Mathematical Analysis Homework

Seminar V

Hus Lucian Antonio

Last Updated: November 11, 2023

Seminar V – Homework

Problem Statement:

8. ★ Let $f : \mathbb{R} \to \mathbb{R}$ be differentiable. To minimize f, consider the *gradient descent* method

$$x_{n+1} = x_n - \eta f'(x_n),$$

where $x_1 \in \mathbb{R}$ and $\eta > 0$ (learning rate). Use Python (numerics or graphics) for the following:
(a) Take a convex f and show that for small η the method converges to the minimum of f.
(b) Show that by increasing η the method can converge faster (in fewer steps).
(c) Show that taking η too large might lead to the divergence of the method.
(d) Take a nonconvex f and show that the method can get stuck in a local minimum.

Solution for convex functions:

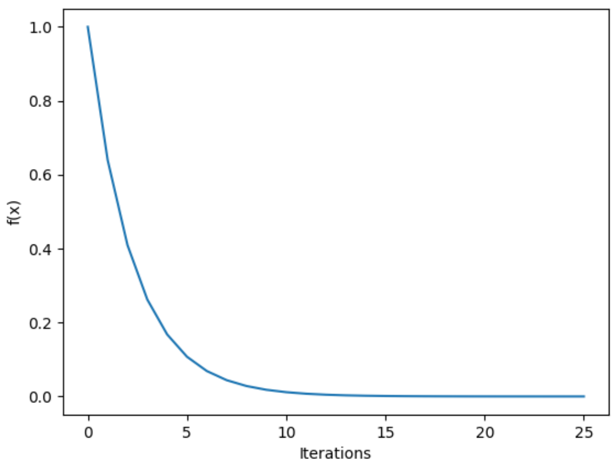
For our convex function tasks, we will take function 'f' to be x^2.

Hence the gradient of the function will be f'=2*x

The starting x0 will always be equal to 1, and we will check the convergence of our function in 25 (twenty-five) steps.

The learning rate 'η' will start with a value of 0.1 and will double at each given test until it reaches the value ten.

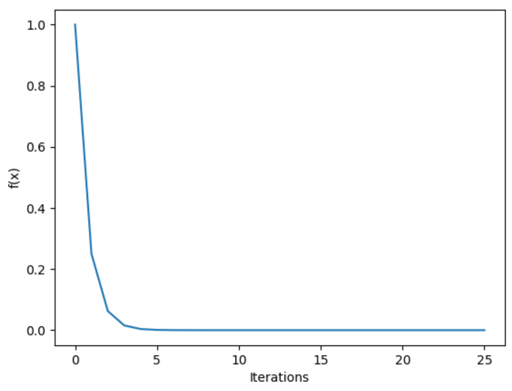
Base Case Test – Learning Rate η=0.1



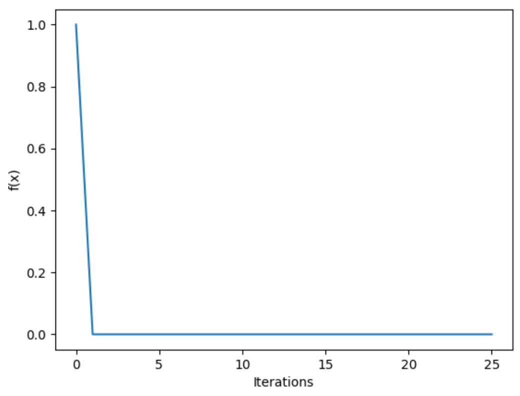
As we can see in the graph above, for a small learning rate the method converges to the minimum of 'f' *(proving point (a))*

