最优化方法实验 2：牛顿法和共轭梯度法

一、实验任务

基于Armijo步长规则，分别利用阻尼牛顿法和共轭梯度法求解无约束优化问：

$$\min_{x\in R^2} f(x)=100(x_1^2-x_2)^2+(x_1-1)^2 。$$

初始点选为$(0,0)^T$，终止准则选为$\|\nabla f(x_k)\|\le 10^{-5}$。

二、实验目的

通过本次实验，掌握阻尼牛顿法和共轭梯度法，并进行编程实现。

三、编程环境

所用设备为机房电脑，Windows 版本为 Windows 10 专业版，处理器为 11th Gen Intel(R) Core(TM) i5-11400 @ 2.60GHz 2.59 GHz，所用编程软件及库版本如表 1 所示。

表 1 编程环境

编程软件	库名	版本号
PyCharm 2019.3.1	numpy	1.24.4
	matplotlib	3.7.5

四、实验步骤

（1） 编写目标函数及其梯度、海森矩阵的程序；
（2） 编写Armijo准则程序；
（3） 编写阻尼牛顿法（Armijo步长）程序；
（4） 编写共轭梯度法（FR公式）程序；
（5） 调用上述算法对问题进行求解，绘制目标函数随迭代次数变化图像，并与实验1中的收敛速度进行比较。
（6）选取不同初始点，观察数值结果（迭代次数和目标函数值变化）差异

五、相关代码

（1） 目标函数、梯度函数及其海森矩阵函数程序如下：

```
# 目标函数
def fun(x):
    f = 100 * (x[0] ** 2 - x[1]) ** 2 + (x[0] - 1) ** 2
    return f

# 梯度函数
def gfun(x):
    gf = np.array([400 * x[0] * (x[0] ** 2 - x[1]) + 2 * (x[0] - 1), -200 * (x[0] **
2 - x[1])])
    return gf

# Hessian矩阵函数
def hfun(x):
    h = np.array([[1200 * x[0] ** 2 - 400 * x[1] + 2, -400 * x[0]],
                  [-400 * x[0], 200]])
    return h
```

（2）Armijo准则程序如下：

```
# Armijo准则
def armijo(xk, dk):
    beta = 0.5
    sigma = 0.2
    m = 0
    mmax = 20
    mk = 0
    while (m <= mmax):
        if (fun(xk + beta ** m * dk) <= fun(xk) + sigma * beta ** m * np.dot(gfun(xk),
dk)):
            mk = m
            break
        m = m + 1
    alpha = beta ** mk
    newxk = xk + alpha * dk
    fk = fun(xk)
    newfk = fun(newxk)
    return mk, alpha, newxk, fk, newfk
```

（3）阻尼牛顿法程序如下：

```
# 阻尼牛顿法
def damped_newton(x0):
    maxk = 5000  # 最大迭代次数
 k = 0
    Err = []
    tk = []
    epsilon = 1e-5
    newton_history = []  # 存储每次迭代的点和函数值
 while (k < maxk):
        g = gfun(x0)  # 计算梯度
   h = hfun(x0)  # 计算Hessian矩阵
   dk = -solve(h, g)  # 计算搜索方向
   m, alpha, x0, fk, newfk = armijo(x0, dk)
        newton_history.append((x0, fun(x0)))  # 存储迭代点和函数值
   Err.append(norm(g))
        tk.append(k)
        if (norm(g) <= epsilon):
            break
        k += 1
    return x0, fun(x0), k, Err, tk, newton_history
```

（4）共轭梯度法程序如下：

```
# 共轭梯度法
def frcg(x0):
    maxk = 5000  # 最大迭代次数
 rho = 0.6
    sigma = 0.4
    k = 0
    Err = []
    tk = []
    epsilon = 1e-5
```

```
    n = len(x0)
    g0 = gfun(x0)  # 计算初始梯度
cg_history = []  # 存储每次迭代的点和函数值
while (k < maxk):
        g = gfun(x0)  # 计算梯度
  itern = k - (n + 1) * int(np.floor(k / (n + 1)))
        itern = itern + 1
        if (itern == 1):
            d = -g
        else:
            beta = np.dot(g, g) / np.dot(g0, g0)
            d = -g + beta * d
        if (norm(g) < epsilon):
            break
        m = 0
        mk = 0
        while (m < 20):
            if (fun(x0 + rho ** m * d) < fun(x0) + sigma * rho ** m * np.dot(g, d)):
                mk = m
                break
            m = m + 1
        x0 = x0 + rho ** mk * d
        cg_history.append((x0, fun(x0)))
  val = fun(x0)
        Err.append(norm(g))
        tk.append(k)
        g0 = g
        k += 1
    return x0, val, k, Err, tk, cg_history
```

