array(p)
p.argmax(), p.argmin()
```

```
(8, 3)
```

To get the months with the largest and smallest percentage valuations we first compute the percentage change in the stock price (starting in the month $t = 1$) in another `numpy` array which we will call $d$. Next we get the `.argmax` and `argmin` values from `d`.

```
import numpy as np
from matplotlib import pyplot as plt
p = [5.0, 6.5, 7.8, 3.5, 4.8, 5.3]
p = p + [8.1, 9.5, 10.1, 7.5, 6.5, 9.9]
p = np.array(p)

p1 = p[1:] # From the 2nd to the last month
p0 = p[:-1] # From the 1st to the penultimate month
d = p1/p0-1
d.argmax()+1, d.argmin()+1
```

```
(6, 3)
```

The average of the stock value can be obtained with `np.mean`:

```
import numpy as np
from matplotlib import pyplot as plt
p = [5.0, 6.5, 7.8, 3.5, 4.8, 5.3]
p = p + [8.1, 9.5, 10.1, 7.5, 6.5, 9.9]
p = np.array(p)
np.mean(p)
```

```
7.041666666666667
```

Finally, an investor who bought 100 shares for a total of R$370.00, bought each share at a unit price of R$3.70. From the table one can see that the most likely month for such a purchase is $t = 3$, which would be April.

## 5.11 Physics Application: Energy Levels in an Atom

In the following example it is recommended to use scientific notation. In Python scientific notation is indicated by the lowercase $e$. So `4.445e8` in python is equal to $4.445.10^8$. The energy levels $n$ in a hydrogen atom are given by the expression $E_n = -\frac{m_e e^4}{8e_0^2 h^2} . \frac{1}{n^2}$ where $m_e = 9.1094.10^{-31}$ kg is the mass of the electron, $e = 1.6022.10^{-19}$ C is the charge of the electron, $\epsilon_0 = 8.8542.10^{-12}$ $C^2$ $s^2$ $kg^{-1}$ $m^{-3}$ is the electric permittivity of vacuum, and $h = 6.6261.10^{-34} Js$. The energy released when an electron moves from a level $n_i$ to a level $n_f$ is given by $\Delta E = -\frac{m_e e^4}{8e_0^2 h^2} . \left(\frac{1}{n_i^2} - \frac{1}{n_f^2}\right)$. You are asked to calculate the first 20 energy levels $E_n$ and the energy released when an electron moves from 1 to 5 levels above the current one.

## 5.12 Application in Ecology: Water Level in a Reservoir

The final volume of a water reservoir is equal to its initial volume plus the amount of rainfall (in cubic meters). Knowing that the initial volume of a reservoir is `4,445e8` cubic meters and a summer storm brought a total of `5e6` cubic meters into it, you are asked to determine the final gross volume of water in the reservoir. Once this value has been calculated, the adjustments described below must be made to determine the final net volume of water in the reservoir:

1. Decrease the value of the variable ‘wetness‘ by 10
2. Increase the volume of the reservoir by 5
3. Decrease the volume in 5
4. Decrease ‘2.5 and 5‘ cubic meters due to water that will be channeled to arid regions.
5. Final answer: 447,627,500.0 cubic meters

## 5.13 Exercise

1. A brokerage firm charges a commission of 1.2 percent on purchases and sales of stocks in the range from 0 to 5000 reals. For purchases exceeding 5000 the firm charges 1.2 percent of the total purchase plus 10 reals. Let $x$ be the amount of stock traded (in reals) and for $f(x)$ be the commission charged as a function of $x$. Based on these data ask:

   (a) Create a table where the value of the commission charged is calculated from the value traded in shares.

   (b) Calculate $f(2000)$ and $f(10000)$

2. Suppose the brokerage firm in the exercise 5.13 decides to keep commission rates fixed at 1.4 percent, for transactions up to 6000 and for purchases of more than 6000 charge 1.5 percent plus 15 per transaction. Based on these data ask:

   (a) Set up a spreadsheet where the value of the commission charged is calculated from the value of the gold purchased

   (b) Calculate $f(100)$ and $f(500)$

3. A market survey indicated that the relationship between the price of sugar (in dollars per kilogram) and the quantity demanded (in thousands of tons) in a given country are related by the function: $p(x) = -0.25 * x + 50$. Knowing that the cost of production $C(x)$ is given by the expression $C(x) = 2.25\,x^2 + 3\,x + 70$ ask:

   (a) Assemble in spreadsheet the expression of the expected profit as a function of the price charged, that is $L(p)$ and of the expected profit as a function of the quantity demanded, that is $L(x)$.

   (b) Determine the price $p$ and the quantity $x$ that make the production of sugar profitable.

4. Suppose that the total cost of production of a certain product is given by the function $C(x) = x^3 - 20\,x^2 + 400\,x + 300$, where $x$ is the number of units of the product manufactured. Based on these data ask:

   (a) Determine the cost of producing 10 units of the product

   (b) Determine the cost of producing only the $10^a$ unit of the product

5. It is known that the average amount of pollutants in the air of a given city is given (in parts per million) by the expression $c(a) = 0.5\,a + 2$ where $a$ is the amount of cars in circulation (in thousands of cars). Assuming that the quantity of cars in circulation $t$ years from the current date obeys the relation $a(t) = 15 + 0.15\,t^2$ ask:

   (a) Determine the amount $c$ of pollutants in the air, starting at time $t$, i.e. $c(t)$.

(b) Determine when the amount of pollutants in the air will reach 8.5 parts per million.

6. Suppose that the number of phone calls needed (in hundreds of calls) to reach $x$ percent of the people in a given city, is given by the function $Nt(x) = \frac{500,x}{400-x}$. Based on this statement ask:

   (a) Determine the values of $x$ for which the function "makes sense", that is the Domain of the function $Nt(x)$.

   (b) How many phone calls will it take to reach 50 percent of the population?

   (c) How many phone calls are needed to reach 100 percent of the population?

   (d) What percentage of the city's population will have been reached when $14,000$ of phone calls have been made?

7. The **average manufacturing cost** is defined as the cost of producing $x$ units of a product (measured in monetary units) divided by the actual quantity manufactured $x$. Supposing a product has a manufacturing cost given by the expression $C(x) = 180 + 3\,x$ ask to determine:

   (a) The average cost of manufacturing 35 units

   (b) The average manufacturing cost of 85 units

   (c) To what value will the average cost tend as the quantity manufactured $x$ increases?

8. Income tax is a percentage amount levied from an individual's average monthly income.

   (a) Assume the following composition :

      i. Individuals with income up to $R\$1,499.15$ are exempt

      ii. Individuals with income from $1,499.16$ to $2,246.75$ pay 7.5 percent and deduct from this total 112.43.

      iii. Individuals with income from $2,246.76$ up to $2,995.70$ pay 15.00 percent and deduct from this total 280.94.

      iv. Individuals with income from $2,995.71$ to $3,743.19$ pay 22.5 percent and deduct from this total 505.62.

      v. Individuals with income above $R\$3,743.19$ pay 27.5 percent and deduct from this total $R\$692.78$.

   (b) Based on these data ask :

      i. Calculate the amount of income tax to be paid for a person with an average monthly income of $R\$3,700.00$ and the net amount to be received

ii. Calculate the amount of income tax to be paid for a person with an average monthly income of $R\$3,800.00$ and the net amount to be received

iii. Based on the above calculation, is the statement that by changing salary band, a salary increase actually turned into a lower net amount to be received valid?

iv. Put together in a spreadsheet a function to calculate the amount of income tax to be paid and the net amount to be received by a person based on the average monthly amount of his taxable income.

9. In a city the water consumption tariff follows the rules shown in table 5.2. Based on the data presented ask:

   (a) The total cost and the average cost for those who consume 9, 18, 27 and 65 $m^3$

   (b) Assemble in spreadsheet a formula for calculating the total cost and average cost in water consumption.

Tabela 5.2: Water Consumption Table

| M3 consumption | Tariff |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |

10. A manufacturing process follows the function $f(x) = -x^3 + 6\,x^2 + 15x$, where $x$ is the number of process hours and $f(x)$ is the quantity of product manufactured, measured in tens of kilograms. Based on these data, ask:

    (a) The quantity of product manufactured two hours after the start of the process.

    (b) The amount of product manufactured between the $2^a$ and $3^a$ hour.

11. After a company uses a piece of land, it is necessary to clean it up by eliminating a certain percentage of the total pollutants released into the environment. Suppose that the cleanup cost is given by the function $C(x) = \frac{250,x}{300-x}$ where $x$ is the percentage to be eliminated and $C(x)$ the cleanup cost in hundreds of thousands of reals. Based on these data ask:

    (a) What is the validity region of the function $C(x)$, that is, what is the domain of $C(x)$

    (b) What is the cost of cleaning up 40 percent of the pollutants?

(c) What is the cost of cleaning up the remaining 40 percent?

(d) What percentage could be cleaned up with $R\$1M$ of reals?

12. Suppose the number of inhabitants in a certain region is given by the function $Np(a) = 39\,a^(0.33)$, where $a$ is the area in square kilometers, ask:

    (a) How many inhabitants can be expected in a region of $10\,km^2$?

    (b) By tripling the area, by what value is the population multiplied?

    (c) To find $10,000$ people, how large a region size is needed?

13. Suppose the demand function for a commodity is given by the expression $Q(p) = \frac{5.437}{p^2}$ where $p$ is the price per kilogram and $Q(p)$ the quantity demanded per week. Assuming that the price per kilogram varies over time according to the function $P(t) = 0.03\,t^2 + 0.25\,t + 10.9$ , with time $t$ measured in weeks, determine:

    (a) The weekly demand as a function of time

    (b) The quantity that will be demanded in 8 weeks

    (c) If and when the demand will reach 27,356 kilograms

14. A cable is to be extended over and parallel to a river to carry electricity from a distribution point to a factory. Suppose the river is 800 meters wide and the factory is $4,500$ meters from the distribution point. The cost of extending the cable over water is 5.5 per meter and over land is 3.8 per meter. Calling $x$ the distance along the river from the distribution point to the point where the cable will reach the other bank, express the total cable extension cost as a function of $x$.

15. A saleswoman's weekly payment depends on her sales volume. If she sells x units of goods, then she receives y = 5x + 60 reals. Provide an interpretation for the angular coefficient and the intersection with the y-axis of this line.

16. In an economy the consumption function $C(y)$, is given by $C(y) = 100{+}0.8y$. Based on this data calculate the income required for an investment (savings) of 20 monetary units to occur.

17. A gas company will pay a landowner $5,000$ for the right to drill on the land to find natural gas, and 0.10 for each thousand cubic meters of gas extracted. Express, as a function of the amount of gas extracted, the total the landowner will receive.

18. The cost to remove percent of a pollutant is given by the function : $f(x) = 50x/(105 - x)$, for $0 <= x <= 100$. What is the domain of the function? Calculate the cost to remove 0.70 of the total pollutant

19. The fixed costs of a pie manufacturing company are the payment of 5 employees at R$600, water, electricity and telephone bills totaling R$500

and rent totaling R$1,500; the cost of raw materials and packaging in pie production is R$5 per pie. Based on this information ask:

(a) Determine the total cost function C(q).

(b) What is the selling price to ensure a profit of 20 percent on 1,000 units sold?

(c) Assume the selling price is 15. Determine the total revenue function R(q) and the break-even point of the process.

(d) Assuming income tax of 20 percent on total profit, what function L(q) represents net profit?

(e) Knowing that the company wants to earn a monthly net profit of Russell 10,000 and that in the last month 5,000 pies were sold, has the profit target been reached?

20. A company produces scraper blades at a unit cost of 2.50 each, in addition to incurring fixed costs to pay the water bill (50), electricity (150), rent for the shed (250), accountant (100), and per diem for five employees (90 each) each month.

    (a) What sales price balances the costs, considering a minimum of 100 spatulas sold per month?

    (b) What is the monthly profit when there is a demand for 200 spatulas per month?

    (c) An employee had an accident and was fired. The TRT-SP (Labor Court of Appeals of São Paulo) won the case and condemned the company to pay $2,500$. How many spatulas need to be sold so that the monthly profit can cover this unexpected expense?

21. A company has implemented a loyalty policy for orders from large buyers. Setting a maximum of 200 products per order, the buyer can choose between a reduction of 0.50 on the price per piece for orders over 100 pieces, or a unit purchase price that varies according to the number of pieces sold, reducing 0.005 per unit sold (i.e., two units come out to 2.49 each and ten units come out to 2.45 each). What is the best strategy, from the buyer's point of view, for orders of 50 pieces, 100 pieces, 125 pieces, 150 pieces, and 200 pieces?

22. An online course company varies its prices per lesson hour according to the demand for lessons. The current strategy, based on demand from the previous week, sets the price at 100/hour when demand for the week is up to 8 hours/week and there is a decrease of 10 when demand increases by 4 hours/class (i.e., for example, 90/hour for 12hours/week).

    (a) What is the maximum revenue that can be obtained, considering a limit of 20 hours/week of work and zero cost of capturing the orders?

(b) Knowing that the main monthly costs of the business are cell phone (250) and landline (50) bills, Web site hosting (50), printing sheets and cards (100), bank fees (50), and that there is additional monthly revenue of 200 in advertising, determine the cost, revenue, and total weekly profit functions, find the break-even point, and the approximate number of hours/week that would result in 1000 of weekly profit.

23. The average cost of producing juices with one fruit is R$1.25, with two fruits is R$1.50, and with three fruits is R$2.50. The snacks have a unit cost of R$1,50$. Besides the costs with raw material, already informed, there are costs with 8 employees (R$1,500$ each), rent in the central hall (R$9,000$) and advertising in newspapers, magazines and specialized sites (R$9,000$). Previous studies indicate that the company sells 20% of the juices with one fruit, and double that for juices with three fruits. Still, for each juice sold there is an average consumption of one snack. Considering taxes of 6.5

    (a) The total cost function C(q).

    (b) The total revenue function, R(q), assuming that the products are sold at twice the cost price.

    (c) The total profit function L(q).

    (d) The break-even point of sales, that is, the minimum number of juices that Armando needs to sell in order not to make a loss.

    (e) How many juices must be sold for the company to have a minimum monthly profit of $12,000$, after taxes and card fees.

24. Create a Python function that calculates the amount of the water bill from the total volume (in cubic meters) consumed in the month. Look up the formula for calculating the water bill on the utility's Web site or on a water bill in your area.

25. Create a Python function that calculates the electricity bill from the total electricity consumption. Look up the calculation methodology on your local electric utility's Web site.

26. A company pays its salespeople a fixed salary plus a commission on the total sales they make. Up to R$10,000.00$ the commission is 3%. The portion above this value has a 5% commission. Create a Python function that takes the fixed salary of the salesperson and the total sales made and returns the total salary.

27. Create a Python function that receives the total monthly salary of an employee and returns the net amount to be received discounting: 1.8% of INSS limited to a maximum of R550.00, 8% of Income Tax for salaries up to R$2,500.00$, 17% for salaries up to R$5,500.00$ with a reduction of R225.00 in the amount to be paid in tax, and 25% for salaries above R$5,500.00$ with a reduction of R440.00.

# Capítulo 6

# Analytic Geometry

We will now extend the concepts presented in the chapter on Functions, delving into ways of plotting graphs and reviewing concepts of analytic geometry. Plotting graphs will range from the basic ideas of Analytic Geometry (such as points and coordinates) to graphs in three dimensions.

## 6.1 Points and Coordinates

The basic idea of Analytic Geometry is centered on the ability to express the positioning of points, lines, plane and spatial figures in general in terms of a coordinate system. This coordinate system in the plane is formed by two lines perpendicular to each other, their intersection being called the origin. A point is then defined by its distance from these lines. The other geometric figures are defined by the distances from each of their points to these lines, which are called *coordinate axes*. The horizontal axis is called the $X$ axis and the vertical axis the $Y$ axis.

1. Draw the points in the plane with the following set of coordinates: (-2; 3), (4; 2) and (-4;-1).

As can be seen in table 6.1 and figure 6.1, the points can be drawn directly using `numpy` arrays and the `matplotlib` library.

```
import numpy as np
x = [-2, 4, -4]; y = [3, 2, 1]

import pandas as pd
df = pd.DataFrame({"x":x, "y":y});

from matplotlib import pyplot as plt
graf = plt.scatter(x, y);
```

```
plt.show();
```

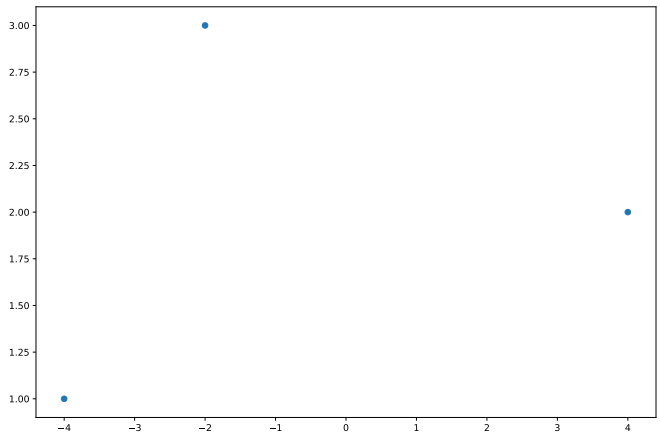



Figura 6.1: Points in the plane with coordinates (-2; 3), (4; 2) and (-4;-1)

Tabela 6.1: Table for Drawing Points in the Plane

| x | y |
|---|---|
| -2 | 3 |
| 4 | 2 |
| -4 | 1 |

## 6.2 Distance Between Points

The distance between two points of coordinates $(x_1, y_1)$ and $(x_2, y_2)$ in the plane is calculated by the formula: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

1. Calculate the distance between the points $A = (1, 2)$ and $B = (4, 6)$

```
import numpy as np
A = np.array([[1],[2]])
B = np.array([[4],[6]])
d = np.sqrt((A[0,0]-B[0,0])**2 + (A[1,0]-B[1,0])**2)
d
```

```
5.0
```

## 6.3 Regions in the xy-Plane

Regions on the XY Plane is the set of points that satisfy a given condition. The following are examples of regions on the XY plane.

1. Draw on the XY plane the regions defined by:
   (a) $(x, y)|x >= 0$
   (b) $(x, y)|y = 1$
   (c) $(x, y)|||y| < 1$

## 6.4 Triangles

Creating geometric figures formed by line segments (polygons) can also be done in Python by using the `.plot` method instead of the `.scatter` method.

2. Draw the triangle formed by the points (0;0), (4;0) and (4;3).

As can be easily seen from the cells below, connecting points with straight lines produces the desired triangle (see figure 6.2). It should be noted that the point (0;0) was supplied twice (at the beginning and end of the `numpy` arrays), so that the triangle would be "closed."

You can create the sequence of `x` and `y` values separately as shown in the code that generated the 6.2 figure:

```
import numpy as np
x = [0,4,4,0]; y = [0,0,3,0]

from matplotlib import pyplot as plt
graf = plt.plot(x,y);
plt.show();
```

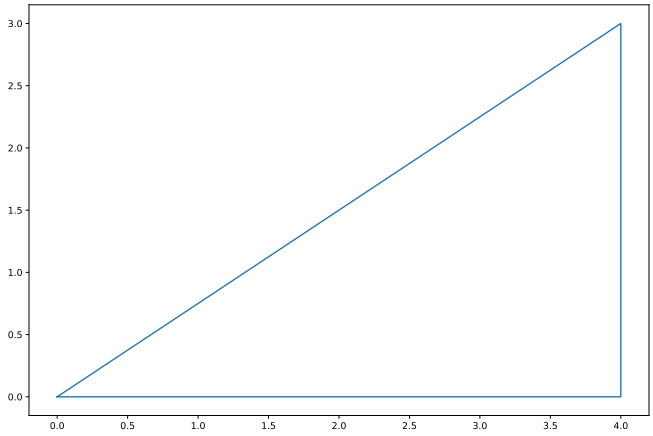



Figura 6.2: Triangle formed by the points (0;0),(4;0) and (4;3), from 1D numpy arrays

Or create the points separately and "stack" them to form a 2D array. The corresponding `x` and `y` columns are then generated from the first and second columns of this 2D array, as shown in fig.6.3.

```
import numpy as np
from matplotlib import pyplot as plt
x1 = np.array([0,0]); x2 = np.array([4,0]); x3 = np.array([4,3])
ptos = np.array([x1,x2,x3,x1]); x = ptos[:,0]; y = ptos[:,1]
graf = plt.plot(x,y);
plt.show(block=graf);
```

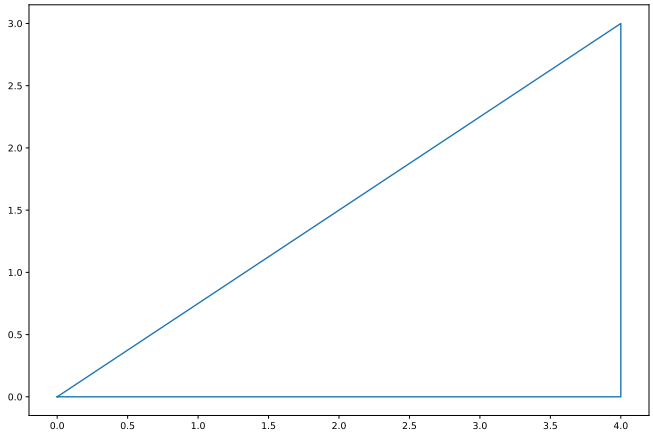



Figura 6.3: Triangle formed by the points (0;0),(4;0) and (4;3), from 2D numpy arrays

The third way to generate the graph is to create a pandas data frame where each column is composed of an `numpy` array and display the values from the columns of this data frame. This way can be seen in figure 6.4, whose design was created from the data in table 6.2

```
import numpy as np
x = [0,4,4,0]; y = [0,0,3,0]

import pandas as pd
df = pd.DataFrame({"x":x, "y":y})

from matplotlib import pyplot as plt
graf = plt.plot(df['x'],df['y']);
plt.show();
```

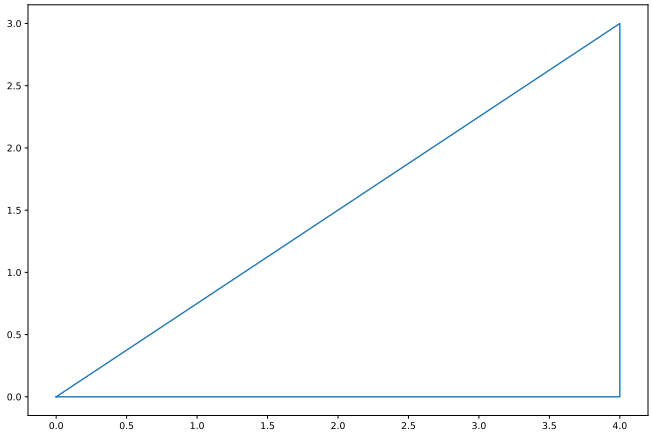



Figura 6.4: Triangle formed by the points (0;0),(4;0) and (4;3), from a pandas dataframe

Tabela 6.2: Table for drawing the triangle with the points (0;0),(4;0) and (4;3)

| x | y |
|---|---|
| 0 | 0 |
| 4 | 0 |
| 4 | 3 |
| 0 | 0 |

3. Starting from the triangle formed by the points (0;0),(4;0) and (4;3), draw the medians of the triangle, showing that they meet at a single point (also called the textquotedblleftbarycentertextquotedblright).

The midpoint of a segment is determined by averaging the coordinates of the points located at its extremities. Applying this rule to each of the sides of the triangle we will obtain three midpoints. By connecting these points to the opposite ends we obtain the medians (figure 6.5).

```
import numpy as np
import sympy as sp
x, y = sp.symbols("x y")

from matplotlib import pyplot as plt
import matplotlib as mpl
```

```
A = np.array([0,0]); B = np.array([4,0]); C = np.array([4,3])
BC = (B + C)/2; AC = (A + C)/2; AB = (A + B)/2

line = lambda x,y : plt.plot([x[0],y[0]],[x[1],y[1]])
g = line(A,B); g = line(B,C); g = line(C,A)
g = line(A,BC); g = line(B,AC); g = line(C,AB)
plt.show()
```

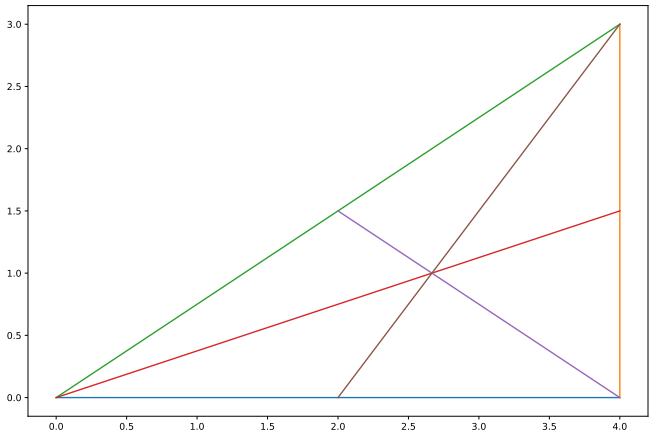



Figura 6.5: Meeting Point of the Medians of a Triangle

## 6.5 Circuferences

Circles are curves whose points are at a constant distance from a point called the center. Curves in general can be expressed in rectangular form, (which in the case of a circle would be $R^2 = x^2 + y^2$), or in polar form in which the coordinates $x$ and $y$ are replaced by a distance to the origin, usually represented by the Greek letter $\rho$ (rho) and the angle formed by the preceding line and the horizontal axis, usually represented by the Greek letter theta $\theta$. The relationship between the rectangular coordinates $x$ and $y$ and the polar coordinates $\rho$ and $\theta$ is given by:

$$x = \rho\cos(\theta) \tag{6.1}$$
$$y = \rho\sin(\theta) \tag{6.2}$$

Polar coordinates are useful in creating graphs in which the function takes on

two distinct values for the same value of $x$, which would be difficult to represent in the format $y = f(x)$. The simplest example in this case is the drawing of a circle.

4. Draw the circle with center (2,3) and radius $r = 4$.

First we must assemble the components $x$ and $y$ as a function of the polar coordinates $\rho$ and $\theta$. For this case we will have:

$$x = 2 + 4\cos(\theta) \tag{6.3}$$
$$y = 3 + 4\sin(\theta) \tag{6.4}$$

Where $\theta$ will range from 0 to 360 degrees.

The figure formed by the scatter plot described in 6.5 can be seen in figure 6.6:

```
import numpy as np
from matplotlib import pyplot as plt
import matplotlib as mpl
mpl.use("TkAgg")

r = 4
theta = np.linspace(0,360,361)
x = r*np.cos(theta)
y = r*np.sin(theta)
g = plt.scatter(x,y, s=1);
plt.show();
```

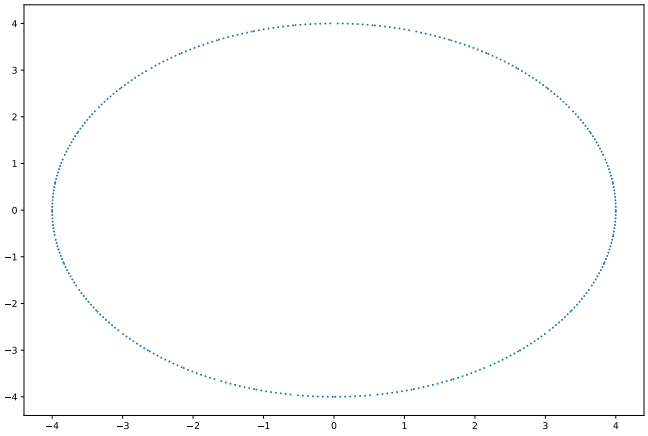


Figura 6.6: Circle with center at (2,3) and radius equal to 5

5. From the points (3;0) and (0;4) draw :
   (a) An equilateral triangle, with side length equal to the distance between points (3;0) and (0;4).
   (b) The heights of each side, showing that they meet at a single point called the orthocenter.
   (c) The circle inscribed in the triangle and passing through the intersections of the heights with the sides of the triangle.

The distance $d$ between the points (3;0) and (0;4) is given by $d = \sqrt{(3-0)^2 + (0-4)^2} = 5$. We therefore want to construct an equilateral triangle of side equal to five, with vertices at the points (3;0) and (0;4).

We now need to determine the $(x, y)$ coordinates of the third vertex of the triangle. Knowing that the triangle is equilateral, this point will be at a distance equal to 5 from both the point (3;0) and the point (0;4). So we can solve the following system of equations:

$$(x-0)^2 + (y-4)^2 = 5^2 \tag{6.5}$$
$$(x-3)^2 + (y-0)^2 = 5^2 \tag{6.6}$$

```
import sympy as sp
x, y = sp.symbols("x y")
eq1 = (x-0)**2 + (y-4)**2 - 5**2
```

```
eq2 = (x-3)**2 + (y-0)**2 - 5**2
sols = sp.solve([eq1, eq2],[x,y])
print(sols[0])
```

```
(3/2 - 2*sqrt(3), 2 - 3*sqrt(3)/2)
```

```
print(sols[1])
```

```
(3/2 + 2*sqrt(3), 2 + 3*sqrt(3)/2)
```

We have two solutions, one for each side that we draw the triangle on. Adopting the solution in the first quadrant, we have $(x;y) = (\frac{3}{2} + 2\sqrt{3}; 2 + \frac{3\sqrt{3}}{2})$. Since the triangle is equilateral, the heights and medians coincide. We can then use a similar scheme to that adopted in drawing the figure 6.5 to draw the medians. The resulting triangle and its medians can be seen in figure 6.7.

```
import numpy as np
A = np.array([0,4]); B = np.array([3,0]); C = np.array(sols[1])
BC = (B + C)/2; AC = (A + C)/2; AB = (A + B)/2

line = lambda x,y : plt.plot([x[0],y[0]],[x[1],y[1]])
g = line(A,B); g = line(B,C); g = line(C,A)
g = line(A,BC); g = line(B,AC); g = line(C,AB)
plt.show();
```

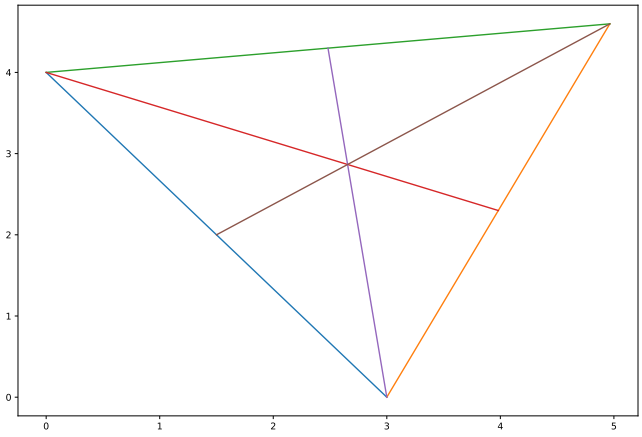



Figura 6.7: Equilateral triangle with vertices at (0;4) and (3;0)

To draw the inscribed circle we need the points AB, BC, CA. With these three points, we can find the equation of the inscribed circle and then draw its graph. Notice below that we first determine the center of the circle and its radius by solving a system of equations. In possession of the center and radius of the circle we will draw its graph in two steps. First we calculate the coordinates of a circle with center at the origin. We will do this using polar coordinates and converting the results to rectangular coordinates. With the $x$ and $y$ coordinates of a circle centered at the origin, we will add the coordinate values of the center of the circle to both $x$ and $y$ and thus obtain the sequence of points to be drawn.

```
import numpy as np
A = np.array([0,4]); B = np.array([3,0]); C = np.array(sols[1])
BC = (B + C)/2; AC = (A + C)/2; AB = (A + B)/2

line = lambda x,y : plt.plot([x[0],y[0]],[x[1],y[1]])
g = line(A,B); g = line(B,C); g = line(C,A)
g = line(A,BC); g = line(B,AC); g = line(C,AB)

import sympy as sp
x, y, R = sp.symbols("x y R", real=True)

Eq1 = (x-AB[0])**2 + (y-AB[1])**2 - R**2
Eq2 = (x-BC[0])**2 + (y-BC[1])**2 - R**2
Eq3 = (x-AC[0])**2 + (y-AC[1])**2 - R**2
xc,yc,Rc = sp.solve([Eq1, Eq2, Eq3],[x,y,R])[1]
theta = np.linspace(0,2*np.pi,100)
x1 = Rc*np.cos(theta) + xc
y1 = Rc*np.sin(theta) + yc
g = plt.plot(x1, y1);
plt.show();
```

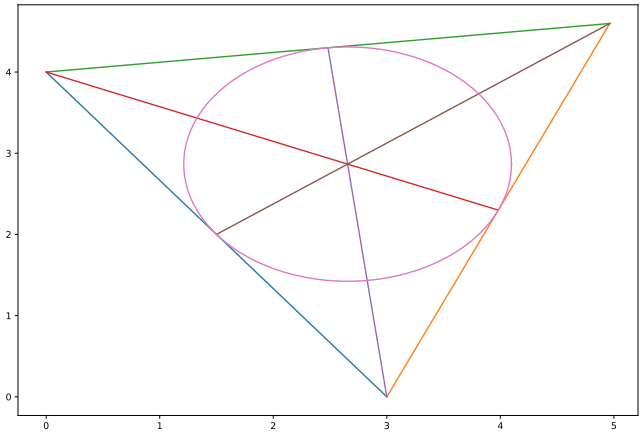



Figura 6.8: Circle Inscribed in the Equilateral Triangle

## 6.6 Trigonometry

In addition to studying the properties of triangles there are functions whose independent variable is an angle value. Angles are measured in this case on a circle of radius = 1 called the trigonometric circle. The units of measure of angles can be degrees (in this case one complete rotation on the trigonometric circle equals 360°) or radians (when one complete rotation equals $2\pi$ radians).

Thus one radian is approximately equal to $\frac{180}{\pi} \approx 57.3^o$. Note that to convert from degrees to radians we use the relation: $degrees = \frac{180}{pi}.radians$ and to convert from radians to degrees we do: $radians = \frac{pi}{180}.degrees$

Given a right triangle $ABC$ (see 6.9), with the angle $\theta$ formed by the meeting of the sides $AB$ and $AC$, where $AB$ is called the adjacent catet to the angle $\theta$, $BC$ is the opposite catet to the angle $\theta$ and $AC$ is the hypotenuse, the most important trigonometric functions are defined as follows:

```
import numpy as np
from matplotlib import pyplot as plt
plt.plot([0,4],[0,0], color='blue'); #AB
```

```
[<matplotlib.lines.Line2D object at 0x7f869486daf0>]
```

```
plt.plot([4,4],[0,3], color='green'); #BC
```

```
[<matplotlib.lines.Line2D object at 0x7f869486de50>]
```

```
plt.plot([0,4],[0,3], color='red'); #AC
```

```
[<matplotlib.lines.Line2D object at 0x7f86947f96a0>]
```

```
plt.show();
```

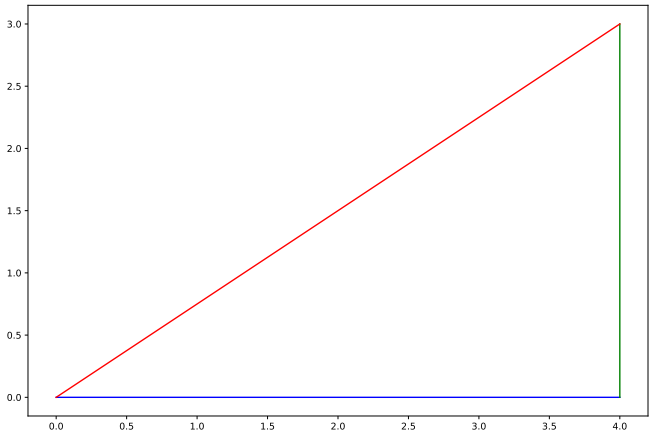



Figura 6.9: Right triangle ABC, with carbets AB (blue) and BC (green) and hypotenuse AC (red)

1. $cos(\theta) = \frac{blue}{red}$
2. $sin(\theta) = \frac{green}{red}$
3. $tg(\theta) = \frac{green}{blue}$

$

With these functions we can define their inverses:

1. $sec(\theta) = \frac{1}{cos(\theta)}$
2. $cossec(\theta) = frac1sin(\theta)$
3. $cotg(\theta) = frac1tg(\theta)$

And the basic relationships between trigonometric functions:

1. $sin(\theta)^2 + cos(\theta)^2 = 1$
2. $1 + tg(\theta)^2 = sec(\theta)^2$

3. $1 + cotg(\theta)^2 = cossec(\theta)^2$

Next we have the relations for the sine and cosine of the sum of two angles $\alpha$ and $\beta$:

1. $sin(\alpha + \beta) = sin(\alpha).cos(\beta) + sin(\beta).cos(\alpha)$
2. $cos(\alpha + \beta) = cos(\alpha).cos(\beta) - sin(\beta).sin(\alpha)$

Finally we can add the basic relationship between sine and cosine and the sum of cosines to get two very useful relationships (see below):

1. $cos^2(\alpha) + sin^2(\alpha) = 1$
2. $cos^2(\alpha) - sin^2(\alpha) = cos(2\alpha)$
3. The sum of the above relations gives us: $cos^2(\alpha) = \frac{1+cos(2\alpha)}{2}$
4. The difference of the above ratios gives us: $sin^2(\alpha) = \frac{1-cos(2\alpha)}{2}$

## 6.7 Trigonometric Functions

The functions $sin(x)$, $cos(x)$ and $tg(x)$ have graphs as follows: (sine at 6.10, cosine at 6.11 and tangent at 6.12). All of them have a period of $\$2^3$ radians.

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-np.pi,+np.pi,1000)
y = np.sin(x)
plt.plot(x,y);
plt.show();
```

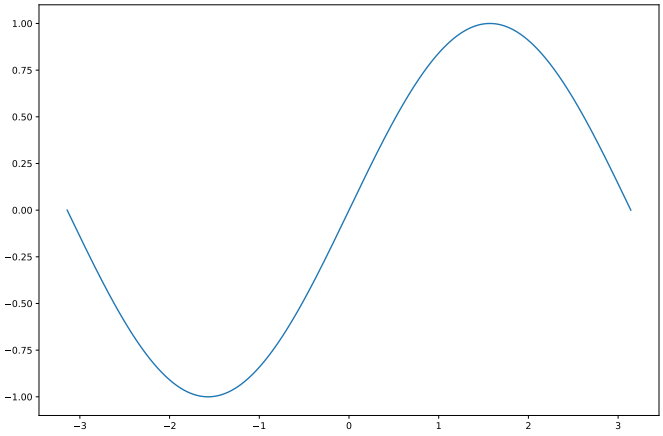



Figura 6.10: Graph of f(x)=sin(x)

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-np.pi,np.pi,1000)
y = np.cos(x)
plt.plot(x,y);
plt.show();
```

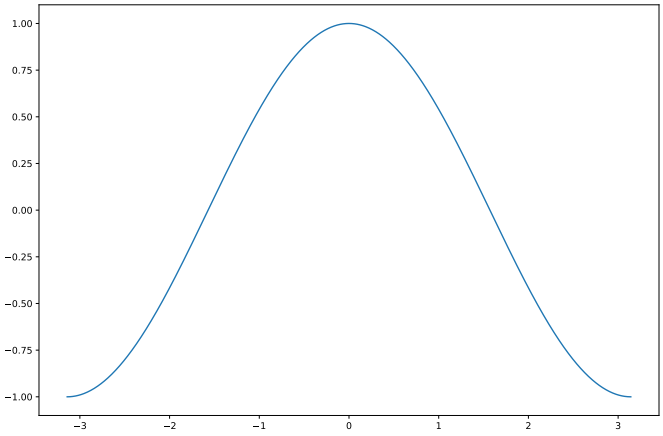



Figura 6.11: Graph of f(x)=cos(x)

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-np.pi,+np.pi,1000)
y = np.tan(x)
plt.scatter(x,y);
plt.show();
```

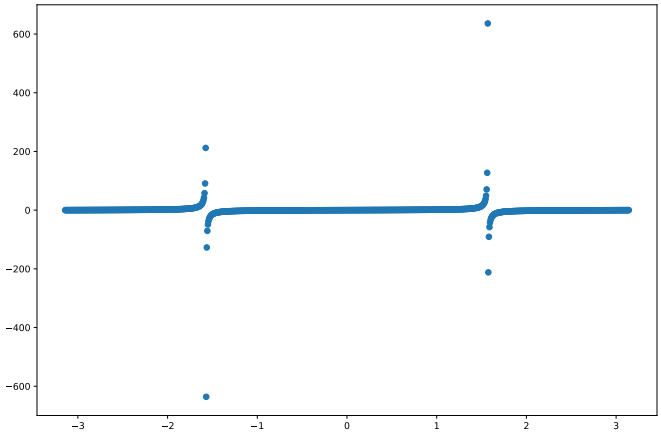



Figura 6.12: Graph of f(x)=tg(x)

## 6.8 Curves in Polar Coordinates

In the exercise presented in 6.8 we used the feature of polar coordinates. This feature transforms a graph which would be difficult to draw in rectangular coordinates (even more so because it is composed of two or more functions of x and y, when expressed in algebraic form and not just one) into a function of a single variable from the variable $\theta$, the angle formed by the line from the point in question to the origin. In this section we will draw some of these curves and incorporate graphical features that allow the determination of each point on the curve.

5. Draw for $\theta$ ranging from $0^o$ to $360^o$, the Archimedes spiral given by the expression $r = \frac{\theta}{2}$

In the figure 6.13 the corresponding graph can be seen. Attention should be paid to the fact that the argument (i.e. the angle) is always passed to the corresponding functions in radians, hence the multiplication by $\pi$ and division by 180.

```
import numpy as np
theta = np.linspace(0,360,361)*2*np.pi/180
r = theta/2
x = r*np.cos(theta)
y = r*np.sin(theta)

from matplotlib import pyplot as plt
import matplotlib as mpl
```

```
mpl.use('TkAgg')
g = plt.plot(x,y);
plt.show();
```

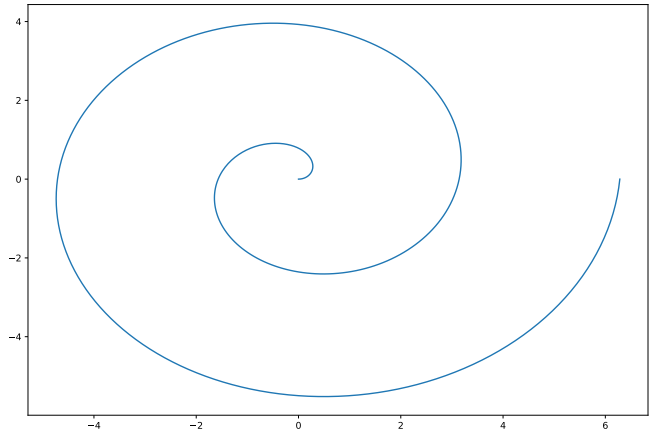



Figura 6.13: Drawing of Archimedes Spiral

6. Draw the function $r = 10\sin(3\theta)$, also known as *three-petal rose*.

See figure 6.14

```
import numpy as np
theta = np.linspace(0,360,361)*np.pi/180
r = 10*np.sin(3*theta)
x = r*np.cos(theta)
y = r*np.sin(theta)

from matplotlib import pyplot as plt
import matplotlib as mpl
mpl.use('TkAgg')
g = plt.plot(x,y);
plt.show();
```

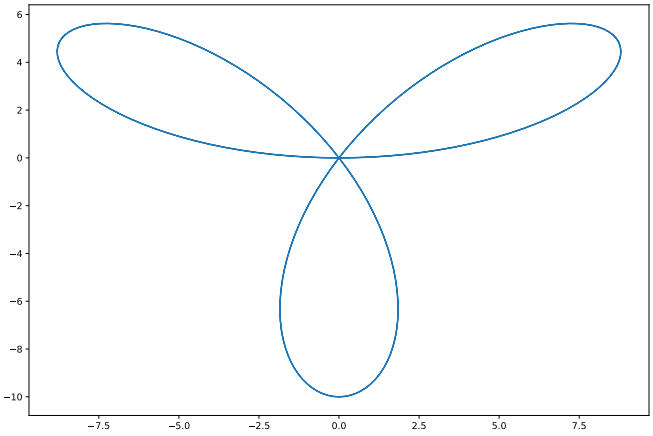



Figura 6.14: Drawing of the Three-Petalted Rose

7. Draw the *Lissajous* figure given parametrically by the expressions :

$$x = sin(2t) \tag{6.7}$$
$$y = sin(3t) \tag{6.8}$$

Figure 6.15 shows the associated graph. Note that the indication of the curve point was made by creating a new data series, with only two points, one of them fixed at (0;0) and another that is calculated according to the parameter $t$, which can be freely modified.

```
import numpy as np
t = np.linspace(0,360,361)*np.pi/180
x = np.sin(2*t)
y = np.sin(3*t)

from matplotlib import pyplot as plt
import matplotlib as mpl
mpl.use('TkAgg')
g = plt.plot(x,y);
plt.show();
```

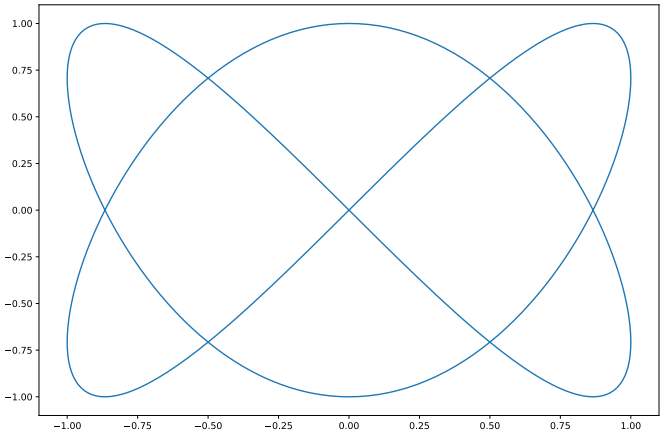



Figura 6.15: Drawing of the figure of Lissajous

8. Draw the graph of the path taken by a satellite orbiting a planet with an orbital period of 1 month, which planet orbits a star with a period of 1 year (or 12 months). Assume circular orbits and that the radius of the satellite's orbit is 20% of the radius of the planet's orbit.

