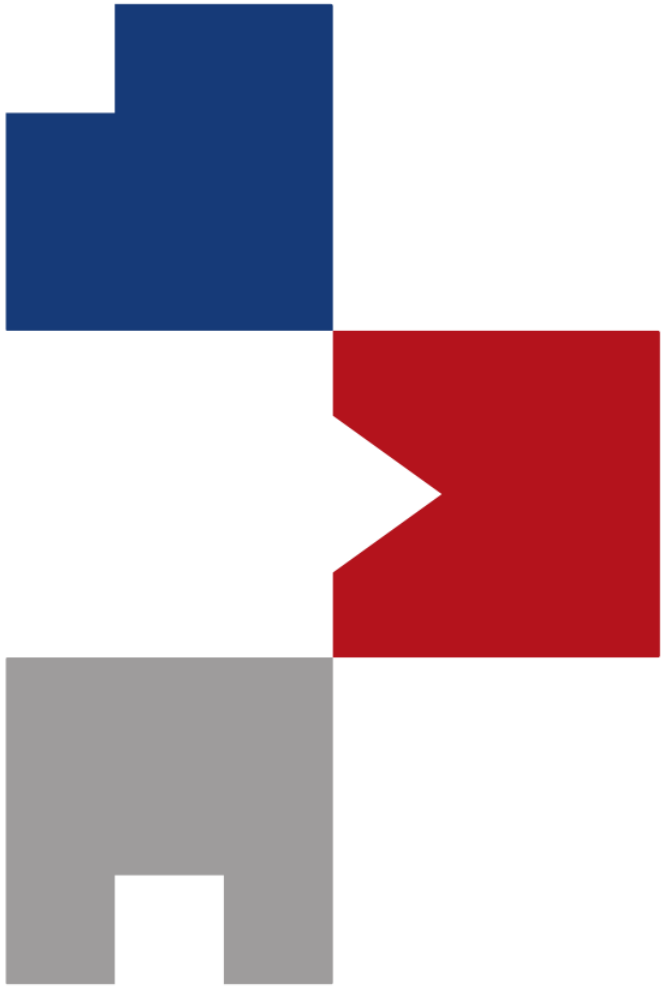
Note) Slides and example codes are available:
https://github.com/mint-lab/prog_meets_math



Math Lab #3:
Multivariate Nonlinear Optimization

Sunglok Choi, Assistant Professor, Ph.D.
Computer Science and Engineering Department, SEOULTECH
sunglok@seoultech.ac.kr | https://mint-lab.github.io/

Overview

- **Prerequisite**
  - Anacodna (Individual Edition)

- **Practice) Multivariate Nonlinear Optimization**
  - The given cost function
  - Expected results
  - Practice with the skeleton code
    - Step #1) Derive the gradient vector
    - Step #2) Implement your gradient descent

- **Assignment**
  - Mission: Complete the given skeleton code

Practice) Multivariate Nonlinear Optimization

- The given cost function: A multivariate function (다변수함수 in Korean) of $\mathbf{x}$
  - 2D Rosenbrock function (a.k.a. Rosenbrock's valley or banana function)
    - $f(\mathbf{x}) = f(x_0, x_1) = (1 - x_0)^2 + 100(x_1 - x_0^2)^2$ where $\mathbf{x} = [x_0, x_1]$
    - The optimal point at $x_0 = 1$ and $x_1 = 1$

- Note) Test functions for optimization
- Note) A univariate function (단변수함수 in Korean) of x