（5）调用上述算法对问题进行求解，绘制目标函数随迭代次数变化图像，如下：

```
x0 = np.array([0, 0])
x_newton, val_newton, k_newton, Err_newton, tk_newton, newton_history = damped_newton(x0)
print("Damped Newton Method:")
print("Optimal point:", x_newton)
print("Minimum value:", val_newton)
print("Iteration history (point, function value):")
iterator = 1
for point, value in newton_history:
    print(f"Iterator: {iterator}, Point: {point}, Function value: {value}")
    iterator += 1
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(tk_newton, Err_newton, label='Damped Newton Method')
plt.xlabel('Iteration')
plt.ylabel('Gradient Norm')
plt.title('Gradient Norm vs. Iteration (Damped Newton)')
plt.legend()
plt.tight_layout()
plt.savefig(r'C:\Users\Administrator\Desktop')
plt.show()
# 共轭梯度法同上，不再重复
```

六、实验结果及分析

两种方法所得最优值点及最优值如表 2。

表 2 实验结果图

初始点	目标函数				策略准则
$(0,0)^T$	$\min_{x\in R^2} f(x)=100(x_1^2-x_2)^2+(x_1-1)^2$				Armijo准则
	方法	迭代次数	最优值点	最优值	终止准则
	阻尼牛顿法	11	(0. 99979926, 0. 99957278)	1. 068e−7	$\|\nabla f(x_k)\|\le 10^{-5}$
	共轭梯度法	20	(1. 0013509, 1. 00279924)	2. 739e−6	

两种方法中目标函数值随迭代次数的变化曲线如图 1 所示。由图可知，基于 Armijo 准则的阻尼牛顿法和共轭梯度法均得了较好的效果，仅通过20次以下的迭代即达到了终止准则条件。

进一步的，将其与实验1中的梯度下降法迭代曲线相比较，可以看出显然阻尼牛顿法和共轭梯度法的梯度下降速度更快（在实验1中，梯度下降法通过2512次迭代才截停迭代），结合课程所学分析原因所在：梯度下降法每次均沿着不一定为最优的负梯度方法进行搜索，且容易产生锯齿现象；阻尼牛顿法在每次迭代时加入了步长搜索，从而使得迭代更快且更精准；共轭梯度法则利用共轭方向的性质，在n维空间搜索时理论上n次迭代即可达到极小值点。

在本次实验中，使用了阻尼牛顿法和共轭梯度法求解与实验1相同的无约束非线性优化问题，通过所得结果及与实验1结果的对比可知对于算法优化的重要性，简单而言能够将计算的复杂度从大指数级别(O(n^n))降低至O(n)，对于实际应用有重要意义。

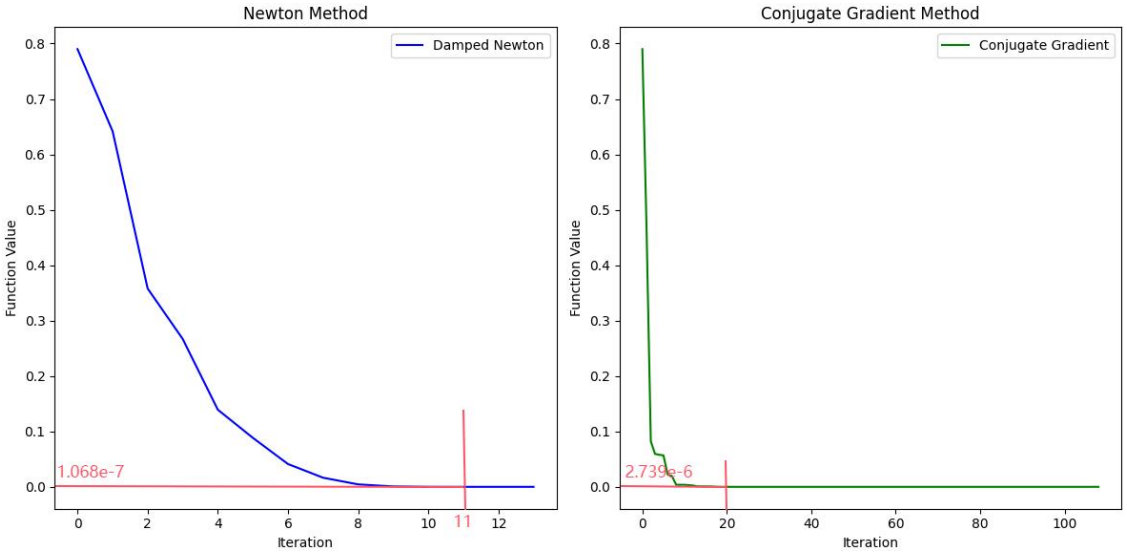

图1 迭代折线图