实验序号： 实验 2　　**实验项目：** 消去法和迭代法

实验成绩：　　**教师签名：**

西 南 大 学 人 工 智 能 学 院
实 验 报 告

学年学期	2023 年秋季学期
课程名称	数值分析
姓　　名	洪浩钦
学　　号	222021335210144
学　　院	含弘学院
专　　业	智能科学与技术
班　　级	袁隆平班
任课教师	钟秀蓉

2023 年 **11** 月 **29** 日

一、实验目的

1. **知识目标：**巩固消去法和迭代法的数值分析理论知识和常规解法。通过实验，学生将回顾和巩固数值分析中消去法和迭代法的基本原理和解法。
2. **能力目标：**能够使用 MATLAB 利用高斯消去法和三角分解法求解线性方程组。学生将通过实验，掌握在 MATLAB 中编写相关函数和算法来求解线性方程组的能力。
3. **素质目标：**培养学生对数值分析的兴趣，树立数值分析意识，并提升自主学习和不断拓展的探索能力。通过实验，学生将体验数值分析的实际应用，并培养对数值分析领域的兴趣和热情，同时培养自主学习和不断拓展的能力，以便在更复杂的问题上进行研究和探索。

二、算法原理概述

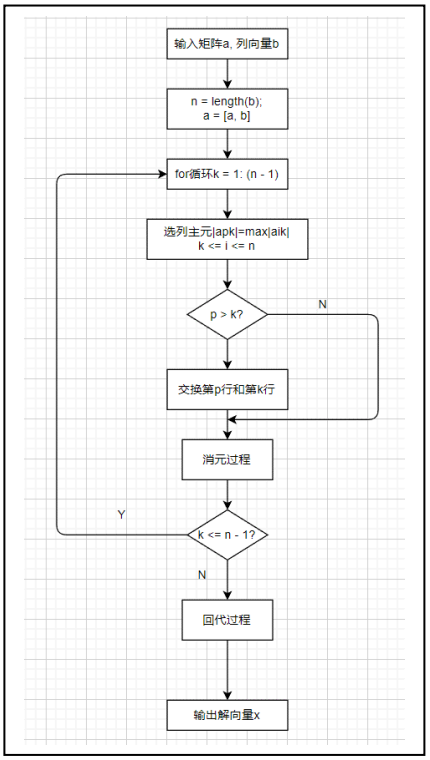
1. 高斯消去法：

高斯消去法是一种直接求解线性方程组的方法。它通过逐步消元将线性方程组的系数矩阵转化为上三角矩阵，然后通过回代求解得到方程组的解。

前向消元：通过逐行操作，将系数矩阵转化为上三角形式。在每一步操作中，通过将某一行乘以一个倍数并加到另一行上，消去该行的一个元素。

回代求解：从最后一行开始，通过回代求解出方程组的解。每一步操作中，将已知解代入到方程中，求解当前未知数，并逐步往上回代，直到求解出所有未知数。

选列主元高斯消去法的流程如图所示：



2. 三角分解法：

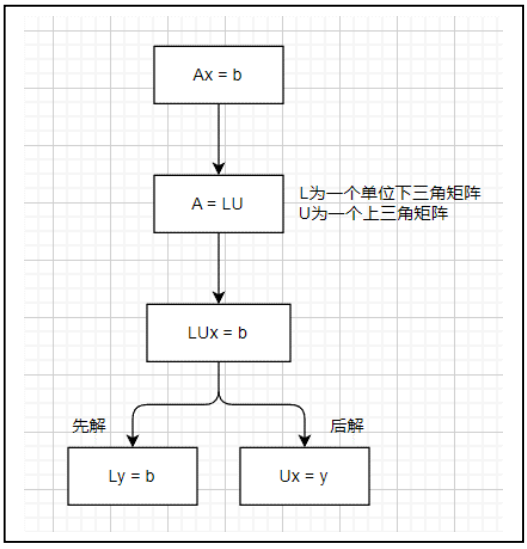
三角分解法是一种利用 LU 分解来求解线性方程组的方法。它将系数矩阵分解为一个下三角矩阵（L）和一个上三角矩阵（U）的乘积形式。

LU 分解：将系数矩阵分解为下三角矩阵 L 和上三角矩阵 U 的乘积形式。在分解过程中，通过逐行操作，将系数矩阵转化为 LU 形式，其中 L 的对角线元素为 1。

前代求解：通过前代法求解下三角矩阵 L 与向量 b 的乘积，得到一个新的向量 y。

回代求解：通过回代法求解上三角矩阵 U 与向量 y 的乘积，得到线性方程组的解向量 x。

三角分解法的流程如图：



3. 牛顿迭代法：

牛顿迭代法是一种求解非线性方程 $f(x) = 0$ 的数值方法。它的基本思想是利用函数的切线来逼近函数的根。牛顿迭代法的步骤如下：

I. 选择一个初始点 $x0$。

II. 计算函数在 $x0$ 处的值 $f(x0)$ 和导数值 $f'(x0)$。

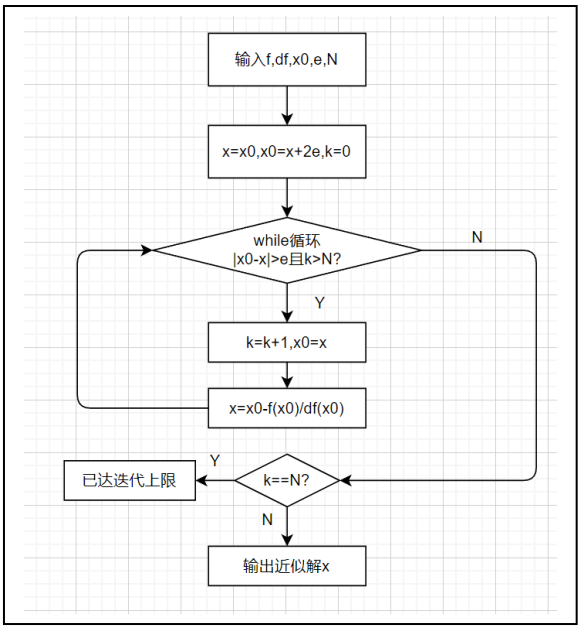
III. 计算下一个点 $x1 = x0 - f(x0) / f'(x0)$。

IV. 如果 $|x1 - x0|$ 小于预定的误差阈值，那么停止迭代，$x1$ 就是方程的一个根。

V. 否则，将 $x1$ 作为新的 $x0$，回到步骤 2。

这个过程会一直重复，直到找到一个满足误差要求的解，或者达到预设的最大迭代次数。在这个过程中，每一次迭代都是用函数在当前点的切线来逼近函数的根。因为切线的斜率就是函数的导数，所以这个方法需要知道函数的导数表达式。这也是牛顿迭代法的一个主要限制，它只适用于能够求导的函数。

