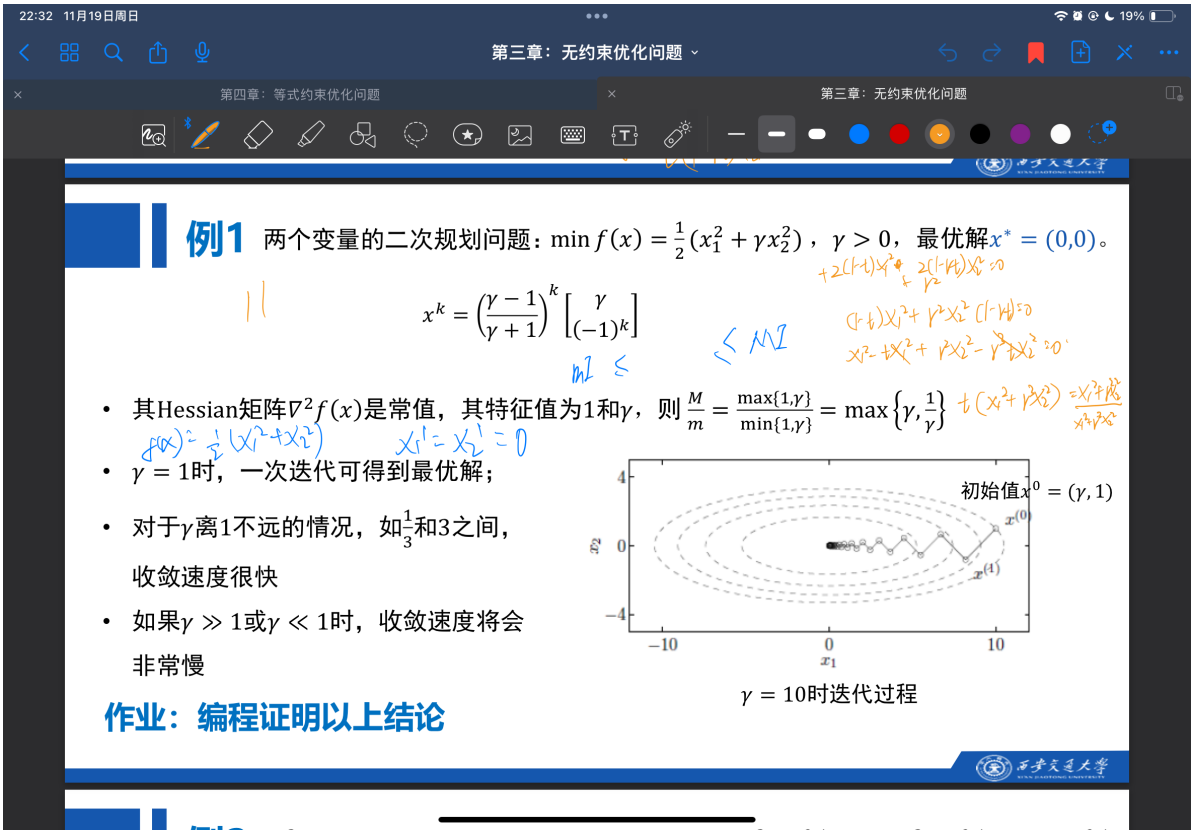
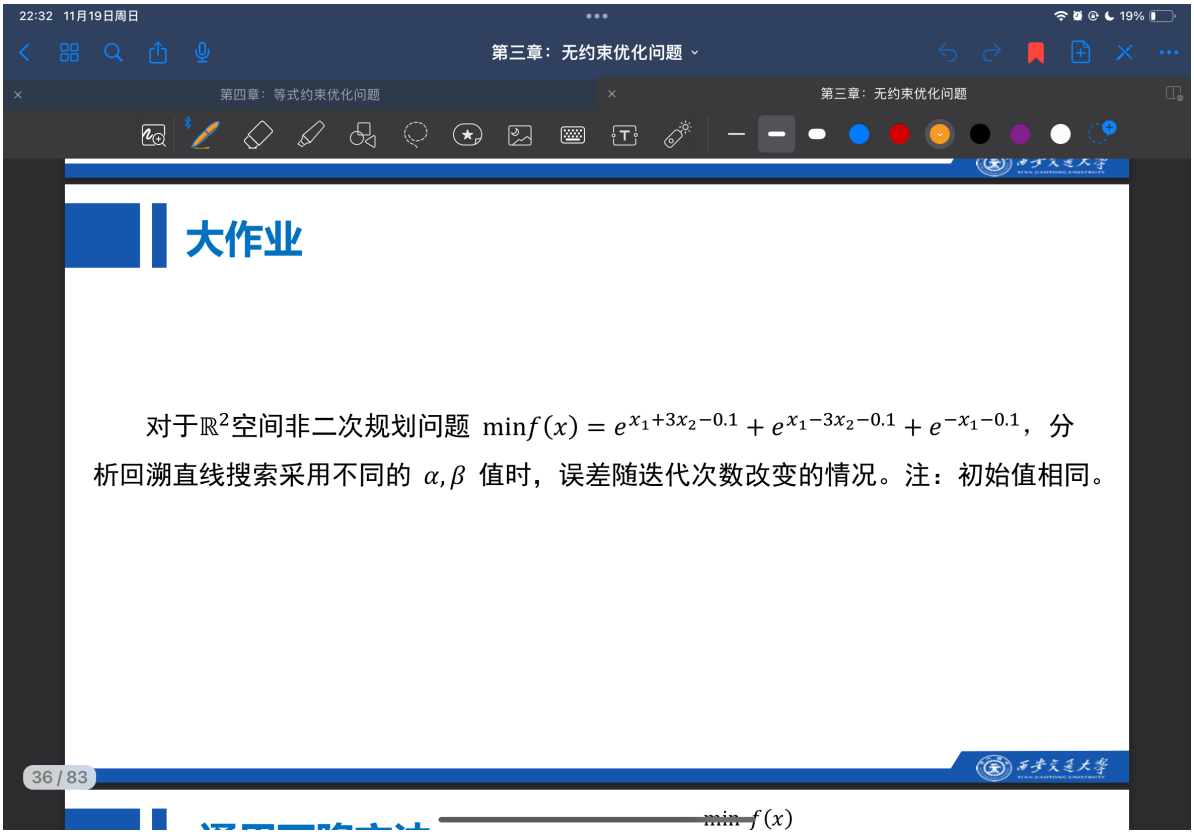
