KING SAUD UNIVERSITY

CHE401: Numerical Methods In Chemical Engineering

(Week 4 Lecture)

Continued

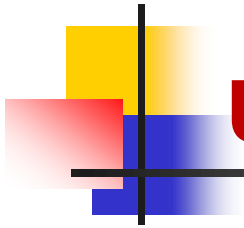
Solution of System of Linear Equations

Using Matlab

1.Preliminary remarks :

In general, when dealing with chemical engineering problems, the students must decrypt a given situation by :

- Identifying all parameters related to a given problem (known and unknowns).
- Drawing diagrams that describe the problem.
- Laying out all pertinent equations, applicable balances, and correlations.
- Obtaining the set of linear equations to solve; reduce this set to exactly n-independent equations and m-unknowns; (note that when n=m one unique solution exists, when n<m the problem is under defined, and if n>m the problem is over specified).



Using Matlab

- Solving the obtained system using a computer tool (here Matlab) Generally, a set of linear equations is written as :

Using Matlab

In a matrix form, the above system may re-written as :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

In a vector form:

$$\underline{\boldsymbol{A}} \cdot \underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}$$

Note that for matrix operations compatibility, when A is an $(m \times n)$, x is an $(n \times 1)$, and b is an $(m \times 1)$.

Using Matlab

b_1

Mathematically, the solution would be (when A is inversible) :

$$\underline{\boldsymbol{x}} = \boldsymbol{A}^{-1} \cdot \underline{\boldsymbol{b}}$$

Note that **conditioning of the system** should be checked by computing the conditioning number (**CN**) of the matrix A; when cond(A) is high (A maybe singular, i.e. A is not inversible) re-arrange the problem to avoid such a drawback by reformulating the problem (re-think the situation at hand reasonably !)

Using Matlab

b_1

- **Matlab solving procedure :**

Case 1: using the line-by-line commands within the Matlab's "commands window"

1. Define the matrix A by : (within brackets and separating matrix lines by semi-column)

Using Matlab

>> A=[a_{11} a_{12} …a_{1n} ; a_{21} a_{22} …a_{2n} ; … ; a_{m1} a_{m2} …a_{mn}];

(then hit enter ! with the use of the symbol ";" to avoid displaying the result on the screen)

2. Define the vector b : (as a 1-column matrix by using a semi-column between elements of the vector)

>> b=[b_1 ; b_2 ; … ; b_m]; *(then hit enter !)*

3. Next calculate the solution : (the back-slash is a Matlab operator that solves when A is not necessarily a square matrix (m≠n))

>> x=**A\b** *(then hit enter ! without the use of the symbol ";" to display the result)*

Using Matlab

Note that when A is square (i.e. m=n) then the syntax would be

```
>> x=inv(A)*b
```

where the command "**inv(A)**" is a call for the built-in function "**inv**" of Matlab which implements matrix inversion procedure.

A=[a_{11} a_{12} …a_{1n} ; a_{21} a_{22} …a_{2n} ; … ; a_{m1} a_{m2} …a_{mn}];

b=[b_1 ; b_2 ; … ; b_m];

cond(A)

rank(A)