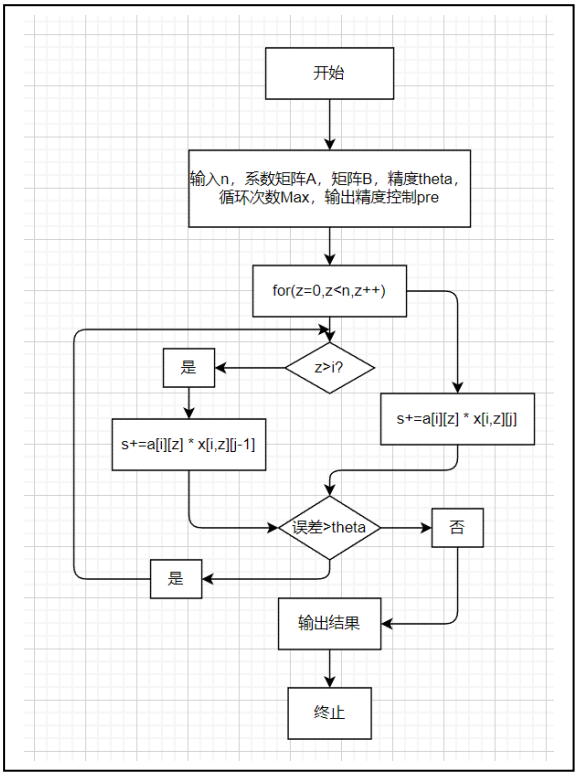
牛顿迭代法的流程如图：



4. 高斯-赛德尔迭代法：

高斯-赛德尔迭代法是一种求解线性方程组的数值方法。它是雅可比迭代法的一个改进版本，每次迭代时都使用最新的解来更新下一个解，这样通常可以加快收敛速度。

高斯-赛德尔迭代法的流程图如图：



三、软件开发环境及工具

MATLAB 2021b

四、实验内容

1、问题提出 **2、解决思路** **3、算法步骤**

4、结果分析 **5、实验核心代码**

消去法实验 1 用矩阵除法解下列线性方程组

（1）[4 1 -1; 3 2 -6; 1 -5 3][x1; x2; x3] = [9; -2; 1]

1）实验代码：

```
% [4 1 -1; 3 2 -6; 1 -5 3][x1; x2; x3] = [9; -2; 1]
A = [4 1 -1; 3 2 -6; 1 -5 3];
b = [9; -2; 1];
x = A\b;
disp(x);
```

2）实验结果：

```
>> disp(x);
    2.3830
    1.4894
    2.0213
```

解得 x1 = 2.3830 x2 = 1.4894 x3 = 2.0213

（2）[4 1; 3 2; 1 -5][x1; x2] = [1; 1; 1]

1）实验代码：

```
% [4 1; 3 2; 1 -5][x1; x2] = [1; 1; 1]
A = [4 1; 3 2; 1 -5];
b = [1; 1; 1];
x = A\b;
disp(x);
```

2）实验结果：

```
>> disp(x);
    0.3311
```

-0.1219

解得 x1 = 0.3311 x2 = -0.1219

消去法实验 2 求下列矩阵的行列式、逆、特征值、特征向量、各种范数和条件数

实验思路分析：

计算行列式：使用 det(A) 函数。计算逆矩阵：使用 inv(A) 函数。

计算特征值和特征向量：使用 [V,D] = eig(A) 函数。这个函数会返回一个包含特征值的对角矩阵 D 和一个包含对应特征向量的矩阵 V。

计算范数：使用 norm(A, type) 函数。这里的 type 可以是 1（一范数）、2（二范数）、'fro'（Frobenius 范数）或 'inf'（无穷范数）。

计算条件数：使用 cond(A) 函数。

（1）A = [4 1 -1; 3 2 -6; 1 -5 3];

1）实验代码：

```
% 定义矩阵
A = [4 1 -1; 3 2 -6; 1 -5 3];

% 计算行列式
det_A = det(A);
disp(['Determinant: ', num2str(det_A)])

% 计算逆矩阵
inv_A = inv(A);
disp('Inverse:')
disp(inv_A)

% 计算特征值和特征向量
[V,D] = eig(A);
disp('Eigenvalues:')
disp(diag(D))
disp('Eigenvectors:')
disp(V)

% 计算各种范数
norm_1 = norm(A, 1); % 1范数
norm_2 = norm(A, 2); % 2范数
norm_inf = norm(A, inf); % 无穷范数
norm_fro = norm(A, 'fro'); % Frobenius范数
disp(['1-norm: ', num2str(norm_1)])
disp(['2-norm: ', num2str(norm_2)])
disp(['Inf-norm: ', num2str(norm_inf)])
disp(['Frobenius norm: ', num2str(norm_fro)])
```

装 订 线

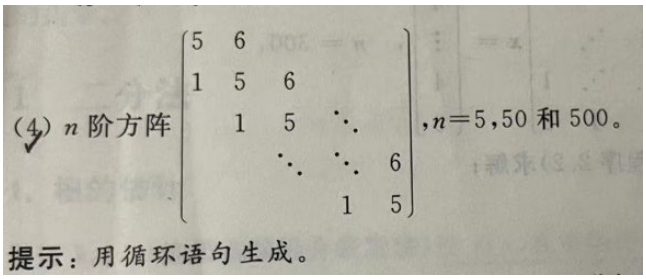
```
% 计算条件数
cond_1 = cond(A, 1); % 1条件数
cond_2 = cond(A, 2); % 2条件数
cond_inf = cond(A, inf); % 无穷条件数
disp(['1-condition number: ', num2str(cond_1)])
disp(['2-condition number: ', num2str(cond_2)])
disp(['Inf-condition number: ', num2str(cond_inf)])
```

