一、 实验目的与要求

1. 熟练最小二乘法优化模型的意义和求解手段；
2. 掌握最小二乘法的正规方程，能实现代码对其求解；
3. 掌握最速梯度下降法求解无约束最小二乘法问题。

二、 问题

读取附件“MatrixA_b.mat”文件中的矩阵A和向量b。建立关于矩阵 $A \in R^{m\times n}$，向量 $b \in R^m$，未知向量 $x \in R^n$ 最小二乘优化模型：$\min\limits_{x} \|Ax-b\|_2^2$

1）通过最小二乘法的正规方程，求出优化模型的准确解；

2）利用梯度下降法迭代求出模型“近似解”，通过设置迭代停止条件，分析“近似解”与“准确解”之间的误差。

解决问题思路，模型建立、性能分析，存在问题等方面进行阐述；梯度下降法迭代求解，可以设置迭代次数或相邻迭代解之间“相对接近程度”，如 $\|x^k - x^{k+1}\|_2 / \|x^k\|_2$，作为迭代停止条件；代码不要放在报告里面，可以作为附件提交！

三、模型建立及求解

一、解决问题思路

本次实验实现过程的思维逻辑如图 1 所示，老师审批辛苦了💦！

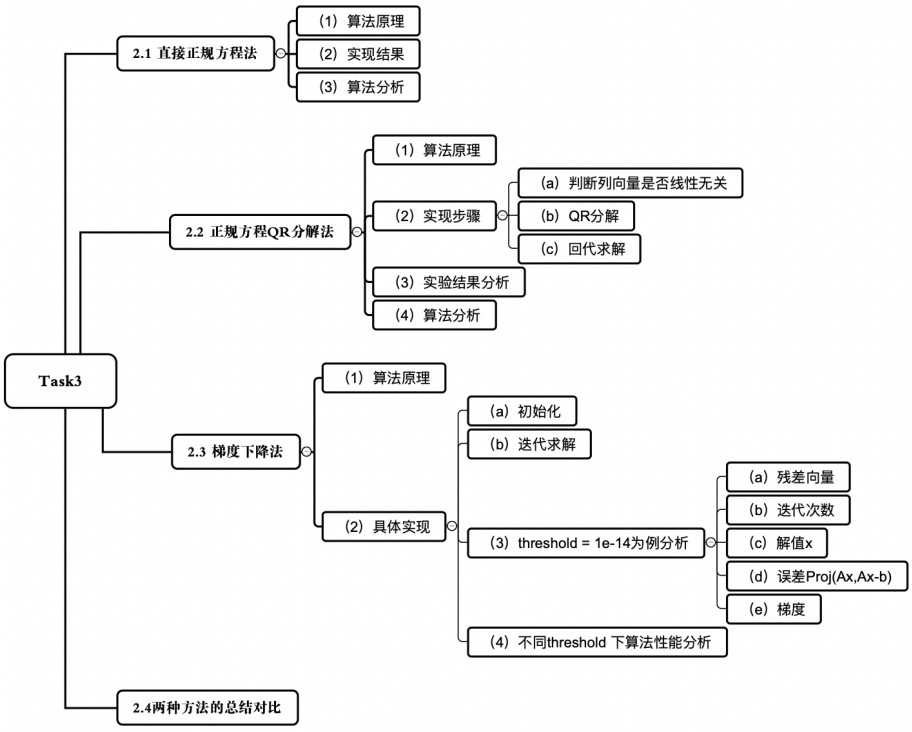

图 1 最小二乘法实验实现过程思维逻辑图

二、模型建立与分析

2.1 直接正规方程法

（1）算法原理

最小二乘法正规方程直接求解最小二乘法正规方程形式如下:

$$A^TAx = A^Tb$$

若要直接求解,则在等式两端同时左乘$(A^TA)^{-1}$

$$\hat{x} = (A^TA)^{-1}A^Tb$$

（2）实现结果

在 MATLAB 上直接编写代码（x1 = (A'*A)\(A'*b)）求解$\hat{x}$，结果如图 2 所示（准确解x是一个40×1的向量，由于这里列向量不方便展示，故笔者将$\hat{x}$向量进行转置展示）

```
>> x1'
ans =
列 1 至 14
 -0.4106   -0.4014   -0.2396    0.1844    0.0198    0.1678    0.0910    0.4264    0.1890   -0.4568    0.0710    0.2867    0.3330   -0.4083
列 15 至 28
 -0.1434   -0.3200   -0.3985    0.0116   -0.5994    0.6976    0.1366    0.4923    0.6429   -0.1005   -0.3836    0.3558    0.1383   -0.1622
列 29 至 40
  0.0608   -0.3770    0.3445    0.1276   -0.2797    0.0727    0.4069   -0.1634   -0.0721    0.1655    0.0888    0.1581
```

图 2 正规方程法求解的准确值x

（3）算法分析

当A列满秩时,直接求解正规方程,总的运算量大约为$mn^2 + \frac{1}{3}n^3 + O(n^2)$,其中大部分的运算量$(mn^2)$是用来计算$A^TA$(由于$A^TA$对称,因此只需计算其下三角部分)。

用正规方程法求解最小二乘问题,运算量最小,而且简单直观，但由于A^TA的条件数是A的条件数的平方,因此对于病态情形(即A的条件数比较大),不建议使用该方法。

2.2 正规方程 QR 分解法

（1）算法原理

要求$\min_x \| Ax - b \|_2^2$,可设$f(x) = \| Ax - b \|_2^2 = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} A_{ij}x_j - b_i\right)^2$,为了使目标函数最小,令最优解$\hat{x} = \arg\min f(x)$。可微函数$f(x)$的最优解$\hat{x}$满足条件:

$$\text{梯度}\nabla f(x) = 0,\text{即}\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_1} \end{bmatrix} = 2A^T(A\hat{x} - b) = 0$$

进而可以得到

$$\nabla f(x) = 2(A^TAx - A^Tb) = 0 \Rightarrow A^TAx = A^Tb$$

当A的列向量无关时,$x = (A^TA)^{-1}A^Tb$,此时对A做QR分解，则可以得到如下模型:

$$\begin{aligned}\hat{x} = (A^TA)^{-1}A^Tb &= ((QR)^T(QR))^{-1}(QR)^Tb \\ &= (R^TQ^TQR)^{-1}R^TQ^Tb \\ &= (R^TR)^{-1}R^TQ^Tb \\ &= R^{-1}Q^Tb\end{aligned}$$

对方程两侧左乘R后方程为:

$$R\hat{x} = Q^Tb$$

$Q^\top b$可以直接计算,而R为上三角矩阵,可用后向回代求出$\hat{x}$。因此正规方程求准确解分为三步:

（a）判断列向量是否线性无关

（b）QR 分解

（c）回代求解

（2）实现步骤

（a）判断列向量是否线性无关

本次实验求解正规方程为 QR 分解法，因此输入矩阵需为非奇异矩阵，即满足列向量线性无关。可以使用求秩法进行判断，本次实验所给矩阵 A 大小为 50*40，笔者使用 matlab 的 rank 函数求出 A 的秩并判断是否与 A 矩阵列的数量相等，若相等，则说明矩阵 A 列向量线性无关。

实验结果显示 rank=40（笔者在这里不再截图运行语句的结果），说明 A 的秩与矩阵 A 列的数量相等，满足列向量线性无关，故可以继续进行 QR 分解。

其实，判断列向量是否线性无关的方法有很多。比如也可以在 QR 分解计算过程中，加入一句判断，若发现 R 的主元中出现 0，说明矩阵列向量线性相关，则停止分解。

（b）QR 分解

Task2 中笔者已对 QR 分解算法进行了，此处不再赘述。在最小二乘法实现中笔者使用性能相对较好的 householder 方法对矩阵 A 进行 QR 分解，往后使用后向回代求解$\hat{x}$。

