最优化作业报告

中山大学
智能工程学院

姓名: 常毅成

学号: 22354010

邮箱 1: 2937212699@qq.com

2025 年 5 月 22 日 20:02 +08

摘　要

本实验报告基于《优化理论与应用》课程的期中考查作业 3，系统分析了最速下降法、拟牛顿法（BFGS）、共轭梯度法和坐标轮换法在不同优化问题中的应用。报告通过数学推导，探讨了目标函数的梯度、Hessian 矩阵及其正定性，验证了函数的凸性与全局最优解的存在性，揭示了二次型函数和非二次型函数的收敛特性。结合 MATLAB 编程实现，验证了理论分析的正确性，并总结了课程知识点、个人学习难点及优化方法在机器学习、控制系统等领域的潜在应用。

目录

1 题目一

1.1 题目要求

使用最速下降法求解以下优化问题：

$$\min f(X) = 5x_1^2 + 2x_2^2 + x_1x_2, \quad X_0 = [3, -2]^T, \quad \varepsilon = 0.001$$

1.2 题目分析

目标函数为二次型函数：

$$f(X) = 5x_1^2 + 2x_2^2 + x_1x_2$$

计算梯度：

$$\nabla f(X) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 10x_1 + x_2 \\ 4x_2 + x_1 \end{bmatrix}$$

Hessian 矩阵：

$$\nabla^2 f(X) = \begin{bmatrix} 10 & 1 \\ 1 & 4 \end{bmatrix}$$

检查 Hessian 正定性：- 主子式：$10 > 0$ - 行列式：$\det(\nabla^2 f) = 10 \cdot 4 - 1 \cdot 1 = 39 > 0$

Hessian 正定，表明函数为凸函数，存在唯一全局最小值。

令梯度为零：

$$\begin{cases} 10x_1 + x_2 = 0 \\ x_1 + 4x_2 = 0 \end{cases}$$

从第一方程：$x_2 = -10x_1$。代入第二方程：

$$x_1 + 4(-10x_1) = x_1 - 40x_1 = -39x_1 = 0 \implies x_1 = 0 \implies x_2 = 0$$

驻点为 $X = [0, 0]^T$，函数值 $f(0, 0) = 0$，为全局最小值。

最速下降法通过沿负梯度方向 $d_k = -\nabla f(X_k)$ 迭代优化，步长 α_k 通过线搜索确定。算法步骤如下：

1. 初始点 $X_0 = [3, -2]^T$，计算初始梯度。
2. 设定搜索方向 $d_k = -\nabla f(X_k)$。
3. 使用线搜索确定步长 α_k。
4. 更新 $X_{k+1} = X_k + \alpha_k d_k$，检查 $\|\nabla f(X_k)\| < \varepsilon = 0.001$。

由于函数为二次型且 Hessian 正定，最速下降法保证收敛，但可能因条件数较高（特征值 $\lambda_1 \approx 10.37, \lambda_2 \approx 3.63$，条件数 $\lambda_1/\lambda_2 \approx 2.86$）需要较多迭代。

1.3 代码与结果

```
% 目标函数
f = @(x) 5*x(1)^2 + 2*x(2)^2 + x(1)*x(2);
% 梯度
grad_f = @(x) [10*x(1) + x(2); 4*x(2) + x(1)];

% 初始点
x = [3; -2];
epsilon = 0.001;
max_iter = 1000;
iter = 0;

fprintf('迭代次数 | x1       | x2       | f(x)     | ||grad||\n');
fprintf('-------------------------------------------------\n');

while norm(grad_f(x)) > epsilon && iter < max_iter
    % 计算搜索方向
    d = -grad_f(x);

    % 线搜索
    line_search = @(alpha) f(x + alpha*d);
    alpha = fminbnd(line_search, 0, 1);

    % 更新点
    x = x + alpha*d;
    iter = iter + 1;

    % 显示结果
    fprintf('%9d | %.6f | %.6f | %.6f | %.6f\n', ...
            iter, x(1), x(2), f(x), norm(grad_f(x)));
end

fprintf('\n最终点: [%.6f, %.6f]\n', x(1), x(2));
fprintf('目标函数值: %.6f\n', f(x));
fprintf('梯度范数: %.6f\n', norm(grad_f(x)));
fprintf('迭代次数: %d\n', iter);
```