2）实验结果

```
>> exp2_1
Determinant: -94
Inverse:
    0.2553   -0.0213    0.0426
    0.1596   -0.1383   -0.2234
    0.1809   -0.2234   -0.0532

Eigenvalues:
   -3.0527
    3.6760
    8.3766

Eigenvectors:
    0.0185   -0.9009   -0.3066
   -0.7693   -0.1240   -0.7248
   -0.6386   -0.4158    0.6170

1-norm: 10
2-norm: 8.6089
Inf-norm: 11
Frobenius norm: 10.0995
1-condition number: 5.9574
2-condition number: 3.7494
Inf-condition number: 5.734
```

装 订 线

（4）



1）循环语句定义矩阵

定义一个稀疏 n 阶矩阵，第一行前两列为 5、6，最后一行后两列为 1、5，其余行从上到下 1、5、6 依次往右移动一列。

2）实验代码

```
% 定义矩阵
n = input('请输入n的值:');
A = zeros(n, n);
A(1, 1) = 5;
A(1, 2) = 6;
for i = 2: n - 1
    A(i, i - 1) = 1;
    A(i, i) = 5;
    A(i, i + 1) = 6;
end
A(n, n - 1) = 1;
A(n, n) = 5;
disp(A)

% 计算行列式
det_A = det(A);
disp(['Determinant: ', num2str(det_A)])

% 计算逆矩阵
inv_A = inv(A);
disp('Inverse:')
disp(inv_A)

% 计算特征值和特征向量
[V,D] = eig(A);
disp('Eigenvalues:')
disp(diag(D))
disp('Eigenvectors:')
disp(V)
```

装 订 线

```
% 计算各种范数
norm_1 = norm(A, 1); % 1范数
norm_2 = norm(A, 2); % 2范数
norm_inf = norm(A, inf); % 无穷范数
norm_fro = norm(A, 'fro'); % Frobenius范数
disp(['1-norm: ', num2str(norm_1)])
disp(['2-norm: ', num2str(norm_2)])
disp(['Inf-norm: ', num2str(norm_inf)])
disp(['Frobenius norm: ', num2str(norm_fro)])

% 计算条件数
cond_1 = cond(A, 1); % 1条件数
cond_2 = cond(A, 2); % 2条件数
cond_inf = cond(A, inf); % 无穷条件数
disp(['1-condition number: ', num2str(cond_1)])
disp(['2-condition number: ', num2str(cond_2)])
disp(['Inf-condition number: ', num2str(cond_inf)])
```

3）实验结果

以 n = 5 为例（其他情况输出数据太多，展示起来不美观）

```
>> exp2_2
请输入 n 的值:5
     5     6     0     0     0
     1     5     6     0     0
     0     1     5     6     0
     0     0     1     5     6
     0     0     0     1     5

Determinant: 665
Inverse:
    0.3173   -0.5865    1.0286   -1.6241    1.9489
   -0.0977    0.4887   -0.8571    1.3534   -1.6241
    0.0286   -0.1429    0.5429   -0.8571    1.0286
   -0.0075    0.0376   -0.1429    0.4887   -0.5865
    0.0015   -0.0075    0.0286   -0.0977    0.3173

Eigenvalues:
```

0.7574
9.2426
7.4495
5.0000
2.5505

Eigenvectors:

-0.7843	-0.7843	-0.9237	0.9860	-0.9237
0.5546	-0.5546	-0.3771	-0.0000	0.3771
-0.2614	-0.2614	0.0000	-0.1643	-0.0000
0.0924	-0.0924	0.0628	0.0000	-0.0628
-0.0218	-0.0218	0.0257	0.0274	0.0257

1-norm: 12
2-norm: 11.2872
Inf-norm: 12
Frobenius norm: 16.5227
1-condition number: 66.0632
2-condition number: 45.2958
Inf-condition number: 66.0632

消去法实验 3 计算实验 2 第（1）小题的选列主元 LU 分解和奇异值分解

1）实验思路分析

选列主元 LU 分解函数的定义：

选列主元的 LU 分解在每一步中都选择剩余子矩阵的最大元素作为主元。

首先初始化了 L、U 和 P。L 和 P 是单位矩阵，U 是原矩阵 A。然后，进行 n 次循环，每次循环对应于 LU 分解的一步。在每一步中，首先找到剩余子矩阵 U(k:n, k) 的最大元素，然后将这个元素所在的行交换到第 k 行。这就是选列主元的步骤，它通过行交换操作保证了主元总是剩余子矩阵的最大元素。

接着，交换 L 和 P 的对应行。注意，L 的交换只涉及到第 1 到 k-1 列，因为 L 的第 k 列及之后的列在这一步还没有计算。

最后，计算 L 的第 k 列和 U 的第 k 行。这是通过高斯消元法完成的，具体来说，就是将 U 的第 j 行减去 L(j, k) * U(k, :)，这样可以消去 U(j, k)。

通过这个过程，最终可以得到 L、U 和 P，满足 PA＝LU。这就是选列主元的 LU 分解的原理。

```
% 选列主元 LU 分解的函数
function [L, U, P] = lu_decomposition(A)
    [m, n] = size(A);
    L = eye(n);
    U = A;
    P = eye(n);
    for k = 1:n
        [~, i] = max(abs(U(k:n, k)));
        i = i + k - 1;
        U([k, i], :) = U([i, k], :);
        L([k, i], 1:k-1) = L([i, k], 1:k-1);
        P([k, i], :) = P([i, k], :);
        for j = k+1:n
            L(j, k) = U(j, k) / U(k, k);
            U(j, k:n) = U(j, k:n) - L(j, k) * U(k, k:n);
        end
    end
end
```

奇异值分解函数的定义：

奇异值分解是一种在线性代数中常用的矩阵分解方法，它将一个矩阵分解为三个矩阵的乘积，即 A＝USV'，其中 U 和 V 是正交矩阵，S 是对角矩阵，对角线上的元素就是 A 的奇异值。