（c）回代求解

将Q^Tb记为d，使用后向回代求解x的过程如下：

$$\begin{aligned}x_n &= \frac{\mathrm{d}_n}{R_{nn}} \\ x_{n-1} &= \frac{\left(d_{n-1} - R_{n-1,n}x_n\right)}{R_{n-1,n-1}} \\ x_{n-2} &= \frac{\left(d_{n-2} - R_{n-2,n-1}x_{n-1} - R_{n-2,n}x_n\right)}{R_{n-2,n-2}} \\ &\vdots \\ x_1 &= \frac{(d_1 - R_{12}x_2 - R_{13}x_3 - \cdots - R_{1n}x_n)}{R_{11}}\end{aligned}$$

（3）实验结果分析

最后求得方程的准确解如图 3 所示，准确解x是一个40×1的向量。（这里列向量不方便展示，笔者将$\hat{x}$向量进行转置展示）

图 3 模型准确解x

因为最小二乘法的核心是让残差向量的二范数平方尽可能小，通过对准确解x做相应检验，残差向量$\min_x \| Ax-b \|_2^2 = 1.377808508765620$。

（4）算法分析

用QR分解来求最小二乘解的运算量大约为$2mn^2$(如果采用 Householder 变换的话,运算量大约为 $2mn^2-\frac{2}{3}n^3$。当$m \gg n$时,大约为正规方程的两倍.当$m=n$时,几乎相同。通常QR算法比较稳定,是求解最小二乘问题的首选方法,特别是当A条件数较大(病态)时。

但由于此方法需要使用 QR 分解，所以只适用于列向量无关的矩阵。若对列向量有关的矩阵求解会出现无效结果。正规方程法所要求的矩阵 A 列向量线性无关，在实际中几乎无法满足。所以不适合把此方法应用到实际问题中。另外，在实际求解过程中，一般无需获得准确解，所以可以使用梯度下降法求得近似解。

2.3 梯度下降法

（1）算法原理

设凸函数$f(x)$为:

$$f(x)=\frac{1}{2}\| Ax-b \|_2^2$$

设$x(k)$是第k步迭代,期望更新$x(k+1)$,满足:

$$f\left(x^{(k+1)}\right) < f\left(x^{(k)}\right)$$

通过梯度下降法迭代求解，迭代方程为:

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} A^T\left(Ax^{(k)}-b\right)$$

其中,$\alpha^{(k)}$为最优步长, 通过线性搜索估计：

$$\alpha^{(k)} = \arg\min_{\alpha} f\left(x^{(k)} - \alpha A^T\left(Ax^{(k)}-b\right)\right)$$

令$g(\alpha)=f\left(x^{(k)} - \alpha A^T\left(Ax^{(k)}-b\right)\right)$是关于$\alpha$的凸函数,则有:

$$\min_{\alpha} g(\alpha) \Rightarrow g'(\alpha)=0 \Rightarrow \alpha^{(k)} = \frac{\left\| A^T\left(Ax^{(k)}-b\right)\right\|_2^2}{\left\| AA^T\left(Ax^{(k)}-b\right)\right\|_2^2}$$

经过k次迭代或满足迭代终止条件后,用$x^{(k)}$作为模型的近似解。故求解思路总结如下：

凸函数$f(x)$,其梯度的反方向指向极小值点,如果x沿着该方向移动,x将更加接近极小值点。但如果x的步长过大,容易出现振荡,步长过小,迭代次数又会显著增加,所以需要一个最优步长。迭代x的目的是为了让函数$f(x)$减小,接近极小值。所以,假设梯度反方向上有一点x',使得$f(x)$在x'能取到这个方向上的极小值。最优步长即为这一点x'与当前x沿着梯度反方向的距离。当$f(x)$的下降不明显时,停止迭代。

