数值分析第一次上机练习报告

——线性方程组解法

力 1 周懿 2021013053

一、 问题的描述

设 $H_n = [h_{ij}] \in \mathbb{R}^{n\times n}$ 是 Hilbert 矩阵，即

$$h_{ij} = \frac{1}{i+j-1}$$

对 n=10,11,...,15 恒成立。

(a) 取 $x = \begin{pmatrix} 1 \\ ... \\ 1 \end{pmatrix} \in \mathbb{R}^n$，令 $b_n = H_n x$. 再用 Gauss 消去法和 Cholesky 分解方法来求解 $H_n y = b_n$，看看误差有多大.

(b) 使用正则化方法改善 (a) 中的结果.

(c) 用共轭梯度法和 GMRES 方法求解 $H_n y = b_n$，并与前面的直接方法作比较.

二、 方法描述

(a) **Gauss 消元法和 Cholesky 分解法**

我们取 $H_n = [\frac{1}{i+j-1}]$。对于 Gauss 消元法，此处因为 Hilbert 矩阵主对角元素均非零，无需使用列主元 Gauss 消去法。先对 H_n 做消元操作，过程如下：

for k=1,2,...,n-1,

$$l_{ik} = \frac{h_{ik}^{(k)}}{h_{kk}^{(k)}}, \qquad i = k+1, ..., n \tag{1}$$

$$h_{ij}^{(k+1)} = \begin{cases} h_{ij}^{(k)}, & i = 1, 2, ..., k; \quad j = 1, 2, ..., n \\ 0, & i = k+1, ..., n; \quad j = 1, 2, ..., k \\ h_{ij}^{(k)} - l_{ij} h_{kj}^{(k)}, & i = k+1, ..., n; \quad j = k+1..., n \end{cases} \tag{2}$$

$$b_i^{(k+1)} = \begin{cases} b_i^{(k)}, & i = 1, ..., k \\ b_i^{(k)} - l_{ik} b_k^{(k)}, & i = k+1, ..., n \end{cases} \tag{3}$$

再对 H_n 做回代操作，过程如下：

$$\begin{cases} x_n = \frac{b_n^{(n)}}{h_{nn}}, \\ x_i = \frac{1}{h_{ii}^{(i)}}(b_i^{(i)} - \sum_{j=i+1}^{n} H_{ij}^{(i)} x_j), \quad i = n-1, n-2, ..., 1 \end{cases} \tag{4}$$

我们根据以上的算法编写程序：对于每一轮消元，我们都根据式(1)给出消元矩阵 L 第 k 列的元素，再由式(2)(3)给出消元后的矩阵 $H_n^{(k)}$ 和向量 $b_n^{(k)}$；消元过程结束后根据式(4)获得最终解 X_n。

对于 Cholesky 分解法，Hilbert 矩阵是一个对称正定阵，所以它可以被分解为 $H_n = LL^T$，其中 L 是对角元素为正的下三角阵。我们也可以使用改进之后的平方根法求解，此处矩阵阶数较低，计算量的差异不明显，因此仍然采用原始的 Cholesky 分解法。我们先对矩阵 H_n 进行 Cholesky 分解：

$$\textbf{for} \quad \textbf{k=1,2,...,n-1,}$$
$$\begin{cases} l_{jj} = \sqrt{h_{jj} - \sum_{k=1}^{j-1} l_{jk}^2} \\ l_{ij} = \frac{h_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}}{l_{jj}}, \qquad i = j+1, ..., n \end{cases} \tag{5}$$

我们根据式(5)编写程序获取分解之后的矩阵 L，然后分别求解 $L\mathbf{y} = \mathbf{b_n}$ 和 $L^T\mathbf{x} = \mathbf{y}$:

$$y_i = \frac{b_i - \sum_{k=1}^{i-1} l_{ik} y_k}{l_{ii}}, \qquad i = 1, 2, ..., n \tag{6}$$

$$x_i = \frac{y_i - \sum_{k=i+1}^{n} l_{ki} x_k}{l_{ii}}, \quad i = n, n-1, ..., 1 \tag{7}$$

