Thematic exercise 2.

The *fundamental theorem of algebra* states that any polynomial $p(Z) \in \mathbb{C}[Z]$ of degree at least one has a root $\lambda \in \mathbb{C}$ (see Theorem 4.6.1 from the textbook). For practical applications it is important to be able to find such a root or at least to find a good numerical approximation. The goal of this thematic exercise is to obtain algorithmic insights on how numerical approximations of roots of polynomials with real coefficients can be found. We explain two methods: the bisection method and the Newton-Raphson method.

Part I: the bisection method

First, open a command line version of Python.

1. To work with roots of polynomials $p(X) \in \mathbb{R}[X]$, we define the function that such a polynomial gives rise to: $p : \mathbb{R} \to \mathbb{R}$, where $x \mapsto p(x)$. The first step is to figure out how to define such a function in Python.

   (a) As an example, consider the polynomial $p(X) = X^3 - X - 6$. The corresponding function $p : \mathbb{R} \to \mathbb{R}$ can be defined in Python as follows:

   ```
   def p(x):
    return x**3 - x - 6
   ```

   Type this code in command line Python (remember to indent the second line). After entering the first line, the prompt will change from `>>>` to `...` After entering the second line, the prompt will still look like `...` , but after pressing the Return-key, the prompt will change back to `>>>`. When it does, you have finished entering the function in Python.

   (b) Now you can compute values of the function p using Python. Type for example $p(10)$ and check by hand that Python returns the correct value.

   (c) Use Python to compute $p(-1)$, $p(0)$, $p(1)$, $p(2)$ and $p(3)$. Can you now indicate a root of the polynomial $p(X)$?

In general, one will not be so lucky to find a root of a polynomial $f(X)$ simply by computing a few of its values. Therefore, we now study a criterion that sometimes can be used to at least determine where a root is located approximately. We will start by formulating a theorem for continuous functions. We will not define what a continuous function is precisely, but very loosely speaking a function is continuous if its graph does not have any jumps in it. You may use freely in this thematic exercise that any function $f : \mathbb{R} \to \mathbb{R}$ such that $x \mapsto f(x)$, where $f(X) \in \mathbb{R}[X]$ is a polynomial, is continuous. Continuous function have nice properties, among other the following theorem, which is known as the *intermediate value theorem*.

Theorem 1 *Let a, b be real numbers satisfying $a < b$. Further, let $f : [a, b] \to \mathbb{R}$ be a continuous function.*

- *If $f(a) < f(b)$ and y satisfies $f(a) < y < f(b)$, then $y = f(x)$ for some x in the interval $[a, b]$.*
- *If $f(a) > f(b)$ and y satisfies $f(b) < y < f(a)$, then $y = f(x)$ for some x in the interval $[a, b]$.*

We will not prove this theorem, but you may use it freely. If $f : \mathbb{R} \to \mathbb{R}$ is a function and $\lambda \in \mathbb{R}$ satisfies $f(\lambda) = 0$, then we call λ a zero of f. If f is defined from a polynomial $f(X)$, then a zero of f is nothing but a root of the polynomial $f(X)$. The intermediate value theorem has a useful consequence concerning zeroes of continuous functions:

Corollary 2 *Let a, b be real numbers satisfying $a < b$. Further, let $f : [a, b] \to \mathbb{R}$ be a continuous function satisfying $f(a) \cdot f(b) < 0$. Then there exists $x \in [a, b]$ such that $f(x) = 0$. In other words: the function f has a zero in the interval $[a, b]$.*

2. The aim of this question is to understand why Corollary 2 is a logical consequence of Theorem 1.

   (a) First of all, show that $f(a) \cdot f(b) < 0$ implies that
   $$f(a) < 0 \ \wedge \ f(b) > 0 \quad \text{or} \quad f(a) > 0 \ \wedge \ f(b) < 0.$$

   (b) Conclude that if $f(a) \cdot f(b) < 0$, then 0 is a value between $f(a)$ and $f(b)$. In other words: conclude that either $f(a) < 0 < f(b)$ or $f(b) < 0 < f(a)$.

   (c) Now apply Theorem 1 to conclude that f has a zero in the interval $[a, b]$.

3. The polynomial $p(X) = X^3 - X - 6$ was on purpose chosen in such a way that it had the "nice" root 2. We now choose a different polynomial without such nice roots.

   (a) Consider the polynomial $f(X) = X^3 - X - 5$ and define in Python the corresponding function $f : \mathbb{R} \to \mathbb{R}$, that is the function such that $x \mapsto x^3 - x - 5$.

   (b) Use Python to compute $f(x)$ for $x \in \{0, 1, 2, 3\}$. Conclude using Corollary 2 and the fact that $f(1) < 0$ and $f(2) > 0$, that the polynomial $f(X) = X^3 - X - 5$ has a root in the interval $[1, 2]$. As mentioned previously, you may use freely that a polynomial function is continuous.