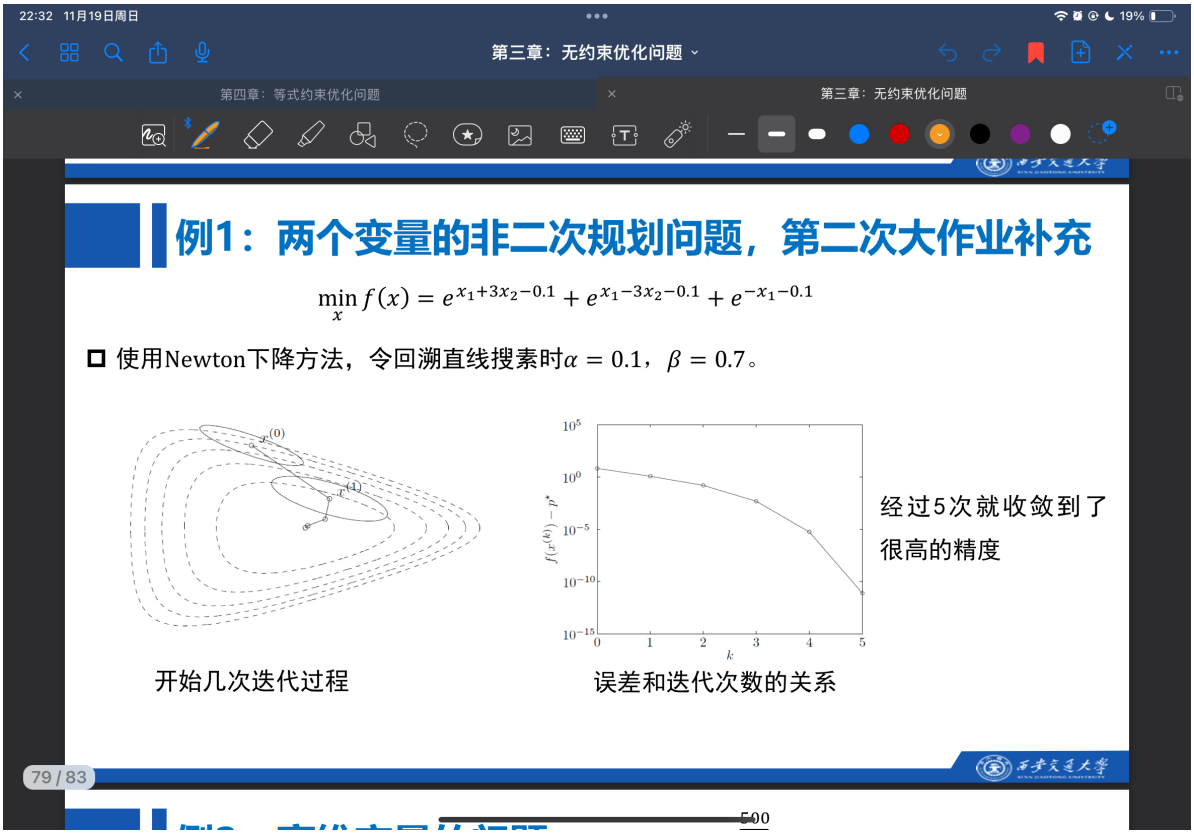
第二次大作业

