Part 1: Gradient Descent

We will study the *gradient descent* algorithm, one of the simplest (and more general) function minimization methods. First we will consider a toy problem: the minimization of a function $f : \mathbb{R}^2 \to \mathbb{R}$. The second part of the practice is on the application of the gradient descent method to remove the noise in an image, via the minimization of a denoising energy.

1 Minimization of a toy function

We will use the gradient descent method to compute the minima of the function $f : \mathbb{R}^2 \to \mathbb{R}$ given by,

$$f(x_1, x_2) = \frac{1}{1000}\left(x_1^4 + x_2^4 - 80x_1^2 - 60x_2^2 + 100x_1 + 50x_2 + 1\right)$$

Assignments

1. *Complete the Python functions* `toy_fun` *and* `toy_gradient`*. These functions implement the function f and its gradient. Follow the comments provided in the code.*

2. *Complete the Python function* `gradient_descent`*. This function implements a gradient descent algorithm. We are going to implement it in a way in which we can use the same gradient descent function for this toy example and for the denoising energy of the next section. Follow the comments provided in the code.*

3. *Run the function* `toy_main` *with several time steps and several initial conditions and answer to the following questions:*

- *How many local minima does the function have in the domain $[-10, 10] \times [-10, 10]$?*
- *Run the gradient descent starting from $x^0 = [-2, -8]$. Does it converge to the global minimum?*
- *Estimate the rate of convergence from the logarithmic plot.*
- *Try different step sizes. Which step sizes yield a faster convergence? Which are more accurate?*