Seminar V – Homework

Gradually increasing our learning rate $\eta = \{0.25, 0.5, 1, 2, 4\}$

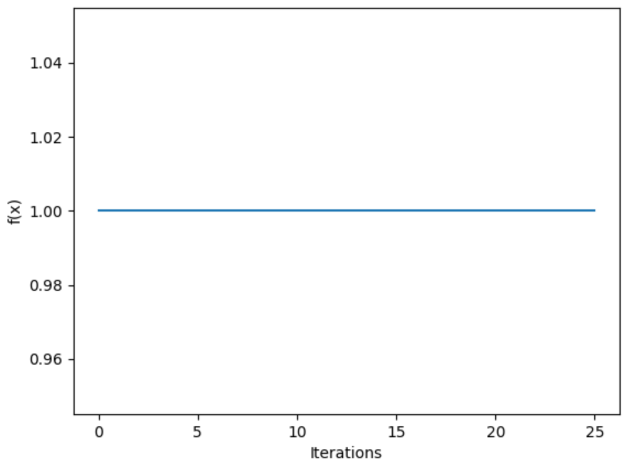


$\eta = 0.25$



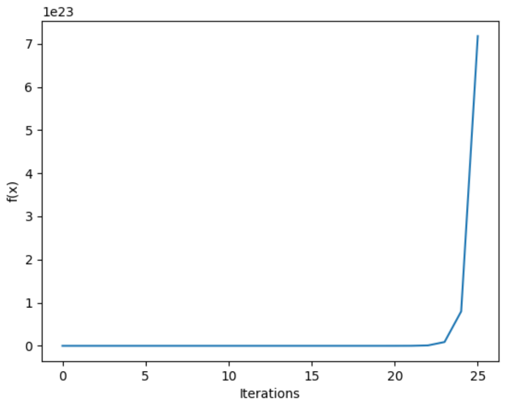
$\eta = 0.5$

For increasing learning rates 'η' we notice a tendency of the method to converge to the minimum value of 'f' in fewer steps *(proving point (b))*

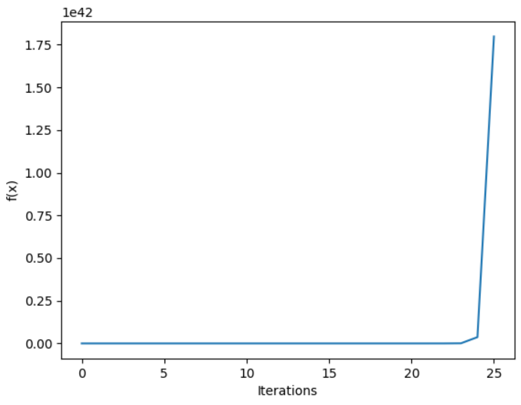


For a learning rate η=1 we notice a behaviour where the function remains constant, which is to be expected as xn = xn-1 – xn-1 = 0 for any natural number n, and real number x.

Seminar V – Homework



$\eta = 2$



$\eta = 4$

For learning rates 'η' greater than one we notice a tendency of the method to diverge, as we can see in the graph, the values of 'f' reach immense numbers in just a few steps *(proving point (c))*

Seminar V – Homework

Solution for non-convex function – point d

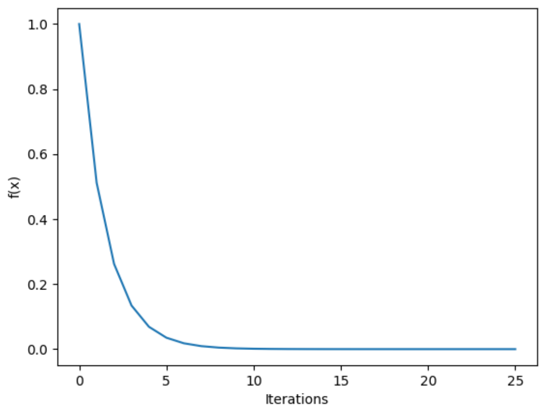
For our non-convex function task, we will take function 'f' to be x^3.

Hence the gradient of the function will be f'=3*(x^2).

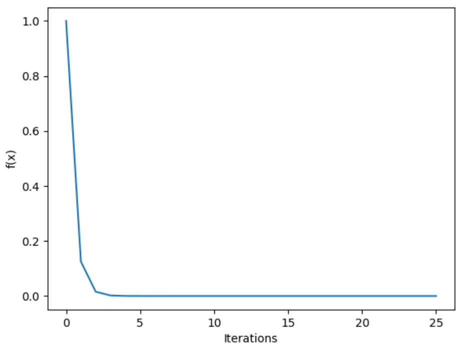
The starting x0 will always be equal to 1, and we will check the convergence of our function in 25 (twenty-five) steps.

The learning rate 'η' will start with a value of 0.1 and will double at each given test until it reaches the value ten.

