数学实验：第三次作业

计算机系 计 73 2017011620 李家昊

2020 年 3 月 23 日

1 实验目的

- 学会用 MATLAB 软件数值求解线性代数方程组，对迭代法的收敛性和解的稳定性作初步分析；
- 通过实例学习用线性代数方程组解决简化的实际问题。

2 问题求解

2.1 Chap5-Ex1 误差（计算题）

2.1.1 算法设计

由题意，需要解下列方程，

$$\boldsymbol{A}_1\boldsymbol{x}_1 = \boldsymbol{b}_1, \quad \boldsymbol{A}_2\boldsymbol{x}_2 = \boldsymbol{b}_2 \tag{1}$$

其中 $\boldsymbol{A}$ 为 Vandermonde 矩阵，

$$\boldsymbol{A}_1 = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{pmatrix}, \quad x_k = 1 + 0.1k, \quad k = 0, 1, \cdots, n-1 \tag{2}$$

$\boldsymbol{A}_2$ 为 Hilbert 矩阵，

$$\boldsymbol{A}_2 = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \ddots & \ddots & \vdots \\ \frac{1}{3} & \ddots & \ddots & \ddots & \frac{1}{2n-3} \\ \vdots & \ddots & \ddots & \frac{1}{2n-3} & \frac{1}{2n-2} \\ \frac{1}{n} & \cdots & \frac{1}{2n-3} & \frac{1}{2n-2} & \frac{1}{2n-1} \end{pmatrix} \tag{3}$$

表 1: 不同 n 值下矩阵 $\boldsymbol{A}_1, \boldsymbol{A}_2$ 的条件数

n	5	7	9
$Cond(\boldsymbol{A}_1)$	3.5740×10^5	8.7385×10^7	2.2739×10^{10}
$Cond(\boldsymbol{A}_2)$	4.7661×10^5	4.7537×10^8	4.9315×10^{11}

并且 $\boldsymbol{b}_1$ 为 $\boldsymbol{A}_1$ 的行和，$\boldsymbol{b}_2$ 为 $\boldsymbol{A}_2$ 的行和。

显然，方程 (1) 的解为，

$$\boldsymbol{x}_1 = \boldsymbol{x}_2 = (1, 1, \cdots, 1)^T \tag{4}$$

对于 $\boldsymbol{x}$ 的数值解，用 Matlab 的左除命令求解即可。对于误差分析，可根据如下结论求出相对误差限。

当 $\boldsymbol{A}$ 受扰动而 $\boldsymbol{b}$ 不变时，若 $||\boldsymbol{A}^{-1}|| \cdot ||\delta\boldsymbol{A}|| < 1$，则 $\boldsymbol{x}$ 的相对误差限为，

$$\frac{||\delta\boldsymbol{x}||}{||\boldsymbol{x}||} \leq \frac{Cond(\boldsymbol{A})}{1 - Cond(\boldsymbol{A}) \cdot \dfrac{||\delta\boldsymbol{A}||}{||\boldsymbol{A}||}} \cdot \frac{||\delta\boldsymbol{A}||}{||\boldsymbol{A}||} \tag{5}$$

当 $\boldsymbol{b}$ 受扰动而 $\boldsymbol{A}$ 不变时，$\boldsymbol{x}$ 的相对误差限为，

$$\frac{||\delta\boldsymbol{x}||}{||\boldsymbol{x}||} \leq Cond(\boldsymbol{A}) \cdot \frac{||\delta\boldsymbol{b}||}{||\boldsymbol{b}||} \tag{6}$$

2.1.2 Matlab 程序

请参见附录4.1。

2.1.3 计算结果

方程求解 当 $n = 5$ 时，求解方程 (1) 得，

$$\boldsymbol{x}_1 = \boldsymbol{x}_2 = (1.0000, 1.0000, 1.0000, 1.0000, 1.0000)^T \tag{7}$$

