VARIOUS GRADIENT METHODS

Theory

Chijie An
Who?
Where?
When?

Contents

1 Code for Optimal Gradient Method(Nestrov's)

```
#optimal gradient method
def opt_grad_method(fun, x0, tol, maxit, m, M):
    k = M / m
    q = (1 - 1/np.sqrt(k)) / (1 + 1/np.sqrt(k))
    x = x0
    y = x0
    f_all = []
    gnorm_all = []

    for _ in range(maxit,):
        print('this is iteration',_)
        print('y is',y)
        _,grad_y = fun(y)
        print('gradient is',grad_y)
        f0,g0 = fun(x)
        gnorm = np.linalg.norm(g0)
        gnorm_all.append(gnorm)

        # 更新x和y
        print(y.shape,grad_y.shape)
        print('1/M is',1/M)
        x_new = y - (1/M) * grad_y
        y = x_new + q * (x_new - x)
        x = x_new
        print('x_new is',x_new)
        # 记录历史信息
        f_all.append(fun(x)[0])
    optimal_value, _ = fun(x)
    return f_all, gnorm_all, optimal_value
```

Figure 1: Code for Optimal Gradient Method

2 Code for Original Gradient Method

3 Example 1

4 Example 2

5 Example 3

6 Example 4

7 Theoritical Proof for Hessian Matrix

We can represent the hessian matrix for Nesterov's worst example as,

$$\nabla^2 F(x) = \frac{M - m}{4} T + mI$$

```python
#the original gradient method with a fixed stepsize of 1/M
def grad_method_fixed1(fun, x0, tol, maxit, m, M ):
    f_all = []
    gnorm_all = []
    x = x0
    alpha = 0.25 #alpha is taken from 0-0.5, usually take 0.25
    beta = 0.5  # step size reduction factor, you may want to adjust this beta in (0,1)

    for _ in range(maxit):
        print(_)
        f0, g0 = fun(x)  # Get the function value and gradient at the current point
        if _==0:
          print('the gradient 0 is',g0)
        if _==1:
          print('x is',x)
          print('1/M is',1/M)
        gnorm = np.linalg.norm(g0)
        gnorm_all.append(gnorm)
        if gnorm < tol:  # Check the stopping criterion
            break

        t = 1/M
        while fun(x - t * g0)[0] > f0 - alpha * t * gnorm**2:  # Backtracking line search
            t *= beta

        x = x - t * g0  # Update the point
        #print('g0 is',g0)
        #print('the new x is',x)
        f_all.append(fun(x)[0])  # Evaluate function at the new point

    return f_all, gnorm_all
```

Figure 2: Code for Original Gradient Method with t=1/M

We show that T is positive semi definite, by choosin any vector v, we have:

$$v^T T v = \sum_{i=1}^{n-1} (z_i - z_{i+1})^2 + z_1^2 + z_n^2 \geq 0$$

Hence, we have that $\frac{M-m}{4}$ is also positive semi-definite. And we also have,

$$\nabla^2 F(x) - mI = \frac{M-m}{4} T \geq 0$$

So we have shown that $mI \leq \nabla^2 F(x)$. And also, the greatest eigenvalue of T is less than 4 (which can be derived using numerical results). We can have that,

$$\nabla^2 F(x) = \frac{M-m}{4} + mI \leq (M-m)I + mI = MI$$

Hence we have derived $\nabla^2 F(x) \leq MI$. Combining above, we can have,

$$mI \leq \nabla^2 F(x) \leq MI$$

Also, this can be further proved by numerical results. For in example 3, the min eigenvalue of $\frac{M-m}{4}\nabla^2 F(x) + m * I - m * i$ is 2.442238596776962e-06+0j, which is greater than zero, meaning $\nabla^2 F(x) - mi$ is positive definite. Also, the max eigenvalue for $\nabla^2 F(x) - MI$ is -2.442238595324157e-06, which means it is negative definite. For example 4, the min eigenvalue of $\nabla^2 F(x) - mI$ is 0.00024666609826732514+0j, which shows that this matrix is positive definite. Also, the max eigenvalue of $\nabla a^2 F(x) - MI$ is -0.0002466660983141487+0j, which shows that this matrix is negative definite.

```
def grad_method_fixed2(fun, x0, tol, maxit, m, M):
    f_all = []
    gnorm_all = []
    x = x0
    alpha = 0.25 #alpha is taken from 0-0.5, usually take 0.25
    beta = 0.5  # step size reduction factor, you may want to adjust this beta in (0,1)

    for _ in range(maxit):
        print(_)
        f0, g0 = fun(x)  # Get the function value and gradient at the current point
        if _==0:
          print('the gradient 0 is',g0)
        if _==1:
          print('x is',x)
        gnorm = np.linalg.norm(g0)
        gnorm_all.append(gnorm)
        if gnorm < tol:  # Check the stopping criterion
            break

        t = 2/(M+m)
        while fun(x - t * g0)[0] > f0 - alpha * t * gnorm**2:  # Backtracking line search
            t *= beta

        x = x - t * g0  # Update the point
        #print('g0 is',g0)
        #print('the new x is',x)
        f_all.append(fun(x)[0])  # Evaluate function at the new point

    return f_all, gnorm_all
```