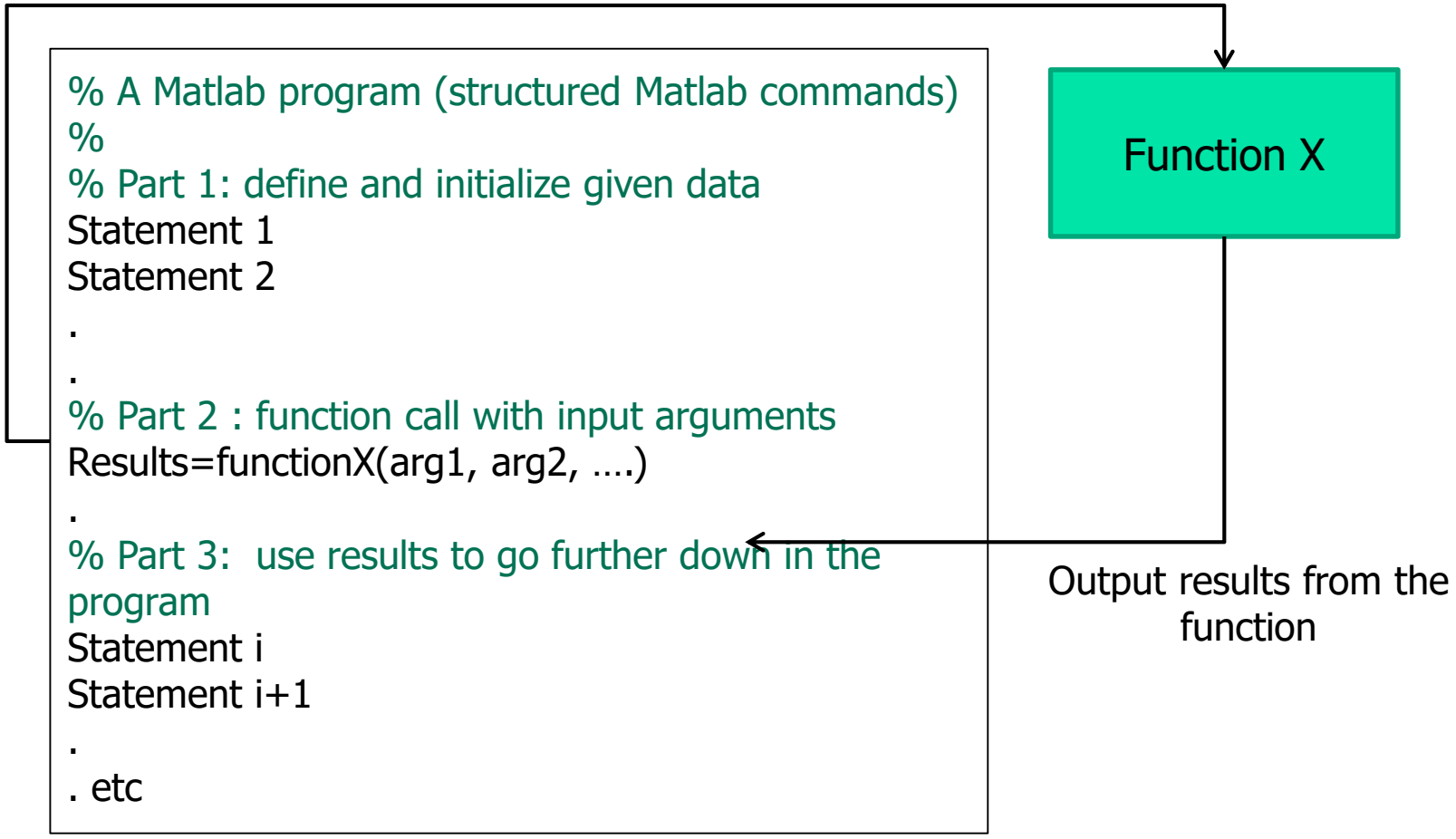
% if cond(A) is high, then re-formulate the problem (if possible) until obtaining a well conditioned problem! If rank gives lower number of independent equations rethink the problem to obtain this exact number of equations

x=**A\b or** x=**inv(A)*b**

Using Matlab

1. Step 1: writing the "main program"

(note that the green characters are comments in Matlab when preceeded by a %)



Using Matlab

1. Step 1: writing the "function" (continued)

- The first line always starts with the word "function" in small letters

```
function Results=functionX(arg1, arg2, ….)
%
% Perform one or more specific computations

Statement 1
Statement 2
      :
      :
return
```

- Note that for the function to work correctly you should save it as

functionX.m

example 1 : solve $\sqrt{2x} - x = 0$

```
% to avoid these iterations one could write a 'while' loop as follows :

xold=0.1; toll=1e-4; err=toll+1;
while err > toll
    x=xold;
    xnew=fss1(x);
    err=abs(xnew-xold);
    xold=xnew;
end
xsol=xnew    % display the final solution
```

```
function y = fss1(x)
y = sqrt(2*x);
```

```
>>
xsol =
    1.9999
```

one or more commands can be executed by selecting them then hitting the F9 button !

also, one could save the file under a name (file0.m) then entering the command be >> **file0**

Example 1 : Consider the following set of equations

$$5x_1 - x_2 + x_3 = 10$$

$$2x_1 + 4x_2 = 12$$

$$x_1 + x_2 + 5x_3 = -1$$

(a) write the system in matrix form

(b) Use Matlab to solve it :

