```python
import matplotlib.pyplot as plt # Import Matplotlib for plotting
import numpy as np  # Import NumPy for numerical operations

def convex_function(x):
    return x ** 2

def gradient_of_convex_function(x): # the derivate
    return 2 * x

def non_convex_function(x):
    return x**4 - 4 * x**2 + x

def gradient_of_non_convex_function(x): # the derivate
    return 4 * x**3 - 8 * x + 1

def gradient_descent(x0: float, f, gradient, learning_rate: float):
    ITERATIONS = 100
    xn, yn = [x0], [f(x0)] # store the first value
    for i in range(ITERATIONS):
        x0 = x0 - learning_rate * gradient(x0) # update x0
        xn.append(x0)
        yn.append(f(x0))
    plt.plot(xn, yn, label=f"η = {learning_rate}", color='r', linestyle='-', marker='o')


# convex funtion
gradient_descent(1, convex_function, gradient_of_convex_function, 0.1) # η to small leads to slow convergence so it can be inneficent
gradient_descent(1, convex_function, gradient_of_convex_function, 0.3) # η larger leads to faster convergence
gradient_descent(1, convex_function, gradient_of_convex_function, 0.5) # η instantly gets to the minimum because it a convex function
gradient_descent(1, convex_function, gradient_of_convex_function, 0.7) # η also converges to the minimum
gradient_descent(1, convex_function, gradient_of_convex_function, 0.9) # η near 1 might get stuck in a local minimum
gradient_descent(1, convex_function, gradient_of_convex_function, 1.1) # η too large leads to divergence

x = np.linspace(-2, 2, 100)  # Generate 100 x-values from -1 to 1
y = convex_function(x)  # Calculate y-values using the function