Figure 3: Code for Original Gradient Method with t=2/M+m

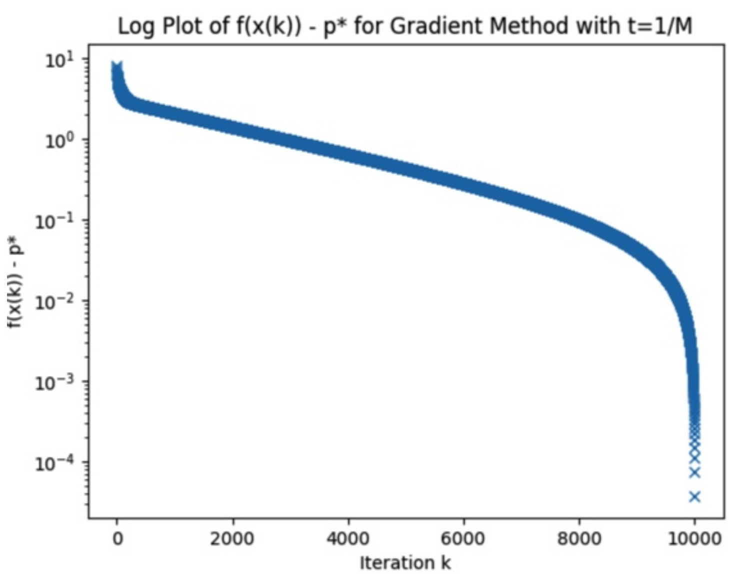

Figure 4: Example 1

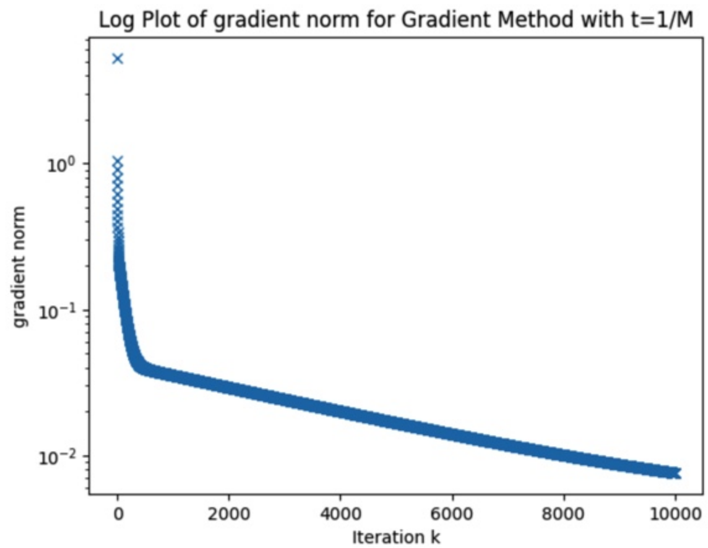

Figure 5: Example 1

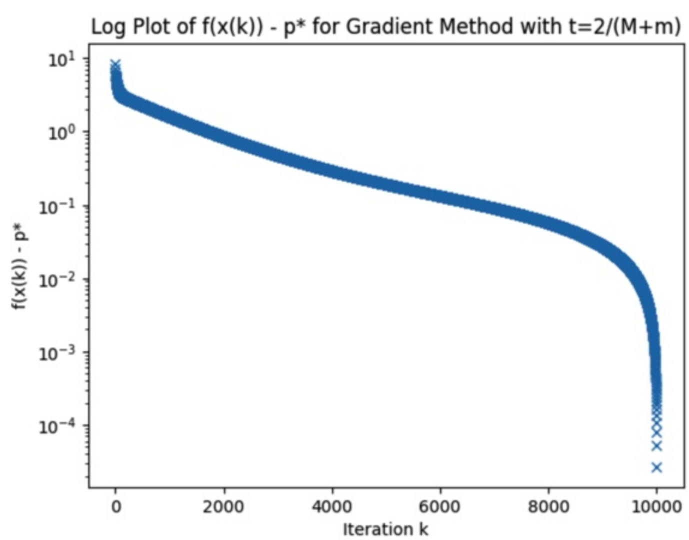