- Define all parameters (A and b)
- Solve for x

(c) Check for conditioning by introducing a 1% and then 2% increases in the vector b. And also using the command "cond()"

Answer:

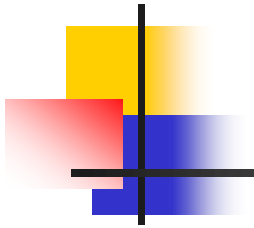
(a) Writing the system in a matrix form allows to quickly identify the parameters A and b :

$$\begin{bmatrix} 5 & -1 & 1 \\ 2 & 4 & 0 \\ 1 & 1 & 5 \end{bmatrix} \begin{bmatrix} X1 \\ X2 \\ x2 \end{bmatrix} = \begin{matrix} 10 \\ 12 \\ -1 \end{matrix}$$

$$A \quad x = b$$

(b) Now enter few lines of commands to obtain the solution

1st define A and b as follows



>> A=[5 -1 1 ; 2 4 0 ; 1 1 5]

A=

5	−1	1
2	4	0
1	1	5

>> cond(A)

ans =

1.5581

(Note that this value is low so could conditioning is expected for this example)

>> b=[10 ; 12 ; -1]

b=

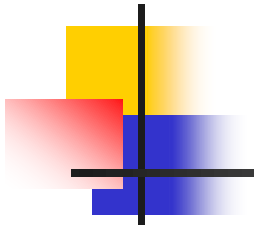
10
12
−1

Secondly, calculate the solution

>> x=**A\b** *or* >> x=**inv(A)*b**

x =

2.5556
1.7222
-1.0556



Finally, one can verify the solution by:

```
>> A*x-b
```

(c) Check for conditioning of the problem

```
% introduce a 1% change in the input (b)
>> b=b*1.01;
>> x2=A\b ;
>> err=abs(x2-x)/x
err =
   0.0100
   0.0100
  -0.0100
```

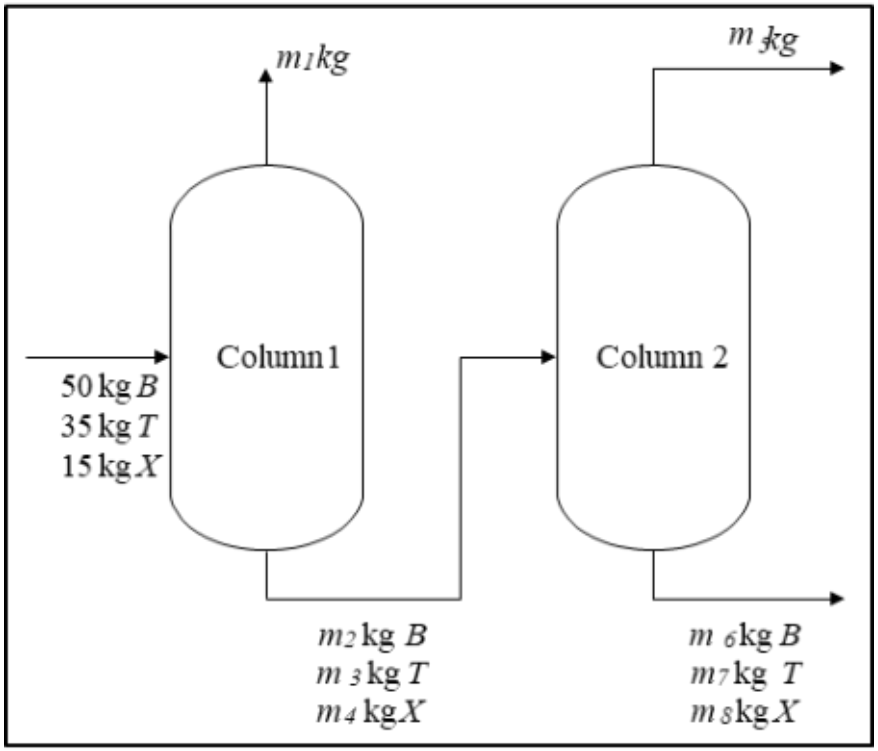
```
% now introduce one more 1% change in the
input (b)
>> b=b*1.01;
>> x2=A\b ;
>> err=abs(x2-x)/x
err =  0.0201
       0.0201
      -0.0201
```

