# 《机械工程中的数值分析技术》

# 作业

学　　生：易弘睿

学　　号：20186103

专业班级：机械一班

作业编号：2021060804

重庆大学-辛辛那提大学联合学院

二〇二一年六月

# Catalog

# Lec8 Direct method for linear Equations

## 1.1 Question 9.4

**9.4** Given the system of equations

$$2x_2 + 5x_3 = 1$$
$$2x_1 + x_2 + x_3 = 1$$
$$3x_1 + x_2 = 2$$

**(a)** Compute the determinant.
**(b)** Use Cramer's rule to solve for the $x$'s.
**(c)** Use Gauss elimination with partial pivoting to solve for the $x$'s. As part of the computation, calculate the determinant in order to verify the value computed in (a)
**(d)** Substitute your results back into the original equations to check your solution.

**The Matlab code is below:**

```
close all;clc;clear

% (a) Compute the deteminant
A = [0,2,5;2,1,1;3,1,0];
detA = det(A);
fprintf('The determinant for this matrix is %d.\n',detA)

% (b) Use Cramer's rule
b = [1;1;2];
A1 = [1,2,5;1,1,1;2,1,0];
A2 = [0,1,5;2,1,1;3,2,0];
A3 = [0,2,1;2,1,1;3,1,2];
x1 = det(A1)/detA;
x2 = det(A2)/detA;
x3 = det(A3)/detA;
fprintf('The solutions obtained by Cramer rule are: x1 = %.4f, x2
= %.4f, x3 = %.4f.\n',x1,x2,x3)

%(c) Use Gauss elimination with partial pivoting
[m,n] = size(A);
nb = n + 1;
```

```
aug = [A b];
for k = 1:n-1
    [big,i] = max(abs(aug(k:n,k)));
    ipr = i+k-1;
    if ipr ~= k
        aug([k,ipr],:) = aug([ipr,k],:);
    end
    for i = k+1:n
        factor = aug(i,k)/aug(k,k);
        aug(i,k:nb) = aug(i,k:nb) - factor*aug(k,k:nb);
    end
end
x = zeros(n,1);
x(n) = aug(n,nb)/aug(n,n);
for i = n-1:-1:1
    x(i) = (aug(i,nb)-aug(i,i+1:n)*x(i+1:n))/aug(i,i);
end
fprintf('The solutions obtained by Gauss elimination are:x1 = %.4f,
x2 = %.4f, x3 = %.4f.\n', x1,x2,x3)

% (d) Substitute results back into the equation
b = zeros(n,1);
b(1,1) = 2*x(2)+5*x(3);
b(2,1) = 2*x(1)+x(2)+x(3);
b(3,1) = 3*x(1)+x(2);
fprintf('The results by substituting solutions back into the original
equations are the following:\n')
disp(b)
```

## The output is below:

The determinant for this matrix is 1.000000e+00.
The solutions obtained by Cramer rule are: x1 = -2.0000, x2 = 8.0000, x3 = -3.0000.
The solutions obtained by Gauss elimination are:x1 = -2.0000, x2 = 8.0000, x3 = -3.0000.
The results by substituting solutions back into the original equations are the following:

    1.0000
    1.0000
    2.0000

## 1.2 Question 9.13

**9.13** A stage extraction process is depicted in Fig. P9.13. In such systems, a stream containing a weight fraction $y_{in}$ of a chemical enters from the left at a mass flow rate of $F_1$. Simultaneously, a solvent carrying a weight fraction $x_{in}$ of the same chemical enters from the right at a flow rate of $F_2$. Thus, for stage $i$, a mass balance can be represented as

$$F_1 y_{i-1} + F_2 x_{i+1} = F_1 y_i + F_2 x_i \qquad \text{(P9.13}a\text{)}$$

At each stage, an equilibrium is assumed to be established between $y_i$ and $x_i$ as in

