8.2

$g(w) = x + b^T w + w^T C w$.

where, $C = \frac{1}{\beta} I$, $I \in \mathbb{R}^{N \times N}$, $\beta > 0$.

Rule: $(v^T C v)' = \nabla(v^T C v) = (C + C^T)v$

$\nabla g(w) = \nabla(x) + \nabla(b^T w) + \nabla(w^T C w)$ ← Rule $\nabla(u^T v) = u$

$\Rightarrow b + (C + C^T) w = 0$

↳ C가 Constant Symmetric Matrix이면, $C^T = C$ 대칭일이 성립하게 됨.

$\Rightarrow b + 2Cw = 0$

$2Cw = -b$

$w = -\frac{1}{2} C^{-1} b$

$w_{new} \leftarrow w_{old} - \lambda \frac{\partial L}{\partial w_{old}}$

$\lambda = \frac{1}{50}(w^4 + w^2 + 10w)$.

Param

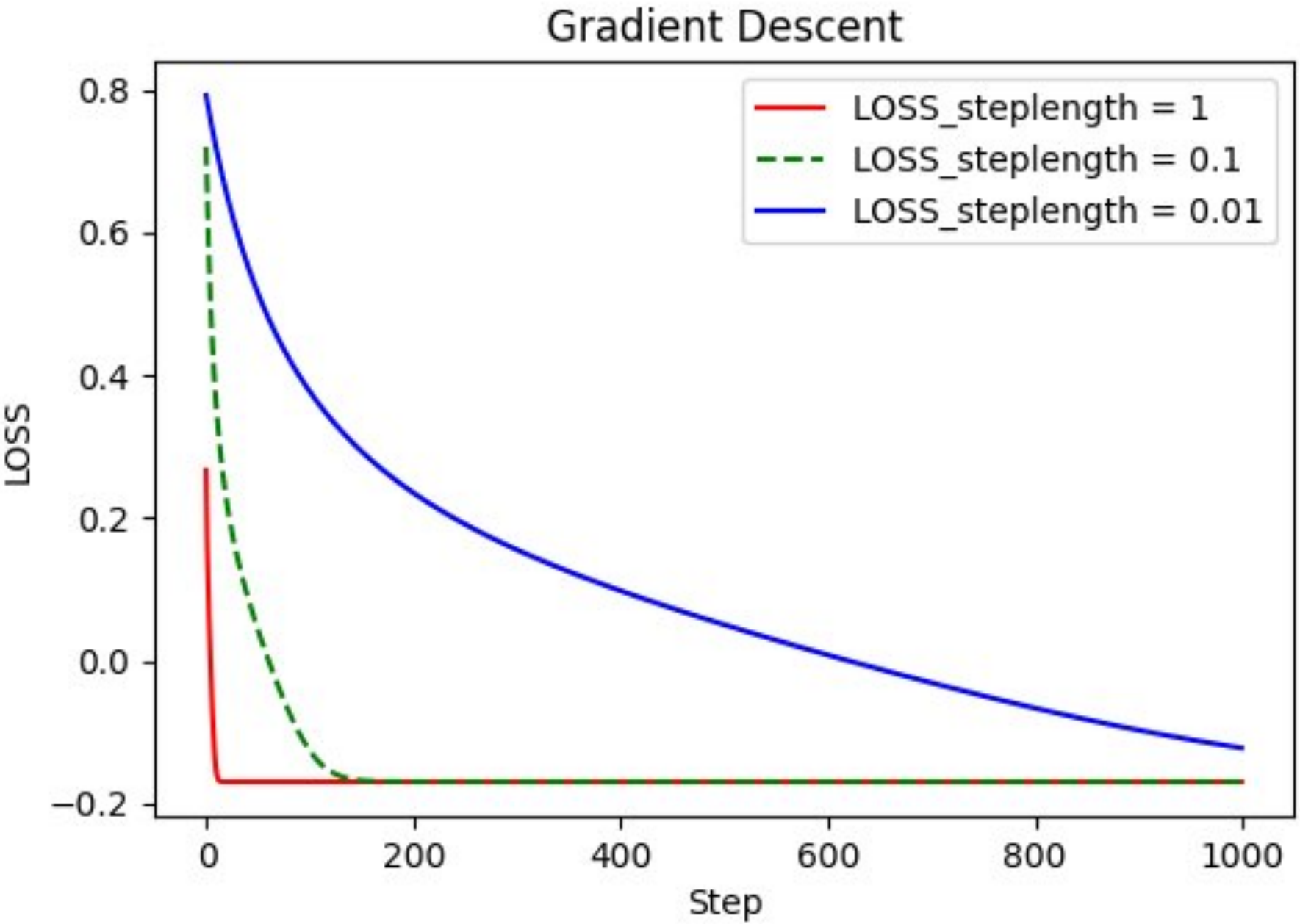
Init $w_0 = 2$.

Step = 1,000.

$\lambda = 1, 10^{-1}, 10^{-2}$

$\frac{\partial L}{\partial w} = \frac{1}{50}(4 \cdot w^3 + 2w + 10)$

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

# ======================================
# steplength가 1일때 stationary point에 converge하는 속도가 나머지 steplength(0.1, 0.01)보다 더 빠르다
# steplength가 클수록 처음 initial value는 더 작았다.
# 0의 근처에 수렴하기는 하였지만, 0에 더 approximate하려면 더 작은 steplength를 필요로 하고 그에 따른 step이 증가해야한다.
# ======================================

def loss_function(w):
  return (1/50) * (w ** 4 + w ** 2 + 10 * w)

def gradient_descent(w_old, steplength):
  w_new = w_old - steplength * ((1/50) * (4 * w_old ** 3 + 2 * w_old + 10))
  return w_new

step_length = [1, 0.1, 0.01]
init_w = 2
num_step = 1000

weight_store = np.zeros((len(step_length), num_step))
loss_store = np.zeros((len(step_length), num_step))

for i, step_length in enumerate(step_length):
    w = init_w
    for j in range(num_step):
        w = gradient_descent(w, step_length)
        weight_store[i, j] = w
        loss_store[i, j] = loss_function(w)

loss_first = loss_store[0, :]
loss_second = loss_store[1, :]
loss_third = loss_store[2, :]

plt.figure(figsize = (6, 4))
plt.plot(loss_first, 'r', label = "LOSS_steplength = 1")
plt.plot(loss_second, 'g--', label = "LOSS_steplength = 0.1")
plt.plot(loss_third, 'b', label = "LOSS_steplength = 0.01")
plt.xlabel('Step')
plt.ylabel('LOSS')
plt.title('Gradient Descent')
plt.legend()
plt.savefig('gradient_descent_plot.png')  # 이미지를 저장합니다.
plt.show()
```


Gradient Descent
LOSS_steplength = 1
LOSS_steplength = 0.1
LOSS_steplength = 0.01
LOSS
Step
0.8
0.6
0.4
0.2
0.0
−0.2
0
200
400
600
800
1000