（2）具体实现

（a）初始化

在迭代运算开始前，需先初始化$x^{(0)}$以及设置迭代终止条件 threshold。初始化对迭代步数有一定的影响，但由于缺乏其他条件，不能找到最佳的初始位置，且选择零向量的迭代步数相较而言不算很大，故初始化$x^{(0)}$为零向量。同时笔者使用相邻迭代之间的残差向量值变化程度(记为$delta = \frac{\|x^k - x^{k+1}\|_2}{\|x^k\|_2}$)作为迭代停止条件。

（b）迭代求解

首先根据公式求出$\alpha^{(k)}$：

$$\alpha^{(k)} = \frac{\|A^T(Ax^{(k)} - b)\|_2^2}{\|AA^T(Ax^{(k)} - b)\|_2^2}$$

然后根据公式计算迭代后的$x^{(k+1)}$：

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)}A^T(Ax^{(k)} - b)$$

根据计算出的$x^{(k+1)}$,得到两次迭代间的$delta$值:

$$delta = \left| \frac{\|Ax^{(k+1)} - b\|_2^2 - \|Ax^{(k)} - b\|_2^2}{\|Ax^{(k)} - b\|_2^2} \right|$$

若$delta$值小于设置的迭代终止条件 threshold 则退出循环,并输出结果。

（3）以$threshold = 1e-14$作为迭代终止条件为例进行分析

（a）残差向量

使用梯度下降法求出的是模型的近似解，此时的残差向量$\| Ax - b \|_2^2$随着x的迭代而接近极小值（如图 4 所示）。

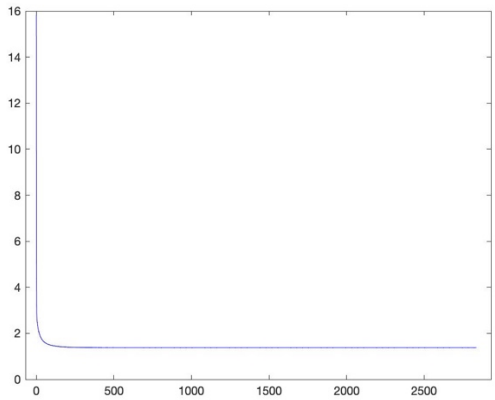

图 4：残差向量随x的迭代接近极小值

当x的变化率$delta = \frac{\|x^k - x^{k+1}\|_2}{\|x^k\|_2}$小于 $threshold = 1e-14$时停止迭代，此时迭代次数2523 次，残差向量$\| Ax - b \|_2^2$ = 1.377808508767179，与使用正规方程求得准确解 x 的残差向量值相差极小：