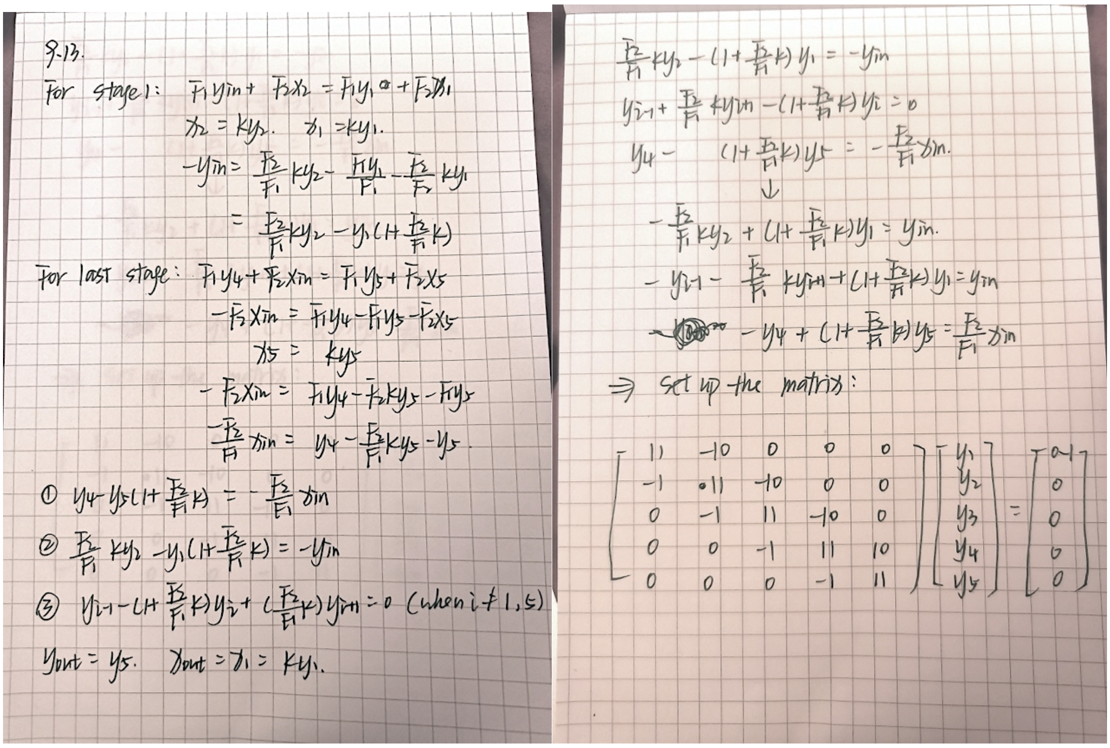
$$K = \frac{x_i}{y_i} \qquad \text{(P9.13}b\text{)}$$

where $K$ is called a distribution coefficient. Equation (P9.13$b$) can be solved for $x_i$ and substituted into Eq. (P9.13$a$) to yield

$$y_{i-1} - \left(1 + \frac{F_2}{F_1}K\right) y_i + \left(\frac{F_2}{F_1}K\right) y_{i+1} = 0 \qquad \text{(P9.13}c\text{)}$$

If $F_1 = 400$ kg/h, $y_{in} = 0.1$, $F_2 = 800$ kg/h, $x_{in} = 0$, and $K = 5$, determine the values of $y_{out}$ and $x_{out}$ if a five-stage reactor is used. Note that Eq. (P9.13$c$) must be modified to account for the inflow weight fractions when applied to the first and last stages.

**The deduction which is used to set up the matrix is below:**



9.13.

For stage 1: $F_1 y_{in} + F_2 x_2 = F_1 y_1 + F_2 x_1$

$x_2 = K y_2$. $x_1 = K y_1$.

$-y_{in} = \frac{F_2}{F_1} K y_2 - \frac{F_1 y_1}{F_1} - \frac{F_2}{F_1} K y_1$

$= \frac{F_2}{F_1} K y_2 - y_1 (1 + \frac{F_2}{F_1} K)$

For last stage: $F_1 y_4 + F_2 x_{in} = F_1 y_5 + F_2 x_5$

$-F_2 x_{in} = F_1 y_4 - F_1 y_5 - F_2 x_5$

$x_5 = K y_5$

$-F_2 x_{in} = F_1 y_4 - F_2 K y_5 - F_1 y_5$

$-\frac{F_2}{F_1} x_{in} = y_4 - \frac{F_2}{F_1} K y_5 - y_5$.

① $y_4 - y_5 (1 + \frac{F_2}{F_1} K) = -\frac{F_2}{F_1} x_{in}$

② $\frac{F_2}{F_1} K y_2 - y_1 (1 + \frac{F_2}{F_1} K) = -y_{in}$

③ $y_{i-1} - (1 + \frac{F_2}{F_1} K) y_i + (\frac{F_2}{F_1} K) y_{i+1} = 0$ (when $i \neq 1, 5$)

$y_{out} = y_5$. $x_{out} = x_1 = K y_1$.