plt.plot(x, y, label='y = x^2', color='b', linestyle='-', marker='')
plt.plot(0, 0, label='minimum', color='g', linestyle='-', marker='o')
plt.title('Gradient Descent for a convex function')


# non convex funtion ##########################################################################################################
gradient_descent(2.0, non_convex_function, gradient_of_non_convex_function, 0.1) # case where it gets stuck in a local minimum
# gradient_descent(-2.0, non_convex_function, gradient_of_non_convex_function, 0.001) # case where gets to the global minimum because it starts in the right place

x = np.linspace(-2, 2, 100)  # Generate 100 x-values from -1 to 1
y = non_convex_function(x)  # Calculate y-values using the function

plt.plot(x, y, label='y = x^2', color='b', linestyle='-', marker='')
plt.plot(-1.47, -5.44, label='minimum', color='g', linestyle='-', marker='o')
plt.title('Gradient Descent for a non convex function')




plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()


```

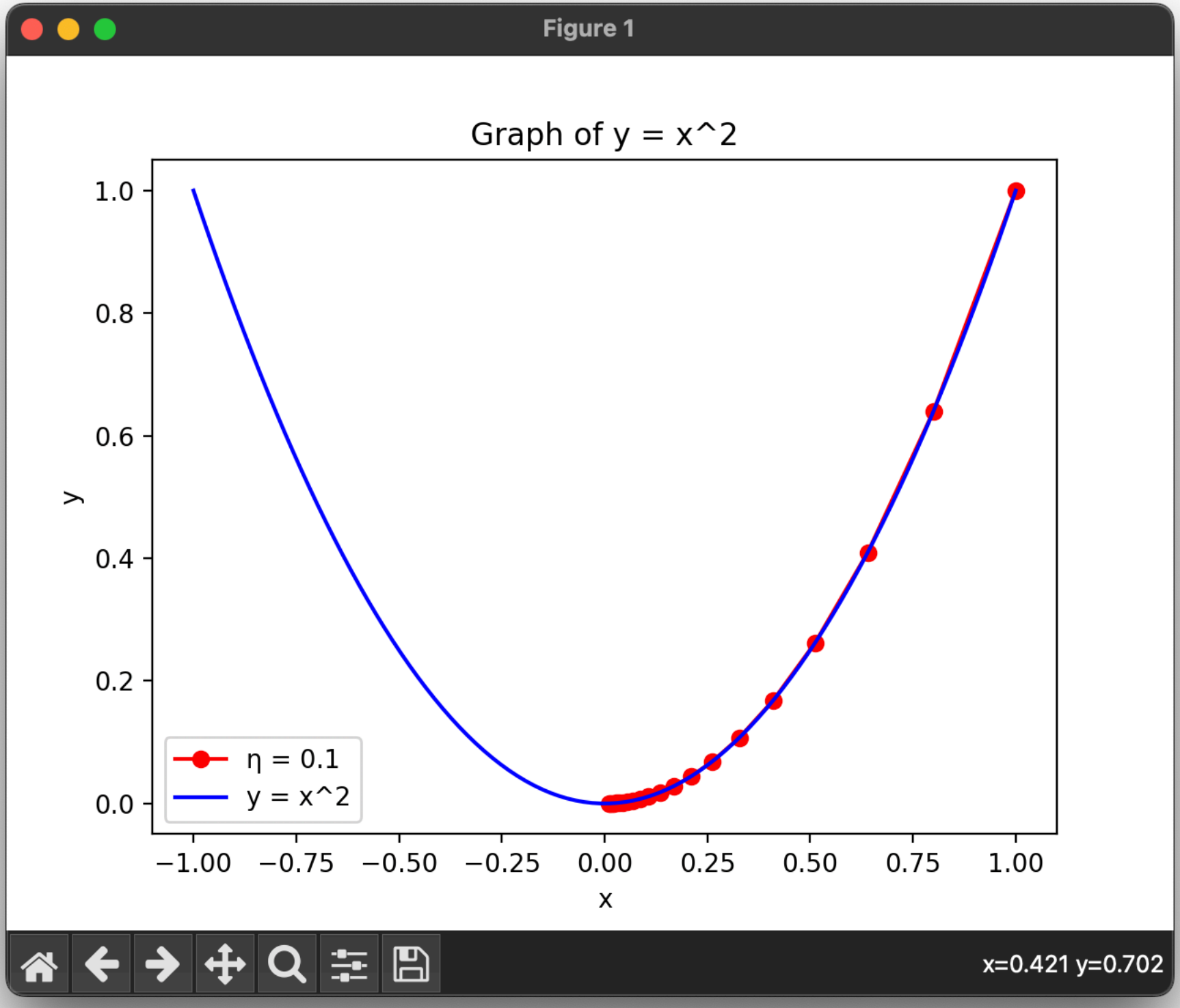
Figure 1
Graph of y = x^2
1.0
0.8
0.6
0.4
0.2
0.0
y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x
η = 0.1
y = x^2
x=0.421 y=0.702

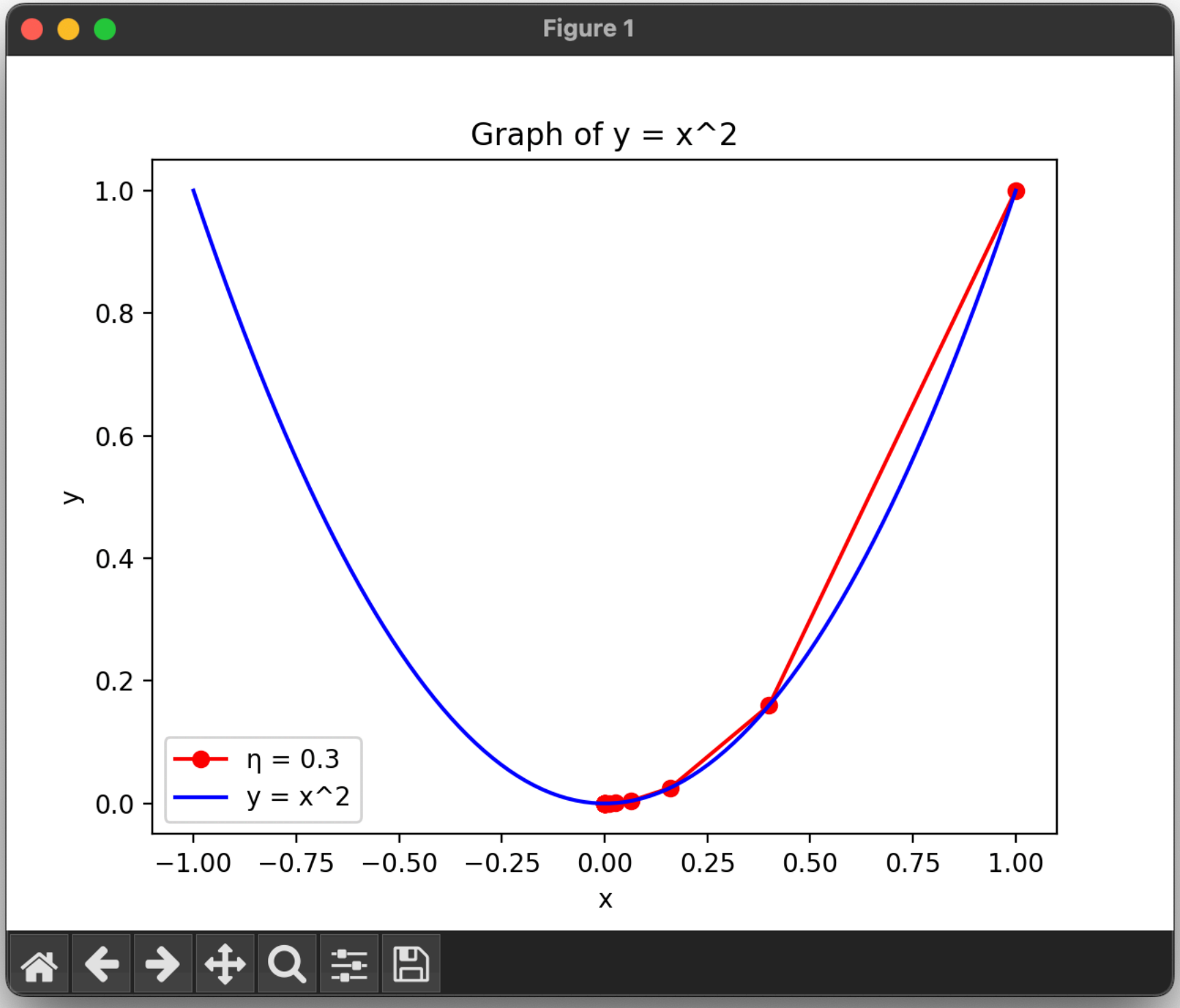
Figure 1
Graph of y = x^2
1.0
0.8
0.6
0.4
0.2
0.0
y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x
η = 0.3
y = x^2

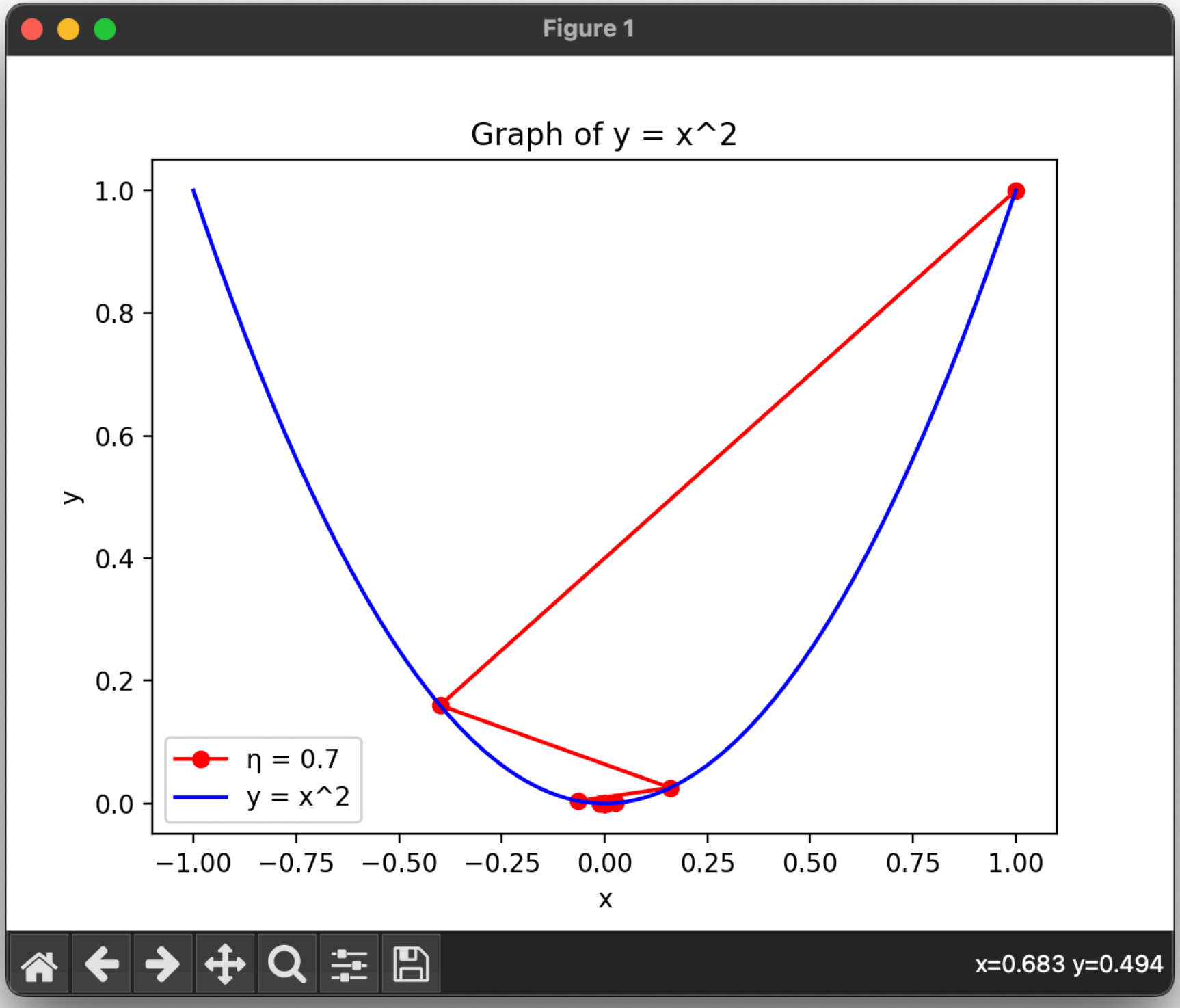
Figure 1
Graph of y = x^2
1.0
0.8
0.6
0.4
0.2
0.0
y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x
η = 0.7
y = x^2
x=0.683 y=0.494

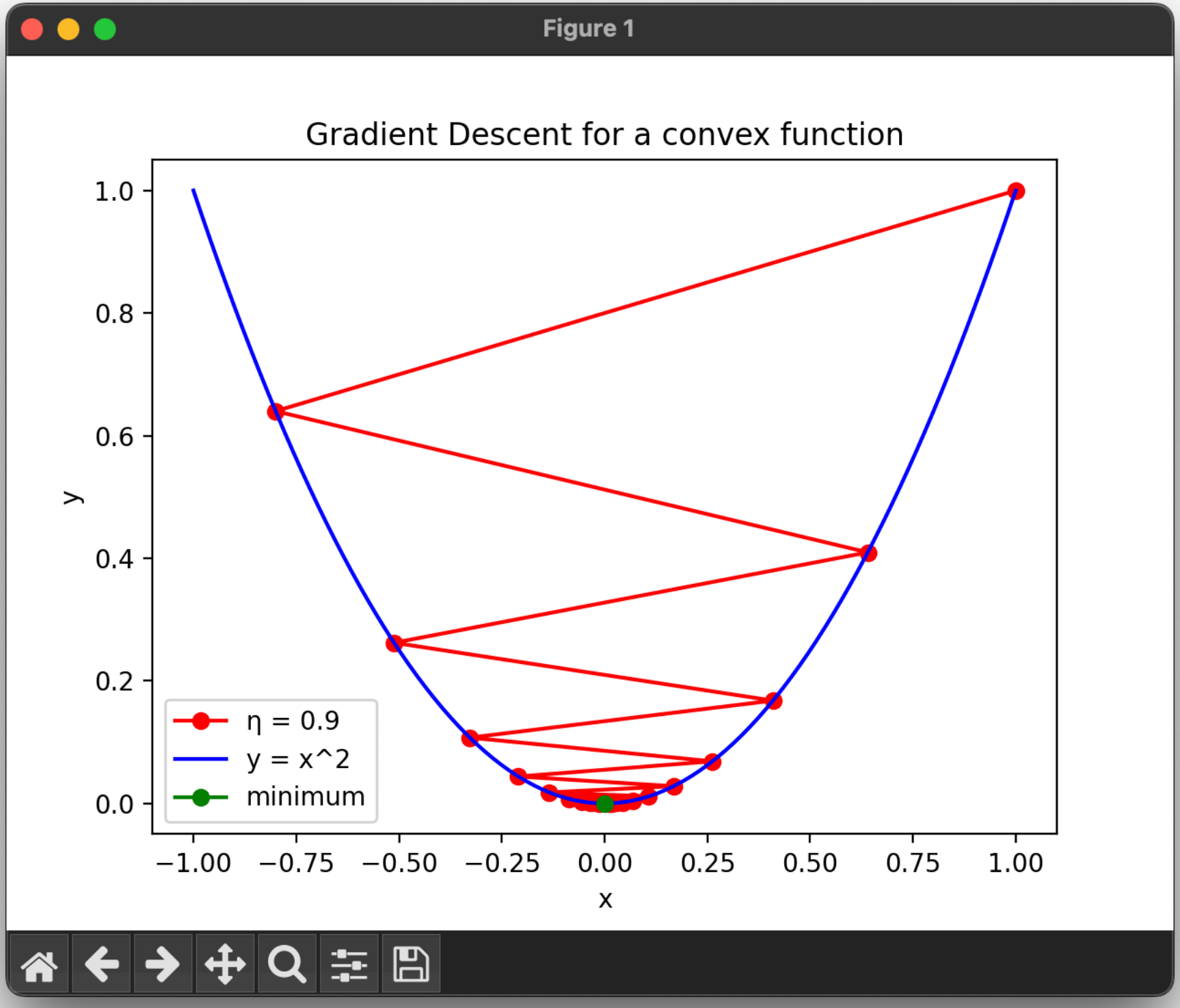