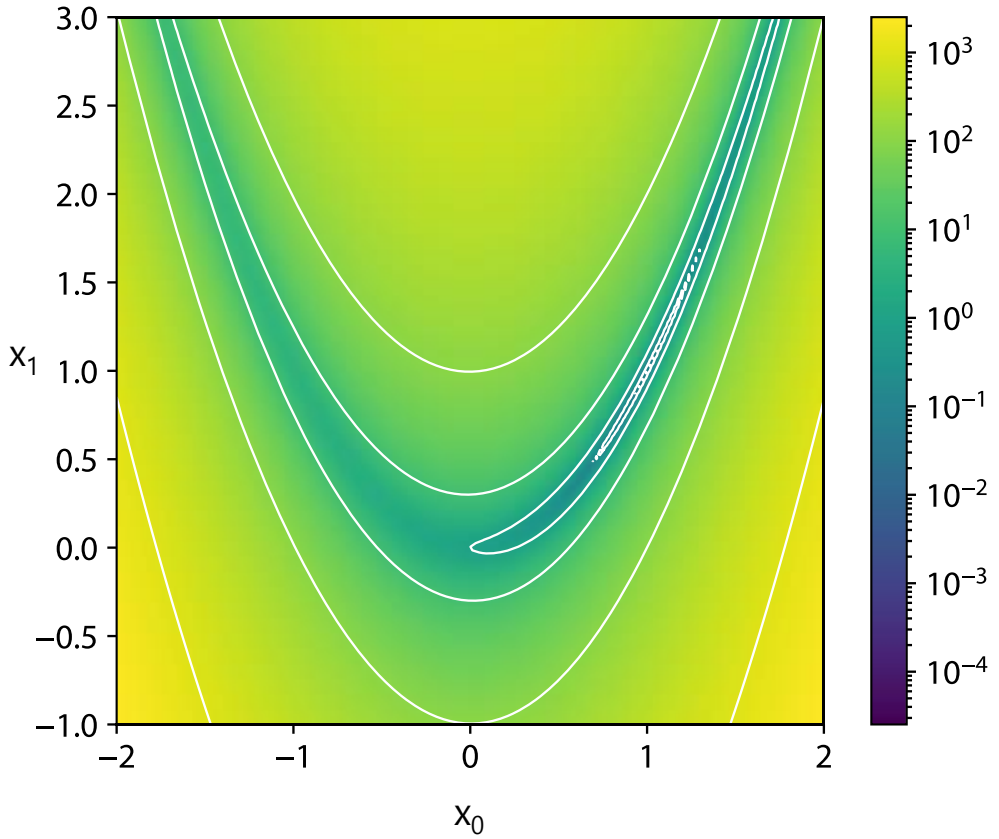
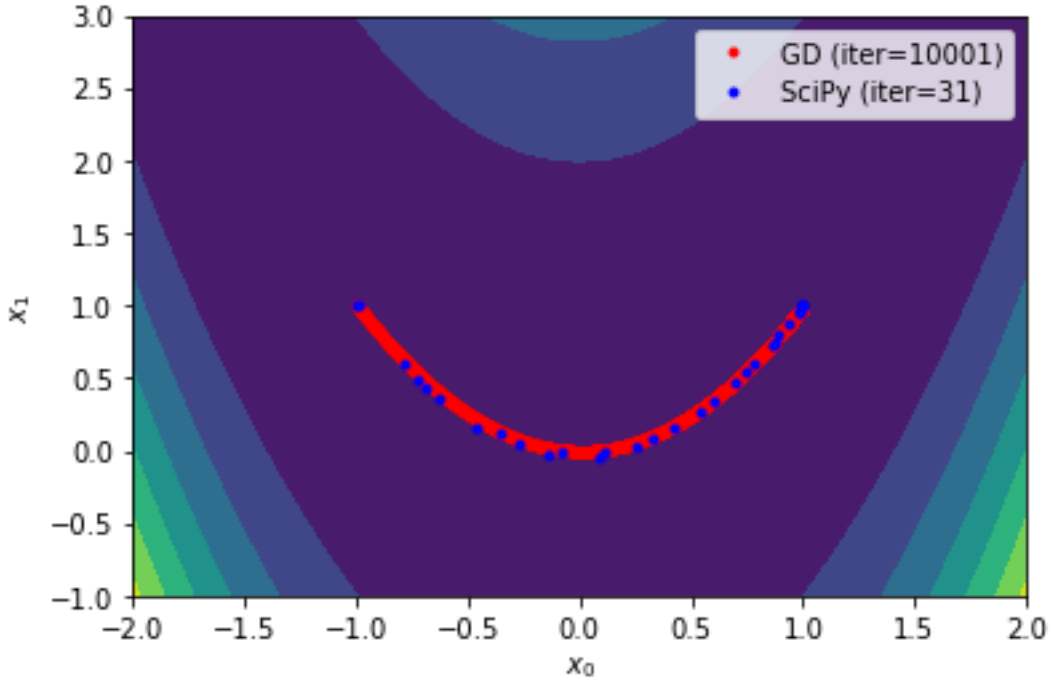


Image: Wikipedia

Practice) Multivariate Nonlinear Optimization

- Expected results @ default configuration
  - My gradient descent (# of iterations: 10001)
  - SciPy `optimize.minimize` (# of iterations: 31)



- Note) Please try other configuration and observe what happens.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize

if __name__ == '__main__':
    # Define a cost function and its gradient
    f   = lambda x: (1 - x[0])**2 + 100*(x[1] - x[0]**2)**2
    fd  = lambda x: np.array([0, 0]) # TODO: Fill the gradient vector

    # Define configuration
    x_init = [-1, 1]      # Please try other initial points
    learning_rate = 0.001 # Please try 0.01, 0.005, and 0.0001
    max_iter = 10000      # Please try 100, 1000, and 100000
    min_tol = 1e-6

    # Optimize the cost function using my gradient descent
    x = np.array(x_init)
    gd_xs = [x]
    for i in range(max_iter):
        # Run the gradient descent
        xp = x
        x = x # TODO: Implement your gradient descent
        gd_xs.append(x)

        # Check the terminal condition
        if np.linalg.norm(x - xp) < min_tol:
            break
    gd_xs = np.array(gd_xs)

    # Optimize the cost function using SciPy
    result = minimize(f, x_init, tol=min_tol, options={'maxiter': max_iter, 'return_all': True})
    sp_xs = np.array(result.allvecs)

    # Visualize the results
    ...
```

- The given skeleton code (`multivar_optimization_skeleton.py`)
  - Step #1) Derive the gradient vector, `fd =` $\left[\frac{\partial f}{\partial x_0} \quad \frac{\partial f}{\partial x_1}\right]$
  - Step #2) Implement your gradient descent

Assignment

- Mission
  - Complete the given skeleton code (`multivar_optimization_skeleton.py`)
  - Submit your code (`multivar_optimization.py`) and its figure (`multivar_optimization.png`)

- Condition
  - Please follow the above filename convention.
  - You can start from scratch (without using the given skeleton code).
    - However, you should use the given data.
  - You can freely change the given skeleton code if necessary.

- Submission
  - Deadline: **October 23, 2024 23:59** (firm deadline; no extension)
  - Where: e-Class > Assignments
  - Score: Max 10 points