2）实验代码

```
% 定义矩阵
A = [4 1 -1; 3 2 -6; 1 -5 3];

% 计算选列主元 LU 分解
[L, U, P] = lu_decomposition(A);
disp('LU decomposition:')
disp('L:')
disp(L)
disp('U:')
disp(U)
disp('P:')
disp(P)

% 计算奇异值分解
[U, S, V] = svd_decomposition(A);
```

```
disp('SVD decomposition:')
disp('U:')
disp(U)
disp('S:')
disp(S)
disp('V:')
disp(V)
```

3）实验结果

```
>> exp3
LU decomposition:
L:
    1.0000         0         0
    0.2500    1.0000         0
    0.7500   -0.2381    1.0000

U:
    4.0000    1.0000   -1.0000
         0   -5.2500    3.2500
         0         0   -4.4762

P:
     1     0     0
     0     0     1
     0     1     0

SVD decomposition:
U:
   -0.3174    0.5778   -0.7519
   -0.7839    0.2862    0.5509
    0.5335    0.7643    0.3621

S:
    8.6089         0         0
```

0 4.7555 0
0 0 2.2961

V:

-0.3587 0.8273 -0.4323
-0.5289 -0.5617 -0.6362
0.7692 -0.0004 -0.6390

消去法实验 8 分别用顺序高斯消去和列选主元高斯消去法求解下列方程组。

1）顺序高斯消去法函数定义及代码

函数 nagauss(a, b, flag) 接收三个参数：系数矩阵 a，右侧列向量 b，以及一个标志 flag，用于决定是否显示中间过程。如果 flag 未给出，它默认为 0，即不显示中间过程。函数首先将 b 向量添加到 a 矩阵的右侧，形成增广矩阵 a。

然后，函数进行高斯消元的前向消元步骤。这个步骤通过行操作将 a 矩阵转化为阶梯形矩阵。具体来说，对于每一行 k，函数都会将 k 行以下的行减去适当的倍数，使得这些行的第 k 列元素变为 0。如果 flag 为 0，函数会在每一步消元后打印出当前的 a 矩阵。在完成所有的前向消元步骤后，函数进行回代步骤。这个步骤从最后一行开始，逐行向上计算解向量 x 的元素。具体来说，对于每一行 k，函数都会计算 x(k) 的值，使得第 k 行的等式成立。通过这个过程，函数最终返回解向量 x，满足 ax = b。这就是顺序高斯消元法的原理。

```
function x = nagauss (a, b, flag)
% Sequential Gaussian elimination solves linear systems of equations ax = b
% a: Coefficient matrix    b: The right column vector
% flag: Showing intermediate processes, default as 0
% x: Solution vector
    if nargin < 3, flag = 0; end
    n = length (b); a = [a, b];

    % Elimination of elements
    for k = 1 : (n - 1)
        a ((k + 1) : n, (k + 1) : (n + 1)) = a ((k + 1) : n, (k + 1) : (n + 1)) - a ((k + 1) : n, k) / a
(k, k) * a (k, (k + 1) : (n + 1));
        a ((k + 1) : n, k) = zeros(n - k, 1);
        if flag == 0
            a
        end
    end

    % Back generation
```

```
    x = zeros (n, 1);
    x (n) = a (n, n + 1) / a (n, n);
    for k = n - 1 : -1 : 1
        x (k) = (a (k, n + 1) - a(k, (k + 1) : n) * x ((k + 1) : n)) / a (k, k);
    end
end
```

2）选列主元高斯消去法的函数定义及代码

函数 selected_nagauss(a, b, flag) 接收三个参数：系数矩阵 a，右侧列向量 b，以及一个标志 flag，用于决定是否显示中间过程。如果 flag 未给出，它默认为 0，即不显示中间过程。

函数首先将 b 向量添加到 a 矩阵的右侧，形成增广矩阵 a。然后，函数进行高斯消元的前向消元步骤。这个步骤通过行操作将 a 矩阵转化为阶梯形矩阵。具体来说，对于每一行 k，函数都会找到 k 列以下的元素中绝对值最大的元素，然后将这个元素所在的行交换到第 k 行。这就是列主元选取的步骤，它通过行交换操作保证了主元总是剩余子矩阵的最大元素。接着，函数会将 k 行以下的行减去适当的倍数，使得这些行的第 k 列元素变为 0。这就是消元的步骤。如果 flag 为 0，函数会在每一步消元后打印出当前的 a 矩阵。在完成所有的前向消元步骤后，函数进行回代步骤。这个步骤从最后一行开始，逐行向上计算解向量 x 的元素。具体来说，对于每一行 k，函数都会计算 x(k) 的值，使得第 k 行的等式成立。通过这个过程，函数最终返回解向量 x，满足 ax = b。这就是列主元选取的高斯消元法的原理。

```
function x = selected_nagauss(a, b, flag)
% Gaussian elimination with selected column pivot elements solves linear systems of equations
ax = b
% a: Coefficient matrix    b: The right column vector
% flag: Showing intermediate processes, default as 0
% x: Solution vector
    if nargin < 3
        flag = 0;
    end
    n = length(b);
    a = [a, b];

    for k = 1 : (n - 1)
        % Select the column pivot
        [~, p] = max(abs(a(k:n, k)));                    % Find the relative position of the
largest number in absolute value
        p = p + k - 1;                                   % Converts relative
```

```
position to absolute position
            if p > k                                                    % Swap lines
                  t = a(k, :);
                  a(k, :) = a(p, :);
                  a(p, :) = t;
            end

            % Elimination of elements
            for j = (k + 1) : n
                  a(j, k+1:end) = a(j, k+1:end) - a(j, k) / a(k, k) * a(k, k+1:end);
                  a(j, k) = 0;
                  if flag == 0
                        a
                  end
            end
      end

      % Back substitution
      x = zeros(n, 1);
      x(n) = a(n, n+1) / a(n, n);
      for k = n - 1 : -1 : 1
            x(k) = (a(k, n+1) - a(k, k+1:n) * x(k+1:n)) / a(k, k);
      end
end
```

3）方程组的求解

```
% 分别用顺序高斯消去法和列选主元高斯消去法求解
% [0.3e-15 59.14 3 1; 5.291 -6.13 -1 2; 11.2 9 5 2; 1 2 1 1][x1; x2; x3; x4]=[59.17; 46.78; 1; 2]

% 顺序高斯消去法
disp('顺序高斯消去法');
format loose
A = [0.3e-15 59.14 3 1; 5.291 -6.13 -1 2; 11.2 9 5 2; 1 2 1 1];
b = [59.17; 46.78; 1; 2];
x = nagauss(A, b);

% 列选主元高斯消去法
disp('列选主元高斯消去法');
format loose
A = [0.3e-15 59.14 3 1; 5.291 -6.13 -1 2; 11.2 9 5 2; 1 2 1 1];
b = [59.17; 46.78; 1; 2];
x = selected_nagauss(A, b);
```