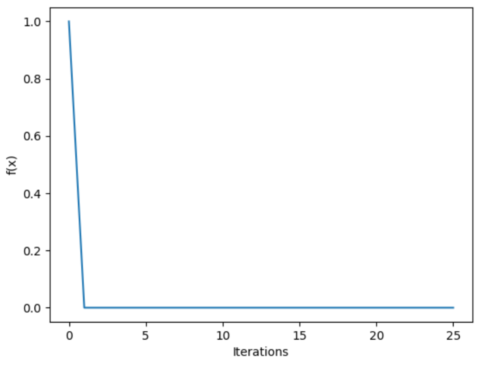
Base Case Test – Learning Rate η=0.1



Gradually increasing our learning rate η = {0.25, 0.5, 1, 2, 4}



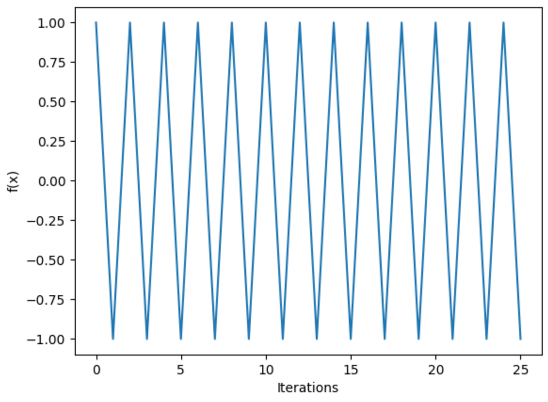
$\eta = 0.25$



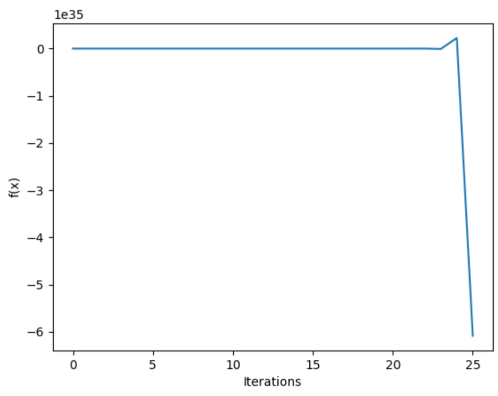
$\eta = 0.5$

Up to this point, we notice a very similar pattern as to the one in the case of a convex function. Things start to change however with larger values of the learning rate.

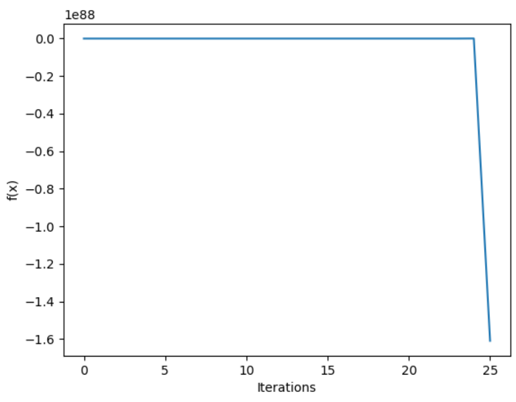
Seminar V – Homework



For a learning rate η=1 our method is no longer constant as for the convex function, but it oscillates. Following below are the graphs of the method having learning rates greater than one, which as we will see "get stuck" or diverge downwards to the minimum value of 'f'.



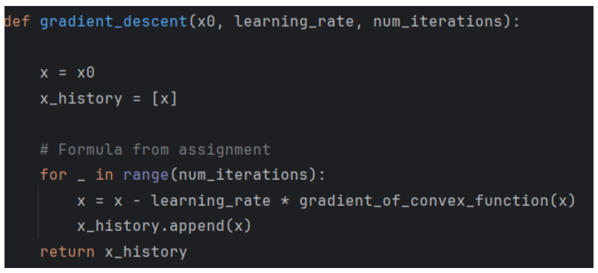
$\eta = 2$



$\eta = 4$

All the graphs provided in this documentation are part of my Python implementation as asked in the assignment which is found in my GitHub Repository. Click here.

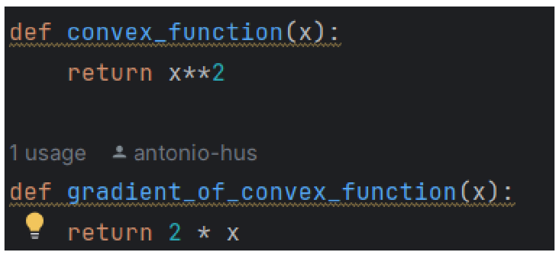
Here are some snippets of the code used in the program:



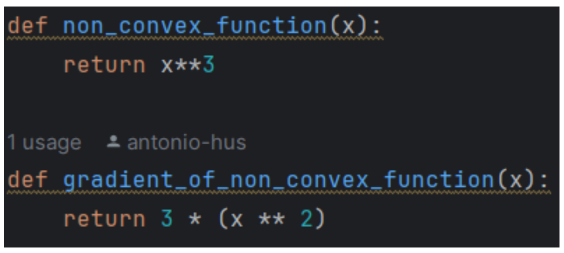
```python
def gradient_descent(x0, learning_rate, num_iterations):

    x = x0
    x_history = [x]

    # Formula from assignment
    for _ in range(num_iterations):
        x = x - learning_rate * gradient_of_convex_function(x)
        x_history.append(x)
    return x_history
```

Gradient Descent Method.

