

## Ex 2.1

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.

$$\begin{aligned} &2x - 2y - z = -2 \\ (a) \quad &4x + y - 2z = 1 \\ &-2x + y - z = -3 \end{aligned}$$

```
a = [[2,-2,-1];[4,1,-2];[-2,1,-1]];
b = [-2,1,-3];
n=3;
solution = Gaussian_elimination(a,b,n)
```

```
solution = 3x1
    1
    1
    2
```

$$\begin{aligned} &x + 2y - z = 2 \\ (b) \quad &3y + z = 4 \\ &2x - y + z = 2 \end{aligned}$$

```
a = [[1,2,-1];[0,3,1];[2,-1,1]];
b = [2,4,2];
n = 3;
solution = Gaussian_elimination(a,b,n)
```

```
solution = 3x1
    1
    1
    1
```

$$\begin{aligned} &2x + y - 4z = -7 \\ (c) \quad &x - y + z = -2 \\ &-x + 3y - 2z = 6 \end{aligned}$$

```
a = [[2,1,-4];[1,-1,1];[-1,3,-2]];
b = [-7,-2,6];
n = 3;
solution = Gaussian_elimination(a,b,n)
```

```
solution = 3x1
   -1
    3
    2
```

## Ex 2.2

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 Exercise 2.

$$(a) \begin{bmatrix} 3 & 1 & 2 \\ 6 & 3 & 4 \\ 3 & 1 & 5 \end{bmatrix}$$

```
a = [[3,1,2];[6,3,4];[3,1,5]];
% the results of LU factorization
[L,U] = LU_factorization(a)
```

```
L = 3x3
    1     0     0
    2     1     0
    1     0     1
U = 3x3
    3     1     2
    0     1     0
    0     0     3
```

```
% verification
L*U-a
```

```
ans = 3x3
    0     0     0
    0     0     0
    0     0     0
```

$$(b) \begin{bmatrix} 4 & 2 & 0 \\ 4 & 4 & 2 \\ 2 & 2 & 3 \end{bmatrix}$$

```
a = [[4,2,0];[4,4,2];[2,2,3]];
% the results of LU factorization
[L,U] = LU_factorization(a)
```

```
L = 3x3
    1.0000000000000000     0     0
    1.0000000000000000    1.0000000000000000     0
    0.5000000000000000    0.5000000000000000    1.0000000000000000
U = 3x3
    4     2     0
    0     2     2
    0     0     2
```

```
% verification
L*U-a
```

```
ans = 3x3
    0     0     0
    0     0     0
    0     0     0
```

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

```
a = [[1,-1,1,2];[0,2,1,0];[1,3,4,4];[0,2,1,-1]];
% results of LU factorization
[L,U] = LU_factorization(a)
```

```
L = 4x4
    1     0     0     0
    0     1     0     0
    1     2     1     0
    0     1     0     1
U = 4x4
    1    -1     1     2
    0     2     1     0
    0     0     1     2
    0     0     0    -1
```

```
% verification
L*U-a
```

```
ans = 4x4
    0     0     0     0
    0     0     0     0
    0     0     0     0
    0     0     0     0
```

## Ex 2.3

For the  $n \times n$  matrix with entries  $A_{ij} = \frac{5}{(i+2j-1)}$ , set  $x = [1, \dots, 1]^T$  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)  $n = 6$

```
n = 6;
A = zeros(n,n);
for i=1:n
    for j=1:n
        A(i,j) = 5/(i+2*j-1);
    end
end
x = ones(n,1);
b = A*x;
x_c = A\b
```

```
x_c = 6x1
    1.0000
    1.0000
```

```
1.0000
1.0000
1.0000
1.0000
```

```
forward_error = norm((x_c-x),"inf");
backward_error = norm((A*x_c-b),"inf");
relative_forward_error = forward_error/norm(x,"inf");
relative_backward_error = backward_error/norm(b,"inf");
error_magnification_factor = relative_forward_error/relative_backward_error;
condition_number = norm(A,"inf")*norm(inv(A),"inf");
forward_error
```

```
forward_error = 4.9943e-11
```

```
error_magnification_factor
```

```
error_magnification_factor = 6.8883e+05
```

```
condition_number
```

```
condition_number = 7.0342e+07
```

(b)  $n = 10$

```
n = 10;
A = zeros(n,n);
for i=1:n
    for j=1:n
        A(i,j) = 5/(i+2*j-1);
    end
end
x = ones(n,1);
b = A*x;
x_c = A\b
```

```
x_c = 10x1
1.0000
1.0000
1.0000
1.0000
0.9999
1.0005
0.9991
1.0010
0.9994
1.0001
```

```
forward_error = norm((x_c-x),"inf");
backward_error = norm((A*x_c-b),"inf");
relative_forward_error = forward_error/norm(x,"inf");
relative_backward_error = backward_error/norm(b,"inf");
error_magnification_factor = relative_forward_error/relative_backward_error;
condition_number = norm(A,"inf")*norm(inv(A),"inf");
forward_error
```