Here we have a circular motion composed of two rotations. Thus, at each instant the satellite's position is composed of a circle with radius equal to 1 plus a circle with radius equal to 0.1. Thus we can decompose the $x$ and $y$ coordinates of the planetary orbit as:

$$x = x_p + x_s \tag{6.9}$$
$$y = y_p + y_s \tag{6.10}$$

When we apply the formulas for circular motion in polar coordinates, these expressions become :

$$x = 10\cos(2\pi.1.t) + 1\cos(2\pi.12.t) \tag{6.11}$$
$$y = 10\sin(2\pi.1.t) + 1\sin(2\pi.12.t) \tag{6.12}$$

The 6.12 expressions can be calculated on the columns of a data frame. The attention that must be paid concerns the parameter t. It must vary from 0 to 1 so that the argument of cos and sin varies from 0 to $2\pi$. To create a very smooth

curve, we will use 1,001 points between 0 and 1 for the parameter $t$. The graph and corresponding table can be seen at 6.16 and 6.3.

```
import numpy as np
t = np.linspace(0,1,1001)
xp = 10*np.cos(2*np.pi*t*1)
yp = 10*np.sin(2*np.pi*t*1)
xm = 2*np.cos(2*np.pi*t*12)
ym = 2*np.sin(2*np.pi*t*12)
x = xp + xm; y = yp + ym
df = pd.DataFrame({"x":x, "y":y})
g = plt.plot(x, y);
plt.show();
```

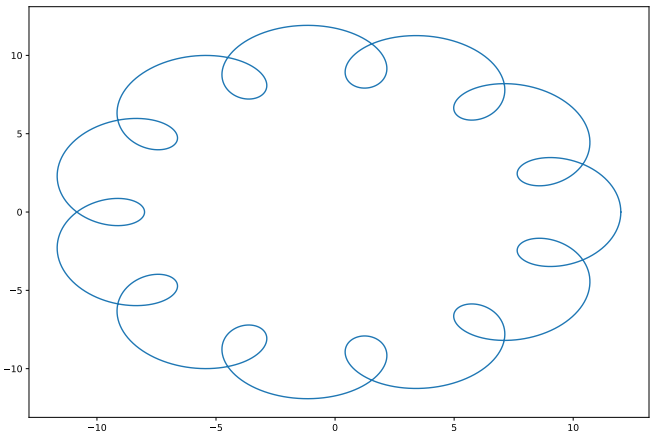



Figura 6.16: Graph of the orbit of a satellite of a planet, view from the star

Tabela 6.3: Table for calculating the orbit points of a satellite

| x | y |
|---|---|
| 12.00000 | 0.0000000 |
| 11.99412 | 0.2134851 |
| 11.97651 | 0.4261116 |
| 11.94728 | 0.6370259 |
| 11.90657 | 0.8453841 |
| 11.85462 | 1.0503567 |

## 6.9 Graphics in 3D

The most practical way to draw 3D surfaces is by `sympy`. To do this, we can use a function of two independent variables, each with its own calculation range. Let's look at an example:

9. Draw the graph of the function $z = x.y$ in the interval $x \in [0, 10]$ and $y \in [0, 10]$.

```
import numpy as np
import sympy as sp
from sympy.plotting import plot3d

x, y = sp.symbols("x y")
g = plot3d(x*y, (x,0,10), (y,0,10));
```

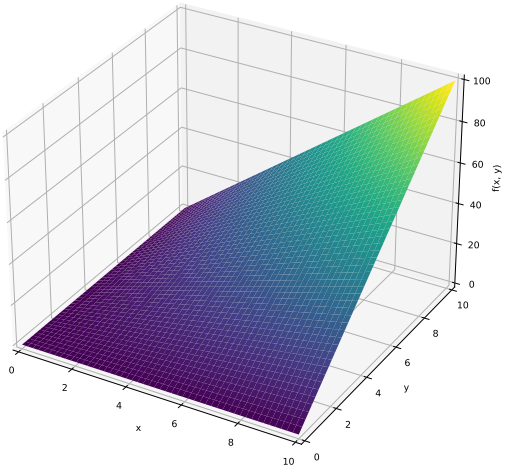



Figura 6.17: Graph of z = x.y

Another interesting graph to generate is the contour lines. It is nothing more than a view from above, with the heights painted in different colors. It is the standard shape for maps in a geography atlas. Such a graph for the function $z = x.y$ can be seen in the figure 6.18. Notice that we first create the plot boundaries on each axis ($x$ and $y$). Next we generate the drawing grid and then the $Z$ function from the grid values. With the grid values for $X$, $Y$ and $Z$, we use the `plt.contourf` function to get the desired effect.

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(0,10,101)
```

```
y = np.linspace(0,10,101)
X,Y = np.meshgrid(x,y)
Z = X*Y
g = plt.contourf(X, Y, Z);
plt.show()
```

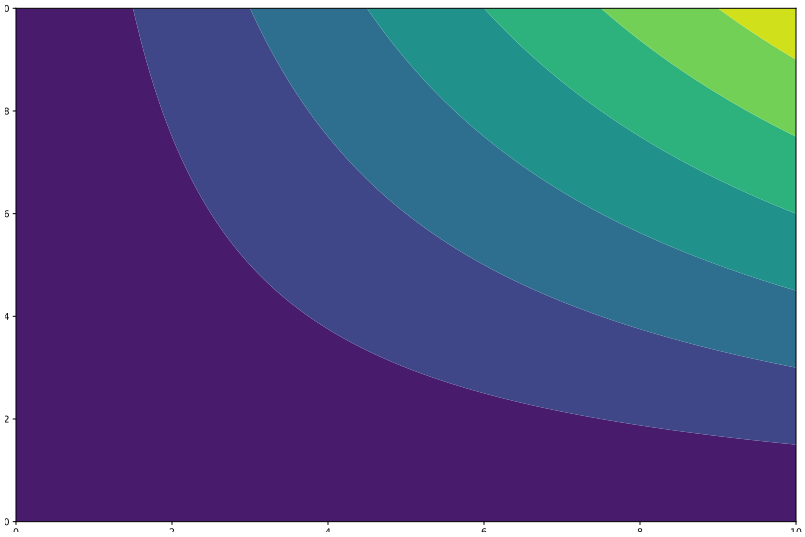


Figura 6.18: Contour graph for z = x.y

10. Draw the graph of the function $z(r) = \frac{sin(r)}{r}$ where $r = \sqrt{x^2 + y^2}$ on the interval $x \in [-6.28; 6.32]$ and $y \in [-6.26; -6.32]$.

```
import sympy as sp
from sympy.plotting import plot3d
x, y = sp.symbols("x y")
r = sp.sqrt(x**2 + y**2)
z = sp.sin(r)/r
g = plot3d(z, (x,-6.28, 6.32), (y,-6.28, 6.32));
```

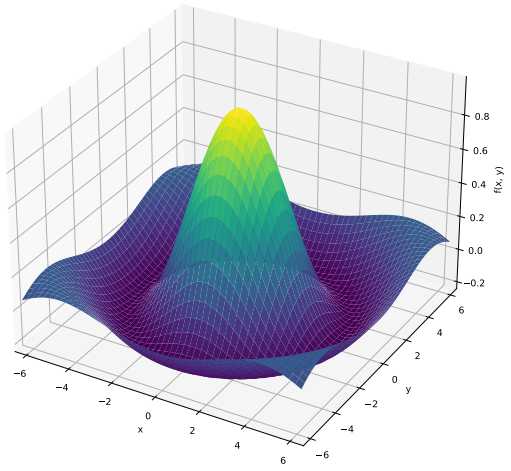



Figura 6.19: Graph of $z(r) = \frac{sin(r)}{r}$ where $r = \sqrt{x^2 + y^2}$

11. Draw the graph of the function $f(t) = 3e^{-4t}cos(5t) - 2e^{-3t}sin(2t) + \frac{t^2}{t+1}$ in the interval $x \in [0, 4]$

```
import sympy as sp
from sympy.plotting import plot
t = sp.symbols("t")
f=3*sp.exp(-4*t)*sp.cos(5*t)-2*sp.exp(-3*t)*sp.sin(2*t)+t**2/(t+1)
g = plot(f, (t,0,4))
```

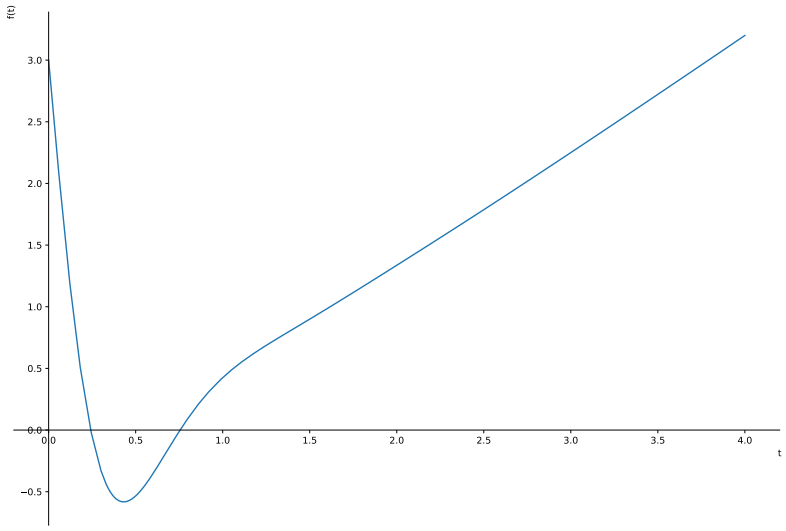



Figura 6.20: Graph of the function $f(t) = 3e^{-4t}\cos(5t) - 2e^{-3t}\sin(2t) + \frac{t^2}{t+1}$ on the interval $x \in (0, 4)$

12. Draw the graph of the function $z = -2x^3 + x + 3y^2 - 1$ in the interval $x \in [-10, 10]$ and $y \in [-10, 10]$.

```
import sympy as sp
from sympy.plotting import plot3d
x, y = sp.symbols("x y")
z=-2*x**3+x+3*y**2-1
g = plot3d(z, (x,-10,10), (y,-10,10));
```

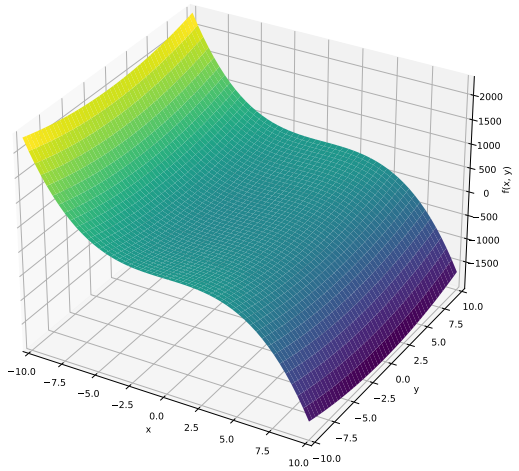



Figura 6.21: $z = -2x^3 + x + 3y^2 - 1$ in the interval $x \in (-10, 10)$ and $y \in (-10, 10)$

13. Draw the graph of the function $V(h, r) = 1/3\pi r^2 h$ in the interval $r \in [0, 4]$ and $h \in [0, 6]$.

```
import sympy as sp
from sympy.plotting import plot3d
h, r = sp.symbols("h r")
V=1/3*sp.pi*r**2*h
g = plot3d(V, (r,0,4), (h,0,6))
```

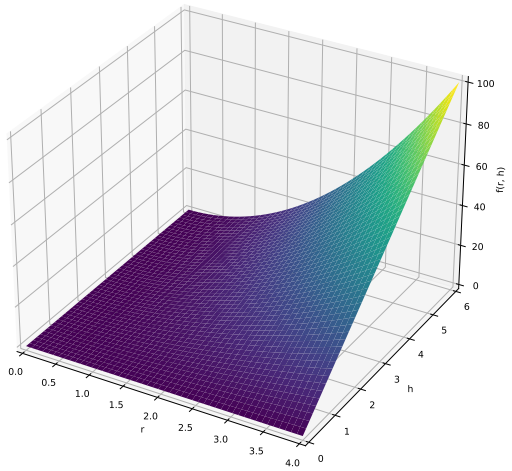



Figura 6.22: graph of the function $V(h, r) = 1/3\pi r^2 h$ on the interval $r \in (0, 4)$ and $h \in (0, 6)$

## 6.10 Exercises

1. Plot the graph of the following functions
   (a) $y = 5$
   (b) $x = 2$
   (c) $y = |x| + 2$
   (d) $xy = 10$
2. Plot the straight line that meets the following conditions:
   (a) Passes through the point (3,7) with slope $\frac{2}{3}$
   (b) Passes through the point (-4,8) with inclination $-\frac{1}{2}$
   (c) passes through the point (2,5) and is parallel to the x-axis
   (d) passes through the point (2,5) and is parallel to the y-axis
   (e) passes through the origin and is parallel to the line $y = 3x + 2$
   (f) passes through the origin and is perpendicular to the line $y = 3x + 2$
3. Plot the circle of radius $R = 4$ that passes through the points $(1, 1)$ and $(2, 3)$.
4. Plot the curve defined by the expression $x^2 + 4y^2 = 4$

5. Draw a regular hexagon with center at the origin and side equal to 1.
6. Starting from a regular hexagon with center at the origin and side equal to 1, draw a circumscribed circle around it.
7. Draw the graphs of the following given functions in rectangular coordinates :
   (a) $y = \frac{2x}{x^2+1}$
   (b) $y = \sin(x) - \frac{1}{3}\sin(3x)$
   (c) $y = e^{x^2}$
   (d) $y = diln(echoes(x))$
8. Draw the graphs of the following given functions in polar coordinates :
   (a) $r = \frac{1}{\sin(\theta)}$
   (b) $r = a(1 + cos(\theta))$ for $a$ equals 1,2 and 3
   (c) $r = \sec(\frac{\theta}{2})^2$
   (d) $r = 10\cos(4\theta)$
9. Draw the graphs of the following functions : (in all cases consider $x$ and $y \in [-10, 10]$ and $h$ equal to 0.2)
   (a) $z = x^2 + y^2$
   (b) $z = x + y$
   (c) $z = \frac{x-y}{x+y}$
   (d) $z = \sin(x)\cos(y)$
10. Draw the contour lines of the following functions : (in all cases consider $x$ and $y \in [-2, 2]$ and $h$ equal to 0,1)
    (a) $z = x^2 + y^2 - 2x^2y + 25$
    (b) $z = x\sin(y) + y\sin(x)$
    (c) $z = \frac{1}{xy}$
11. Describe the domain of f(x) = 8x / ( (x-1).(x-2) )
12. Convert from degrees to radians: a) 270o, b)-45o
13. Convert radians to degrees: a) $3\pi/2$ b) $-\pi/4$
14. Determine the length of the arc of a circle with 45° and a radius of 10 cm.
15. Demonstrate that $\frac{2tg(\theta)}{1+tg^2(\theta)} = sen(2\theta)$

16. Create a Python function that receives the x and y coordinates of two points and then calculates and returns the Euclidean distance and the Manhattan distance between them. NOTE: The Euclidean distance is the "direct"distance between the points. The Manhattan distance is calculated by walking along the "sides"of the triangle (as would occur in a city).

17. Knowing that the area of a triangle can be calculated by the formula $A = \frac{p}{2}\left[\left(\frac{p}{2} - a\right)\left(\frac{p}{2} - b\right)\left(\frac{p}{2} - c\right)\right]$, where $a$, $b$ and $c$ are the length of each side and $p$ is the perimeter, create a Python function that takes the length of each side and returns the area of the corresponding triangle. Assume that the user will provide values such that $a < b + c$ where $a$ is the largest value. This condition will ensure that the line segments form a triangle.

18. Create a Python function that receives the Cartesian coordinates of three points A, B, and C. From the distances between each pair of the three points the function should return one of the following messages: "The points form a triangle"or "The points do not form a triangle".

19. Create a Python function that takes three lengths, a, b, and c and returns whether or not the lengths form a triangle and if they do, what type of triangle is represented. To determine these conditions, assume that a is the largest of the three values entered. Under these conditions:

    (a) If $a > b + c$ then no triangle is formed

    (b) If $a^2 = b^2 + c^2$ then they form a right triangle

    (c) If $a^2 > b^2 + c^2$ then they form an obtuse triangle

    (d) If $a^2 < b^2 + c^2$ then they form an acute triangle

    (e) If $a = b$ and $b = c$ then they form an equilateral triangle

    (f) If $a = b$ or $b = c$ or $a = c$ and $a \neq b$ or $a \neq c$ then they form an isosceles triangle.

# Capítulo 7

# Rational Functions and Polynomials

Functions involving only integer powers of a variable $x$ are given the name polynomial functions. The study of their variation and the calculation of their roots, that is the values of $x$ such that $f(x) = 0$, occupy a central place in Algebra.

## 7.1 Fundamental Theorem of Algebra

The fundamental theorem of algebra says that if $f(x)$ is a polynomial of degree $n$, with n>0, then the function $f(x)$ will have $n$ roots. These roots can be real or complex (of the form $a + b.i$, where $i = \sqrt{(-1)}$ also known as the imaginary number. If there are complex roots they will always appear in pairs, conjugate complexes. This means that if $f(a + b.i) = 0$ then $f(a - bi)$ will also equal 0.

## 7.2 Linear Factorization Theorem

If $f(x)$ is a polynomial of degree $n$, where $n > 0$, then $f(x)$ has precisely $n$ linear factors such that: $f(x) = a_n(x - x_1)(x - x_2)...(x - x_n)$ where $x_1$, $x_2$, ..., $x_n$ are complex numbers. We must remember that every real number also belongs to the set of complexes, but with its imaginary part $b.i$ null.

## 7.3 Roots of a Polynomial Function

To find the zeros (roots of a function, whether polynomial or not) we must make $f(x) = 0$ and solve the resulting equation for $x$. If $x = x_0$ is one of the roots of

$f(x)$, if we calculate the polynomial quotient $f(x)/(x-x_0)$ the result must have remainder equal to 0.

Furthermore, if the polynomial has integer roots, knowing that $f(x) = a_n x^n + a_{n-1}x^{n-1}+...+a_2x^2+a_1x+a_0$ and that therefore $f(x) = a_n(x-x_1)(x-x_2)...(x-x_n)$, the integer roots must be divisors of the independent term. This can be verified by equating the two representations of the polynomial and comparing the terms of equal power in $x$. If we proceed this way we see that the independent term $a_0$ must be equal to $a_0 = a_n.(-x_1).(-x_2)...(-x_n)$

1. As an initial example let us calculate the roots of $f(x) = x^2 - 6x + 9$.

The divisors of are $\pm1$, $\pm3$, and $\pm9$. Testing the possible integer roots of $f(x)$ we see (see below) that $x = 3$ is a double root and $f(x)$ can be factored into $f(x) = (x-3)(x-3)$

```
import numpy as np
f = lambda x: x**2 -6*x + 9
f(-9), f(-3), f(-1), f(1), f(3), f(9)
```

```
(144, 36, 16, 4, 0, 36)
```

```
import sympy as sp
x = sp.symbols("x")
f = (x-3)*(x-3)
f.expand()
```

```
x**2 - 6*x + 9
```

The root could have been calculated directly through `sympy` by using the `solve` function as seen below:

```
import sympy as sp
x = sp.symbols("x")
f = x**2 - 6*x + 9
sp.solve(f)
```

```
[3]
```

2. Calculate the roots of $f(x) = x^4 - x^3 + x^2 - 3x - 6$

Knowing that $a_0 = a_n.(-x_1)(-x_2)...(-x_n)$ we will have in this case, if there are integer roots $-6 = (-x-1).(-x_2)(-x_3)(-x_4)$. The divisors of 6 are $\pm1$, $\pm2$, $\pm3$ and $\pm6$. Testing (see below)

```
import numpy as np
f = lambda x: x**4 -x**3 +x**2 -3*x -6
x = np.array([-6,-3,-2,-1,1,2,3,6])
y = f(x); y
```

```
array([1560,  120,   28,    0,   -8,    0,   48, 1092])
```

Then we have as real roots $x = -1$ and $x = 2$. Dividing $f(x)$ first by $x + 1$ we get:

$$\frac{x^4 - x^3 + x^2 - 3x - 6}{x + 1} = x^3 - 2x^2 + 3x - 6$$

Which can be confirmed by doing:

```
import sympy as sp
x = sp.symbols("x")
f = x**4 -x**3 +x**2 -3*x -6
f1 = x + 1
g = f/f1
g.simplify()
```

```
x**3 - 2*x**2 + 3*x - 6
```

Dividing the resulting polynomial $g(x) = x^3 - 2x^2 + 3x - 6$ by $x - 2$ we get:

$$\frac{x^3 - 2x^2 + 3x - 6}{x - 2} = x^2 + 3$$

Which can also be confirmed by doing:

```
import sympy as sp
x = sp.symbols("x")
f = x**4 -x**3 +x**2 -3*x -6
f1 = x + 1
f2 = x - 2
g = f/f1
h = g/f2
h.simplify()
```

```
x**2 + 3
```

This last polynomial $h(x) = x^2 + 3$ has two conjugate complex roots equal to $x = -\sqrt{3}i$ and $x = -\sqrt{3}i$.

We could have calculated all the roots of $f(x)$ directly using the `solve` function, which is shown below and confirms our previous calculation.

```
import sympy as sp
x = sp.symbols("x")
f = x**4 -x**3 +x**2 -3*x -6
sp.solve(f,x)
```

```
[-1, 2, -sqrt(3)*I, sqrt(3)*I]
```

## 7.4 Rational Functions

A rational function is given by $y = f(x) = \frac{P(x)}{Q(x)}$ where $P(x)$ and $Q(x)$ are polynomial functions. The rational function $f(x)$ will have discontinuity points where $Q(x) = 0$.

## 7.5 Horizontal and Vertical Asymptotes

$y = L$ is said to be a horizontal asymptote of the curve $y = f(x)$ if $\lim_{x\to+\infty} f(x) = L$ or $\lim_{x\to-\infty} f(x) = L$

$x = L$ is said to be a vertical asymptote of the curve $y = f(x)$ if $\lim_{x\to L} f(x) = +\infty$ or $\lim_{x\to L} f(x) = -\infty$

## 7.6 Application in Marketing: Effect of Advertising on Revenues

1. Suppose that the revenue $R$ of a product from the amount invested in advertising $x$ is given by the function $R(x) = \frac{100x+300}{x+10}$. Sketch the graph of this function, point out the values of $x$ that form its domain, and indicate which values have real meaning if $x$ is taken to be the amount invested in advertising.

To sketch the graph of the function we observe that it has a zero at $x = -3$ and the point $x = -10$ does not belong to its domain, since in this case $x + 10 = 0$. From a practical point of view, only non-zero values of $x$ represent possible amounts for advertising spending. Furthermore, if we calculate the value of the function for large values of both negative and positive $x$ we will see that the function approaches 100. Thus, this function has a horizontal asymptote at $x = 100$ and a vertical asymptote at $x = -10$.

We can plot the function graph in two ways through Python, by `numpy` and by `sympy`. Both ways can be seen below in 7.1 and 7.2.

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

x1 = np.linspace(-100.1,-12,1111)
y1 = (100*x1+300)/(x1+10)
plt.plot(x1,y1);

x2 = np.linspace(-8,100,1111)
```

```
y2 = (100*x2+300)/(x2+10)
plt.plot(x2,y2);

plt.show();
```

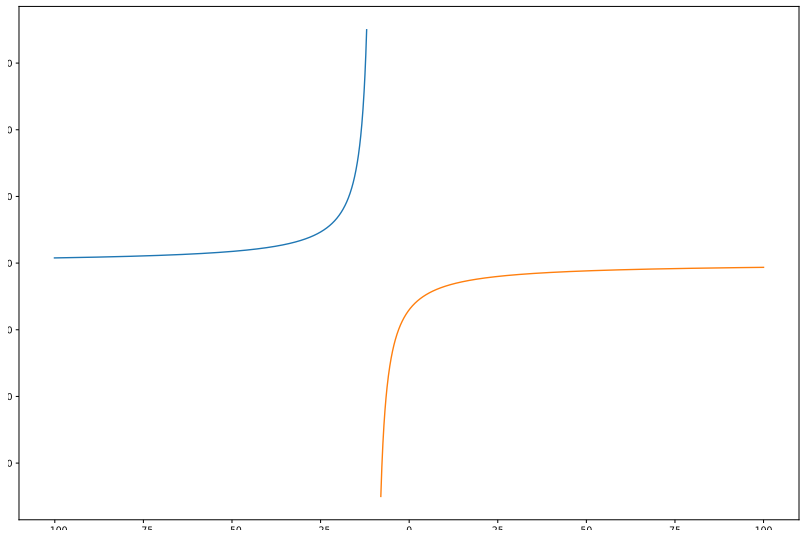


Figura 7.1: Sketch of f(x) via numpy

```
import sympy as sp
x = sp.symbols("x")
f = (100*x+300)/(x+10)
graf1 = sp.plot(f, (x,-100,-12), show=False)
graf2 = sp.plot(f, (x,-8,100), show=False);
graf1.append(graf2[0]);
graf1.show()
```

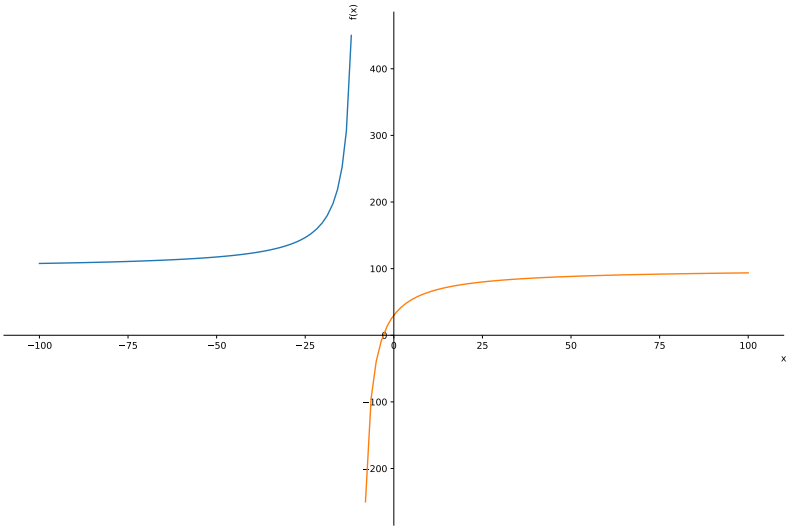



Figura 7.2: Sketch of f(x) via sympy

## 7.7 Exercises

As seen in the examples, both `numpy` and `sympy` allow you to perform operations with polynomials and rational functions. Below are exercises that can be solved using both `numpy` and `sympy`.

1. Evaluate the function $f(x) = \frac{500x+2000}{x+50}$ by determining:
   (a) Discontinuity points in the domain of $f(x)$
   (b) Zero points of $f(x)$
   (c) Horizontal asymptotes
   (d) Vertical asymptotes
   (e) Assuming that $f(x)$ represents the carrying capacity of an item from the amount of product available in stock, what is the meaning of $f(0)$.
   (f) If $f(x)$ is the revenue that can be earned by manufacturing a product $x$ what does $\lim_{x\to\infty} f(x)$ mean?
2. Suppose that the supply function of a product is given by the function $f(x) = \frac{2000x+400}{x+40}$ where $f$ is the quantity in tons and $x$ is the unit price in Real per ton. Based on these data do the following exercises:
   (a) Plot the graph of $f(x)$, determining: a) points of zero, b) points of discontinuity in the function domain, c) horizontal asymptotes and d) vertical asymptotes

(b) What does $f(0)$ mean from the mathematical point of view and from the managerial point of view in this case?

(c) What does $\lim_{x\to\infty} f(x)$ mean from both a mathematical and managerial point of view in this case?

3. suppose that a firm's cost of production is given by the function $f(x) = 4x^2 + 12x + 100$ where $x$ is the quantity of product manufactured (in thousands of units). From this function determine:

   (a) The total cost to produce 1000, 2000, 4000, and 20000 units

   (b) Average production cost per unit manufactured for each of the above production volumes

   (c) Suppose the average cost is described by a function $g(x)$ where $x$ is the quantity manufactured. What is the algebraic expression for $g(x)$

   (d) What is the managerial meaning of $f(0)$ and of $\lim_{x\to\infty} f(x)$

   (e) What is the managerial meaning of $\lim_{x\to 0} g(x)$ and $\lim_{x\to\infty} g(x)$

   (f) Sketch the graph of $f(x)$

   (g) Sketch the graph of $g(x)$

4. Expand and simplify the following polynomial expressions:

   (a) $4(x+6) + 3(2x-5)$

   (b) $(x+5)(x+2)$

   (c) $(\sqrt{x} + \sqrt{y})(\sqrt{x} - \sqrt{y})$

   (d) $(3x+2)^3$

   (e) $(x+3)^2$

5. Factor the following polynomial expressions:

   (a) $8x^2 - 50$

   (b) $x^2 + \frac{5}{2}x - 6$

   (c) $x^3 - 3x^2 - 4x + 12$

   (d) $x^4 + 125x$

   (e) $3x^{\frac{5}{2}} - 9x^{\frac{3}{2}} + 6x^{\frac{1}{2}}$

   (f) $x^5y^2 - 4x^2y^2$

6. Simplify the following polynomial expressions:

   (a) $\frac{x^2+5x+4}{x^2+2x+1}$

   (b) $\frac{x^2}{x^2-16} - \frac{x+1}{x+4}$

(c) $\frac{2x^2-x-1}{x^2-81} \cdot \frac{x+9}{2x+1}$

(d) $\frac{\frac{x}{y}-\frac{y}{x}}{\frac{1}{x}-\frac{1}{y}}$

7. Evaluate the value of the polynomial $p(x) = x^5 - 7\,x^4 + 16\,x^2 + 25\,x + 52$ at the point $x = 1$
8. Calculate the value of the polynomial $p(x) = x^5 - 7\,x^4 + 16\,x^2 + 25\,x + 52$ at the point $x = 1 + j$
9. Calculate the quotient of the division of the polynomial $p(x) = x^7 - 3x^5 + 5x^3 + 7x + 9$ by $2x^6 - 8x^5 + 4x^2 + 10x + 12$
10. Calculate the scalar product of the polynomial $x^4 + 2x^3 + 3x^2 + 4x + 5$ by $-2x^4 + 6x^3 - 3x^2 + 8x + 7$
11. Calculate the offset of the polynomial $x^2 - 6x + 20$ by $x + 5$
12. Draw the plot of the real, imaginary, and modulus of $Z(\omega)$ as a function of frequency $\omega$ for $Z(\omega) = 10 + \frac{10^4 - j10^6/\omega}{10 + j(0.1\omega - 10^5/\omega)}$ with $\omega$ ranging from 0 to $2,200 rad/s$
13. Calculate the roots of the polynomial: $p(x) = x^5 - 7\,x^4 + 16\,x^2 + 25\,x + 52$, in both literal and numerical forms.
14. Determine the coefficients of the polynomial formed by the roots: [-1, -2, -3, 4+5j, 4-5j] by ‘sympy‘.

# Capítulo 8

# Exponential and Logarithm Functions

Two functions of great importance in the study of Management are the Exponential and Logarithm functions which are studied below.

## 8.1 The Exponential Function

An exponential function is given by the expression $y = f(x) = b.a^x$, with $a > 0$, $b \neq 0$ and $a \neq 1$.

If $a > 1$ the exponential is said to be increasing (8.1 and 8.2).

```
import numpy as np
from matplotlib import pyplot as plt
b = 5
a = 2
x = np.linspace(-2,2,100)
y = b*(a**x)
plt.plot(x,y);
plt.show();
```

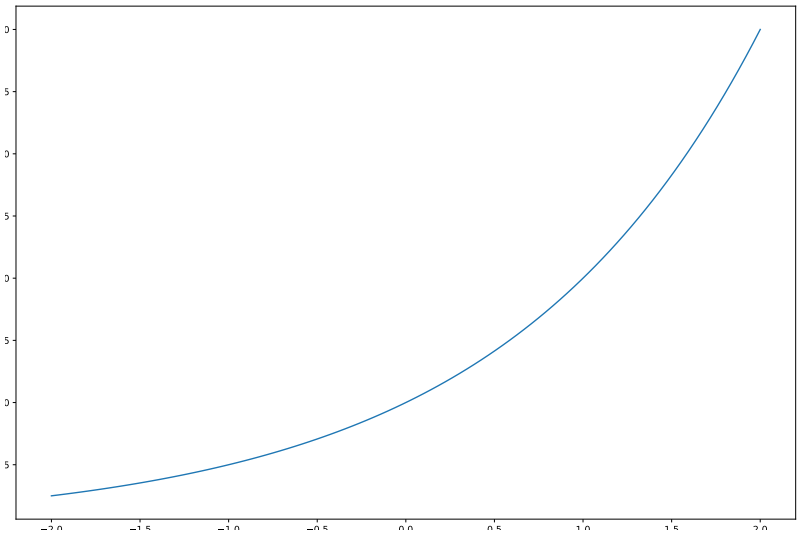


Figura 8.1: Increasing exponential with b=5, a=2

```
import sympy as sp
x = sp.symbols("x")
b = 5
a = 2
y = b*(a**x)
sp.plot(y, (x,-2, 2));
```

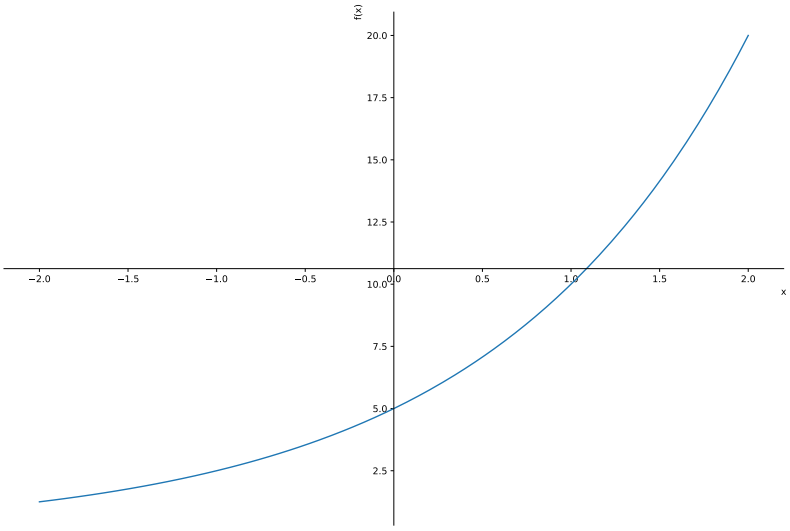



Figura 8.2: Increasing exponential with b=5, a=2 via sympy

If $a < 1$ the exponential will be said to be decreasing (8.3 and 8.4)

```
import numpy as np
from matplotlib import pyplot as plt
b = 5
a = 0.5
x = np.linspace(-2,2,100)
y = b*(a**x)
plt.plot(x,y);
plt.show();
```

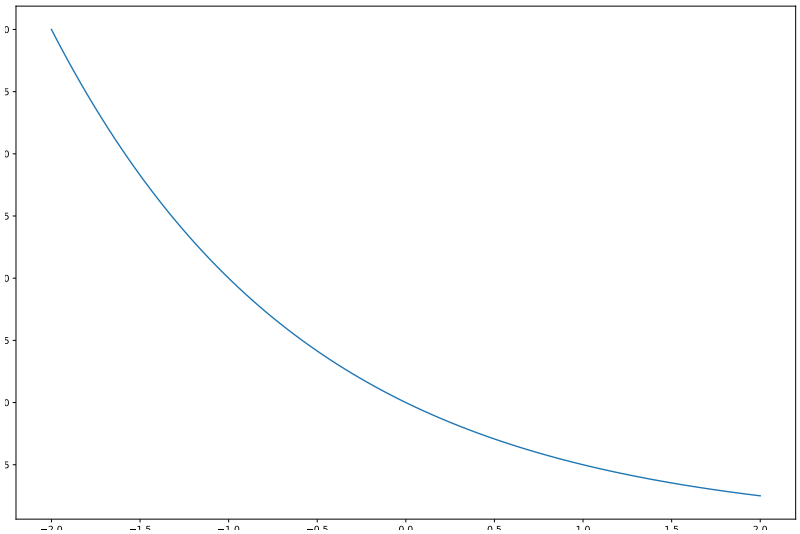


Figura 8.3: Decreasing exponential with b=5, a=2 and k=-1

```
import sympy as sp
x = sp.symbols("x")
b = 5
a = 0.5
y = b*(a**x)
sp.plot(y, (x,-2, 2));
```

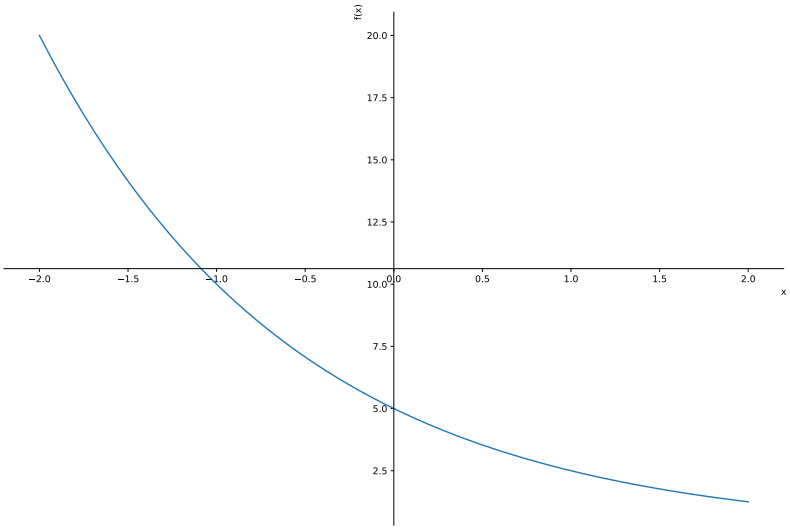



Figura 8.4: Increasing exponential with b=5, a=2 via sympy

If $a = 1$ the function transforms (degenerates) into the constant function $y = f(x) = b$ (see 8.5 and 8.6).

```
import numpy as np
from matplotlib import pyplot as plt
b = 5
a = 1
x = np.linspace(-2,2,100)
y = b*(a**x)
plt.plot(x,y);
plt.show();
```

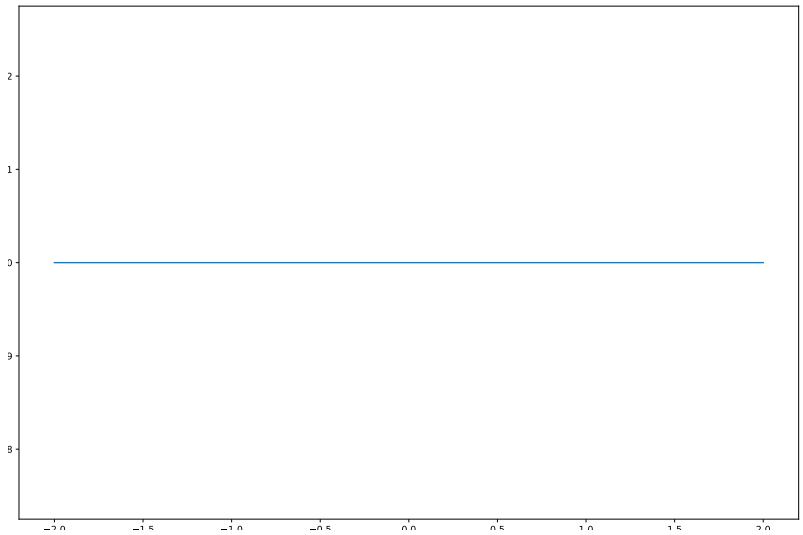


Figura 8.5: Exponential 'degenerates' into the constant function with b=5, a=1

```
import numpy as np
x = sp.symbols("x")
b = 5
a = 1
y = b*(a**x)
sp.plot(y, (x,-2, 2));
```

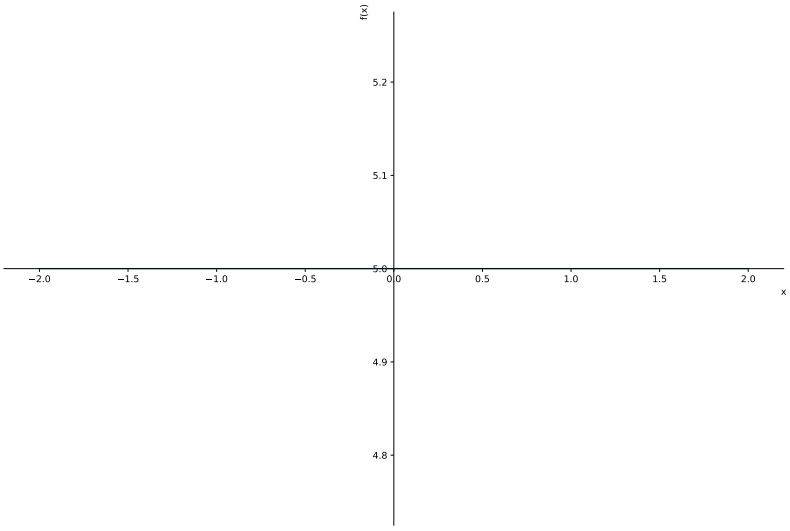



Figura 8.6: Exponential 'degenerate' on constant function with b=5, a=1, via sympy

A special and very important case occurs when $a = e$ where $e$ is Euler's number, an irrational constant whose approximate value is 2.71828...

## 8.2 The Logarithm Function

The logarithm function is the inverse of the exponential function. So if $y = a^x$ we say that $log_a(y) = x$. If $a = 10$ we have the decimal logarithms, usually represented by $log(y) = x$ (and hence $10^x = y$). In case $a = e$ we have the natural or neperian logarithms, represented by $ln(y) = x$ (which implies in this case that $e^x = y$).

## 8.3 Properties of Logarithms

The main algebraic properties of logarithms (independent of their basis of calculation) are:

1. $ln(A.B) = ln(A) + ln(B)$
2. $ln(\frac{A}{B}) = ln(A) - ln(B)$
3. $ln(A^k) = k.ln(A)$
4. $ln_B(A) = \frac{ln(A)}{ln(B)}$

The graphical form of the logarithm function can be seen below for base 2 (8.7) and for base $\frac{1}{2}$ (8.8). Note that in Python, in both `numpy` and `sympy` the function `log` refers to the natural logarithm.

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(0.1,10,1000)
y = np.log(x)/np.log(2)
plt.plot(x,y);
plt.show();
```

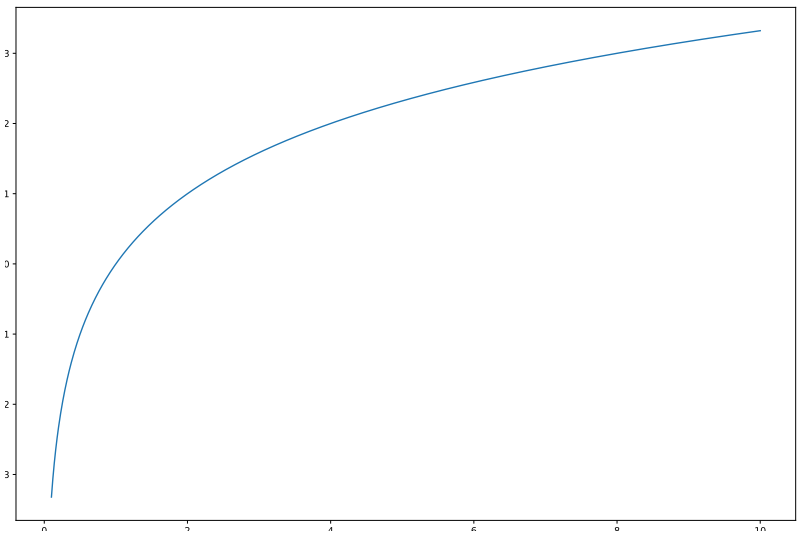


Figura 8.7: Logarithm function in base 2

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(0.1,10,1000)
y = np.log(x)/np.log(0.5)
plt.plot(x,y);
plt.show();
```

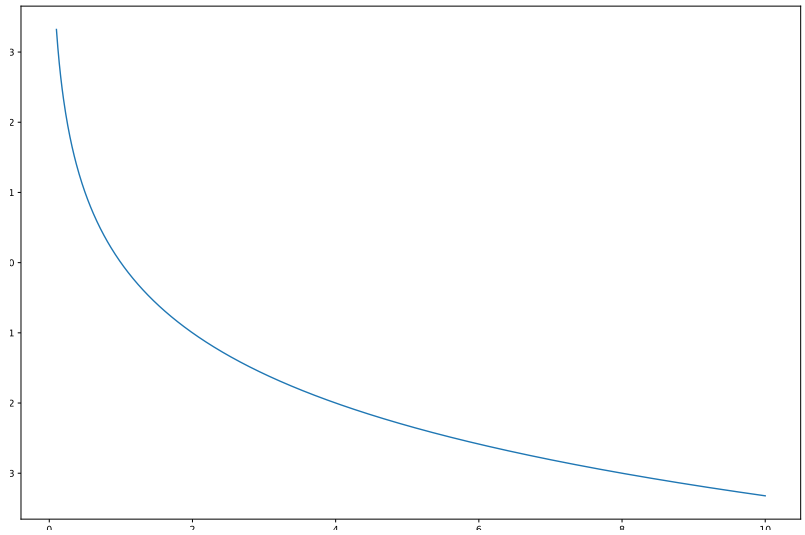


Figura 8.8: Logarithm function in base 0.5

## 8.4 Application to Geography: Population Growth Model

1. Suppose a region that has initial population $P_0 = 1000$, which will grow at a rate of 0.02 per year. What expression will determine the approximate number of people who will be living in this region $t$ years after the initial instant $t_0$?