图 1: 题目一代码

```
>> t1
迭代次数 | x1       | x2       | f(x)     | ||grad||
-------------------------------------------------
        1 | 0.042820 | -1.471932 | 4.279308 | 5.937368
        2 | 0.273147 | -0.182098 | 0.389627 | 2.589703
        3 | 0.003899 | -0.134018 | 0.035475 | 0.540592
        4 | 0.024870 | -0.016580 | 0.003230 | 0.235790
        5 | 0.000355 | -0.012202 | 0.000294 | 0.049220
        6 | 0.002264 | -0.001510 | 0.000027 | 0.021468
        7 | 0.000032 | -0.001111 | 0.000002 | 0.004481
        8 | 0.000206 | -0.000137 | 0.000000 | 0.001955
        9 | 0.000003 | -0.000101 | 0.000000 | 0.000408

最终点: [0.000003, -0.000101]
目标函数值: 0.000000
梯度范数: 0.000408
迭代次数: 9
```

图 2: 题目一结果

2 题目二

2.1 题目要求

使用拟牛顿法（BFGS）求解:

$$\min f(X) = x_1^2 + 4x_2^2 + 3x_1x_2, \quad X_0 = [1,1]^T, \quad \varepsilon = 0.01$$

2.2 实验记录

目标函数：

$$f(X) = x_1^2 + 4x_2^2 + 3x_1x_2$$

计算梯度：

$$\nabla f(X) = \begin{bmatrix} 2x_1 + 3x_2 \\ 8x_2 + 3x_1 \end{bmatrix}$$

Hessian 矩阵：

$$\nabla^2 f(X) = \begin{bmatrix} 2 & 3 \\ 3 & 8 \end{bmatrix}$$

- 主子式：$2 > 0$ - 行列式：$2 \cdot 8 - 3 \cdot 3 = 16 - 9 = 7 > 0$

Hessian 正定，函数为凸函数，存在唯一全局最小值。

令 $\nabla f(X) = 0$：

$$\begin{cases} 2x_1 + 3x_2 = 0 \\ 3x_1 + 8x_2 = 0 \end{cases}$$

解得：$x_1 = 0, x_2 = 0$。驻点 $X = [0, 0]^T$，$f(0, 0) = 0$，为全局最小值。

拟牛顿法（BFGS）通过近似 Hessian 逆 H_k 计算搜索方向 $d_k = -H_k \nabla f(X_k)$，并使用线搜索确定步长。BFGS 更新公式为：

$$H_{k+1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k} \right) H_k \left(I - \frac{y_k s_k^T}{y_k^T s_k} \right) + \frac{s_k s_k^T}{y_k^T s_k}$$

其中 $s_k = X_{k+1} - X_k$，$y_k = \nabla f(X_{k+1}) - \nabla f(X_k)$。初始 H_0 取单位矩阵，步长 α_k 通过线搜索确定。

函数为二次型，BFGS 理论上可在 $n = 2$ 次迭代内收敛到精确解，但数值计算可能因线搜索精度需更多步。

2.3 代码与结果

```
% 目标函数
f = @(x) x(1)^2 + 4*x(2)^2 + 3*x(1)*x(2);
% 梯度
grad_f = @(x) [2*x(1) + 3*x(2); 8*x(2) + 3*x(1)];

% 初始点
x = [1; 1];
epsilon = 0.01;
max_iter = 100;
iter = 0;

% 初始化Hessian逆近似为单位矩阵
H = eye(2);

fprintf('迭代次数 | x1       | x2       | f(x)     | ||grad||\n');
fprintf('-------------------------------------------------\n');

% 初始梯度
g = grad_f(x);
fprintf('%9d | %.6f | %.6f | %.6f | %.6f\n', ...
        iter, x(1), x(2), f(x), norm(g));

while norm(g) > epsilon && iter < max_iter
    % 计算搜索方向
    d = -H * g;

    % 线搜索
    line_search = @(alpha) f(x + alpha*d);
    alpha = fminbnd(line_search, 0, 1);

    % 更新点
    s = alpha * d; % 位移
    x_new = x + s;

    % 计算梯度差
    g_new = grad_f(x_new);
    y = g_new - g;

    % BFGS更新H
    if y'*s > 0 % 确保更新稳定性
        rho = 1 / (y'*s);
        H = (eye(2) - rho*s*y') * H * (eye(2) - rho*y*s') + rho*(s*s');
    end

    % 更新变量
    x = x_new;
    g = g_new;
    iter = iter + 1;

    % 显示结果
    fprintf('%9d | %.6f | %.6f | %.6f | %.6f\n', ...
            iter, x(1), x(2), f(x), norm(g));
end

fprintf('\n最终点: [%.6f, %.6f]\n', x(1), x(2));
fprintf('目标函数值: %.6f\n', f(x));
fprintf('梯度范数: %.6f\n', norm(g));
fprintf('迭代次数: %d\n', iter);
```