   (c) Given an interval $[a, b]$, one calls the value $(a + b)/2$ the *midpoint* of that interval and the value $b - a$ the *width* of the interval. For example, the interval $[1, 2]$ has midpoint $(1+2)/2 = 1.5$ and width $2-1 = 1$. Now compute $f(x)$ in the midpoint of the interval $[1, 2]$. Does the polynomial $X^3 - X - 5$ have a root in the interval $[1, 1.5]$ or in the interval $[1.5, 2]$? Note that these intervals have width $1/2$, that is to say, half the width of the interval $[1, 2]$.

   (d) Considering the value $f(x)$ for the midpoint of the interval you just found, determine an interval of width $1/4$ containing a root of the polynomial $X^3 - X - 5$.

   (e) One can in principle continue the procedure from 3 (d), each time finding an interval containing a root of $X^3 - X - 5$ with half the width of the previous one. Perform two more steps and determine an interval of width $1/16$ containing a root of the polynomial $X^3 - X - 5$.

The ideas given in the previous give rise to an algorithm to find approximations of the roots of a polynomial with real coefficients. More precisely: given $p(X) \in \mathbb{R}[x]$ and $a, b \in \mathbb{R}$ such that $a < b$ and $p(a) \cdot p(b) < 0$, we can recursively define a sequence of intervals $[a_0, b_0], [a_1, b_1], [a_2, b_2], \ldots$ as follows:

$$[a_n, b_n] = \begin{cases} [a, b] & \text{if } n = 0 \\ \left[a_{n-1}, \dfrac{a_{n-1}+b_{n-1}}{2}\right] & \text{if } n \geq 1 \text{ and } p(a_{n-1}) \cdot p\left(\dfrac{a_{n-1}+b_{n-1}}{2}\right) \leq 0 \\ \left[\dfrac{a_{n-1}+b_{n-1}}{2}, b_{n-1}\right] & \text{if } n \geq 1 \text{ and } p(a_{n-1}) \cdot p\left(\dfrac{a_{n-1}+b_{n-1}}{2}\right) > 0 \end{cases}$$

Taking the midpoints of all the found intervals then gives rise to better and better approximations of a root of the polynomial $p(X)$. In other words: the sequence of real numbers $r_0, r_1, r_2, \ldots$ defined as

$$r_n = \frac{a_n + b_n}{2}$$

give approximations of a root of $p(X)$. The larger the value of n is, the better the approximation will become. This way of approximating roots is called the bisection method.

4. (a) For the polynomial $f(X) = X^3 - X - 5$ and $[a, b] = [1, 2]$, use Python to compute r_4. You can reuse the intervals you already compute in the previous exercise. Also use Python to compute $f(r_4)$. Answer: $r_4 = 1.90625$ and $f(r_4) = 0.020660400390625$.

   (b) Still assuming that $f(X) = X^3 - X - 5$ and $[a, b] = [1, 2]$, show using induction on n that the width of the interval $[a_n, b_n]$ is equal to $1/2^n$ for all $n \in \mathbb{Z}_{\geq 0}$. A consequence of this is that the distance from r_n to a root of $f(X)$ is at most $1/2^{n+1}$.

Remark 3 *As a small warning, note that in a real life application, one may actually only know the coefficients of $p(X)$ up to a certain numerical precision. Then if $(a_{n-1} + b_{n-1})/2$ is already close to a root of $p(X)$, it may not be possible to reliably determine whether $p(a_{n-1}) \cdot p((a_{n-1} + b_{n-1})/2) > 0$ or $p(a_{n-1}) \cdot p((a_{n-1} + b_{n-1})/2) \leq 0$, but only that $p(a_{n-1}) \cdot p((a_{n-1} + b_{n-1})/2)$ is very close to zero. These type of considerations are the topic of numerical analysis and are important when developing so-called numerical algorithms.*

Part II: Newton-Raphson's method for finding roots of polynomials with real coefficients

If all went well, you found in the previous part that the polynomial $X^3 - X - 5$ has a root contained in the interval $[1.875, 1.9375]$. The midpoint of this interval is therefore a reasonable approximation of a root of $X^3 - X - 5$. As already computed in the previous exercise, this midpoint is equal to $(1.875 + 1.9375)/2 = 1.90625$. We will now consider another way to find numerical approximations of roots of a polynomial $p(X) \in \mathbb{R}[X]$.

5. Let $p(X) \in \mathbb{R}[X]$ be a polynomial of degree at least one. Further, suppose that $\lambda \in \mathbb{R}$ is a root of $p(X)$. As before, we define a function that this polynomial gives rise to $p : \mathbb{R} \to \mathbb{R}$, where $x \mapsto p(x)$.