Solution: $P(t) = P_0(1 + 0.02)^t$

1. A city has 20,000 inhabitants today and that number grows at a rate of 0.03 per year. Based on this information ask:
   (a) What is the number of inhabitants in 10 years?
   (b) If ten years from now the number of inhabitants equals 30,000, having started from a total of 20,000 inhabitants, what would be the annual growth rate?

Solution:

The amount of inhabitants in 10 years is calculated as follows (via `numpy`)

```
import numpy as np
p0 = 20000
r = 0.03
p = lambda t:p0*(1+r)**t
p(10)
```

```
26878.327586882446
```

The annual growth rate for a population in 10 years of 30,000 people, starting growth at 20,000 people is calculated below (via `sympy`). Note that this time the numerical solver of `sympy` via the `nsolve` function was used. This solver requires a starting point to find the solution, which we chose as zero.

```
import sympy as sp
r = sp.symbols("r")
p = 20000*(1+r)**10
sp.nsolve(p-30000,r,0)
```

```
0.0413797439924106
```

## 8.5 Application in Finance: Compound Interest

The same population growth logic can be used to calculate the future value of an investment subjected to a compound interest rate. Based on this information we are asked to solve the next exercise:

1. A new car is worth 20,000 today and depreciates by 0.15 per year. In how long will its value halve from its original value?

```
import sympy as sp
t = sp.symbols("t")
p = 20000*(1-0.15)**t
sp.nsolve(p-10000,t,0)
```

```
4.26502428179873
```

## 8.6 Exercises

The following section includes exercises in Financial Mathematics, given the wide use of the exponential and logarithm functions in solving problems in finance.

1. Knowing that the relationship between the present value (PV) and the future value (FV) of an amount, when the interest rate (i) has continuous capitalization, is given by FV = PV.ei.t , where t is time measured in years, determine the interest rate (i), knowing that FV=200, PV=100, and t=2 years.
2. The present value of an investment is 100 and it is known that in 3 years its value will be 200. Calculate what is the continuous interest rate of this investment.

3. What is the continuous compounding interest rate that makes an amount of $100,000$ become $90,000$ in 2 years?

4. Suppose a company issues bonds costing 200 and paying compound interest monthly. Interest accrues until the bond reaches maturity. Suppose that after 5 years the bond has a value of 500.00. What is the annual interest rate?

5. Suppose a couple has made an investment of R$4,000 each year for four years in an investment that yields 8

6. Find the present value of R$10,000 to be paid out at the end of 5 years if the money is invested at 12% compound interest continuously?

7. A portfolio of stocks has appreciated in value from $100,000$ to $117,000$ over 2 years. What interest rate, compounded continuously, was earned on this investment?

8. What is the future value of a $1,000$ investment made today, 2.5 years from now, assuming that the interest rate is 7.5% per year. Present the complete result, the integer and fractional parts of it to two decimal places.

9. Create a Python function that takes as parameters: a) PV ("present value"), b) i ("interest rate"), and c) n ("number of periods of the investment") and returns the future value of an investment after n periods under compound interest.

10. Suppose two amounts of 1000 and 2000 Reais deposited in different bank accounts, which pay simple interest equal to: 10% and 5% per period respectively. Create a Python function that receives the initial amounts in each account and the interest rate for each and returns how many periods will be required for the account with smaller balance to exceed the account with larger balance.

11. Suppose that the population of a city A is about $100,000$ inhabitants with an annual growth rate of 2.8%, and the population of a city B is about $200,000$ inhabitants with an annual growth rate of 1.3%. Determine using a Python function that will take as parameters the initial population values in each city and their associated growth rates and return how many years it will take for the population of city A to exceed the population of city B.

## 8.7 Limits

## 8.8 Introduction

We will now delve deeper into our study of functions. As you can see from the last exercises in the previous chapter, a recurring theme is the analysis of the variation of the values of a function.

To assist us in this process we will use Python's own concept of functions. A Python function can be used to get the value of an expression from the value of a variable. So let's create a function that receives a value and performs the desired calculation for us.

To get such a result we first need to tell Python what name our function will have, for example $f$. To just define a name as a function just do:

```
def f():
  return()
```

In the above case we tell Python that `f` is a function (via `def`), that this function does not currently require any arguments (via the parentheses) and that it does not perform any computations, but returns a null (empty) value (via `return()`). Let's improve on this definition.

```
def f(x):
  return()
```

Now `f` is a function that will take an argument `x`, but does not yet perform any computation on it. To make it operational we then do:

```
def f(x):
  y = x**2 + 3*x + 40;
  return(y)
```

Now we have a complete Python function. It is called `f`, needs an argument, called `x`, performs the operation $x^2 + 3 * x + 40$ with this argument, assigns the result of this operation to a variable `y` and returns the value of this variable as the result.

To calculate the value of this function at $x = 6$ we can now simply do:

```
f(6)
```

```
94
```

## 8.9 Calculating repeated values of a function.

In the case of a spreadsheet, to calculate a series of values of a function we can put the values of interest of the variable $x$ in a column and simply copy the function formula into the column next to it.

If we are working in Python, we take advantage of the possibility of assigning a sequence of values to a variable x (which will be called `vector') and use this vector as an argument to the desired function. Thus, if the input vector has six values (-2, -1, 0,`

2, 4, and 5 for example), the output will also be a vector of six elements, each one being the result of the function for the specific input value. A common misconception among novice users of Python is to assume that alistis equivalent to anvector. Lists do not automatically allow you to perform operations element by element. To have such functionality we need to resort tonumpy‘ arrays. See the following:

```
import numpy as np
x = np.array([-2, -1, 0, 2, 4, 5])
```

Here we made x receive the sequence of numbers (-2, -1, 0, 2, 4, and 5) in the form of a `vector`. We have actually converted the list `[-2, -1, 0, 2, 4, 5]` into an array `numpy` with the same elements. Now let's use this array `x` as an input argument to `result` and observe the result:

```
f(x)
```

```
array([38, 38, 40, 50, 68, 80])
```

You could say that an Excel spreadsheet is faster to use. But when the calculations increase in complexity Python becomes much more practical or even essential for the task we want to perform.

## 8.10 Numeric Approximation via Python

Let's now work with higher math problems. In the case of this section we will show how to calculate function limits with Python, either purely numerically (with `numpy`) or algebraically (with `sympy`). An assumption of this text is that the reader is a spreadsheet user who wishes to experiment with using Python to solve calculus problems.

Limit is defined as the value to which the function $f(x)$ tends as the independent variable $x$ approaches a certain value. Most often, to determine the limit of $f(x)$ when $x$ tends to a specific value $a$, it is enough to calculate the value of $f(a)$.

Suppose one wants to calculate $\lim_{x\to 3} x^2 + 2\,x + 4$. In this case the result would be immediate, since $f(3) = 3^2 + 2\,3 + 4 = 19$. The calculation of the limit is not always direct. Sometimes the simple replacement of the variable $x$ by the value to which it tends produces: a) a forbidden operation (a division by zero), b) an operation whose result is outside the computational capacity of the machine (a multiplication by very large numbers) or c) because the software is not prepared to handle the necessary numbers (a square root of negative numbers). In these situations it is necessary to perform a numerical calculation of it or to eliminate the error situation by algebraic manipulation.

For example, calculate $\lim_{x\to 0} frac1x$. To determine this limit we must calculate the value of $\frac{1}{x}$ for values of $x$ closer and closer to 0 without however reaching 0. This is the so-called approximation calculation. Look at the table below:

| x | 1/x |
|---|---|
| 1 | 1 |
| 0,1 | 10 |
| 0,01 | 100 |
| 0,001 | 1000 |

From the table above we observe that as $x$ approaches 0 the value of $\frac{1}{x}$ increases more and more. So we can extrapolate this relationship and say that if it were possible for $x$ to reach 0 (the smallest value), the value of $\frac{1}{x}$ would reach the largest number (a numerical impossibility). This concept is called *infinite* and represented by the symbol $\infty$. Thus, from a formal point of view we say that $lim_{x\to 0}\frac{1}{x} = \infty$

## 8.11 Exercises

### 8.11.1 Calculus of Limits

1. Polynomial functions, with square root and involving indeterminacies of type $+\infty - \infty$
   (a) $\lim_{x\to -\infty} -x^3 + 3x + 10$
   (b) $\lim_{x\to \infty} \sqrt[3]{x+a} - \sqrt[3]{x}$
   (c) $\lim_{n\to x_0} \frac{\sqrt[n]{x} - \sqrt[n]{x_0}}{x-x_0}$
   (d) $\lim_{n\to \infty} \left|\sqrt{x+1} - \sqrt{x+3}\right|$
2. Rational functions and involving indeterminacies of type $\frac{0}{0}$ or $\frac{infty}{\infty}$
   (a) $\lim_{x\to \infty} \frac{1000x}{x^2-1}$
   (b) $\lim_{x\to \infty} \frac{(2x+2)^3(3x-4)^2}{x^5-1}$
   (c) $\lim_{x\to \infty} \frac{(2x^2+3x+4)}{\sqrt{x^4-1}}$
   (d) $\lim_{x\to 2} \frac{x^2-6x+8}{x^2-5x+6}$
   (e) $\lim_{x\to \infty} \frac{3x-1}{x+4}$
   (f) $\lim_{x\to \infty} \frac{3x^2-3}{x^2-4}$
   (g) $\lim_{x\to \infty} \frac{x^2-6}{x^-4}$

(h) $\lim_{x\to\infty} \frac{6x^2-3}{x^2-4}$

(i) $\lim_{x\to\infty} \frac{4(x-1)(x-2)}{(x-3)(x-4)}$

(j) $\lim_{x\to\infty} \frac{2x^2+3x+4}{\sqrt{x^4-1}}$

(k) $\lim_{x\to 1} \frac{x^3+5x^2-7x+9}{x^2+1}$

(l) $\lim_{x\to 2} \frac{x^2+5x-14}{x^2-2x}$

(m) $\lim_{x\to\infty} \frac{x^2-x-1}{x^3+x}$

(n) $\lim_{x\to\infty} \frac{x^4-3x^2+7x-1}{2x^4+3x+9}$

(o) $\lim_{x\to 2} \frac{x^3-2x^2-x+15}{x^2+4}$

(p) $\lim_{x\to 2} \frac{x^2+5x-14}{x^2-2x}$

(q) $\lim_{x\to-\infty} \frac{x^3-3x^2+4x-2}{3x^3+3x+9}$

(r) $\lim_{x\to\infty} \frac{x^2-2x+1}{x^3-3x}$

(s) $\lim_{x\to 2} \frac{-x^4+2x^2+8}{x-2}$

(t) $\lim_{x\to 1} \frac{x^2-3x+2}{x^2+1}$

(u) $\lim_{x\to 1} \frac{(2x+5)\ln(x)}{(3x-1)}$

(v) $\lim_{x\to 2} \frac{x^2+5x-14}{x^2-2x}$

(w) $\lim_{x\to\infty} \frac{x^2-x-1}{x^3+x}$

(x) $\lim_{x\to\infty} \frac{-x^4-3x^2+7x-1}{2x^4+3x+9}$

3. Exponential functions and involving indeterminacy of type $1^\infty$

(a) $\lim_{x\to 0} (1+ax)^{\frac{1}{x}}$

(b) $\lim_{x\to\infty} (1+\frac{0,4}{x})^x$

(c) $\lim_{x\to 0} (1+0,4\,x)^{\frac{1}{x}}$

(d) $\lim_{x\to\infty} e^{-x^2}$

(e) $\lim_{x\to-\infty} e^{x^2}$

(f) $\lim_{x\to\infty} \frac{e^x}{e^{2x-1}}$

(g) $\lim_{x\to\infty} \frac{e^{3x}}{e^{x-1}}$

(h) $\lim_{x\to\infty} \frac{e^x}{e^{3x-1}}$

(i) $\lim_{x\to\infty} \left(\frac{x}{x+1}\right)^x$

(j) $\lim_{x\to\infty}\left(\frac{x-1}{x+3}\right)^{x+2}$

4. Trigonometric functions:
   (a) $\lim_{x\to 0}\frac{\sin(3\,x)}{x}$
   (b) $\lim_{x\to 0}1+\sin(x)\frac{1}{x}$
   (c) $\lim_{x\to a}\frac{\cos(x)-\cos(a)}{x-a}$
   (d) $\lim_{x\to 0}\frac{\sin 3x}{x}$
   (e) $\lim_{n\to 0}\frac{\cos(mx)-\cos(nx)}{x^2}$
   (f) $\lim_{x\to 0}\frac{1-\cos(x)}{x^2}$

### 8.11.2 Sketch Graphs

For the following functions determine: a) domain and image, b) roots and points of intersection with the y-axis, c) left and right limits of discontinuity points, d) limits for $x\to\pm\infty$, e) vertical and horizontal asymptotes, f) intervals of growth and decrease, g) points of local minimum and maximum, and h) sketch of the graph

1. First Degree Functions
   (a) $f(x)=-2x+1$
   (b) $g(x)=3x-5$
   (c) $f(x)=-2x+1$
   (d) $g(t)=3t-2$
2. Polynomial Functions
   (a) $j(x)=x^2-2x+1$
   (b) $g(x)=(x-4)^2+1$
   (c) $m(t)=(t-1)^3$
   (d) $n(x)=(x+1)^4-1$
   (e) $h(t)=t^2+2$
   (f) $h(t)=(t-2)^2$
   (g) $j(t)=(t+1)^2-2$
3. Functions involving roots
   (a) $f(t)=\sqrt{-4t+17}$
   (b) $n(x)=sqrtx^2-7x+6$

(c) $f(x) = \sqrt{x^2 - 5x + 4}$

(d) $l(x) = \sqrt{x}$

(e) $p(x) = sqrtx + 1$

4. Rational functions

(a) $h(x) = \frac{x-2}{2x-6}$

(b) $j(t) = frac2tt^2 + 16$

(c) $f(x) = \frac{1}{x}$

(d) $g(x) = \frac{2x-3}{x-1}$

(e) $h(x) = \frac{3x^2-7x+9}{4x-16}$

(f) $i(x) = \frac{4x^7-3x+15}{2x^7-6x}$

(g) $j(x) = frac-2x + 3$

(h) $l(x) = frac3x - 1 + 1$

(i) $m(x) = \frac{x^2+x}{x}$

(j) $f(x) = \frac{x^3-x^2-x+2}{x^3-4x^2}$

(k) $f(x) = \frac{4x^3+x^2-x-5}{x^3-8x^2}$

(l) $f(x) = \frac{x^2+2x-3}{x^2+3x}$

(m) $f(x) = \frac{x^2-3x+2}{100-x^2}$

(n) Given the function: $g(x) = \frac{x-6}{3-x}$. For this function also determine: the area of the triangle formed by the line tangent to the graph at $g(0)$ and the asymptotes of the function $g(x)$.

5. Rational and Square Root Functions

(a) $p(x) = \sqrt{\frac{x^2-8x+15}{-2x+8}}$

(b) $q(x) = \sqrt{\frac{x^3-4x}{(x+2)(-x+1)}}$

(c) $r(x) = \sqrt[3]{\frac{x^2-4x+3}{3x(2x-10)}}$

(d) $v(x) = \sqrt{\frac{(-x+2)(x^2-9)}{(4x-8)(-5x+15)}}$

(e) $w(x) = \sqrt[4]{\frac{x^{14}}{x^2+16}}$

(f) $y(x) = \sqrt[5]{\frac{x^4}{x^3+8}}$

(g) $f(x) = \frac{\sqrt{x^2+2x-6}}{1-x}$

6. Logarithmic and Exponential functions

   (a) $m(x) = \log(-4x - 12)$

   (b) $f(k) = \ln(k^4 + 5k^2 + 6)$

   (c) $r(x) = 2^x$

   (d) $s(x) = \log_2(x)$

   (e) $h(x) = \log(\frac{-x+4}{x+2})$.

   (f) $y = 2e^{x^2-4x}$.

7. Segment Defined Functions

   (a) $f(x) = \begin{cases} 2x - 5, \text{ se } x > 2 \\ x - 3, \text{ se } x \leq 2 \end{cases} \text{ em } p = 2$

   (b) $g(x) = \begin{cases} 1/x, \text{ se } x > 0 \\ x^2, \text{ se } x < 0 \\ 1, \text{se } x = 0 \end{cases} \text{ em } p = 0$

   (c) $h(x) = \begin{cases} x^2 + 5x - 1, \text{ se } x \neq 1 \\ 4x + 1, \text{ se } x = 1 \end{cases} \text{ em } p = 1$

   (d) $f(x) = \begin{cases} 4x - 1, \text{ se } x > 2 \\ 7, \text{ se } x \leq 2 \end{cases} \text{ em } p = 2$

   (e) $g(x) = \begin{cases} 4x, \text{ se } x < 0 \\ x^3, \text{ se } x > 0 \text{ em } p = 0 \\ 0, \text{se } x = 0 \end{cases}$

   (f) $h(x) = \begin{cases} x^2 + 2x + 1, \text{ se } x \neq 1 \\ 2(x + 1), \text{ se } x = 1 \end{cases} \text{ em } p = 1$

   (g) $f(x) = \begin{cases} 3x - 1, \text{ se } x > 1 \\ x + 1, \text{ se } x \leq 1 \end{cases} \text{ em } p = 2$

   (h) $f(x) = \begin{cases} 1/x, \text{ se } x > 0 \\ -1/x, \text{ se } x < 0 \text{ em } p = 0 \\ 0, \text{se } x = 0 \end{cases}$

### 8.11.3 Applications

1. Let $g(x) = x^3 - x^2 - x + 4$ be continuous and derivable. Judge the following statements as true (V) or false (F):

   (a) Dom = IR

   (b) Im = IR – {4}

(c) The function grows in the interval ]- ∞;-1/3] ∪ [1;+∞[.

(d) The function has upward concavity in the interval [-1/3;+∞[

(e) The point x = 1 is the minimum point of the function.

(f) The point x = -1/3 is the inflection point of the function.

2. Decide whether the following statements are true (V) or false (F), justifying your answer.

(a) The function $p(x) = \frac{x^3-1}{1-x^2}$ has as domain any $x \neq 1$ and the image of the $g(x) = \sqrt{x^2 - 5x + 6}$ is any real number.

(b) The break-even point of a product whose total cost function is $C(q) = 20q + 50$ and whose total revenue is $R(q) = 30q$ occurs for 6 units sold.

(c) If the demand for a product is linear and we know that we sell 100 units at $R\$2 and sell 50 units at R and 3$, then the demand function is $p(q) = -\frac{q}{50} + 4$ .

(d) The inverse of the function $f(x) = x^2 + 4$ has as image the set of reals.

(e) A function admits a horizontal asymptote $y = a$ when $\lim_{x\to\infty} f(x) = a$ .

(f) The function $p(x) = \frac{x^3-1}{x^2-9}$ has as domain any x other than 3 or -3 and the image of the function $g(x) = \sqrt{-x^2 + 5x - 6}$ is any non-negative real number.

(g) The break-even point of a product whose total cost function is $C(q) = 20q + 100$ and whose total revenue is $R(q) = 25q$ occurs for 20 units sold.

(h) If the demand for a product is linear and we know that we sell 100 units at R\$2 and sell 50 units at R\$3, then the demand function is $p(q) = \frac{-1}{100}q + 4$.

(i) The inverse of the function $f(x) = x^2 + 1$ has as image only the non-negative real numbers.

(j) A function admits vertical asymptote $y = a$ when $\lim_{x\to a} f(x) = \pm infty$ .

3. In 2017 Facebook is expected to reach the 2 billion active users mark, with only half of the Earth's population, i.e. 3.5 billion people, having access to the Internet. The upward curve of the number of users can be represented by the logistic function $U(t) = \frac{L}{1+e^{-k(t-t_0)}}$. Where $t$ is time in years, $U(t)$ is number of billion active users. For the Facebook case $L$ equals 3.5 billion users, $k$ equals 0.3 and $t_0$ equals 12 years. So the logistic function for the Facebook case is: $U(t) = \frac{3.5}{1+e^{-0.3(t-12)}}$. Thus answer:

(a) What are the domain and the intercepts of the function $U(t)$, considering the Facebook case?

(b) Make the study of the growth of the function $U(t)$, and answer whether there are points of maximum and/or minimum.

(c) Make the study of the concavity of the function $U(t)$, and answer whether there are any inflection points. What do the inflection point(s) found mean for the Facebook infrastructure forecast?

(d) Perform analysis of the asymptotes of the function $U(t)$. What does $\lim_{t\to+\infty} U(t)$ mean for Facebook?

(e) Sketch the graph of the function $U(t)$ for the case of Facebook and discuss what modifications to Facebook's strategy should be adopted to maintain its revenue growth.

## 8.12 Derivatives

## 8.13 Introduction

The derivative is the rate of change of a function $y = f(x)$ when the value of the independent variable $x$ changes from a very small value (i.e. tends to zero). Symbolically we say that :

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}$$

Since the change in the function occurred at a later point $x + h$ with respect to the point we are calculating the derivative, we say that it is calculated *forward*. If the change occurs at a point before the point where we are calculating the derivative, we say that the derivative is calculated *backward*. In this case the calculation procedure is :

$$f'(x) = \lim_{h\to 0} \frac{f(x) - f(x-h)}{h}$$

However there is another way of calculating the derivative, which produces even more accurate results, because it combines forward and backward changes. In this case we say that the derivative is *central* and calculate it according to :

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x-h)}{2h}$$

## 8.14 Numerical Calculation of Derivatives

In this section examples will be presented that demonstrate the great agreement between the numerical approximation of the derivative and its algebraically obtained value. Whenever possible the central derivative approximation will be used.

1. Calculate the first and second numerical derivatives of $x^3 - 20x^2 + 11x + 30$, presenting the graph of the function and its two derivatives in the interval $x = [-1,5; 4,0]$.

The calculation of the function and its two derivatives as well as the associated graph are presented in 8.9.

```
import numpy as np
import pandas as pd
import matplotlib as mpl; mpl.use("TkAgg")
from matplotlib import pyplot as plt
x = np.linspace(-1.5, 4.0, 100);
f = lambda x0:x0**3-20*x0**2+11*x0+30;
df = lambda x0,h=0.0001: (f(x0+h)-f(x0-h))/(2*h);
data = pd.DataFrame({'x':x, 'y':f(x), 'dy':df(x)});
g = data.plot(x='x', y=['y','dy']);
plt.show();
```

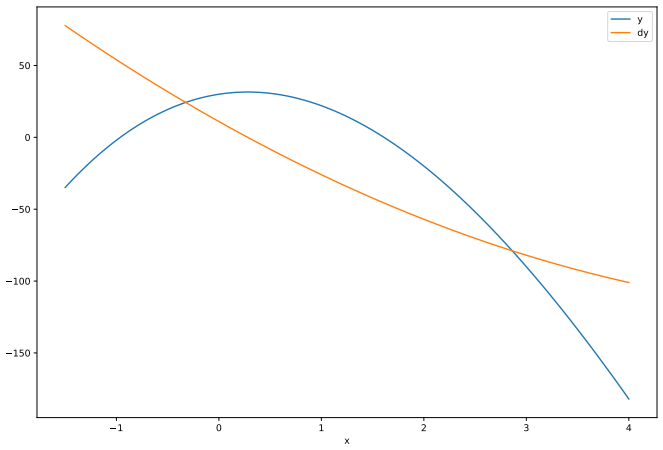



Figura 8.9: Calculation and Graph of the function and derivative of $f(x) = x^3 - 20x^2 + 11x + 30$

2. Calculate the derivative of $f(t) = \sin(\theta)$ by numerical (central) approxi-

mation and compare the result with the analytical value of the derivative $f'(t) = \cos\theta$, presenting both in a graph.

Notice that the approximation is good enough that the graphs of the numerical and algebraic derivatives match.

```
import numpy as np
import matplotlib as mpl; mpl.use("TkAgg")
from matplotlib import pyplot as plt
x = np.linspace(-1.5, 4.0, 100);
f = lambda x0:np.sin(x0);
df = lambda x0,h=0.0001: (f(x0+h)-f(x0-h))/(2*h);

import pandas as pd
data = pd.DataFrame({'x':x, 'y':f(x), 'dy':df(x),
                     'dy_exact':np.cos(x)});
g = data.plot(x='x', y = ['dy', 'dy_exact']);
plt.show();
```

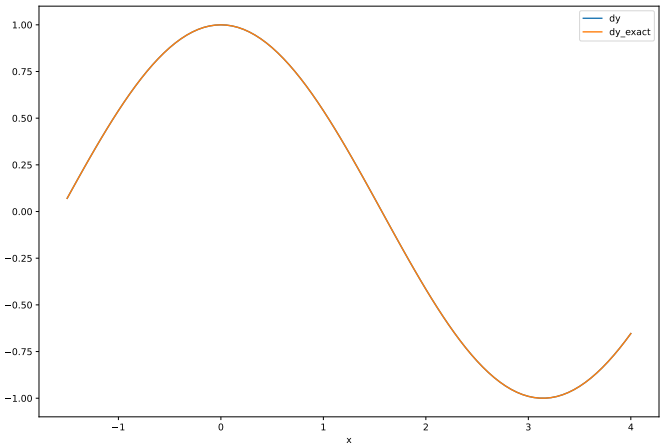



Figura 8.10: Graph the derivative of $f(t) = \sin(\theta)$ by numerical approximation and exact function

## 8.15 Newton-Raphson method

3. Calculate one of the roots of $f(x) = x^2 + 3\,x + 1$ using Newton-Raphson's method. Use the central derivative to calculate $f'(x)$.

The Newton-Raphson method is an iterative method in which the approxima-

tion of the derivative is calculated according to the value of the current root approximation $x_i$, the value of the function at $x_i$, $f(x_i)$ and the derivative of the function, also at $x_i$, $f'(x_i)$, according to the expression: $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$. (see 8.2)

```
import numpy as np
f = lambda x0: x0**2+3*x0+1
df = lambda x0,h=0.0001: (f(x0+h)-f(x0-h))/(2*h)

x = [10]
for i in range(10):
  x.append(x[-1] - f(x[-1])/df(x[-1]))

import pandas as pd
data = pd.DataFrame({'x':x, 'f':map(f,x), \
                     'df':map(df,x)})
```

Tabela 8.2: Table for Calculating the Root of $f(x) = x^2 + 3x + 1$ via Newton-Raphson's method

| x | f | df |
|---|---|---|
| 10.0000000 | 131.0000000 | 23.000000 |
| 4.3043478 | 32.4404537 | 11.608696 |
| 1.5098518 | 7.8092080 | 6.019704 |
| 0.2125773 | 1.6829211 | 3.425155 |
| -0.2787643 | 0.2414166 | 2.442471 |
| -0.3776054 | 0.0097696 | 2.244789 |
| -0.3819575 | 0.0000189 | 2.236085 |
| -0.3819660 | 0.0000000 | 2.236068 |
| -0.3819660 | 0.0000000 | 2.236068 |
| -0.3819660 | 0.0000000 | 2.236068 |
| -0.3819660 | 0.0000000 | 2.236068 |

## 8.16 Graphing the derivative

4. Knowing that

$$x^3 + y^3 = 2xy \tag{8.1}$$

plot the graph of $\frac{dy}{dx}$.

First, we can plot the graph of the implicit function $y = f(x)$ represented by the above equation using `sympy` and the `plot_implicit` function.

```
import sympy as sp
from sympy.plotting import plot_implicit
```

```
x, y = sp.symbols("x y")
f = x**3 + y**3 -2*x*y
g = plot_implicit(f, x_var=(x,-1.2,1.2), \
                     y_var=(y,-1.2,1.2));
```

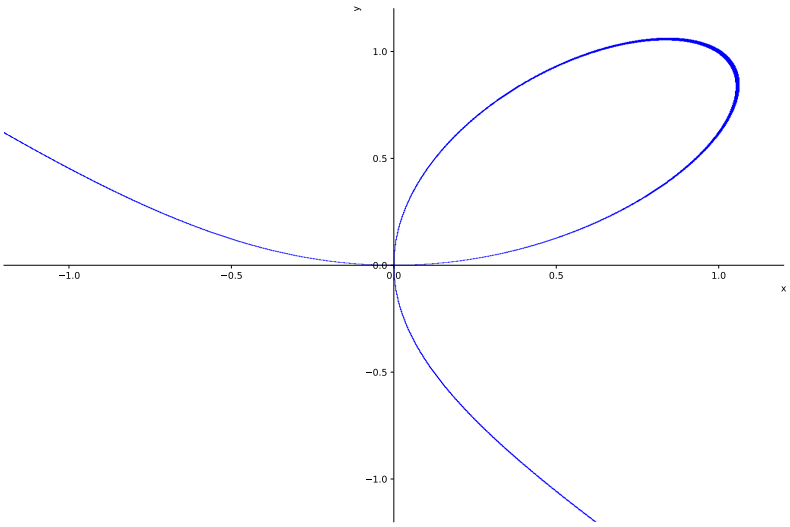



Figura 8.11: Graph the Implicit Function $x^3 + y^3 = 2xy$

The drawing of the graph in 8.11 is achieved through interval mathematics. Instead of calculating a single pair $(x, y)$ and drawing a point at its coordinates, the `plot_implicit` function calculates small rectangles and "paints" the interior of them. What you can do is calculate the midpoint of each sub-interval (either on the $x$ or $y$ axis) and take these values as the "true" pair $(x, y)$. To get the intervals from the values stored in the variable `g` we do:

```
intervs, _ = g[0].get_points()
intervs = np.array(intervs, dtype='object')
```

We convert the variable `intervs` to an `numpy` array which makes it easier to manipulate (above). Next we will extract from each interval its average value (via the `.mid` method) and save the result in a pandas data frame.

```
import numpy as np
data = [(x0.mid, y0.mid) for x0, y0 in intervs]
data = pd.DataFrame(data = data, columns=['x','y'])
data.head()
```

```
          x          y
```

```
0 -1.177734  0.598828
1 -1.175390  0.596484
2 -1.175390  0.598828
3 -1.173046  0.596484
4 -1.173046  0.598828
```

This whole procedure can be compressed into a single Python function, which takes an implicit function and the calculation ranges for $x$ and $y$ and returns the corresponding data frame.

```
import sympy as sp
from sympy.plotting import plot_implicit
x, y = sp.symbols("x y")
f = x**3 + y**3 -2*x*y
interv_x = (x,-1.2,1.2)
interv_y = (y,-1.2,1.2)

import pandas as pd
def gera_xy(Eq0, intervX, intervY):
  g = sp.plot_implicit(Eq0, x_var = intervX, \
                       y_var = intervY, \
                       show = False);
  intervs, _ = g[0].get_points()
  intervs = np.array(intervs, dtype='object')
  ptos = [(x0.mid, y0.mid) for x0, y0 in intervs]
  data = pd.DataFrame(data = ptos, \
                       columns = ['x','y'])
  return(data)

data = gera_xy(f, interv_x, interv_y); data.head()
```

```
          x         y
0 -1.177734  0.598826
1 -1.175391  0.596482
2 -1.175391  0.598826
3 -1.173047  0.596482
4 -1.173047  0.598826
```

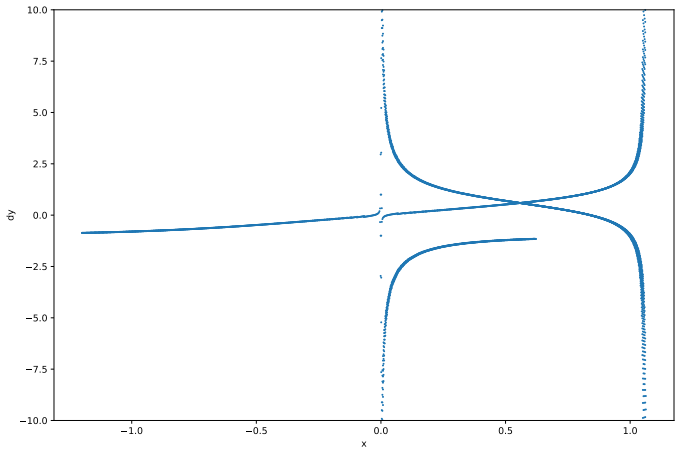
The derivative of the expression in 8.1 can be obtained using the `idiff` function of `sympy`.

```
import sympy as sp
df = sp.idiff(f, y, x, 1); df
```

```
(3*x**2 - 2*y)/(2*x - 3*y**2)
```

With this expression and the values available in the data frame pandas, we can calculate the values of the derivative at each pair $(x, y)$ and then plot their graph. For performance reasons, we will create a numeric function with the same expression as `dEq` to perform the value substitutions

```
df = lambda x0,y0:(3*x0**2 - 2*y0)/(2*x0 - 3*y0**2)
x, y = data['x'].values, data['y'].values
yL = [df(x0,y0) for x0, y0 in zip(x, y)]
data['dy'] = np.array(yL, dtype='float')
g = data.plot(x='x', y='dy', kind='scatter', s=1);
plt.ylim(-10,10);
plt.show();
```




1. Show that the slope of the logistic equation: $y = \frac{1}{1+e^{-ax}}$ at its midpoint is equal to $a/4$.

Let's solve this problem numerically. In the following listing are Python functions for calculating the values of the logistic function and its derivative at any given point $x$ and for values to be chosen from the parameter $a$. As can be seen, whenever the derivative is calculated for $x = 0$, its value coincides within reasonable precision with the value $a/4$, for $a$ varying in the range $[-2, 2]$.

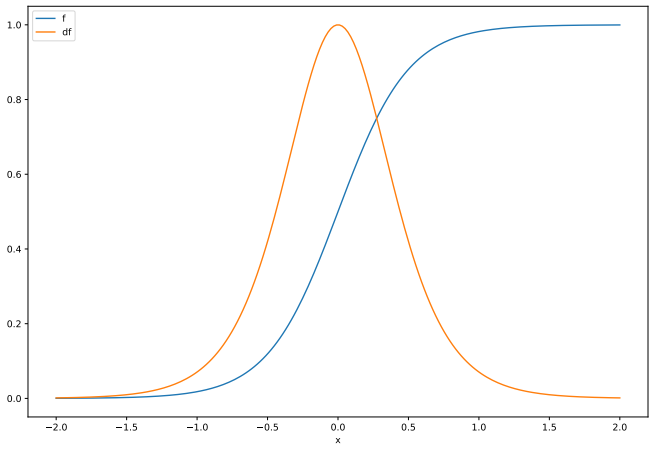
```
import numpy as np
a = np.linspace(-2,2,100);
f = lambda x0,a: 1/(1+np.exp(-a*x0));
df = lambda x0,a,h=0.0001: (f(x0+h,a)-f(x0-h,a))/(2*h)

import pandas as pd
data = pd.DataFrame({'df(0,a)':df(0,a), 'a/4':a/4})
data.head()
```

```
    df(0,a)       a/4
0 -0.500000 -0.500000
1 -0.489899 -0.489899
2 -0.479798 -0.479798
3 -0.469697 -0.469697
4 -0.459596 -0.459596
```

The graph of the logistic function and its derivative, for $a = 4$ are shown below:

```
import numpy as np
x = np.linspace(-2,2,200)
f = lambda x0: 1/(1+np.exp(-4*x0))
df = lambda x0,h=0.0001: (f(x0+h)-f(x0-h))/(2*h)
data = pd.DataFrame({'x':x, 'f':f(x), 'df':df(x)})
g = data.plot(x='x', y=['f', 'df']);
plt.show();
```




## 8.17 Partial Derivatives

If the function has more than one independent variable, for example $z = f(x, y)$ we will have independent rates of change, one for each variable. Symbolically we say $z_x = \frac{deltaz}{deltax}$ and $z_y = \frac{deltaz}{deltay}$. The calculation is very similar to that of a derivative in a single variable. When we are deriving with respect to $x$ for example, all other independent variables $y$ for example will be considered constants.

We can have second derivatives in the same variable $z_{xx}$ or $z_{yy}$ for example or in

alternating variables $z_{xy}$. An interesting result is that, for continuous functions, $z_{xy} = z_{yx}$, i.e. the order of derivation does not influence the final result of the derivative calculation.

## 8.18 Exercises

### 8.18.1 Calculation of Derivatives

For the following exercises, calculate the derivatives of the following functions by a) the definition of derivative (using the formula $f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x)-f(x)}{\Delta x}$, b) the algebraic method (using the basic formulas $f(x) = ax^n \rightarrow f'(x) = anx^{n-1}$ and $f(x) = be^{ax} \rightarrow f'(x) = abx^{ax}$) and c) the `limit` function of `sympy`.

1. $3x$
2. $x^x$
3. $x^4$
4. $\frac{1}{x}$
5. $\sqrt{x}$
6. $\ln(3x)$
7. $e^{\sin(x)}$
8. $\tan(x)$
9. $\frac{1}{\sin(x)}$
10. $\sqrt{x}$
11. $e^{ax}$
12. $e^x.x^2$
13. $e^x.ln(x)$
14. $e^{3x}.ln(x)$
15. $e^x.ln(2x)$
16. $\ln(1/x)$
17. $\ln(3/x^2)$
18. $\ln(\frac{1}{x^3})$
19. $\ln(\frac{4}{x^2})$
20. $\frac{\sqrt{x}}{ln(x)}$
21. $\frac{\sqrt{3x}}{ln(x)}$

22. $\frac{\sqrt{x}}{ln(2x)}$
23. $\frac{1}{(x^2+2)^3}$
24. $\frac{e^x}{x+1}$
25. $\frac{e^{2x}}{x+3}$
26. $\frac{3}{(x^2-1)^8}$
27. $(3x^2+x-2)^2$
28. $\frac{1}{x^{1/7}}$
29. $x^4+\frac{x^3}{3}+\frac{x^2}{4}+2x-1$
30. $\sqrt{x}+\sqrt[3]{x^2}$
31. $(t^2+5t-14)^4$
32. $(t^4+2t)^2.(1-t^2)$
33. $ln(\frac{x}{3+5x})+e^{(2x+1)^2}$
34. $\ln(2x^2+x).(4x+1)^2$
35. $\tan(t+4)$
36. $\frac{(x^2-x)^3}{(2x-1)^4}$
37. $\sin(\frac{u^2-1}{u+1})$
38. $-x^4+\frac{x^3}{3}-\frac{x^2}{4}+1$
39. $\sqrt{4x+1}-\sqrt[5]{x^3}$
40. $(t^4+5t+1)^3$
41. $(t^3+2t)(2+3t^2)$
42. $\frac{sin(x+1)}{\ln(2x-1)}$
43. $\ln(2x^2-3x)-e^{(4x-3)^2}$
44. $\ln(\frac{x^2-x}{2x+5})$
45. $\frac{(x^2-x)^7}{(2x-1)^4}$

### 8.18.2 Approximate Value of Functions and Derivatives

1. Calculate the derivative of the function $g(x)=(\frac{1}{x}+3).e^{2x}$ and determine the value of the derivative at t = 1.7.

2. Calculate the approximate value of $\frac{d\left[e^{2x}\right]}{dx}$ with h=0.1, at point x=2 or the exact value by algebraic derivation

### 8.18.3 Calculation of Roots of Equations - Newton-Raphson Method

1. Starting from $x = 0.1$, calculate two other approximations for one of the roots of the function $f(x) = e^{-2x}cos(2pix)$ using Newton-Raphson's method.
2. Starting from $x = 0$, use the Newton-Raphson method to calculate an approximation for one of the roots of $(x) = x^3 - 2x^2 + x - 1$ with at least four digits of precision.
3. Find the solution of the equation $x^3 + 2x^2 + x + 1 = 0$ using Newton-Raphson's method. The solution found should have $\Delta x\% < 0.1\%$.
4. Solve the problems below numerically using Newton-Raphson's method. The solutions should be obtained with a percentage error $\epsilon$ less than 0.01%.
   (a) Calculate the value of $\sqrt{5}$
   (b) Calculate the roots of $x^2 + 4, x + 3 + sin(x) - x, cos(x)$

### 8.18.4 Applications in Geometry

1. Find the angular coefficient of the line tangent to the curve $y = x^3$ at the point $(4, 64)$ and write the equation of this line.
2. At what point on the curve $y^2 = 2x^3$ is the tangent perpendicular to the line $4x - 3y + 2 = 0$?
3. Calculate the distance between the points for which the curves $y = 4x^2 + 2x - 8$ and $y = x^3 - x + 10$ are parallel to each other.
4. Calculate the area of the triangle formed by the line tangent to the circle $x^2 + y^{=}1$ at the point $x = 0.5$ and the coordinate axes.
5. Air is being pumped continuously into a spherical balloon at a rate of $5cm^3/min$. Determine the rate at which the radius of the balloon grows when its diameter is 20 cm.

### 8.18.5 Implicit Derivation

1. Suppose you have the following equation: $x^2 + y^2 = 1$. Without isolating the variable $y$ what is the expression of $\frac{dy}{dx}$?
2. Calculate the derivative of y with respect to x in the expression $y^3 = x^2 - 1$ implicitly.
3. Calculate $\frac{dy}{dx}$ knowing that $x^3 + y^3 = a^3$

4. A particle is moving on a curve implicitly defined by: $xy^4 - yx^4 = x - y^2$. When the particle passes through the point $(1, 1)$, its $x$ coordinate is changing $1/4$ unit per second. At what rate is the $y$ coordinate changing at this point?

5. Calculate using implicit derivatives the derivative $\frac{dy}{dx}$ of $3y^2 + 2x^2 = R^2$ where $R$ is a constant

### 8.18.6 Level Curves and Partial Derivatives

1. Draw the level curves with z=0, 1 and 2 for the functions :
   (a) $z = f(x, y) = 2x - y$
   (b) $z = f(x, y) = -x^2 + 2y$
2. Calculate the expressions $\frac{sigmax^3y^2}{sigmax}$ and $\frac{sigmax^3y^2}{sigmay}$
3. Determine which of the following functions $f(x, y)$ satisfy the equation $f_{xx} + f_{yy} = 0$
   (a) $f(x, y) = x^2 - y^2$
   (b) $f(x, y) = xy$
   (c) $f(x, y) = x.e^y - y.e^x$
   (d) $f(x, y) = ((x - 1)^2 + (y + 3)^2)^{(-\frac{1}{2})}$

### 8.18.7 Applications in Economics

1. The utility of a consumer to purchase x units of one product and y units of a second product is given by the utility function $U(x, y) = 6x^2y^3$. Suppose that the consumer purchases 2 units of the first product and 2 units of the second product each month. Calculate $U(2, 2)$ and plot the level curve for this value.
2. Consider the total cost function $C(q) = q^2 - 4q - 2$. Based on this information ask:
   (a) Calculate the marginal cost using the definition of derivative. Calculate $C\_mg(10)$ and interpret its meaning.
   (b) Determine the equation of the tangent line to the graph of C(q) at points a = 2 and b = 4.
   (c) Find the intervals of growth and decrease of the total cost.
   (d) Show that the total cost function has upward concavity throughout its domain.
3. Let $L(q) = q^3 - q^2 - q + 1$ be the total profit function for $q$ units of popcorn sold by the street vendor in front of the school on Wednesdays, according

to a market survey conducted in the first half of 2008. The popcorn vendor counts on the talent and mathematical knowledge of the students of the Administration course of this school to know:

(a) For what quantities is the total profit zero?

(b) Determine the popcorn maker's marginal profit function. Calculate and interpret $L\_mg(5)$.

(c) In which interval(s) is the total profit function increasing? What quantity minimizes the popcorn maker's profit?

(d) On which interval(s) does the total profit function have upward concavity?

(e) Using an appropriate boundary, show what happens to the total profit when the number of popcorn sold grows infinitely.

(f) Based on the findings in the previous items, sketch the graph of total profit as a function of the number of popcorn sold.

4. Consider a country that has annual health care spending (in billions of dollars) given by the function $f(t) = 27e^{0.106t}$, where $t = 0$ for the year 1960. Calculate:

   (a) What is the amount of spending in 1972?

   (b) What is the growth rate of spending in 1976?

   (c) When did the country reach the spending amount of 120 billion?

   (d) When the growth rate was 20 billion per year?

5. On each day the output of a coal mine, after t hours of operation, is approximately $p(t) = 40t + t^2 - 1/15t^3$ tons, for $0 <= t <= 12$. What is the production rate (in tons of coal per hour) at $t = 5$ hours?

6. What is the meaning of a) marginal cost, b) marginal revenue, and c) marginal profit?

7. A factory produces $P(x, y) = 60x^{0.75}y^{0.25}$ units of a given product per week, where x is the amount of labor (in man-hours) and y is the amount of machines used. On this basis ask: a) determine the output for 81 units of labor and 16 machines, b) if each unit of labor is paid for by its marginal productivity, determine the expression for the value of labor (i.e. for wages), and c) determine the value of wages when 81 units of labor and 16 machines are used.

8. The daily profit (in dollars) of a retailer selling two brands of similar products is given by $P(x, y) = (x-40)(55-4x+5y)+(y-45)(70+5x-7y)$, where $x$ is the unit price of the first product and $y$ is the unit price of the second product. At the moment the first brand sells for \$0.70 a unit and the second brand sells for \$0.73 a unit. Based on these data ask:

(a) Find the marginal profit functions Px and Py.