1.问题描述







2.问题求解：

问题一：取不同的γ值进行计算，将每次迭代的结果保存，绘制到平面图。

部分代码如下：

```
5    def cal(r, epsilon):
6        k = 0
7        points = []
8        while True:
9            xk = ((r - 1) / (r + 1)) ** k * np.array([r, (-1) ** k])
10           points.append(xk)
11           if np.linalg.norm(xk, ord=2) <= epsilon:
12               break
13           k += 1
```

问题二：使用梯度下降算法，首先求出一个近似最优解，再在相同初始点，不同的α,β的值的情况下，观察误差随迭代次数的变化。

部分代码如下：

```python
def f(x):
    return np.exp(x[0] + 3 * x[1] - 0.1) + \
           np.exp(x[0] - 3 * x[1] - 0.1) + \
           np.exp(-x[0] - 0.1)


def df(x):
    return np.array([np.exp(x[0] + 3 * x[1] - 0.1) +
                     np.exp(x[0] - 3 * x[1] - 0.1) -
                     np.exp(-x[0] - 0.1),
                     3 * np.exp(x[0] + 3 * x[1] - 0.1) -
                     3 * np.exp(x[0] - 3 * x[1] - 0.1)])

def d2f(x):
    return np.array([[np.exp(x[0]+3*x[1]-0.1)+np.exp(x[0]-3*x[1]-0.1)+np.exp(-x[0]-0.1),
                      3*np.exp(x[0]+3*x[1]-0.1)-3*np.exp(x[0]-3*x[1]-0.1)],
                     [3*np.exp(x[0]+3*x[1]-0.1)-3*np.exp(x[0]-3*x[1]-0.1),
                      9*np.exp(x[0]+3*x[1]-0.1)+9*np.exp(x[0]-3*x[1]-0.1)]])
```

```python
def minf(x, alpha, beta, epsilon):
    while True:
        d = -df(x)
        if np.linalg.norm(df(x), ord=2) <= epsilon:
            break
        t = 1
        while f(x + t * d) > f(x) + alpha * t * np.transpose(df(x)).dot(d):
            t = beta * t
        x = x + t * d
    return f(x)

def cal(x, alpha, beta, num, epsilon=1e-5):
    errors = []
    while True:
        errors.append(f(x) - num)
        d = -df(x)
        if np.linalg.norm(df(x), ord=2) <= epsilon:
            break
        t = 1
        while f(x + t * d) > f(x) + alpha * t * np.transpose(df(x)).dot(d):
            t = beta * t
        x = x + t * d
    return errors
```

问题三：使用牛顿下降法，首先求出一个近似最优解，再在相同初始点，不同的α,β的值的情况下，观察误差随迭代次数的变化。

部分代码如下：

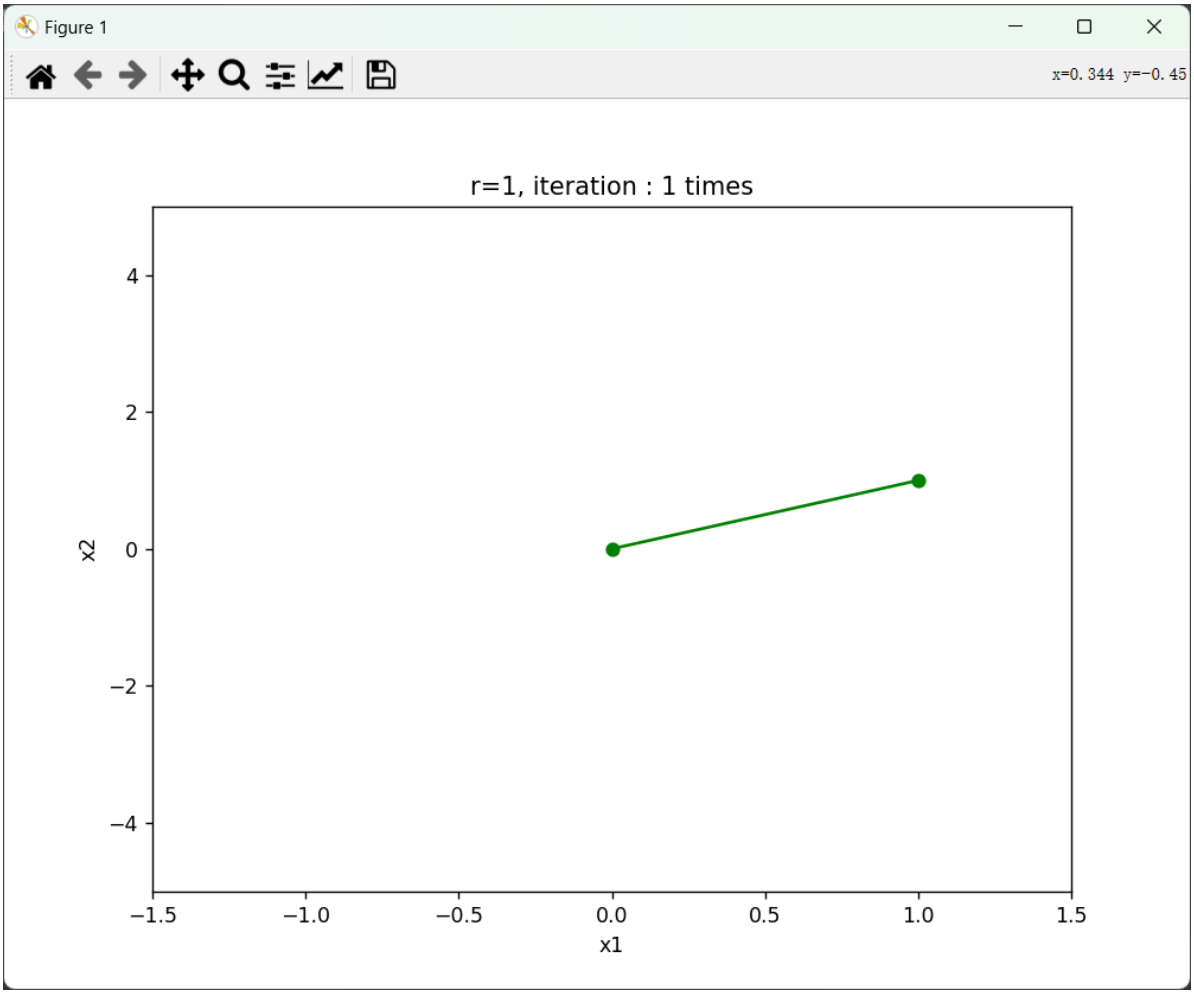
```python
def minf(x, alpha, beta, epsilon):
    while True:
        d = -df(x)
        if np.linalg.norm(df(x), ord=2) <= epsilon:
            break
        t = 1
        while f(x + t * d) > f(x) + alpha * t * np.transpose(df(x)).dot(d):
            t = beta * t
        x = x + t * d
    return f(x)

def newton(x, alpha, beta, num, epsilon):
    errors = []
    while True:
        d = df(x)
        d2 = d2f(x)
        errors.append(f(x) - num)
        d = -np.linalg.inv(d2).dot(d)
        lambda2 = d.transpose().dot(d2).dot(d)
        if 0.5 * lambda2 <= epsilon:
            break
        t = 1
        while f(x + t * d) > f(x) - alpha * t * lambda2:
            t = beta * t
        x = x + t * d
    return errors
```

```
其中f(x) df(x) d2f(x)的实现与第二题相同
```