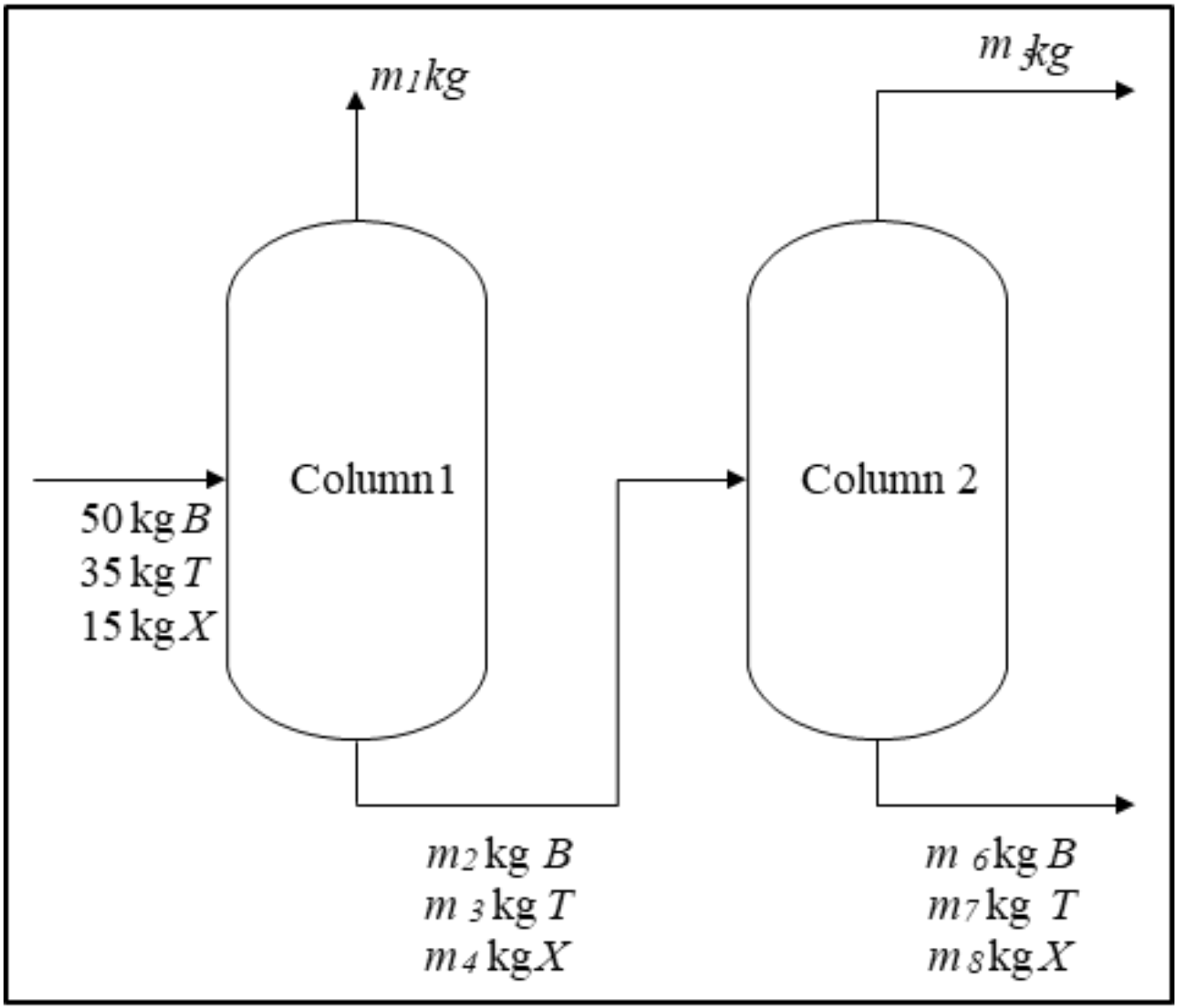
So, a 2% perturbation in the input data, causes a 2% deviation from the previous result

➔ The problem is well conditioned ; this can be confirmed by executing >> cond(A)