(b) Determine the value of both marginal profits at current price levels

(c) Use the concept of partial derivatives to estimate the change in daily profit that will result if the retailer decides to increase the price of the first product by \$0.01 and the price of the second product by \$0.02 simultaneously.

9. Suppose a demand function (price as a function of quantity) for a given product is $P(q) = \frac{45}{ln(q)}$, determine the marginal revenue function and the marginal revenue for $q = 20$. (Please note: you are asking for MARGINAL billing, not just simple billing).

10. Let x be the quantity of cars manufactured by an automaker and $f(x)$ be the cost in dollars to manufacture them What is the meaning of $f(1000) = 15,000,000$ and $f'(1000) = 10,000$. What is the marginal cost when the quantity manufactured is 1000 cars.

11. The cost of manufacturing x units of a given product is given by the expression $C(x) = 0.2x^3 - 0.3x^2 + 200x + 200$. Knowing that the revenue is given by the expression $R(x) = -4x^2 + 500x$, obtain:

    (a) An approximation for the cost of manufacturing the 21st unit, from the cost of manufacturing 20 units and the marginal cost value at x=20.

    (b) An approximation for the revenue from the 15th unit, from the total revenue of 14 units, plus the marginal revenue value at x=14

    (c) The value of x that will make the marginal cost equal to the marginal revenue.

    (d) An interpretation of the value of x obtained in the previous item.

12. An estimate shows that the monthly production of e-books of the work "Vidas Secas" by Graciliano Ramos is given by the function: $P(x, y, z) = 900x + 360y + 2x^2y + text10yz + 12x\sqrt{z}$ (in units), where $x$ is the number of editors, $y$ the number of skilled workers and $z$ the number of unskilled workers. At the moment, the available labor force consists of 2 editors, 18 skilled workers, and 36 unskilled workers. The publisher has decided to hire a single additional employee who can be an editor, a skilled worker, or an unskilled worker. Use the concept of partial derivatives to find the best solution for the publisher to increase production.

13. It is estimated that the weekly production of a certain factory is given by the function $Q(x, y) = 1200x + 600y + 2x^2y - x^3 - 0.5y^2$ units, where x is the number of skilled workers and y is the number of unskilled workers. At the moment, the available labor force consists of 20 skilled laborers and 60 unskilled laborers.

(a) Use the concept of partial derivatives to estimate the change in output if 1 unskilled worker is hired and the number of skilled workers remains constant.

(b) Calculate the exact difference: $Q(20.61) - Q(20.60)$

(c) What is the percentage difference between the values obtained in a) and b) (Note: use as reference value the exact result, that is, from letter b).

14. A publisher has $60,000.00$ to invest in the production and advertising of Graciliano Ramos's e-book. It is estimated that if x thousand Reais are spent on production and y thousand Reais are spent on advertising, approximately $f(x, y) = 20x^{1,5}y$ e-book units will be sold. Use the Lagrange Multiplier to analyze how much the publisher must invest in production and advertising to get as many e-books sold as possible.

15. Currently 1,800 people per day use a certain train line to go to work, paying 4 per ticket. The number of people q willing to take the train when the fare is $p$ is $q = 600(5 - \sqrt{p})$. The company operating the transportation system wishes to increase its revenue. a) Is demand elastic or inelastic when $p = 4$? b) Should the price be increased or not? Why should it be increased?

16. Suppose the demand for a certain metal is $q = 100 - 2p$, where $p$ is the price per kilogram and $q$ is the quantity demanded (in thousands of tons). (Hint: remember that elasticity is the rate of change of a function with respect to itself, plus a negative sign.) Answer:

(a) What quantity is demanded at R$30 a pound?

(b) What is the function $E(p)$ (Price Elasticity)

(c) What is the elasticity at $p = 30$. Interpret the result.

(d) What is the elasticity at $p = 20$. Interpret the result.

17. Consider two goods $X$ and $Y$ with prices $p$ and $q$, respectively. Let $f(p, q) = a/p^2q$ be the demand for good $X$ and $g(p, q) = b/(pq)$ be the demand for good $Y$. Find the conditions on $a$ and $b$ for goods $X$ and $Y$ to be substitutes and also the conditions for them to be complements.

18. Consider the demand functions for two goods W and Z, given by the equations $q_W = \frac{7p_Z}{1+8p_W}$ and $q_Z = \frac{2\sqrt{p_W^5}}{7+p_Z^3}$, where $p_W$ and $p_Z$ are the unit prices of $W$ and $Z$.

(a) Calculate the partial marginal demands $\frac{\partial q_W}{\partial p_W}, \frac{\partial q_W}{\partial p_Z}, \frac{\partial q_Z}{\partial p_W}, \frac{\partial q_Z}{\partial p_Z}$.

(b) Are products $W$ and $Z$ substitutes or complements? Justify.

(c) If there is a third product U with demand given by the equation $q_U = 117 - 7p_U + 2p_W$ where $p_U$ is the unit price of U, which of the

options below reduces the demand for U the most:

i. Decrease the price of W by one unit

ii. Increase the price of U by one unit.

19. Given the demand functions of two products, determine whether they are substitutes or complements

    (a) $D_1 = 500 - 6p_1 + 5p_2$, $D_2 = 200 + 2p_1 - 5p_2$

    (b) $D_1 = 1000 - 0.02p_1^2 - 0.05p_2^2$, $D_2 = 800 - 0.001p_1^2 - p_1p_2$

    (c) $D_1 = 200p_1^{-1/2}p_2^{-1/2}$, $D_2 = 300p_1^{-1/2}p_2^{-3/2}$

    (d) $D_1 = 2000 + \frac{100}{p_1+2} - 25p_2$, $D_2 = 1500 - \frac{p_2}{p_1+7}$

    (e) $D_1 = \frac{7p_2}{1+p_1^2}$, $D_2 = \frac{p_1}{1+p_2^2}$

    (f) $q_W = \frac{2p_Z}{3+5p_W}$ and $q_Z = \frac{sqrt[3]P_w}{2+p_Z^2}$

20. A news story is broadcast by the mass media to a potential audience of 50,000 people. After $t$ days $f(t) = 50,000(1 - e^{-0.3t})$ people have learned of the news.

    (a) How many people will hear about the news after 10 days?

    (b) At what rate is the news initially spreading?

    (c) How soon will 22,500 people hear about it?

    (d) Approximately when will the news be spreading at a rate of 2,500 people per day?

    (e) At what rate will the news be spreading, when half the audience already knows about it?

21. A story is spread by word of mouth to a potential audience of 10,000 people. After r days, $f(t) = \frac{10,000}{1+50e^{-0.4t}}$ people knew about it. At what rate will the news be spreading spreading when half of the potential audience has already heard the news?

22. Item When the jury found the mayor of a certain town guilty of receiving illegal commissions, newspapers, radio and television immediately began broadcasting the news. Within an hour, a quarter of the citizens knew about the decision. Estimate when three quarters of the city will know about the news, knowing that $P'(t) = k.[1 - P(t)]$ , where $P(t)$ is the proportion of the city's inhabitants who know about the news at time $t$ and that at the initial time no one knew about the news. Hint: This problem can be expressed as follows: $P(0) = 0$; $P(1) = 0.25$; $P'(t) = k.[1 - -P(t)]$. Use the formula $P(t + h) = P(t) + P'(t).h$ and find $t$ for $P(t) = 0.75$. Use $h = 0.1$.

23. The proportion of people who know about a news story after its release is given by $P(t) = 1 - e^{-kt}$ . In a given city, three hours after the release of a news item, 25

24. It is known that the speed of diffusion of information is proportional to the percentage of people in a group who have not yet learned about it. In one group, it took 4 hours for 30

### 8.18.8 Series Exercises

1. Since $S = 1 + 1/2 + 1/3 + 1/4 + ... + 1/N$, create a Python function that takes $n$ and returns S with $n$ terms. If $n = 100$ the result should be 5.1972785077386305.

2. Being $S = 1/1+2/3+3/5+4/7+5/9+...+n/m$, create a Python function that takes $n$ and $m$ and returns the value of S. If the first 100 terms of this series are calculated, the value of $S$ must be 52.14465865683989.

3. Create a Python function that takes a number $n$ and then returns the $n$ first terms of the Fibonacci series. The Fibonacci series starts with 0 and 1 and the next term is always the sum of the previous two terms.

4. Create a Python function that takes a number $n$ and then prints the ratio between two terms of the Fibonacci series of type $\frac{n(i+1)}{n(i)}$.

5. Create a Python function that takes a number $n$ and returns the $n$ first prime numbers

6. Create a Python function that takes a number $n$ and returns the $n$ first prime numbers that belong to the Fibonacci series.

7. Create a Python function that takes a number $n$ and returns the value of $e$ calculated using the associated power series. NOTE: $e^x \approx x^0/0! + x^1/1! + x^2/2! + x^3/3! + x^4/4! + ... + x^n/n!$.

8. Create a Python function that takes a number $n$ and returns the value of sen(1) where sen(1) is given by the associated power series. NOTE: $sen(x) \approx x - x^3/3! + x^5/5! - x^7/7! + ... + \frac{(-1)^n x^{2n+1}}{(2n+1)!}$

9. Create a Python function that takes a number $n$ and returns the value decos(1) where cos(1) is given by the associated power series. NOTE: $cos(x) \approx 1 - x^2/2! + x^4/4! - x^6/6! + ... + \frac{(-1)^n x^{2n}}{(2n)!}$

10. python can work with imaginary numbers. An imaginary number $j$ is defined as $j = \sqrt{-1}$. To perform operations involving the creation of complex numbers from the square root function import the cmath package. Then create a program that will calculate the value of $e^j$ using the associated power series with 30 terms. Compare the result obtained by the power series with the values obtained for $cos(1)$ and $sen(1)$. What can you conclude from these results?

11. Create a Python function that takes a number $n$ and returns the sum of the first $n$ terms of an arithmetic progression of reason $x$.
12. Create a Python function that takes a number $n$ and returns the sum of the first $n$ terms of a geometric progression of reason $x$.
13. Create a Python function that takes a number $n$ and returns the sum of the terms of a geometric progression, which will start at 1 and have as its ratio x where $x < 1$.
14. Create a Python function that takes a number $n$ and returns the following sum: $S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + ... + \frac{1}{n}$.

### 8.18.9 Infinite series

1. Present the limit of the series $S = R + R^2 + R^3 + R^4 + R^5 + ...$ for $R = \frac{1}{2}$
2. Consider the series $S = LR + LR^2 + LR^3 + LR^4 + LR^5 + ...$ where $L$ is the profit received per period from a given stock, $S$ is the present value of the infinite series of receipts, and $R = \frac{1}{1+i}$, where $i$ is the discount (or interest) rate. Answer the following questions:
   (a) What would be the condition for $P$ to be a fair price for the stock (in terms of $S$)
   (b) Let $P$ be the price paid for the stock. What is the expected effect on $P$ of an increase of $i$? Demonstrate your statement in mathematical terms.
3. Suppose you buy a stock for P=R$100.00 and that this stock will give a profit of L=R$10.00 every year indefinitely. Under these conditions the present value of the first 10.00 will be $\frac{10}{1+i}$, of the second 10.00 will be $\frac{10}{1+i}^2$ and so on. Find, using the sum of the terms of a geometric series the relationship between $P$, $L$, and $i$, and calculate the value of $i$ for the above case.
4. Consider a perpetuity that promises to pay 100 at the beginning of each month. Suppose the interest rate is 12% compounded monthly. Then the present value of US$100 in $k$ months is $100(1.012)^{-k}$. Find the sum of the infinite series.
5. Calculate the total new spending created by cutting income tax revenues by 10 billion, when the marginal propensity to consume is 95%. If the marginal propensity to consume is 96%, by how much do the new expenditures increase?
6. Calculate the effect of a 20 billion cut in the total income tax collected, when the population's marginal propensity to consume is 98%? What is the multiplier in this case?

7. Suppose that the American Central Bank buys 100 million of Federal Government bonds from private individuals. By doing so it introduces 100 million of new money and starts a chain reaction because of the monetary injection system. When the US100 million is deposited in bank accounts, the banks keep 15 percent in reserves and can lend the remaining 85 percent by creating more new money, $0.85 * 100$ million and so on, lending $0.85 * 100$ million. This process repeats itself for an infinite number of times. Calculate the total amount of new money that can be created, in theory, by this process in addition to the original 100 million.

### 8.18.10 Power Series

1. Compute via power series expansion of f(x), with at least 5 nonzero terms, each of the following functions f(x):
   (a) $e^x$
   (b) $e^{2x}$
   (c) $e^{-3x}$
   (d) $e^{3x}$
   (e) $e^{-x}$
   (f) $e^{2x}$
   (g) $5e^{2x}$
   (h) $e^{-2x}$
   (i) $e^{-x/2}$
   (j) $\sin(x)$
   (k) $\cos(x)$
   (l) $\cos(2x)$
   (m) $\cos(-x)$
   (n) $\cos(3x)$
   (o) $\cos(\frac{\pi}{2} - 3x)$
   (p) $\cos(\pi - 5x)$
   (q) $\sqrt{4x+1}$
   (r) $\frac{1}{x+2}$
   (s) $xe^{3x}$
   (t) $\sqrt{1-x}$
   (u) $e^{2x}\sin(3x)$

### 8.18.11 Advanced

The following exercises are advanced level exercises. They can be skipped in a beginning calculus course:

1. Demonstrate Euler's formula, $e^{ix} = \cos x + isenx$
2. Calculate $i^i$
3. It is known that $e^{i\pi} = -1$ e que $e^{i3\pi} = -1$. Could this not lead us to conclude that $1 = 3$? Where is the error in this statement?
4. What is $\ln(-1)$?
5. Calculate the $\cos(ix)$. (Hint: power series expansion).

## 8.19 Optimization

## 8.20 Numerical Optimization

In optimization problems we seek to find the value of one or more independent variables ($x$) that will cause the dependent variable $y$ to reach the highest or lowest possible value. To do this we use the relations between the derivatives and the original function to determine this point as precisely as possible.

The aforementioned relations are the fact that at its maximum point $x_{max}$, a function $f(x_{max})$ will have null first derivative (i.e. equal to 0) $f'(x_{max}) = 0$ and negative second derivative $f''(x_{max}) < 0$. At a point of minimum a function will also have $f'(x_{min}) = 0$, but $f''(x_{min}) > 0$. If the function at this point $x_{max}$ has both the first and second derivatives null, we say that it has at $x_{max}$ an inflection point. With this knowledge we can start a wide range of exercises, as you will see in the following sections.

1. Calculate the maximum profit that could be made in one week by a manufacturer who estimates that the profit on producing x units of a commodity is given by $L(x) = -x^2 + 40x$ dollars/week. Use the central derivative expression for the numerical approximation of the derivative. Remember that at the point of maximum of the function its derivative will equal zero.

Since we want to calculate the root of $L'(x)$ using Newton-Raphson's method, we will need the values of the second derivative $L''(x)$.

We will use the following numerical approximations for the values of $f(x)$, $f'(x)$, $f''(x)$ 1. For the calculation of the following numerical approximation of the root : $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$ 2. For the numerical calculation of the derivative : $f'(x_i) = \frac{f(x_i+h)-f(x_i-h)}{2\,h}$ 3. For the numerical calculation of the $2^a$ derivative : $f''(x_i) = \frac{f(x_i+h)+f(x_i-h)-2\,f(x_i)}{h^2}$

The calculation sequence can be seen in the table 8.3. As can be seen the result is $x = 20$ and $L(x) = 400$.

```
import numpy as np
import pandas as pd
f = lambda x: -x**2+40*x
df = lambda x,h=0.001: (f(x+h)-f(x-h))/(2*h)
df2 = lambda x,h=0.001: (df(x+h,h)-df(x-h,h))/(2*h)

x = [1]
for i in range(4):
  x.append(x[-1]-df(x[-1])/df2(x[-1]))

data = pd.DataFrame({'x':x, 'f':map(f,x), \
                     'df':map(df,x), 'df2':map(df2,x)})
```

Tabela 8.3: Calculating the point at which $L'(x) = 0$

| x | f | df | df2 |
|---|---|---|---|
| 1 | 39 | 38 | -2 |
| 20 | 400 | 0 | -2 |
| 20 | 400 | 0 | -2 |
| 20 | 400 | 0 | -2 |
| 20 | 400 | 0 | -2 |

There are situations where the function has more than one point of maximum and minimum in its domain.

2. Compute the critical points of the function $f(x) = -x^3 + 3.x^2 + 9.x - 1$.

Let's create a function that runs the Newton-Raphson method in search of the root of the derivative of $f(x)$. By default, the function will iterate 100 times with $h = 0.0001$ and will return the value of $x$ that makes the derivative of $f(x)$ null and the values of $f(x)$, $f'(x)$ and $f''(x)$ at this point. Note the following:

```
import numpy as np
f = lambda x0: -(x0**3) + 3 * (x0**2) + 9*x0 -1
df = lambda x0,h=0.0001: (f(x0+h)-f(x0-h))/(2*h)
df2 = lambda x0,h=0.0001: (df(x0+h)-df(x0-h))/(2*h)
def newton(x0,n=100,h=0.0001):
  for i in range(n):
    x0 = x0 - df(x0)/df2(x0)
  return(x0, f(x0), df(x0), df2(x0))

np.round(newton(-20),5)
```

```
array([-1., -6.,  0., 12.])
```

The program before being executed had as its starting point $x = -20$. After running it we got the results $x = -1$, $f(x) = -6$, $f'(x) = 0$ and $f''(x) = 12 > 0$. Then for $x = -1$ we have $f(-1) = -6$, $f'(-1) = 0$, $f''(-1) = 12 > 0$. Since the first derivative is zero and the second positive at $x = -1$ we can say that $f(-1)$ will be a local minimum of $f(x)$. Changing the initial value to $x = 20$ and restarting the iterative process we arrive at the result shown below. We can then say that $f(3) = 26$ is a local maximum of $f(x)$, since $f'(3) = 0$ and $f''(3) = -12$.

```
import numpy as np
f = lambda x0: -(x0**3) + 3 * (x0**2) + 9*x0 -1
df = lambda x0,h=0.0001: (f(x0+h)-f(x0-h))/(2*h)
df2 = lambda x0,h=0.0001: (df(x0+h)-df(x0-h))/(2*h)
def newton(x0,n=100,h=0.0001):
  for i in range(n):
    x0 = x0 - df(x0)/df2(x0)
  return(x0, f(x0), df(x0), df2(x0))

np.round(newton(20),5)
```

```
array([  3.,  26.,   0., -12.])
```

## 8.21 Algebraic Optimization

The optimization can also be calculated algebraically using `sympy`.

3. For example, determine the minimum of $f(x, y) = x + y$ , when $x.y = 100$ and $x > 0$ and $y > 0$.

Without specifying the constraints of $x$ and $y$ greater than 0 we get the result below. We choose the solution with only positive real values for $x$ and $y$.

```
import sympy as sp
x = sp.symbols("x", real=True)
y = 100/x
f = x+y
sp.solve(f.diff(x),x)
```

```
[-10, 10]
```

4. Suppose that 20,000 people will attend a soccer game if the ticket price is \$5.00 and that each increase of \$1.00 will cause a decrease in the number of fans of 500 people. How much should the price vary to maximize revenue?

To solve, we must first find the demand function, which will relate price to quantity. We can do this by assuming that $p$ and $x$ are on a line, which must pass through the points $(p1 = 5; x1 = 20,000)$ and $(p2 = 6; x2 = 19,500)$. This will form our constraint. Making $R = p.x$ gives us the objective function. By passing this data into the format used in the previous example we easily solve the problem.

```
import sympy as sp
p = sp.symbols("p", real=True)
x = (19500 - 20000)/(6 - 5)*(p-5) + 20000
R = p*x
sp.solve(R.diff(p),p)
```

```
[22.5000000000000]
```

5. The demand equation of a monopolist is $p = 1050 - 0.03x$ and the cost function is $C(x) = 150x + 750,000$, where $x$ is the number of units produced. Based on these data ask:

   (a) The value of $x$ that maximizes profit and the corresponding price

   (b) Suppose the government reduces taxes by $1,000,000 so as to increase economic activity (causing an increase in the monopolist's net profit), what is the result in terms of $x$ and $p$.

```
import sympy as sp
x, T = sp.symbols("x T")

p = 1050 - 0.03 * x
c = 150 * x + 750000
L = p * x - c + T

sp.solve(L.diff(x), x)
```

```
[15000.0000000000]
```

The taxes are calculated using the variable $T$. Notice that the value of x that maximizes the monopolist's profit $L$ is independent of $T$ and equals $x = 15000$. Thus, a tax reduction that only affects the monopolist's profit, without directly affecting the quantity produced, will at least have no effect on economic activity. One more example: The radius of a sphere is increasing. For what value of $r$, the rate of increase of volume $dV/dt$ is equal to $64.\pi$ times the rate of increase of $r$?

Before we start solving the problem in spreadsheet we need to remember that $V = \frac{4}{3}\pi$,r^{3}$. Our problem now is to calculate the $r$ for which $V' = 64\,\pi\,r'$. However $\frac{dV}{dt} = \frac{dV}{dr}, \frac{dr}{dt} = \frac{dV}{dr}, r'$. Since we want $V' = 64\,\pi\,r'$, we actually want to

calculate $\frac{dV}{dr}drdt = 64\,\pi\,\frac{dr}{dt}$. By simplifying the $\frac{dr}{dt}$ on both sides of the equation we conclude that our final problem is to calculate $\frac{dV}{dr} = 64\,\pi$. In possession of these relationships we set up the Python command sequence shown below, the answer then becoming $r = 4$ (four).

```
import sympy as sp
V, r = sp.symbols("V r")
V = 4/3*sp.pi*r**3
sp.solve(V.diff(r)-64*sp.pi, r)
```

```
[-4.00000000000000, 4.00000000000000]
```

6. Plot the graph of : $y = e^{-x^2}$ and its derivative on the interval $x = [-3; 3]$ and find the point of maximum of the derivative.

To find the maximum of $y'(x)$ we must calculate the point $x$ at which $y''(x) = 0$. But to calculate one of the roots of $y''(x)$ by a Newton-Raphson type numerical approximation method, we must calculate $y'''(x)$ because the method requires iterations of the type :

$$x_{i+1} = x_i - \frac{f''(x_i)}{f'''(x_i)} \tag{8.2}$$

At 8.12 the graph of $y(x)$ and $y'(x)$ is shown, and at 8.4 the calculation of the maximum of $f'(x)$ is done (which implies the calculation of the root of $f''(x)$, which in turn requires the calculation of $f'''(x)$).

```
import numpy as np
import matplotlib as mpl; mpl.use("TkAgg");
from matplotlib import pyplot as plt

x = np.linspace(-3,3,100);
f = lambda x0:np.exp(-x0**2);
df = lambda x0,h=0.0001 : (f(x0+h)-f(x0-h))/(2*h);

import pandas as pd
data = pd.DataFrame({'x':x, 'y':f(x), 'dy':df(x)})
g = data.plot(x='x', y=['y', 'dy'])
plt.show();
```

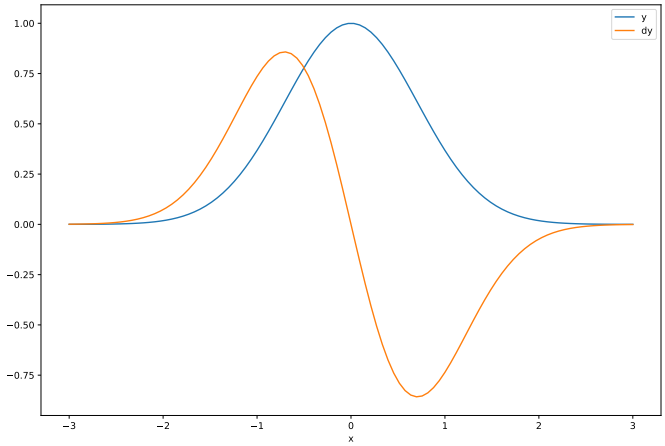



Figura 8.12: Graph of $e^{-x^2}$ and its derivative

```
df2 = lambda x0, h=0.0001 : (df(x0+h)-df(x0-h))/(2*h)
df3 = lambda x0, h=0.0001 : (df2(x0+h)-df2(x0-h))/(2*h)

def newton(x=1, n=100):
  res = [x]
  res.append(res[-1] - df2(res[-1])/df3(res[-1]))
  return(res)

x = newton()

import pandas as pd
dados = pd.DataFrame({'x':x,
                      'y':map(f,x),
                      'dy':map(df,x),
                      'dy2':map(df2,x),
                      'dy3':map(df3,x)})
```

Tabela 8.4: Table for Calculating the Maximum Point of dy

| x | y | dy |
|---|---|---|
| -3.0000000 | 0.0001234 | 0.0007405 |
| -2.9393939 | 0.0001769 | 0.0010398 |
| -2.8787879 | 0.0002517 | 0.0014490 |
| -2.8181818 | 0.0003554 | 0.0020034 |

| x | y | dy |
|---|---|---|
| -2.7575758 | 0.0004983 | 0.0027484 |
| -2.6969697 | 0.0006936 | 0.0037411 |
| -2.6363636 | 0.0009582 | 0.0050525 |
| -2.5757576 | 0.0013142 | 0.0067701 |
| -2.5151515 | 0.0017892 | 0.0090002 |
| -2.4545455 | 0.0024181 | 0.0118704 |
| -2.3939394 | 0.0032440 | 0.0155319 |
| -2.3333333 | 0.0043202 | 0.0201611 |
| -2.2727273 | 0.0057114 | 0.0259610 |
| -2.2121212 | 0.0074953 | 0.0331610 |
| -2.1515152 | 0.0097643 | 0.0420163 |
| -2.0909091 | 0.0126272 | 0.0528047 |
| -2.0303030 | 0.0162099 | 0.0658222 |
| -1.9696970 | 0.0206569 | 0.0813756 |
| -1.9090909 | 0.0261311 | 0.0997734 |
| -1.8484848 | 0.0328141 | 0.1213128 |
| -1.7878788 | 0.0409047 | 0.1462653 |
| -1.7272727 | 0.0506168 | 0.1748582 |
| -1.6666667 | 0.0621765 | 0.2072551 |
| -1.6060606 | 0.0758172 | 0.2435339 |
| -1.5454545 | 0.0917737 | 0.2836641 |
| -1.4848485 | 0.1102753 | 0.3274843 |
| -1.4242424 | 0.1315371 | 0.3746814 |
| -1.3636364 | 0.1557498 | 0.4247723 |
| -1.3030303 | 0.1830698 | 0.4770909 |
| -1.2424242 | 0.2136069 | 0.5307807 |
| -1.1818182 | 0.2474135 | 0.5847956 |
| -1.1212121 | 0.2844731 | 0.6379093 |
| -1.0606061 | 0.3246897 | 0.6887358 |
| -1.0000000 | 0.3678794 | 0.7357589 |
| -0.9393939 | 0.4137634 | 0.7773737 |
| -0.8787879 | 0.4619641 | 0.8119369 |
| -0.8181818 | 0.5120047 | 0.8378259 |
| -0.7575758 | 0.5633123 | 0.8535035 |
| -0.6969697 | 0.6152253 | 0.8575868 |
| -0.6363636 | 0.6670044 | 0.8489146 |
| -0.5757576 | 0.7178485 | 0.8266134 |
| -0.5151515 | 0.7669136 | 0.7901534 |
| -0.4545455 | 0.8133355 | 0.7393959 |
| -0.3939394 | 0.8562540 | 0.6746243 |
| -0.3333333 | 0.8948393 | 0.5965595 |
| -0.2727273 | 0.9283187 | 0.5063557 |
| -0.2121212 | 0.9560019 | 0.4055765 |
| -0.1515152 | 0.9773047 | 0.2961529 |

| x | y | dy |
|---|---|---|
| -0.0909091 | 0.9917696 | 0.1803217 |
| -0.0303030 | 0.9990821 | 0.0605504 |
| 0.0303030 | 0.9990821 | -0.0605504 |
| 0.0909091 | 0.9917696 | -0.1803217 |
| 0.1515152 | 0.9773047 | -0.2961529 |
| 0.2121212 | 0.9560019 | -0.4055765 |
| 0.2727273 | 0.9283187 | -0.5063557 |
| 0.3333333 | 0.8948393 | -0.5965595 |
| 0.3939394 | 0.8562540 | -0.6746243 |
| 0.4545455 | 0.8133355 | -0.7393959 |
| 0.5151515 | 0.7669136 | -0.7901534 |
| 0.5757576 | 0.7178485 | -0.8266134 |
| 0.6363636 | 0.6670044 | -0.8489146 |
| 0.6969697 | 0.6152253 | -0.8575868 |
| 0.7575758 | 0.5633123 | -0.8535035 |
| 0.8181818 | 0.5120047 | -0.8378259 |
| 0.8787879 | 0.4619641 | -0.8119369 |
| 0.9393939 | 0.4137634 | -0.7773737 |
| 1.0000000 | 0.3678794 | -0.7357589 |
| 1.0606061 | 0.3246897 | -0.6887358 |
| 1.1212121 | 0.2844731 | -0.6379093 |
| 1.1818182 | 0.2474135 | -0.5847956 |
| 1.2424242 | 0.2136069 | -0.5307807 |
| 1.3030303 | 0.1830698 | -0.4770909 |
| 1.3636364 | 0.1557498 | -0.4247723 |
| 1.4242424 | 0.1315371 | -0.3746814 |
| 1.4848485 | 0.1102753 | -0.3274843 |
| 1.5454545 | 0.0917737 | -0.2836641 |
| 1.6060606 | 0.0758172 | -0.2435339 |
| 1.6666667 | 0.0621765 | -0.2072551 |
| 1.7272727 | 0.0506168 | -0.1748582 |
| 1.7878788 | 0.0409047 | -0.1462653 |
| 1.8484848 | 0.0328141 | -0.1213128 |
| 1.9090909 | 0.0261311 | -0.0997734 |
| 1.9696970 | 0.0206569 | -0.0813756 |
| 2.0303030 | 0.0162099 | -0.0658222 |
| 2.0909091 | 0.0126272 | -0.0528047 |
| 2.1515152 | 0.0097643 | -0.0420163 |
| 2.2121212 | 0.0074953 | -0.0331610 |
| 2.2727273 | 0.0057114 | -0.0259610 |
| 2.3333333 | 0.0043202 | -0.0201611 |
| 2.3939394 | 0.0032440 | -0.0155319 |
| 2.4545455 | 0.0024181 | -0.0118704 |
| 2.5151515 | 0.0017892 | -0.0090002 |

| x | y | dy |
|---|---|---|
| 2.5757576 | 0.0013142 | -0.0067701 |
| 2.6363636 | 0.0009582 | -0.0050525 |
| 2.6969697 | 0.0006936 | -0.0037411 |
| 2.7575758 | 0.0004983 | -0.0027484 |
| 2.8181818 | 0.0003554 | -0.0020034 |
| 2.8787879 | 0.0002517 | -0.0014490 |
| 2.9393939 | 0.0001769 | -0.0010398 |
| 3.0000000 | 0.0001234 | -0.0007405 |

## 8.22 Optimization in Several Variables

Just as there is the theory of optimizing functions of one variable by calculating derivatives, functions of two or more variables can also have their endpoints calculated by similar techniques.

In the case of functions of one variable we have *maximum*, *minimum* and *inflection* points. In the case of functions of two variables there are *maximum*, *minimum* and the so-called *saddle point*, where the function has a maximum when viewed from one of the coordinate axes and a minimum when viewed from the other.

The algebraic form of determining such points is given below.

## 8.23 Points of maximum

In this case we have a point $(x_0, y_0)$ in the domain of a function $z = f(x, y)$ which meets the following conditions:

$$z_x(x_0, y_0) = z_y(x_0, y_0) = 0$$

$$z_{xx}(x_0, y_0) < 0$$

and

$$z_{yy}(x_0, y_0) < 0$$

And the determinant:

$$D = \begin{vmatrix} z_{xx}(x_0, y_0) & z_{xy}(x_0, y_0) \\ z_{yx}(x_0, y_0) & z_{yy}(x_0, y_0) \end{vmatrix} > 0 \tag{8.3}$$

## 8.24 Points of minimum

In this case we have a point $(x_0, y_0)$ on the domain of a function $z = f(x, y)$ which meets the following conditions:

$$z_x(x_0, y_0) = z_y(x_0, y_0) = 0$$

$$z_{xx}(x_0, y_0) > 0$$

and

$$z_{yy}(x_0, y_0) > 0$$

And the determinant

$$D = \begin{vmatrix} z_{xx}(x_0, y_0) & z_{xy}(x_0, y_0) \\ z_{yx}(x_0, y_0) & z_{yy}(x_0, y_0) \end{vmatrix} > 0 \tag{8.4}$$

## 8.25 Saddle points

In this case we have a point $(x_0, y_0)$ on the domain of a function $z = f(x, y)$ which meets the following conditions:

$$z_x(x_0, y_0) = z_y(x_0, y_0) = 0$$

And the determinant

$$D = \begin{vmatrix} z_{xx}(x_0, y_0) & z_{xy}(x_0, y_0) \\ z_{yx}(x_0, y_0) & z_{yy}(x_0, y_0) \end{vmatrix} < 0 \tag{8.5}$$

## 8.26 Possible problems

It is very rare, but if the situation occurs where the determinant:

$$D = \begin{vmatrix} z_{xx}(x_0, y_0) & z_{xy}(x_0, y_0) \\ z_{yx}(x_0, y_0) & z_{yy}(x_0, y_0) \end{vmatrix} = 0 \tag{8.6}$$

we say that the test of the second derivative fails. Such cases will not be studied in this text.

## 8.27 Examples

7. Analyze the endpoints of the function: $f(x,y) = \frac{1}{x^2+y^2+3x-2y+1}$

Let's perform this example starting with `sympy`. First we calculate its first derivatives with respect to x and y. For $z_x$ we have:

```
import sympy as sp
x, y = sp.symbols("x y")
z = 1/(x**2 + y**2 + 3*x -2*y +1)
sp.diff(z,x)
```

```
(-2*x - 3)/(x**2 + 3*x + y**2 - 2*y + 1)**2
```

```
sp.diff(z,y)
```

```
(2 - 2*y)/(x**2 + 3*x + y**2 - 2*y + 1)**2
```

We now calculate $(x_0, y_0)$ such that $z_x(x_0, y_0) = 0$ and $z_y(x_0, y_0) = 0$.

```
import sympy as sp
x, y = sp.symbols('x y');
f = 1/(x**2 + y**2 + 3*x -2*y + 1);
sp.solve([f.diff(x), f.diff(y)],[x,y])
```

```
{x: -3/2, y: 1}
```

For assurance, we compute the values of $x$ that make the denominator of $z_x$ and $z_y$ null for $y = 1$. In this case we do:

```
import sympy as sp
x, y = sp.symbols('x y');
f = 1/(x**2 + y**2 + 3*x -2*y + 1)
print("f = ", f);
```

```
f =  1/(x**2 + 3*x + y**2 - 2*y + 1)
```

```
dfx = f.diff(x);
print("dfx = ", dfx);
```

```
dfx =  (-2*x - 3)/(x**2 + 3*x + y**2 - 2*y + 1)**2
```

```
dfx_y1 = dfx.subs(y,1)
print("dfx_y1 = ", dfx_y1);
```

```
dfx_y1 =  (-2*x - 3)/(x**2 + 3*x)**2
```

```
den_dfx_y1 = sp.fraction(dfx_y1)[1];
print("den_dfx_y1 = ", den_dfx_y1);
```

```
den_dfx_y1 =  (x**2 + 3*x)**2
```

```
sp.solve(den_dfx_y1,x)
```

```
[-3, 0]
```

Then the candidate point is

$$(x, y) = (-\frac{3}{2}, 1)$$

Let us now calculate the second derivatives at the point $(-3/2, 1)$. We start with $z_{xx}$. In this case:

```
import sympy as sp
x, y = sp.symbols('x y');
f = 1/(x**2 + y**2 + 3*x -2*y + 1);
f.diff(x,2).subs(x,-3/2).subs(y,1)
```

```
-0.395061728395062
```

We now calculate, the second derivative with respect to $y$.

```
import sympy as sp
x, y = sp.symbols('x y');
f = 1/(x**2 + y**2 + 3*x -2*y + 1);
f.diff(y,2).subs(x,-3/2).subs(y,1)
```

```
-0.395061728395062
```

Next we get the value of the cross derivative,

```
import sympy as sp
x, y = sp.symbols('x y');
f = 1/(x**2 + y**2 + 3*x -2*y + 1);
f.diff(x).diff(y).subs(x,-3/2).subs(y,1)
```

```
0
```

As a last step we calculate the determinant, which is greater than 0.

```
import sympy as sp
x, y = sp.symbols('x y');
f = 1/(x**2 + y**2 + 3*x -2*y + 1);
f_xx = f.diff(x,2)
f_yy = f.diff(y,2)
f_xy = f.diff(x).diff(y)
A = sp.Matrix([[f_xx, f_xy],[f_xy, f_yy]])
detA = sp.det(A)
detA.subs(x,-3/2).subs(y,1)
```

```
0.156073769242493
```

This set of results: $z_x(-\frac{3}{2},1)=0$, $z_y(-\frac{3}{2},1)=0$, $z_{xx}(-\frac{3}{2},1)=-\frac{31}{81}$, $z_{yy}(-\frac{3}{2},1)=-\frac{31}{81}$ and $D>0$ allow us to state that the point $z(-\frac{3}{2},1)$ is a maximum.

8. Analyze the endpoints of the function: $f(x,y)=xye^{\frac{-9y^2-16x^2}{288}}$

The sympy resolution can be seen below. First we compute the candidate points, that is, the pairs of values $(x,y)$ for which $f_x=0$ and $f_y=0$

```
import sympy as sp
x, y = sp.symbols("x y")
f = x*y*sp.exp((-9*y**2-16*x**2)/288)
f_x = sp.diff(z,x);
print("f_x = ", f_x);
```

```
f_x =  (-2*x - 3)/(x**2 + 3*x + y**2 - 2*y + 1)**2
```

```
f_y = sp.diff(z,y);
print("f_y = ", f_y);
```

```
f_y =  (2 - 2*y)/(x**2 + 3*x + y**2 - 2*y + 1)**2
```

```
sols = sp.solve([f_x, f_y],[x,y]);
print("Ptos candidates = ", sols)
```

```
Ptos candidates =  {x: -3/2, y: 1}
```

We calculate the value of the determinant algebraically :

```
import sympy as sp
x, y = sp.symbols("x y")
f = x*y*sp.exp((-9*y**2-16*x**2)/288)
f_x = f.diff(x); f_y = f.diff(y);
f_xx = f.diff(x,2); f_yy = f.diff(y,2)
```

```
f_xy = f.diff(x).diff(y)
A = sp.Matrix([[f_xx, f_xy],[f_xy, f_yy]])
detA = sp.det(A); detA.simplify()
```

```
(-16*x**4*y**2 - 256*x**4 - 9*x**2*y**4 + 720*x**2*y**2 + 4608*x**2 - 81*y**4
```

And the points of interest (or candidate points):

```
sols = sp.solve([f_x, f_y],[x,y])
sols
```

```
[(-3, -4), (-3, 4), (0, 0), (3, -4), (3, 4)]
```

Next we create a function to calculate the value of the determinant and the second derivatives at a candidate point and use it to evaluate each of them.

```
def evaluate(pto):
  x0 = pto[0]; y0 = pto[1]
  detAx0y0 = detA.subs(x,x0).subs(y,y0).evalf()
  fxx = f.diff(x,2).subs(x,x0).subs(y,y0).evalf()
  if detAx0y0<0:
    result = "saddle"
  else:
    if fxx <0:
      result = "maximum"
    else:
      result = "minimum"
  detAx0y0 = np.round(float(detAx0y0),5)
  fxx = np.round(float(fxx),5)
  return(pto, detAx0y0, fxx, result)
```

```
for sol in sols:
  print(evaluate(sol))
```

```
((-3, -4), 0.54134, -0.98101, 'maximum')
((-3, 4), 0.54134, 0.98101, 'minimum')
((0, 0), -1.0, 0.0, 'saddle')
((3, -4), 0.54134, 0.98101, 'minimum')
((3, 4), 0.54134, -0.98101, 'maximum')
```

9. Evaluate the endpoints of the function:

$$f(x,y) = x.ln\left(\frac{y^2}{x}\right) - x.y^2 + 3.x$$

In this exercise the attempt to solve the system of equations, directly by `sympy` fails.

```
import sympy as sp
x, y = sp.symbols("x y", real = True)
f = x * sp.log((y**2)/x) - x*(y**2) + 3*x
sp.solve([f.diff(x), f.diff(y)],[x,y])
```

```
[(x, -1), (x, 1)]
```

If we try to solve the equations separately, we get the pairs of values. First $f_y = 0$ for $y$. We get as expected $y = \pm 1$

```
import sympy as sp
x, y = sp.symbols("x y", real = True)
f = x * sp.log((y**2)/x) - x*(y**2) + 3*x
sp.solve(f.diff(y),y)
```

```
[-1, 1]
```

Replacing $y$ by $-1$ or by $+1$ in $f_x = 0$ and solving for $x$ we get $x = e$:

```
sp.solve(f.diff(x).subs(y,-1),x)
```

```
[E]
```

We can then calculate the second derivatives of $f$ and analyze the points $[e, -1]$ and $[e, +1]$. Below we calculate the expression $f_{xx}.f_{yy} - f_{xy}^2$

```
import sympy as sp
x, y = sp.symbols("x y", real = True)
f = x * sp.log((y**2)/x) - x*(y**2) + 3*x
fxx = f.diff(x,2)
fyy = f.diff(y,2)
fxy = f.diff(x).diff(y)
detA = fxx * fyy - fxy**2
detA.simplify()
```

```
-4*y**2 + 10 - 2/y**2
```

Evaluating the two expressions at $(e, -1)$ we see that this is a point of maximum.

```
detA.subs(x,sp.E).subs(y,-1), fxx.subs(x,sp.E).subs(y,-1).evalf()
```

```
(4, -0.367879441171442)
```

Similarly, $(e, +1)$ is also a maximum point.

```
detA.subs(x,sp.E).subs(y,1), fxx.subs(x,sp.E).subs(y,1).evalf()
```

```
(4, -0.367879441171442)
```

This is only possible if there is a discontinuity in the interval from $y = -1$ to $y = 1$. The result in this case checks out, because at $y = 0$ the expression $\ln(\frac{y^2}{x})$ has a limit tending to $-\infty$.

```
import sympy as sp
x, y = sp.symbols("x y")
f = sp.log(y**2/x)
sp.limit(f,y,0)
```

```
-oo
```

10. Evaluate the endpoints of the function: $\frac{x}{y^2+x^2+4}$

For this problem we will create a Python function that receives an expression, computes each critical point, and determines whether it is a saddle point, a minimum, or a maximum.

```
import numpy as np
import sympy as sp
x, y = sp.symbols("x y")
z = x/(y**2 + x**2 + 4)

def evaluate(f):
  fx = f.diff(x)
  fy = f.diff(y)
  ptos = sp.solve([fx, fy],[x,y])
  fxx= f.diff(x,2)
  fyy = f.diff(y,2)
  fxy = f.diff(x).diff(y)
  ptos = sp.solve([fx,fy],[x,y])
  results = []
  for pto in ptos:
    x0 = pto[0]
    y0 = pto[1]
    fxx0 = fxx.subs(x,x0).subs(y,y0).evalf()
    fyy0 = fyy.subs(x,x0).subs(y,y0).evalf()
    fxy0 = fxy.subs(x,x0).subs(y,y0).evalf()
    detA =  fxx0 * fyy0 - fxy0**2
    if detA < 0:
      result = "saddle"
    else:
      if fxx0 < 0:
        result = "maximum"
      else:
```

```
        result = "minimum"
    results.append([pto, detA.round(4), \
                    fxx0.round(4), result])
  return(results)

for solution in evaluate(z):
  print(solution)
```

```
[(-2, 0), 0.0039, 0.0625, 'minimum']
[(2, 0), 0.0039, -0.0625, 'maximum']
```

## 8.28 Univariate Maximums and Minimums Exercises

1. Given the function $f\,(x) = 3x3 + x2 - 4x - 1$, use the formula: $f\text{'}\,(x) = [f\,(x+h) - f\,(x)]\,/h$ , with $h = 0.1$ and find the value of $x$ that makes the derivative of $f\,(x)$ equal to 0.
2. Calculate the points of maximum and minimum (if any) of the functions f(x) =
   (a) $x.\ln x$
   (b) $x - 1.\ln x$
   (c) $2x.\ln x$
3. Find the zeros, maxima, minima, and plot the graph of the function: $f(x) = \frac{1}{(x-0,1)^2+0.01} + \frac{1}{(x-1,2)^2+0.04} - 10$ on the interval [-2;2]
4. Find the value of x that makes $f(x) = -x^2 + 3x - 4$ a maximum, by whatever method you find most convenient
5. Calculate the local maximum for the function: $f(x) = x^3 - 5x^2 + 3x - 4$
6. Determine the coefficients $p$ and $q$ in the expression $y = x^2 + px + q$ such that it has a minimum at $x = 1$ and $y = 3$
7. Given the function $f\,(x) = 2x3 + x2 - 5x - 1$, use the formula $f'(x) = f\frac{f(x+h)-f(x)}{h}$, with $h = 0.1$ and find the point of maximum of the function $f(x)$.

## 8.29 Application Exercises in Geometry

1. A person is at the edge of a circular lake of radius R. She can swim across the lake with speed V1 or walk on its edge with speed V2. You are asked to determine the trajectory that will get the person to the opposite edge of the lake in the shortest possible time.

