KING SAUD UNIVERSITY

CHE406: Numerical Methods In Chemical Engineering

(Week 8-10 Lecture)

Numerical solutions for Systems of non-linear equations

"The fsolve moethod"

Newton's Method for Simultaneous Nonlinear Equations

Suppose we need to solve $f_1(x_1, x_2)=0$, $f_2(x_1, x_2)=0$

$$f_1(x_1,x_2) = f_1(x_1^{(1)},x_2^{(1)}) + \left.\frac{\partial f_1}{\partial x_1}\right|_{x^{(1)}} (x_1 - x_1^{(1)}) + \left.\frac{\partial f_1}{\partial x_2}\right|_{x^{(1)}} (x_2 - x_2^{(1)}) + \ldots$$

$$f_2(x_1,x_2) = f_2(x_1^{(1)},x_2^{(1)}) + \left.\frac{\partial f_2}{\partial x_1}\right|_{x^{(1)}} (x_1 - x_1^{(1)}) + \left.\frac{\partial f_2}{\partial x_2}\right|_{x^{(1)}} (x_2 - x_2^{(1)}) + \ldots$$

$$\left.\frac{\partial f_1}{\partial x_1}\right|_{x^{(1)}} (x_1 - x_1^{(1)}) + \left.\frac{\partial f_1}{\partial x_2}\right|_{x^{(1)}} (x_2 - x_2^{(1)}) = -f_1(x_1^{(1)},x_2^{(1)})$$

$$\left.\frac{\partial f_2}{\partial x_1}\right|_{x^{(1)}} (x_1 - x_1^{(1)}) + \left.\frac{\partial f_2}{\partial x_2}\right|_{x^{(1)}} (x_2 - x_2^{(1)}) = -f_2(x_1^{(1)},x_2^{(1)})$$

Newton’s Method for Simultaneous Nonlinear Equations

$$\delta_1^{(1)} = x_1 - x_1^{(1)} \qquad \delta_2^{(1)} = x_2 - x_2^{(1)}$$

$$\left.\frac{\partial f_1}{\partial x_1}\right|_{x^{(1)}} \delta_1^{(1)} + \left.\frac{\partial f_1}{\partial x_2}\right|_{x^{(1)}} \delta_2^{(1)} = -f_1(x_1^{(1)}, x_2^{(1)})$$

$$\left.\frac{\partial f_2}{\partial x_1}\right|_{x^{(1)}} \delta_1^{(1)} + \left.\frac{\partial f_2}{\partial x_2}\right|_{x^{(1)}} \delta_2^{(1)} = -f_2(x_1^{(1)}, x_2^{(1)})$$

$$\begin{bmatrix} \left.\frac{\partial f_1}{\partial x_1}\right|_{x^{(1)}} & \left.\frac{\partial f_1}{\partial x_2}\right|_{x^{(1)}} \\ \left.\frac{\partial f_2}{\partial x_1}\right|_{x^{(1)}} & \left.\frac{\partial f_2}{\partial x_2}\right|_{x^{(1)}} \end{bmatrix} \begin{bmatrix} \delta_1^{(1)} \\ \delta_2^{(1)} \end{bmatrix} = -\begin{bmatrix} f_1^{(1)} \\ f_2^{(1)} \end{bmatrix}$$

Newton’s Method for Simultaneous Nonlinear Equations

Hand calculation is possible, but when the size of the system increases this becomes unpractical (a programming software is thus needed

$$\delta_1^{(1)} = \frac{\left[f_1 \frac{\partial f_2}{\partial x_2} - f_2 \frac{\partial f_1}{\partial x_2} \right]}{\left[\frac{\partial f_1}{\partial x_1} \frac{\partial f_2}{\partial x_2} - \frac{\partial f_2}{\partial x_1} \frac{\partial f_1}{\partial x_2} \right]}$$

$$\delta_2^{(1)} = \frac{\left[f_2 \frac{\partial f_1}{\partial x_1} - f_1 \frac{\partial f_2}{\partial x_1} \right]}{\left[\frac{\partial f_1}{\partial x_1} \frac{\partial f_2}{\partial x_2} - \frac{\partial f_2}{\partial x_1} \frac{\partial f_1}{\partial x_2} \right]}$$

$$\Longrightarrow x_i^{(n+1)} = x_i^{(n)} + \delta_i^{(n)}$$

Newton's Method for Simultaneous Nonlinear Equations

$f_1(x_1, \ldots x_k)=0$

……..

