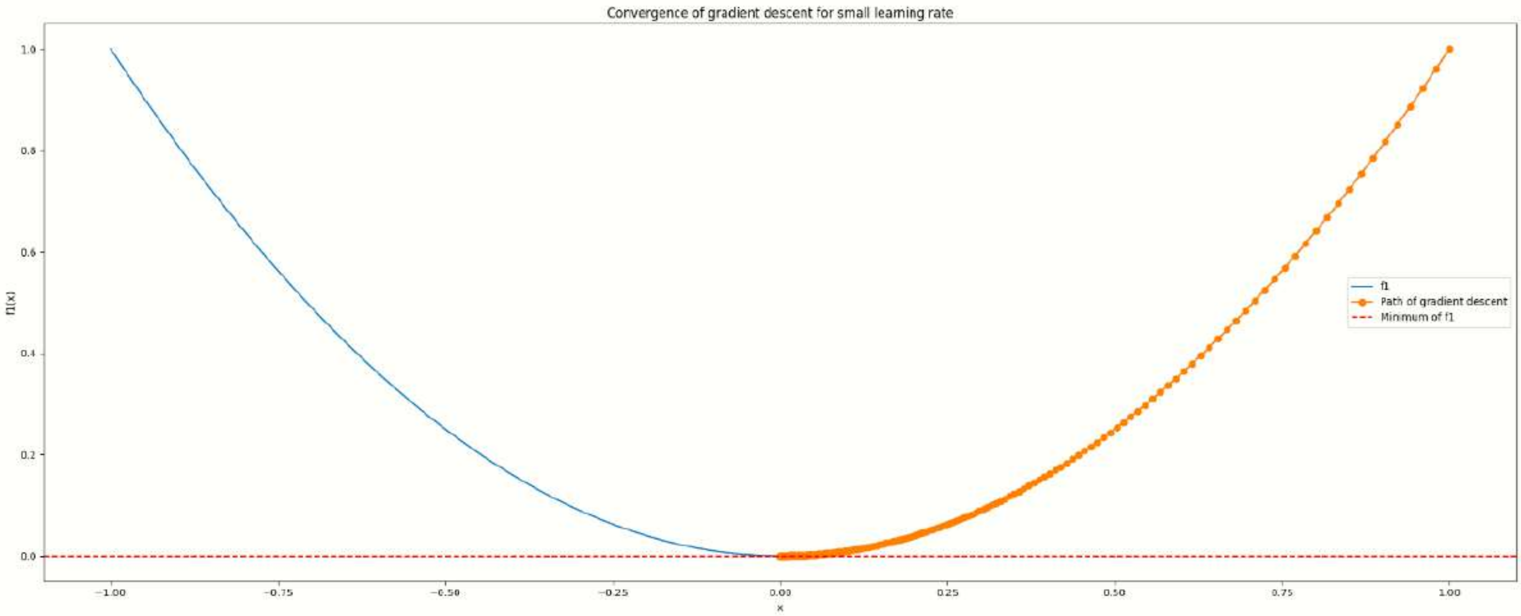
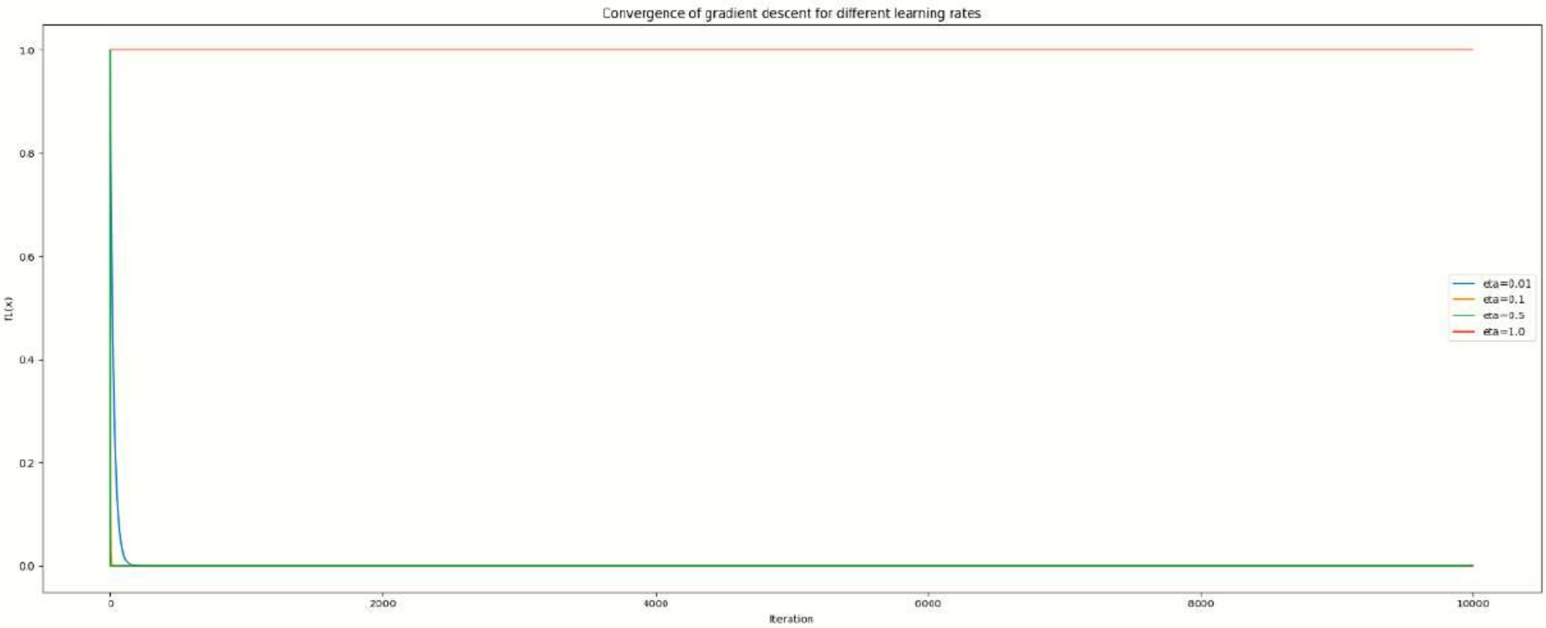
```python
import matplotlib.pyplot as plt
import numpy as np

# Define the functions and their derivatives
def f1(x): return x**2  # Convex function
def df1(x): return 2*x

def f2(x): return x**4 - 3*x**2 + 1  # Non-convex function
def df2(x): return 4*x**3 - 6*x

def gradient_descent(f, df, x0, eta, max_iter=10000, max_value=1e10):
    iter = 0
    x = x0
    xs = [x0]  # list to store x values
    while iter < max_iter:
        x_new = x - eta * df(x)
        if abs(f(x_new)) > max_value:
            print("Divergence!")
            break
        x = x_new
        xs.append(x)  # store x value
        iter += 1
    return xs, iter

x0 = 1

# Part (a)
eta = 0.01
xs, num_iter = gradient_descent(f1, df1, x0, eta)
print(f"Part (a) - For eta={eta}, minimum of f1: {xs[-1]:.6f} (found in {num_iter} iterations)")

# Plot f1
x = np.linspace(-1, 1, 400)
y = f1(x)
plt.plot(x, y, label="f1")
```

```python
# Plot f1
x = np.linspace(-1, 1, 400)
y = f1(x)
plt.plot(x, y, label="f1")

# Plot path of gradient descent
plt.plot(xs, [f1(x) for x in xs], 'o-', label="Path of gradient descent")