与预期结果相符。

条件数计算 当 $n = 5, 7, 9$ 时，计算 $\boldsymbol{A}_1$ 和 $\boldsymbol{A}_2$ 的条件数如表 1所示，可以看出，随着 n 的增大，$\boldsymbol{A}_1, \boldsymbol{A}_2$ 的条件数随之增大，病态严重。此外，在相同的 n 下，$\boldsymbol{A}_2$ 的条件数比 $\boldsymbol{A}_1$ 更大，因此 $\boldsymbol{A}_2$ 的病态更严重。

扰动 $\boldsymbol{A}_1, \boldsymbol{A}_2$ 后的计算结果及误差 令 $n = 5, 7, 9$，$\boldsymbol{b}_1$ 和 $\boldsymbol{b}_2$ 不变，在 $\boldsymbol{A}_1(n, n)$ 和 $\boldsymbol{A}_2(n, n)$ 上分别加上扰动 $\varepsilon = 10^{-10}, 10^{-8}, 10^{-6}$，限于篇幅，详细的计算结果请参见附录5.1中的表 10及表 11。经扰动得到的解记作 $\tilde{\boldsymbol{x}}$，则相对误差为 $\dfrac{||\tilde{\boldsymbol{x}} - \boldsymbol{x}||}{||\boldsymbol{x}||}$，$\boldsymbol{x}_1, \boldsymbol{x}_2$ 的相对误差计算结果分别如表 2，表 3所示，其中相对误差限的计算省略了 $||\boldsymbol{A}^{-1}|| \cdot ||\delta\boldsymbol{A}|| \geq 1$ 的情形。

表 2: 扰动 $\boldsymbol{A}_1(n,n)$ 后 $\boldsymbol{x}_1$ 的实际相对误差及相对误差限

$\varepsilon\backslash n$	5		7		9	
	Actual	Limit	Actual	Limit	Actual	Limit
10^{-10}	0.0000	0.0000	0.0000	0.0003	0.0000	0.0134
10^{-8}	0.0000	0.0004	0.0003	0.0300	0.0033	/
10^{-6}	0.0021	0.0444	0.0319	/	0.3295	/

表 3: 扰动 $\boldsymbol{A}_2(n,n)$ 后 $\boldsymbol{x}_2$ 的实际相对误差及相对误差限

$\varepsilon\backslash n$	5		7		9	
	Actual	Limit	Actual	Limit	Actual	Limit
10^{-10}	0.0000	0.0000	0.0021	0.0295	0.7280	/
10^{-8}	0.0005	0.0031	0.1893	/	3.1999	/
10^{-6}	0.0490	0.4371	1.7383	/	3.3124	/

扰动 $\boldsymbol{b}_1, \boldsymbol{b}_2$ 后的计算结果及误差 令 $n = 5, 7, 9$，$\boldsymbol{A}_1$ 和 $\boldsymbol{A}_2$ 不变，在 $\boldsymbol{b}_1(n)$ 和 $\boldsymbol{b}_2(n)$ 上分别加上扰动 $\varepsilon = 10^{-10}, 10^{-8}, 10^{-6}$，限于篇幅，详细的计算结果请参见附录5.1中的表 12及表 13。$\boldsymbol{x}_1, \boldsymbol{x}_2$ 的相对误差计算结果分别如表 4，表 5所示。

2.1.4 结果分析

根据表 2，表 3，表 4，表 5中的实际相对误差及相对误差限的计算结果，可以验证相对误差不超过其相对误差限。总体来看，实际相对误差的值主要集中在相对误差限的 1% 到 10% 的区间内，因此，利用相对误差限来估算大致的实际相对误差是可行的。

Vandermonde 矩阵和 Hilbert 矩阵都是病态程度非常严重的矩阵，若矩阵或右端项稍有扰动，方程的解就会千差万别，例如，当规模 $n = 9$，扰动大小 $\varepsilon = 10^{-6}$ 时，解的相对误差竟达到将近 10^4 级别，这样的结果在实际应用中是没