Example 2 : (Pb 2.2 Tutorial # 2)

A 100kg stream containing 50wt% Benzene, 35% Toluene, and the rest (Xylene) is fed to a distillation column (1). The overhead product from column 1 contains 91.4 wt% Benzene and 8.3% Toluene. The bottom product is fed to a second column. The overhead product from the second column contains 4.4% Benzene and 92.66% Toluene. Of the Toluene fed to the process, 10% is recovered in the bottom




$m_1$kg
$m_5$kg
Column 1
Column 2
50 kg B
35 kg T
15 kg X
$m_2$kg B
$m_3$kg T
$m_4$kg X
$m_6$kg B
$m_7$kg T
$m_8$kg X

(a) Layout the overall system of equations that describes the problem ?

Benzene: $50 = 0.914\ m_1 + m_2$

Toluene: $35 = 0.083\ m_1 + m_3$

Xylene: $15 = 0.003\ m_1 + m_4$

Benzene: $m_2 = 0.044\ m_5 + m_6$

Toluene: $m_3 = 0.9266\ m_5 + m_7$

Xylene: $m_4 = (1-0.9266-0.044)\ m_5 + m_8$

10% Toluene recovery: $m_7 = 0.1\ (35) = 3.5$

93.3% Xylene recovery: $m_8 = 0.933\ (15) = 14$

Then reduce the above system to only independent equations and independent unknowns.

(b)Layout this system in a matrix form.

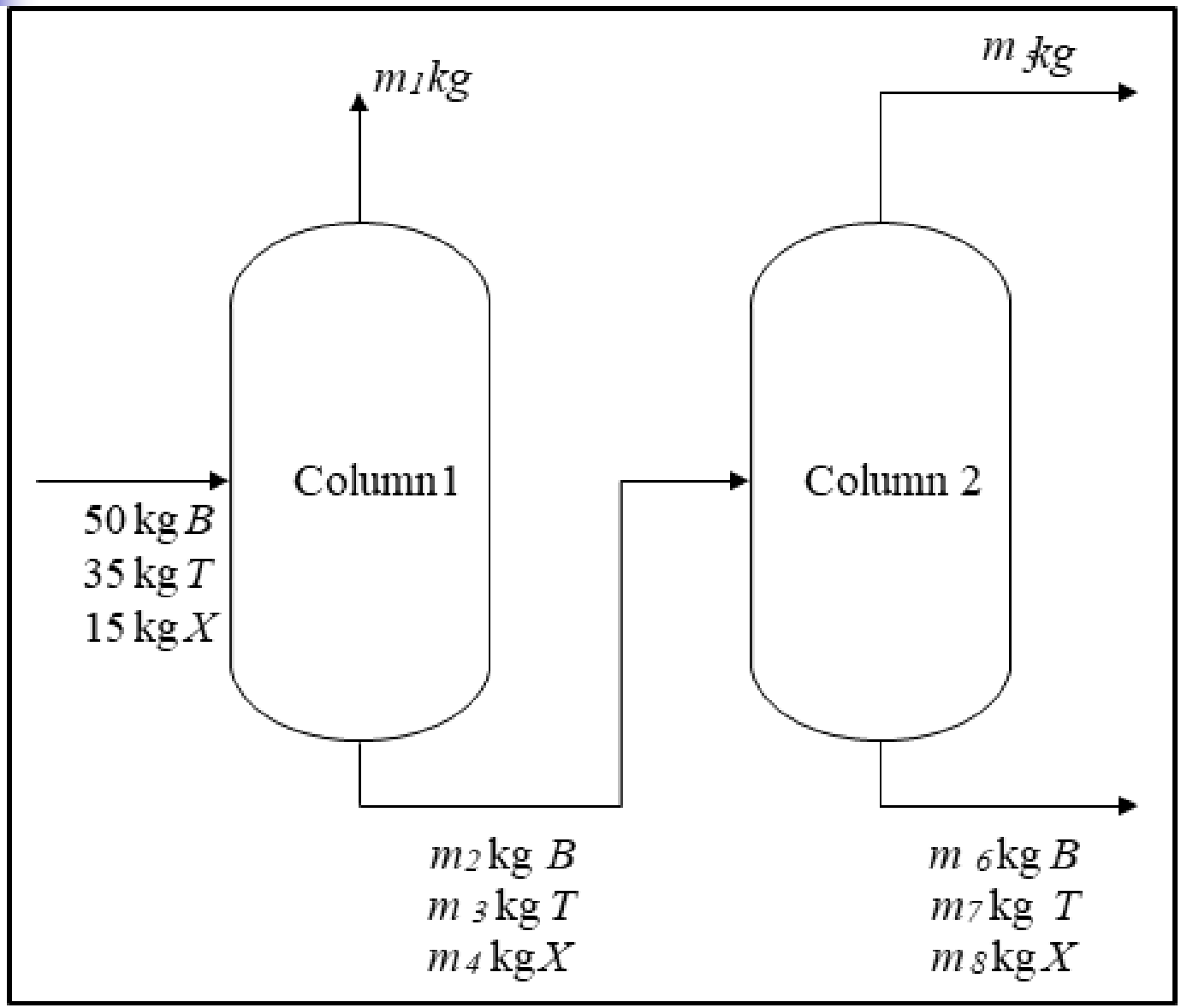
(c)Use Matlab to define all parameters and solve the system of equations. Comment on the result ?