$\frac{F_2}{F_1} K y_2 - (1 + \frac{F_2}{F_1} K) y_1 = -y_{in}$

$y_{i-1} + \frac{F_2}{F_1} K y_{i+1} - (1 + \frac{F_2}{F_1} K) y_i = 0$

$y_4 - (1 + \frac{F_2}{F_1} K) y_5 = -\frac{F_2}{F_1} x_{in}$.

$\downarrow$

$-\frac{F_2}{F_1} K y_2 + (1 + \frac{F_2}{F_1} K) y_1 = y_{in}$.

$-y_{i-1} - \frac{F_2}{F_1} K y_{i+1} + (1 + \frac{F_2}{F_1} K) y_i = y_{in}$

$-y_4 + (1 + \frac{F_2}{F_1} K) y_5 = \frac{F_2}{F_1} x_{in}$

$\Rightarrow$ set up the matrix:

$$\begin{bmatrix} 11 & -10 & 0 & 0 & 0 \\ -1 & 11 & -10 & 0 & 0 \\ 0 & -1 & 11 & -10 & 0 \\ 0 & 0 & -1 & 11 & -10 \\ 0 & 0 & 0 & -1 & 11 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

## The Matlab code is below:

```
clear;close all;clc
F1 = 400;
y_in = 0.1;
F2 = 800;
x_in = 0;
K = 5;

A = [11,-10,0,0,0;-1,11,-10,0,0;0,-1,11,-10,0;0,0,-1,11,-10;0,0,0,-
1,11];
b = [0.1;0;0;0;0];
x_exact = A\b;  % exact solution
% Solution for a tridiagonal system
r = [0.1;0;0;0;0];
f = [11,11,11,11,11];
e = [0,-1,-1,-1,-1];
g = [-10,-10,-10,-10,0];
n = length(f);
for k = 2:n
    factor = e(k)/f(k-1);
    f(k) = f(k) - factor*g(k-1);
    r(k) = r(k) - factor*r(k-1);
end
% back substitution
x(n) = r(n)/f(n);
```

```
for k = n-1:-1:1
    x(k) = (r(k)-g(k)*x(k+1))/f(k);
end
y_out = x(5);
x_out = K*x(1);
fprintf('The values of y_out and x_out are %.7f
and %.7f',y_out,x_out)
```

### The output is below:

The values of y_out and x_out are 0.0000009 and 0.050000

## 1.3 Question 9.16

**9.16** A *pentadiagonal* system with a bandwidth of five can be expressed generally as

$$\begin{bmatrix} f_1 & g_1 & h_1 & & & & & \\ e_2 & f_2 & g_2 & h_2 & & & & \\ d_3 & e_3 & f_3 & g_3 & h_3 & & & \\ & & \cdot & \cdot & \cdot & & & \\ & & & \cdot & \cdot & \cdot & & \\ & & & & \cdot & \cdot & \cdot & \\ & & & & d_{n-1} & e_{n-1} & f_{n-1} & g_{n-1} \\ & & & & & d_n & e_n & f_n \end{bmatrix} \times \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ \cdot \\ \cdot \\ x_{n-1} \\ x_n \end{Bmatrix} = \begin{Bmatrix} r_1 \\ r_2 \\ r_3 \\ \cdot \\ \cdot \\ \cdot \\ r_{n-1} \\ r_n \end{Bmatrix}$$

Develop an M-file to efficiently solve such systems without pivoting in a similar fashion to the algorithm used for tridiagonal matrices in Sec. 9.4.1. Test it for the following case:

$$\begin{bmatrix} 8 & -2 & -1 & 0 & 0 \\ -2 & 9 & -4 & -1 & 0 \\ -1 & -3 & 7 & -1 & -2 \\ 0 & -4 & -2 & 12 & -5 \\ 0 & 0 & -7 & -3 & 15 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = \begin{Bmatrix} 5 \\ 2 \\ 1 \\ 1 \\ 5 \end{Bmatrix}$$