表 4: 扰动 $\boldsymbol{b}_1(n)$ 后 $\boldsymbol{x}_1$ 的实际相对误差及相对误差限

$\varepsilon\backslash n$	5		7		9	
	Actual	Limit	Actual	Limit	Actual	Limit
10^{-10}	0.0000	0.0000	0.0000	0.0001	0.0000	0.0068
10^{-8}	0.0000	0.0002	0.0003	0.0137	0.0033	0.6808
10^{-6}	0.0021	0.0200	0.0319	1.3690	0.3303	68.0786

表 5: 扰动 $\boldsymbol{b}_2(n)$ 后 $\boldsymbol{x}_2$ 的实际相对误差及相对误差限

$\varepsilon\backslash n$	5		7		9	
	Actual	Limit	Actual	Limit	Actual	Limit
10^{-10}	0.0000	0.0000	0.0021	0.0124	0.9330	11.1158
10^{-8}	0.0005	0.0015	0.2103	1.2389	93.3044	1111.5847
10^{-6}	0.0512	0.1519	21.0324	123.8889	9330.4374	111158.4664

有任何意义的。此外，当扰动 ε 越大，方程的解的相对误差限就越大，其实际相对误差在一般情况下也越大。当 n 越大，则两个矩阵的条件数越大，病态越严重，方程越不稳定，对扰动越敏感，从而越难获得精确的解。在相同的 n 值下，Hilbert 矩阵的条件数比 Vandermonde 矩阵的条件数更大，病态更严重，在相同的扰动下，解的相对误差更大。

2.1.5 结论

矩阵的条件数越大，病态越严重，方程越不稳定，对扰动越敏感，从而越难获得精确的解；Vandermonde 矩阵和 Hilbert 矩阵都是病态矩阵，在相同规模下，Hilbert 矩阵的病态程度比 Vandermonde 矩阵更严重；在实际应用中，例如在神经网络的矩阵运算中，应当尽可能避免病态矩阵所带来的不稳定性，从而增强系统的鲁棒性。

2.2 Chap5-Ex3 迭代法（计算题）

2.2.1 算法设计

题目需要用 Jacobi 迭代法和 Gauss-Seidel 迭代法求解方程 $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$，其中 $\boldsymbol{A}\in\mathbb{R}^{n\times n}$，这里 $n=20$，定义为，

$$\boldsymbol{A}=\begin{pmatrix} 3 & -1/2 & -1/4 & & \\ -1/2 & 3 & \ddots & \ddots & \\ -1/4 & \ddots & \ddots & \ddots & -1/4 \\ & \ddots & \ddots & 3 & -1/2 \\ & & -1/4 & -1/2 & 3 \end{pmatrix} \tag{8}$$

首先将 $\boldsymbol{A}$ 分解为 $\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U}$ 的形式，其中 $\boldsymbol{D}=\mathrm{diag}(a_{11},a_{22},\cdots,a_{nn})$，

$$\boldsymbol{L}=-\begin{pmatrix} 0 & & & \\ a_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix},\quad \boldsymbol{U}=-\begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & 0 & \ddots & \vdots \\ & & \ddots & a_{n-1,n} \\ & & & 0 \end{pmatrix} \tag{9}$$

表 6: 不同的初始向量 $\boldsymbol{x}^{(0)}$ 和右端项 $\boldsymbol{b}$ 的选取

i	1	2	3
$\boldsymbol{x}_i^{(0)}$	$(1,1,...,1)^T$	$(1,2,...,n)^T$	$(-1,1,...,(-1)^n)^T$
$\boldsymbol{b}_i$	$(1,1,...,1)^T$	$(1,2,...,n)^T$	$(-1,1,...,(-1)^n)^T$

采用迭代法求解时，记迭代总数为 m，在第 k 次迭代时，有状态方程，

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{B}\boldsymbol{x}^{(k)} + \boldsymbol{f}, \quad k = 0, 1, 2, \cdots, m-1 \tag{10}$$

