Exercises Laboratorio di Calcolo: Practicing NumPy + Matplotlib

Exercise 1

Represent the following matrices in NumPy. Write for each of the matrices a Python function that generates a NumPy array of a given size in the right format. Avoid the use of explicit for-loops.

- A matrix of size (m, n) with checkerboard pattern:

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

- A matrix of size (m, n) with the numbers 1, 2, ..., $m \cdot n$:

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}$$

- A square matrix of size (n, n) with the numbers 1, 2, ..., n on both the diagonal and the anti-diagonal:

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

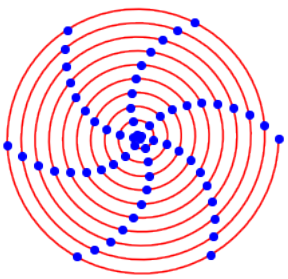
Exercise 2

Write Python functions with the following utilities. Avoid the use of explicit for-loops.

- A function that returns the closest value (to a given scalar) in a 1D array
- A function that inserts 3 consecutive zeros between each element in a 1D array
- A function that swaps two rows of a 2D array

Exercise 3

Write a Python program that visualizes a spiral (in red color) with bullets (in blue color) on top of it, as shown in the picture on the right.



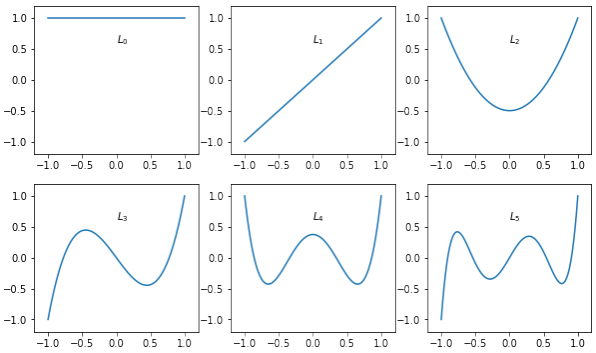
The Archimedean spiral can be parameterized as
$x = t \cos(t)$
$y = t \sin(t)$

Exercise 4

Legendre polynomials (discovered by Adrien-Marie Legendre in 1782) are a set of complete and orthogonal polynomials, with several nice properties and plenty of applications. Write a Python program to visualize the first n Legendre polynomials in the interval [-1, 1]. The built-in function `np.polynomial.Legendre.basis` from the module `np.polynomial` can be useful.

Try out different configurations. For example, for $n = 6$, produce the plots

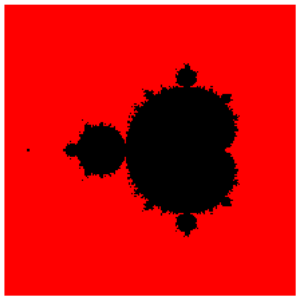
and



Exercise 5: the Mandelbrot set

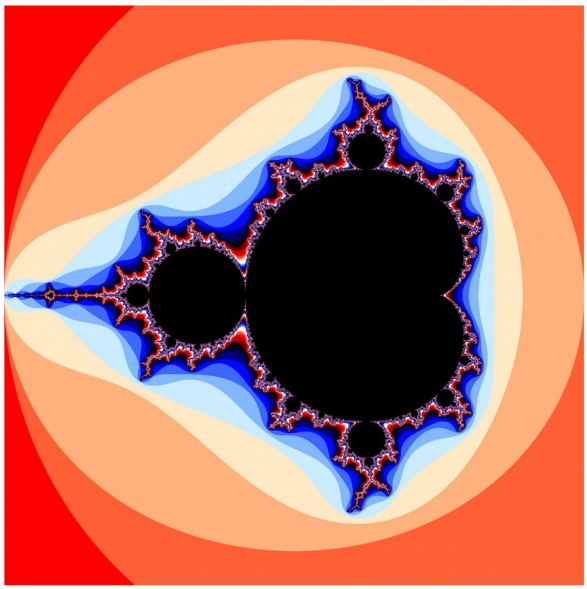
The Mandelbrot set is the set of complex numbers c for which the sequence $z_{i+1} = z_i^2 + c$ does not diverge, starting from $z_0 = 0$. This fractal set is named after the mathematician Benoit Mandelbrot.

In the lecture slides a basic Python implementation of the Mandelbrot set is given considering complex points in the region [-2, 1] + [-1.5, 1.5]j. A boolean matrix is returned containing `False` for the diverging points, and `True` otherwise. This results in the picture shown on the right (`False` = red, `True` = black).



Let us now bring some more variation in the Mandelbrot set.

Write a Python function that returns a matrix containing for each complex point the minimal iteration step i where the corresponding value $|z_{i+1}| > 2$. Then, visualize this matrix using `plt.imshow` and choose a high resolution (dpi) in order to fully appreciate the fractal structure.



Enhance your function so that you can also specify the considered rectangle in the complex plane. This will allow for making nice zooms of the mandelbrot fractal.

For example, in the region [-0.7450, -0.7425] + [0.1300, 0.1325]j, you get