图 3: 题目二代码

```
>> t2
迭代次数 | x1       | x2       | f(x)     | ||grad||
-------------------------------------------------
         0 | 1.000000 | 1.000000 | 8.000000 | 12.083046
         1 | 0.458457 | -0.191395 | 0.093472 | 0.376475
         2 | 0.110896 | -0.046296 | 0.005469 | 0.091065
         3 | 0.000007 | -0.000003 | 0.000000 | 0.000006

最终点: [0.000007, -0.000003]
目标函数值: 0.000000
梯度范数: 0.000006
迭代次数: 3
```

图 4: 题目二结果

3 题目三

3.1 题目要求

使用共轭梯度法求解：

$$\min f(X) = 3x_1^4 + 2x_2^2 - 4x_1x_2, \quad X_0 = [2,-1]^T, \quad \varepsilon = 0.01$$

3.2 题目分析

目标函数：

$$f(X) = 3x_1^4 + 2x_2^2 - 4x_1x_2$$

计算梯度：

$$\nabla f(X) = \begin{bmatrix} 12x_1^3 - 4x_2 \\ 4x_2 - 4x_1 \end{bmatrix}$$

令梯度为零：

$$\begin{cases} 12x_1^3 - 4x_2 = 0 \\ 4x_2 - 4x_1 = 0 \end{cases}$$

第二方程：$x_2 = x_1$。代入第一方程：

$$12x_1^3 - 4x_1 = 4x_1(3x_1^2 - 1) = 0 \implies x_1 = 0 \text{ 或} x_1 = \pm\frac{1}{\sqrt{3}}$$

- $x_1 = 0, x_2 = 0, f(0,0) = 0$ - $x_1 = \frac{1}{\sqrt{3}}, x_2 = \frac{1}{\sqrt{3}}, f = 3\left(\frac{1}{\sqrt{3}}\right)^4 + 2\left(\frac{1}{\sqrt{3}}\right)^2 - 4\left(\frac{1}{\sqrt{3}}\right)\left(\frac{1}{\sqrt{3}}\right) = -\frac{1}{3}$ - $x_1 = -\frac{1}{\sqrt{3}}$，$x_2 = -\frac{1}{\sqrt{3}}$，$f = -\frac{1}{3}$

Hessian 矩阵：

$$\nabla^2 f(X) = \begin{bmatrix} 36x_1^2 & -4 \\ -4 & 4 \end{bmatrix}$$

- 在 $[0,0]$，Hessian 行列式 $0 \cdot 4 - (-4)^2 = -16 < 0$，为鞍点。- 在 $\left[\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right]$，Hessian 为 $\begin{bmatrix} 12 & -4 \\ -4 & 4 \end{bmatrix}$，行列式 $12 \cdot 4 - (-4)^2 = 32 > 0$，且 $12 > 0$，正定，为局部最小值。

全局最小值为 $f = -\frac{1}{3}$，在 $\left[\pm\frac{1}{\sqrt{3}}, \pm\frac{1}{\sqrt{3}}\right]$。

共轭梯度法通过共轭方向迭代，方向更新使用 Polak-Ribière 公式：

$$\beta_k = \max\left(0, \frac{\nabla f(X_{k+1})^T(\nabla f(X_{k+1}) - \nabla f(X_k))}{\|\nabla f(X_k)\|^2}\right)$$