采用 Jacobi 迭代法时，

$$\boldsymbol{B} = \boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U}), \quad \boldsymbol{f} = \boldsymbol{D}^{-1}\boldsymbol{b} \tag{11}$$

采用 Gauss-Seidel 迭代法时，

$$\boldsymbol{B} = (\boldsymbol{D} - \boldsymbol{L})^{-1}\boldsymbol{U}, \quad \boldsymbol{f} = (\boldsymbol{D} - \boldsymbol{L})^{-1}\boldsymbol{b} \tag{12}$$

如此，就可以利用迭代法求解线性方程组 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ 了。对于迭代矩阵 $\boldsymbol{B}$ 的谱半径，可通过 `max(abs(eig(B)))` 命令计算。

2.2.2 Matlab 程序

请参见附录4.2。

2.2.3 计算结果

这里设置迭代误差限为 $\varepsilon = 10^{-5}$，当迭代误差 $||\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}||_\infty < \varepsilon$ 时，算法结束，输出此时的迭代次数。

不同的初始向量和右端项 选取不同的初始向量 $\boldsymbol{x}^{(0)}$ 和右端项 $\boldsymbol{b}$ 如表 6所示。其中包括了常量向量，顺序递增向量，以及周期向量。在不同的初始向量和右端项下，采用 Jacobi 迭代法和 Gauss-Seidel 迭代法求解的迭代次数如表 7所示。

成倍增长 $\boldsymbol{A}$ 的主对角线 取定初始向量和右端项为，

$$\boldsymbol{x}^{(0)} = (1,1,...,1)^T, \quad \boldsymbol{b} = (1,1,...,1)^T \tag{13}$$

将 $\boldsymbol{A}$ 的主对角线元素增长为原来的 k 倍，用 Jacobi 迭代法计算，迭代次数及迭代矩阵的谱半径如表 8所示。

表 7: 在不同的初始向量 $\boldsymbol{x}^{(0)}$ 和右端项 $\boldsymbol{b}$ 下，给定迭代误差为 10^{-5} 时，Jacobi (J) 和 Gauss-Seidel (G-S) 迭代法的迭代次数。

$\boldsymbol{b}\backslash\boldsymbol{x}^{(0)}$	$\boldsymbol{x}_1^{(0)}$		$\boldsymbol{x}_2^{(0)}$		$\boldsymbol{x}_3^{(0)}$	
	J	G-S	J	G-S	J	G-S
$\boldsymbol{b}_1$	15	11	20	14	16	11
$\boldsymbol{b}_2$	20	14	19	13	20	14
$\boldsymbol{b}_3$	17	12	20	14	10	9

表 8: 将 $\boldsymbol{A}$ 的主对角线元素增长为原来的 k 倍后，给定迭代误差 10^{-5} 时，Jacobi 迭代法的迭代次数 m 及迭代矩阵 $\boldsymbol{B}$ 的谱半径 $\rho(\boldsymbol{B})$。

k	1	2	4	8	16	32	64	128	256	512
m	15	9	7	6	5	4	4	4	3	3
$\rho(\boldsymbol{B})$	0.489	0.245	0.122	0.061	0.031	0.015	0.008	0.004	0.002	0.001

2.2.4 结果分析

在不同的初始向量和右端项下，Jacobi 和 Gauss-Seidel 迭代法产生的迭代向量序列均收敛，相比于 Jacobi 迭代法，Gauss-Seidel 迭代法的迭代次数更少，收敛速度更快。这是由迭代矩阵 $\boldsymbol{B}$ 的谱半径决定的，通过计算得出，Jacobi 的迭代矩阵 $\boldsymbol{B}$ 的谱半径 $\rho(\boldsymbol{B}) = 0.4893$，Gauss-Seidel 的迭代矩阵 $\boldsymbol{B}$ 的谱半径 $\rho(\boldsymbol{B}) = 0.2523$，它们都小于 1，因此两种迭代法均收敛，由于 Gauss-Seidel 的迭代矩阵 $\boldsymbol{B}$ 的谱半径小于 Jacobi 的迭代矩阵，因此 Gauss-Seidel 迭代法的收敛速度更快。从另一个角度来看，由于 $\boldsymbol{A}$ 是严格对角占优的，因此也可以判断出两种迭代法均收敛。