4）实验结果

exp5

顺序高斯消去法

```
a =
    1.0e+18 *
     0.0000      0.0000      0.0000      0.0000      0.0000
           0     -1.0430     -0.0529     -0.0176     -1.0436
           0     -2.2079     -0.1120     -0.0373     -2.2090
           0     -0.1971     -0.0100     -0.0033     -0.1972

a =
    1.0e+18 *
     0.0000      0.0000      0.0000      0.0000      0.0000
           0     -1.0430     -0.0529     -0.0176     -1.0436
           0           0     -0.0000     -0.0000           0
           0           0           0      0.0000           0

a =
    1.0e+18 *
     0.0000      0.0000      0.0000      0.0000      0.0000
           0     -1.0430     -0.0529     -0.0176     -1.0436
           0           0     -0.0000     -0.0000           0
           0           0           0      0.0000           0
```

列选主元高斯消去法

```
a =
    11.2000      9.0000      5.0000      2.0000      1.0000
           0    -10.3817     -3.3621      1.0552     46.3076
     0.0000     59.1400      3.0000      1.0000     59.1700
     1.0000      2.0000      1.0000      1.0000      2.0000
```

```
a =
   11.2000    9.0000    5.0000    2.0000    1.0000
         0  -10.3817   -3.3621    1.0552   46.3076
         0   59.1400    3.0000    1.0000   59.1700
    1.0000    2.0000    1.0000    1.0000    2.0000


a =
   11.2000    9.0000    5.0000    2.0000    1.0000
         0  -10.3817   -3.3621    1.0552   46.3076
         0   59.1400    3.0000    1.0000   59.1700
         0    1.1964    0.5536    0.8214    1.9107


a =
   11.2000    9.0000    5.0000    2.0000    1.0000
         0   59.1400    3.0000    1.0000   59.1700
         0         0   -2.8354    1.2307   56.6946
         0    1.1964    0.5536    0.8214    1.9107


a =
   11.2000    9.0000    5.0000    2.0000    1.0000
         0   59.1400    3.0000    1.0000   59.1700
         0         0   -2.8354    1.2307   56.6946
         0         0    0.4929    0.8012    0.7137


a =
   11.2000    9.0000    5.0000    2.0000    1.0000
         0   59.1400    3.0000    1.0000   59.1700
         0         0   -2.8354    1.2307   56.6946
         0         0         0    1.0151   10.5689
```

装 订 线

顺序高斯消元法在每一步中都使用当前行的主元来消元，而列选主元高斯消元法在每一步中都选择剩余子矩阵中绝对值最大的元素作为主元。这两种方法在大多数情况下都能得到正确的解，但在某些特殊情况下，顺序高斯消元法可能会因为主元过小而产生大的舍入误差。

顺序高斯消元法和列选主元高斯消元法的结果差异较大，可能是由于顺序高斯消元法的舍入误差导致的。

消去法实验 10 （追赶法的速度）考虑 n 阶三对角方程组

循环语句定义矩阵

定义一个稀疏 n 阶矩阵 A，第一行前两列为 2、1，最后一行后两列为 1、2，其余行从上到下 1、2、1 依次往右移动一列，已知 Ax = b。

（1）用选列主元高斯消去法求解

1）实验代码

```
% （追赶法的深度）考虑 n 阶三对角方程组
n = 300;
A = zeros(n, n);
A(1, 1) = 2;
A(1, 2) = 1;
for i = 2: n - 1
    A(i, i - 1) = 1;
    A(i, i) = 2;
    A(i, i + 1) = 1;
end
A(n, n - 1) = 1;
A(n, n) = 2;

% 用选列主元高斯消去法求解
b = ones(n, 1) * 4;
b(1) = 3;
b(end) = 3;
x = hidden_selected_nagauss(A, b)
euclidean_distance = norm(x - b)
```

2）实验结果

>> exp6_1

历时 0.143721 秒。

x = [1; 1; … …; 1; 1] (输出的结果太长了)

euclidean_distance =

51.8652

（2）用追赶法求解

1）追赶法函数的定义

托马斯算法是一种特殊的高斯消元法，专门用于解决三对角线性方程组。三对角线性方程组的特点是，除了主对角线上的元素和主对角线相邻的元素外，其他元素都为零。

```
function x = thomas(A, b)
    tic;

    n = length(b);
    x = zeros(n, 1);
    c = zeros(n, 1);
    d = zeros(n, 1);

    c(1) = A(1, 2) / A(1, 1);
    d(1) = b(1) / A(1, 1);

    for i = 2 : n - 1
        denominator = A(i, i) - A(i, i - 1) * c(i - 1);
        c(i) = A(i, i + 1) / denominator;
        d(i) = (b(i) - A(i, i - 1) * d(i - 1)) / denominator;
    end

    d(n) = (b(n) - A(n, n - 1) * d(n - 1)) / (A(n, n) - A(n, n - 1) * c(n - 1));

    x(n) = d(n);
    for i = n - 1 : -1 : 1
        x(i) = d(i) - c(i) * x(i + 1);
    end
    toc;
end
```

2）利用追赶法求解 n 阶三对角方程组

```
% （追赶法的深度）考虑 n 阶三对角方程组
n = 300;
A = zeros(n, n);
A(1, 1) = 2;
A(1, 2) = 1;
for i = 2: n - 1
```

```
    A(i, i - 1) = 1;
    A(i, i) = 2;
    A(i, i + 1) = 1;
end
A(n, n - 1) = 1;
A(n, n) = 2;

% 用追赶法求解
b = ones(n, 1) * 4;
b(1) = 3;
b(end) = 3;
x = thomas(A, b)
euclidean_distance = norm(x - b)
```