算法从 $d_0 = -\nabla f(X_0)$ 开始，结合线搜索更新点。由于函数非二次型（含 x_1^4），收敛可能较慢。

3.3 代码与结果

```
% 目标函数
f = @(x) 3*x(1)^4 + 2*x(2)^2 - 4*x(1)*x(2);
% 梯度
grad_f = @(x) [12*x(1)^3 - 4*x(2); 4*x(2) - 4*x(1)];

% 初始点
x = [2; -1];
epsilon = 0.01;
max_iter = 1000;
iter = 0;

% 初始方向
d = -grad_f(x);

fprintf('迭代次数 | x1       | x2       | f(x)     | ||grad||\n');
fprintf('-------------------------------------------------\n');

while norm(grad_f(x)) > epsilon && iter < max_iter
    % 线搜索
    line_search = @(alpha) f(x + alpha*d);
    alpha = fminbnd(line_search, 0, 1);

    % 更新点
    x_new = x + alpha*d;

    % 计算beta (Polak-Ribière)
    g = grad_f(x);
    g_new = grad_f(x_new);
    beta = max(0, (g_new' * (g_new - g)) / (g' * g));

    % 更新方向
    d = -g_new + beta*d;
    x = x_new;
    iter = iter + 1;

    % 显示结果
    fprintf('%9d | %.6f | %.6f | %.6f | %.6f\n', ...
            iter, x(1), x(2), f(x), norm(grad_f(x)));
end

fprintf('\n最终点: [%.6f, %.6f]\n', x(1), x(2));
fprintf('目标函数值: %.6f\n', f(x));
fprintf('梯度范数: %.6f\n', norm(grad_f(x)));
fprintf('迭代次数: %d\n', iter);
```

图 5: 题目三代码

```
>> t3
迭代次数 | x1       | x2       | f(x)     | ||grad||
-------------------------------------------------
        1 | -0.615324 | -0.686161 | -0.317144 | 0.287913
        2 | -0.604890 | -0.597052 | -0.330030 | 0.269515
        3 | -0.577474 | -0.578254 | -0.333332 | 0.003778

最终点: [-0.577474, -0.578254]
目标函数值: -0.333332
梯度范数: 0.003778
迭代次数: 3
```

图 6: 题目三结果

4 题目四

4.1 题目要求

使用 MATLAB 编程求解以下优化问题:

$$\min f(X) = x_1^2 + x_2^2 - 3x_1 - x_1x_2$$

初始点 $X_0 = [0, 0]^T$, 精度要求 $\varepsilon = 0.1$。

4.2 题目分析

目标函数为：

$$f(X) = x_1^2 + x_2^2 - 3x_1 - x_1x_2$$

计算梯度：

$$\nabla f(X) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 - 3 - x_2 \\ 2x_2 - x_1 \end{bmatrix}$$

计算 Hessian 矩阵：

$$\nabla^2 f(X) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

检查 Hessian 正定性：- 主子式：$2 > 0$ - 行列式：$\det(\nabla^2 f) = 2 \cdot 2 - (-1) \cdot (-1) = 4 - 1 = 3 > 0$

Hessian 矩阵在任意点均为正定，表明函数为凸函数，存在唯一全局最小值。

令梯度为零以找到临界点：

$$\begin{cases} 2x_1 - 3 - x_2 = 0 \\ 2x_2 - x_1 = 0 \end{cases}$$

从第二方程：$x_1 = 2x_2$。代入第一方程：

$$2(2x_2) - 3 - x_2 = 4x_2 - 3 - x_2 = 3x_2 - 3 = 0 \implies x_2 = 1$$

然后：

$$x_1 = 2 \cdot 1 = 2$$

因此，临界点为 $X = [2, 1]^T$。代入函数验证：

$$f(2, 1) = 2^2 + 1^2 - 3 \cdot 2 - 2 \cdot 1 = 4 + 1 - 6 - 2 = -3$$

坐标轮换法每次优化一个变量：- 固定 x_2，优化 x_1：对 $f(x_1, x_2)$ 关于 x_1 求导并令其为零：

$$\frac{\partial f}{\partial x_1} = 2x_1 - 3 - x_2 = 0 \implies x_1 = \frac{3 + x_2}{2}$$

- 固定 x_1，优化 x_2：对 $f(x_1, x_2)$ 关于 x_2 求导并令其为零：

$$\frac{\partial f}{\partial x_2} = 2x_2 - x_1 = 0 \implies x_2 = \frac{x_1}{2}$$

迭代过程从 $X_0 = [0, 0]^T$ 开始，交替更新 x_1 和 x_2 直到满足 $\|\nabla f(X_k)\| < \varepsilon = 0.1$。