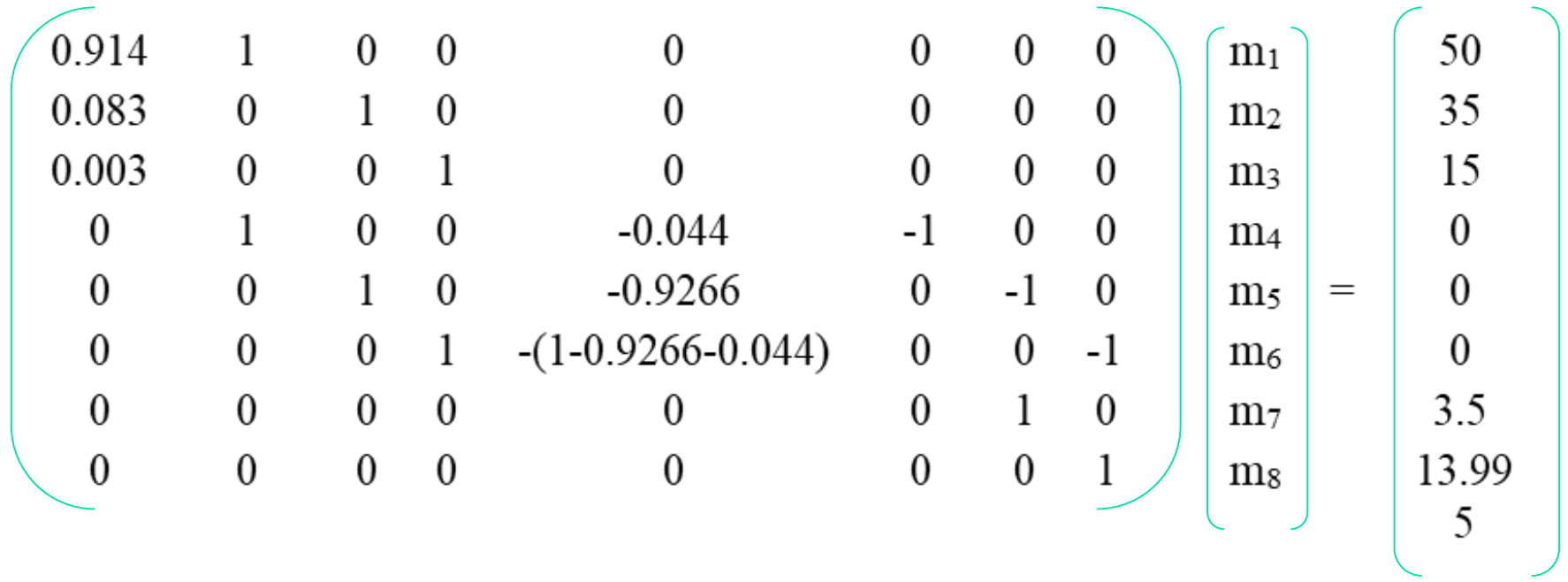
(d)Use the following matrix commands "cond(A)" and "rank(A)" to check for conditioning and number of independent equations of the problem.

