Assignment 2

Zou Yuan
Student No. 21960216

October 6, 2019

1 Gauss Elimination

The problems to be solved in this section are below:

Put together the code fragments in this section to create a Matlab program for “naive” Gaussian elimination (meaning no row exchanges allowed). Use it to solve the systems of Exercise 2:

$$\begin{aligned} 2x - 2y - z &= -2 \\ x + y - 2z &= 1 \\ -2x + y - z &= -3 \end{aligned} \tag{1}$$

$$\begin{aligned} x + 2y - z &= 2 \\ 3y + z &= 4 \\ 2x - y + z &= 2 \end{aligned} \tag{2}$$

$$\begin{aligned} 2x + y - 4z &= -7 \\ x - y + z &= -2 \\ -x + 3y - 2z &= 6 \end{aligned} \tag{3}$$

The Matlab code is as follows:

```
function [r_matrix] = GaussElimination(a,b)
%   a:系数矩阵，为n*n维方阵
%   b:输出向量，为n*1维矩阵
```

```
%     r_matrix:计算结果向量,为n*1为矩阵

% 判断输入矩阵维度是否满足要求
[row_coeff,col_coeff] = size(a);
[row_load,~] = size(b');
% 初始化r_matrix矩阵
r_matrix = zeros(row_load,1);
% 判断输入的维度是否满足要求
if (row_coeff ~= col_coeff) || (row_coeff ~=
    row_load)
% 不满足则输出错误提示
print('输入错误！');
%消去过程
else
n=row_coeff;
for j = 1 : n-1
if abs(a(j,j))<eps; error('zero pivot
    encountered');
end
for i = j+1 : n
mult = a(i,j)/a(j,j);
for k = j+1:n
a(i,k) = a(i,k) - mult*a(j,k);
end
b(i) = b(i) - mult*b(j);
end
end
%回代过程
r_matrix(n) = b(n)/a(n,n);
for k = n-1:-1:1
sum_temp = 0;
for j = k+1:n
sum_temp = sum_temp + a(k,j)*r_matrix(j);
```

```
        end
        r_matrix(k) = (b(k) - sum_temp)/a(k,k);
        end
        end % 条件判断结束
        end
```

The inputs are the coefficient matrixs and output vectors of the following expression:

$$\begin{bmatrix} 2 & -2 & -1 \\ 4 & 1 & -2 \\ -2 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \\ -3 \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} 1 & 2 & -1 \\ 0 & 3 & 1 \\ 2 & -1 & 1 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 2 \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} 2 & 1 & -4 \\ 1 & -1 & 1 \\ -1 & 3 & -2 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -7 \\ -2 \\ 6 \end{bmatrix} \tag{3}$$

The output are as below:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \\ 2 \end{bmatrix} \tag{3}$$

2 LU Factorization

The problems to be solved in this section are below:

Use the code fragments for Gaussian elimination in the previous section to write a Matlab script to take a matrix A as input and output L and U. No row exchanges are allowed—the program should be designed to shut down if it encounters a zero pivot. Check your program by factoring the matrices in Gauss Elimination.

The Matlab code is as follows:

```
function [L_matrix,U_matrix] = LU(a)
% 判断输入矩阵维度是否满足要求
[row_coeff,col_coeff] = size(a);
if (row_coeff ~= col_coeff)
error('输入错误！');
end
% 初始化L_matrix矩阵
n=row_coeff;
L_matrix = zeros(n,n);
for i = 1:n;     L_matrix(i,i) = 1;   end
for j = 1 : n-1
if abs(a(j,j))<eps
error('zero pivot encountered');
end
for i = j+1 : n
mult = a(i,j)/a(j,j);
L_matrix(i,j) = mult;
for k = j:n
a(i,k) = a(i,k) - mult*a(j,k);
end
end
U_matrix=a;
end
```

The output is:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 3 & 1 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix} \tag{1}$$

$$L = \begin{bmatrix} 1.0000 & 0 & 0 \\ 1.0000 & 1.0000 & 0 \\ 0.5000 & 0.5000 & 1.0000 \end{bmatrix} \qquad U = \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 2 \\ 0 & 0 & 2 \end{bmatrix} \tag{2}$$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 1 & -1 & 1 & 2 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{3}$$

3 SOURCES Of ERROR

The problem to be solved in this section is shown below: For the$n \times n$ matrix with entries $A_{ij} = 5/(i + 2j–1)$, set $x = [1, ..., 1]^T$ and $b = Ax$. Use the Matlab program from Computer Problem 2.1.1 or Matlab' s backslash command to compute x_c, the double precision computed solution. Find the infinity norm of the forward error and the error magnification factor of the problem $Ax = b$, and compare it with the condition number of $A : (a)n = 6(b)n = 10$.

The Matlab code is as follows:

```
function [ ] = Matrix(n)
matrix_A=zeros(n);
for i=1:n
for k=1:n
matrix_A(i,k)=5/(i+2*k-1);
end
end
x = ones(1,n);
b = matrix_A*(x');
xc = matrix_A\(x');
r = b-matrix_A*xc;
back_error=norm(r,inf)/norm(b,inf);
forward_error=norm(x-xc,inf)/norm(x,inf);
cond=forward_error/back_error;
fprintf('back_error=%d',back_error)
```

```
fprintf('forward_error=%d',forward_error)
fprintf('cond=%d',cond)
```

The $n = 6$ output is:

$$back\_error = 8.367347e - 01$$
$$forward\_error = 5.636625e + 03$$
$$cond = 6.736454e + 03$$

The $n = 10$ output is:

$$back\_error = 8.634331e - 01$$
$$forward\_error = 4.597081e + 06$$
$$cond = 5.324188e + 06$$

4 PA=LU

The problem to be solved in this section is to find the PA=LU factorization (using partial pivoting) of the following matrices:

$$\begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 3 \\ 2 & 1 & 1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 & -3 \\ 2 & 4 & 2 \\ -1 & 0 & 3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ -2 & 1 & 0 \end{bmatrix}$$

The matlab script is:

```
A=[1,1,0;2,1,-1;-1,1,-1];
B=[0,1,3;2,1,1;-1,-1,2];
C=[1,2,-3;2,4,2;-1,0,3];
D=[0,1,0;1,0,2;-2,1,0];
[L,U,P]  =lu(A)
[L,U,P]=lu(B)
[L,U,P]=lu(C)
[L,U,P]=lu(D)
```

The outputs are:

$$L = \begin{bmatrix} 1.0000 & 0 & 0 \\ -0.5000 & 1.0000 & 0 \\ 0.5000 & 0.3333 & 1.0000 \end{bmatrix} U = \begin{bmatrix} 2.0000 & 1.0000 & -1.0000 \\ 0 & 1.5000 & -1.5000 \\ 0 & 0 & 1.0000 \end{bmatrix} \quad P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$L = \begin{bmatrix} 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 \\ -0.5000 & -0.5000 & 1.0000 \end{bmatrix} \quad U = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 4 \end{bmatrix} \quad P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$L = \begin{bmatrix} 1.0000 & 0 & 0 \\ -0.5000 & 1.0000 & 0 \\ -0.5000 & 0 & 1.0000 \end{bmatrix} \quad U = \begin{bmatrix} 2 & 4 & 2 \\ 0 & 2 & 4 \\ 0 & 0 & -4 \end{bmatrix} \quad P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$L = \begin{bmatrix} 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 \\ -0.5000 & -0.5000 & 1.0000 \end{bmatrix} U = \begin{bmatrix} -2 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$