4.3 代码与结果

```
f = @(x) x(1)^2 + x(2)^2 - 3*x(1) - x(1)*x(2);

% Gradient function
gradf = @(x) [2*x(1) - 3 - x(2); 2*x(2) - x(1)];

% Initial point and tolerance
x = [0; 0]; % X0 = [0, 0]^T
epsilon = 0.1;
max_iter = 1000;
iter = 0;

% Store history for visualization
x_history = x';

while iter < max_iter
    x_old = x;

    % Update x1 (fix x2)
    x1_new = (3 + x_old(2)) / 2;
    x(1) = x1_new;

    % Update x2 (fix x1)
    x2_new = x(1) / 2;
    x(2) = x2_new;

    % Check convergence
    if norm(gradf(x)) < epsilon
        break;
    end

    x_history = [x_history; x'];
    iter = iter + 1;
end

% Display results
fprintf('Optimal point: [%.4f, %.4f]\n', x(1), x(2));
fprintf('Optimal value: %.4f\n', f(x));
fprintf('Number of iterations: %d\n', iter);
fprintf('Gradient norm: %.4f\n', norm(gradf(x)));
```

图 7: 题目四代码

```
>> t4
Optimal point: [1.9688, 0.9844]
Optimal value: -2.9993
Number of iterations: 2
Gradient norm: 0.0469
```

图 8: 题目四结果

5 题目五

5.1 题目要求

总结《优化理论与应用》课程前半学期所学知识点，分析方法优缺点，并结合个人学习经验讨论难点及优化方法在其他课程和科研中的应用。

5.2 知识点总结与分析

前半学期课程聚焦于无约束优化问题的核心方法，包括线搜索方法、最速下降算法、牛顿法、拟牛顿法、共轭梯度法、最小二乘法和坐标轮换法。以下是对这些方法的详细总结与分析，逻辑上从基础方法到高级方法展开，并探讨其理论基础、实现方式及优缺点。

线搜索方法 线搜索是许多优化算法的核心组成部分，用于在给定搜索方向上确定合适的步长 (α)，以确保目标函数值有效下降。常见准则包括：

- 精确线搜索：通过最小化单变量函数 ($\phi(\alpha) = f(X_k + \alpha d_k)$) 找到最优步长。精确线搜索计算复杂度高，但能保证最大下降。
- 不精确线搜索：
  - Armijo 准则：确保函数值下降满足 ($f(X_k + \alpha d_k) \leq f(X_k) + \rho\alpha\nabla f(X_k)^T d_k$)，其中 ($\rho \in (0, 0.5)$)（如 ($\rho = 0.2$)）。简单易实现，但可能导致步长偏小。
  - Wolfe 准则：在 Armijo 准则基础上，增加曲率条件 ($\nabla f(X_k + \alpha d_k)^T d_k \geq \sigma\nabla f(X_k)^T d_k$)，其中 ($\sigma \in (\rho, 1)$)（如 ($\sigma = 0.8$)），确保步长不过小，提升效率。

分析：线搜索方法平衡了计算成本与收敛速度。精确线搜索适合二次函数，但对非线性函数计算量大；不精确线搜索（如 Armijo-Wolfe）更实用，广泛应用于最速下降法和共轭梯度法。MATLAB 实现中，fminbnd 是常用的精确线搜索工具，而 Armijo 准则需手动实现步长回溯。

最速下降算法 最速下降法（Steepest Descent）是最简单的基于梯度的优化方法，搜索方向为负梯度 ($d_k = -\nabla f(X_k)$)，步长通过线搜索确定。

- 理论基础：负梯度方向保证局部下降，但收敛速度受 Hessian 矩阵条件数影响。对于病态问题（Hessian 特征值差异大），出现“之字形”迭代，收敛缓慢。
- 实现：从初始点 (X_0)，计算梯度，沿 (d_k) 进行线搜索，更新 ($X_{k+1} = X_k + \alpha_k d_k$)，直到 ($|\nabla f(X_k)| < \varepsilon$)。
- 优缺点：简单易实现，适合初学者理解优化原理，但对非二次函数或高条件数问题效率低。例如，作业 3 题目 1（$f(X) = 5x_1^2 + 2x_2^2 + x_1x_2$) 中，二次型函数收敛较快，但仍需多次迭代。