Figure 6: Example 1

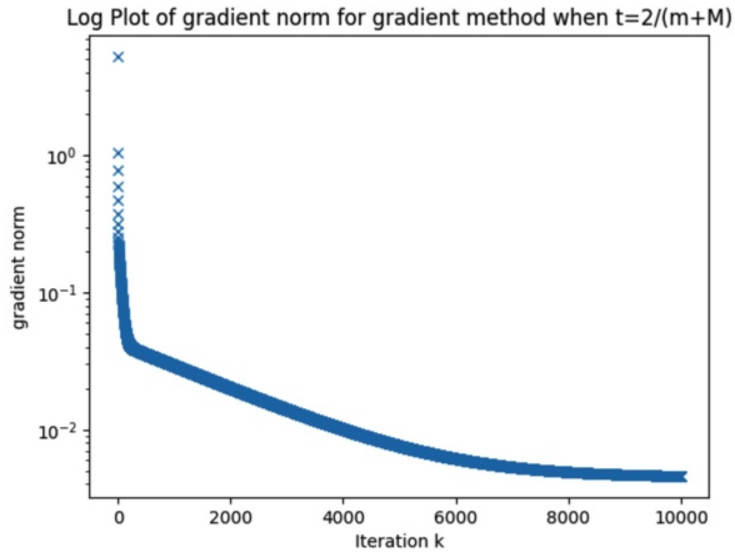

Figure 7: Example 1

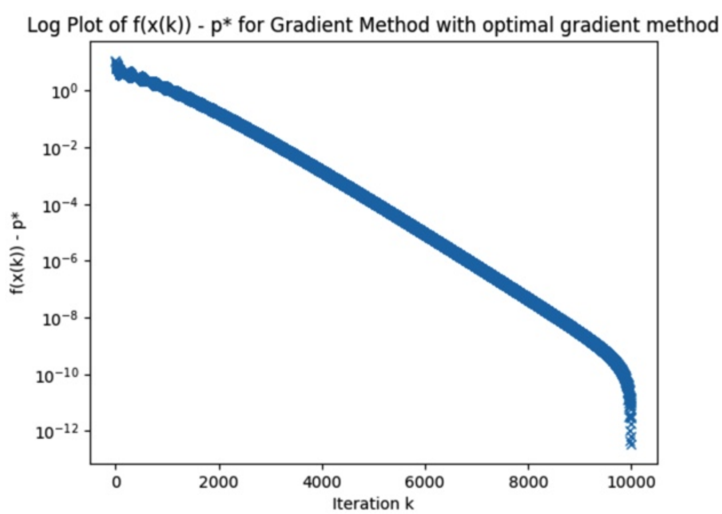

Figure 8: Example 1

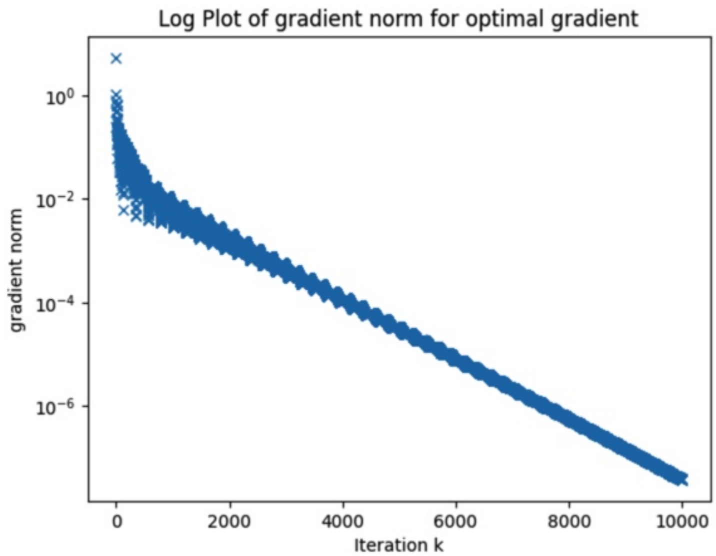

Figure 9: Example 1