3）实验结果

>> exp6_2

历时 0.000213 秒。

x = [1; 1; … …; 1; 1] (输出的结果太长了)

euclidean_distance =

51.8652

（3）用矩阵除法求解

1）实验代码

```
% （追赶法的深度）考虑 n 阶三对角方程组
n = 300;
A = zeros(n, n);
A(1, 1) = 2;
A(1, 2) = 1;
for i = 2: n - 1
    A(i, i - 1) = 1;
    A(i, i) = 2;
    A(i, i + 1) = 1;
end
A(n, n - 1) = 1;
A(n, n) = 2;

% 用矩阵除法求解
b = ones(n, 1) * 4;
b(1) = 3;
b(end) = 3;
tic;
```

```
x = A \ b

toc;

euclidean_distance = norm(x - b)
```

2）实验结果

>> exp6_3

x = [1; 1; … …; 1; 1]　(输出的结果太长了)

历时 0.061960 秒。

euclidean_distance =

51.8652

（4）比较 3 种方法的计算时间

由实验结果可知，计算时间上，追赶法最快，选列主元高斯消去法最慢。

迭代法实验 2 用二分法和牛顿迭代法求解下列方程的正根

xln(sqrt(x^2 - 1)+x) – sqrt(x^2-1) – 0.5x = 0

1）二分法定义与代码

二分法是一种求解非线性方程的数值方法。它的基本思想是：如果函数在区间 [a, b] 上连续，并且在端点 a 和 b 处的函数值异号，即 f(a) * f(b) < 0，那么根据中值定理，函数在区间 (a, b) 内至少有一个根。

```
% 二分法解非线性方程 f(x) = 0
function x = nabisect(fname, a, b, e)
% 格式: x = nabisect（fname, a, b, e）
% fname 为函数句柄或内嵌函数表达式 f(x)
% a, b 为区间端点, e 为精度 (默认值为1e-4), 函数在两端点值必须异号
if nargin < 4, e = 1e-4; end;
fa = fname(a);
fb = fname(b);

if fa * fb > 0, error('函数在两端点值必须异号'); end
x = (a + b) / 2;

while (b - a) > (2 * e)
    fx = fname(x);
    if fa * fx < 0, b = x; fb = fx; else a = x; fa = fx; end
```

```
    x = (a + b) / 2;
end
```

2）牛顿迭代法定义与代码

牛顿迭代法的基本思想是利用函数的切线来逼近函数的根。nanewton 函数的输入参数包括：函数句柄 fname，它表示函数 f(x)；函数句柄 dfname，它表示函数 f(x) 的导数；初始迭代值 x0；精度要求 e；以及迭代次数上限 N。

```
% 牛顿迭代法解线性方程 f(x) = 0
function x = nanewton(fname, dfname, x0, e, N)
% 格式: x = nanewton(fname, dfname, x0, e, N)
% fname 和 dfname 分别为表示 f(x) 及其导函数的M函数句柄或内嵌函数
% x0 为迭代初值，e 为精度要求（默认1e-4）
% N 为计算过程中的迭代次数上限
if nargin < 5, N = 500; end
if nargin < 4, e = 1e-4; end
x = x0;
x0 = x + 2 * e;
k = 0;

while abs(x0 - x) > e && k < N
    k = k + 1;
    x0 = x;
    x = x0 - fname(x0)/dfname(x0);
end

if k == N, warning('已达到迭代次数上限') ;end
```

3）求解方程正根代码

```
% 用二分法和牛顿迭代法求下列方程的正根
% xln(sqrt(x ^ 2 - 1) + x) - sqrt(x ^ 2 - 1) - 0.5x = 0

syms
x;                                                                    %
创建符号变量 x
fun = x * log(sqrt(x ^ 2 - 1) + x) - sqrt(x ^ 2 - 1) - 0.5 * x; % 定义函数
dfun = diff(fun, x);                                                  % 求
解 fun 的导数

% 将符号表达式转换为函数句柄
fun = matlabFunction(fun);
dfun = matlabFunction(dfun);
```

```
% f(2) = -0.0981 f(3) = 0.9598
x0 = 2;
x1 = 3;
e = 1e-4;

% 二分法求解正根
x = nabisect(fun, x0, x1, e)

% 牛顿迭代法求解正根
N = 500;
x = nanewton(fun, dfun, x0, e, N)
```

4）实验结果

```
>> exp7
x =
    2.1155

x =
    2.1155
```

迭代法实验 4 求解下列非线性方程组在原点附近的根

求解下列非线性方程组在原点附近的根：

$$\begin{cases} 9x^2+36y^2+4z^2=36, \\ x^2-2y^2-20z=0, \\ 16x-x^3-2y^2-16z^2=0。 \end{cases}$$

1）实验代码

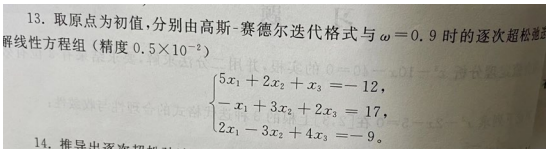
```
% 求解下列非线性方程组在原点附近的根
% 9 * x^2 + 36 * y^2 + 4 * z^2 = 36
% x^2 - 2 * y^2 - 20 * z = 0
% 16 * x - x^3   - 2 * y^2 - 16 * z^2 = 0

fun = @(x) [9 * x(1)^2 + 36 * x(2)^2 + 4 * x(3)^2 - 36;
            x(1)^2 - 2 * x(2)^2 - 20 * x(3);
            16 * x(1) - x(1)^3 - 2 * x(2)^2 - 16 * x(3)^2];

% 定义初始点
x0 = [0; 0; 0];
```

```
% 使用 fsolve 函数求解非线性方程组
x = fsolve(fun, x0);

% 显示结果
disp(x);
```

2）实验结果

```
>> exp8
Equation solved.
fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.
<stopping criteria details>
    0.1342
    0.9972
   -0.0985
```

迭代法实验 7 （雅可比迭代）编写雅可比迭代解线性方程组


13. 取原点为初值，分别由高斯-赛德尔迭代格式与 $\omega=0.9$ 时的逐次超松弛
解线性方程组（精度 0.5×10^{-2}）

$$\begin{cases}5x_1+2x_2+x_3=-12,\\-x_1+3x_2+2x_3=17,\\2x_1-3x_2+4x_3=-9.\end{cases}$$