分析：最速下降法是入门级算法，为理解其他方法（如共轭梯度法）奠定基础，但实际应用中常被更高效方法替代。

牛顿法 牛顿法(Newton's Method)利用二阶信息,搜索方向为 $(d_k = -[\nabla^2 f(X_k)]^{-1}\nabla f(X_k))$,无需线搜索(步长通常为 1)。

- 理论基础:基于函数的二次泰勒展开,假设 Hessian 矩阵正定,牛顿法对二次函数可一次收敛。非二次函数需多次迭代,且 Hessian 非正定时可能失败。
- 实现:计算梯度和 Hessian 矩阵,求解线性系统 $(\nabla^2 f(X_k)d_k = -\nabla f(X_k))$,更新 $(X_{k+1} = X_k + d_k)$。MATLAB 中使用 $A\backslash b$ 求解线性系统。
- 优缺点:对二次函数(如作业 3 题目 2)收敛极快,但 Hessian 计算和求逆成本高,且对非凸函数可能发散。例如,作业 3 题目 4 的 Hessian 非正定,导致牛顿法不适用。

分析:牛顿法适合小规模凸问题,但对大规模或非凸问题需改进(如拟牛顿法)。

拟牛顿法 拟牛顿法(Quasi-Newton Methods)通过近似 Hessian 矩阵或其逆,避免直接计算二阶导数。

- 理论基础:使用梯度差更新 Hessian 近似(如 BFGS 或 DFP 公式)。BFGS 公式为:
$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}, \quad s_k = X_{k+1} - X_k, \quad y_k = \nabla f(X_{k+1}) - \nabla f(X_k)$$
确保近似矩阵正定,兼顾收敛速度与计算效率。
- 实现:初始化近似 Hessian(如单位矩阵),迭代更新 (B_k),求解 $(d_k = -B_k^{-1}\nabla f(X_k))$,结合线搜索更新点。
- 优缺点:避免 Hessian 计算,适合大规模问题,但存储和更新 (B_k) 仍需较多内存。相比牛顿法,对非凸问题更稳健。

分析:拟牛顿法是牛顿法与最速下降法的折中,广泛应用于机器学习模型优化。

共轭梯度法 共轭梯度法(Conjugate Gradient Method)专为二次函数设计,搜索方向为共轭方向集合,结合负梯度与前一步方向。

- 理论基础:对于二次函数 $(f(X) = \frac{1}{2}X^T AX - b^T X)$,共轭方向满足 $(d_i^T A d_j = 0 (i \neq j))$。使用 Polak-Ribière 公式计算 (β_k),更新方向 $(d_{k+1} = -\nabla f(X_{k+1}) + \beta_k d_k)$。
- 实现:从初始方向 $(d_0 = -\nabla f(X_0))$,通过线搜索确定步长,迭代更新方向和点。作业 3 题目 3(含 (x_1^4))非二次,需多次迭代。
- 优缺点:对二次函数收敛快(理论上 (n) 次迭代,(n) 为变量数),内存需求低,但非二次函数收敛较慢。

分析:共轭梯度法适合大规模稀疏问题,优于最速下降法,但在非线性问题中需结合线搜索优化。

最小二乘法 最小二乘法（Least Squares Method）解决超定方程组 ($Ax = b$) 的近似解，目标为最小化 ($|Ax - b|_2^2$)。

- 理论基础：推导正规方程 ($A^TAx = A^Tb$)，或使用 QR 分解求解。适用于作业 2 题目 4 的超定系统。
- 实现：MATLAB 中，$x = A\backslash b$ 自动使用 QR 分解，或显式计算 ($x = (A^TA)^{-1}A^Tb$)。
- 优缺点：简单高效，适合线性问题，但对病态矩阵敏感，需正则化处理。

分析：最小二乘法广泛应用于数据拟合，与优化算法结合可解决非线性最小二乘问题。

坐标轮换法 坐标轮换法（Coordinate Descent）每次优化一个变量，固定其他变量。

- 理论基础：对 ($f(X)$)，沿坐标轴 (e_i) 优化 ($f(x_1, \ldots, x_i, \ldots, x_n)$)。对于二次函数，可解析求解（如作业 3 题目 4）。
- 实现：迭代更新每个变量，检查梯度范数收敛。作业 3 题目 4 因 Hessian 非正定，函数无界，导致发散。
- 优缺点：实现简单，适合可分离问题，但对强耦合变量（如 (x_1x_2) 项）效率低，易发散。