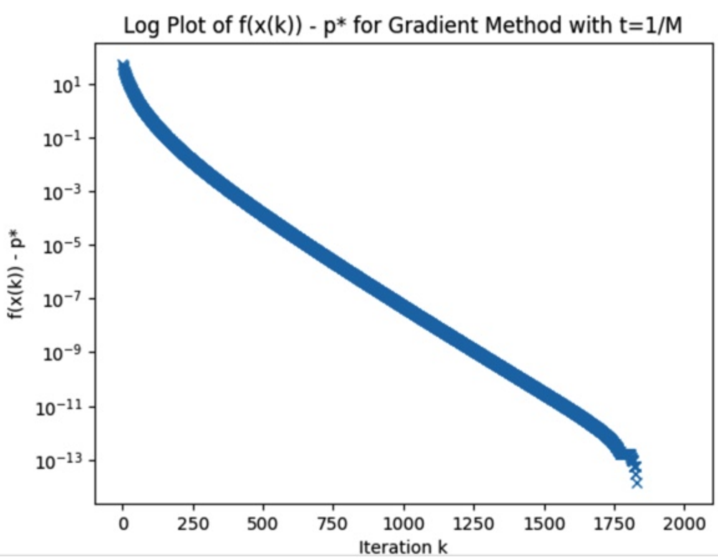

Figure 10: Example 2

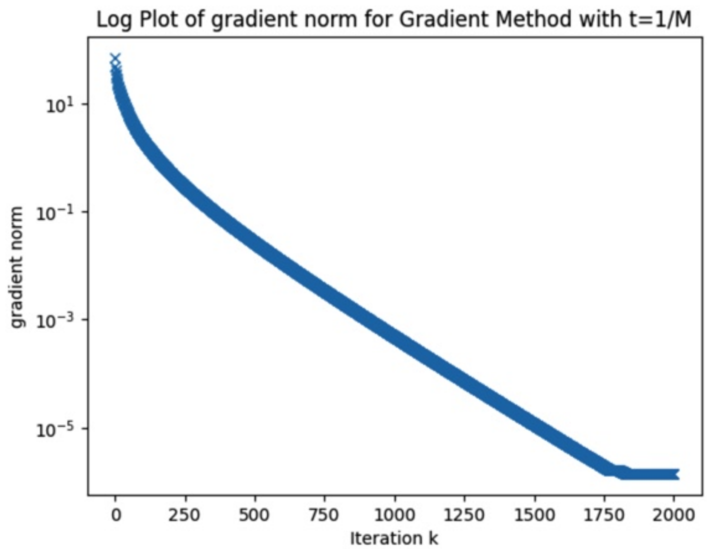

Figure 11: Example 2

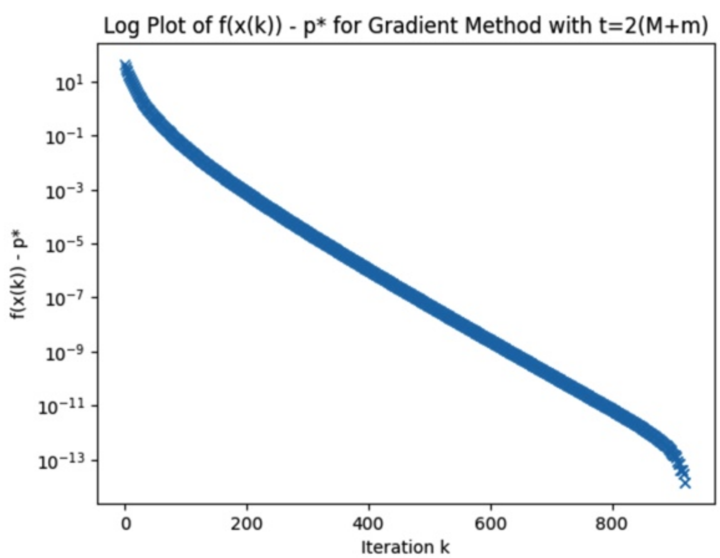

Figure 12: Example 2

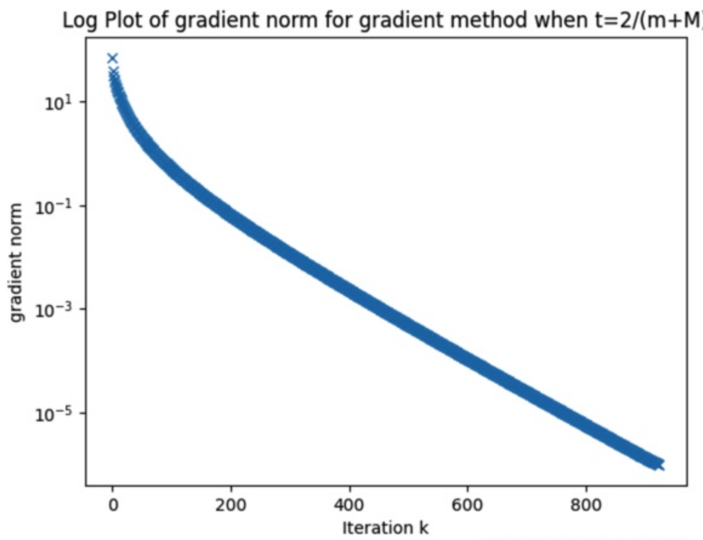