我们根据以上算法编写程序，根据式(6)解得中间解 $\mathbf{y_n}$，再根据式(7)获得最终解 $\mathbf{x_n}$。

理论上利用这两种直接法获取的解是真实解，精度应该较高。但实际上，Hilbert 矩阵的条件数 $cond(H) >> 1$，随着阶数 n 逐渐增加可达 10^5 以上，病态程度极高。由于病态矩阵对于微小扰动十分敏感，而计算机又存在舍入误差的问题，对于 Hilbert 矩阵这类典型的病态矩阵，使用直接法求解就会因为计算机浮点运算带来的扰动而产生很大的误差。因而，使用直接法求解病态方程组通常需要对矩阵进行预处理，从而降低矩阵条件数。所以我们将在直接法求解的基础上引入 Tikhonov 正则化。

(b)**Tikhonov 正则化**

为了降低 Hilbert 矩阵的条件数，我们对矩阵 H_n 使用 Tikhonov 正则化：

$$(\alpha \mathbf{I} + H_n^* H_n)\mathbf{x}_\alpha = H_n^* \mathbf{b} \tag{8}$$

其中 α 是正则化因子，H_n^* 是希尔伯特矩阵 H_n 的共轭转置矩阵，$\mathbf{x}_\alpha$ 是 $\mathbf{x}$ 的一个近似解。我们根据式(8)编写程序，先获得矩阵 H_n 的共轭转置矩阵 H_n^*，再根据式(8)求解 $\mathbf{x}_\alpha$。

(c) **共轭梯度法和 GMRES 法**

对于共轭梯度法，这里我们使用未简化的共轭梯度法。为了保持梯度下降法的数值稳定性，我们希望每次搜索找到的是所有搜索方向组成线性空间的最小值点。设初始解为 $\mathbf{0}$，每次搜索方向为 $\mathbf{b}^{(k)}$，则第 $k+1$ 步时，有

$$\mathbf{x}^{(k+1)} = \alpha_0 \mathbf{p}^{(0)} + \alpha_1 \mathbf{p}^{(1)} + ... + \alpha_k \mathbf{p}^{(k)}$$

为了简便，我们令 $\mathbf{x} = \mathbf{y} + \alpha\mathbf{p}^{(k)}$，$y \in span\{\mathbf{p}^{(0)}, ...\mathbf{p}^{(k-1)}\}$。于是

$$\mathcal{F}(\mathbf{x}) = \mathcal{F}(\mathbf{y}) + \alpha(A\mathbf{y}, \mathbf{p}^{(k)}) - \alpha(\mathbf{b}, \mathbf{p}^{(k)}) + \frac{\alpha^2}{2}(A\mathbf{p}^{(k)}, \mathbf{p}^{(k)})$$

理论证明，为了使 $\mathcal{F}$ 达到最小，我们有

$$\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)} \tag{9}$$

$$\alpha_k = \frac{(\mathbf{r}^{(k)}, \mathbf{p}^{(k)})}{(A\mathbf{p}^{(k)}, \mathbf{p}^{(k)})} \tag{10}$$

$$\beta_k = -\frac{(\mathbf{r}^{(k+1)}, \mathbf{p}^{(k)})}{(A\mathbf{p}^{(k)}, \mathbf{p}^{(k)})} \tag{11}$$

基于上述理论，我们通过编程实现以下步骤：

1. 令 $k = 0$，$\mathbf{x}^{(0)} = \mathbf{0}$;
2. 根据式(9)(10)计算出 $\mathbf{r}^{(0)}, \alpha_k$，进而得到

   $$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{p}^{(k)}$$

   若 $||\alpha_k\mathbf{p}^{(k)}|| < \epsilon$ 则停止迭代，否则继续；
3. 根据式(9)计算新残差 $\mathbf{r}^{(k+1)}$，以及根据式(10)(11)更新 α_k, β_k 并且给出新的搜索方向 $\mathbf{p}^{(k+1)} = \mathbf{r}^{(k+1)} + \beta_k\mathbf{p}^{(k)}$
4. $k = k + 1$，回到第 2 步。