Seminar V – Homework

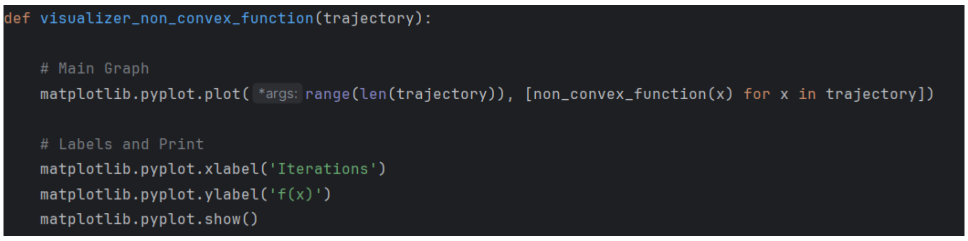


```python
def convex_function(x):
    return x**2

1 usage   antonio-hus
def gradient_of_convex_function(x):
    return 2 * x
```



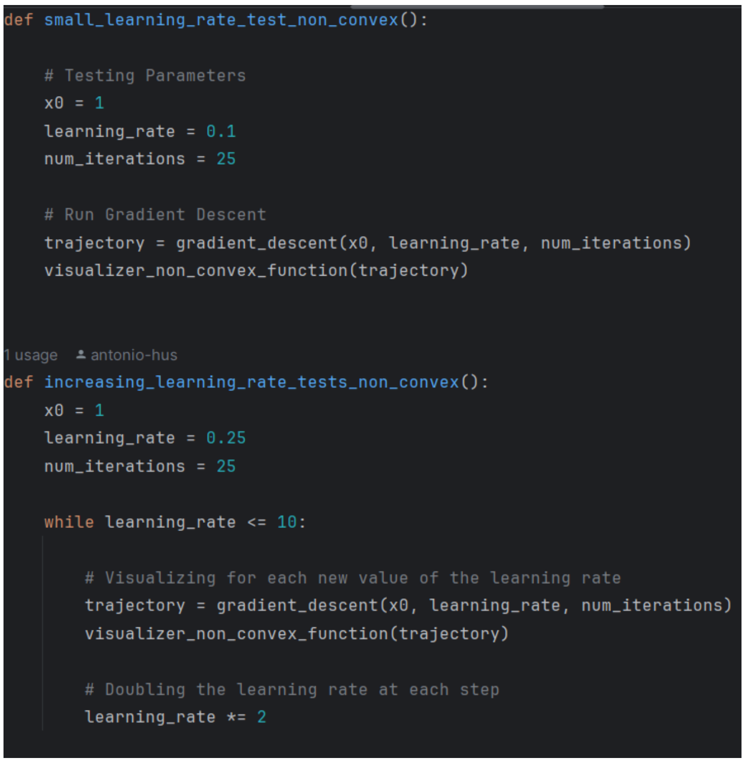
```python
def non_convex_function(x):
    return x**3

1 usage   antonio-hus
def gradient_of_non_convex_function(x):
    return 3 * (x ** 2)
```

Function initializations – convex and non-convex



```python
def visualizer_non_convex_function(trajectory):

    # Main Graph
    matplotlib.pyplot.plot( *args: range(len(trajectory)), [non_convex_function(x) for x in trajectory])

    # Labels and Print
    matplotlib.pyplot.xlabel('Iterations')
    matplotlib.pyplot.ylabel('f(x)')
    matplotlib.pyplot.show()
```

Matplotlib – Pyplot – Plot Visualizer for the methods



```python
def small_learning_rate_test_non_convex():

    # Testing Parameters
    x0 = 1
    learning_rate = 0.1
    num_iterations = 25

    # Run Gradient Descent
    trajectory = gradient_descent(x0, learning_rate, num_iterations)
    visualizer_non_convex_function(trajectory)


1 usage  antonio-hus
def increasing_learning_rate_tests_non_convex():
    x0 = 1
    learning_rate = 0.25
    num_iterations = 25

    while learning_rate <= 10:

        # Visualizing for each new value of the learning rate
        trajectory = gradient_descent(x0, learning_rate, num_iterations)
        visualizer_non_convex_function(trajectory)

        # Doubling the learning rate at each step
        learning_rate *= 2
```

Learning Rate Variation Tests

Again, for the full source code please visit my GitHub Repository.