Figure 1
Gradient Descent for a convex function
1.0
0.8
0.6
0.4
0.2
0.0
y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x
η = 0.9
y = x^2
minimum

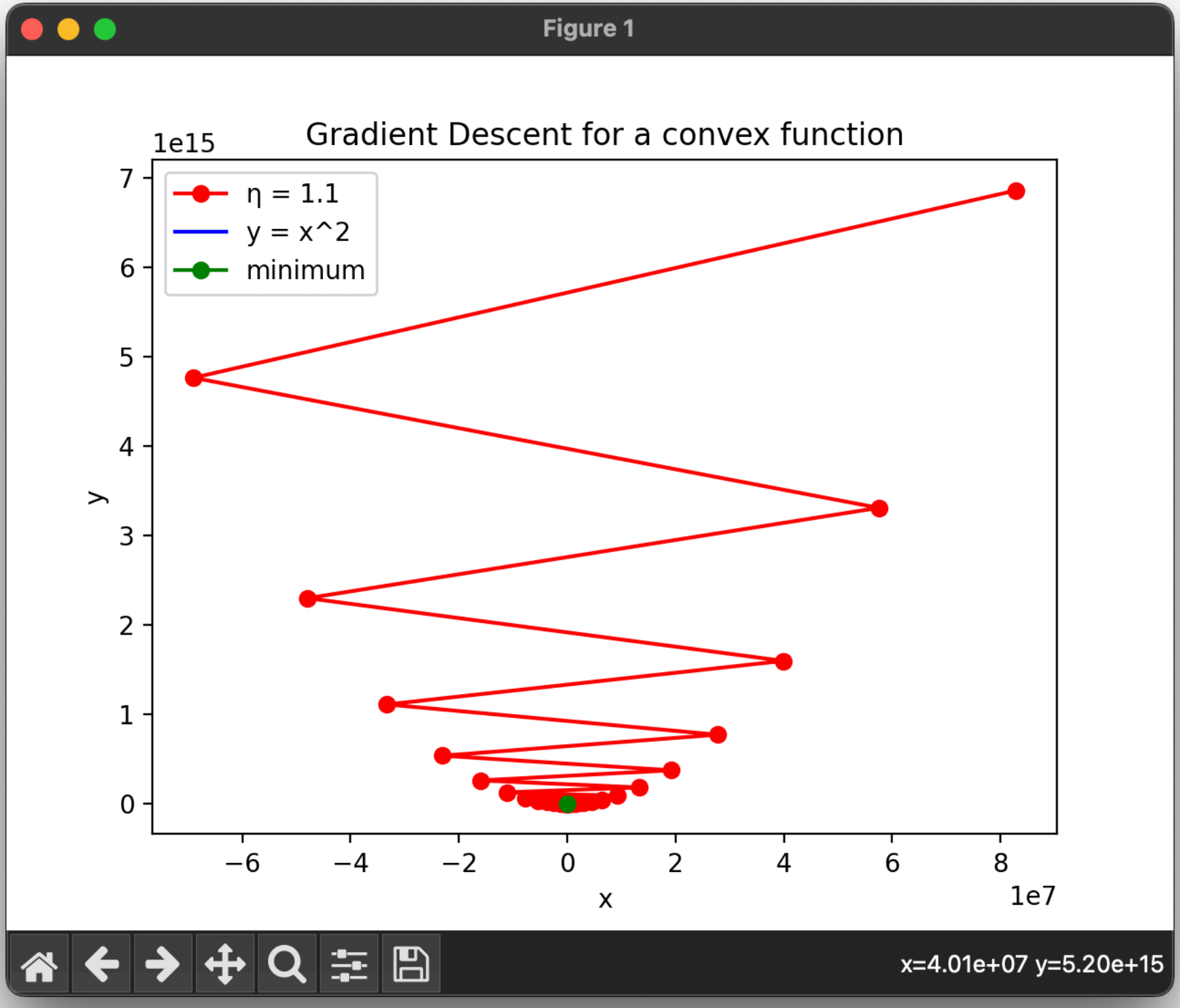
Figure 1
Gradient Descent for a convex function
1e15
η = 1.1
y = x^2
minimum
y
x
1e7
x=4.01e+07 y=5.20e+15

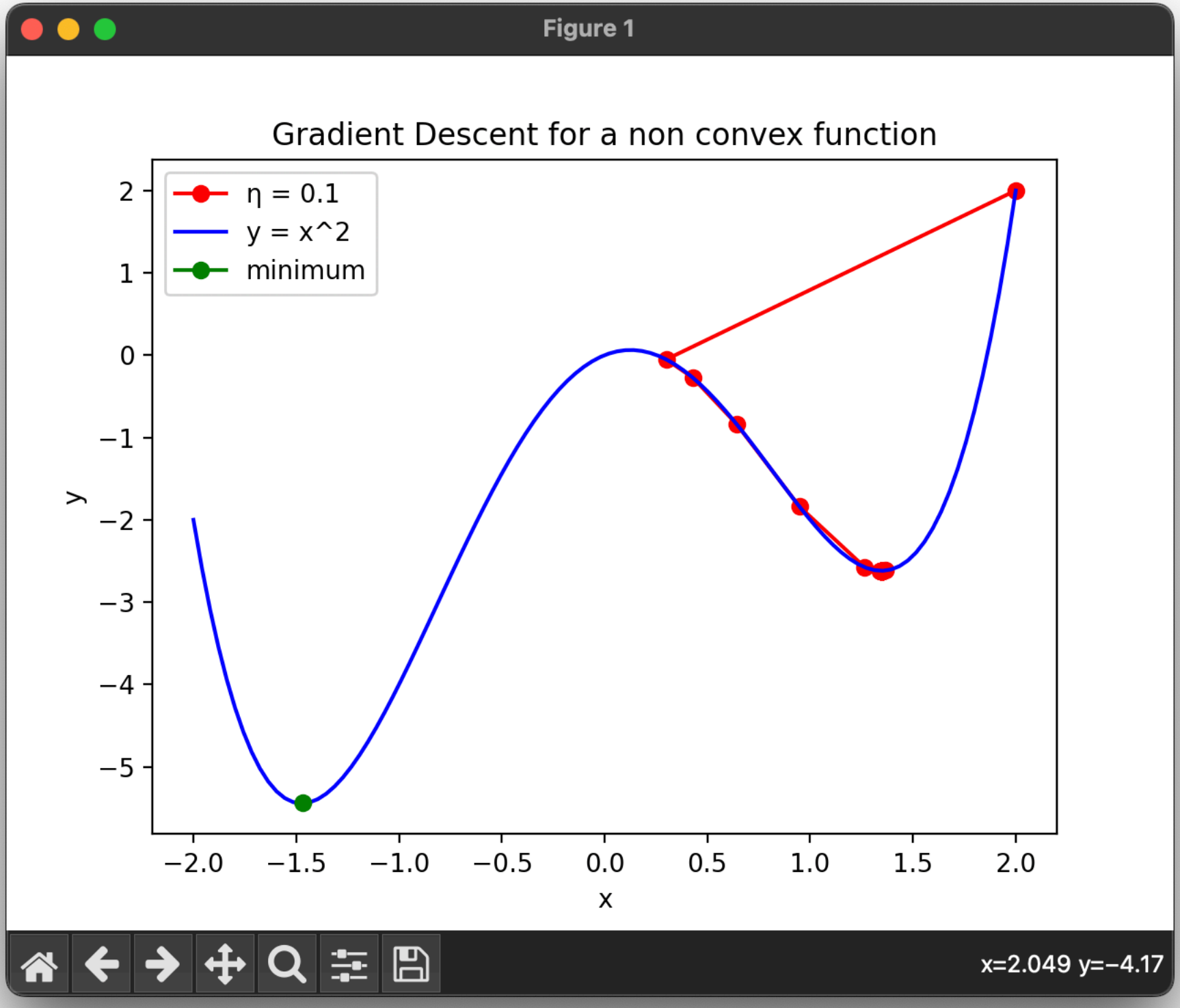
Figure 1
Gradient Descent for a non convex function
η = 0.1
y = x^2
minimum
2
1
0
−1
−2
−3
−4
−5
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x
x=2.049 y=−4.17