只要初始误差 ϵ 设置得当，我们就可以获得非常精确的解。

对于 GMRES 法，我们需要找到子空间 K_m, L_m，使得在子空间 K_m 中可以找到近似解 $\mathbf{z_m}$，使得残差 $\mathbf{r_0} - A\mathbf{z_m}$ 与子空间 L_m 中所有向量正交，即

$$(\mathbf{r_0} - A\mathbf{z_m}, \mathbf{w}) = 0, \forall\mathbf{w} \in L_m \tag{12}$$

取一组基底 $W_m = (\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_m})$，则我们有

$$(W_m^T A V_m)\mathbf{y_m} = W_m^T\mathbf{r_0} \tag{13}$$

根据 Cayley-Hamiltom 定理，我们取 $K_m = span\{\mathbf{r_0}, A\mathbf{r_0}, ..., A^{m-1}\mathbf{r_0}\}$，也就是 Krylov 子空间，为了避免恶性中断，取 $L_m = AK_m$。从 $m = 1$ 开始，理论上当 $m = n$ 时，我们就将获得精确解。基于上述理论，我们可以通过编程实现以下步骤：

1. 取 $\mathbf{x_0} = \mathbf{0}$，计算 $\mathbf{r_0} = \mathbf{b} - A\mathbf{x_0}$，$\beta = ||\mathbf{r_0}||, \mathbf{v_1} = \frac{\mathbf{r_0}}{||\mathbf{r_0}||}$, 重复步骤 2-4 直至收敛;
2. 使用 Arnoldi 过程求出 $\{\mathbf{v_i}\}_{i=1}^m$ 和 $\widetilde{H}_m$;
3. 求解最小二乘问题 $min_{\mathbf{y_m}\in\mathbb{R}^m}||\beta\mathbf{e_1} - \widetilde{H}_m\mathbf{y_m}||$ 得到 $\mathbf{y_m}$，这里为了求解该问题，我们对方程组使用 Givens 旋转将 Hessenberg 矩阵进行 QR 分解转为上三角矩阵;
4. 计算 $\mathbf{x_k} = \mathbf{x_0} + V_m\mathbf{y_m}$.

三、 方案设计

我们通过编写 MATLAB 程序来进行线性方程组的求解。在程序文件中：

1. Hilbert.m 是主程序文件;

2. gauss.m 是使用 Gauss 消元法解线性方程组的子程序文件，给它任意一个主对角元素非零的矩阵 A 和向量 $\mathbf{b}$ 及其阶数 n，此程序将返回解向量 X_1；

3. cholesky.m 是使用 Cholesky 分解法解线性方程组的子程序文件，给它任意一个正定的矩阵 A 和向量 $\mathbf{b}$ 及其阶数 n，此程序将返回解向量 X_2；

4. tikhonov.m 是使用 Tikhonov 正则化优化矩阵的子程序文件，给它任意一个矩阵 A 和向量 $\mathbf{b}$ 及其阶数 n，此程序将返回优化之后的矩阵 T 和向量 $\mathbf{b_T}$;

5. tikhonov_gauss.m 是使用 Tikhonov 正则化改良 Gauss 消元法求解结果的子程序文件，给它任意一个矩阵 A 和向量 $\mathbf{b}$ 及其阶数 n，此程序此程序先调用 tikhonov.m 优化矩阵，最后将返回优化之后的解向量 X_3；

6. tikhonov_chol.m 是使用 Tikhonov 正则化改良 Cholesky 分解法求解结果的子程序文件，给它任意一个矩阵 A 和向量 $\mathbf{b}$ 及其阶数 n，此程序先调用 tikhonov.m 优化矩阵，最后将返回优化之后的解向量 X_4；