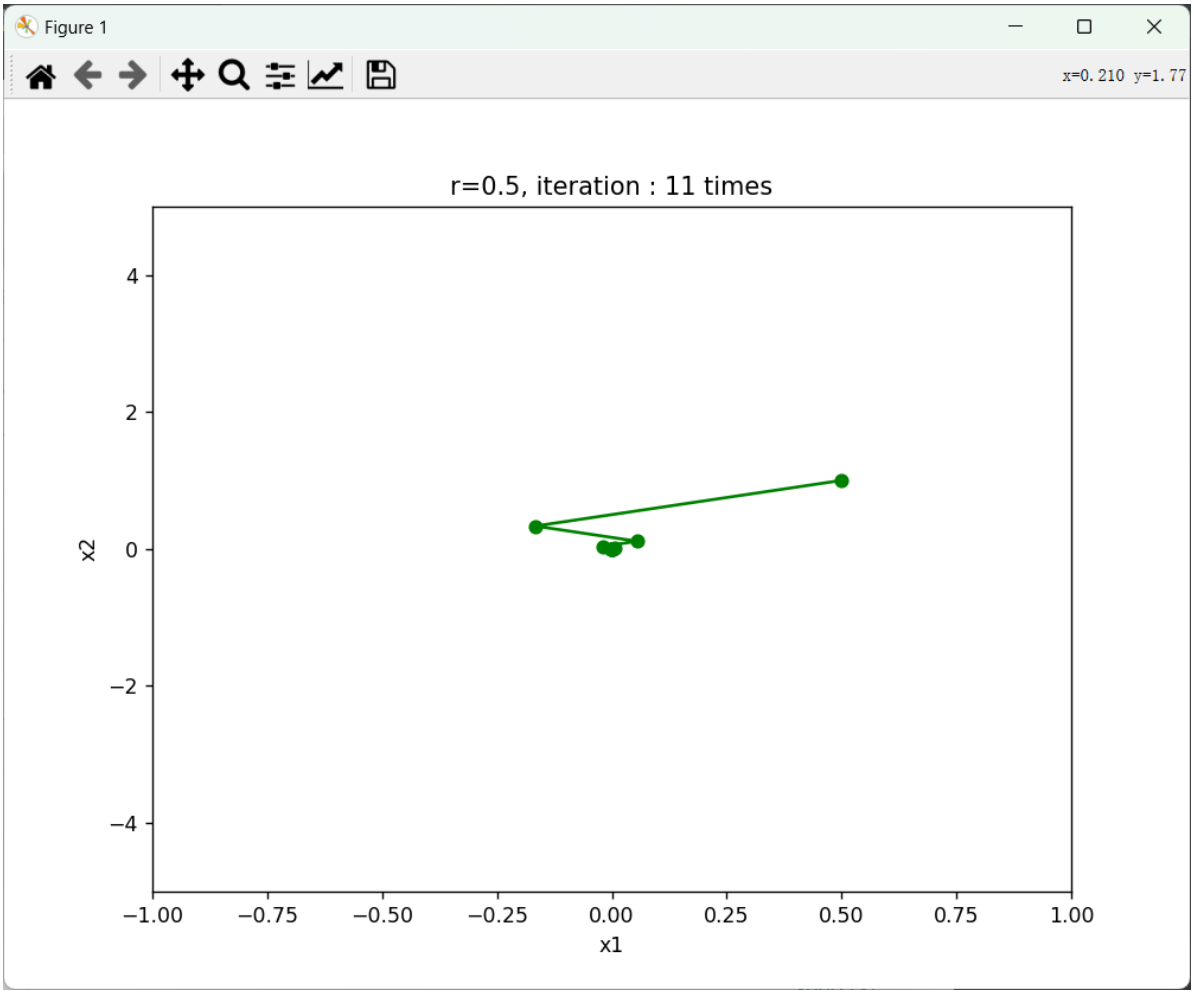
3.结果分析

题目一：

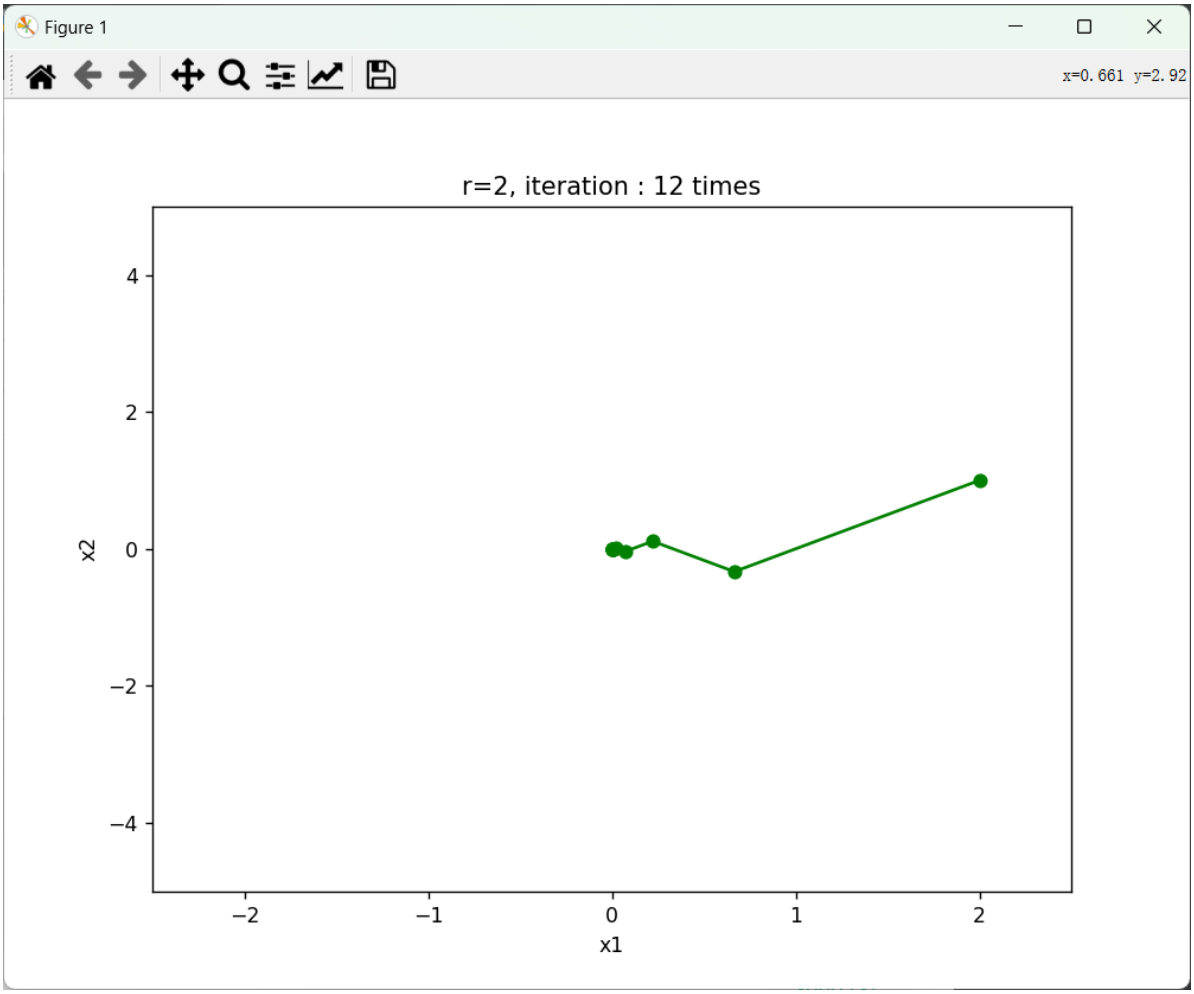
当γ=1时，只用了一次迭代就收敛