### The Matlab code is below:

```
clear;close all;clc
A = [8,-2,-1,0,0;-2,9,-4,-1,0;-1,-3,7,-1,-2;0,-4,-2,12,-5;0,0,-7,-
3,15];
b = [5;2;1;1;5];
P9_16(A,b);
ans
function x = P9_16(A,b)
    % A = pentadiagonal matrix
    % b = right hand side vector
```

```
% x = solution of the system
[m,n] = size(A);
if m ~= n
    error('Not a suqare matrix.')
end
if length(b) ~= m
    error('Number of rows doesn not match.')
end

d = [0;0;diag(A,-2)];
e = [0;diag(A,-1)];
f = diag(A);
g = diag(A,1);
h = diag(A,2);

omiga = zeros(n,1);
beta = zeros(n-1,1);
gamma = zeros(n-2,1);
epsilon = zeros(n,1);
a = zeros(n,1);

% Solve for each band
% Decomposition
omiga(1) = f(1);
beta(1) = g(1)/omiga(1);
gamma(1) = h(1)/omiga(1);
epsilon(2) = e(2);
omiga(2) = f(2)-epsilon(2)*beta(1);
beta(2) = (g(2)-epsilon(2)*gamma(1))/omiga(2);
gamma(2) = h(2)/omiga(2);

for k = 3:n-2
    epsilon(k) = e(k)-d(k)*beta(k-2);
    omiga(k) = f(k)-d(k)*gamma(k-2)-epsilon(k)*beta(k-1);
    beta(k) = (g(k)-epsilon(k)*gamma(k-1))/omiga(k);
    gamma(k) = h(k)/omiga(k);
end

epsilon(n-1) = e(n-1)-d(n-1)*beta(n-3);
omiga(n-1) = f(n-1)-d(n-1)*gamma(n-3)-epsilon(n-1)*beta(n-2);
beta(n-1) = (g(n-1)-epsilon(n-1)*gamma(n-2))/omiga(n-1);
epsilon(n) = e(n) - d(n)*beta(n-2);
omiga(n) = f(n)-d(n)*gamma(n-2)-epsilon(n)*beta(n-1);
```

```
    % Forward substitution
    a(1) = b(1)/omiga(1);
    a(2) = (b(2)-epsilon(2)*a(1))/omiga(2);
    for k = 3:n
        a(k) = (b(k)-d(k)*a(k-2)-epsilon(k)*a(k-1))/omiga(k);
    end
    % Back substitution
    x(n) = a(n);
    x(n-1) = a(n-1)-beta(n-1)*x(n);
    for k=n-2:-1:1
        x(k) = a(k)-beta(k)*x(k+1)-gamma(k)*x(k+2);
    end
end
```

**The output is below:**

ans =

    1.0825    1.1759    1.3082    1.1854    1.1809

# Lec.10 LU decomposition method for Linear Equations

## 2.1 Question 10.3

**10.3** Use naive Gauss elimination to factor the following system according to the description in Section 10.2:

$$\begin{aligned} 7x_1 + 2x_2 - 3x_3 &= -12 \\ 2x_1 + 5x_2 - 3x_3 &= -20 \\ x_1 - \phantom{2}x_2 - 6x_3 &= -26 \end{aligned}$$

Then, multiply the resulting $[L]$ and $[U]$ matrices to determine that $[A]$ is produced.

**The Matlab code is below:**

```
clear;clc;close all
```

```
% define matrix
A = [7,2,-3;2,5,-3;1,-1,-6];
b = [-12;-20;-26];
% get aug matrix
[m,n] = size(A);
nb = n + 1;
aug = [A b];
L = eye(n);
% Gauss naive elimination
for k = 1:n-1
    for i = k+1:n
        factor = aug(i,k)/aug(k,k);
        aug(i,k:nb) = aug(i,k:nb) - factor*aug(k,k:nb);

        % Get the lower matrix
        if k == 1
            lower_factor = A(i,k)/A(k,k);
            L(i,k) = lower_factor;
        else
            lower_factor = (A(i,k)-A(1,k)*(A(i,1)/A(1,1)))/aug(k,k);
            L(i,k) = lower_factor;
        end
    end
end
U = aug(:,1:n); % Get the upper matrix
fprintf('The upper matrix is:\n')
disp(U)

% Get the lower matrix
% factor21 = A(2,1)/A(1,1);
% factor31 = A(3,1)/A(1,1);
% factor32 = (A(3,2)-A(1,2)*(A(3,1)/A(1,1)))/aug(2,2);
% L = eye(n);
% L(2,1) = factor21;
% L(3,1) = factor31;
% L(3,2) = factor32;
fprintf('The lower matrix is:\n')
disp(L)

result = L*U;
fprintf('The resulting matrix is:\n')
disp(result)
```

**The output is below:**

The upper matrix is:

| | | |
|---|---|---|
| 7.0000 | 2.0000 | -3.0000 |
| 0 | 4.4286 | -2.1429 |
| 0 | 0 | -6.1935 |

The lower matrix is:

| | | |
|---|---|---|
| 1.0000 | 0 | 0 |
| 0.2857 | 1.0000 | 0 |
| 0.1429 | -0.2903 | 1.0000 |