2. What is the largest garden that can be built next to a house with a 40m fence?

3. A box with a square base, without a lid, must have a total surface area of 12m2. Determine the length of the base edge and the height of the box of maximum volume.

4. Find the largest area that can be used by a 500m perimeter athletics track consisting of two semicircles and two straight, parallel lanes

5. An engineer is designing a sports stadium for a city and the rectangular field is to be inscribed within a grandstand represented in his design by the following ellipse: $\frac{1}{16}x^2 + \frac{1}{9}y^2 = 1$ What are the dimensions for this field to have the largest possible area? Prove

6. A cattle farmer stands at a point $A$ located 200 meters from the bank of a river. He wants to take his herd to drink water in the river and from there he wants to proceed to a grazing point $B$ located 400 meters from the river bank. Assume that the river bank is in a straight line. The horizontal distance separating the point where the farmer is now and the grazing point is 1,000 meters. You are asked to calculate at which point on the bank the farmer should take the cattle to drink water so that the total distance traveled is minimal.

7. A wire 10 cm long is to be cut into two pieces, one of which will be twisted to form a square and the other to form a circle. In what way should it be cut so that the sum of the areas of the regions bounded by the figures obtained is minimal?

8. 320 is available to be spent on building a fence in a garden. The fence on the side of the garden facing the street costs 6 per meter, and the fence on the other three sides costs 2 per meter. Calculate the dimensions of the garden that maximizes the area.

## 8.30 Application Exercises in Economics

1. The billing function of a firm producing a single product is: $R(x) =$ 200–1600/ $(x + 8)$ –$x$. Find the value of x that results in the maximum revenue.

2. A firm producing a single product estimates that its daily cost function (in appropriate units) is $C(x) = x3–6x2 + 13x + 15$, and its billing function is $R(x) = 28x$. Find the value of x that maximizes the daily profit.

3. Consider the profit function $L = R - C$ where $R$ is the revenue and $C$ is the cost. Express the optimal profit condition in terms of marginal revenue and marginal revenue.

4. After an advertising campaign, sales of a product often increase and, after

some time, decrease. Suppose that t days after the end of the campaign, the daily sales are $f(t) = -3t^2 + 9t + 100$. What is the growth rate of sales on the fourth day? After how many days would sales reach a maximum (or minimum)? What is the value of sales at this point? Is this a local maximum or minimum point?

5. The demand equation for a certain good is $p = (1/12)x^2 - 10x + 300$, for $0 \leq x \leq 60$. Find the value of $x$ that corresponds to the price that maximizes revenue.

6. Suppose that the demand equation of a monopolist is given by $p = 150 - 0.02.x$, where p is the price, $x$ is the quantity sold. Suppose the cost function is given by $C(x) = 10x + 300$, find the value of x that maximizes profit. Prove that it is a maximum by calculating the value of the 2nd derivative at this point.

7. A market survey indicated that the relationship between the price of sugar (in dollars per kilogram) and the quantity demanded (in thousands of tons) in a given country are related by the function: $p(x) = -0.25.x + 50$. Knowing that the cost of production $C(x)$ is given by the expression $C(x) = 2.25x^2 + 3x + 70$ ask (a) assemble in spreadsheet the expression of the expected profit as a function of the price charged, that is $L(p)$ and of the expected profit as a function of the quantity demanded, that is $L(x)$ and (b) determine the price $p$ and the quantity $x$ that make the production of sugar as profitable as possible

8. A few years ago it was estimated that the demand for steel approximately satisfied the equation $p = 256 - 50x$ and the total cost to produce x units of steel was $C(x) = 182 + 56x$. The quantity x was measured in millions of tons, and the price and total cost were measured in millions of dollars. Determine the level of output and corresponding price that maximizes profit

9. The monthly demand equation for an electric company is estimated to be: $p = 60 - (10 - 5).x$ where p is measured in dollars and x is measured in thousands of kilowatt-hours. The company's fixed costs are $7,000,000$ per month and variable costs are 30 per $1,000$ kilowatt-hours of electricity generated, so the cost function is: $C(x) = 7,106 + 30x$. Find the value of x and the price per 1,000 kilowatt-hours that maximizes the company's profit.

10. Suppose that a company's total revenue function is $R(x) = 300ln(x + 1)$, such that selling $x$ units of a product produces a return of $R(x)$ dollars. Assume also that the total cost to produce x units is $C(x)$ dollars with $C(x) = 2x$. Find the value of x for which the profit function $R(x) - C(x)$ will be maximized. Show that the profit function has a relative maximum and not a relative minimum for such a value of $x$

11. When the average ticket price for an opera performance is 50, the audience

is 4,000 people. When the price was raised to 52, the audience declined to 3,800 people. Consider the linear demand curve and from this data calculate the ticket price that will maximize the theater's revenue.

12. The average traffic on a highway is 36,000 cars per day when the toll is 1 per car. A poll found that increasing the toll would mean 300 fewer cars for every cent increase. How high is the toll to maximize revenue?

13. Suppose that on a certain route a regional airline carries 8000 passengers a month, each paying 50. The airline wants to increase the fare. However, the market research department estimates that for every 1 increase, the airline will lose 100 passengers. Determine the price that maximizes revenue.

14. A travel agency offers a cruise to several Caribbean islands for a period of 3 days and 2 nights. For a group of 12 people, the cost per person is 800. For each additional person the cost is reduced by 20 per person. The maximum number of people that can be considered for the cruise is 23. Which group size gives the highest revenue for the travel agency?

15. A furniture store expects to sell 640 sofas at regular intervals next year. The manager plans to order these sofas from a manufacturer by placing several orders of the same size and equally spaced during the year. The cost of each delivery is 160, and the storage cost, based on the average number of sofas in stock, is 32 per year for each sofa. Determine the quantity of sofas in each order that will minimize the total stocking cost.

16. A furniture store expects to sell 740 sofas at regular intervals next year. The manager plans to order these sofas from a manufacturer by placing several orders of the same size and equally spaced during the year. The cost of each delivery is 140, and the storage cost, based on the average number of sofas in stock, is 42 per year for each sofa. Determine the quantity of sofas in each order that will minimize the total cost of stocking.

## 8.31 Multivariate Maximums and Minimums Exercises

1. Rank the critical points of the following functions:
   (a) $x^2 + 3xy + 4y^2 - 6x + 2y$
   (b) $x^2 + y^3 + xy - 3x - 4y + 5$
   (c) $x^3 + 2xy + y^2 - 5x$
   (d) $-x^2 + y^2 + 2xy + 4x - 2y$
   (e) $x^3 - 3x^2y + 27y$
   (f) $x^2 - 4xy + 4y^2 - x + 3y + 1$

(g) $x^5 + y^5 - 5x - 5y$

2. A financial institution intends to open a bank branch near shopping centers. To this end, the planning area of the financial institution has found that there are three major shopping centers at the following points $A(1, 5)$, $B(0, 0)$, and $C(8, 0)$, the units being in kilometers. At which point $P(x, y)$ should the branch be installed so that the sum of the distances from point $P$ to points $A, B$, and $C$ is as small as possible?

3. A monopolist sells his product in two countries and can charge different prices in each country. Let x be the number of units to be sold in the first, and y be the number of units to be sold in the second country. As a consequence of the laws of demand, the monopolist needs to set the price at $97 - (x/10)$ in the first country, and $83 - (y/20)$ in the second country, to be able to sell all the units. The cost in producing these units is $20,000 + 3(x + y)$. Find the values of $x$ and $y$ that maximize profit.

4. Let $P(x, y) = 10 - 2x^2 + xy - y^2 + 5y$ be a production function, where x and y are quantities of two inputs used in the manufacture of the quantity $P(x, y)$ of a product. The unit price of each input is $R$\$3.00 and the finished product is sold for $R$\$6.00 a unit. Calculate the maximum profit, the quantities $x$ and $y$ for which it occurs, and prove that the point found is a point of maximum by the test of the second derivative.

5. A certain company that produces bags for carrying any sporting goods charges different prices to its retail and wholesale customers. The demand function for the retail market is $p = 500 - x$, where $p$ represents the price of each bag and x the quantity sold. The demand function for the wholesale market is $q = 350 - 1.5y$, where q represents the unit price and y represents the quantity sold by the firm. The total cost function is given by: $C(x, y) = 50000 + 20(x + y)$. All data are in reals. What prices p and q should the firm set to maximize its profit on the sale of the bags?

6. The store SPORTIST offers all kinds of sporting goods. A research made by the Marketing department showed that, in general, the demand for each product depends not only on its own price, but on the competitor's price. So, if the Nadal racket is sold for x reals and the Federer racket for y reals, the demand for the Nadal racket will be 300 - 20x + 30y rackets per year and the demand for the Federer racket will be 200 + 40x – 10y rackets per year.

   (a) Express the total annual revenue of the store R(x, y) in terms of x and y.

   (b) The store decides to rapidly increase sales. Use the concept of partial derivatives to decide whether the store should increase the price of the Nadal racquet or the price of the Federer racquet. Consider that the Nadal racquet currently costs R\$100.00 and the Federer racquet, R\$200.00.

7. A factory decides to produce blackboards in two versions: blackboard and whiteboard. The unit cost of producing blackboards is 40.00 and the unit cost of producing whiteboards is 60.00. The Marketing Department of the factory estimates that if blackboards are sold at x reals and whiteboards at y reals each, then $500(y-x)$ copies of blackboards and $45,000+500(x-2y)$ copies of whiteboards will be sold. Determine the selling price the factory must set for each version in order to maximize its profit. Justify.

8. A company produces and sells two products, denoted by I and II, which are sold for \$10 and \$9 per unit. The cost to produce x units of product I and y units of product II is: $C(x,y) = 400+2x+3y+0.01(3x^2+xy+3y^2)$. Find the values of x and y that maximize the company's profit. Prove by the test of the second derivative that the point you find is a maximum.

## 8.32 Least Squares Exercises

1. Given the table of points $\begin{bmatrix} x & y \\ 1 & 1 \\ 2 & 6 \\ 3 & 9 \end{bmatrix}$, find the parameters b0 and b1 of the straight line $y = b0 + b1.x$, which passes as close as possible to the three points, by whatever method you find most convenient.

2. A laboratory produces one type of serum in quantity x. The table **??** shows the quantities (in liters) demanded as a function of the price of each liter. Based on the data presented, calculate the linear regression expression of demand and the demand (in liters) that can be expected for a price of R\$15.00 per liter.

$$\begin{bmatrix} P\left(\frac{R\$}{Litro}\right) & D\,(Litros) \\ 2,00 & 5,00 \\ 6,00 & 4,58 \\ 11,00 & 4,00 \\ 18,00 & 3,00 \\ 23,00 & 2,00 \end{bmatrix}$$

3. A company intends to launch a pharmaceutical product in two countries, x and y. The plant will have an annual fixed cost of $500,000$ per year and a unit production cost of 2.0 per unit of production. Fractional quantities of the product can be sold. Market research to assess the sensitivity of demand to the unit selling price of the product in countries x and y yielded the following results (all prices and costs are in US dollars). Based on the data presented below ask:

   (a) Estimate the demand function for the product in region x, assuming a linear relationship between price and quantity

(b) Estimate the demand function for the product in region y, assuming a linear relationship between price and quantity

(c) Determine the price to be charged in region x and in region y so as to maximize the total profit of the company

(d) Calculate the value of the maximum profit of the company

$$\begin{bmatrix} Px & Qx \\ 10 & 34 \\ 15 & 32 \\ 20 & 24 \end{bmatrix}$$

$$\begin{bmatrix} Py & Qy \\ 12 & 31 \\ 18 & 30 \\ 24 & 25 \end{bmatrix}$$

4. Suppose that a car factory, in order to analyze the fuel consumption of a specific model, made 7 trips and recorded the distance traveled (km) and consumption (l), obtaining then the following 7 pairs of values available in the table 8.5. Based on these data answer:

   (a) Write the equation of the estimated regression line that relates distance to consumption (approximate your answer to two decimal places).

   (b) With 16 liters of fuel, which of the two distances seems more likely to be traveled to you: 190 km or 205 km? Explain.

   (c) Assuming the price of a liter of gasoline is 2.52, what is the (estimated) amount spent on a 820 km journey?

Tabela 8.5: Pairs Distance x Consumption

| y_distance | x_consumption |
|---|---|
| 20 | 2 |
| 40 | 3 |
| 80 | 5 |
| 120 | 9 |
| 160 | 12 |
| 200 | 14 |
| 250 | 18 |

5. From the table **??** , relate the variable Number of Cars from an expression to the variable Time, of type Number = b0 + b1.Time using the matrix formula $\vec{b} = (X^t.X)-1.(X^t.Y)$

Tabela 8.6: Annual Car Sales Table {chunkotimtable3}

| Year | Cars |
|---|---|
| 1980 | 104.6 |
| 1985 | 114.7 |
| 1989 | 122.8 |
| 1990 | 123.3 |
| 1991 | 123.3 |
| 1992 | 120.3 |
| 1993 | 121.1 |
| 1994 | 122.0 |

6. From the 8.7 table, relate the variable Sales from an expression to the variable Advertisement, of type $V = \beta_1.P + \beta_0$, using the matrix formula $\vec{enumerate} = (X^T.X)^{-1}.(X^T.Y)$

Tabela 8.7: Sales and Advertisement

| P | V |
|---|---|
| 10 | 155.35 |
| 11 | 161.55 |
| 12 | 169.67 |
| 13 | 185.77 |
| 14 | 174.05 |
| 15 | 186.03 |
| 16 | 183.03 |
| 17 | 193.57 |
| 18 | 185.97 |
| 19 | 197.98 |
| 20 | 203.31 |

7. Find the exponential regression function $y = ae^{mathitbx}$ that minimizes the squared error for the points in table 8.32:

$$\begin{bmatrix} x & y \\ 0 & 3,10 \\ 1 & 5,50 \\ 2 & 17,60 \\ 3 & 57,30 \\ 4 & 199,70 \end{bmatrix}$$

8. Find $a$ and $b$ in the exponential regression $y = b.x^a$ in the table below

| Income | Pop with Income greater than |
|---|---|
| 50 | 25168 |
| 60 | 18224 |
| 70 | 13533 |
| 80 | 9950 |
| 90 | 7642 |
| 100 | 6129 |

Tabela 8.8: Pareto distribution

| P/L | Risco | Retorno |
|---|---|---|
| 7,4 | 1,0 | 7,6 |
| 11,1 | 1,3 | 13,0 |
| 8,7 | 1,1 | 8,9 |
| 11,2 | 1,2 | 10,9 |
| 11,6 | 1,7 | 12,1 |
| 12,2 | 1,3 | 12,8 |
| 12,5 | 1,2 | 11,3 |
| 12,5 | 1,3 | 14,1 |
| 13,0 | 1,6 | 14,8 |
| 13,4 | 1,4 | 16,7 |

Tabela 8.9: Return x PL and Risk

$$\begin{bmatrix} x & y \\ 0,1 & 2,4 \\ 1,0 & 3,6 \\ 2,0 & 5,0 \\ 3,0 & 7,2 \\ 4,0 & 12,1 \\ 5,0 & 17,5 \end{bmatrix}$$

9. Studying the distribution of wealth in a population, economists assume that it is distributed according to the Pareto curve $y = b/x^a$, where a and b are parameters to be determined. The table 8.8 contains the U.S. Census data for 2001. In this table, x represents annual income in thousands of dollars and y represents the number of men (in thousands) with annual income greater than or equal to x. Find the Pareto curve $y = b/x^a$, which best fits the data. Calculate the regression using the method you find most convenient and detail all the steps of your solution.

10. Taking the values from table 8.9 as a basis, find the best estimate for the Return when the P/L equals 9.5 and the risk equals 11.0. Use as the estimation expression $Return = b_0 + b_1.(P/L) + b_2.(Risk)$.

11. From the table 8.10, relate the variable Price (y) to the variable Property

| Área | Preço |
|---|---|
| 104 | 69 |
| 114 | 73 |
| 124 | 76 |
| 145 | 78 |
| 166 | 86 |
| 187 | 97 |
| 197 | 103 |
| 197 | 108 |
| 218 | 129 |
| 218 | 137 |
| 228 | 160 |

Tabela 8.10: Price x Real Estate Area

Area (x), according to the regression expression $y = b_0 + b_1.x + b_2.x_2 + b_3.x_3$, using the matrix formula $\vec{b} = (X^t.X)^{-1}.(X^t.Y)$

12. From the 8.11 table, assemble a ranking expression for the listed companies, using Solver in obtaining the expression and use such criteria to rank the companies in the table itself:

13. From the following 8.12 table, put together a classification expression to determine whether or not a person will be a prospective buyer of lawn mowers. Use Solver to obtain the expression and use this criteria to classify the people in the table itself as Owners or not. Note: Propr.=1 Person does NOT own a lawnmower, Propr=2 Person HAS a lawnmower.

Observation & Income & Tam.Res. & Propr.

## 8.33 Conditional Optimization Exercises

1. Find the maximum of:
   (a) $f = x^2 + y^2$ se $x + y = 6$
   (b) $f = 49 - x^2 - y^2$ se $x + 3y = 10$
   (c) $f = 3x + 4y$ se $x^2 + y^2 = 1$
   (d) $f = \sqrt{(x-1)^2 + (y-2)^2}$ se $x^2 + y^2 = 45$
   (e) $f = \sqrt{(x-14)^2 + (y-1)^2}$ se $y = x^2 + 2x - 3$
2. The management of a store wants to build a 600m2 rectangular fence in the store's parking lot to display a piece of equipment. On three of the sides wooden fences will be built, at a cost of $14 per meter length. The fourth side will be built using cement blocks at a cost of $28 per meter

| Company | Grup | Index 1 | Index 2 |
|---|---|---|---|
| 1 | Bom | 8,1 | 0,6 |
| 2 | Bom | 6,6 | 1,0 |
| 3 | Bom | 5,8 | 0,7 |
| 4 | Bom | 12,3 | 0,8 |
| 5 | Bom | 4,5 | 0,7 |
| 6 | Bom | 9,1 | 0,7 |
| 7 | Bom | 1,1 | 0,6 |
| 8 | Bom | 8,9 | 0,8 |
| 9 | Bom | 0,7 | 0,6 |
| 10 | Bom | 9,8 | 0,7 |
| 11 | Ruim | 7,3 | 0,6 |
| 12 | Ruim | 14,0 | 0,5 |
| 13 | Ruim | 9,6 | 0,7 |
| 14 | Ruim | 12,4 | 0,4 |
| 15 | Ruim | 18,4 | 0,5 |
| 16 | Ruim | 8,0 | 0,5 |
| 17 | Ruim | 12,6 | 0,3 |
| 18 | Ruim | 9,8 | 0,7 |
| 19 | Ruim | 8,3 | 0,5 |
| 20 | Ruim | 20,6 | 0,8 |

Tabela 8.11: Good and Bad Companies x 2 Indexes

| Observation | Property | Income | Propr.Size |
|---|---|---|---|
| 1 | 60,0 | 18,4 | 1 |
| 2 | 85,5 | 16,8 | 1 |
| 3 | 64,8 | 21,6 | 1 |
| 4 | 61,5 | 20,8 | 1 |
| 5 | 87,0 | 23,6 | 1 |
| 6 | 110,1 | 19,2 | 1 |
| 7 | 108,0 | 17,6 | 1 |
| 8 | 82,8 | 22,4 | 1 |
| 9 | 69,0 | 20,0 | 1 |
| 10 | 93,0 | 20,8 | 1 |
| 11 | 51,0 | 22,0 | 1 |
| 12 | 81,0 | 20,0 | 1 |
| 13 | 75,0 | 19,6 | 2 |
| 14 | 52,8 | 20,8 | 2 |
| 15 | 64,8 | 17,2 | 2 |
| 16 | 43,2 | 20,4 | 2 |
| 17 | 84,0 | 17,6 | 2 |
| 18 | 49,2 | 17,6 | 2 |
| 19 | 59,4 | 16,0 | 2 |
| 20 | 66,0 | 18,4 | 2 |
| 21 | 47,4 | 16,4 | 2 |
| 22 | 33,0 | 18,8 | 2 |
| 23 | 51,0 | 14,0 | 2 |
| 24 | 63,0 | 14,8 | 2 |

Tabela 8.12: Property x Income and Property Size

length. Find the dimensions of the fence that will minimize the total cost of construction materials.

3. The demand equation of a monopolist is p = 200 -3x and the cost function is $C(x) = 75 + 80x–x2$ for $0 \leq x \leq 40$. Determine:
   (a) The value of x and the corresponding profit-maximizing price
   (b) Suppose the government levies a tax on the monopolist of $4 per unit produced. Determine the new profit-maximizing price
   (c) Suppose the government levies a tax of T dollars per unit produced, so that the new cost function is $isC(x) = 75 + (80 + T)\,x–x2for0 \leq x \leq 40$. Determine the new value of x that maximizes the monopolist's profit as a function of T
   (d) Assuming that the monopolist decreases his output to the level in item c) express the government revenue generated by collecting the tax as a function of T.
   (e) Determine the value of T that will maximize the revenue received by the government
4. Develop a spreadsheet model to calculate the coordinates of the vertices of the largest rectangle that can be inscribed in the circle $x^2 + y^2 = 4$. Use Solver to determine the coordinates of its vertices.
5. Using the Lagrange Multiplier Method, find the three positive numbers whose sum is 15 and whose product is as large as possible.
6. A multi-sport club wants to buy in the store SPORTSMAN, shorts and socks for the soccer practice of its members. Each pair of socks costs $R20.00 and each pair of shorts costs R10.00$. The utility function of the club to purchase x socks and y shorts, can be expressed by: $U(x, y) = 10x^{0.5}y^{0.5}$.
   (a) Use the Lagrange multiplier to determine the optimal consumption combination for the multi-sport club. Justify your answer.
   (b) Calculate the value of $\lambda$ corresponding to the values of x and y from item A.
7. The production function of a firm (i.e. the number of units produced) is given by: $f(x, y) = 64x^{\frac{3}{4}}y^{\frac{1}{4}}$ , where x and y are the number of units of labor and capital used. Suppose labor costs $96 per unit, capital 162$ per unit, and the firm wants to produce a total of 3,456 units. Based on this information you are asked to calculate the number of units of labor and capital that will minimize the total cost of production.
8. The utility of a consumer to purchase x units of one product and y units of a second product is given by the utility function $U(x, y) = x^2y$. Suppose that the consumer purchases 3 units of the first product and 3 units of the second product each month.

(a) Compute U(3,3) and plot the level curve for this value.

(b) Suppose that in a given month, a consumer decides to spend 120.00 to purchase and stock a certain quantity of the two products. How much of each should the consumer buy to maximize his utility? The price per unit for the first product is 4.00 and for the second product is 5.00?

9. The amount of space required by a firm is $f(x, y) = \sqrt{6x^2 + y^2}$, where x and y are respectively the number of units of labor and capital used. Assume that labor costs \$480 per unit and capital costs \$40 per unit and that the firm has \$5,000 to spend. Determine the amount of labor and capital that minimizes the total space used.

10. Suppose a firm produces two products, A and B that use the same raw materials and given amounts of labor, so that quantity x of product A and quantity y of product B must satisfy $9x^2 + 4y^2 = 18,000$. If the profit for each unit of A is \$3 and that for each unit of B is \$4, determine the quantities that must be produced to meet the production constraint and maximize the firm's profit.

11. An employee of the store has decided to take his vacation in July and go to a bar and a disco. He pays \$10.00 to enter the bar and \$60.00 to enter the disco. His monthly salary is R\$2,400.00. He has decided to spend 10 percent of his salary to pay for the entrance fees. Consider that the utility function is given by $U(x, y) = xy$ where $x$ is the number of times he goes into the bar and $y$ is the number of times he goes into the disco.

    (a) In how many times should you go to the bar and in how many times to the disco to maximize the utility (satisfaction) function?

    (b) Suppose now that the employee decides to go out every night during the 31 days of July. Without limiting the amount he will spend. He will go to the bar, the nightclub, or the bar and the nightclub. Logically, consider that he will not be in and out of the bar or the disco on the same night.

    (c) How many times should he go to the bar or the disco in order to maximize his utility function $U(x, y) = xy$?

12. Suppose that each unit of capital (x) costs 150 and each unit of labor (y) costs 250. Knowing that total output is given by the function $P(x, y) = x^{1/4}.y^{3/4}$ and that you have a budget of \$50,000, what are the quantities x and y that maximize your output?

13. A manufacturer produces two types of machines: $x$ and $y$. Suppose the profit function is given by $L(x, y) = x^2 + 3xy - 6y$. How many units $x$ and $y$ must be produced if the manufacturer has enough resources to manufacture at most a total of 42 machines?

14. Use Lagrange multipliers to find expressions for $x$ and $y$ that maximize the output given by the Cobb-Douglas function $P(x, y) = K.x^{\alpha}.y^{\beta}$ where $K$, $\alpha$, and $\beta$ are positive constants. The function $P(x, y)$ is subject to the cost constraint $x.p_x + y.p_y = D$ where $p_x$, $p_y$ and $D$ are constants.

15. We have two points on a piece of land, point A and point B. At point A are the cattle. At point B is the stable. We have a river flowing by near points A and B. You can think of this river as a $y = f(x)$ Based on this data, the ranchers asked a mathematician to develop a method to determine which point on the river to take the cattle to drink water and then take them to the stable, so that the distance the cattle will travel is minimal.

\end{enumerate}

## 8.34 Linear Optimization Exercises Lagrange, Graph, Simplex, Solver

1. A factory produces two types of boxes, using the inputs *labor* and *metal*. To produce a type 1 box requires 10 man-hours and a type 2 box requires 2 man-hours. Each box of type 1 requires 10 sheets of metal, and each box of type 2 requires 5 sheets of metal. In a given period of time, the factory has 200 man-hours and 260 sheets of metal available. If each box of type 1 is sold for 200.00 and each box of type 2 is sold for 90.00, which production maximizes the sales revenue in this period?

2. A certain company manufactures two products P1 AND P2. The unit profit of product P1 is 1000 reais and the unit profit of P2 is 1800 reais. The company needs 20 hours to manufacture one unit of P1 and 30 hours to manufacture one unit of P2. The annual available production time is 1200 hours. The expected demand for each product is 40 annual units of P1 and 30 annual units of P2.

   (a) What will be the profit if the company decides to manufacture 10 units of product 1 and 20 units of product 2.

   (b) Is it feasible to manufacture such quantities?

   (c) How much should the company manufacture product P1 and product P2 to maximize its profit?

3. The marketing manager of the Ice Mountain soft drink needs to decide how many TV and magazine ads to run during the next quarter. Each TV spot costs 500, 000 and should increase sales by 300,000 cans. Each magazine ad costs R$2,000 and should increase sales by 500,000 cans. A total of 100, 000 can be spent on TV and magazine ads; however, Frozen Mountain wants to spend no more than 70, 000 on TV commercials and no more than 50, 000 on magazine ads. In addition, Ice Cream Mountain

earns a profit of R$0.05 on each can that is sold. It is asked on the basis of this information:

(a) Formulate an algebraic model for this problem.

(b) Sketch the solution region for this model.

(c) Find the optimal solution (quantities of TV and Magazine ads) for this problem using the graphical method, plus the corresponding maximum profit.

4. A company manufactures 2 types of bathtub, AquaSpa and HydroLux. Each AquaSpa bathtub needs 2 pumps, 12 hours of work and 16 meters of pipe, giving a unit profit of 400. Each HydroLux bath needs 1 pump, 8 hours of work and 12 meters of pipe, making 300 Reais profit per unit. There are 200 pumps, 1566h and 2880m of pipe in stock. It is asked in order to maximize the total profit:

   (a) Express the problem in algebraic form

   (b) Sketch the graph that would represent the solution to the problem

5. A furniture company can produce two types of table: table type A, which consumes 1 liter of paint, 9 hours of work and 12 meters of wood, or table type B which consumes 1 liter of paint, 6 hours of work and 16 meters of wood. The profit of the tables is 350 reais for type A and 300 reais for B. The company has available 200 liters of paint, 1566 working hours and 2880 meters of wood. Based on these data ask:

   (a) Which production should be aimed at the highest profit?

   (b) Which of the products in stock can be reduced without affecting the maximum profit?

6. A local television network has the following problem: It has been discovered that a program $A$ with 20 minutes of music and 1 minute of advertising attracts 30,000 viewers, while a program $B$, with 10 minutes of music and 1 minute of advertising draws the attention of 10,000 viewers. In the course of a week, the sponsor insists on using at least 5 minutes for its advertising, and that there is no budget for more than 80 minutes of music. How many times a week must each program be aired to reach the maximum number of viewers?

7. For a good diet, the body needs vitamins and proteins. The minimum requirement for vitamins is 32 units per day, and the minimum requirement for protein is 36 units per day. A person has available meat and eggs to eat. Each unit of meat contains 4 units of vitamins and 6 units of protein. Each unit of egg contains 3 units of vitamins and 6 units of protein. How much meat and eggs should be consumed each day to meet the vitamin and protein needs at the lowest possible cost. If each unit of meat costs

3 reals and each unit of egg costs 2.5 reals. Solve the problem using the graphical method.

8. A humanitarian organization proposes to carry out projects to improve agricultural production in two underdeveloped countries. These projects require the sending of tractors, specialists, and money. The organization has 20 tractors, 40 specialists, and 270 million dollars. The needs per project and country are as follows: a) Country A: 2 tractors, 2 specialists and 30 million dollars per project (i.e. each project needs 2 tractors, 2 specialists and 30 million dollars), b) Country B: 1 tractor, 3 specialists and 10 million dollars per project.

   (a) What is the maximum total number of projects that this organization can undertake?

   (b) It is known that for each project in country A, 6,000 people benefit, while projects in country B benefit 2,500 people each. What is the largest number of people that can benefit?

9. The company PC Tech assembles and then tests two models of computers, Basic and XP. For the next month, the company wants to decide how many models of each to assemble and then test. No computers are in inventory from the previous month, and since these models will change after this month, the company does not want to keep any inventory after this month. He believes that the most he can sell this month is 600 Basic and 1200 XPs. Each Basic sells for 300 and each XP sells for 450. The cost of parts for a Basic is 150; for an XP it costs 225. Labor is required for assembly and testing. There is a maximum of 10,000 assembly hours and 3000 test hours available. Each hour of assembly labor costs 11, and each hour of test labor costs 15. Each Basic requires five hours for assembly and one hour for test, and each XP requires six hours for assembly and two hours for test. PC Tech wants to know how many models each must produce (assemble and test) to maximize its net profit, but it cannot use more labor hours than are available and does not want to produce more than it can sell. You are asked to: a) formulate an algebraic model for this problem, b) sketch the solution region for this model, c) find the optimal solution to this problem using the graphical method (frontier point analysis).

10. A cosmetics company produces two types of perfume, Sense and Sensibility. The net profit obtained from the sale of each of the perfumes is 12.00 and 60.00 respectively. The manufacturing process involves the steps of blending, quality control, and packaging. The blending process requires 15 minutes for each unit of Sense, and 30 minutes for each unit of Sensibility. The quality control process requires 6 minutes for each Sense unit, and 45 minutes for each Sensibility unit. The packaging process requires 6 and 24 minutes respectively for each Sense and Sensibility unit produced. The time available per week for the mixing, quality control, and packaging processes is 36, 22, and 15 hours. The company wants to determine how

| Recurso: | Mold1 | Mold2 | Mold3 | Mold4 |
|---|---|---|---|---|
| Trabalho (horas) | 2 | 1 | 3 | 2 |
| Metal (kg) | 3 | 2 | 1 | 2 |
| Vidro (kg) | 6 | 2 | 1 | 2 |
| Preço Unit. | 28,50 | 12,50 | 29,25 | 21,50 |

Tabela 8.13: Company Data Monet Frames

many units it should produce of each of the perfumes to maximize its Profit, respecting the time constraints in the production process steps.

\end{enumerate}

## 8.35 Modeling Exercises and Managerial Applications

1. The Monet company produces four different types of picture frames. Each frame requires a certain amount of labor, metal, and glass, is sold for a specific price, and has a demand limit that is distinct from the others, as shown in the table 8.13. Monet can get up to 4,000 hours of labor for R$8.00 per hour, 6,000 kg of metal for $0.50 per kg, and 10,000 kg of glass for R$0.75 per kg. Determine how much to manufacture during the next week in order to maximize profit, without leaving frames left over in the company.

2. A refinery produces three types of gasoline: green, blue, and regular. Each type requires pure gasoline, octane, and additive that are available in quantities of 9,600,000, 4,800,000, and 2,200,000 liters per week, respectively. The specifications of each type are shown below. As a production rule, based on market demand, the refinery's planning has stipulated that the quantity of regular gasoline must be at least equal to 16 times the quantity of green gasoline and that the quantity of blue gasoline must be at most equal to 600,000 liters per week. The company knows that each liter of green, blue, and regular gasoline gives a profit contribution margin of $0.30, $0.25, and $0.20 respectively, and its objective is to determine the production schedule that maximizes the total profit contribution margin.

   (a) one liter of green gasoline requires 0.22 liter of pure gasoline, 0.50 liter of octane, and 0.28 liter of additive;

   (b) one liter of blue gasoline requires 0.52 liter of pure gasoline, 0.34 liter of octane and 0.14 liter of additive;

   (c) one liter of regular gasoline requires 0.74 liter of pure gasoline, 0.20 liter of octane and 0.06 liter of additive.

3. The company General Flakes advertises its product in a series of television

| Program: | Friends | MNF | Malcolm | Sports | TRL | Lifetime | CNN | JAG | Min. |
|---|---|---|---|---|---|---|---|---|---|
| H:18-35 | 6,0 | 6,0 | 5,0 | 0,5 | 0,7 | 0,1 | 0,1 | 1,0 | 60,0 |
| H:36-55 | 3,0 | 5,0 | 2,0 | 0,5 | 0,2 | 0,1 | 0,2 | 2,0 | 60,0 |
| H:$\geq$ 55 | 1,0 | 3,0 | 0,0 | 0,3 | 0,0 | 0,0 | 0,3 | 4,0 | 28,0 |
| F:18-35 | 9,0 | 1,0 | 4,0 | 0,1 | 0,9 | 0,6 | 0,1 | 1,0 | 60,0 |
| F:36-55 | 4,0 | 1,0 | 2,0 | 0,1 | 0,1 | 1,3 | 0,2 | 3,0 | 60,0 |
| F:$\geq$ 55 | 2,0 | 1,0 | 0,0 | 0,0 | 0,0 | 0,4 | 0,3 | 4,0 | 28,0 |
| Cost Un.: | $160 | $100 | $80 | $9 | $13 | $15 | $8 | $85 | |

Tabela 8.14: Company Data General Flakes

| Nutrients | Feed | Corn | Alfafa | Min.Necessary |
|---|---|---|---|---|
| Carbohidrate | 90 | 20 | 40 | 190 |
| Protein | 30 | 80 | 60 | 210 |
| Vitamin | 10 | 20 | 60 | 160 |
| Cost | 84 | 72 | 60 | |

Tabela 8.15: Feed Mixing Data

commercials, which vary in cost according to the program. Viewers were segmented into six categories. The cost (in $1,000s per ad) and the amount of viewers in each segment (in 1Ms of people) that an ad in a given program provides are given in the table 8.14 where `M=Men` and `M=Women`. The company wants to know how many ads it should place in each program to maximize total exposure, obtaining the desired minimum exposure in each viewer segment, with a maximum cost of $2.1 million.

4. A farmer is raising pigs commercially. He wants to determine the amounts of feed, corn, and alfalfa needed to feed the animals. The pigs will eat any mixtures of these three feeds, so the goal is to determine which mixture will contain the least amount of some nutrients for the least cost. The amount of each nutrient contained in one pound of each of the three foods, the minimum amounts needed daily, and the prices are summarized in table 8.15. (Hint: Suppose the farmer wants to obtain 1,000 kg of feed in total and get the percentages from there).

5. A chemical company manufactures a compound that is widely used in college and university chemistry laboratories. This compound must contain at least 20

   (a) Model the problem in a spreadsheet

   (b) Determine the best mixture that meets the requirements for the final product and has the lowest possible cost.

   (c) Enter the minimum cost obtained in the answer below.

6. A company has innovated in the sale of animal feed by preparing its pro-

| Compound | Sulfuric Acid | Iron Oxide | Potassium |
|---|---|---|---|
| 1 | 20% | 60% | 20% |
| 2 | 40% | 30% | 30% |
| 3 | 10% | 40% | 50% |

Tabela 8.16: Chemical Mixing Data

| Nutrient | feed 1 | feed 2 | feed 3 | feed 4 |
|---|---|---|---|---|
| Corn | 30% | 5% | 20% | 10% |
| Grain | 10% | 30% | 15% | 10% |
| Minerals | 20% | 20% | 20% | 30% |
| Cost/kg | $0.25 | $0.30 | $0.32 | $0.15 |

Tabela 8.17: Mix Feed Raw Material

ducts according to customer specifications, using among other components corn, grains and minerals. This "customization"has a great added value, because the appropriate feed for a particular animal changes regularly depending on weather conditions, pasture conditions, etc. The company stocks four types of feed that it can mix to meet customer specifications. Table 8.17 summarizes the composition of the four types and their cost. The company has received an order for 8,000 kilograms of feed from a local farmer. The farmer wants the feed to contain at least 20

7. Sucofresco is a company that grows oranges and prepares juice for commercialization. Its plantations are in three different cities, with 275,000 bushels in Laranjal, 400,000 bushels in Rio Claro, and 300,000 bushels in Santa Clara. Sucofresco has orange processing plants in three locations, the São Carlos plant has the capacity to process the equivalent of 200,000 bushels, Jandira 600,000, and Santa Rita 225,000. It hires a local company to transport the orange from the orchard to the factories, which charges a fixed rate per kilometer transported of the equivalent hectare of orange. The table 8.18 summarizes the distances from the orchards to the factories. Sucofresco wants to determine how many bushels it will send from each orchard to each factory in order to minimize the transportation cost. To do this we ask you to assemble the problem in graphical form, draw the solving spreadsheet, present the solver parameters, and the numerical solution.

| Pomar | São Carlos | Jandira | Santa Rita |
|---|---|---|---|
| Laranjal | 21 | 50 | 40 |
| Rio Claro | 35 | 30 | 22 |
| Santa Clara | 55 | 20 | 25 |

Tabela 8.18: Distances between Farms and Mills

# Capítulo 9

# Integrals

## 9.1 Algebraic Integration

Algebraic integration can be performed in Python via the `sympy` package. Below is calculated the integral of the function $sin(x)$ over the interval $[0, \pi]$

```
import sympy as sp
x = sp.symbols("x")
sp.integrate(sp.sin(x),[x,0,sp.pi])
```

```
2
```

## 9.2 Rule of Trapezoids

Next we calculate the same value using the trapezoids rule by dividing the integration interval into $n = 2000$ subintervals.

```
import numpy as np
def fL(x):
  return(np.sin(x))

n = 2000

x0 = 0
x1 = np.pi
dx = (x1 - x0)/n
x = np.linspace(x0,x1,n)

integral = 0
for xi in x:
```

```
  integral = integral + (fL(xi+dx) + fL(xi))/2*dx

integral
```

```
1.998998355477553
```

By way of comparison we perform the same calculation for $\int(\frac{1}{x}dx$ on the interval $[1, 2]$ whose exact result (obtained by algebraic integration) we know to be $ln(2)$.

```
import sympy as sp
x = sp.symbols("x")
sp.integrate(1/x,[x,1,2]), \
   sp.integrate(1/x,[x,1,2]).evalf()
```

```
(log(2), 0.693147180559945)
```

```
import numpy as np
def fL(x):
  return(1/x)

n = 20000

x0 = 1
x1 = 2
dx = (x1 - x0)/n
x = np.linspace(x0,x1,n)

integral = 0
for xi in x:
  integral = integral + (fL(xi+dx) + fL(xi))/2*dx

integral
```

```
0.6931375236696611
```

## 9.3 Gaussian 2-point squaring

Another technique of numerical integration, especially for continuous functions in the interval $[-1, +1]$ with excellent results and extremely fast to calculate, is Gaussian quadrature. Below we calculate its parameters for a 2-point interpolation.

```
import sympy as sp
x = sp.symbols("x")
I1 = sp.integrate(1,[x,-1,1]); I1
```

```
2
```

```
I2 = sp.integrate(x,[x,-1,1]); I2
```

```
0
```

```
I3 = sp.integrate(x**2,[x,-1,1]); I3
```

```
2/3
```

```
I4 = sp.integrate(x**3,[x,-1,1]); I4
```

```
0
```

```
a, x0, b, x1 = sp.symbols("a x0 b x1")
Eq0 = a*1 + b*1 - I1; Eq0
```

```
a + b - 2
```

```
Eq1 = a*x0 + b*x1 - I2; Eq1
```

```
a*x0 + b*x1
```

```
Eq2 = a*x0**2 + b*x1**2 -I3; Eq2
```

```
a*x0**2 + b*x1**2 - 2/3
```

```
Eq3 = a*x0**3 + b*x1**3 - I4; Eq3
```

```
a*x0**3 + b*x1**3
```

```
sols = sp.solve([Eq0, Eq1, Eq2, Eq3],[a,x0,b,x1]);
for sol in sols:
  print(sol)
```

```
(1, -sqrt(3)/3, 1, sqrt(3)/3)
(1, sqrt(3)/3, 1, -sqrt(3)/3)
```

Exact solution for $\int_0^{\pi} sin(x)dx = 2$

```
import numpy as np
sp.integrate(sp.sin(x),[x,0,sp.pi])
```

```
2
```

Solution by 2-point Gaussian quadrature

```
x, z = sp.symbols("x z")
A = sp.Matrix([[x,z,1],[0,-1,1],[sp.pi,1,1]]); A
```

```
Matrix([
[ x,  z, 1],
[ 0, -1, 1],
[pi,  1, 1]])
```

```
g = sp.solve(sp.det(A),x)[0]; g
```

```
pi*(z + 1)/2
```

```
g.diff(z) #dx/dz
```

```
pi/2
```

```
c1 = sols[0][0]; print(c1)
```

```
1
```

```
p1 = sols[0][1]; print(p1)
```

```
-sqrt(3)/3
```

```
c2 = sols[0][0]; print(c2)
```

```
1
```

```
p2 = sols[0][3]; print(p2)
```

```
sqrt(3)/3
```

```
aprox = c1*sp.sin(g.subs(z,p1))*g.diff(z)
aprox = aprox + c2*sp.sin(g.subs(z,p2))*g.diff(z)
aprox.evalf()
```

```
1.93581957465114
```

## 9.4 Three-Point Gaussian Quadrature

```
import sympy as sp
x = sp.symbols("x")
I0 = sp.integrate(1,[x,-1,1]); print(I0)
```

```
2
```

```
I1 = sp.integrate(x,[x,-1,1]); print(I1)
```

```
0
```

```
I2 = sp.integrate(x**2,[x,-1,1]); print(I2)
```

```
2/3
```

```
I3 = sp.integrate(x**3,[x,-1,1]); print(I3)
```

```
0
```

```
I4 = sp.integrate(x**4,[x,-1,1]); print(I4)
```

```
2/5
```

```
I5 = sp.integrate(x**5,[x,-1,1]); print(I5)
```

```
0
```

```
a, x0, b, x1, c, x2 = sp.symbols("a x0 b x1 c x2")
Eq0 = a + b + c - I0; print(Eq0)
```

```
a + b + c - 2
```

```
Eq1 = a*x0 + b*x1 + c*x2 - I1; print(Eq1)
```

```
a*x0 + b*x1 + c*x2
```

```
Eq2 = a*x0**2 + b*x1**2 + c*x2**2 - I2; print(Eq2)
```

```
a*x0**2 + b*x1**2 + c*x2**2 - 2/3
```

```
Eq3 = a*x0**3 + b*x1**3 + c*x2**3 - I3; print(Eq3)
```

```
a*x0**3 + b*x1**3 + c*x2**3
```

```
Eq4 = a*x0**4 + b*x1**4 + c*x2**4 - I4; print(Eq4)
```

```
a*x0**4 + b*x1**4 + c*x2**4 - 2/5
```

```
Eq5 = a*x0**5 + b*x1**5 + c*x2**5 - I5; print(Eq5)
```

```
a*x0**5 + b*x1**5 + c*x2**5
```

```
sols = sp.solve([Eq0, Eq1, Eq2, Eq3, Eq4, Eq5],
                [a,x0,b,x1,c,x2])
for sol in sols:
  print(sol)
```

```
(5/9, -sqrt(15)/5, 5/9, sqrt(15)/5, 8/9, 0)
(5/9, -sqrt(15)/5, 8/9, 0, 5/9, sqrt(15)/5)
(5/9, sqrt(15)/5, 5/9, -sqrt(15)/5, 8/9, 0)
(5/9, sqrt(15)/5, 8/9, 0, 5/9, -sqrt(15)/5)
(8/9, 0, 5/9, -sqrt(15)/5, 5/9, sqrt(15)/5)
(8/9, 0, 5/9, sqrt(15)/5, 5/9, -sqrt(15)/5)
```

```
from IPython import display
from IPython.display import Math, HTML
Math(sp.latex(sols))
```

```
<IPython.core.display.Math object>
```

```
c1 = sols[0][0]; print(c1)
```

```
5/9
```

```
p1 = sols[0][1]; print(p1)
```

```
-sqrt(15)/5
```

```
c2 = sols[0][2]; print(c2)
```

```
5/9
```

```
p2 = sols[0][3]; print(p2)
```

```
sqrt(15)/5
```

```
c3 = sols[0][4]; print(c3)
```

```
8/9
```

```
p3 = sols[0][5]; print(p3)
```

```
0
```

Exact

```
import numpy as np
sp.integrate(sp.sin(x),[x,0,np.pi])
```

```
2.00000000000000
```

By quadrature

```
x, z = sp.symbols("x z")
A = sp.Matrix([[x,z,1],[0,-1,1],[sp.pi,1,1]]); A
```

```
Matrix([
[ x,  z, 1],
[ 0, -1, 1],
[pi,  1, 1]])
```

```
g = sp.solve(sp.det(A),x)[0]; g
```

```
pi*(z + 1)/2
```

```
g.diff(z) #dx/dz
```

```
pi/2
```

```
((c1*sp.sin(g).subs(z,p1) + \
  c2*sp.sin(g).subs(z,p2) + \
  c3*sp.sin(g).subs(z,p3))*g.diff(z)).evalf()
```

```
2.00138891360774
```

## 9.5 Exercises

### 9.5.1 Indefinite Integrals

1. $\int 9x^8\,dx$
2. $\int 3x\,dx$
3. $\int e^{-3x}\,dx$
4. $\int 3dx$
5. $\int -4x\,dx$
6. $\int \frac{dx}{x^2}$
7. $\int x^2\cos(x)\,dx$
8. $\int xe^{-x^2}\,dx$
9. $\int \frac{dx}{(x-1)(x+2)(x-3)}$
10. $\int e^x\sin(x)\,dx$
11. $\int \frac{x^3-5x^2+3x+\ln(x)}{x}\,dx$
12. $\int \frac{\ln(4x)}{\sqrt{x}}\,dx$
13. $\int \frac{x}{\sqrt{x^2+1}+x}\,dx$
14. $\int_0^1 t.e^{-t}\,dt$
15. $\int \frac{dx}{x^2+7}$
16. $\int xsin(x)\cos(x)\,dx$
17. $\int \ln(x+a)\,dx$
18. $\int \frac{\ln(4x)}{\sqrt{x}}\,dx$
19. $\int \frac{x^3-5x^2+3x+\ln(x)}{x}\,dx$
20. $\int \frac{dx}{x^2\sqrt{4-x^2}}$
21. $\int x\arcsin(x)\,dx$

### 9.5.2 Defined Integrals

1. Calculate the value of the integral of $f(x) = x^3$ for x ranging from 2 to 5.
2. Calculate the value of the integral of $f(x) = e^{-3x}$ for x ranging from 0 through 3.