Figure 13: Example 2

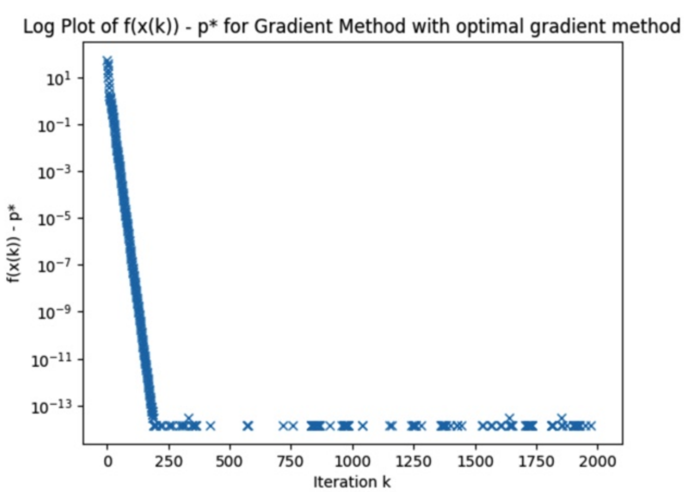

Figure 14: Example 2

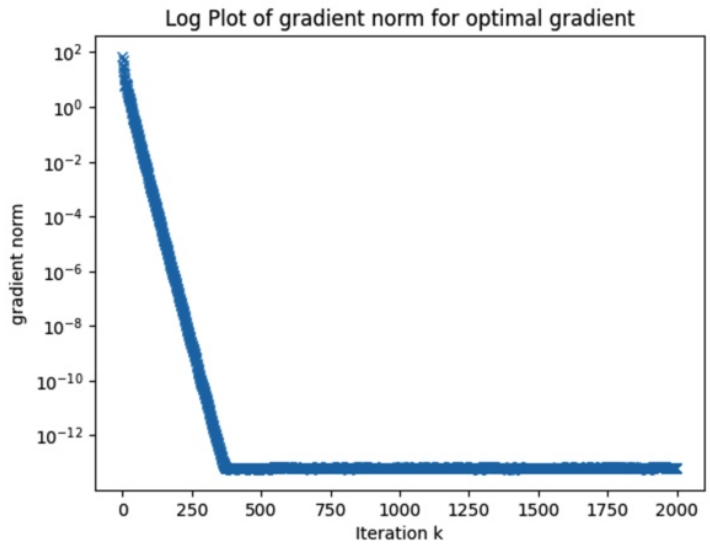

Figure 15: Example 2

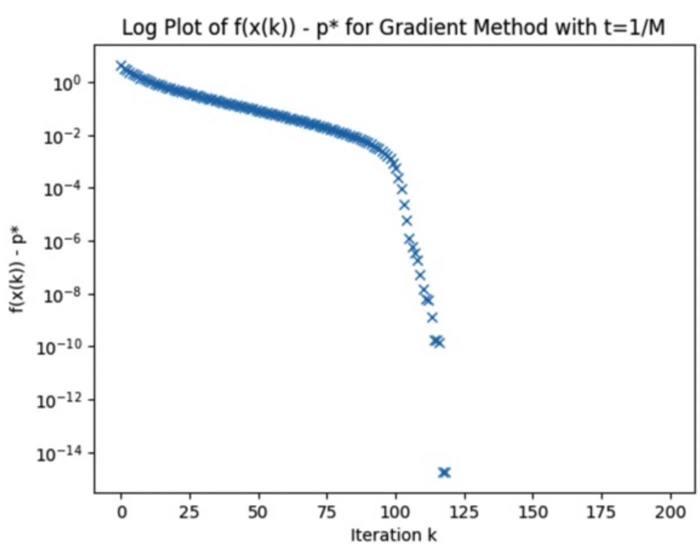

Figure 16: Example 3

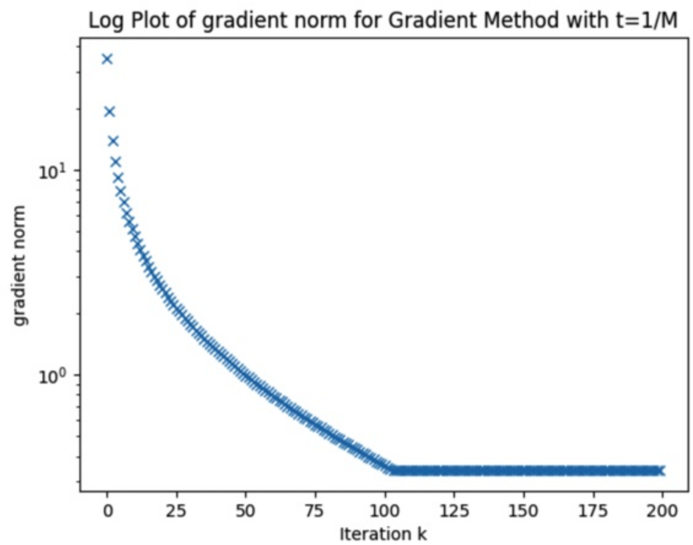