14. 推导出逐次超松

1）雅可比迭代的定义及代码

雅可比迭代法的基本思想是将线性方程组的每一个方程都表示为一个未知数关于其他未知数的函数，然后通过迭代来逐步逼近解。

函数的主体部分是一个 for 循环，循环的次数最多为 maxIter。在每次迭代中，首先保存当前的解 x 到 x_old，然后根据雅可比迭代法的公式更新 x。这个公式是 x(i) = (b(i) - A(i, [1:i-1, i+1:end]) * x_old([1:i-1, i+1:end])) / A(i, i)，它表示的是第 i 个方程的第 i 个未知数关于其他未知数的函数。如果新的解 x 和旧的解 x_old 之间的最大差值小于误差阈值 tol，那么停止迭代。函数的输出是求解得到的 x，它是线性方程组的解。

```
% (雅可比迭代) 编写雅可比迭代解线性方程组
% 5 * x1 + 2 * x2 + x3 = -12
```

```
% x1 + x2 - 4 * x3 = 7
% -8 * x1 + x2 + x3 = 1

% 矩阵的定义
A = [5 2 1; 1 1 -4; -8 1 1];
b = [-12; 7; 1];
x0 = [0; 0; 0];
tol = 1e-6;
maxIter = 100;

x = jacobi(A, b, x0, tol, maxIter);
disp(x);
```

2）雅可比迭代解线性方程组

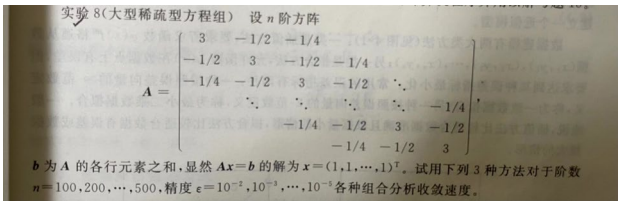
```
% (雅可比迭代) 编写雅可比迭代解线性方程组
% 5 * x1 + 2 * x2 + x3 = -12
% x1 + x2 - 4 * x3 = 7
% -8 * x1 + x2 + x3 = 1

% 矩阵的定义
A = [5 2 1; 1 1 -4; -8 1 1];
b = [-12; 7; 1];
x0 = [0; 0; 0];
tol = 1e-6;
maxIter = 100;

x = jacobi(A, b, x0, tol, maxIter);
disp(x);
```

3）实验结果

```
>> exp9
    1.0e+45 *

   -0.4810
    3.1611
   -0.4037
```

装 订 线

迭代法实验 8 （大型稀疏型方程组）


实验 8(大型稀疏型方程组) 设 n 阶方阵

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

b 为 A 的各行元素之和，显然 $Ax=b$ 的解为 $x=(1,1,\cdots,1)^T$。试用下列 3 种方法对于阶数 $n=100,200,\cdots,500$，精度 $\varepsilon=10^{-2},10^{-3},\cdots,10^{-5}$ 各种组合分析收敛速度。

1）n 阶方阵的定义：

```
n = input('请输入n的值:');
A = zeros(n, n);

A(1, 1) = 3;
A(1, 2) = -1/2
A(1, 3) = -1/4;
A(2, 1) = -1/2;
A(2, 2) = 3;
A(2, 3) = -1/2
A(2, 4) = -1/4;

for i = 3: n - 2
    A(i, i - 2) = -1/4;
    A(i, i - 1) = -1/2;
    A(i, i) = 3;
    A(i, i + 1) = -1/2
    A(i, i + 2) = -1/4;
end

A(n - 1, n) = -1/2;
A(n - 1, n - 1) = 3;
A(n - 1, n - 2) = -1/2
A(n - 1, n - 3) = -1/4;
A(n, n) = 3;
A(n, n - 1) = -1/2
A(n, n - 2) = -1/4;
```

以输入 n = 5 为例，输出的方阵如下所示：

A =

```
    3.0000   -0.5000   -0.2500         0         0
   -0.5000    3.0000   -0.5000   -0.2500         0
   -0.2500   -0.5000    3.0000   -0.5000   -0.2500
         0   -0.2500   -0.5000    3.0000   -0.5000
         0         0   -0.2500   -0.5000    3.0000
```

（1） 选列主元的高斯消去法：

```
function x = hidden_selected_nagauss(a, b, flag)
    tic;
% Gaussian elimination with selected column pivot elements solves linear systems of equations
ax = b
% a: Coefficient matrix    b: The right column vector
% flag: Showing intermediate processes, default as 0
% x: Solution vector
    if nargin < 3
        flag = 0;
    end
    n = length(b);
    a = [a, b];

    for k = 1 : (n - 1)
        % Select the column pivot
        [~, p] = max(abs(a(k:n, k)));                    % Find the relative position of the
largest number in absolute value
        p = p + k - 1;                                   % Converts relative
position to absolute position
        if p > k                                         % Swap lines
            t = a(k, :);
            a(k, :) = a(p, :);
            a(p, :) = t;
        end

        % Elimination of elements
        for j = (k + 1) : n
            a(j, k+1:end) = a(j, k+1:end) - a(j, k) / a(k, k) * a(k, k+1:end);
            a(j, k) = 0;
            if flag == 0
                a;
            end
        end
    end

    % Back substitution
    x = zeros(n, 1);
    x(n) = a(n, n+1) / a(n, n);
    for k = n - 1 : -1 : 1
        x(k) = (a(k, n+1) - a(k, k+1:n) * x(k+1:n)) / a(k, k);
    end
    toc;
end
```

装 订 线

（2） 高斯-赛德尔迭代，普通存储方式

高斯-赛德尔迭代法，用于求解线性方程组 Ax = b。它首先初始化解向量 x 和迭代次数 k，然后在满足精度要求或达到迭代上限之前，不断迭代更新解向量 x。

```
function x = nags(A, b, x0, e, N)
% A 为系数矩阵，b为右端向量，x返回解向量
% x0为初始向量，e为精度，N为迭代上限
n = length(b);
if nargin < 5, N = 500; end
if nargin < 4, e = 1e-4; end
if nargin < 3, x0 = zeros(n, 1); end
x = x0;
x0 = x + 2 * e;
k = 0; Al = tril(A); iAl = inv(Al);
tic;
while norm(x0 - x, inf) > e && k < N
    k = k + 1;
    x0 = x;
    x = -iAl * (A - Al) * x0 + iAl * b;
    x';
end
toc;
if k == N, disp('已达到最大迭代次数！'); end
```