3. There is a line passing through the origin that divides the region bounded by the parabola $y = x - x^2$ and the axis $x$ into two regions of equal areas. What is the slope of this straight line?
4. Prove by integration that the area of a circle is $\pi.R^2$
5. Prove by integration that the volume of a sphere is $\frac{4\pi R^3}{3}$
6. Prove by integration that the volume of a cone of height $h$ and radius of base $R$ is given by $V = \frac{\pi R^2 h}{3}$
7. Calculate the volume that lies between the solids formed by rotating the curves $y = x^2$ and $y = \sqrt{x}$
8. Calculate the volume formed by rotating the curve $y = \sin^2(x)$ from $x = 0$ to $x = \pi$ around the $x$ axis.
9. Find the area of the region between the curves $y = x^2$ and $y = \sqrt{x}$
10. Determine the area of the region bounded by the curve $y = \sqrt{x}$ and the straight lines $x = 4$ and $y = \frac{-x}{4}$.

### 9.5.3 Numerical Integration

1. Calculate $\int_1^3 \frac{1}{t}\,dt$ by the algebraic method and by the numerical method (in this case with h=0.1) and compare the two results, giving the percentage difference between the two. Hints: $\frac{dln(t)}{dt} = \frac{1}{t}$ and Dif% between A and B is equal to: (A-B)/B, where B is the exact value and A is the approximate value.
2. Let $g(x)$ be the antiderivative of $f(x) = x + 5 - 2\sqrt{x^2+3}$. Such that $g(0) = -4$ (here a hint fits: when $x$ is large (positive or negative), then $\sqrt{x^2+3} \cong |x|$). Find the values of x for the critical points of $g(x)$. Use the test of the first or second derivative to show whether the points found are minimums, maximums, or neither. (Clearly indicate which test you used. Find the values of x for the inflection points of $g(x)$. Indicate the concavity in the intervals

### 9.5.4 Applications in Economics

1. It was found in 1940 that the population density, within r kms of downtown New York City, was approximately 120e^(-0.2r) thousand people per square km. Estimate the population that lived in an area between 1km and 2km from downtown New York City.
2. The rate of world water consumption t years after 1960 was approximately $860,e^{0.04t}$ $km^3$ per year. How much water was used between 1960 and 1995?

3. The United States has been consuming iron ore at a rate of R'(t) million tons per year at time t, where t=0 corresponds to 1980. Knowing that R'(t) = $94e^{0.016t}$, calculate the total value of gas consumed between 1985 and 1995 using Excel. Calculate this same value algebraically and compare the two results.

4. Since 1987 the rate of natural gas production in the United States has been approx. R'(t) quadrillion British thermal units per year at time t, where t=0, corresponding to 1987, and R'(t) = $17.04e^{0.016t}$. Based on these data, calculate the amount consumed between 1987 and 1992.

5. Suppose the rate of oil consumption with the 1974 crisis is given by the formula $R\_1(t) = 21.3e^{0.039(t-4)}$ billion barrels/year for t textgreater =4 where t = 4 corresponds to 1974. If the consumption rate without the crisis was $R\_0(t) = 17.2e^{0.074t}$ billion barrels/year for $t \geq 0$ where $t = 0$ corresponds to 1970, calculate the total amount of oil that was not consumed between 1976 and 1980.

6. The rate of change of US oil consumption in the 1980s (in billions of barrels per year) can be represented by the function: $C(t) = 27.08e^{\frac{t}{25}}$. Where $t$ is the number of years after January 1, 1980. That being so determine

   (a) The total consumption of oil in the US from January 1, 1980 to January 1, 1990.

   (b) Knowing that the rate of change in US oil production in the 1980s obeyed the following function: $P(t) = \ln(10t) + 25$. and that the US imported all of its oil production deficit, determine how many barrels of oil the US had to import in the 1980s. TIP: Use definite integrals

7. A company's marginal cost is $0.019x^2 - 2x + 79$ where x is the number of units produced in a day. The company has fixed costs of $1100 per day. It is asked:

   (a) Determine the cost to produce x units per day

   (b) Suppose the current production level is x=35. Determine by how much the costs would increase if the production level were raised to x=47

8. A small tie store comes to the conclusion that when the sales level is x ties per day, its marginal profit is $MP(x)$ dollars per tie, where $MP(x) = 1.30 + 0.06x - 0.0018x^2$ In addition the store will lose 95 per day if the production level is 0 ties. Find the profit earned from operating the store when the sales level is $x$ ties per day

9. A soap producer estimates his marginal cost of producing soap powder to be $0.2x + 1$ hundred dollars per ton produced when the production level is $x$ tons per day. The fixed costs are 200 per day. Find the cost of producing 20 tons of soap powder per day.

10. Suppose that the process of drilling an oil well has a fixed cost of US$10,000 and a marginal cost of C'(x) $= 1000 + 50x$ per meter, where $x$ is the depth in meters. Find an expression for $C(x)$, the total cost to drill x meters. (Note $C(0) = 10,000$)

11. The marginal profit of a certain company and $MP1(x) = -x^2 + 14x - 24$. The company expects the daily production level to increase from $x = 6$ to $x = 8$ units. Management is considering a plan that should have the effect of changing the marginal profit to $M2(x) = -x^2 + 12x - 20$. Should the company adopt this plan or not? Determine the area between the graphs of the two marginal profit functions from $x = 6$ to $x = 8$. Interpret this area in the context of economics

12. The world consumption of cigarettes (in trillions of cigarettes per year) since 1960 is approximately given by the function $c(t) = 0.1t + 2.4$ where $t = 0$ corresponds to 1960. Determine the number of cigarettes sold from 1980 to 1998.

13. Forest cutting is one of the biggest problems faced in southern Sahara, Africa. Even though the main reason for cutting down forests has been to establish farms, the growing demand for charcoal is becoming an important factor. The rate of charcoal consumption ( in millions of cubic meters per year ) in Sudan t years after 1980 is approximately given by the function $c(t) = 76.2e^{0.03t}$. Given that the growth rate of new trees ( in millions of cubic meters per year ) in Sudan t years after 1980 is approximately given by the function $g(t) = 50 - 6.03e^{0.09t}$ , obtain the value of net destroyed forest, that is taking into account reforestation and deforestation in the country.

14. The administrative area of a municipality needs to calculate the IPTU of a plot of land of area R bounded by the graphs of the functions $f(x) = x^2 - 2x - 1$ e $g(x) = -e^x - 1$ and by the vertical lines $x = -1$ e $x = 1$

    (a) Using set integral techniques, calculate the area of the plot R (in thousands of $m^2$).

    (b) The factory that is installed in this area R will produce a specific good that has the following cost function $C(x) = 100x^2 - 100x + IPTU$, considering that the IPTU for the area costs R$ 100 per thousand square meters, calculate the output $x$ that minimizes $C(x)$ and the final cost for this output.

### 9.5.5 Statistical Applications

1. Knowing that the probability that a normalized variable (z) lies between $a$ and $b$ is given by the expression: $P(a < z < b) = \int_a^b \frac{1}{\sqrt{2\pi}} e^{\frac{-z^2}{2}} dz$, and that z = (x-Mean)/Standard Deviation, calculate the probability that a variable x of mean 3.5 and standard deviation 0.9 is situated between 4

and 5.

# Capítulo 10

# Ordinary Differential Equations

Solving differential equations numerically gives values for the solution function by computing it point by point from any initial value. So the basic structure of the solving process is: given a differential equation $y' = f(t, y)$ and an initial value $y(t_0)$ for the solution function $y(t)$, we are asked to determine the value of $y(t)$ when $t = t_1$. There are several numerical methods to solve such a problem. If the solution function is a function of a single variable the differential equation is called an ordinary differential equation and is abbreviated **EDO**. We will cover some examples of numerical methods for solving ODEs below.

## 10.1 Euler's Method

Modern science began to have an effect on people's everyday lives when man became able to predict what would happen in the real world from highly accurate theoretical models. Since in the world, anything you want to measure varies over time, the preferred model of researchers is one that predicts what will happen in the future by determining how our variable of interest changes its value. This however is much simpler than it seems to be stated. What we will say is only that the future value of something we are measuring will be equal to its current value plus a variation. In turn, this variation will be given by the product of the rate or its rate of change times the time it has spent varying at this rate.

Let's call the current value of our measure, for example the number of people in a certain region $f(t)$. This symbol simply means that at time $t$ the number of people was $f(t)$. To find $f(t)$ from $t$ we usually have to perform some mathematical operations. For example if we say that $f(t) = t^2 + 2$, this indicates that when $t = 3$, $f(t)$ will be equal to 11. On the other hand if we say $f(t + h)$ this means that we will have to calculate $(t + h)^2 + 2$.

Now suppose you don't know the mathematical form of $f(t)$ but you do know the form of $f'(t)$ that is its rate of change in addition to the numerical value of $f(t)$ at some instant of time, for example at $t = 0$ and you want to calculate the value of $f(t)$ at a near instant, but in the future, say 0.1 time units ahead. So you make $f(t + h)$ i.e. $f(0 + 0.1)$ approximately equal to its current value, i.e. $f(0)$ plus its rate of change at the current time $f'(t)$ times the time you go into the future $h$ i.e. 0.1 time units. In mathematical language we simply write :

$$f(t + h) \approx f(t) + f'(t).h$$

$$f(0 + 0.1) = f(0.1) \approx f(0) + f'(0).0.1$$

$

What if we want to calculate the value of the function at $t = 0.2$ how to proceed? Consider the following, we now have the value of $f(0, 1)$ and a mathematical way to calculate $f'(0, 1)$. With this we reapply the reasoning described above and make :

$$f(t + h) \approx f(t) + f'(t).h$$

$$f(0, 1 + 0, 1) = f(0, 2) \approx f(0, 1) + f'(0, 1).0, 1$$

$

This reasoning can be applied as many times as we like (as long as $h$ is small, something like 0.1 in most cases) and so we can calculate the value of $f(t)$ at any future instant. For example, if we want to calculate $f(t)$ at $t = 1$ we continue the previous process:

$$f(0, 2 + 0, 1) = f(0, 3) \approx f(0, 2) + f'(0, 2).0, 1$$

$$f(0.3 + 0.1) = f(0.4) \approx f(0.3) + f'(0.3).0.1$$

$

$$f(0.4 + 0.1) = f(0.5) \approx f(0.4) + f'(0.4).0.1$$

$

$$f(0.5 + 0.1) = f(0.6) \approx f(0.5) + f'(0.5).0.1$$

$

$$f(0.6 + 0.1) = f(0.7) \approx f(0.6) + f'(0.6).0.1$$

$$

$$f(0.7 + 0.1) = f(0.8) \approx f(0.7) + f'(0.7).0.1$$

$$

$$f(0.8 + 0.1) = f(0.9) \approx f(0.8) + f'(0.8).0.1$$

$$

and finally :

$$f(0,9 + 0,1) = f(1,0) \approx f(0,9) + f'(0,9).0,1$$

Notice that the process is very simple, yet tedious and repetitive. This is where fast function implementation software like Python comes in. With Python we can repeat these calculations with a minimum of effort and concentrate our time on modeling the problem, that is on determining from the problem statement, the form of the rate of change of $f(t)$, that is $f'(t)$.

1. Population Growth. Suppose that in a given region, the amount of people $P(t)$ at time $t = 0$ is $1,000$ individuals, i.e. $P(0) = 1000$ and its rate of change is 0.03 of the current population per year, i.e. $P'(t) = 3/100.P(t)$. Based on these data estimate the population in year $t = 20$. Consider $h = 1/10$ as the value of the forward step.

The Python listing solving this problem is shown below, the Pandas data frame with the results can be seen at **??** and the resulting graph can be seen at 10.1. The result is 1,820 people.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

def euler(xi, yi, xf, h = 0.1, draw=True):
  yL = lambda y0:0.03*y0
  x = [xi]; y = [yi]
  while x[-1] <= xf:
    x0 = x[-1]; x1 = x0 + h; x.append(x1)
    y0 = y[-1]; yL0 = yL(y0); y1 = y0 + yL0*h
    y.append(y1)
  data = pd.DataFrame({'x':x, 'y':y})
  if draw:
    g = data.plot(x='x', y='y')
    plt.show();
  return(data)

data = euler(0,1000,20,0.1)
```

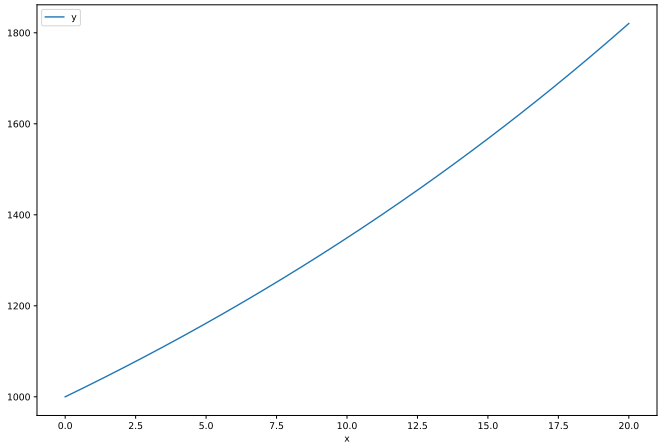



Figura 10.1: Graph of the Population Growth Problem

```
tail(py$data)
```

```
        x         y
196  19.5 1793.420
197  19.6 1798.800
198  19.7 1804.196
199  19.8 1809.609
200  19.9 1815.038
201  20.0 1820.483
```

2. Population growth with limit

The previous example has a deficiency in that the population can grow without a limit, which does not occur in practice. In reality, when approaching an ideal value and then exceeding it, the growth rate falls and then starts to assume negative values, causing the population to fall.

The first to think about this issue was Malthus when he realized that food production grows at a linear rate (in the absence of significant technological changes), which is proportional to the amount of area planted, and the population grows at an exponential rate, which is proportional to its current value (in the absence of effective birth control mechanisms).

In this example the rate of population change over time is given by $y' = a * (M - y).y$. Notice that the graph of the derivative is now a parabola. So, assuming that at t=0, y is small but positive, the derivative will also be positive and small, i.e., the function will grow slowly. As time passes, and y grows, the

derivative $y'$ will reach a maximum value (when the function y will have its maximum slope).

Then the derivative $y'$ will still be positive but smaller and smaller, i.e. y will have a smaller and smaller slope, until for a given instant of time $y'$ will reach zero and y will be horizontal.

So the function $y(t)$ satisfying the equation $y' = a * (M - y).y$ is a function with exponential growth for small values of $y$ and limited growth after a certain value of $y$.

```
import numpy as np
import pandas as pd

M = 10000
a = 0.03
y = np.linspace(0,M,10000)
yL = a*(M-y)*y

from matplotlib import pyplot as plt
g = plt.plot(y,yL);
plt.show()
```

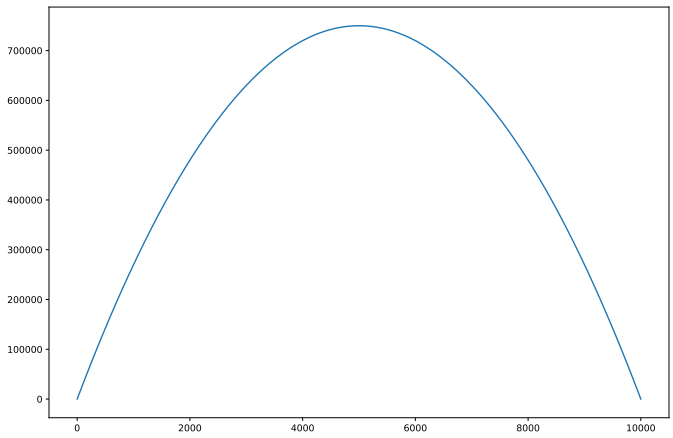



Figura 10.2: Graph of y' vs. y in the Logistic Function

The development of the algebraic solution is shown below:

$$\frac{dy}{dt} = a(M - y)y$$

$$\frac{dy}{(M-y)y} = adt$$

$$\frac{1}{M(M-P)} + \frac{1}{MP}dp = adt$$

$$\frac{dp}{(M-P)} + \frac{dp}{P} = aMdt$$

$$-ln(M-y) + ln(y) = aMt + C$$

$$ln(\frac{y}{M-y}) = aMt + C$$

$$\frac{y}{M-y} = e^{aMt+C}$$

$$y = \frac{M}{1+Ce^{-aMt}}$$

Applying the initial condition $y(0) = 100$ we have:

$$y_0 = \frac{M}{1+C}$$

$$C = \frac{M-y_0}{y_0}$$

which leads us to:

$$y = \frac{M}{1+\frac{M-y_0}{y_0}e^{-aMt}}$$

In figure 10.3 and **??** can be seen the graphs and table of $y(t)$ obtained using Euler's method and the exact solution function, for $M = 10,000$, $y(0) = 100$, $a = 5.10^{-6}$ and $t = [0, 200]$. As can be seen, Euler's method shows great agreement with the exact solution in this case.

```
import numpy as np

def euler(xi, yi, xf, h, draw=True):
  M = 10000; a = 5*10**(-6); y0 = 100.0; C = (M-y0)/y0
  yL = lambda x: a*(M-x)*x
  x = [xi]; y = [yi];
  while x[-1] ≤ xf:
    x.append(x[-1]+h)
    y.append(y[-1] + yL(y[-1])*h)
  data = pd.DataFrame({'x':x, 'y':y})
  f = lambda x: M/(1+C*np.exp(-a*M*x))
  data['yexata'] = list(map(f,data['x'].values))
  if draw:
    g = data.plot(x='x', y=['y','yexata'])
```

```
    plt.show()
  return(data)

data = euler(0,100,200,0.1)
```

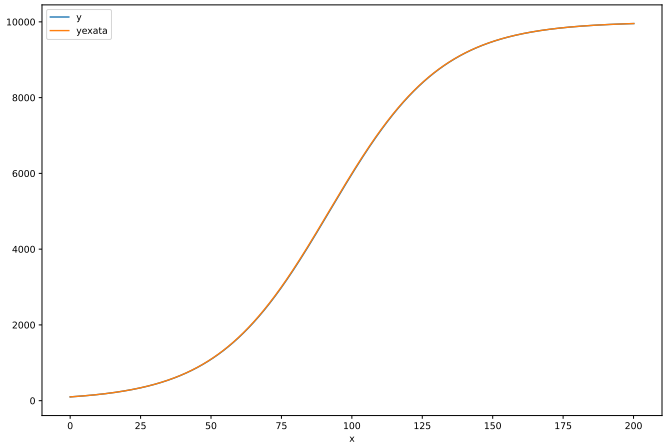



Figura 10.3: Graph of the Solution of the Logistic Equation by Euler's Method

What are the limitations of Euler's method? The main one is related to the step size $h$. For the method to work properly the step size must be small enough. But this can in some cases lead to an increase in rounding error, due to the large number of operations needed to obtain a result. On the other hand, very large step sizes can lead to instability of the method. Therefore, for the use of the Euler method a compromise between the required precision and the stability of the calculation process.

3. Present the solution plots of the logistic equation for the equal step values $h$ of 1, 35, and 50.

With $h = 1$ we have a large agreement of the numerical solution with the exact solution. At $h = 35$ the solution starts to oscillate and becomes totally distinct from the correct result. This oscillation occurs due to the distance $h$ is being used in each iteration. After a certain value everything happens as if from one instant to the next the population had exceeded the limit value, therefore the derivative becomes negative, but as the step is too long the function becomes too small, requiring another radical value adjustment. This process as can be seen, may or may not be corrected over time. If the step is still smaller than a threshold value, the oscillation is slowly eliminated, but if the step is large enough, this can lead the solution to a permanent oscillation.

```
from matplotlib import pyplot as plt
fig, ax = plt.subplots()

hs = [1, 35, 50]
for i, h in enumerate(hs):
  data = euler(0,100,350,h,False)
  data.rename({'y':'h'+str(h)}, axis=1, inplace=True)
  if i == 1:
    ax = plt.plot(data['x'].values, \
                  data['yexata'].values, \
                  label = 'exact');
  ax = plt.plot(data['x'].values, \
                data['h'+str(h)].values, \
                label = 'h'+str(h));

g = plt.legend();
plt.show()
```

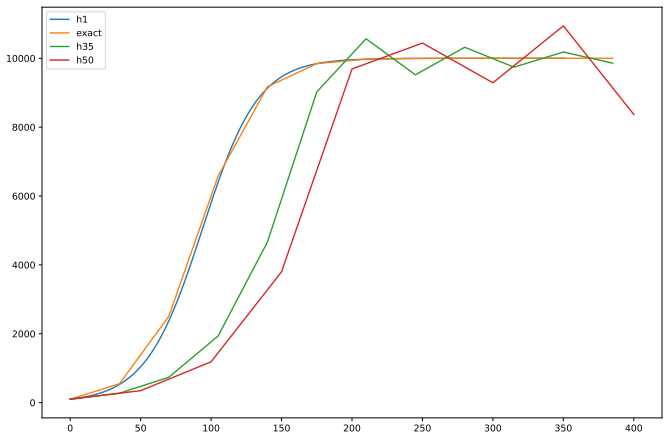



Figura 10.4: Solutions of the Logistic Equation for Different Steps h

4. Oscillation of a Mass-Spring System

In this problem we will analyze the case of a mass-spring system, horizontal, with dynamic friction, governed by the differential equation :

$$m\frac{d^2x}{dt^2} = -kx - \mu\frac{dx}{dt} \tag{10.1}$$
$$x(0) = x_0 \tag{10.2}$$
$$x'(0) = v_0 \tag{10.3}$$

Where $m$ is the mass of the spring, $k$ is the elastic constant of the spring, $x_0$ the initial displacement of the system, $v_0$ the initial velocity, and $t$ the independent variable. The goal of this problem is to plot the graph of the $x$ coordinate of the center of mass of the object suspended by the spring over time, considering different combinations of mass, elastic constant, and initial displacement.

To solve this problem numerically in Python, we transform the second order linear ordinary differential equation above into a system of first order linear ordinary differential equations by making :

$$v' = -\frac{k}{m}x - \frac{mu}{m}v \tag{10.4}$$
$$x' = v \tag{10.5}$$

We then solve the system using the Euler method for the following set of parameters :

$$\mu = 0.28\ k = 0.1\ m = 6kg\ x(0) = 5m\ v(0) = -10m/s\ h = 0.05$$

Recall that Euler's method solves a differential equation by calculating the value of the point-to-point function through the following relationship : $f(t+h) \approx f(t) + f'(t).h$ . In other words, the method states that the value of f(t) at point t+h is obtained from the current value of f(t) and the current value of f'(t) by calculating f(t+h) from the above relation.

The data frame with the results can be seen at **??** and the graph corresponding to the value of the position on the horizontal axis over time can be seen at 10.5.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

vL = lambda t,x,v : -(0.1*x+0.28*v)/6
xL = lambda t,x,v : v

t,x,v = 0, 5, -10
tf, dt = 100, 0.1
```

```
sol = []

while t<tf:
  sol.append([t,x,v])
  t, x, v = t + dt, x + xL(t,x,v,)*dt, v + vL(t,x,v)*dt
sol = pd.DataFrame(data = sol, columns = ['t','x','v'])

graf = sol.plot(x='t',y=['x','v'])
plt.show()
```

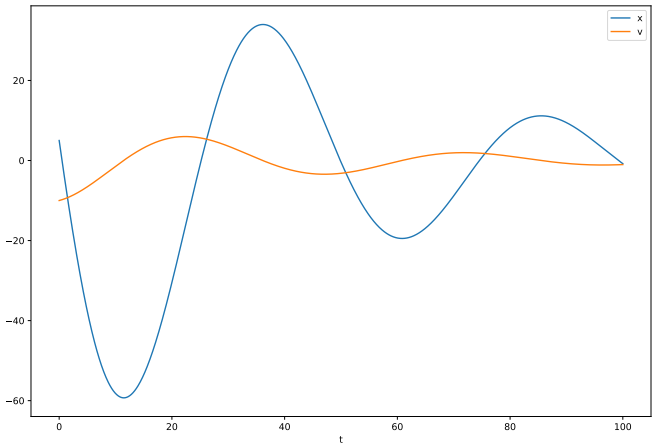



Figura 10.5: Resolution of the Spring Mass Oscillation System

5. Limitations of Euler's method: increasing amplitude when $\mu = 0$ (damping equal to 0)

By making $m = 6$ and $\mu = 0$ (which is equivalent to making the damping zero) we obtain the solution below. As can be seen the resulting graph of position over time has increasing amplitude, meaning that the system would be gaining energy, which is clearly not the case.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

vL = lambda t,x,v : -(0.1*x+0.0*v)/6
xL = lambda t,x,v : v

t,x,v = 0, 5, -10
```

```
tf, dt = 1000, 0.1
sol = []

while t<tf:
  sol.append([t,x,v])
  t, x, v = t + dt, x + xL(t,x,v,)*dt, v + vL(t,x,v)*dt
sol = pd.DataFrame(data = sol, columns = ['t','x','v'])

graf = sol.plot(x='t',y=['x','v']);
```

```
/home/gustavo/miniconda3/envs/deeplearn/lib/python3.9/site-packages/pandas/p
  fig = self.plt.figure(figsize=self.figsize)
```

```
plt.show()
```

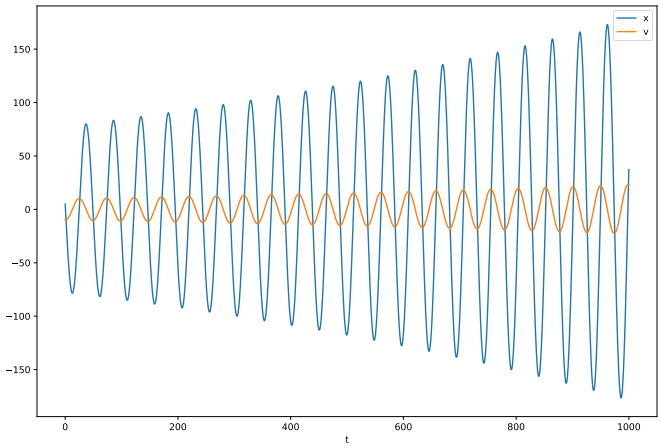



Figura 10.6: Amplitude of oscillation increasing when $\mu = 0$

What happens in this case is that Euler's method produces results at first with a small error. However, the recursive nature of the method (remember that to calculate the value at f(t=3) we must first calculate the value at f(t=3-h) and so on) leads to an accumulation of errors. At first glance, the solution would be to use a much smaller value for the step h, as this would decrease the error in each calculation. However this approach has two drawbacks : $1^o$) it increases the amount of operations needed, which increases the time and processing power required and $2^o$) by increasing the amount of operations needed, the rounding error inherent to operations done on a computer comes into play. This limitation

will be covered in more detail in the next sections when the most widely used method for solving differential equations numerically, called the Runge-Kutta Method of $4^a$ order, is presented.

## 10.2 Runge-Kutta Method of 4! order

In the example 10.1 (Logistic Population Growth) it was shown that Euler's method has limitations concerning step size. Another limitation arose in the example 5, when the damping $\mu$ was set equal to zero. To minimize such problems, there is another more accurate method, called the Runge-Kutta Method of $4^a$ order. This method actually belongs to a family of methods, all named Runge-Kutta Methods after the German mathematicians who developed such a set of techniques in the early 20th century. The most famous (because it has the greatest applicability) is the 4th order method that we describe below. In this method the value of $f(t+h)$ is calculated from the following set of operations :

$$\begin{aligned} y' &= f(t,y) \\ k_1 &= f(t,y) \\ k_2 &= f(x+\frac{h}{2}; y+\frac{h}{2}.k_1.h) \\ k_3 &= f(x+\frac{h}{2}.h; y+\frac{h}{2}.k_2.h) \\ k_4 &= f(x+h; y+k_3.h) \\ y(t+h) &= y(t)+\frac{(k_1+2k_2+2k_3+k_4)}{6} \end{aligned}$$

Let's exemplify this method by calculating the solution in the case of population growth with limit, example 10.1.

6. Population growth with limit, solving by the Runge-Kutta method of $4^a$ order.

In this case we will be solving the differential equation :

$$y(t) = \frac{M}{1+ke^{-aMt}} \tag{10.6}$$

$$k = \frac{M-y_0}{y_0} \tag{10.7}$$

For the following set of parameters and conditions :

$$M = 200.000 \tag{10.8}$$
$$y_0 = 1.000 \tag{10.9}$$
$$a = 5.10^{-6} \tag{10.10}$$

Next in 10.7, for a solution step $\Delta t = 50$ three solution curves are presented: exact, Runge-Kutta and Euler. Note that although it has a reasonable error in the middle part, the Runge-Kutta method still converges to the correct boundary value, while the Euler method diverges.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

a = 5e-06; M = 10000; p0 = 100
pL = lambda t,p : a*(M-p)*p
euler = lambda t,p,dt : [t+dt, p+pL(t,p)*dt]
exact = lambda t : M / (1 + (M-p0)/p0 * np.exp(-a*M*t))

def rk4(t,p,dt):
  k1 = pL(t,p)
  k2 = pL(t+dt/2, p+k1*dt/2)
  k3 = pL(t+dt/2, p+k2*dt/2)
  k4 = pL(t+dt, p+k3*dt)
  p = p + dt/6*(k1+2*k2+2*k3+k4)
  t = t+ dt
  return([t,p])

def edo(t, p, dt, tf, metodo):
  sol = [[t,p]]
  while t<=tf:
    t,p = metodo(t, p, dt)
    sol.append([t,p])
  sol = pd.DataFrame(data = sol, columns=['t','p'])
  return(sol)

ft = edo(t=0, p=100, dt=50, tf=1000, metodo=euler)
ft.rename(columns = {'p':'euler'},inplace=True)
ft_rk4 = edo(t=0, p=100, dt=50, tf=1000, metodo=rk4)
ft['rk4'] = ft_rk4['p']
ft['exact'] = list(map(exact,ft['t']))
graf = ft.plot(x='t',y=['euler','rk4','exact'])
plt.show()
```

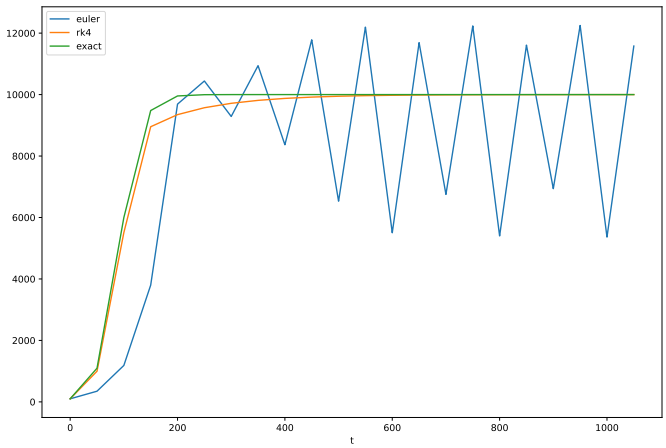



Figura 10.7: Solution of the Exact and Numerical Logistic Equation by Euler and Runge-Kutta Methods 4th Order

7. Mass-Spring System: Solution when $\mu = 0$ by the 4th Order Runge-Kutta Method

In the 5 example, the Mass-Spring system was analyzed for the case where the damping $\mu$ became zero. It is expected that the amplitude of the oscillation remains constant, because given that $\mu = 0$ the system neither gains nor loses energy. We will solve this case using the 4th$^a$ order Runge-Kutta Method and verify that it produces a correct result in this situation.

The following is the sequence of calculations for an RK4 on a system of two first-order equations, as was done with the Euler Method when applied to the system of the example 5.

The implementation is shown step by step below, with $h = 0.05$ , $u = 0$ , $k = 0.1$ , $m = 4kg$ , $x(0) = 50m$ and $v(0) = 50m/s$

$$\begin{aligned}
x' &= f(t,x,v) = v \\
x'(0) &= v(0) = 50 \\
k1 &= v(0) = 50 \\
k2 &= v(0) + 0,5.h.k1 = 50 + 0,5.0,05.50 = 51,25 \\
k3 &= v(0) + 0,5.h.k2 = 50 + 0,5 * 0,05 * 51,25 = 51,281 \\
k4 &= v(0) + h.k3 = 50 + 0,05 * 51,28125 = 52,564 \\
x(0+0,05) &= x(0) + h\frac{(k1+2.k2+2.k3+k4)}{6} \\
&= 50 + 0,05\frac{(50+51,25+51,281+52,564)}{6} = 52,564 \\
x(0,05) &= 52,564
\end{aligned} \tag{10.11}$$

| X | V |
|---|---|
| $x' = f(t,x,v)$ | $v' = g(t,x,v)$ |
| $k_1 = f(t,x,v)$ | $k_1 = g(t,x,v)$ |
| $k_2 = f(t+\frac{h}{2}; x+k_1\frac{h}{2}; v+k_1\frac{h}{2})$ | $k_2 = g(t+\frac{h}{2}; x+k_1\frac{h}{2}; v+k_1\frac{h}{2})$ |
| $k_3 = f(t+\frac{h}{2}; x+k_2\frac{h}{2}; v+k_2\frac{h}{2})$ | $k_3 = g(t+\frac{h}{2}; x+k_2\frac{h}{2}; v+k_2\frac{h}{2})$ |
| $k_4 = f(t+h; x+k_3.h; v+k_3.h)$ | $k_4 = g(t+h; x+k_3.h; v+k_3.h)$ |
| $x(t+h) = x(t) + \frac{h}{6}(k_1+2k_2+2k_3+k_4)$ | $v(t+h) = v(t) + \frac{h}{6}(k_1+2k_2+2k_3+k_4)$ |

Tabela 10.1: Sequence of steps for applying the Runge-Kutta Method of $4^a$ order to the 5

example

$$
\begin{aligned}
v' &= g(t,x,v) = -\tfrac{K}{m}x - \tfrac{u}{m}.v \\
v' &= g(t,x,v) = -\tfrac{0,1}{4}x(t) - \tfrac{0}{4}v(t) = -0,025.x(t) \\
v' &= -0,025.x \\
x(0) &= 50 \\
v(0) &= 50 \\
v'(0) &= -0,025.x(0) = -0,025.50 = -1,25 \\
k1 &= g(t,x,v) = v'(0) = -1,25 \\
k2 &= g(t+0,5.h.k1; x+0,5.h.k1; v+0,5h.k1) = -0,025(x+0,5.h.k1) \\
k2 &= -0,025 * [50+0,5*0,05*(-1,25)] = -1,249 \\
k3 &= g(t+0,5.h.k2; x+0,5.h.k2; v+0,5h.k2) = -0,025(x+0,5.h.k2) \\
k3 &= -0,025[50+0,5*0,05*(-1,249)] = -1,249 \\
k4 &= g(t+h.k3; x+h.k3; v+h.k3) = -0,025(x+h.k3) \\
k4 &= -0,025 * [50+0,05*(-1,249)] = -1,248 \\
v(0+0,05) &= v(0) + h\tfrac{(k1+2.k2+2.k3+k4)}{6} = 50+0,05\tfrac{(-1,25-1,249-1,249-1,248)}{6} = 49,938 \\
v(0,05) &= 49,938
\end{aligned}
\tag{10.12}
$$

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

vL = lambda t,x,v : -(0.1*x+0.0*v)/6
xL = lambda t,x,v : v

t,x,v = 0, 5, -10
tf, dt = 1000, 0.1
sol = []

while t<tf:
  sol.append([t,x,v])
```

```
  [k1, l1] = [fg(t,x,v) for fg in [xL,vL]]
  [k2, l2] = [fg(t+dt/2, \
              x+k1*dt/2, \
              v+l1*dt/2) for fg in [xL,vL]]
  [k3, l3] = [fg(t+dt/2, \
              x+k2*dt/2, \
              v+l2*dt/2) for fg in [xL,vL]]
  [k4, l4] = [fg(t+dt, \
              x+k3*dt, \
              v+l3*dt) for fg in [xL,vL]]
  t, x, v = t + dt, \
            x + (k1+2*k2+2*k3+k4)*dt/6, \
            v + (l1+2*l2+2*l3+l4)*dt/6
sol = pd.DataFrame(data = sol, \
                   columns = ['t','x','v'])

graf = sol.plot(x='t',y=['x','v'])
plt.show()
```

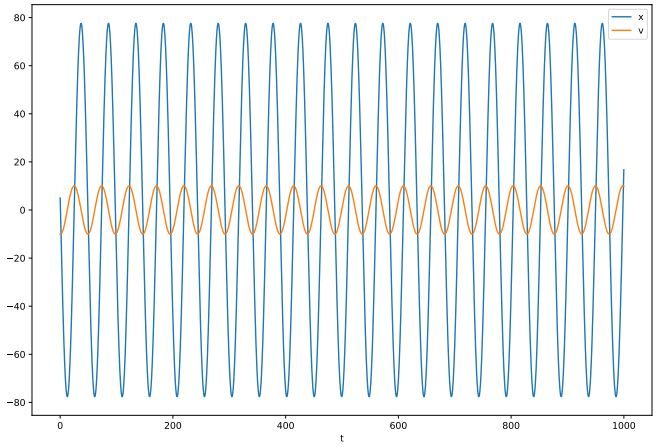



Figura 10.8: Spring Mass System with $\mu = 0$, solution by the 4th Order Runge-Kutta Method

8. Oscillation Period of a Mass-Spring System

Our goal is to determine how the period of oscillation of the system changes as the mass of the object increases. To do this we will determine the times at which the system crosses any two peaks and calculate the difference between the consecutive peaks. In the end we define the period as the average of this

sequence of values.

To determine the moments when the function reaches an extreme point (maximum or minimum) we take into account that at this point the first derivative of the function will change sign (in algebraic terms we say that $x'(t + \Delta h) * x'(t) < 0$).

Solving this problem, for a system with the same parameters as in the previous example and distinct values of $m$ can be seen below:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

m = 6
vL = lambda t,x,v : -(0.1*x+0.28*v)/m
xL = lambda t,x,v : v

t, x, v, dt, tf = 0, 5, -10, 0.1, 100

def sedol(t,x,v,dt,tf):
  sol = [[t,x,v]]
  while t<tf:
    [k1, l1] = [fg(t,x,v) for fg in [xL,vL]]
    [k2, l2] = [fg(t+dt/2, \
                x+k1*dt/2, \
                v+l1*dt/2) for fg in [xL,vL]]
    [k3, l3] = [fg(t+dt/2, \
                x+k2*dt/2, \
                v+l2*dt/2) for fg in [xL,vL]]
    [k4, l4] = [fg(t+dt, \
                x+k3*dt, \
                v+l3*dt) for fg in [xL,vL]]
    t, x, v = t + dt, \
              x + (k1+2*k2+2*k3+k4)*dt/6, \
              v + (l1+2*l2+2*l3+l4)*dt/6
    sol.append([t,x,v])
  sol = pd.DataFrame(data = sol, \
                     columns = ['t','x','v'])
  return(sol)

periods = []
for m in np.linspace(1,12,12):
  ft_runge = sedol(t, x, v, dt, tf)
  ft_runge = ft_runge[ft_runge['v'] * \
                      ft_runge['v'].shift(1) < 0]
  period = (ft_runge.t - \
            ft_runge.t.shift(1)).mean()
```

```
  periods.append([m,period])

periods = pd.DataFrame(data=periods, \
                       columns=['mass','period'])
graf = periods.plot(x='mass', \
                    y='period', \
                    kind='scatter')
plt.show()
```

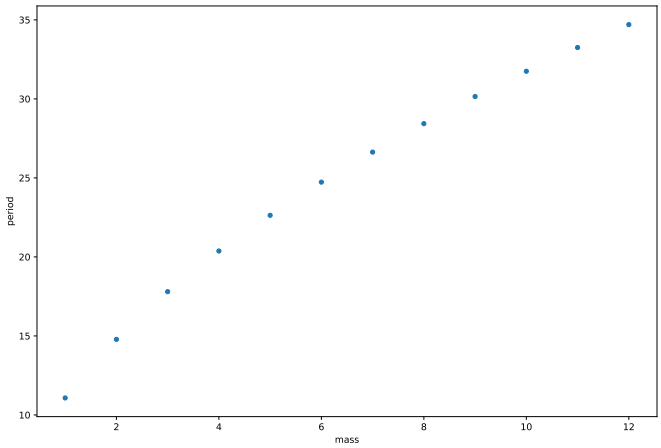



Figura 10.9: Oscillation Period of a Mass-Spring System

9. Two-body system

We will now increase the complexity of the problem. We will no longer have one body oscillating around its equilibrium point, but two bodies connected by a spring to each other, being able to move freely through a plane, with friction. In this condition we will have the following set of equations defining the problem:

$$m_1.x_1'' = (x_2 - x_1)K - x_1'.\mu_1$$

$$m_2.x_2'' = F - (x_2 - x_1)K - x_2'.\mu_2$$

Making $v_1 = x_1'$ and $v_2 = x_2'$ we have

$$v_1' = (x_2 - x_1)\frac{K}{m_1} - \frac{(v_1\mu_1)}{m_1}$$

$$v_2' = \frac{F}{m_2} - (x_2 - x_1)\frac{K}{m_2} - \frac{(v_2\mu_2)}{m_2}$$

Assuming: - $m_1 = 10kg$ - $m_2 = 5kg$ - $k = 0,1$ - $u_1 = u_2 = 0,15$ - $x_1(0) = x_2(0) = 0$ - $v_1(0) = 0$ - $v_2(0) = -3m/s$ - $F$ a $1N$ pulse from $t = 20$ seconds to $t = 50$ seconds,

we will have the time evolution for the respective centers of mass of bodies $x_1$ and $x_2$, shown below:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

m1, m2, k, u1, u2 = 10, 5, 0.1, 0.15, 0.15
x1, x2, v1, v2 = 0, 0, 0, -3

x1L = lambda t, x1, v1, x2, v2, F : v1
v1L = lambda t, x1, v1, x2, v2, F : \
             (x2-x1)*k/m1 - v1*u1/m1
x2L = lambda t, x1, v1, x2, v2, F : v2
v2L = lambda t, x1, v1, x2, v2, F : \
             F/m2 -(x2-x1)*k/m2 -v2*u2/m2

def sedol(t, x1, v1, x2, v2, dt, tf):
  sol = [[t,x1,v1,x2,v2]]
  while t<tf:
    if (t >= 20) and (t <= 50):
      F = 1
    else:
      F = 0
    [a1, b1, c1, d1] = [fg(t,x1,v1,x2,v2,F) \
                  for fg in [x1L,v1L,x2L,v2L]]
    [a2, b2, c2, d2] = [fg(t+dt/2, \
                           x1+a1*dt/2, \
                           v1+b1*dt/2, \
                           x2+c1*dt/2, \
                           v2+d1*dt/2, F) \
                  for fg in [x1L,v1L,x2L,v2L]]
    [a3, b3, c3, d3] = [fg(t+dt/2, \
                           x1+a2*dt/2, \
                           v1+b2*dt/2, \
                           x2+c2*dt/2, \
                           v2+d2*dt/2, F) \
                  for fg in [x1L,v1L,x2L,v2L]]
    [a4, b4, c4, d4] = [fg(t+dt, x1+a3*dt, \
                           v1+b3*dt, x2+c3*dt, \
```