随着 $\boldsymbol{A}$ 主对角线元素的成倍增长，Jacobi 迭代法的迭代次数越来越少，收敛速度越来越快。这是因为当 $\boldsymbol{A}$ 的主对角线元素增长时，$\boldsymbol{A}$ 的对角线更占优，从表 8可以看出，其对应的迭代矩阵 $\boldsymbol{B}$ 的谱半径更小，因此收敛速度更快。

2.2.5 结论

若线性方程组的系数矩阵是严格对角占优的，则 Jacobi 和 Gauss-Seidel 两种迭代法的迭代序列均收敛，一般来说，Gauss-Seidel 迭代法比 Jacobi 迭代法的收敛速度更快。

2.3 Chap5-Ex9 种群（计算题）

2.3.1 算法设计

根据题意，种群年龄为 $k=1,2,...,n$，当年年龄 k 的种群数量记作 x_k，繁殖率记作 b_k，自然存活率记作 s_k，收获量记作 h_k，来年年龄 k 的种群数量为 $\tilde{x}_k$，稳定种群内，有 $\tilde{x}_k = x_k$。

矩阵模型 已知 b_k, s_k, h_k 时，在稳定种群内 $\tilde{x}_k = x_k$，因此有，

$$x_1 = \sum_{k=1}^{n} b_k x_k, \quad x_{k+1} = s_k x_k - h_k, \quad k = 1,2,...,n-1 \tag{14}$$

记 $\boldsymbol{x} = (x_1, x_2, ..., x_n)^T$，$\boldsymbol{h} = (0, h_1, h_2, ..., h_{n-1})^T$，记 Leslie 矩阵为，

$$\boldsymbol{L} = \begin{pmatrix} b_1 & b_2 & \cdots & \cdots & b_n \\ s_1 & 0 & \cdots & \cdots & 0 \\ 0 & s_2 & \ddots & & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & s_{n-1} & 0 \end{pmatrix} \tag{15}$$

则可将方程 (14) 表示为矩阵形式，

$$\boldsymbol{x} = \boldsymbol{L}\boldsymbol{x} - \boldsymbol{h} \tag{16}$$

化简得，

$$(\boldsymbol{L} - \boldsymbol{I})\boldsymbol{x} = \boldsymbol{h} \tag{17}$$

即为稳定种群数量的矩阵模型。

数值求解 利用 Matlab 自带的左除法，即可求出方程 (17) 的数值解。

2.3.2 Matlab 程序

请参见附录4.3。

2.3.3 计算结果

由题目数据，可确定 Leslie 矩阵，

$$\boldsymbol{L} = \begin{pmatrix} 0 & 0 & 5 & 3 & 0 \\ 0.4 & 0 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0.4 & 0 \end{pmatrix} \tag{18}$$

表 9: 各年龄种群数量计算结果的绝对误差

绝对误差 ($\times 10^{-12}$)	e_1	e_2	e_3	e_4	e_5
结果19	-1.023	0.455	0.227	0.000	-0.028
结果20	-1.023	0.455	0.000	-0.114	0.000
结果21	-1.023	0.455	0.000	-0.114	0.000
结果22	0.000	-1.819	0.000	0.000	0.000

当 $\boldsymbol{h} = (0, 500, 400, 200, 100)^T$ 时，计算得出，

$$\boldsymbol{x} = (8481, 2892, 1335, 601, 141)^T \tag{19}$$

当 $\boldsymbol{h} = (0, 500, 500, 500, 500)^T$ 时，计算得出，

$$\boldsymbol{x} = (10981, 3892, 1835, 601, -259)^T \tag{20}$$

然而，种群数量 x_k 不可能为负数。产生负数的原因可能是收获量过大，供不应求，打破了种群的平衡状态。因此，为了能达到给定的收获量，需要采取相应的措施来维持生态平衡。