# Add horizontal line at y=0
plt.axhline(0, color='r', linestyle='--', label="Minimum of f1")

plt.title("Convergence of gradient descent for small learning rate")
plt.xlabel("x")
plt.ylabel("f1(x)")
plt.legend()
plt.show()

# Part (b)
etas = [0.01, 0.1, 0.5, 1.0]  # different values of eta
for eta in etas:
    xs, num_iter = gradient_descent(f1, df1, x0, eta)
    print(f"Part (b) - For eta={eta}, the method converged to the minimum of f1: {xs[-1]:.6f} in {num_iter} iterations")
    plt.plot([f1(x) for x in xs], label=f"eta={eta}")

plt.title("Convergence of gradient descent for different learning rates")
plt.xlabel("Iteration")
plt.ylabel("f1(x)")
plt.legend()
plt.show()

# Part (c)
etas = [1.1, 1.5, 2.0]  # different large values of eta
for eta in etas:
```

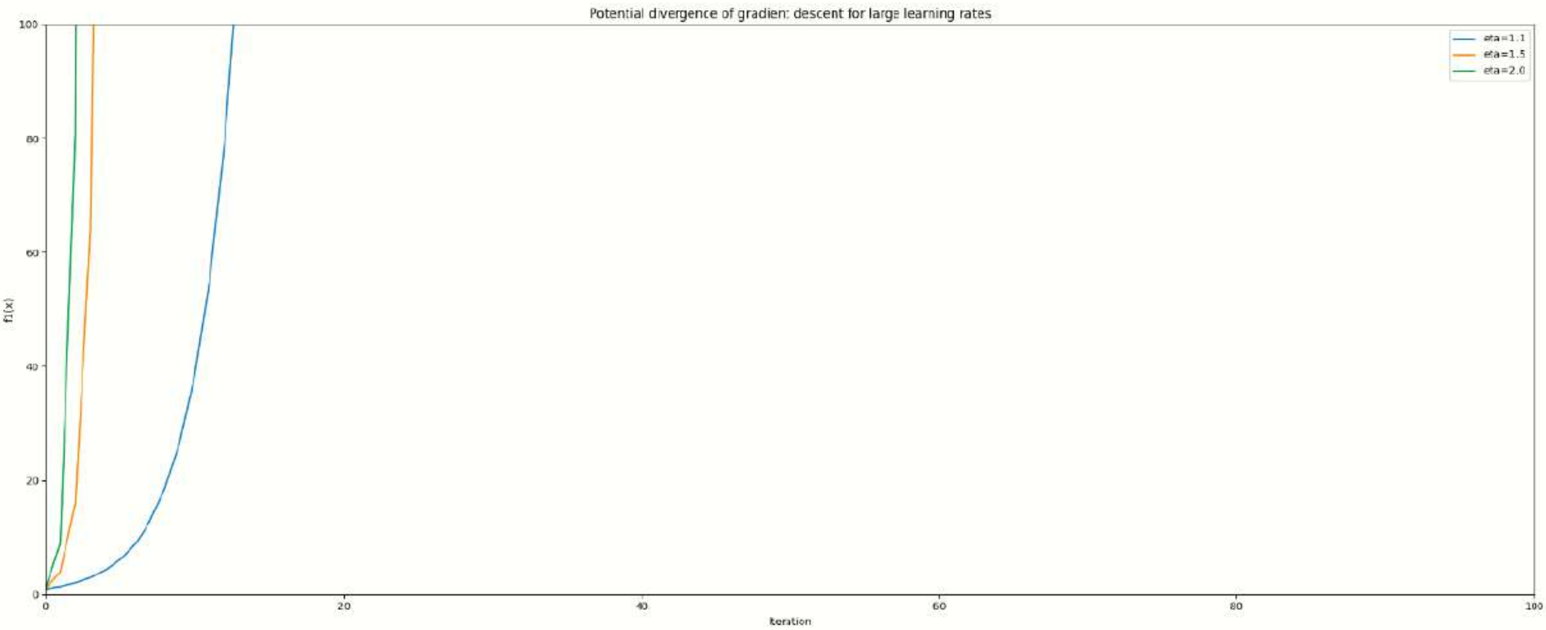
```python
# Part (c)
etas = [1.1, 1.5, 2.0]  # different large values of eta
for eta in etas:
    xs, num_iter = gradient_descent(f1, df1, x0, eta)
    if xs[-1] > 1e10:
        print(f"Part (c) - For eta={eta}, the method diverged")
    else:
        print(f"Part (c) - For eta={eta}, the method converged to the minimum of f1: {xs[-1]:.6f} in {num_iter} iterations")