```
                              v2+d3*dt, F) \
                    for fg in [x1L,v1L,x2L,v2L]]
    t, x1, v1, x2, v2 = t + dt, \
                          x1 + (a1+2*a2+2*a3+a4)*dt/6, \
                          v1 + (b1+2*b2+2*b3+b4)*dt/6, \
                          x2 + (c1+2*c2+2*c3+c4)*dt/6, \
                          v2 + (d1+2*d2+2*d3+d4)*dt/6
    sol.append([t,x1,v1,x2,v2])
  sol = pd.DataFrame(data = sol, \
                columns = ['t','x1','v1','x2','v2'])
  return(sol)

ft = sedol(0, x1, v1, x2, v2, 0.1, 100)
graf = ft.plot(x='t', y=['x1','x2'])
plt.show()
```

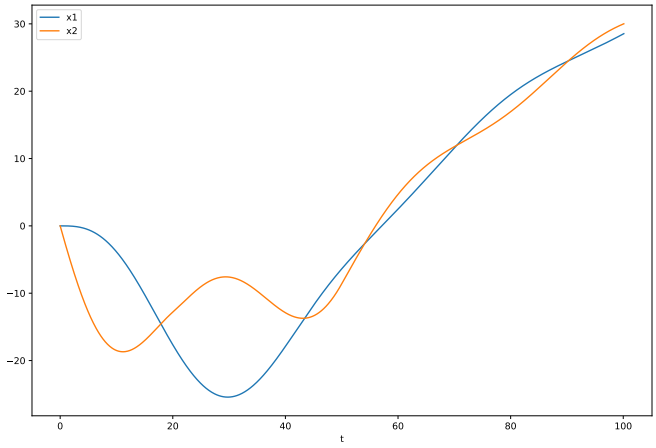



Figura 10.10: System with Two Bodies

10. Three-body system

We will now solve a system with three variables, using the 4th order Runge-Kutta method implemented with matrix operations, to decrease the complexity of the expressions. The basis of the reasoning is that from the matrix point of view the three expressions can be combined into a single matrix operation, $\vec{\mathbf{X}}' = \mathbf{A}.\vec{\mathbf{X}}$. The system is shown below:

$$\begin{aligned} x_1' &= -0,1\,x_1 - 1\,x_2 - 0,4\,x_3 \\ x_2' &= 0,003\,x_1 - 0,4\,x_2 + 2\,x_3 \\ x_3' &= 0,2\,x_1 - 0,02\,x_2 - 1\,x_3 \\ x_1(0) &= 1; x_2(0) = 2; x_3(0) = 3 \end{aligned} \tag{10.13}$$

- Implementation of the matrix operation $text\vec{b}fX^{'} = \mathbf{A}.text\vec{b}fX$

The Runge-Kutta 4th order method then requires implementing the following set of operations, for each of the variables :

$$\begin{aligned} k1 &= f\left[t; x_1; x_2; x_3\right] \\ k2 &= f\left[t + \tfrac{h}{2}; x_1 + k_1\,\tfrac{h}{2}; x_2 + l_1\,\tfrac{h}{2}; x_3 + m_1\,\tfrac{h}{2}\right] \\ k3 &= f\left[t + \tfrac{h}{2}; x_1 + k_2\,\tfrac{h}{2}; x_2 + l_2\,\tfrac{h}{2}; x_3 + m_2\,\tfrac{h}{2}\right] \\ k4 &= f\left[t; x_1 + k_3\,h; x_2 + l_3\,h; x_3 + m_3\,h\right] \\ x_1(t+h) &= x_1(t) + \tfrac{1}{6}\left(k_1 + 2\,k_2 + 2\,k_3 + k_4\right) \end{aligned} \tag{10.14}$$

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

h = np.array([0.1]).reshape(1,1)

C = np.array([[-0.100, -1.00, -0.4],
              [ 0.003, -0.40, 2.0],
              [ 0.200, -0.02, -1.0]])

t0 = np.zeros(1).reshape(1,1)
x0 = np.array([1,0,3]).reshape(3,1)
sol = np.concatenate((t0,x0),axis=0)

for i in range(300):
  k1 = C.dot(x0)
  k2 = C.dot(x0 + k1*h/2)
  k3 = C.dot(x0 + k2*h/2)
  k4 = C.dot(x0 + k3*h/2)

  t0 = t0 + h
  x0 = x0 + h/6*(k1+2*k2+2*k3+k4)
  sol0 = np.concatenate((t0,x0),axis=0)
  sol = np.concatenate((sol,sol0),axis=1)

sol = sol.T
df = pd.DataFrame(data = sol, \
                  columns = ['t','x1','x2','x3'])
```

Finally, the graphs of the time evolution of the variables (10.11) and the phase diagram (variables dependent on each other, at 10.12) are shown.

```
g = df.plot(x='t', y=['x1','x2','x3'])
plt.show()
```

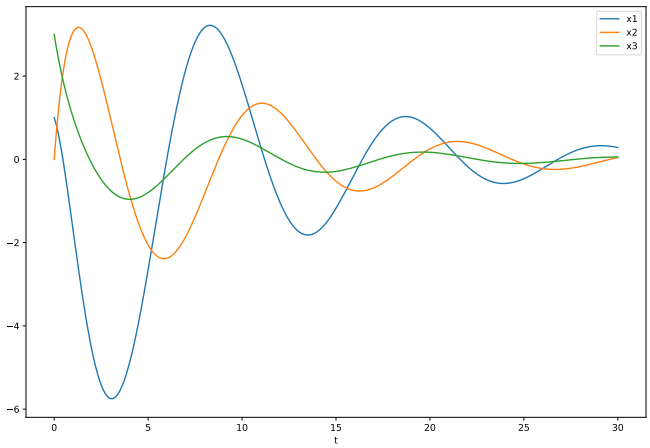



Figura 10.11: Temporal Evolution of Variables System with Three Bodies

```
g = df.plot(x='x1', y=['x2','x3'])
plt.show()
```

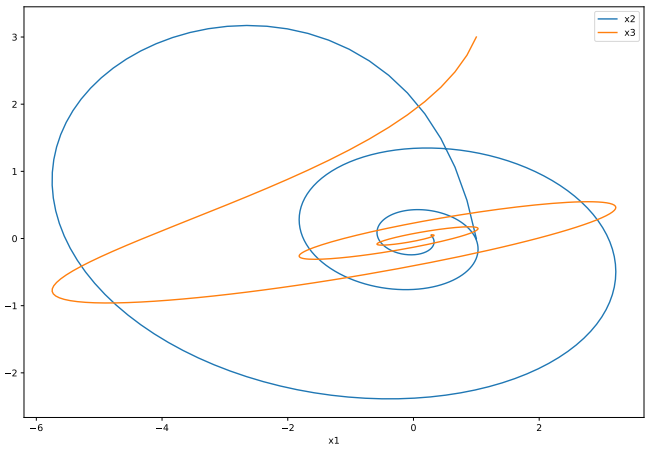



Figura 10.12: Phase Diagram, Variables $x_2$ and $x_3$ vs. $x_1$, System with Three Bodies

11. Drawing the family of solutions of $y' = \sqrt{|y^2 - 1|}$

In this case we wish to draw several graphs of $y(t)$, for different initial conditions $y_0 = y(0)$.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

def fL(t,y):
  result = np.sqrt(np.abs(y**2-1))
  return(result)

def runge4(t,y0,dt):
  k1 = fL(t,y0)
  k2 = fL(t+dt/2, y0 + k1*dt/2)
  k3 = fL(t+dt/2, y0 + k2*dt/2)
  k4 = fL(t+dt, y0 + k3*dt/2)
  y1 = y0 + dt/6*(k1+2*k2+2*k3+k4)
  return(y1)

ti = 0; tf = 3; n = 1000
ts = np.linspace(ti, tf, n); dt = ts[1] - ts[0]
y0s = np.linspace(-0.9,0,10)

sols = np.concatenate(([ti],y0s)).reshape(1,-1)
```

```
for t0 in ts:
  y1s = [ runge4(t0,y0,dt) for y0 in sols[-1,1:]]
  t1 = t0+dt
  sol1 = np.concatenate(([t1],y1s)).reshape(1,-1)
  sols = np.append(sols,sol1,axis=0)

names = list('t')
for i in range(len(y0s)):
  name = 'y' + str(i)
  names.append(name)

df = pd.DataFrame(data=sols, columns=names)
g = df.plot(x = 't', y = df.columns[1:])
plt.show()
```

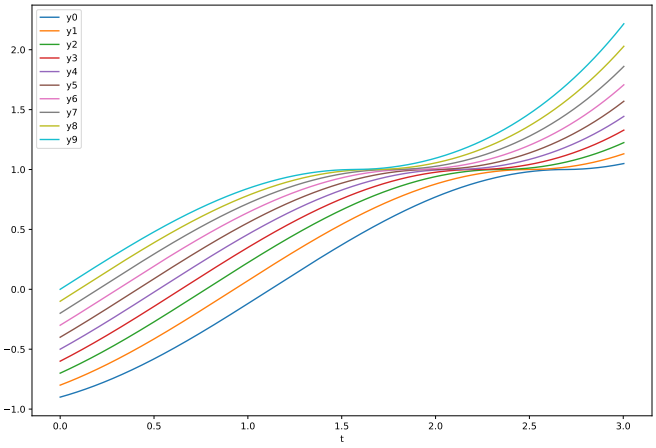



Figura 10.13: Family of Solutions of the Bernoulli Equation

## 10.3 Finite Differences Method

This is a method that starts from the following reasoning : the derivative can be calculated either by the definition, i.e. $\frac{f(t+h)-f(t)}{h}$ or by the average of its own expression, i.e. $\frac{f'(t+h)+f'(t)}{2}$. Equalizing the terms we will have :

$$\frac{f(t+h)-f(t)}{h} = \frac{f'(t+h)+f'(t)}{2} \tag{10.15}$$

Isolating $f(t+h)$ we will have :

$$f(t+h) = \frac{h}{2}\,[f'(t+h) + f'(t)] + f(t) \tag{10.16}$$

Since the value of $f'(t+h)$ to be calculated depends on the value of $f(t+h)$ we have an iterative process that converges line by line to the correct value.

12. Compare the solution by Euler's method, Runge-Kutta 4th Order and Finite Differences, with $h = 0.5$ for the solution of the equation representing the mass balance in a reactor, viz :

$$V\,\frac{dc}{dt} = F - Q\,c - k\,V\,c^2 \tag{10.17}$$

where $V = 12m^3$ is the reactor volume, $c$ is the concentration to be determined over time, $F = 175g/min$ is the feed rate, $Q = 1m^3/min$ is the flow rate and $k = 0.15m^3/g/min$ is the reaction rate that "consumes" the product. You want to solve the equation until the concentration $c$ of the chemical inside the reactor is stable, starting from different values for the initial concentration $c(0)$, which should range from 0 to 4. Also analyze the (hypothetical) case of a negative initial concentration by comparing the solution obtained by each of the three methods.

```
import numpy as np
import pandas as pd

def cL(c):
  V = 12; F = 175; Q = 1; k = 0.15
  resultado = (F - Q*c - k*V*(c**2))/V
  return(resultado)

def euler(c0,dt):
  c1 = c0 + cL(c0)*dt
  return(c1)

def runge4(c0,dt):
  k1 = cL(c0)
  k2 = cL(c0 + k1*dt/2)
  k3 = cL(c0 + k2*dt/2)
  k4 = cL(c0 + k3*dt)
  c1 = c0 + dt/6*(k1+2*k2+2*k3+k4)
  return(c1)

def diffin(c0,dt):
  # f(t+h) = h/2 * [ fL(t+h) + fL(t) ] + f(t)
  cL0 = cL(c0)
  c1 = c0 + cL0*dt
```

```
  i = 0; n_iter_max = 100;
  dif_perc = 1; dif_perc_max = 1e-6
  while ((i<n_iter_max) and (dif_perc > dif_perc_max)):
    cL1 = cL(c1)
    c1_new = dt/2 * ( cL1 + cL0 ) + c0
    dif_perc = abs(c1_new - c1)/c1_new
    c1 = c1_new
  return(c1)

ti = 0; tf = 2; n = 5;
ts = np.linspace(ti, tf, n); dt = ts[1] - ts[0]

c0 = 0
sols = np.array([ti, c0, c0, c0]).reshape(1,-1)
for t in ts:
  c0_euler = sols[-1,1]
  c1_euler = euler(c0_euler, dt)
  c0_runge = sols[-1,2]
  c1_runge = runge4(c0_runge, dt)
  c0_diffin = sols[-1,3]
  c1_diffin = diffin(c0_diffin, dt)
  sols0 = np.array([t+dt, \
                    c1_euler, \
                    c1_runge, \
                    c1_diffin]).reshape(1,-1)
  sols = np.concatenate((sols, sols0), axis=0)

df = pd.DataFrame(data=sols, \
                  columns=['t','euler',\
                           'runge','dif_fin'])
graf = df.plot(x='t', y=['euler','runge',\
                         'dif_fin'])
plt.show()
```

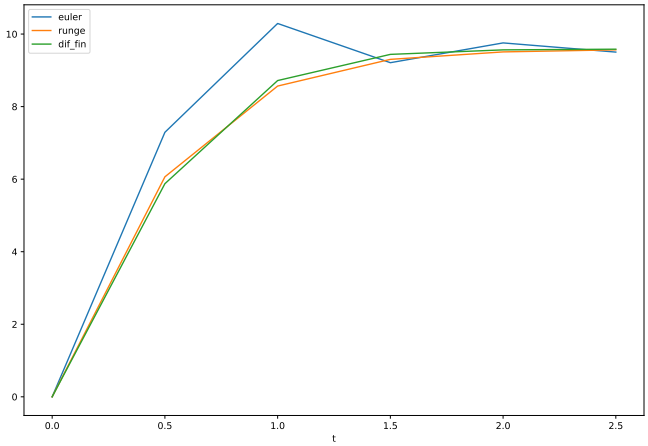



Figura 10.14: Solving an ODE by Finite Difference Method

In addition figures 10.15 and 10.16 show the plots of the time evolution of $c(t)$ obtained with the three methods for the initial cases $c(0) = 0$ and $c(0) = -8$, always remembering that the latter situation (where the initial concentration is negative) is purely theoretical. As you can see, there is a great deal of agreement between the practical result obtained by applying the 4th order Runge-Kutta method with extended precision and the finite difference method. For this example we will create a function that will generate the family of solutions.

```
import numpy as np
import pandas as pd

def cL(c0):
  V = 12; F = 175; Q = 1; k = 0.15
  resultado = (F - Q*c0 - k*V*(c0**2))/V
  return(resultado)

def euler(c0, dt):
  c1 = c0 + cL(c0)*dt
  return(c1)

def runge(c0, dt):
  k1 = cL(c0)
  k2 = cL(c0 + k1*dt/2)
  k3 = cL(c0 + k2*dt/2)
  k4 = cL(c0 + k3*dt)
  c1 = c0 + dt/6*(k1+2*k2+2*k3+k4)
```

```
  return(c1)

def difin(c0,dt):
  dif_max = 1e-6
  cL0 = cL(c0)
  c1 = c0 + cL0 * dt

  for i in range(100):
    cL1 = cL(c1)
    c1_new = c0 + (cL1 + cL0)/2 * dt
    dif_per = abs((c1_new - c1)/c1_new)
    c1 = c1_new
    if dif_per < dif_max:
      break
  return(c1)

def familia(c0, ti, tf, n):
  ts = np.linspace(ti, tf, n); dt = ts[1] - ts[0]
  sols = np.array([ti, c0, c0, c0]).reshape(1,-1)
  for t in ts:
    t1 = sols[-1,0] + dt
    c0_euler = sols[-1,1]
    c1_euler = euler(c0_euler,dt)
    c0_runge = sols[-1,2]
    c1_runge = runge(c0_runge,dt)
    c0_difin = sols[-1,3]
    c1_difin = difin(c0_difin,dt)
    sol0 = np.array([t1, \
                     c1_euler, \
                     c1_runge, \
                     c1_difin]).reshape(1,-1)
    sols = np.concatenate((sols, sol0), axis=0)
  df = pd.DataFrame(data = sols, \
                    columns = ['t','euler',\
                               'runge','difer'])
  return(df)

c0 = 0
ti = 0; tf = 2; n = 5
df = familia(c0, ti, tf, n)
graf = df.plot(x = 't', \
               y = ['euler','runge','difer'])
plt.show()
```

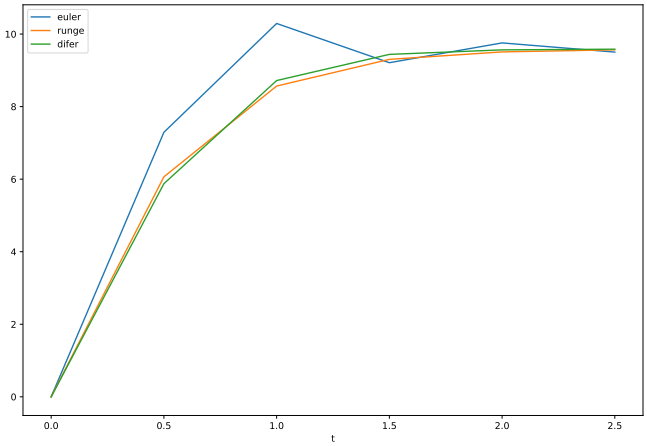



Figura 10.15: Temporal Evolution of $c(t)$ for $c(0) = 0$

```
c0 = -8
ti = 0; tf = 2; n = 5
df = familia(c0, ti, tf, n)
graf = df.plot(x = 't', \
               y = ['euler','runge','difer'])
plt.show()
```

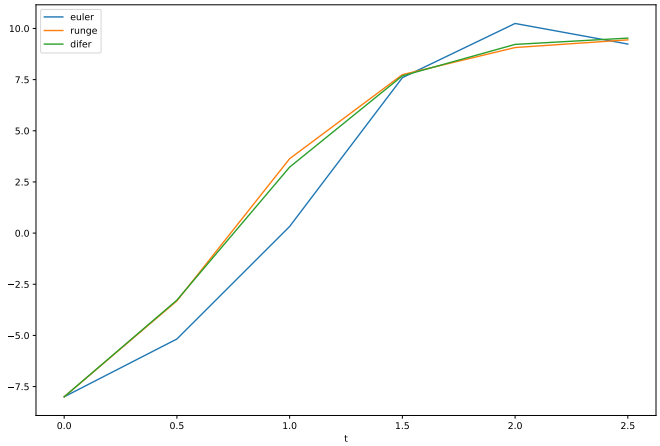



Figura 10.16: Temporal Evolution of $c(t)$ for $c(0) = 0$

13. Solve the differential equation $mx'' + u.x' = F(t)$ for $m = 2$, $u = 0.2$ and integration step $h = 0.5$ in the interval $t = [0; 50]$ by the Finite Differences method. Consider that the force $F(t)$ acts according to the following formation rule :

$$F(t) = \begin{cases} 0,5 & \text{se} \quad 0 \leq t < 4 \\ 1 - \frac{t}{8} & \text{se} \quad 4 \leq t < 8 \\ 0 & \text{se} \quad 8 \leq t \end{cases} \tag{10.18}$$

**Solution** First a transformation is performed on the equation so as to make it a system of linear differential equations of $1^a$ order. This transformation is shown in equations 10.19 and 10.20.

$$\begin{aligned} mx'' + u.x' &= F(x) \\ x''m &= F(t) - u.x' \\ x'' &= \frac{F(t) - u.x'}{m} \end{aligned} \tag{10.19}$$

$$\begin{cases} z' = \frac{F(t) - u.z}{m} \\ x' = z \end{cases} \tag{10.20}$$

Then the system with two equations is solved next and the result can be seen in 10.17

```
import numpy as np
import pandas as pd
u = 0.2; m = 2
```

```
def F(t):
  if (0≤t) and (t<4):
    return(0.5)
  elif (4≤t) and (t<8):
    return(1-t/8)
  else:
    return(0)

def xL(x,z,t):
  return(z)

def zL(x,z,t):
  return((F(t)-u*z)/m)

def euler(x0,z0,t0,dt):
  x1 = x0 + xL(x0,z0,t0)*dt
  z1 = z0 + zL(x0,z0,t0)*dt
  return(x1,z1)

def runge(x0,z0,t0,dt):
  k1 = xL(x0,z0,t0)
  l1 = zL(x0,z0,t0)

  k2 = xL(x0+k1*dt/2, z0+l1*dt/2, t0+dt/2)
  l2 = zL(x0+k1*dt/2, z0+l1*dt/2, t0+dt/2)

  k3 = xL(x0+k2*dt/2, z0+l2*dt/2, t0+dt/2)
  l3 = zL(x0+k2*dt/2, z0+l2*dt/2, t0+dt/2)

  k4 = xL(x0+k3*dt, z0+l3*dt, t0+dt)
  l4 = zL(x0+k3*dt, z0+l3*dt, t0+dt)

  x1 = x0 + dt/6*(k1+2*k2+2*k3+k4)
  z1 = z0 + dt/6*(l1+2*l2+2*l3+l4)
  return(x1,z1)

def difin(x0,z0,t0,dt):
  xL0 = xL(x0,z0,t0)
  zL0 = zL(x0,z0,t0)

  x1 = x0 + xL0*dt
  z1 = z0 + zL0*dt
  t1 = t0 + dt

  for i in range(100):
```

```
    xL1 = xL(x1,z1,t1)
    zL1 = zL(x1,z1,t1)

    xL1 = (xL1 + xL0)/2
    zL1 = (zL1 + zL0)/2

    x1 = x0 + xL1*dt
    z1 = z0 + zL1*dt

  return(x1,z1)

def familia(ti, tf, n, x0, z0):
  ts = np.linspace(ti,tf,n)
  sol = np.array([0,x0,z0,x0,\
                  z0,x0,z0]).reshape(1,-1)

  for t0 in ts:
    t0 = sol[-1,0]
    t1 = t0 + dt

    x0 = sol[-1,1]; z0 = sol[-1,2]
    x1_euler, z1_euler = euler(x0,z0,t0,dt)

    x0 = sol[-1,3]; z0 = sol[-1,4]
    x1_runge, z1_runge = runge(x0,z0,t0,dt)

    x0 = sol[-1,5]; z0 = sol[-1,6]
    x1_difin, z1_difin = difin(x0,z0,t0,dt)

    sol0 = np.array([t1,\
                     x1_euler,\
                     z1_euler,\
                     x1_runge,\
                     z1_runge,\
                     x1_difin,\
                     z1_difin]).reshape(1,-1)
    sol = np.concatenate((sol,sol0),axis=0)

  df = pd.DataFrame(data = sol,
                    columns = ['t','x_euler',\
                    'z_euler','x_runge',\
                    'z_runge','x_difin','z_difin'])
  return(df)

dt = 0.5; ti = 0; tf = 50; x0 = 0; z0 = 0
n = int((tf - ti) / dt + 1)
```

```
df = familia(ti, tf, n, x0, z0)
graf = df.plot(x='t', y=['x_euler',\
                         'x_runge','x_difin'])
plt.show()
```

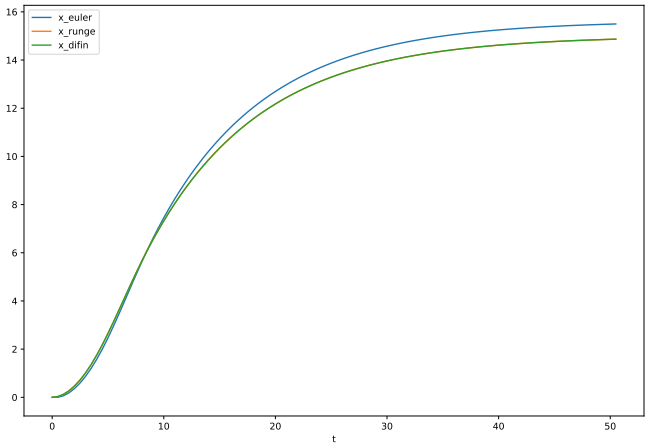



Figura 10.17: Solving the Differential Equation with Distinct Forces by Time Interval

14. Solve the nonlinear differential equation of the pendulum $\theta''(t) = -g/l * sin(\theta)$ for $\theta(0) = 10^o$ and $\theta'(0) = 0$ by the methods of Euler, Runge-Kutta 4th order, XNumbers with Runge-Kutta 4th order and Finite Differences. Compare the results obtained with the analytical solution of the linearized equation $\theta''(t) = -g/l * \theta$ which is valid for small values of $\theta$. Consider $g = 9.8m/s^2$ and $l = 2m$.

**Solution** We will reuse most of the previous code. We just need to change the definitions of x and z. The graph below shows the resolution by the three methods. In the case of Euler's method, as can be seen from the graph, even with a step of only 0.005 two cycles are enough to cause a divergence to appear. The errors in this case are caused by an accumulation of rounding errors (due to the amount of calculations) and truncation errors (due to the method itself).

```
import numpy as np
import pandas as pd
g = 9.8; L = 2

#Theta será x
```

```
def xL(x,z,t):
  return(z)

#Theta' será z
def zL(x,z,t):
  return(-g/L*np.sin(x))

x0_graus = 10; x0 = x0_graus * np.pi / 180
z0_graus_s = 0;  z0 = z0_graus_s * np.pi / 180

dt = 0.2; ti = 0; tf = 5
n = int((tf - ti) / dt + 1)

df = familia(ti, tf, n, x0, z0)
graf = df.plot(x='t', y=['x_euler',\
                         'x_runge','x_difin'])
plt.show()
```

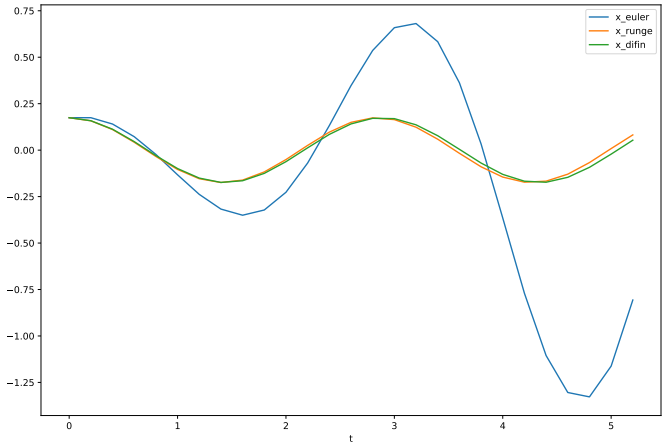



Figura 10.18: Solution of the Pendulum Nonlinear Differential Equation

15. Solve the system of nonlinear differential equations :

$$\begin{cases} x' = 1,2\,x - 0,6\,x\,y \\ y' = -0,8\,y + 0,3\,x\,y \\ x(0) = 2; y(0) = 1 \end{cases} \tag{10.21}$$

also known as the Predator-Prey Model, because it can be used to describe the ecological equilibrium between a predator species (Y) and a prey

population (X). Make in addition to the graph of the time evolution of X and Y, the X-Y phase graph. Solve the system by the 4th order Runge-Kutta method via XNumbers.

**Solution** The model describes a prey population X which has an associated growth rate (in the case 1.2) and a mortality rate arising from predator-prey encounters (in the case 0.6), described by the factor $x.y$ in the first equation. The predator population in turn tends to decrease with its own population increase (because of competition among predators themselves, factor -0.8 in the second equation) and to increase due to predator-prey encounters, also described in the second equation by the factor $x.y$ and the factor 0.3.

The solution is presented in the following graphs. The first graph shows the time evolution of the solution obtained by the runge kutta method only. Notice the clear oscillation of populations as the relative quantities of predator and prey vary over time. The following figure shows the phase diagram for the solution via runge kutta (solid line) and by the Euler method (dashed line). Notice that the error accumulation caused by Euler's method causes the graph to not "close'', generating increasing oscillations.

```
import numpy as np
import pandas as pd

#Theta will be x
def xL(x,z,t):
  return(1.2*x-0.6*x*z)

#Theta' will be z
def zL(x,z,t):
  return(-0.8*z+0.3*x*z)

x0=2; z0=1

dt = 0.02; ti = 0; tf = 10
n = int((tf - ti) / dt + 1)

df = familia(ti, tf, n, x0, z0)
graf = df.plot(x='t', y=['x_euler',\
                         'x_runge','x_difin'])
plt.show()
```

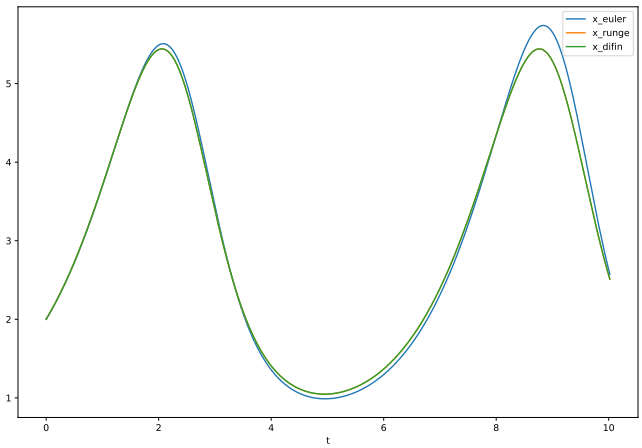



Figura 10.19: Solution of the Lotka-Voltera (Predator-Force) Model

```
graf1 = df.plot(x='x_euler', y='z_euler')
graf2 = df.plot(x='x_runge', y='z_runge')
graf3 = df.plot(x='x_difin', y='z_difin')
plt.show()
```

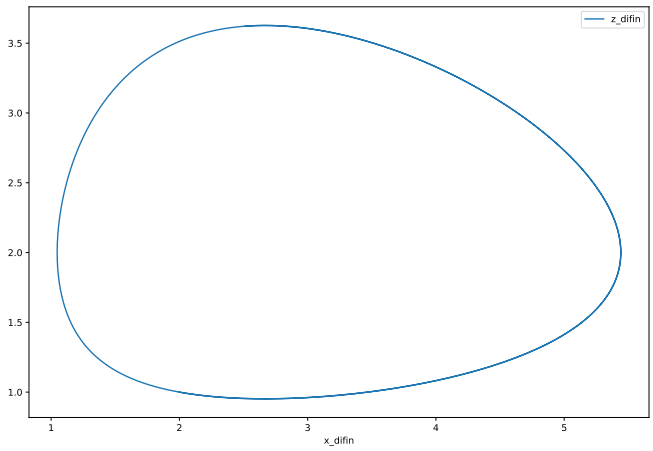



Figura 10.20: Phase plot of the dependent variables, $x$ and $y$, Predator-Wait Model, resolutions by Euler Method, dashed line and by runge kutta 4th order, continuous line

16. The system of equations shown in 10.22 is known by the name Van der Walls System of Equations and its equations are classified as rigid. Solve the same for $t = [0; 100]$.

$$\begin{cases} y_1' = y_2 \\ y_2' = u\left(1 - y_1^2\right) y_2 - y_1 \\ y_1(0) = y_2(0) = 1 \end{cases} \tag{10.22}$$

**Solution** The intrinsic characteristics of these equations make them difficult to solve numerically, requiring a special method called *Piecewise-Integration*. The graph of the time evolution of the variable $y_1$, Van de Walls equations, is shown below. The reader can try to solve this system by the Runge-Kutta 4th order method and verify that it fails completely. This is because the slope of the function being nearly vertical, this causes error in the system approximations of the slope of the solution function when using the runge kutta method.

```
# TO BE IMPLEMENTED
```

## 10.4 Comparison of Solving Methods

In this section we will compare the efficiency of spreadsheet numerical methods by applying them to long term solving on a system whose output will present an oscillating function. This combination will take the solving power and the

accuracy of the spreadsheet methods to the extreme, because not only will we be looking for a solution for a long time, but it will oscillate, which will cause the derivative to take on positive and negative values.

17. Calculate the current $i(t)$ for a series RLC circuit, with input voltage $E(t) = E_0 \sin \omega\, t$ for a time period $t$ from 0 to 100 seconds. The integration step will be $h = 0.1$ seconds. It is desired to compare the exact solution with the solutions obtained with the methods of : Euler, Finite Differences, Runge-Kutta 4th order and by a System of 1st order ODEs, both the last two methods by the functions implemented by the XNumbers package.

**Theoretical Solution**

Applying Kirchoff's Law of meshes to the series RLC circuit, we will have :

$$L\,\frac{di}{dt} + R\,i + \frac{q}{C} - E(t) = 0 \tag{10.23}$$

Isolating $E(t)$ :

$$L\,\frac{di}{dt} + R\,i + \frac{q}{C} = E(t) \tag{10.24}$$

Changing the derivative notation :

$$L.i' + R.i + q/C = E(t) \tag{10.25}$$

Isolating $L\,i'$ :

$$L.i' = -R.i - q/C + E(t) \tag{10.26}$$

Passing L to the other side by dividing :

$$i' = -\frac{R}{L}\,i - \frac{1}{C\,L}\,q + \frac{E(t)}{L} \tag{10.27}$$

System of $1^a$ order ODEs in $i'$ and $q'$ :

$$\begin{array}{ll} i' & = -\frac{R}{L}\,i - \frac{1}{C\,L}\,q + \frac{E(t)}{L} \\ q' & = i \end{array} \tag{10.28}$$

Assuming $E(t) = E_0 \sin \omega\, t$ and passing to matrix form we will have :

$$\begin{bmatrix} i' \\ q' \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{1}{C\,L} \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} i \\ q \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} E(t) \tag{10.29}$$

Solving in a literal way for the variable $q$, with $R = 0\,\Omega$, $L = 1\,H$, $C = 0,25\,F$, $E_0 = 1\,V$ and $\omega = \sqrt{3,5}$ we will have (the resolution is not presented in this text):

$$q = -\frac{k\,\omega}{p}\,\sin p\,t + k\,\sin\omega\,t \tag{10.30}$$

where

$$p = \frac{1}{\sqrt{L\,C}} \tag{10.31}$$

e

$$k = \frac{E_0}{L\,(p^2 - \omega^2)} \tag{10.32}$$

### Code for Solution

The following is the code for numerical solving and calculating the values of the exact solution. Most of the code (embedded in the `familia` function) is reused.

```
import numpy as np
import pandas as pd

R = 0; L=1; C=0.25; E0=1; w = np.sqrt(3.5)
p = 1/np.sqrt(L*C); k = E0/(L*(p**2 - w**2))

def x_exact(t):
  return(-k*w/p*np.sin(p*t)+k*np.sin(w*t))

#q será x; xL = q'
def xL(x,z,t):
  return(z)

#i será z, pois i = q'
def zL(x,z,t):
  E = E0 * np.sin(w*t)
  iLinha = -R/L*z-1/(C*L)*x+E/L
  return(iLinha)

x0=0; z0=0

dt = 0.02; ti = 0; tf = 400
n = int((tf - ti) / dt + 1)

df = familia(ti, tf, n, x0, z0)
```

```
df['x_exact'] = [x_exact(t) for t in df['t']]
```

### Euler's method

The following figure shows the graph comparing the exact solution with the solution obtained by Euler's method. As can be seen, the accumulation of errors causes the numerical solution to diverge from the exact solution already in the first cycle of fast oscillation, making the method not very useful for analyzing the solution after 5 seconds.

```
t_end = 40
graf = df[df['t']<t_end].plot(x='t', \
                              y=['x_euler',\
                                 'x_exact'])
plt.show()
```

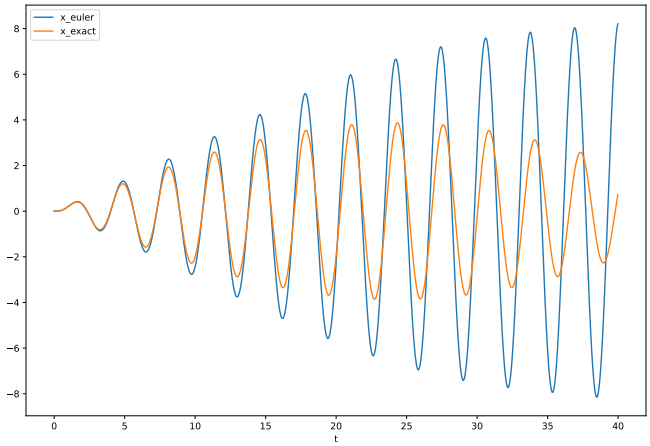



Figura 10.21: Solution of Oscillating Circuit by Euler's Method

### Runge-Kutta and Finite Differences Methods

In 10.22 the graph comparing the exact solution with the numerical solution via runge kutta 4th order and finite differences is shown. As can be seen, both methods provide great stability in the solution. It should be considered that the Runge Kutta method achieves similar accuracy as the finite differences method with a much smaller number of operations.

```
graf = df[df['t']<100].plot(x='t', \
                            y=['x_runge',\
                               'x_difin',\
```

```
                                        'x_exact'])
plt.show()
```

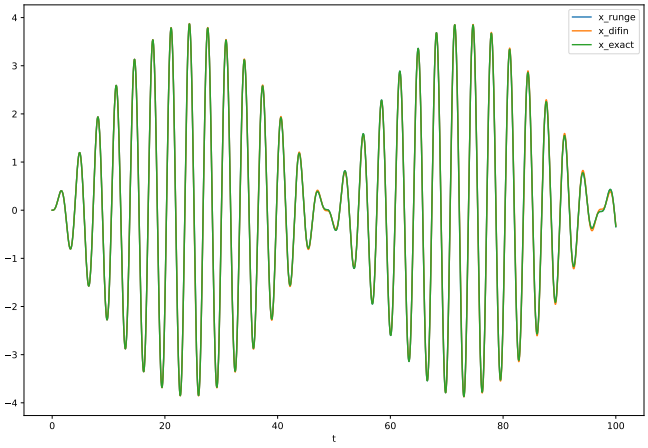



Figura 10.22: Solution of Oscillating Circuit by Runge-Kutta 4th Order and Finite Difference Methods

## 10.5 Exercises

1. Let be the following set of equations: $Y'(t) = c.X(t).a.Y(t) - d.X(t)eX'(t) = -a.Y.X + b.Y$ with the following initial conditions X(0)=0, Y(0)=100,000. Knowing that $c = 0,5\%; d = 1,0\%; a = 0,003\%; b = 0,5\%$ plot the graphs of $X(t)$ e $Y(t)$ and the graph of $Y$ with $X$ on the horizontal axis. Hint: The latter graph should generate a spiral in the first quadrant. Use $h = 1$ and $t$ varying $from 0 to 3000$.

2. The population of a city t years after 1990 satisfies the differential equation y'= 0.02y. What is the growth constant? How fast will the population grow when it reaches 3 million people? At what population level will the population be growing at a rate of 100,000 people per year?

3. Resolve numerically (by the 4th order Euler and Runge-Kutta methods) the following differential equations:

   (a)
   $$y^{+}2y = 1; y(0) = 1$$

   (b)
   $$y^{+}\frac{y}{1+t} = 20; y(0) = 10$$

4. Resolve by the algebraic method of separation of variables, by Euler's numerical method, and by the Runge-Kutta 4th order method, the ordinary differential equation $y(x) = 0.4y(x)$ with $y(0) = 10$, for the interval $x = [0; 10]$ with $Dt = 0.1$. Compare the percentage error between the numerical methods and the exact solution.

5. Suppose the price of a commodity changes so that its rate of change is proportional to the scarcity of the product given by $\frac{dp}{dt} = k(D - S)$ , where $D$ and $S$ are the demand and supply functions of the product given as a function of price by $D$=7-$p$ and $S$=$p$+1. If the price is \$6 when t=0 and \$4 when t=4 find p(t). Show as time increases the price approaches the value for which Demand and Supply become equal.

6. Solve numerically the differential equations below implementing the Euler, Finite Differences and Runge-Kutta 4th order methods , drawing the graph of the solution function. Use as integration step $h = 0.1$.

   (a) $y' = y\,x^2 - 1,1\,y$
   $y(0) = 1$

   (b) $y' = z$ e $z' = -4\,z - 3\,y$
   $y(0) = 3$ e $z(0) = 4$

   (c) $y' = z$ e $z' = -4\,z - 3\,y + 3\,exp(-2\,t)$
   $y(0) = 1$ e $z(0) = -1$

   (d) $y' = z$ e $z' = -6\,z - 9\,y + 0$
   $y(0) = -1$ e $z(0) = 1$

   (e) $y' = z$ e $z' = -5\,z - 6\,y + 3\,exp(-2\,t)$
   $y(0) = 1$ e $z(0) = -1$

7. Draw the family of solutions of the differential equations: $y' = y - x^2\,y^2$, exact solution: $y(x) = \frac{1}{x^2 - 2\,x + 2 + \left(\frac{1}{y_0} - 2\right)e^{-x}}$ by drawing the family of solutions of both the exact and numerical solution (Runge-Kutta 4th order). Compare the percentage difference between the approximate numerical value obtained with the 4th order Runge-Kutta method and the exact value.

# Capítulo 11

# Fourier analysis

## 11.1 Fourier series

The Fourier series allows one to decompose a signal in time, periodic, with period equal to $T$ into a weighted sum of sines and cosines. The series is represented by the relation :

$$f(t) = a_0 + \sum_{k=0}^{\infty} a_k \, cos\left(\frac{2\,\pi}{T}\,k\,t\right) + b_k \, sin\left(\frac{2\,\pi}{T}\,k\,t\right) \tag{11.1}$$

where the coefficients $a_k$ and $b_k$ are calculated by the following formulas :

$$a_0 = \int_{-\frac{T}{2}}^{+\frac{T}{2}} f\,(t)\,dt \tag{11.2}$$

$$a_k = \int_{-\frac{T}{2}}^{\frac{T}{2}} f\,(t)\;cos\left(\frac{2\,\pi}{T}\,k\,t\right) dt \tag{11.3}$$

$$b_k = \int_{-\frac{T}{2}}^{\frac{T}{2}} f\,(t)\;sin\left(\frac{2\,\pi}{T}\,k\,t\right) dt \tag{11.4}$$

If the function $f(t)$ is given as a set of points (and not through an algebraic expression) the coefficients $a_k$ and $b_k$ can be calculated using the method of least squares, since the expression 11.1 is a linear expression for the coefficients $a_k$ and $b_k$. For calculation using the Python language, one can either use the `fft` function of `numpy`, numerical integration, or linear regression by the matrix formula $\beta = (X^t.X)^{-1}.(X^t.Y)$, as will be shown in the following examples.

1. Calculate the coefficients $a_k$ and $b_k$ of the Fourier series representing the function $f(t) = t$ for $t = [-1, 1]$.

This function represents a line segment that repeats indefinitely in time. To calculate the coefficients, we first determine that $T$ is equal to $1 - (-1) = 2$ and then do :

$$a_0 = \int_{-1}^{+1} t dt = \left[\frac{t^2}{2}\right]_{-1}^{+1} = 0 \tag{11.5}$$

$$a_k = \int_{-1}^{+1} t\, cos\,(\pi\, k\, t)\, dt = \left[\frac{\pi\, k\, t\, sin\,(\pi\, k\, t) + cos\,(\pi\, k\, t)}{\pi^2\, k^2}\right]_{-1}^{+1} = 0 \tag{11.6}$$

$$b_k = \int_{-1}^{+1} t\, sin\,(\pi\, k\, t)\, dt = \left[\frac{sin\,(\pi\, k\, t) - \pi\, k\, t\, cos\,(\pi\, k\, t)}{\pi^2\, k^2}\right]_{-1}^{+1} = \frac{2\ (sin\,(\pi\, k) - \pi\, k\, cos\,(\pi\, k))}{\pi^2\, k^2} \tag{11.7}$$

The calculation of the coefficients and the reconstitution of the signal using `numpy` is presented below. Our goal is to reconstruct a signal $f(t) = t$, which has a fundamental period $T = 2$, with up to the $n$th $n = 5$ pair of coefficients in an interval $t_i, t_f$ equal in this case to $-1, +1$. We will use $n_p = 100$ points by default for reconstructing the sign in this interval.

Once we have defined the goal of our function (reconstruct the signal with precision of $n$ pairs of Fourier series coefficients), our first step is to calculate the associated coefficients. For this we will assume there is a function $generate_cns$ which receives the function $f(t)$, the number of coefficient pairs to calculate $n$ and the fundamental period $T$ returning the desired $n$ coefficient pairs $a_n, b_n$ in an array `numpy`.

With the coefficients ($cns$), the fundamental period ($T$), the signal reconstitution interval ($[t_i, t_f]$) and the desired number of points $np$ let's assume there is a function that takes these values as parameters and returns a pandas data frame with three columns $t$, $f(t)$ and $f_n(t)$, that is, with the value of $t$, the original function $f(t)$ and the reconstructed function $f_n(t)$.

We will now work on the development of the function $gera_cns$. The function `generates_cns` gets the function $f(t)$ to generate the coefficients, the number $n$ of pairs of coefficients to be generated, and the fundamental period $T$. It then returns an array `numpy` with the corresponding pairs of coefficients. For this it assumes the existence of a function $c_n(i, T)$ which takes the order $i$ of the pair of coefficients to be generated, the fundamental period $T$ and returns the corresponding pair. This function will be executed $n$ times and will be developed in the next listing.

As stated in the previous paragraph, the function `c_n` takes the function $f(t)$, the index $i$ and the fundamental period $T$ and returns a tuple of values with the

corresponding pair of values $a_n, b_n$. The process involves numerical integration. By default we use an integration step of $dt = 1e-4$ and assume that a coefficient equals 0 if it is less than $1e-12$

We will now work on the function `generate_fn` which will receive the function $f(t)$, the array `numpy` named `cns` with the $n$ pairs of coefficients of the associated Fourier series, the fundamental period $T$, the integration interval $t_i$, $t_f$ and the number of points $n_p$ desired in the signal reconstruction and will return a data frame with three columns $t$, $f(t)$ and $f_n(t)$.

This function assumes the existence of another function `f_n` which receives the function $f(t)$, the array `numpy` with the associated pairs of values of the coefficients $cns$, the fundamental period $T$ and a value of $t$ and computes the value of the reconstructed signal at this point.

Next we test this sequence of functions for the signal $f(t) = t$, with $n = 5$, $T = 2$, on the interval $t_i = -1$ to $t_i = +1$ and the interval divided into 100 points. (See 11.1)

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

f = lambda t:t
n = 5; T = 2; ti = -1; tf = 1; n_p = 100

def c_n(f, i, T, dt=1e-4, zero=1e-12):
  sum_a = 0; sum_b = 0
  ti = -T/2; tf = +T/2; t = ti
  while t ≤ tf:
    sum_a = sum_a + f(t)*np.cos(2*np.pi*i*t/T)*dt
    sum_b = sum_b + f(t)*np.sin(2*np.pi*i*t/T)*dt
    t = t + dt
  if abs(sum_a)<zero:
    sum_a = 0
  if abs(sum_b)<zero:
    sum_b = 0
  return(sum_a, sum_b)

def gera_cns(f, n, T):
  cns = []
  for i in range(n):
    par = c_n(f, i, T)
    cns.append(par)
  cns = np.array(cns)
  return(cns)

def f_n_t(cns, T, t):
```

```
  soma = 0
  for k, cn in enumerate(cns):
    soma = soma + cn[0]*np.cos(2*np.pi*k*t/T) + \
                  cn[1]*np.sin(2*np.pi*k*t/T)
  return(soma)

def gera_fn(cns, T, ti, tf, n_p):
  t = [ti]
  f_fourier = [f_n_t(cns, T, t[-1])]
  h = (tf-ti)/n_p
  while t[-1] <= tf:
    t.append(t[-1] + h)
    f_fourier.append(f_n_t(cns, T, t[-1]))
  data = pd.DataFrame({'t':t, \
                       'f_fourier':f_fourier})
  return(data)

def f_n(f, n, T, ti, tf, n_p):
  cns = gera_cns(f, n, T)
  data = gera_fn(cns, T, ti, tf, n_p)
  return(data)

data = f_n(f, n, T, ti, tf, n_p)
data['f_exact'] = list(map(f, data['t'].values))
g = data.plot(x='t', y=['f_exact', 'f_fourier'])
plt.show()
```

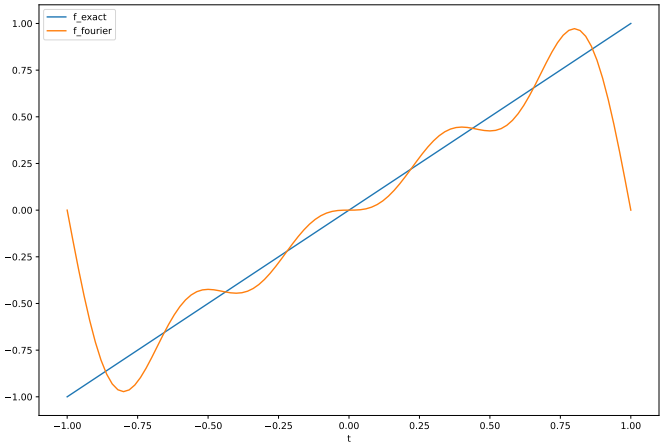



Figura 11.1: Reconstitution of the signal $f(t) = t$ with n=5, in the interval $(-1, 1)$

In the following figure (11.2) the calculation using the method of least squares is shown. First the bases for the regression are generated (the columns with the constant term, equal to 1, and the 5 columns for $cos(\frac{2\pi nt}{T})$ and $sin(\frac{2\pi nt}{T})$. This constitutes the matrix $\mathbf{X}$. The matrix $\mathbf{Y}$ is generated from the function $f(t)$ for the $n_p$ points in the interval between $[-1, 1]$. Then the matrix formula $\beta = (X^t X)^{-1}(X^t Y)$ is applied. From the *beta* values in this formula we calculate the values of the easily reconstructed function via $X.b$.

```
import numpy as np
import pandas as pd

ti = -1
tf = +1
n_p = 100
f = lambda t:t
T = 2

data = pd.DataFrame({'t':np.linspace(ti,tf,n_p)})
data['f_exact']=list(map(f,data['t'].values))
data['a0'] = 1

for k in range(1,6):
  data['a'+str(k)] = \
       np.cos(2*np.pi*k*data['t'].values/T)
  data['b'+str(k)] = \
```

```
        np.sin(2*np.pi*k*data['t'].values/T)

X = data.loc[:,'a0':].values; X.shape
```

```
(100, 11)
```

```
Y = data.loc[:,['f_exact']].values; Y.shape
```

```
(100, 1)
```

```
b = np.linalg.inv(X.T.dot(X)).dot(X.T.dot(Y));
data['f_fourier'] = X.dot(b)
g = data.plot(x='t',y=['f_exact', 'f_fourier'])
plt.show()
```

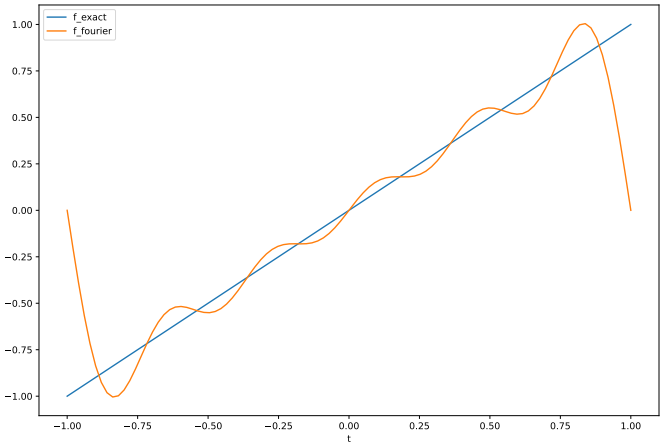



Figura 11.2: Calculation of Fourier Series Coefficients by the Method of Least Squares

## 11.2 Frequency Response Graphs

One of the most widely used techniques in Electrical Engineering is frequency response analysis. Mathematically, it is the plot of a function of complex variable $f(z)$ where the argument of the function assumes purely imaginary values $j\omega$, generating then the function $f(j\omega)$. For presentation purposes and better understanding, the graph is presented with the $\omega$ ordinate in logarithmic scale.