一种可能的措施是通过改善种群的生活条件，提高种群的存活率。考虑到 x_5 出现负数，应当尽可能提高 x_4 的存活率 s_4 从而增加 x_5 的数量，如果将 s_4 提高到 0.9，则计算得出，

$$\boldsymbol{x} = (10981, 3892, 1835, 601, 41)^T \tag{21}$$

达到了种群数量的平衡。

另一种可能的措施是通过增加种群规模，调节种群的繁殖率，从而提高种群的稳定性。如果将 b_3 调节为 3，则计算得出，

$$\boldsymbol{x} = (35132, 13553, 7632, 4079, 1132)^T \tag{22}$$

也达到了种群数量的平衡，此时种群规模更大，平衡性更强。

当然，两种方式可结合使用，使种群更稳定，满足收获量的需求。

2.3.4 结果分析

误差分析 记种群数量的计算值为 $\boldsymbol{x}^*$，收获量的真实值为 $\boldsymbol{h}$，则绝对误差为 $\boldsymbol{e} = (\boldsymbol{L} - \boldsymbol{I})\boldsymbol{x}^* - \boldsymbol{h}$，通过计算得出各结果的绝对误差，如表 9所示，其中 e_i 为 $\boldsymbol{e}$ 的第 i 个分量。可以看出，所有计算结果的绝对误差均控制在 10^{-11} 以内，结果有效。

稳定性分析 通过计算得出，由方程 (18) 确定的矩阵 $\boldsymbol{L}-\boldsymbol{I}$ 的条件数为 87.19，病态程度一般，模型的稳定性一般。

2.3.5 结论

若要求种群内年龄为 $1\sim5$ 的收获量分别为 500,400,200,100,100，则稳定种群内年龄为 $1\sim5$ 的个体数量分别为 8481,2892,1335,601,141。要使种群各年龄收获量均为 500，则可以通过提高存活率，调节繁殖率以达到目的。

3 收获与建议

在本次实验中，我通过使用 Matlab，掌握了数值求解线性代数方程组的方法，对迭代法的收敛性和解的稳定性有了更深的理解，用线性代数方程组解决了简化的实际问题，在解决实际问题的过程中，我对数学方法的原理和应用有了更深刻的理解。

希望助教能对每次的实验进行详细的解答，希望老师在未来的课堂上介绍更多数学应用的前沿知识。

4 附录：Matlab 程序代码

4.1 Chap5-Ex1

```
perturb_b = 1;

for n = [5,7,9]
    for eps = [1e-10, 1e-8, 1e-6]
        A1 = fliplr(vander(1 + 0.1 * (0:n-1)));
        A2 = hilb(n);
        b1 = sum(A1, 2);
        b2 = sum(A2, 2);
        % back up
        org_A1 = A1;
        org_A2 = A2;
        org_b1 = b1;
        org_b2 = b2;
        % perturbation
        if perturb_b
            b1(n) = b1(n) + eps;
            b2(n) = b2(n) + eps;
        else
            A1(n,n) = A1(n,n) + eps;
```

```
            A2(n,n) = A2(n,n) + eps;
        end
        % solve
        x1 = A1 \ b1;
        x2 = A2 \ b2;
        % relative error
        x = ones(n, 1);
        err1 = norm(x1 - x) / norm(x);
        err2 = norm(x2 - x) / norm(x);
        if perturb_b
            err_lim1 = cond(org_A1) * norm(b1 - org_b1) / norm(org_b1);
            err_lim2 = cond(org_A2) * norm(b2 - org_b2) / norm(org_b2);
        else
            err_A1 = norm(A1 - org_A1) / norm(A1);
            err_lim1 = cond(org_A1) / (1 - cond(org_A1) * err_A1) * err_A1;
            err_A2 = norm(A2 - org_A2) / norm(A2);
            err_lim2 = cond(org_A2) / (1 - cond(org_A2) * err_A2) * err_A2;
        end
        % output
        fprintf('n: %d, cond(A1): %g, cond(A2): %g\n', n, cond(A1), cond(A2));
        fprintf('n: %d, eps: %g, err1: %.4f/%.4f, err2: %.4f/%.4f\n', ...
            n, eps, err1, err_lim1, err2, err_lim2);
        x1'
        x2'
    end
end
```