Re-think the problem if conditioning is "bad"!

(Reduce the size of the system; obtain a system composed only of independent equations)

(e)Comment !


$m_1$kg
$m_5$kg
Column 1
Column 2
50 kg B
35 kg T
15 kg X
$m_2$kg B
$m_3$kg T
$m_4$kg X
$m_6$kg B
$m_7$kg T
$m_8$kg X


0.914 1 0 0 0 0 0 0
0.083 0 1 0 0 0 0 0
0.003 0 0 1 0 0 0 0
0 1 0 0 -0.044 -1 0 0
0 0 1 0 -0.9266 0 -1 0
0 0 0 1 -(1-0.9266-0.044) 0 0 -1
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1
m1 m2 m3 m4 m5 m6 m7 m8
=
50 35 15 0 0 0 3.5 13.995

(c) Using Matlab we get (line-by-line commands within the ‘‘commands window’’

1st define A and B as follows

```
>> A=[0.914,1,0,0,0,0,0,0,0;0.083,0,1,0,0,0,0,0,0;(1-0.914-0.083),0,0,1,0,0,0,0,0;0,1,0,0,-0.044,-
1,0,0;0,0,1,0,-0.9266,0,-1,0;0,0,0,1,-(1-0.9266-0.044),0,0,-1;0,0,0,0,0,0,0,1,0;0,0,0,0,0,0,0,0,1] ;
```

```
>> B=[50;35;15;0;0;0;3.5;13.995] ;
```

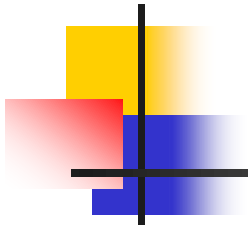
Next compute the solution by :

```
>> m=A\B or >> m=inv(A)*B
m' =
15.1148  36.1850  33.7455  14.9547  32.6413  34.7488  3.5000 13.9950
```

(d) Study conditioning of the problem

```
% compute conditioning of the matrix A
>> cond(A)
ans =

   1.4735e+04
% This high value indicates bad conditioning of the problem so we should re-think the
% problem
% to confirm bad conditioning let us make minor changes in input B for example
% 49 replaces 50, 35.5 replaces 35 and 15.5 replaces 15
>> BB=[49;35.5;15.5;0;0;0;0.1*35.5;.933*15.5];

% the new solution becomes
>> m_new=A\BB
m_new =
    67.5621  -12.7518   29.8923   15.2973   28.4290  -14.0027    3.5500   14.4615

% note that the new result contains negative elements (masses) which is not valid solutions !
% the problem should be analyzed further to avoid bad conditioning and will be discussed
% during lab
```

```
% Define the coefficient matrix A
A = [0.914 1 0 0 0 0 0 0;
     0.083 0 1 0 0 0 0 0;
     0.003 0 0 1 0 0 0 0;
     0 1 0 0 -0.044 -1 0 0;
     0 0 1 0 -0.9266 0 -1 0;
     0 0 0 1 -(1-0.9266-0.044) 0 0 -1;
     0 0 0 0 0 0 1 0;
     0 0 0 0 0 0 0 1]

% Define the vector of constants b
b = [ 50; 35; 15; 0; 0; 0; 3.5; 0.933*15 ]
Cond = cond(A)
Rank = rank(A)
m = A\b

% introduce a small perturbation into b
% by replacing 50 to 49, 35 to 35.5 and 15 to 15.5 and check output
bb = [ 49; 35.5; 15.5; 0; 0; 0; 3.55; 0.933*15.5 ]
Cond = cond(A)
Rank = rank(A)
m_new = A\bb
% Notice the negative floows in the answer111 Does it make sense?
```

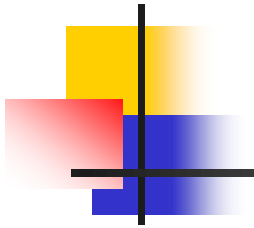


Cond = 1.4735e+04

Rank = 8

m =

15.1148
36.1850
33.7455
14.9547
32.6413
34.7488
3.5000
13.9950



Cond = 1.4735e+04

Rank = 8

m_new =

67.5621
-12.7518
29.8923
15.2973
28.4290
-14.0027
3.5500
14.4615