2. Given the function $Z(\omega) = 10 + \frac{10^4 - j\,\frac{10^6}{\omega}}{10 + j\left(0{,}1\,\omega - \frac{10^5}{\omega}\right)}$ draw the graph of the real part, the imaginary part and the modulus of $f(\omega)$

**Solution** python supports the use of complex numbers in their native form. The listing with the calculations and the corresponding graph (in linear scale) can be seen in figures 11.3 and 11.4.

```
f = lambda w: 10 + \
              (1e4 - 1j*1e6/w)/(10 + 1j*(0.1*w-1e5/w))
data = pd.DataFrame({'w':np.linspace(600,1600,1000)})
data['f_'] = list(map(f,data['w']))
data['f_re'] = np.real(data['f_'])
data['f_im'] = np.imag(data['f_'])
data['f_abs'] = np.abs(data['f_'])

g = data.plot(x='w', y=['f_re', 'f_im'])
plt.show()
```

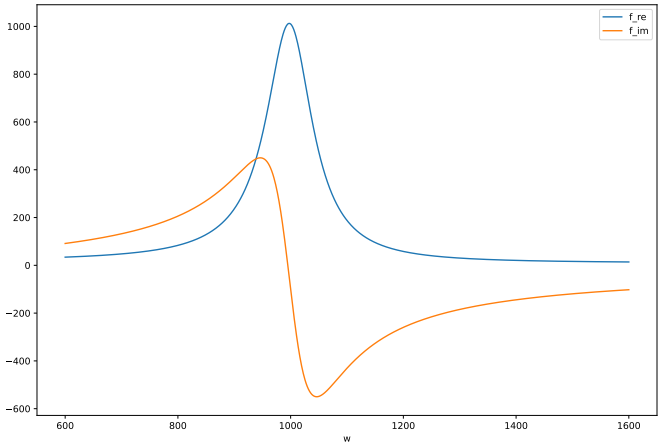



Figura 11.3: Graph of the Real and Imaginary Parts of f(w)

```
g = data.plot(x='w', y='f_abs')
plt.show()
```

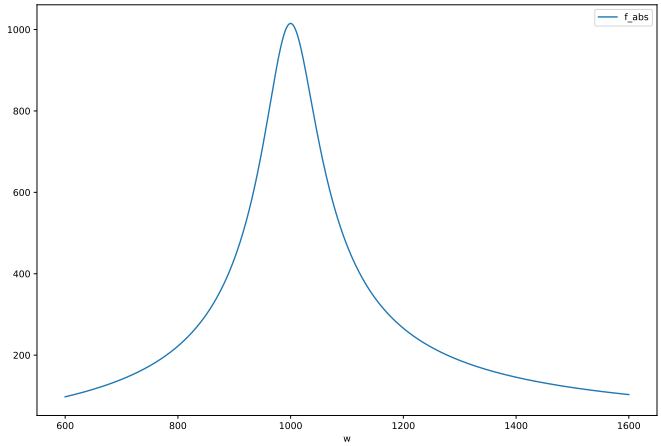



Figura 11.4: Graph of the Modulus of f(w)

## 11.3 Frequency Response of an RC Circuit

We will now look at the frequency response of a series RC circuit. You want to get the amplitude and phase relationship of the output sinusoid from the frequency of the input sinusoid. The output sinusoid is the voltage in the resistor and the input sinusoid is the actual input signal of the circuit. To do this we apply a voltage divider in the frequency domain to calculate the voltage $V_r$ by doing :

$$eqn71 V_r(s) = V(s)\,\frac{R}{R + \frac{1}{s\,C}} \tag{11.8}$$

Then substituting $s$ for $\jmath\,\omega$ we get :

$$eqn72 V_r(s) = V(s)\frac{R}{R + \frac{1}{\jmath\,\omega\,C}} \tag{11.9}$$

From which we can obtain the relationship between the output voltage $V_r(s)$ and the input voltage $V(s)$, as :

$$T(s) = \frac{V_r(s)}{V(s)} = \frac{R}{R + \frac{1}{\jmath\,\omega\,C}} \tag{11.10}$$

We can then generate a graph of the complex variable function $T(s)$ described in 11.10 by making it range from very negative values of $\jmath\omega$ to very positive values of $\jmath\omega$. Let us show a graph obtained for the values $R = 1.5kMega$ and $C = 1mega$, with $mega$ ranging from $0.1Mega$ to $1MMega$. As always we will use logarithmic scales for the gain ($20\log(|T(s)|)$) and for $\omega$. The predicted cutoff frequency for this circuit is given by the expression $f_c = \frac{1}{R\,C}$ which in this case gives $f_c = \frac{1}{10^4\ 10^{-6}} = 100\,Hz$.

```
T = lambda s,R=1500,C=1e-6: R/(R + 1/(s*C))
data = pd.DataFrame({'w':np.linspace(0.1,1e6,10000)})
data['f_'] = list(map(T,1j*data['w']))
data['f_re'] = np.real(data['f_'])
data['f_im'] = np.imag(data['f_'])
data['f_abs'] = np.abs(data['f_'])

g = data.plot(x='w', y='f_abs', logx=True, logy=True)
plt.show()
```

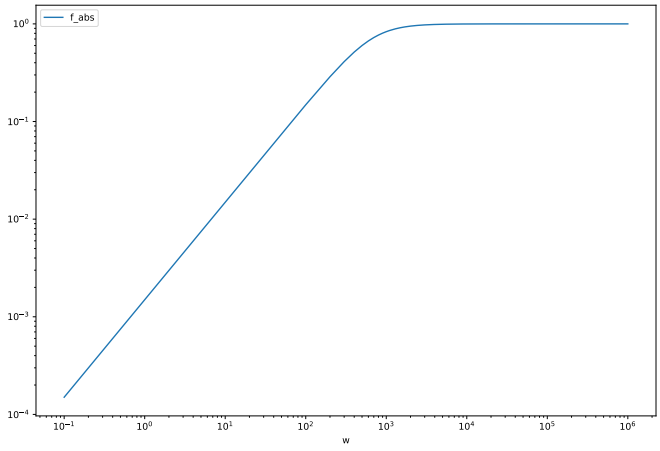



Figura 11.5: Graph of the Modulus of T(s)

## 11.4 Expansion into Partial Fractions

A very common problem in signal analysis and in solving systems in the frequency domain is that of expanding a rational function into a sum of partial fractions. In this type of problem we want to transform an expression like :

$$\frac{s^3 + 3\,s^2 + 3}{s^5 + 2\,s^4 + 7\,\mathbf{s}^3 + 8\,s^2 + 9\,s + 4} \tag{11.11}$$

in a series of terms of the type :

$$\frac{A_1}{s + r_1} + \frac{A_2}{s + r_2} + \frac{A_3}{s + r_3} + \frac{A_4}{s + r_4} + \frac{A_5}{s + r_5} \tag{11.12}$$

where the $r_n$ are the roots of the denominator of 11.11 and the values $A_n$ must be calculated. This can be done by `scipy` with the `residue` function (signal processing module named `signal`) as can be seen below:

```
from scipy.signal import residue
numer = [1,3,0,3]
denom = [1,2,7,8,9,4]
r, p, k = residue(numer, denom)
```

`r` is the residue itself.

```
r
```

```
array([ 0.653+0.j   , -0.253+0.461j, -0.253-0.461j, -0.074-0.561j,
       -0.074+0.561j])
```

p“ are the poles, ordered ascending, corresponding to each residue

```
p
```

```
array([-0.625+0.j   , -0.394+1.181j, -0.394-1.181j, -0.294+2.01j ,
       -0.294-2.01j ])
```

‘`k` is the direct polynomial term (if the order in the numerator is greater than that in the denominator). In this case, the term `k` is an empty list.

```
k
```

```
array([], dtype=float64)
```

So the partial fraction expansion in this example equals :

$$\begin{aligned}\frac{s^3 + 3\,s^2 + 3}{s^5 + 2\,s^4 + 7\,s^3 + 8\,s^2 + 9\,s + 4} = \frac{0,653}{s + 0,625} + \frac{-0,253 + 0,461j}{s + 0,394 - 1,181j} + \ldots \\ \ldots + \frac{-0,253 - 0,461j}{s + 0,394 + 1,181j} + \frac{-0,074 - 0,561j}{s + 0,294 - 2,010j} + \frac{-0,074 + 0,561j}{s + 0,294 + 2,010j}\end{aligned} \tag{11.13}$$

## 11.5 Inversion to the Time Domain

Considering that the expansion into partial fractions has already occurred and we now wish to obtain the equivalent function in the time domain $t$ we must proceed with the inversion, or anti-transformation from the function in $s$ to the function in $t$. So we assume that the inversion occurs for a function of variable $s$ which was obtained by the Laplace Transform of a function at $t$. This task is done using the basic pair of Transforms

$$f(t) = e^{at} \longleftrightarrow F(s) = \frac{1}{s-a} \tag{11.14}$$

Let's then apply this basic idea to the case of the 11.13 equation. By inspection, the first term of the expansion would become :

$$\frac{0.653}{s+0.625} \longleftrightarrow 0.653e^{-0.625t} \tag{11.15}$$

As for terms with complex values, to make the inversion faster we can realize the following :

$$\frac{a+b\jmath}{s-re-im\jmath} + \frac{a-b\jmath}{s-re+im\jmath} \rightarrow (a+b\jmath).e^{re+im\jmath} + (a+b\jmath).e^{re-im\jmath} \tag{11.16}$$

Separating the real terms (without $\jmath$) from the imaginary terms (with $\jmath$):

$$\begin{aligned} (a+b\jmath).e^{re+im\jmath} + (a-b\jmath).e^{re-im\jmath} = \\ 2.a.e^{re}.(\frac{e^{im\jmath}+e^{-im\jmath}}{2}) + 2.b.\jmath.e^{re}.(\frac{e^{im\jmath}-e^{-im\jmath}}{2}) = \\ 2.a.e^{re}.(\frac{e^{im\jmath}+e^{-im\jmath}}{2}) - 2.b.e^{re}.(\frac{e^{im\jmath}-e^{-im\jmath}}{2.\jmath}) = \\ 2.a.e^{re}.\cos(im.t) - 2.b.e^{re}.\sin(im.t) = \\ 2.e^{re}.[a.\cos(im.t) - b.\sin(im.t)] \end{aligned} \tag{11.17}$$

In this case we assumed the residue $a+b\jmath$ associated with the root $re+im.\jmath$. If this residue is associated with the root $re-im.\jmath$, simply change the sign from $-b$ to $+b$ in the equation 11.17

In this way, the transform pairs will be :

$$\begin{aligned} \frac{-0,059+0,229\jmath}{s+0,34-0,953\jmath} + \frac{-0,059-0,229\jmath}{s+0,34+0,953\jmath} \rightarrow \\ 2.e^{-0,34.t}.[-0,059.\cos(0,953.t) - 0,229.\sin(0,953.t)] \end{aligned} \tag{11.18}$$

In this case (equation 11.18) we had the $-$ sign in the denominator because the residue $-0.059 + 0.229\jmath$ was associated with the root $-0.34 + 0.953\jmath$. In the following case (equation 11.19), the residue $-0.144 - 0.232\jmath$ is associated with the root $-0.585 + 2.751\jmath$, so the sign of the denominator will be $+$. So we have :

$$\frac{-0,144 - 0,232\jmath}{s + 0,585 - 2,751\jmath} + \frac{-0,144 + 0,232\jmath}{s + 0,585 + 2,751\jmath} \rightarrow$$
$$2.e^{-0,585.t}.[-0,144.\cos(2,751.t) + 0,232.\sin(2,751.t)] \tag{11.19}$$

## 11.6 Exercises

3. Expand into partial fractions the following polynomial quotients:
   (a) $\frac{2\,s^2+6\,s}{s^3+8\,s^2+10\,s+4}$
   (b) $\frac{x^2+3\,x+2}{x^3+5\,x^2+10{,}5\,x+9}$
   (c) $\frac{x^2-16}{x^3+8\,x^2+24\,x+32}$
   (d) $\frac{x+1}{x^3+6\,x^2+11\,x+6}$

# Capítulo 12

# Quantum Machine Learning

## 12.1 A Poetic Introduction

Dear coleagues, it's been a while since I started reading about Quantum Computing. In the mean time, I have also developed some courses on what is called now Machine Learning. Ten, fifteen years ago Machine Learning was called Statistics. Then, the CSVs files started becoming huge, with millions of rows and hundreds, thousands of columns. Statistics became Big Data.

Some statistics rules for selecting which column (variable) to use, like p-value became difficult to be explained to customers, specially executives, which need something that can be quickly learned, easily remembered and effortlessly applied on a daily basis.

P-value also started giving strange answers when used in these colossal tables (after all, if 5% of your data is a 100.000 rows with 100 columns, who knows what will appear in this subset of your data???).

It was at this moment that somebody "remembered" that p-values had been developed originally with way much more simpler tables, tables in which you were forced to use all the rows and columns in the development of your model. In this context p-values were a way of estimating the other data that was "out there." But with tables made of millions of rows you could just separate 500.000 of them to be the "world out there," use the 4.500.000 to develop (train) your model and afterwards see ("test") what would happen in the real world when your model was put to work. Statistics that had became Big Data was now renamed also to Machine Learning.

The algorithms were still being used to model simple phenomena. Phenomena that could be described with straight lines. Science, the scientific revolution started 500 years ago when a group of geniuses tried to apply simple rules to simple phenomena. It worked for quite a while (astonishing well), but the rule

was always: simple phenomena that can be described preferably with straight lines, at most with a parabola.

The world is incredibly complex, but nonetheless our minds are capable of dealing with all kinds of phenomena, be they describable by straight lines or not. And then somebody tried to apply a method of finding non linear patterns in the data called neural networks. Statistics, now mingled with computer science became Deep Learning.

But all this has a cost. The cost of dealing with millions of rows, thousands of columns, to train, validade and test our models. And this cost is paid in the most precious commoditty that a human being can spend: TIME! Our computers are becoming overwhelmed by this combination of Big Data, Machine Learning and Deep Learning that has accomplished a lot of things but needs an ever increasing ammount of data to be processed.

It was then that another technology appeared. A technology that is still in its infancy, but is promising us with a future when computers will be able to digest an unthinkable ammount of data in a matter of seconds, process it and tell us where the results are. This technology is called Quantum Computing.

What is Quantum Computing and how it can be used in the Machine Learning way of finding the utmost complex patterns in mountains of data is the purpose of a field now called Quantum Machine Learning.

This field blends together two of my mostly cherished areas of study: Computing and Physics (and of course, the possibility of making some money along the line, after all, bills doesn't stop arriving each month for us to have them paid...). And this group of Jupyter Notebooks is my attempt to make this area of study better know by as many people as possible.

Quantum Machine Learning. This is the Holly Grail of my knowledge dreams. Understand this dream, make it available to others, is the objective of this and other Jupyter Notebooks that I will try writing in the near future.

In this journey, Python will be our driver, and Qiskit our compass. The destiny: translating to the world of Quantum Computing the Machine Learning algorithms that have made us devise a future with Intelligent Machines at our disposal.

Hope you enjoy the ride. It will be bumpy, sometimes frustating, but if we keep our focus and always remember why we are here, I promise you that there will be a pot of gold at the end of the rainbow waiting for us.

Happy Journey! And Let Quantum Machine Learning BE!

## 12.2 Machine Learning

Machine learning is the process of tagging data in a data frame. The first thing that needs to be choosen is a target variable. Depending on whether you have

examples of your target variable or not, the task can fall in one of two main categories:

1. **Supervised Learning**: when you have past examples of your input and output data. Depending on whether your output variable is categorical (discrete) or continuous, the supervised learning task can be called:
   (a) **Classification**: when your output variable is discrete (categorical). Logistic regression is the main example of classification technique.
   (b) **Regression**: when your output variable is continuous. Linear regression is the main example.
2. **Unsupervised Learning**: when you don't have past examples of your output variable or not even know what it could be. In this case you want the machine to group your data so you can try finding some pattern in it. Depending on wether you want to group the rows or the columns of you data frame, this task can be called:
   (a) **Clustering**: when you want the machine to tag (group) the rows of your data frame. This tagging can become in the future your output variable to be used in a supervised learning classification problem.
   (b) **PCA**: when you want the machine to tag (group) the columns of your data frame. In this case you want a new column that will replace some of the original ones through a simple formula.

The process of solving a problem using Machine Learning usually involves five steps:

1. **Problem Definition**: decide which variable will be predicted. This will be called your output variable.
2. **Data Preparation**: decide which variables you need to try predicting your output variable. These variables will be called input variables. In this step you will also find examples of your variables and have them cleaned and ready to be used.
3. **Model Development**: Based on your output variable type availability you will decide if you're going to develop a classification, regression, clustering or PCA model.
4. **Results Analysis**: When you link the math results from trainning and testing the model developed in stage 3 with your business objectives and present your results for both a technical and a business oriented audience.
5. **Model Deployment**: When your model goes live. Here usually you deal with problems of interfacing your model with real people (managerial issues) or with other systems in your client (a task usually called data engineering).

## 12.3 Quantum Computing

The idea behind quantum computing is based in three concepts:

1. **Superposition of states**: the Qubit is neither in the $|0\rangle$ or $|1\rangle$ state but in a mixed state called superposition of states. Bare in mind that the notion that it is at the same time in the $|0\rangle$ and $|1\rangle$ state is wrong. The system is in a state that is formed from the basis states $|0\rangle$ and $|1\rangle$.

2. **Interference**: is the possibility of making states interact with one another (or with themselves) in such a way that the desired pattern (the one that holds the answer for our problem) is reinforced while the other patterns are cancelled out.

3. **Entanglement**: an extremely strong correlation between states that cannot happen in classical physics. Because of it, two particles when they interact with one another remain entangled ("amarradas"in portuguese) after that. This entangled state means that their states are grouped in such a way that they cannot be represented individually.

The concepts presented above are further elaborated on below:

1. **Dirac notation**: through this we can represent the superposition of $n$ states and amplitudes in a $\psi$ system as follows: $\psi = c_0|x_0\rangle + c_1|x_1\rangle + ... + c_{n-1}|x_{n-1}\rangle$

2. **Wave Function Collapse**: we can only know "what is going on"with a subatomic particle (a photon of light for example) if we try to measure its state. At this point the particle will "respond"to us with any of the possible $x_i$ states above. At this moment the particle "stops"its superposition and shows us a momentary photograph of itself. We call this phenomenon the collapse of the wave function associated with the particle.

3. **Amplitudes** $c_0$ ... $c_{n-1}$ and **probabilities of states** $x_0$ ... $x_{n-1}$ **of a particle**: Each measurable state $x_0$ ... $x_{n-1}$ has an amplitude $c_i$ associated with it. This amplitude can be any complex number, with any value (negative, null, or positive) in either its real or imaginary part.

The probability that we get as a response to a measurement of the state of a particle the value $x_i$ is associated with the product $c_i.c_i^*$ associated with the amplitude $c_i$ of the state $x_i$.

Since the numerical value $c_i$ can be any complex number, we must remember that $c_i^*$ is its complex conjugate (the number obtained by changing the imaginary part of it from sign to sign). For example if $c_i = 3 + 4i$, its complex conjugate $c_i^*$ will be equal to $c_i^* = 3 - 4i$.

But notice that the product $c_i.c_i^*$ will always be a real number. If we assume that the sum of all products $c_i.c_i^*$ for $i$ going from 0 to $n-1$ is equal to 1, we can say that each product $c_i.c_i^*$ will represent the probability of getting the associated state $x_i$ at some measure.

All the "strangeness" of Quantum Mechanics stems from the possibility that these $n$ amplitudes $c_i$ can be made up of any possible number. Being able to mix positive and negative values we can obtain as a result for a combination of distinct $c_i$s even the value 0. This means that the state $x_i$ associated to the amplitude $c_i$ will have 0% probability of being measured, which means that we will not measure it. We therefore say that that result "disappeared" from our measurements, in other words, in that situation the particle "disappeared." Since such a sum with a null result can only appear due to the interaction of other states, we say that a phenomenon of **interference** of the particle has occurred. But we are led to conclude that, since the particle's states interact with each other (while it is in superposition), the particle is interacting with itself.

From the philosophical point of view, this can generate (or rather has generated for more than a hundred years) a huge confusion, with great names of science trying to turn their thoughts around and performing real mental juggling to find ways to explain such phenomena. For our part, what interests us is that the superposition and its consequences can be easily analyzed through the machinery of complex numbers and linear algebra. We will follow here the *philosophical* line very well described by a sentence pronounced by the physicist David Mermin to one of his students when he began to generate the famous unanswered questions that every Quantum Mechanics professor has become accustomed to answering with a puzzled smile. The phrase simply was: *Shut Up and Calculate.*

By the style of it, it has been attributed to Nobel Prize-winning physicist Richard Feynman. He did not utter it, but to paraphrase the words of a great Brazilian humorist Dias Gomes, creator of the unforgettable mayor of Sucupira, Odorico Paraguassu: "If he didn't say it, he should have said it.

From this moment on, our elementary particles will start to constitute abstract entities called **Qubits**. Such elements will have the ability to enter into superposition of their fundamental states $x\_i$ and will be removed from it when they are measured, at which time they will present us as a result one of the possible $x_i$. The probability that we will measure the result $x_i$ will be given by the product $c_i.c_i^*$. We will only be interested in situations where the sum of all products $c_i.c_i^*$ will give 1 (100%).

4. **NISQ: Noisy Intermediate Scale Quantum Computing**: Quantum computing today is based in a technique called **NISQ**. This technology uses both classical and quantum computers. Classical computers handle the machine learning problem the same way they usually do. The task of "crunching"the numbers (get the parameters for the machine learning model and having it tested) is handled by the quantum counterpart. The algorithms that handle the problem this way are called **Variational Quantum-Classical Algorithms**

## 12.4 Quantum States

The representation of the superposition of the amplitudes and states of a $\$\psi$ system with $n$ possible states is shown below:

$$\psi = c_0|x_0\rangle + + + c_1|x_1\rangle \ + \ +... + \ + \ + c_{n-1}|x_{n-1}\rangle$$

1. Calculate the probability that the following systems, when measured, indicate the state $|1\rangle$

```
import numpy as np
p_e = lambda vector,x : (np.linalg.norm(vector[x])/np.linalg.norm(vector))**

psi = [1/np.sqrt(3), np.sqrt(2/3)]
phi = [1j/2, np.sqrt(3)/2]
chi = [(1+1j)/np.sqrt(3), -1j/np.sqrt(3)]

[float(x.__format__('.3f')) for x in [p_e(psi,1), p_e(phi,1), p_e(chi,1)]]
```

```
[0.667, 0.75, 0.333]
```

2. What is the probability that the system Phi is in the $3^th$ state?

```
import numpy as np
import sympy as sp
Phi = sp.Matrix([-3 - sp.I, -2*sp.I, sp.I, 2])
num1 = Phi[2] * Phi[2].conjugate()
den1 = sp.sqrt(sum(sp.matrix_multiply_elementwise(Phi,Phi.conjugate())))
P = sp.simplify((num1/den1)**2)
P
```

3. Given the amplitudes $c_{up} = 3 - 4j$ and $c_{down} = 7 + 2j$ applied to an orthonormal basis, calculate the probabilities associated with each of the amplitudes.

```
import numpy as np
c_up = 3-4j
c_down = 7+2j
ket = np.array([c_up,c_down]).reshape(-1,1)
p_up = np.linalg.norm(c_up)**2/np.linalg.norm(ket)**2
p_down = np.linalg.norm(c_down)**2/np.linalg.norm(ket)**2
[float(x) for x in [p_up.__format__('.3f'), p_down.__format__('.3f')]]
```

```
[0.321, 0.679]
```

4. Present the *ket* representation of the $\psi$ vector with values $2 - i, 2i, 1 - i, 1, -2i, 2$

```
import numpy as np
import sympy as sp
psi = sp.Matrix([2 - sp.I, 2*sp.I, 1 - sp.I, 1, -2*sp.I, 2])
expr = "| \\rangle =" + sp.latex(psi)
expr
```

```
'| \\rangle =\\left[\\begin{matrix}2 - i\\\\2 i\\\\1 - i\\\\1\\\\- 2 i\\\\2\\
```

5. Present the *bra* representation of the vector $\psi$ with values $2-i, 2i, 1-i, 1, -2i, 2$

```
psiT = sp.conjugate(sp.Matrix.transpose(psi))
expr = "\langle \\psi | =" + sp.latex(psiT)
expr
```

```
'\\langle \\psi | =\\left[\\begin{matrix}2 + i & - 2 i & 1 + i & 1 & 2 i & 2
```

6. Present the *ket* representation of the $\psi$ vector with values $1/2, -i/2, \sqrt{2}/2$, the associated *bra* representation, and the product of the *bra* representation with the $A$ matrix (see below) and the *ket* representation.

```
psi_ket = sp.Matrix([[sp.Rational(1,2)],
                     [-sp.I/2],
                     [1/sp.sqrt(2)]])
psi_ket
```

```
Matrix([
[      1/2],
[     -I/2],
[sqrt(2)/2]])
```

```
psi_bra = psi_ket.adjoint()
psi_bra
```

```
Matrix([[1/2, I/2, sqrt(2)/2]])
```

```
A = sp.Matrix([[0,1/sp.sqrt(2),0],
               [1,1/sp.sqrt(2),0],
               [0,0,0]])
psi_bra, A, psi_ket, (psi_bra * A * psi_ket).expand(complex=True)
```

```
(Matrix([[1/2, I/2, sqrt(2)/2]]), Matrix([
[0, sqrt(2)/2, 0],
[1, sqrt(2)/2, 0],
[0,         0, 0]]), Matrix([
```

```
[       1/2],
[     -I/2],
[sqrt(2)/2]]), Matrix([[sqrt(2)/8 - sqrt(2)*I/8 + I/4]]))
```

7. Calculate the modulus of the vector $\psi$

```
expr = sp.simplify(psiT.multiply(psi))
expr
```

```
Matrix([[20]])
```

8. Prove that the vectors $||psi_1\rangle = [1+i, i]^T$ and $|psi_2\rangle = [2+4i, 3i-1]^T$ differ by the factor $3+i$

```
psi1 = np.array([1+1j,1j]).reshape(2,1)
psi2 = np.array([2+4j,3j-1]).reshape(2,1)
psi2/psi1
```

```
array([[3.+1.j],
       [3.+1.j]])
```

9. Can the vectors $|\psi_1\rangle = [1+i, 2-i]^T$ and $|\psi_2\rangle = [2+2i, 1-2i]^T$ represent the same state? Ans: No, because their components differ by different values.

```
psi1 = np.array([1+1j,2-1j]).reshape(-1,1)
psi2 = np.array([2+2j,1-2j]).reshape(-1,1)
psi2/psi1
```

```
array([[2. +0.j ],
       [0.8-0.6j]])
```

10. What is the length of the vector $|\psi\rangle = [2-3i, 1+2i]^T$?

```
psi1 = np.array([2-3j,1+2j]).reshape(-1,1)
print(np.sqrt(psi1.conjugate().T.dot(psi1)[0,0]).real)
```

```
4.242640687119285
```

```
np.linalg.norm(psi1)
```

```
4.242640687119285
```

11. Normalize ket $|\psi\rangle = [3-i, 2+6i, 7-8i, 6.3+4.9i, 13i, 0, 21.1]^T$

```
psi1 = np.array([3-1j, 2+6j, 7-8j, 6.3+4.9j, 13j, 0, 21.1]).reshape(-1,1)
psi1_norm = (psi1 / np.linalg.norm(psi1))
psi1_norm
```

```
array([[0.103-0.034j],
       [0.069+0.207j],
       [0.241-0.276j],
       [0.217+0.169j],
       [0.   +0.448j],
       [0.   +0.j   ],
       [0.728+0.j   ]])
```

12. Verify that the two states $x_1 = [\frac{sqrt2}{2}, -\frac{sqrt2}{2}]^T$ and $x_2 = [\frac{sqrt2}{2}, -\frac{sqrt2}{2}]^T$ have modulus equal to 1 and give the vector that represents the overlap of these states.

The vector is: $|\psi\rangle = c_1|x_1\rangle + c_2|x_2\rangle$

```
import numpy as np
x1 = np.array([np.sqrt(2)/2,np.sqrt(2)/2]).reshape(-1,1)
x2 = np.array([np.sqrt(2)/2,-np.sqrt(2)/2]).reshape(-1,1)
print(np.linalg.norm(x1), np.linalg.norm(x2))
```

```
1.0 1.0
```

13. Enter the Hermitian matrix of $\Omega$ shown below

```
omega = np.array([[-1,-1j],
                  [1j, 1]])
sp.Matrix(omega)
```

```
Matrix([
[ -1.0, -1.0*I],
[1.0*I,    1.0]])
```

```
sp.Matrix(np.matrix(omega).getH())
```

```
Matrix([
[ -1.0, -1.0*I],
[1.0*I,    1.0]])
```

14. Calculate the resultant vector of the action of the $\Omega$ operator on $\psi$

```
psi = np.array([-1+0j,-1-1j])
omega = np.array([[-1,-1j],
                  [1j, 1]])
```

```
expr = "\Omega =" + sp.latex(sp.Matrix(omega))
expr = expr + sp.latex('\psi =') + sp.latex(sp.Matrix(psi))
expr = expr + sp.latex('\Omega . \psi =') + sp.latex(sp.Matrix(omega.dot(psi
expr
```

```
'\\Omega =\\left[\\begin{matrix}-1.0 & - 1.0 i\\\\1.0 i & 1.0\\end{matrix}\\
```

15. Calculate the following operations

```
X = sp.Matrix([[0,1],[1,0]])
psi_bra = sp.Matrix([[1/sp.sqrt(3),sp.sqrt(sp.Rational(2,3))]]) # sp.Rationa
psi_ket = psi_bra.T
psi_bra * X * psi_ket
```

```
Matrix([[2*sqrt(2)/3]])
```

```
exp_X = psi_bra * X * psi_ket
exp_X2 = psi_bra * X * X * psi_ket # X squared expectation
exp_X_2 = exp_X ** 2 # X expectation squared
desv_X = sp.sqrt(exp_X2[0] - exp_X_2[0]) # X standard deviation
exp_X2, exp_X_2, desv_X
```

```
(Matrix([[1]]), Matrix([[8/9]]), 1/3)
```

16. Using the dynamics given in Equation (3.4), determine what the state of the system would be if you start with the state $[5,5,0,2,0,15]^T$. The Dinamics is given by matrix M below:

```
import sympy as sp
M = sp.Matrix([[0,0,0,0,0,0],
               [0,0,0,0,0,0],
               [0,1,0,0,0,1],
               [0,0,0,1,0,0],
               [0,0,1,0,0,0],
               [1,0,0,0,1,0]])
X = (sp.Matrix([5,5,0,2,0,15]))
M
```

```
Matrix([
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0],
[0, 1, 0, 0, 0, 1],
[0, 0, 0, 1, 0, 0],
[0, 0, 1, 0, 0, 0],
[1, 0, 0, 0, 1, 0]])
```

```
X
```

```
Matrix([
[ 5],
[ 5],
[ 0],
[ 2],
[ 0],
[15]])
```

```
M*X
```

```
Matrix([
[ 0],
[ 0],
[20],
[ 2],
[ 0],
[ 5]])
```

17. For the matrix M given in Equation (3.4), calculate $M^6$. If all the marbles start at vertex 2, where will all the marbles end up after 6 time steps?

```
Mn = lambda M,n : ["M"+str(n)+" =", M**n]
n = 6
expr1 = "M^{} =".format(n) + sp.latex(M**n)
expr2 = "X =" + sp.latex(X)
expr3 = "M^{}.X =".format(n) + sp.latex((M**n)*X)
expr = expr1 + expr2 + expr3
expr
```

```
'M^6 =\\left[\\begin{matrix}0 & 0 & 0 & 0 & 0 & 0\\\\0 & 0 & 0 & 0 & 0 & 0\\\
```

18. Consider the following graph representing city streets. Singleheaded arrows (->) correspond to one-way streets and double-headed arrows (<–>) correspond to two-way streets. Imagine that it takes one time click to traverse an arrow. You may assume that everyone must move at every time click. If every corner starts with exactly one person, where will everyone be after one time click? After two time clicks? After four time clicks?

```
M = sp.Matrix([[0,1,0,0,0,0,0,0,0],
               [1,0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0,0],
               [0,0,0,1,0,0,0,0,0],
               [0,0,0,0,0,0,0,0,0],
```

```
                [0,0,1,0,0,0,0,0,0],
                [0,0,0,0,0,0,0,0,0],
                [0,0,0,0,0,0,1,0,0],
                [0,0,0,0,0,0,1,0,1]])
X = sp.Matrix([1,1,1,1,1,1,1,1,1])
expr1 = M*X
expr2 = (M**2)*X
expr4 = (M**4)*X
expr = "X_1 =" + sp.latex(expr1) + "X_2 =" + \
       sp.latex(expr2) + "X_4 =" + sp.latex(expr4)
expr
```

```
'X_1 =\\left[\\begin{matrix}1\\\\1\\\\0\\\\1\\\\0\\\\1\\\\0\\\\1\\\\3\\end{m
```

19. Suppose the state transition matrix M shown below. Compute the element-to-element product of the matrix M with its conjugate transpose.

```
import sympy as sp
M = sp.Matrix([[           0,                    0,0,0,0,0,0,0],
               [1/sp.sqrt(2),                    0,0,0,0,0,0,0],
               [1/sp.sqrt(2),                    0,0,0,0,0,0,0],
               [           0,(-1+sp.I)/sp.sqrt(6),
0,1,0,0,0,0],
               [           0,(-1-sp.I)/sp.sqrt(6),
0,0,1,0,0,0],
               [           0,(+1-sp.I)/sp.sqrt(6),(-1+sp.I)/sp.sqrt(6),0,0,1
               [           0,                    0,(-1-sp.I)/sp.sqrt(6),0,0,0
               [           0,                    0,(+1-sp.I)/sp.sqrt(6),0,0,0

X1 = sp.Matrix([1,0,0,0,0,0,0,0])
expr = "M =" + sp.latex(M) + "X_1 =" + sp.latex(X1)
expr
```

```
'M =\\left[\\begin{matrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\\\\\frac{\\sqrt{2}}
```

```
P1 = sp.simplify(sp.matrix_multiply_elementwise(M,M.conjugate()))
expr = "P_1 =" + sp.latex(P1)
expr
```

```
'P_1 =\\left[\\begin{matrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\\\\\frac{1}{2} & (
```

```
M2 = sp.simplify(M*M)
expr = "M_2 =" + sp.latex(M2)
expr
```

```
'M_2 =\\left[\\begin{matrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\\\0 & 0 & 0 & 0 &
```

## 12.5 Interference and Overlapping

If the weights are real numbers, the probabilities associated with them can only increase when they are added.

In contrast, if the weights are complex numbers, the probabilities are associated with their modulus squared. Therefore if you add two weights that are complex numbers, the probability can either increase, add up to 0 or decrease.

When two complex numbers are added and cancel each other (they add up to 0) this is referred to as **interference**.

Let the state of the system be given by $X = [c_0, c_1, ..., c_{n01}], T \in C_n$. It is incorrect to say that the probability of the photon's being in position $k$ is $|c_k|$

Rather, to be in state $X$ means that the particle is in some sense in all positions simultaneously. The photon passes through the top slit and the bottom slit simultaneously, and when it exits both slits, it can cancel itself out. A photon is not in a single position, rather it is in many positions, a **superposition**.

Matrix below shows the probability of starting in column $j$ a photon will end up in row $i$ after 2 time periods (or 2 ticks of clock).

For instance, if a photon is in state 0 (first column), after two time periods it will end up in state 3 (forth row) with probability $\frac{1}{6}$.

Now look at the element $P_2[5, 0]$. It is equal to 0. This means that the probability of a photon being in state 0 in instant 0 and in state 5 in instant 2 is 0, so there will be no photon at all there.

This is caused by **interference**.

The difficult thing to understand (to accept would be a better term) is that it is wrong to say that there's a probability for the photon to be in a state. In reality the photon is itself in all states, and each state has a definite probability associated with it.

Probability of what? Of us, when trying to measure where the photon "is" will find it in that state at that time.

BUT, when we're not looking (i.e.trying to measure it), photon is in a **superposition** of states. In other words: it is simultaneously in all states at the same time!

A classical computer can be in one state at a time (0 or 1). A quantum computer uses the superposition phenomenon to be in all classical states at the same time. In other words: a quantum computer is the ultimate idea in parallel processing.

Both matter and light manifest a particle-like and a wave-like behavior

## 12.6 Interconexão de Sistemas

Assembling Systems: Compute the tensor product of M with N. What does it represent in terms of quantum circuits?

```
import sympy as sp
from sympy.physics.quantum import TensorProduct
M = sp.Matrix([[0,                sp.Rational(1,6),sp.Rational(5,6)],
               [sp.Rational(1,3),sp.Rational(1,2),sp.Rational(1,6)],
               [sp.Rational(2,3),sp.Rational(1,3),
0]])

N = sp.Matrix([[sp.Rational(1,3),sp.Rational(2,3)],
               [sp.Rational(2,3),sp.Rational(1,3)]])
MxN = TensorProduct(M,N)
expr1 = sp.latex('M = ') + sp.latex(M)
expr2 = sp.latex('N = ') + sp.latex(N)
expr3 = sp.latex('M x N = ') + sp.latex(MxN)
expr = expr1 + expr2 + expr3
expr
```

```
'\\mathtt{\\text{M = }}\\left[\\begin{matrix}0 & \\frac{1}{6} & \\frac{5}{6}
```

## 12.7 O Dataset Bancalvo

The following database will be used to exemplify the application of Quantum Computing to solving Machine Learning problems

```
site = "https://github.com/"
diretorio = "gustavomirapalheta/classes_datasets/blob/master/"
arquivo = "bancalvo.xlsx?raw=true"
link = site + diretorio + arquivo
train = pd.read_excel(link)

lista = list(train['UF'].unique())
lista.remove('SP')
train['novaUF'] = train['UF'].replace(lista, 'NOSP')
train['lRENDA'] = np.log10(train['RENDA'])
train = train.drop(['IDENTIDADE','UF','RENDA'], axis=1)
train = pd.get_dummies(train, drop_first=True)
X = train.drop('STATUS_inad', axis=1).copy()
y = train['STATUS_inad'].copy()

from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, random_state = 53, tes
Xtrain.shape, Xtest.shape, ytrain.shape, ytest.shape
```

# Bibliografia


Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2021. *Rmarkdown: Dynamic Documents for r.* https://CRAN.R-project.org/package=rmarkdown.

Canale, Raymond P., and Steven C. Chapra. 2016. *Métodos Numéricos Para Engenharia.* 5a ed. São Paulo: McGraw-Hill.

Demidovitch, B. 1974. *Problems in Mathematical Analysis.* 2nd ed. Moscow: MIR Publishers.

Dirac, P. A. M. 1957. *The Principles of Quantum Mechanics.* 4th ed. USA: Snowball Publishing. www.snowballpublishing.com.

McMahon, David. 2007. *Quantium Computing Explaneid.* 1st ed. Wiley-IEEE Computer Society Pr. www.ieee.org.

Nielsen, Michael A., and Isaac L. Chuang. 2016. *Quantum Computation and Quantum Information.* 4th ed. Cambridge, UK: Cambridge University Press. www.cambridge.org.

Strang, Gilbert. 2009. *Computational Science and Engineering.* 1st ed. Wellesley-Cambridge Press. www.cambridge.org.

Xie, Yihui. 2021. *Knitr: A General-Purpose Package for Dynamic Report Generation in r.* https://yihui.org/knitr/.