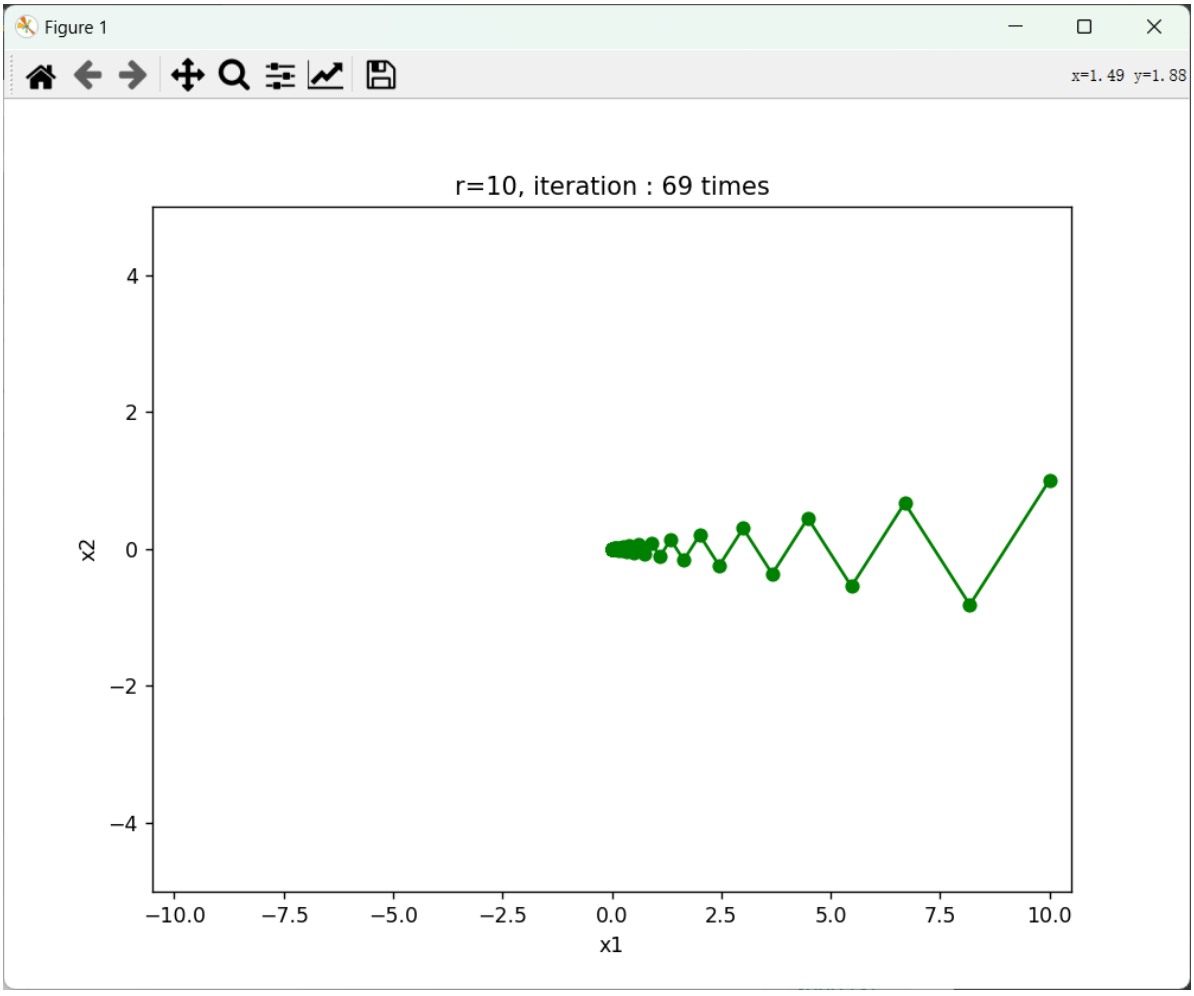
当γ=0.5时，进行了11次迭代



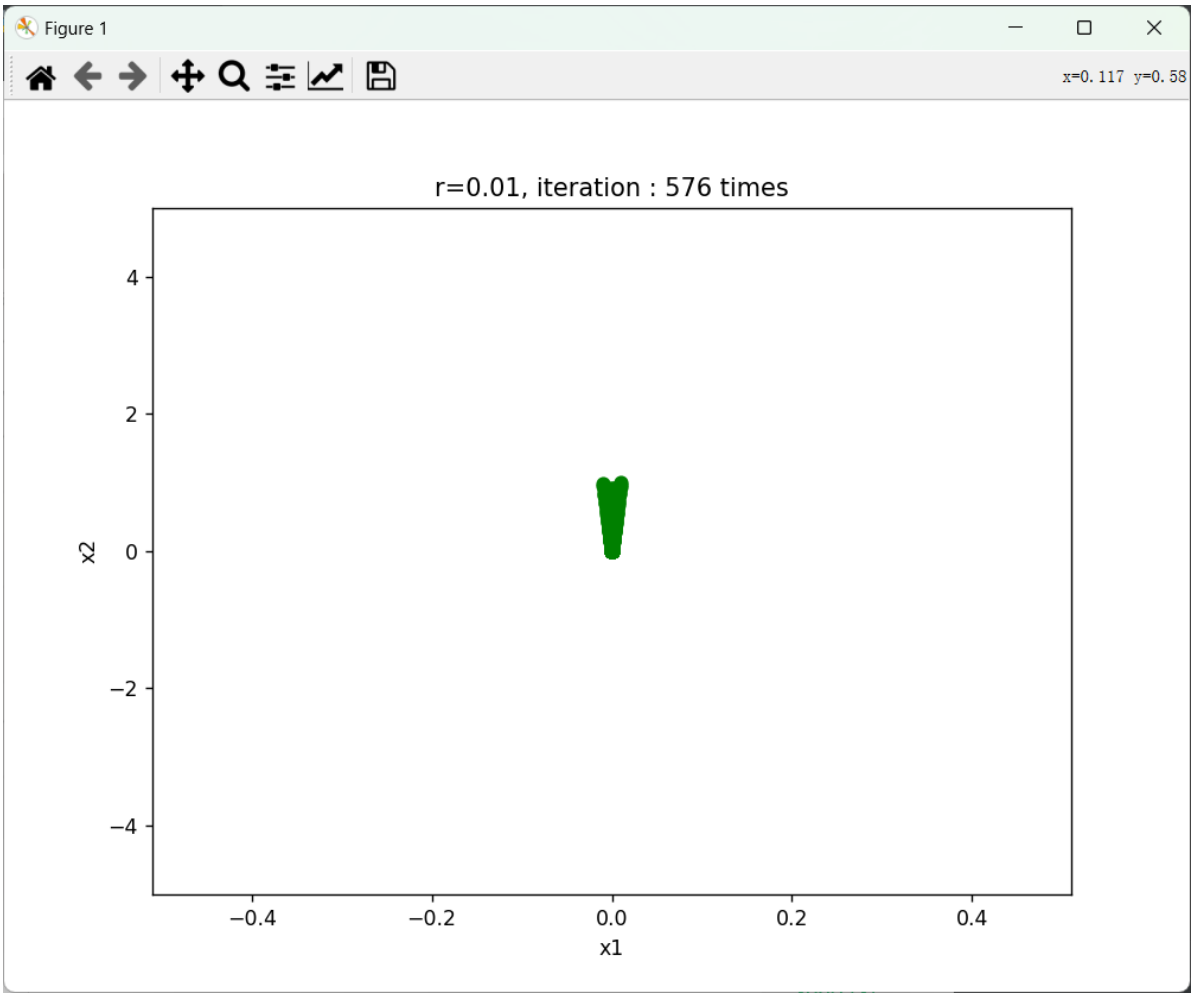
当γ=2时，进行了12次迭代



当γ=10时，进行了69次迭代