7. cg.m 是使用共轭梯度法求解线性方程组的子程序文件，给它任意一个矩阵 A 和向量 $\mathbf{b}$ 及其阶数 n，以及停止收敛时的误差限度 ϵ，此程序将返回解向量 X_5 和迭代需要的步数；

8. GMRES.m 是使用 GMRES 法求解线性方程组的子程序文件，给它任意一个矩阵 A 和向量 $\mathbf{b}$ 及其阶数 n，此程序将返回解向量 X_6。

我们在主程序中计算残差向量各元素的均方误差（MSE），作为相对误差（Err）。

四、 计算结果及其分析

图 1是我们根据程序计算结果得到的数据，其中分别给出了矩阵阶数 n、使用对应求解方法（Gauss 消去法、Cholesky 分解法、Tikhonov 正则化、共轭梯度法、GMRES 法）计算出来的解 X，以及相对误差（此处使用解向量各元素的均方误差作为相对误差）。

n	Gauss	Err	Cholesky	Err	Tikhonov-Gauss	Err	Tikhonov-Cholesky	Err	CG	Err	GMRES	Err
	1.0000		1.0000		1.0038		1.0038		1.0000		1.0000	
	1.0000		1.0000		0.9831		0.9831		1.0000		1.0000	
	1.0000		1.0000		0.9930		0.9930		1.0000		1.0000	
	1.0000		1.0000		1.0121		1.0121		0.9999		0.9999	
n=10	0.9999	5.0074E-08	0.9999	4.6156E-08	1.0226	3.7237E-04	1.0226	3.7237E-04	1.0001	6.1085E-09	1.0001	2.5683E-09
	1.0003		1.0003		1.0225		1.0225		1.0000		1.0000	
	0.9996		0.9996		1.0140		1.0140		0.9999		0.9999	
	1.0004		1.0004		0.9993		0.9993		1.0000		1.0000	
	0.9998		0.9998		0.9807		0.9807		1.0001		1.0001	
	1.0000		1.0000		0.9595		0.9595		1.0000		1.0000	
	1.0000		1.0000		1.0031		1.0031		1.0000		1.0000	
	1.0000		1.0000		0.9893		0.9893		1.0000		1.0000	
	1.0000		1.0000		0.9874		0.9874		1.0000		1.0000	
	1.0002		1.0002		1.0044		1.0044		0.9999		0.9999	
	0.9991		0.9989		1.0180		1.0180		1.0001		1.0001	
n=11	1.0033	2.1538E-05	1.0038	3.0129E-05	1.0229	3.3861E-04	1.0229	3.3861E-04	1.0000	1.2224E-08	1.0000	6.1084E-09
	0.9929		0.9916		1.0197		1.0197		0.9999		0.9999	
	1.0097		1.0115		1.0102		1.0102		0.9999		0.9999	
	0.9919		0.9904		0.9961		0.9961		1.0001		1.0001	
	1.0037		1.0044		0.9790		0.9790		1.0001		1.0001	
	0.9993		0.9991		0.9597		0.9597		0.9999		0.9999	
	1.0000		1.0000		1.0024		1.0024		1.0000		1.0000	
	1.0000		1.0000		0.9953		0.9953		1.0000		1.0000	
	0.9999		0.9998		0.9832		0.9832		1.0001		1.0001	
	1.0011		1.0021		0.9976		0.9976		0.9998		0.9998	
	0.9919		0.9844		1.0129		1.0129		1.0001		1.0001	
n=12	1.0354	9.1122E-03	1.0688	3.6272E-02	1.0214	3.3330E-04	1.0214	3.3330E-04	1.0001	3.9385E-07	1.0001	1.2224E-08
	0.9023		0.8080		1.0225		1.0225		0.9999		0.9999	
	1.1754		1.3479		1.0172		1.0172		0.9999		0.9999	
	0.7958		0.5917		1.0070		1.0070		1.0000		1.0000	
	1.1487		1.2994		0.9933		0.9933		1.0001		1.0001	
	0.9385		0.8754		0.9772		0.9772		1.0002		1.0002	
	1.0110		1.0225		0.9594		0.9594		0.9999		0.9999	