Figure 17: Example 3

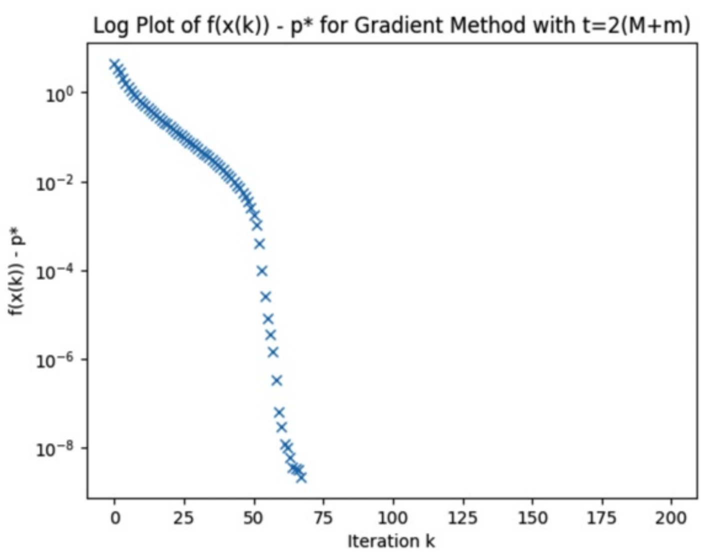

Figure 18: Example 3

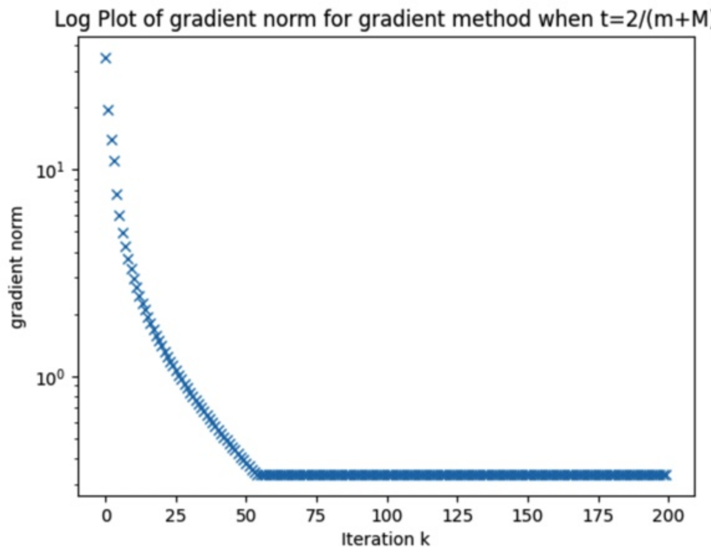

Figure 19: Example 3

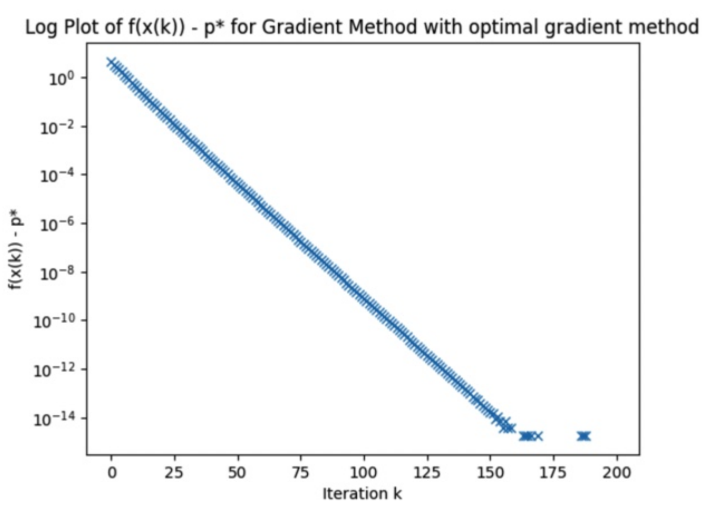

Figure 20: Example 3