    plt.plot([f1(x) for x in xs if f1(x) < 1e10], label=f"eta={eta}")

plt.title("Potential divergence of gradient descent for large learning rates")
plt.xlabel("Iteration")
plt.ylabel("f1(x)")
plt.xlim([0, 100])  # adjust x-axis range
plt.ylim([0, 100])  # adjust y-axis range
plt.legend()
plt.show()

# Part (d)
eta = 0.01
xs, num_iter = gradient_descent(f2, df2, x0, eta)
print(f"Part (d) - For eta={eta}, minimum of f2: {xs[-1]:.6f} (found in {num_iter} iterations)")

# Plot f2
x = np.linspace(-2, 2, 400)
y = f2(x)
plt.plot(x, y, label="f2")

# Plot path of gradient descent
plt.plot(xs, [f2(x) for x in xs], 'o-', label="Path of gradient descent")

plt.title("Gradient descent can get stuck in a local minimum for non-convex functions")
plt.xlabel("x")
plt.ylabel("f2(x) = x^4 - 3x^2 + 1")
plt.legend()
```



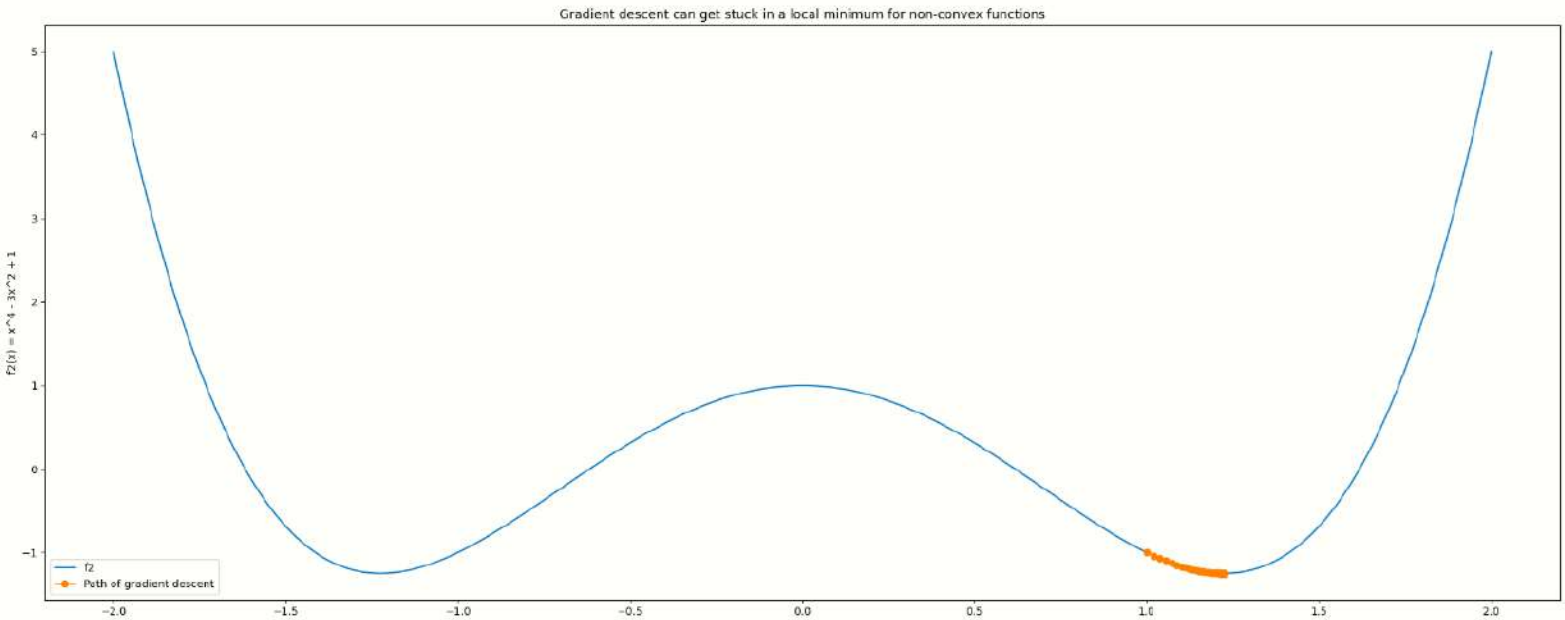
Convergence of gradient descent for small learning rate
1.0
0.8
0.6
0.4
0.2
0.0
f1(x)
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x
f1
Path of gradient descent
Minimum of f1



Convergence of gradient descent for different learning rates
1.0
0.8
0.6
0.4
0.2
0.0
f(x)
0
2000
4000
6000
8000
10000