   (a) Suppose that λ is a root of $p(X)$ of multiplicity m, where m is at least two. Use Definition 4.6.1 to conclude that in that case there exists a polynomial $g(X)$ such that $f(X) = (X - \lambda)^m \cdot g(X)$. Now use the assumption that m is at least two, to deduce that there exist a polynomial $h(X)$ such that $f(X) = (X - \lambda)^2 \cdot h(X)$.

   (b) Given a polynomial
   $$f(X) = a_0 + a_1 X + a_2 X^2 + \cdots + a_n X^n,$$
   the derivative of $f(X)$, denoted by $f'(X)$ or sometimes also by $f(X)'$, is defined in the usual way as
   $$f'(X) = a_1 + 2a_2 X + \cdots + n a_n X^{n-1}.$$
   Show that if λ is a root of $f(X)$ of multiplicity at least two, then λ is a root of $f'(X)$. Hint: use that $f(X) = (X - \lambda)^2 \cdot h(X)$ for some polynomial $h(X)$ and use the product rule for differentiation.

   (c) Suppose that λ is a root of $f(X)$ of multiplicity one. Show that in this case there exists a polynomial $g(X)$ such that $f(X) = (X - \lambda) \cdot g(X)$ and $g(\lambda) \neq 0$.

   (d) Show that if λ is a root of $f(X)$ of multiplicity one, then $f'(\lambda) \neq 0$. Hint: use again the product rule to find an expression for $f'(x)$, this time on the product $f(X) = (X - \lambda) \cdot g(X)$. Then use the fact that $g(\lambda) \neq 0$.

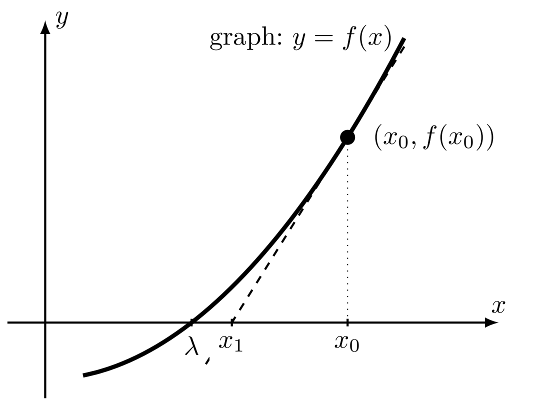
A root of a polynomial $f(X)$ of multiplicity one is also called a *simple* root. In a more general setting, if $f : \mathbb{R} \to \mathbb{R}$ is a differentiable function, then an element $\lambda \in \mathbb{R}$ is called a *simple zero* of f if $f(\lambda) = 0$ and $f'(\lambda) \neq 0$.

6. Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function and let $x_0 \in \mathbb{R}$ be given. You may assume that $f'(x_0) \neq 0$.

   (a) Check that the tangent line to the graph of f at the point $(x_0, f(x_0))$ is given by the equation $y = f'(x_0) \cdot (x - x_0) + f(x_0)$.

   (b) Show that the intersection of this tangent line and the x-axis is given by the point $(x_1, 0)$, where
   $$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

If the initial "guess" x_0 is close to λ, then x_1 tends to be closer to λ than x_0. See the following figure for an illustration.



The idea of the Newton-Raphson method (often called the Newton-Raphson algorithm) is now to iterate this procedure: once $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$ is computed, one can define $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$. If the starting point x_1 was already closer to λ than x_0, one would expect x_2 to be even closer! If we want to try to find an even better approximation of λ, one can simply iterate the procedure even more times. More formally, what we find is a sequence of real numbers $x_0, x_1, x_2, \dots$ recursively defined as follows:

$$x_n = \begin{cases} x_0 & \text{if } n = 0 \\ x_{n-1} - \dfrac{f(x_{n-1})}{f'(x_{n-1})} & \text{if } n \geq 1 \end{cases}$$

It turns out that the following is true:

Theorem 4 *Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function and suppose that λ is a simple zero of f. Then there exists a real number $\epsilon > 0$ such that for any $x_0 \in]\lambda - \epsilon, \lambda + \epsilon[$ the real numbers x_n defined above tend to λ as n increases.*

Formally, one says that the sequence $x_0, x_1, x_2, \dots$ *converges* to λ, or in other words that $\lim_{n\to\infty} x_n = \lambda$. Convergence of sequences is a topic for other courses though and we will only use this terminology in an informal way. Let us try out the Newton-Raphson algorithm in a specific example.

7. Let $f : \mathbb{R} \to \mathbb{R}$ be defined by $f(x) = x^3 - x - 5$ and let $x_0 = 2$.

   (a) Compute x_1 and x_2 using Python directly from the formulas $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$ and $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$.