	1.0000		1.0000		1.0016		1.0016		1.0000		1.0000	
	1.0000		0.9998		1.0009		1.0009		1.0000		1.0000	
	1.0006		1.0069		0.9803		0.9803		1.0001		1.0001	
	0.9898		0.8843		0.9916		0.9916		0.9998		0.9998	
	1.0920		2.0504		1.0077		1.0077		1.0002		1.0002	
	0.5010		-4.7497		1.0188		1.0188		1.0002		1.0002	
n=13	2.7409	9.3971E+00	21.2157	1.3071E+03	1.0232	3.4716E-04	1.0232	3.4716E-04	0.9999	2.1725E-08	0.9999	2.1725E-08
	-3.0361		-46.1909		1.0214		1.0214		0.9998		0.9998	
	7.2839		74.9208		1.0146		1.0146		0.9999		0.9999	
	-5.4935		-75.8021		1.0040		1.0040		1.0000		1.0000	
	5.2709		51.7601		0.9906		0.9906		1.0002		1.0002	
	-0.6182		-18.3172		0.9753		0.9753		1.0002		1.0002	
	1.2688		4.2220		0.9586		0.9586		0.9998		0.9998	
	1.0000		1.0000		1.0009		1.0009		1.0000		1.0000	
	1.0000		1.0000		1.0060		1.0060		1.0000		1.0000	
	1.0003		1.1000		0.9783		0.9783		1.0001		1.0001	
	0.9962		-0.5000		0.9865		0.9865		0.9997		0.9997	
	1.0199		15.0000		1.0026		1.0026		1.0002		1.0002	
	0.9817		-76.1000		1.0156		1.0156		1.0002		1.0002	
n=14	0.6752	2.0814E+01	272.6000	2.2090E+05	1.0226	3.7247E-04	1.0226	3.7247E-04	1.0000	3.5405E-08	1.0000	3.5406E-08
	2.9229		-634.3000		1.0237		1.0237		0.9998		0.9998	
	-4.4981		997.8000		1.0198		1.0198		0.9998		0.9998	
	10.5160		-1035.9000		1.0119		1.0119		0.9999		0.9999	
	-9.4282		686.8000		1.0011		1.0011		1.0001		1.0001	
	8.0948		-259.8000		0.9880		0.9880		1.0003		1.0003	
	-1.7410		44.3000		0.9733		0.9733		1.0002		1.0002	
	1.4602		1.1000		0.9575		0.9575		0.9997		0.9997	
	1.0000		1.0000		1.0001		1.0001		1.0000		1.0000	
	1.0000		1.0000		1.0106		1.0106		1.0000		1.0000	
	1.0000		1.6000		0.9772		0.9772		1.0000		1.0002	
	0.9986		-8.7000		0.9822		0.9822		1.0001		0.9996	
	1.0257		89.4000		0.9979		0.9979		0.9999		1.0001	
	0.7819		-484.4000		1.0121		1.0121		1.0000		1.0003	
	2.0668		1712.4000		1.0211		1.0211		1.0001		1.0001	
n=15	-2.2790	1.1371E+01	-4003.8000	8.2027E+06	1.0247	4.0362E-04	1.0247	4.0362E-04	1.0000	1.8722E-09	0.9998	5.4017E-08
	7.5324		6287.4000		1.0232		1.0232		1.0000		0.9997	
	-7.3550		-6541.4000		1.0178		1.0178		0.9999		0.9998	
	7.3807		4328.9000		1.0092		1.0092		1.0000		1.0000	
	-1.1290		-1643.7000		0.9981		0.9981		1.0000		1.0002	
	0.4257		271.4000		0.9853		0.9853		1.0001		1.0003	
	1.7333		3.4000		0.9712		0.9712		1.0001		1.0001	
	0.8180		0.5000		0.9563		0.9563		0.9999		0.9996	