当γ=0.01时，进行了576次迭代

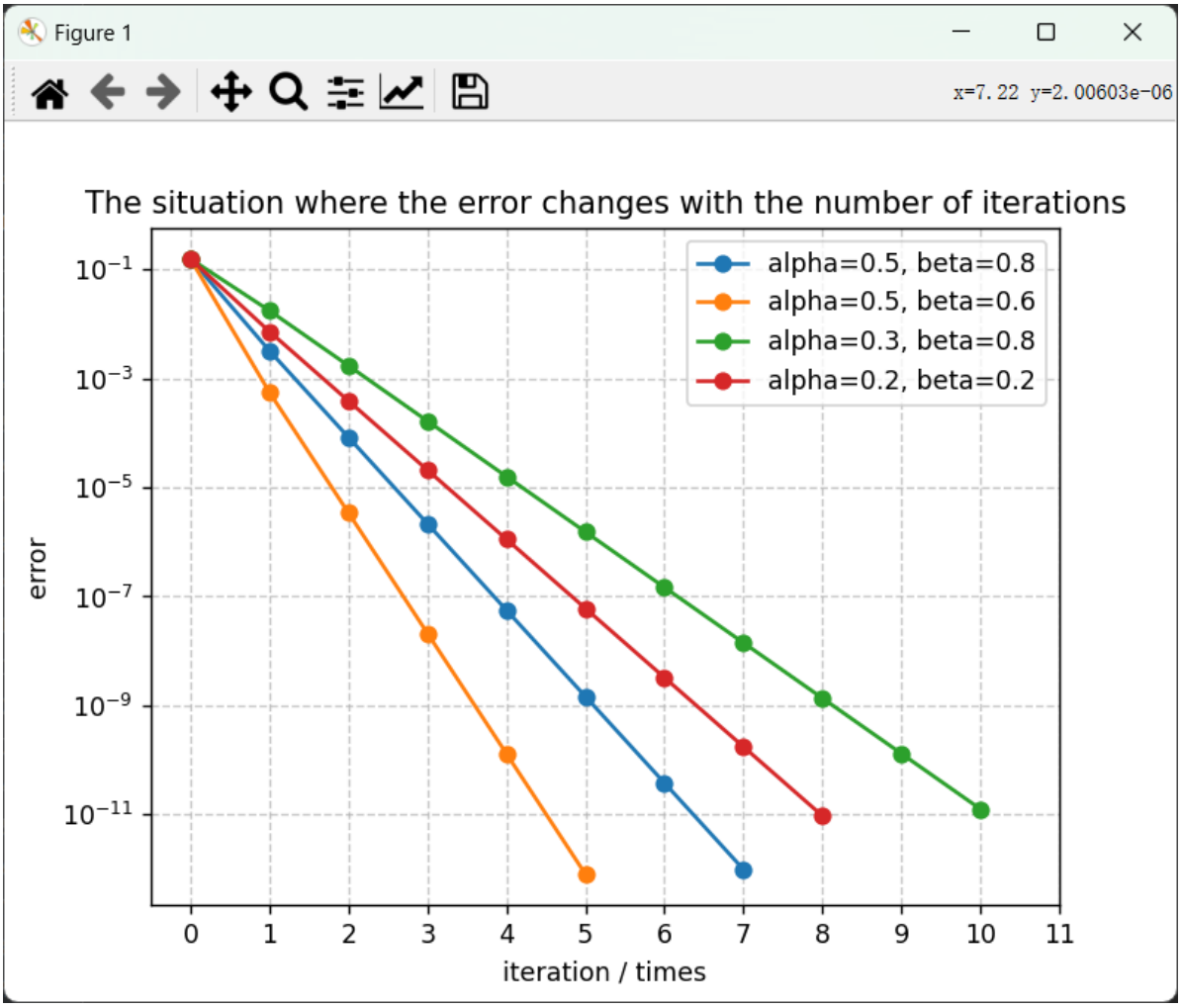


结果符合规律：

当$\gamma = 1$时，一次选代可得到最优解；当γ离1不远时，10次选代得到了最优解；