4.2 Chap5-Ex3

```
n = 20;
A = 3*eye(n) - diag(ones(n-1,1),1)/2 - diag(ones(n-1,1), -1)/2 - ...
    diag(ones(n-2, 1), 2)/4 - diag(ones(n-2, 1), -2)/4;

b = ones(n,1);
x0 = ones(n,1);
iter_err = 1e-5;

% ground truth
```

```
x_gt = A \ b;

% decomposition of A
D = diag(diag(A));
L = -tril(A, -1);
U = -triu(A, 1);

% Jacobi
B = D \ (L + U);
f = D \ b;
[x, cnt] = solve_iter(B, f, x0, iter_err);
abs_err = max(abs(x - x_gt));
rho = max(abs(eig(B)));
fprintf('Jacobi rho: %.3f, iter: %d, abs err: %f\n', rho, cnt, abs_err);

% Gauss-Seidel
B = (D - L) \ U;
f = (D - L) \ b;
[x, cnt] = solve_iter(B, f, x0, iter_err);
abs_err = max(abs(x - x_gt));
rho = max(abs(eig(B)));
fprintf('Gauss-Seidel rho: %f, iter: %d, abs err: %f\n', rho, cnt, abs_err);

function [x, cnt] = solve_iter(B, f, x0, err)
    cnt = 0;
    x = x0;
    while 1
        x_old = x;
        x = B * x + f;
        cnt = cnt + 1;
        if max(abs(x - x_old)) < err
            break
        end
    end
end
```

4.3 Chap5-Ex9

```
b = [0 0 5 3 0];
s = [0.4 0.6 0.6 0.4];
h = [0 500 400 200 100]';
n = length(b);
```

```

L = [b; diag(s) zeros(n-1, 1)];
I = eye(n);
x = (L - I) \ h
err = (L - I) * x - h
cond(L - I)
```

5 附录：详细计算结果

5.1 Chap5-Ex1

表 10: 在 $\boldsymbol{A}_1(n,n)$ 处加扰动后的 $\boldsymbol{x}_1$ 求解结果

(n,ε)	$\boldsymbol{x}_1^T$
$(5,10^{-10})$	$1.0000, 1.0000, 1.0000, 1.0000, 1.0000$
$(5,10^{-8})$	$1.0000, 1.0000, 1.0000, 1.0000, 1.0000$
$(5,10^{-6})$	$0.9993, 1.0025, 0.9967, 1.0019, 0.9996$
$(7,10^{-10})$	$1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000$
$(7,10^{-8})$	$0.9999, 1.0002, 0.9995, 1.0005, 0.9997, 1.0001, 1.0000$
$(7,10^{-6})$	$0.9950, 1.0245, 0.9503, 1.0536, 0.9676, 1.0104, 0.9986$
$(9,10^{-10})$	$1.0000, 1.0000, 1.0000, 1.0001, 0.9999, 1.0000, 1.0000, 1.0000, 1.0000$
$(9,10^{-8})$	$0.9998, 1.0015, 0.9961, 1.0060, 0.9944, 1.0034, 0.9987, 1.0003, 1.0000$
$(9,10^{-6})$	$0.9758, 1.1481, 0.6062, 1.5958, 0.4389, 1.3367, 0.8743, 1.0267, 0.9975$