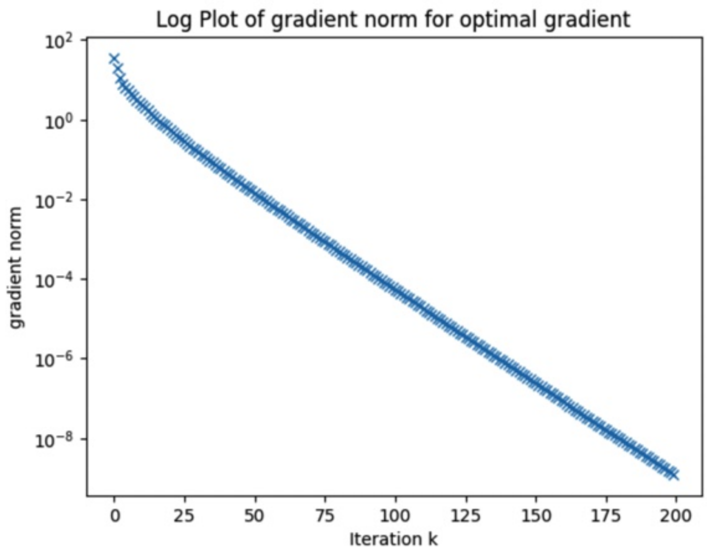

Figure 21: Example 3

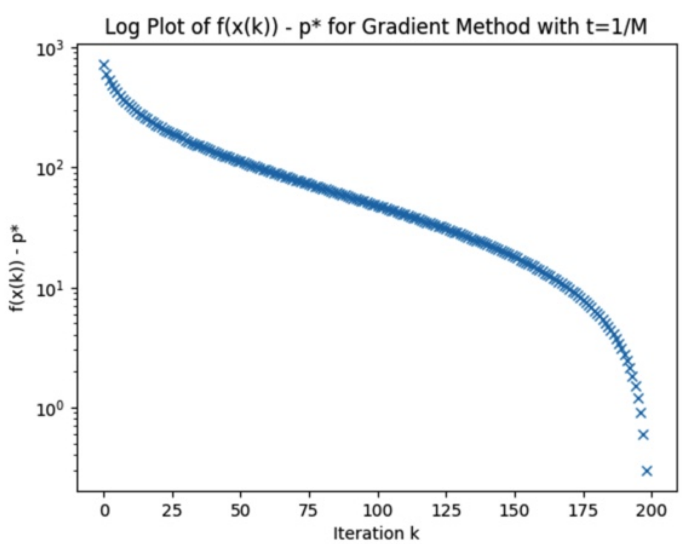

Figure 22: Example 4

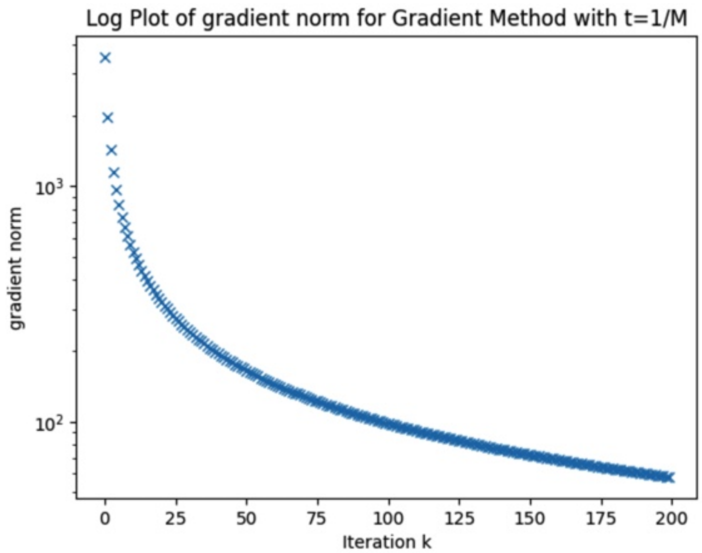

Figure 23: Example 4

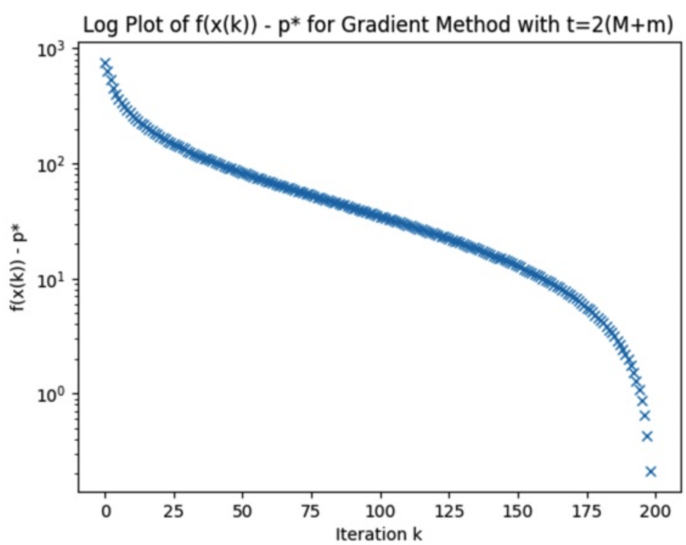