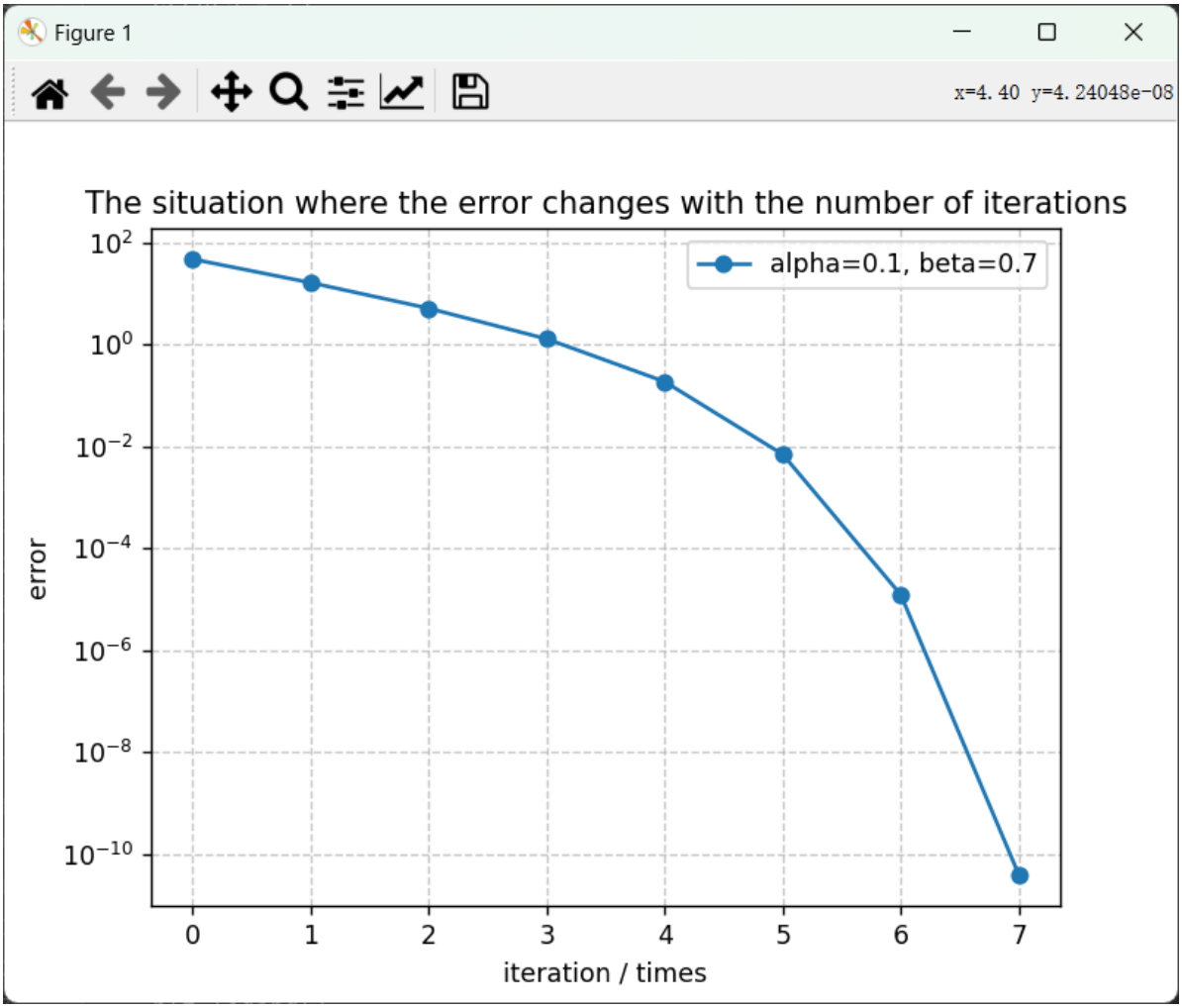
当$\gamma >> 1$或$\gamma << 1$时，经过了非常多的选代才能收敛。

问题二：



可以看出α,β的值需要在适中的范围内，过大或过小都会导致迭代次增加。当β 过小时误差的改变会非常慢，过大时回溯搜索需要迭代很多次。α对于迭代速度的影响没有β大，但仍然会影响收敛速度。（可以在代码内修改α,β的值观察得出，因考虑美观仅绘制部分）

问题三：



可在代码中修改γ和α的值，观察不同参数下牛顿下降方法的误差收敛情况