forward\_error = 9.6569e-04

error\_magnification\_factor

error\_magnification\_factor = 7.9615e+12

condition\_number

condition\_number = 1.3134e+14

## Ex 2.4

Find the PA=LU factorization (using partial pivoting) of the following matrices:

$$(a) \begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix}$$

Solution:

$$\begin{aligned} \begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix} &\xrightarrow{\text{swap : } P_{213}} \begin{bmatrix} 2 & 1 & -1 \\ 1 & 1 & 0 \\ -1 & 1 & -1 \end{bmatrix} \xrightarrow{\text{elim : } L_{21}\left(-\frac{1}{2}\right), L_{31}\left(\frac{1}{2}\right)} \begin{bmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{3}{2} & -\frac{3}{2} \end{bmatrix} \xrightarrow{\text{swap : } P_{132}} \begin{bmatrix} 2 & 1 & -1 \\ 0 & \frac{3}{2} & -\frac{3}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \\ &\xrightarrow{\text{elim : } L_{32}\left(-\frac{1}{3}\right)} \begin{bmatrix} 2 & 1 & -1 \\ 0 & \frac{3}{2} & -\frac{3}{2} \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Therefore, the PA=LU factorization is

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, A = \begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix}, L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ \frac{1}{2} & \frac{1}{3} & 1 \end{bmatrix}, U = \begin{bmatrix} 2 & 1 & -1 \\ 0 & \frac{3}{2} & -\frac{3}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$(b) \begin{bmatrix} 0 & 1 & 3 \\ 2 & 1 & 1 \\ -1 & -1 & 2 \end{bmatrix}$$

Solution:

$$\begin{bmatrix} 0 & 1 & 3 \\ 2 & 1 & 1 \\ -1 & -1 & 2 \end{bmatrix} \xrightarrow{\text{swap : } P_{213}} \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 3 \\ -1 & -1 & 2 \end{bmatrix} \xrightarrow{\text{elim : } L_{31}\left(\frac{1}{2}\right)} \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 3 \\ 0 & -\frac{1}{2} & \frac{5}{2} \end{bmatrix} \xrightarrow{\text{elim : } L_{32}\left(\frac{1}{2}\right)} \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 4 \end{bmatrix}$$

Therefore, the PA=LU factorization is

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, A = \begin{bmatrix} 0 & 1 & 3 \\ 2 & 1 & 1 \\ -1 & -1 & 2 \end{bmatrix}, L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & 1 \end{bmatrix}, U = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 4 \end{bmatrix}$$

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

Solution:

$$\begin{bmatrix} 1 & 2 & -3 \\ 2 & 4 & 2 \\ -1 & 0 & 3 \end{bmatrix} \xrightarrow{\text{swap} : P_{213}} \begin{bmatrix} 2 & 4 & 2 \\ 1 & 2 & -3 \\ -1 & 0 & 3 \end{bmatrix} \xrightarrow{\text{elim} : L_{21}\left(-\frac{1}{2}\right), L_{31}\left(\frac{1}{2}\right)} \begin{bmatrix} 2 & 4 & 2 \\ 0 & 0 & -4 \\ 0 & 2 & 4 \end{bmatrix} \xrightarrow{\text{swap} : P_{132}} \begin{bmatrix} 2 & 4 & 2 \\ 0 & 2 & 4 \\ 0 & 0 & -4 \end{bmatrix}$$

Therefore, the PA=LU factorization is

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, A = \begin{bmatrix} 1 & 2 & -3 \\ 2 & 4 & 2 \\ -1 & 0 & 3 \end{bmatrix}, L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{bmatrix}, U = \begin{bmatrix} 2 & 4 & 2 \\ 0 & 2 & 4 \\ 0 & 0 & -4 \end{bmatrix}$$

$$(d) \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ -2 & 1 & 0 \end{bmatrix}$$

Solution:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ -2 & 1 & 0 \end{bmatrix} \xrightarrow{\text{swap} : P_{321}} \begin{bmatrix} -2 & 1 & 0 \\ 1 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix} \xrightarrow{\text{elim} : L_{21}\left(\frac{1}{2}\right)} \begin{bmatrix} -2 & 1 & 0 \\ 0 & \frac{1}{2} & 2 \\ 0 & 1 & 0 \end{bmatrix} \xrightarrow{\text{swap} : P_{132}} \begin{bmatrix} -2 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & 2 \end{bmatrix} \xrightarrow{\text{elim} : L_{32}\left(-\frac{1}{2}\right)} \begin{bmatrix} -2 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Therefore, the PA=LU factorization is

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ -2 & 1 & 0 \end{bmatrix}, L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}, U = \begin{bmatrix} -2 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

## Appendix

```
function [L, U] = LU_factorization(a)
L = tril(ones(size(a)));
n = size(a,1);
% forward elimination
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(i,j) = mult;
        for k=j+1:n
            a(i,k) = a(i,k)-mult*a(j,k);
        end
    end
end
U = triu(a);
end

function x = Gaussian_elimination(a, b, n)
x = zeros(n,1);
% forward elimination
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
% back substitution
for i=n:-1:1
    for j=i+1:n
        b(i) = b(i)-a(i,j)*x(j);
    end
    x(i) = b(i)/a(i,i);
end
end

```