Iteration
eta=0.01
eta=0.1
eta=0.5
eta=1.0



Potential divergence of gradien: descent for large learning rates
eta=1.1
eta=1.5
eta=2.0
f1(x)
iteration
0
20
40
60
80
100


Gradient descent can get stuck in a local minimum for non-convex functions
f2(x) = x^4 - 3x^2 + 1
f2
Path of gradient descent

```
Part (a) - For eta=0.01, minimum of f1: 0.000000 (found in 10000 iterations)
Part (b) - For eta=0.01, the method converged to the minimum of f1: 0.000000 in 10000 iterations
Part (b) - For eta=0.1, the method converged to the minimum of f1: 0.000000 in 10000 iterations
Part (b) - For eta=0.5, the method converged to the minimum of f1: 0.000000 in 10000 iterations
Part (b) - For eta=1.0, the method converged to the minimum of f1: 1.000000 in 10000 iterations
Divergence!
Part (c) - For eta=1.1, the method converged to the minimum of f1: -97368.504802 in 63 iterations
Divergence!
Part (c) - For eta=1.5, the method converged to the minimum of f1: 65536.000000 in 16 iterations
Divergence!
Part (c) - For eta=2.0, the method converged to the minimum of f1: 59049.000000 in 10 iterations
Part (d) - For eta=0.01, minimum of f2: 1.224745 (found in 10000 iterations)
```