   (b) To compute further values of x_n, it is more convenient to use a recursively defined function $F : \mathbb{Z}_{\geq 0} \to \mathbb{R}$ where $F(n) = x_n$. To define it, we first need to define in Python the functions $f : \mathbb{R} \to \mathbb{R}$, with $x \mapsto x^3 - x - 5$ and a function $fprime : \mathbb{R} \to \mathbb{R}$ with $x \mapsto 3x^2 - 1$. Note that $fprime(x) = f'(x)$. Defining this functions in Python can be done in the following way:

```
def f(x):
 return x**3 - x - 5
def fprime(x):
 return 3*x**2 - 1
```

Now we set x_0 equal to 2 and define the function $F : \mathbb{Z}_{\geq 0} \to \mathbb{R}$ where $F(n) = x_n$ in a recursive way as follows in Python:

```
x0=2
def F(n):
 if n==0:
  return x0
 else:
  return (F(n-1)-f(F(n-1))/fprime(F(n-1)))
```

Now type the code given above in Python to set x_0 to 2 and to define the functions f, $fprime$ and F. Verify in Python that $F(0)$, $F(1)$ and $F(2)$ give the right output $F(0) = 2$, $F(1) = x_1$ and $F(2) = x_2$. You can compare with the values of x_1 and x_2 that you already computed in the previous part of this exercise.

(c) Use the recursively defined function F to compute x_3 and x_4.

(d) Compute x_5 and compare it to x_4. What is the truth value of the logical expression `F(4)==F(5)` according to Python? Any comments about Pythons output?

(e) Recall that using the bisection method, one obtained after four iterations the approximation $r_4 = 1.90625$ for a root of $f(X) = X^3 - X - 5$, which satisfied $f(r_4) = 0.020660400390625$. Now compute $f(x_4)$. Which approach gives the better approximation of a root of $X^3 - X - 5$ after four iterations: the bisection method of the Newton-Raphson algorithm?

Remark 5 *One can show that (roughly speaking) with each iteration of the bisection method, the number of correct decimals increases by the same amount. By contrast, if your choice of starting value x_0 is reasonably close to an actual root (and that root has multiplicity one), then using the Newton-Raphson method, the number of correct decimals doubles in each iteration! In practice, one combines both methods: first the bisection method with only a few iterations is used to find a reasonable approximation of a root, then the Newton-Raphson method is used to find a very good approximation of this root very fast.*

End of thematic exercise 2.

For those wanting more, here are a few more questions, but making them is purely voluntary!

We will be using complex numbers now, so we start by loading the same package as in thematic exercise 1.

```
import cmath
```

Just as a reminder, the complex number $2 + 3i$ can be defined as follows.

```
z = complex(2,3)
```

Displaying the complex number can now be done easily by typing:

```
z
```

Note that z is displayed by Python as $2 + 3j$. Hence command line Python prefers to use 'j' rather than 'i'.

8. We have seen an example where Newton-Raphson's algorithm worked very well. Its weakness is that the initial guess x_0 needs to be a good one. We consider an example that demonstrates this. We choose in this exercise $g : \mathbb{C} \to \mathbb{C}$ defined by $g(x) = x^3 - 2x^2$.
   (a) Choose $x_0 = -1$ and use Python to compute x_n for some small values of n. Conclude that as before the sequence $x_0, x_1, x_2, \ldots$ seems to tend to a root of the polynomial $X^3 - 2X^2$.
   (b) Now choose $x_0 = 0$ and use Python to (try to) compute x_1. What goes wrong?
   (c) Let us choose $x_0 = 0.1$ to avoid the division by zero in the first step. Compute x_n for n up to 10. Does the sequence $x_0, x_1, x_2, \ldots$ seem to converge?

9. As a final example, we show that Newton-Raphson's algorithm can also find approximations of complex (simple) roots of polynomials. We choose in this exercise $h : \mathbb{C} \to \mathbb{C}$ defined by $h(z) = z^3 - 1$.
   (a) The roots of the polynomial $Z^3 - 1$ are the complex numbers 1, $-\frac{1}{2} + \frac{1}{2}\sqrt{3} \cdot i$ and $-\frac{1}{2} - \frac{1}{2}\sqrt{3} \cdot i$. Verify this using the theory of binomial equations.
   (b) Now choose $x_0 = i$ and use Python to compute x_n for some small values of n. To which root of $Z^3 - 1$ does the sequence $x_0, x_1, x_2, \ldots$ converge?
   (c) Experiment with the value of x_0 and find a value of it such that the resulting sequence $x_0, x_1, x_2, \ldots$ converges to the root $-\frac{1}{2} - \frac{1}{2}\sqrt{3} \cdot i$.