图 1: 计算结果

(a) **使用 Gauss 消去法和 Cholesky 分解的求解结果**

计算结果如图 1所示表格的第 1-5 列所示。由表中数据可以看出，当 n 逐渐增加时，相对误差大致成递增趋势，并且增长速度很快（特别是 Cholesky 分解法）。但是通过观察表中数据可以发现，无论使用何种直接求解法，无论矩阵阶数多少，获得的解的前几位元素在保留四位小数的情况下与精确解一致，这可能与矩阵的前几行经历的消元步骤较少，因此误差并未放大有关。

(b) **使用正则化改善后的求解结果**

计算结果如图 1所示表格的第 6-9 列所示。由表中数据可以看出，获得的解误差一直稳定在 10^-4 量级。但是在矩阵阶数较低时，误差反而比直接法要高，这可能是因为阶数较低时，矩阵的病态程度也相应较低，直接法获取的解本身就非常精确。如果使用正则化，因为多出了正则化因子，获得的解虽然仍是一个较好的近似，但并不是精确解。

(c) **使用共轭梯度法和 GMRES 法的求解结果**

计算结果如图 1所示表格的第 10-13 列所示。此处对共轭梯度法，我们取 $\epsilon = 9.3 * 10^{-10}$，由表中数据可以看出如果 ϵ 设置的较低的话，最后会得到更加精确的解，但是迭代的步数可能会增加；我们通过设置合理的误差限度 ϵ，就可以在尽量少的迭代步数下获取一个较为精确的解。但是进一步的实验发现，虽然 Hilbert 矩阵的病态程度与矩阵阶数正相关，但是在本实验中，如果我们使用更大范围的 n 进行实验，会发现同一精度下迭代的步数并不都和矩阵阶数正相关，关于这一现象出现的原因还有待进一步研究。

为了能够更加清晰的看出不同的求解方法的误差随线性方程组阶数的变化，我们给出不同求解方法求解结果的误差随阶数的变化曲线，如图 2所示。由图 2可以看出，Gauss 消元法和 Cholesky 分解法在阶数较低 ($n <= 11$) 时误差较小，从 n=12 开始误差显著增大。Hilbert 矩阵是一个典型的病态矩阵，且其病态程度与阶数正相关，说明这两种解法对于病态程度较高的矩阵不能得到一个较为精确的解，特别是 Cholesky 分解法，误差更是达到 10^6 量级。经过 Tikhonov 正则化之后，由图 2可以看出，解的精确度稳定性显著提高，说明 Hilbert 矩阵在经过 Tikhonov 正则化之后得到的新矩阵病态程度显著降低。同时我们还可以发现，对两种直接法分别使用 Tikhonov 正则化之后，求解得到的结果非常相近，可见这两种直接法在方程病态程度低的时候并没有显著的差别，但是对比病态条件下的情况我们可以发现，相比于 Gauss 消去法，Cholesky 分解法虽然节约了步数，但是在分解过程中误差被进一步放大了。

而当我们使用共轭梯度法和 GMRES 法时，由图 2可以看出解的误差显著低于 Tikhonov 正则化改良之后的结果，且始终稳定在极低的水平。这是因为迭代法在误差未达到设置值时就会继续迭代，直到误差达到设定值以下才停止迭代。但是值得注意的是，病态矩阵还是应当尽量避免，因为病态程度较高时，可能出现迭代法收敛非常慢的情况。