$$1.377808508767179(梯度) - 1.377808508765620(正规) = 1.5590e-12$$

（b）迭代次数

同样的条件下，若将迭代次数限制为 500 次，得到的残差向量与使用正规方程求得准确解 x 的残差向量值相差将进一步扩大：

$$1.378228665804850(梯度) - 1.377808508765620(正规) = 4.2016e-04$$

（c）解值x

在这个迭代停止条件下，对每次迭代得到 x 的结果进行可视化，如图 5 所示，可以看到在迭代约 500 次后，算法达到基本收敛：

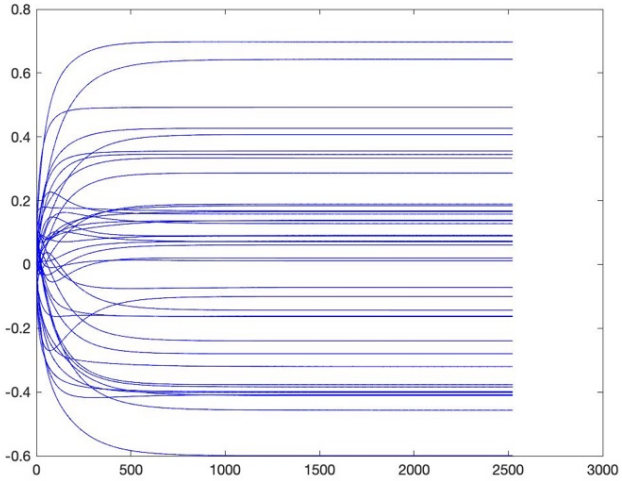

图 5 x的 40 个分量迭代变化

（d）误差$Proj(Ax, Ax-b)$

当$f(x) = \| Ax - b \|_2^2$为凸函数,且其取得极小值时,向量$Ax-b$与range (A)正交,而Ax为$range(A)$内一元素,故$Ax-b$与Ax正交,即向量$Ax-b$在向量Ax上的投影$Proj(Ax, Ax-b)$应当为零。所以笔者使用$Proj(Ax, Ax-b)$ 来衡量梯度下降算法的误差。从图 X 可以看到误差$Proj(Ax, Ax-b)$随着x的迭代而一直在震荡收敛，即使在迭代 2001-2500 次也是在震荡收敛的过程，最终迭代 2523 次时，$Proj(Ax, Ax-b)$ = 1.3406e-07 ，以接近 0，再一次说明梯度下降的有效性。但是从图 6 又可以看到，残差向量随x的迭代是单调递减的过程，最后收敛于极小值，而$Proj(Ax, Ax-b)$确是一个震荡收敛的过程，这其中的原因还有待笔者进行研究探讨。