表 11: 在 $\boldsymbol{A}_2(n,n)$ 处加扰动后的 $\boldsymbol{x}_2$ 求解结果

(n,ε)	$\boldsymbol{x}_2^T$
$(5,10^{-10})$	$1.0000, 1.0000, 1.0000, 1.0000, 1.0000$
$(5,10^{-8})$	$1.0000, 1.0001, 0.9994, 1.0009, 0.9996$
$(5,10^{-6})$	$0.9994, 1.0121, 0.9457, 1.0845, 0.9578$
$(7,10^{-10})$	$1.0000, 1.0001, 0.9995, 1.0020, 0.9962, 1.0033, 0.9989$
$(7,10^{-8})$	$0.9999, 1.0045, 0.9546, 1.1816, 0.6594, 1.2997, 0.9001$
$(7,10^{-6})$	$0.9990, 1.0417, 0.5830, 2.6679, -2.1273, 3.7520, 0.0827$
$(9,10^{-10})$	$1.0000, 1.0012, 0.9785, 1.1577, 0.4085, 2.2304, -0.4355, 1.8789, 0.7803$
$(9,10^{-8})$	$0.9999, 1.0054, 0.9055, 1.6933, -1.6000, 6.4079, -5.3093, 4.8628, 0.0343$
$(9,10^{-6})$	$0.9999, 1.0056, 0.9021, 1.7177, -1.6914, 6.5980, -5.5310, 4.9986, 0.0004$

表 12: 在 $\boldsymbol{b}_1(n)$ 处加扰动后的 $\boldsymbol{x}_1$ 求解结果

(n,ε)	$\boldsymbol{x}_1^T$
$(5,10^{-10})$	$1.0000, 1.0000, 1.0000, 1.0000, 1.0000$
$(5,10^{-8})$	$1.0000, 1.0000, 1.0000, 1.0000, 1.0000$
$(5,10^{-6})$	$1.0007, 0.9975, 1.0033, 0.9981, 1.0004$
$(7,10^{-10})$	$1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000$
$(7,10^{-8})$	$1.0001, 0.9998, 1.0005, 0.9995, 1.0003, 0.9999, 1.0000$
$(7,10^{-6})$	$1.0050, 0.9755, 1.0497, 0.9464, 1.0324, 0.9896, 1.0014$
$(9,10^{-10})$	$1.0000, 1.0000, 1.0000, 0.9999, 1.0001, 1.0000, 1.0000, 1.0000, 1.0000$
$(9,10^{-8})$	$1.0002, 0.9985, 1.0039, 0.9940, 1.0056, 0.9966, 1.0013, 0.9997, 1.0000$
$(9,10^{-6})$	$1.0243, 0.8516, 1.3948, 0.4027, 1.5624, 0.6625, 1.1260, 0.9732, 1.0025$

表 13: 在 $\boldsymbol{b}_2(n)$ 处加扰动后的 $\boldsymbol{x}_2$ 求解结果

(n,ε)	$\boldsymbol{x}_2^T$
$(5,10^{-10})$	$1.0000, 1.0000, 1.0000, 1.0000, 1.0000$
$(5,10^{-8})$	$1.0000, 0.9999, 1.0006, 0.9991, 1.0004$
$(5,10^{-6})$	$1.0006, 0.9874, 1.0567, 0.9118, 1.0441$
$(7,10^{-10})$	$1.0000, 0.9999, 1.0005, 0.9980, 1.0038, 0.9967, 1.0011$
$(7,10^{-8})$	$1.0001, 0.9950, 1.0505, 0.7982, 1.3784, 0.6670, 1.1110$
$(7,10^{-6})$	$1.0120, 0.4955, 6.0450, -19.1802, 38.8378, -32.2973, 12.0991$
$(9,10^{-10})$	$1.0000, 0.9984, 1.0276, 0.7978, 1.7581, -0.5768, 2.8397, -0.1263, 1.2816$
$(9,10^{-8})$	$1.00, 0.84, 3.76, -19.22, 76.81, -156.69, 184.97, -111.63, 29.16$
$(9,10^{-6})$	$1, -15, 277, -2021, 7582, -15768, 18398, -11262, 2817$