Figure 24: Example 4

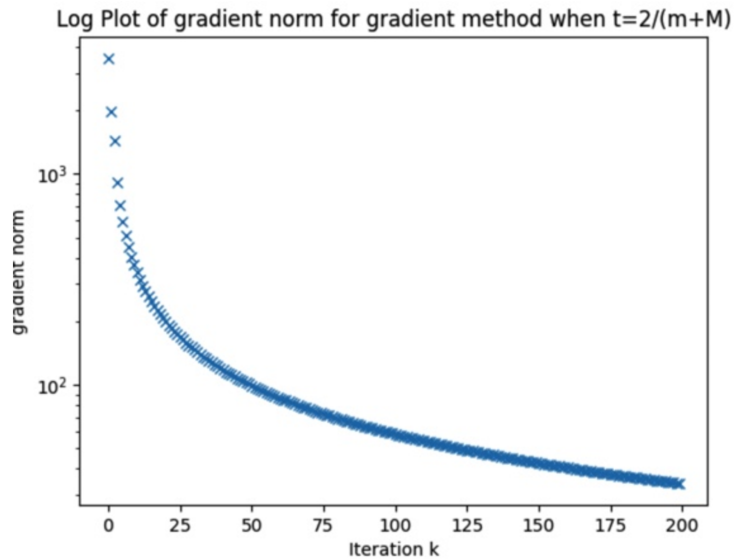

Figure 25: Example 4

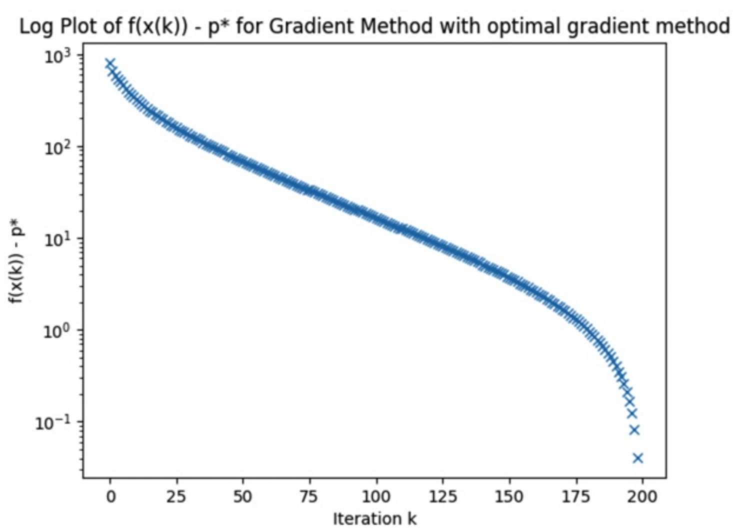

Figure 26: Example 4

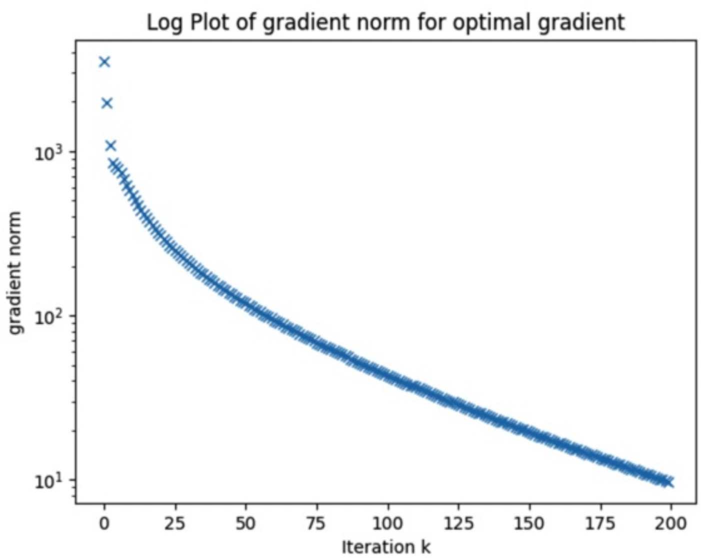

Figure 27: Example 4