分析：坐标轮换法适合高维稀疏问题，但需谨慎处理非凸或无界函数。

综合分析 这些方法形成了一个从简单到复杂的优化体系：

- 线搜索是基础，支撑最速下降法和共轭梯度法。
- 最速下降法简单但效率低，适合教学。
- 牛顿法和拟牛顿法利用二阶信息，效率高但计算成本不同。
- 共轭梯度法在二次函数中高效，适合大规模问题。
- 最小二乘法专注线性问题，应用广泛。
- 坐标轮换法简单但对函数性质敏感。

理论上，二次凸函数（如作业 3 题目 1、2）适合牛顿法或共轭梯度法，而非凸或无界函数（如题目 4）需特殊处理。MATLAB 实现中，精确线搜索（fminbnd）和 Hessian 求逆（$A\backslash b$) 是关键技术点。

5.3 个人角度的课程难点

学习优化方法时，遇到了以下具体难点，结合实际案例分析：

线搜索参数调优

- 难点：选择合适的线搜索参数（如 Armijo 的 (ρ)、Wolfe 的 (σ)) 困难，参数不当会导致迭代过慢或发散。
- 案例：在作业 3 题目 1（最速下降法）中，使用 Armijo 准则（$\rho = 0.2$) 实现线搜索时，初始步长 ($\alpha = 1$) 常导致函数值不满足 Armijo 条件，需多次减小步长（如 ($\alpha = \alpha/2$))。调试时发现，$\rho = 0.1$ 收敛更快，但作业要求固定 ($\rho = 0.2$)，增加了迭代次数。MATLAB 代码中，手动实现回溯线搜索时，循环逻辑易出错，例如忘记更新步长导致无限循环。
- 感受：参数调优需要反复试验，理论公式与实际效果有差距，调试耗时。

非凸函数的收敛性

- 难点：处理非凸或无界函数（如作业 3 题目 4）时，算法可能发散或陷入鞍点，难以判断问题来源。
- 案例：作业 3 题目 4（$f(X) = 2x_1^2 + x_2^2 - 3x_1x_2$) 中，坐标轮换法迭代 6000 次后出现 NaN，因为 Hessian 非正定，函数值沿某些方向趋向负无穷。调试时，检查 Hessian 行列式为 (-1)，确认函数无界，但最初未意识到题目无解，浪费时间尝试修改代码（如增加步长限制）。与最速下降法对比后，确认发散是函数性质而非代码错误。
- 感受：非凸问题需要更强的理论分析能力，MATLAB 调试无法直接揭示函数无界，需结合数学推导。

Hessian 矩阵的数值稳定性

- 难点：牛顿法和拟牛顿法中，Hessian 矩阵或其近似的计算和求逆可能导致数值不稳定。
- 案例：在实现牛顿法（作业 3 题目 2）时，Hessian 矩阵 ($\begin{bmatrix} 2 & 3 \\ 3 & 8 \end{bmatrix}$) 正定，理论上收敛快。但在调试早期版本代码时，误用符号运算（syms）计算 Hessian，导致运行速度慢。切换到数值计算后，矩阵求逆（inv 或 \）对小规模问题稳定，但在其他练习中遇到病态 Hessian 时，MATLAB 报“矩阵接近奇异”警告，需添加正则化。
- 感受：Hessian 的数值计算需要仔细验证矩阵条件数，初学者易忽略数值稳定性问题。

MATLAB 代码调试

- 难点：MATLAB 代码实现中，矩阵操作、循环逻辑和收敛条件易出错，调试复杂。

- 案例：在共轭梯度法（作业 3 题目 3）中，Polak-Ribière 公式的 (β_k) 计算中，分子可能为负，导致方向非下降。添加 max(0, beta) 后解决，但初期未发现此问题，迭代结果异常。另一次，梯度范数计算 (norm(grad$_f(x)$)) $MATLAB$

5.4 优化方法在其他课程和科研中的应用设想

优化方法在大学课程和科研中有广泛应用，以下针对大学生常见课程和实际场景提出具体设想：

机器学习课程：梯度下降与线搜索

- 课程：本科“机器学习”或“人工智能导论”。
- 应用：最速下降法和线搜索用于优化线性回归或逻辑回归的损失函数。例如，训练逻辑回归模型时，最小化交叉熵损失 ($L(w) = -\sum[y_i \log(\sigma(w^T x_i)) + (1-y_i)\log(1-\sigma(w^T x_i))]$)。通过 Armijo 线搜索调整步长，可加速收敛并避免过拟合。MATLAB 实现中，可用 fminbnd 优化步长，结合数据集（如 UCI 数据集）进行实验。
- 价值：学生可通过优化算法理解模型训练过程，掌握超参数调优。