The resulting matrix is:

| | | |
|---|---|---|
| 7.0000 | 2.0000 | -3.0000 |
| 2.0000 | 5.0000 | -3.0000 |
| 1.0000 | -1.0000 | -6.0000 |

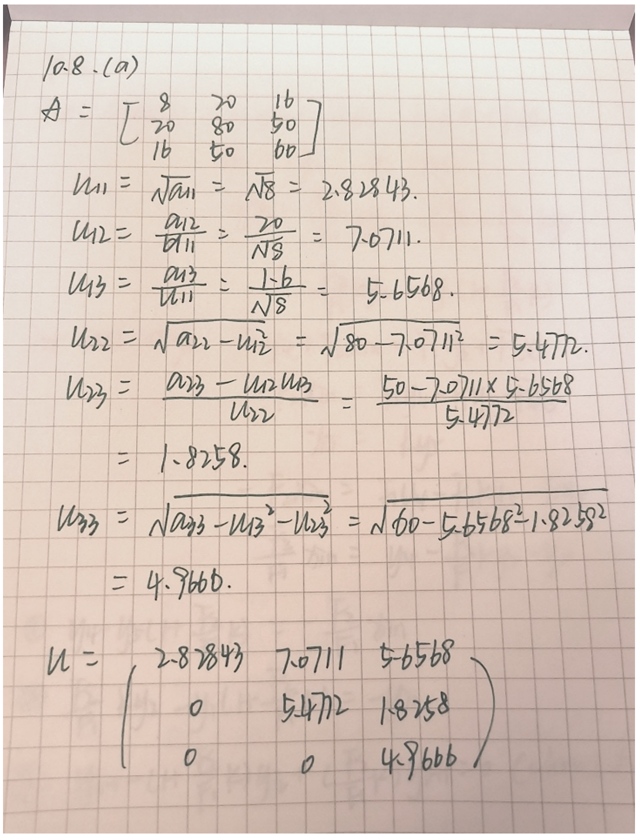
## 2.2 Question 10.8

**10.8** **(a)** Perform a Cholesky factorization of the following symmetric system by hand:

$$\begin{bmatrix} 8 & 20 & 16 \\ 20 & 80 & 50 \\ 16 & 50 & 60 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 100 \\ 250 \\ 100 \end{Bmatrix}$$

**(b)** Verify your hand calculation with the built-in `chol` function. **(c)** Employ the results of the factorization $[U]$ to determine the solution for the right-hand-side vector.

a) **The calculation process is below:**



10.8.(a)

$A = \begin{bmatrix} 8 & 20 & 16 \\ 20 & 80 & 50 \\ 16 & 50 & 60 \end{bmatrix}$

$u_{11} = \sqrt{a_{11}} = \sqrt{8} = 2.82843.$

$u_{12} = \frac{a_{12}}{u_{11}} = \frac{20}{\sqrt{8}} = 7.0711.$

$u_{13} = \frac{a_{13}}{u_{11}} = \frac{16}{\sqrt{8}} = 5.6568.$

$u_{22} = \sqrt{a_{22} - u_{12}^2} = \sqrt{80 - 7.0711^2} = 5.4772.$

$u_{23} = \frac{a_{23} - u_{12}u_{13}}{u_{22}} = \frac{50 - 7.0711 \times 5.6568}{5.4772}$

$= 1.8258.$

$u_{33} = \sqrt{a_{33} - u_{13}^2 - u_{23}^2} = \sqrt{60 - 5.6568^2 - 1.8258^2}$

$= 4.9666.$

$u = \begin{pmatrix} 2.82843 & 7.0711 & 5.6568 \\ 0 & 5.4772 & 1.8258 \\ 0 & 0 & 4.9666 \end{pmatrix}$

## b) The Matlab code is below:

```
clear;close all;clc
A = [8,20,16;20,80,50;16,50,60];
U = chol(A);
disp (U);
```

**The output is below:**

```
2.8284    7.0711    5.6569
     0    5.4772    1.8257
     0         0    4.9666
```

## c) The Matlab code is below:

```
clear;close all;clc
% find the upper matrix using cholesky factorization
A = [8,20,16;20,80,50;16,50,60];
U = chol(A);

% determine thesolution for the vector
b = [100;250;100];
d = U'\b;
x = U\d;
fprintf('The solution is x1 = %.4f, x2 = %.4f, x3
```

```
= %.4f.',x(1),x(2),x(3))
```

## The output is below:

The solution is x1 = 17.2297, x2 = 1.3514, x3 = -4.0541.