（3） 高斯-赛德尔迭代，稀疏存储方式

稀疏存储是一种特殊的数据存储格式，主要用于存储稀疏矩阵，即矩阵中大部分元素都是零的矩阵。在代码中，sparse 函数被用来将向量或矩阵转换为稀疏格式。例如，x0 = sparse(x0);，b = sparse(b); 和 A = sparse(A); 将初始向量 x0，右端向量 b 和系数矩阵 A 都转换为稀疏格式。使用稀疏存储可以大大提高存储和计算的效率，特别是对于大型稀疏矩阵。在进行矩阵运算时，MATLAB 会自动利用稀疏矩阵的存储格式来优化计算。然后，在迭代完成后，full 函数被用来将稀疏矩阵 x 转换回常规的满元素矩阵，即 x = full(x);。

```
function x = naspgs(A, b, x0, e, N)
% A 为系数矩阵，b为右端向量，x返回解向量
% x0为初始向量，e为精度，N为迭代上限
n = length(b);
if nargin < 5, N = 500; end
if nargin < 4, e = 1e-4; end
if nargin < 3, x0 = zeros(n, 1); end

x0 = sparse(x0);
b = sparse(b);
```

```
A = sparse(A); % 使用稀疏存储

x = x0;
x0 = x + 2 * e;
x0 = sparse(x0);
k = 0;
Al = tril(A);
iAl = inv(Al);

tic;
while norm(x0 - x, inf) > e && k < N
    k = k + 1;
    x0 = x;
    x = -iAl * (A - Al) * x0 + iAl * b;
end

x = full(x); % 稀疏矩阵转成满元素矩阵
toc;
if k == N
    disp('已达到最大迭代次数！');
end
```

2）实验代码

```
n = input('请输入n的值:');
A = zeros(n, n);

A(1, 1) = 3;
A(1, 2) = -1/2;
A(1, 3) = -1/4;
A(2, 1) = -1/2;
A(2, 2) = 3;
A(2, 3) = -1/2;
A(2, 4) = -1/4;

for i = 3: n - 2
    A(i, i - 2) = -1/4;
    A(i, i - 1) = -1/2;
    A(i, i) = 3;
    A(i, i + 1) = -1/2;
    A(i, i + 2) = -1/4;
end

A(n - 1, n) = -1/2;
A(n - 1, n - 1) = 3;
A(n - 1, n - 2) = -1/2
```

```
A(n - 1, n - 3) = -1/4;
A(n, n) = 3;
A(n, n - 1) = -1/2;
A(n, n - 2) = -1/4;

b = sum(A, 2);

% 选列主元的高斯消去法
disp('选列主元的高斯消去法:')
x = hidden_selected_nagauss(A, b);

% 高斯-赛德尔迭代，普通存储方式
disp('高斯-赛德尔迭代，普通存储方式')
x = nags(A, b);

% 高斯-赛德尔迭代，稀疏存储方式
disp('高斯-赛德尔迭代，稀疏存储方式')
x = naspgs(A, b);
```

3）实验结果

精度为 1e-4

>> exp10

请输入 n 的值:100

选列主元的高斯消去法:

历时 0.013371 秒。

高斯-赛德尔迭代，普通存储方式

历时 0.001117 秒。

高斯-赛德尔迭代，稀疏存储方式

历时 0.001295 秒。

精度为 1e-2

>> exp10

请输入 n 的值:100

选列主元的高斯消去法:

历时 0.012980 秒。

高斯-赛德尔迭代，普通存储方式

历时 0.002042 秒。

高斯-赛德尔迭代，稀疏存储方式
历时 0.001636 秒。

精度为 1e-4
>> exp10
请输入 n 的值:200
选列主元的高斯消去法:
历时 0.060763 秒。
高斯-赛德尔迭代，普通存储方式
历时 0.003203 秒。
高斯-赛德尔迭代，稀疏存储方式
历时 0.005338 秒。

精度为 1e-2
>> exp10
请输入 n 的值:200
选列主元的高斯消去法:
历时 0.052907 秒。
高斯-赛德尔迭代，普通存储方式
历时 0.002253 秒。
高斯-赛德尔迭代，稀疏存储方式
历时 0.002107 秒。

精度为 1e-4
>> exp10
请输入 n 的值:500
选列主元的高斯消去法:
历时 0.431816 秒。
高斯-赛德尔迭代，普通存储方式
历时 0.039537 秒。
高斯-赛德尔迭代，稀疏存储方式
历时 0.023767 秒。

装 订 线

精度为 1e-2
>> exp10
请输入 n 的值:500
选列主元的高斯消去法:
历时 0.402803 秒。
高斯-赛德尔迭代，普通存储方式
历时 0.021444 秒。
高斯-赛德尔迭代，稀疏存储方式
历时 0.011092 秒。

由实验结果可知，精度设置得越小，收敛速度就越慢，并且选列主元的高斯消去法 <高斯-赛德尔迭代，普通存储方式 <高斯-赛德尔迭代，稀疏存储方式。

装

订

线

五、实验总结

通过这次系列实验,我学习和练习了数值分析中的一些重要算法。

首先,了解了高斯消去法、三角分解法等直接解法求解线性方程组。通过实践,掌握了这些方法在 MATLAB 中的实现。

其次,学习了二分法、牛顿迭代法等常见的非线性方程求根算法。实现了这些算法在 MATLAB 中的应用,能独立解决一些典型非线性问题。

此外,还学习了 Jacobi 迭代法等迭代算法解线性方程组。掌握了迭代算法的原理及 MATLAB 实现。

通过与高斯-赛德尔迭代法的对比,认识到稀疏存储在大型方程组求解中的优势。此外,通过多个实例应用各种算法,对算法的收敛性和性能有了初步认识。例如收敛速度为高斯-赛德尔>Jacobi 等。

总体来说,这次实验使我系统地学习了数值分析中一些重要的原理和算法。不仅理论知识得到了巩固,也掌握了算法在 MATLAB 中的实际运用。之后还需要更多练习,对一些高级算法有更深入的理解。这些知识也为我今后应用数值分析方法提供基础。