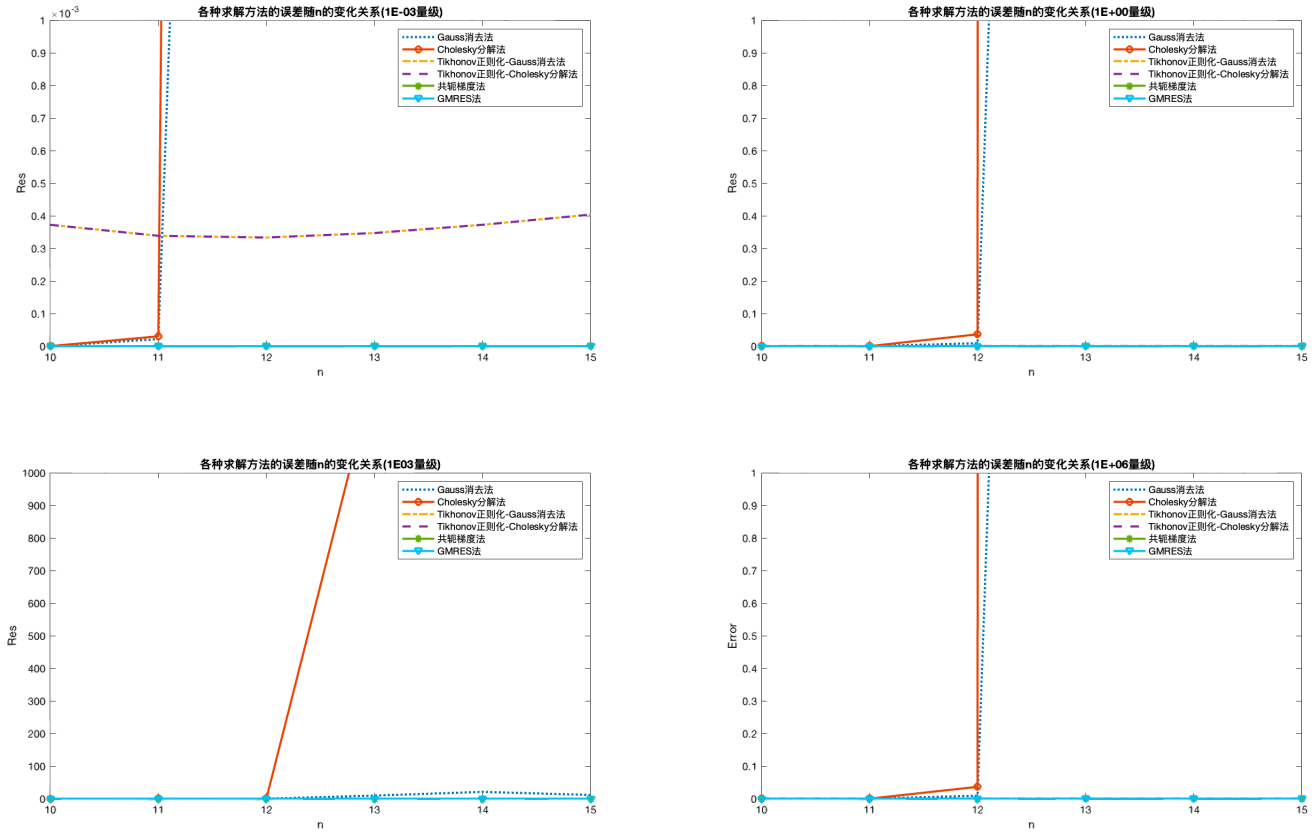

图 2: 不同求解方法的误差随线性方程组阶数的变化关系

五、 结论

线性方程组的解法中最基本的方法是 Gauss 消元法，为了节省计算量而衍生出 Cholesky 分解法，这两种均为直接法，理论上可以求出矩阵的精确解。但实际上病态矩阵对于微小扰动非常敏感，由于计算机的浮点运算带来的舍入误差对矩阵有扰动作用，对病态方程组使用直接法将会导致很大的误差。我们通过实验的实际计算，发现在矩阵病态程度较高时，使用直接法确实会导致很大的误差，对于一些条件数增长速度很快的病态矩阵（如 Hilbert 矩阵），在阶数很低的情况下就已经无法再获得精确的解。因此对于病态矩阵，我们需要先用正则化方法对矩阵进行预处理，降低矩阵条件数，或者是利用迭代法使得解收敛到误差范围以内才停止迭代。在本实验中，我们分别使用 Tikhonov 正则化、共轭梯度法和 GMRES 法分别进行求解，均取得了较为精确和稳定的结果。