控制系统课程：最小二乘法

- 课程：本科“自动控制原理”或“信号与系统”。
- 应用：最小二乘法用于系统辨识，估计控制系统的参数。例如，识别线性时不变系统的传递函数参数，解决超定方程 ($y = H\theta$)，其中 (H) 为输入矩阵，(θ) 为参数向量。MATLAB 的 lsqr 函数可高效求解，结合课程实验数据（如电机响应）优化控制器设计。
- 价值：帮助学生将优化理论应用于实际工程问题，提升控制精度。

数值分析课程：牛顿法与拟牛顿法

- 课程：本科“数值分析”或“计算方法”。
- 应用：牛顿法和拟牛顿法用于求解非线性方程组或优化问题。例如，求解 ($f(x) = x^3 - 2x + 1 = 0$)，牛顿法通过迭代 ($x_{k+1} = x_k - f'(x_k)^{-1} f(x_k)$) 快速收敛。拟牛顿法（如 BFGS）可优化更复杂的目标函数（如多变量多项式），在 MATLAB 中结合 fsolve 验证结果。
- 价值：强化学生对数值方法的理解，连接理论与计算实践。

数据科学课程：共轭梯度法

- 课程：本科“数据科学基础”或“大数据分析”。
- 应用：共轭梯度法用于大规模稀疏数据集的优化，如主成分分析（PCA）中的协方差矩阵优化。MATLAB 实现中，处理高维数据集（如图像特征）时，共轭梯度法比最速下降法更高效，适合内存受限场景。
- 价值：学生可将算法应用于真实数据集，提升数据处理能力。

科研应用：坐标轮换法在图像处理

- 场景：本科生参与导师的图像处理或机器学习科研项目。
- 应用：坐标轮换法用于稀疏优化问题，如图像去噪中的 Lasso 问题：$\min \frac{1}{2}|Ax-b|_2^2+\lambda|x|_1$。每次优化一个像素值，固定其他变量，适合高维稀疏信号恢复。MATLAB 中，结合开源图像数据集（如 MNIST）实现去噪算法。
- 价值：为学生提供科研实践机会，培养算法开发能力。

综合设想 这些方法在大学课程中帮助学生将理论转化为实践。例如，在机器学习课程实验中，学生可用最速下降法训练模型，比较不同线搜索策略；在控制系统课程中，最小二乘法优化控制器参数；在科研中，拟牛顿法和共轭梯度法处理高维数据。这些应用不仅加深对优化理论的理解，还为未来职业发展（如数据科学家、工程师）奠定基础。

6 自我感想与收获总结

通过本次《优化理论与应用》课程的学习和作业 3 的完成，我对无约束优化问题有了更深刻的理解。从最速下降法到拟牛顿法（BFGS），再到共轭梯度法和坐标轮换法，每一种方法的理论推导和 MATLAB 实现让我感受到优化算法的多样性与实用性。特别是通过题目分析，我学会了如何判断函数的凸性和最优解的存在性，例如题目 1 和题目 2 中的 Hessian 正定分析让我认识到凸函数的优化稳定性，而题目 3 和题目 4 的非凸与无界问题则让我意识到算法选择的复杂性。

在实践过程中，MATLAB 编程让我从理论走向应用，尤其是坐标轮换法的实现让我体会到算法设计的严谨性，同时也暴露了我对数值计算细节的不足，例如题目 4 中梯度范数收敛条件的调试让我认识到代码逻辑的重要性。此外，课程总结部分让我对优化方法在机器学习、控制系统等领域的应用有了更清晰的认识，例如梯度下降法在逻辑回归中的应用场景让我对未来的学习方向充满期待。

总的来说，这次作业不仅提升了我的数学分析能力和编程水平，也让我对优化理论的实际意义有了更深的感悟。我深刻体会到优化算法是连接理论与实践的桥梁，未来我

希望将这些知识应用到更复杂的科研问题中，例如图像处理或数据分析领域，进一步提升自己的综合能力。