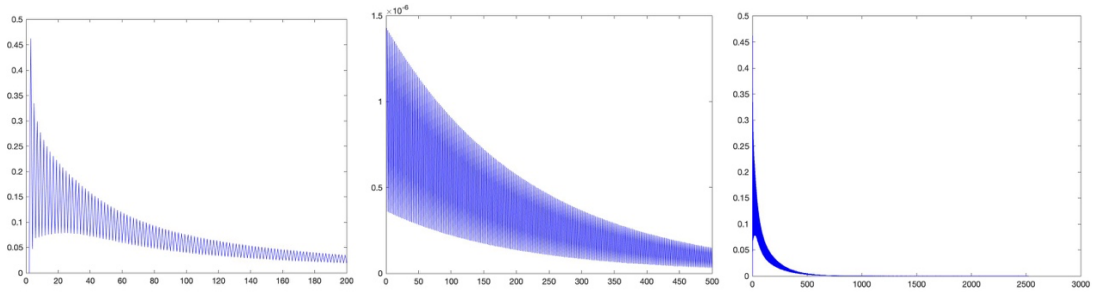

图 6 $Proj(Ax, Ax-b)$大小

（从左至右，分别为第 1-500 次迭代时、第 2001-2500 次迭代时、第 1-2523 次迭代时）

（e）梯度

最优步长的选择可以让x沿着梯度的反方向移动，从而让 $f(x)$下降最大的步长，即x的下一轮移动必然与前一轮移动的方向正交（如图 7 所示）。

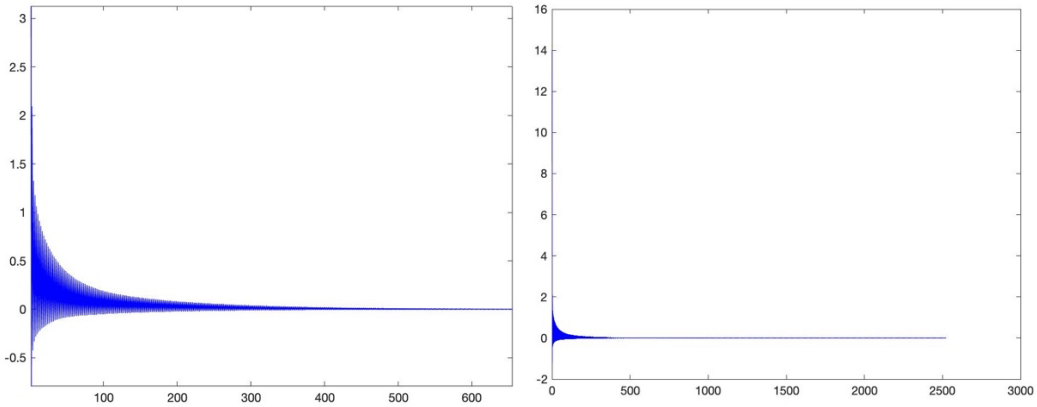

图 7 相邻梯度正交（左：放大图；右：全局图）

（4）不同$threshold$ 下算法性能分析

笔者接着对比分析不同$threshold$ 下的算法性能，实验结果如表 1 所示，随着$threshold$的值从 1.00E-10 到 1.00E-16，算法的迭代次数不断增加，其残差向量也不断减小，梯度下降求解的残差向量与使用正规方程求得准确解 x 的残差向量值之差也不断缩小，$Proj(Ax, Ax-b)$也不断减少。说明随着$threshold$精度的提高，迭代次数增加，算法性能提高。

但当$threshold$的值大于 1.00E-16 后，算法的性能不再提升，此时的迭代次数为 2837，残差向量值为 1.37780850876571，梯度和正规残差向量之差为 9.48E-14，$Proj(Ax, Ax-b)$为 3.29E-08，算法性能已接近准确解。

表 1：不同$threshold$ 下算法性能分析

threshold	迭代次数	残差向量	梯度和正规残差向量之差	误差 Proj(Ax,Ax-b)
1.00E-10	1493	1.37780852400640	1.52E-08	1.58E-05
1.00E-11	1749	1.37780851030252	1.54E-09	4.60E-06
1.00E-12	2007	1.37780850891969	1.54E-10	1.39E-06
1.00E-13	2264	1.37780850878125	1.56E-11	1.00E-07
1.00E-14	2523	1.37780850876717	1.56E-12	1.34E-07

1.00E-15	2745	1.37780850876583	2.16E-13	4.96E-08
1.00E-16	2837	1.37780850876571	9.48E-14	3.29E-08
1.00E-17	2837	1.37780850876571	9.48E-14	3.29E-08
1.00E-18	2837	1.37780850876571	9.48E-14	3.29E-08

2.4 两种方法的总结对比

通过表 2 最小二乘法与梯度下降法的比较，可以知道：当矩阵 A 的列向量线性相关时，梯度下降法是更优的选择；当矩阵 A 的列向量线性无关时，因为最小二乘法相对于梯度下降法的复杂度更低，最小二乘法是更优的选择。

需要注意的是，在本次实验中，笔者设计最小二乘法时，首先需要对矩阵 A 是否是列线性无关进行判断，在判断该条件时需要损失一定的算力和时间，若矩阵 A 的维度较大，即使判断其可逆采用最小二乘法进行求解，这个过程所用的时间也可能导致大于直接使用梯度下降法所用的时间。

表 2 最小二乘法与梯度下降法的比较

方法	复杂度	适用范围	备注
最小二乘法	$O(2mn^2)$	只能适用于 A 列向量线性无关	A 为 m*n 的矩阵
梯度下降法	$O(2mn^2I)$	可适用于矩阵 A 的任意情况	A 为 m*n 的矩阵， I 为梯度迭代次数

四、小结（可含个人心得体会）

通过本次实验，笔者深刻理解并掌握了最小二乘优化问题的多种解法的原理和细节，通过在 MATLAB 上将其实现并进行对比。总的来说，笔者也更喜欢梯度下降法，因为其不仅普适性更强，而且通过实验发现，只要设置的精度足够，梯度下降法不断迭代得到的结果和解正规方程得到的最优解几乎相等，说明梯度下降法准确性之高。

本次实验还用到了实验二的 QR 分解的算法，学以致用，环环相扣，笔者更加感觉到了学习的成就感和算法的魅力。