$f_k(x_1, \ldots. x_k)=0$

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_k} \\ & \cdots\cdots & \\ \frac{\partial f_k}{\partial x_1} & \cdots & \frac{\partial f_k}{\partial x_k} \end{bmatrix} \begin{bmatrix} \delta_1 \\ . \\ . \\ . \\ \delta_k \end{bmatrix} = - \begin{bmatrix} f_1 \\ . \\ . \\ . \\ f_k \end{bmatrix}$$

$$\Rightarrow \mathbf{J} \,.\, \boldsymbol{\delta} = -\mathbf{f}$$

Newton's Method for Simultaneous Nonlinear Equations

- Where **J** is the Jacobian matrix, **δ** is the correction vector, and **f** is the vector functions.
- After solving for **δ** one can obtain the new estimate by :

$$x^{(n+1)} = x^{(n)} + \delta^{(n)}$$

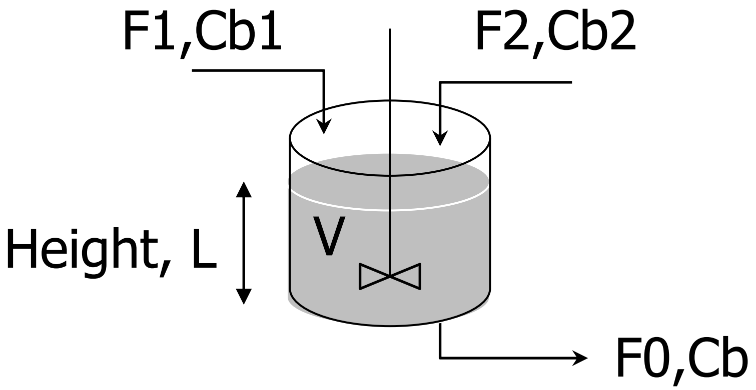
Note : strongly nonlinear equations likely to diverge, therefore *relaxation* is generally used to stabilize the iterative process (ρ varies between 0 and 1
typically ρ ~ 0.5):

$$x^{(n+1)} = x^{(n)} + \rho\delta$$

Illustrative example

It is desired to estimate the steady state values of the height (L) of the solution in CSTR and the final concentration (Cb)

- → **Use the overall conservation of mass and component balances to obtain a system of two equations as a function of L and Cb**
- → **Write the steady state system of equations and solve**



Illustrative example

- Since you have not studied reaction engineering yet, the pertaining equations will be given here without going in depth in deriving them.

Illustrative example

The system of equations is as follow

$$A\frac{dL}{dt} = F_1 + F_2 - \alpha\sqrt{L}$$

$$\frac{d(C_B)}{dt} = \frac{F_1}{AL}(C_{B1} - C_B) + \frac{F_2}{AL}(C_{B2} - C_B) - \frac{k_1 C_B}{(1 + k_2 C_B)^2}$$

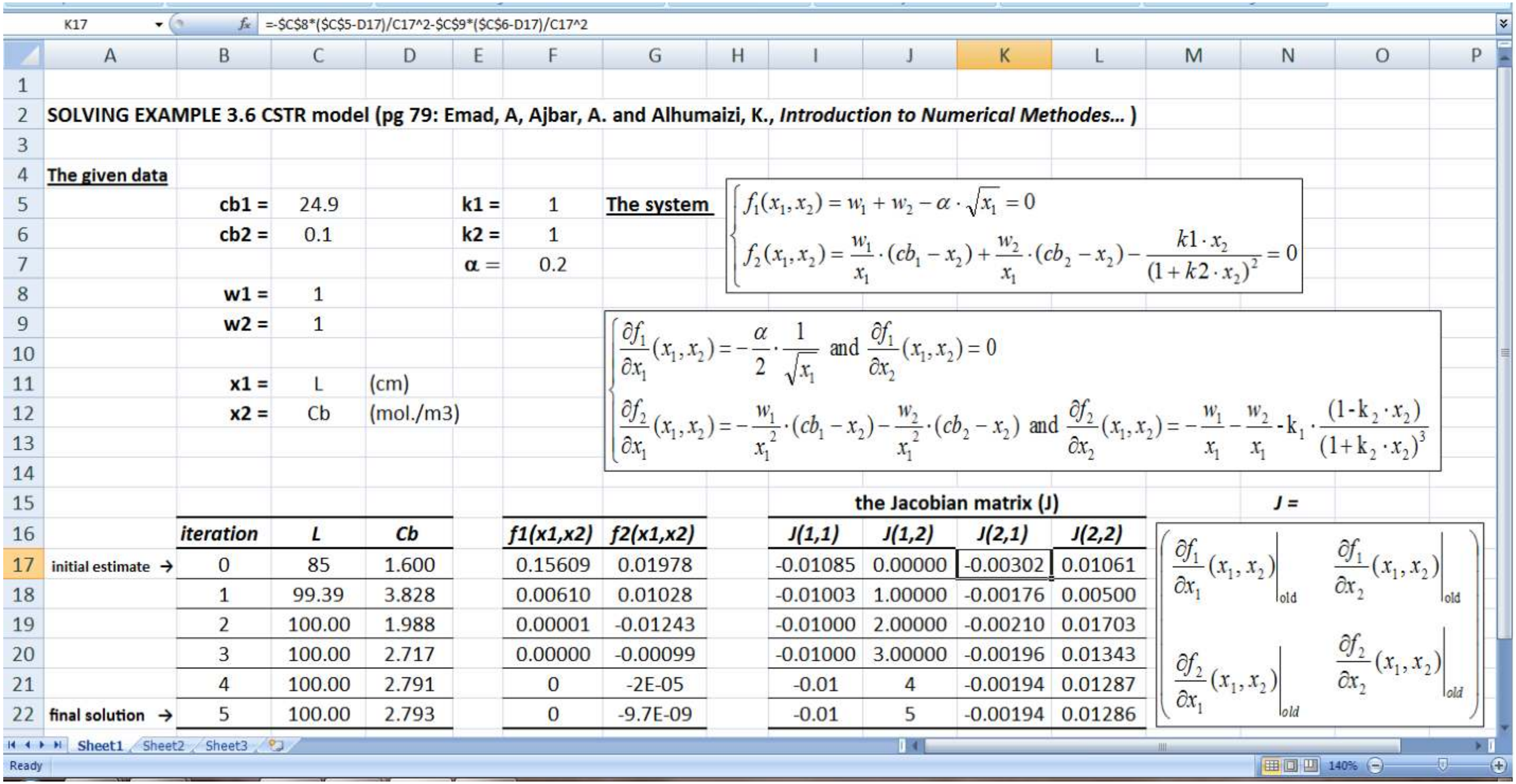
At steady state, the system is reduced to:

$$0 = F_1 + F_2 - \alpha\sqrt{L}$$

$$0 = \frac{F_1}{AL}(C_{B1} - C_B) + \frac{F_2}{AL}(C_{B2} - C_B) - \frac{k_1 C_B}{(1 + k_2 C_B)^2}$$

Implementing Newton’s Method for Simultaneous Nonlinear Equations using computer tools

For quick and simple way to solve theses types of equations, one could use : Excel or Matlab.



=-C8*(C5-D17)/C17^2-C9*(C6-D17)/C17^2

SOLVING EXAMPLE 3.6 CSTR model (pg 79: Emad, A, Ajbar, A. and Alhumaizi, K., *Introduction to Numerical Methodes…*)

The given data

cb1 = 24.9
cb2 = 0.1
w1 = 1
w2 = 1
x1 = L (cm)
x2 = Cb (mol./m3)

k1 = 1
k2 = 1
α = 0.2

The system

$$\begin{cases} f_1(x_1,x_2) = w_1 + w_2 - \alpha\cdot\sqrt{x_1} = 0 \\ f_2(x_1,x_2) = \dfrac{w_1}{x_1}\cdot(cb_1 - x_2) + \dfrac{w_2}{x_1}\cdot(cb_2 - x_2) - \dfrac{k1\cdot x_2}{(1+k2\cdot x_2)^2} = 0 \end{cases}$$

$$\begin{cases} \dfrac{\partial f_1}{\partial x_1}(x_1,x_2) = -\dfrac{\alpha}{2}\cdot\dfrac{1}{\sqrt{x_1}} \text{ and } \dfrac{\partial f_1}{\partial x_2}(x_1,x_2) = 0 \\ \dfrac{\partial f_2}{\partial x_1}(x_1,x_2) = -\dfrac{w_1}{{x_1}^2}\cdot(cb_1 - x_2) - \dfrac{w_2}{{x_1}^2}\cdot(cb_2 - x_2) \text{ and } \dfrac{\partial f_2}{\partial x_2}(x_1,x_2) = -\dfrac{w_1}{x_1} - \dfrac{w_2}{x_1} - \mathrm{k}_1\cdot\dfrac{(1-\mathrm{k}_2\cdot x_2)}{(1+\mathrm{k}_2\cdot x_2)^3} \end{cases}$$

the Jacobian matrix (J)

	iteration	*L*	*Cb*	*f1(x1,x2)*	*f2(x1,x2)*	*J(1,1)*	*J(1,2)*	*J(2,1)*	*J(2,2)*
initial estimate →	0	85	1.600	0.15609	0.01978	-0.01085	0.00000	-0.00302	0.01061
	1	99.39	3.828	0.00610	0.01028	-0.01003	1.00000	-0.00176	0.00500
	2	100.00	1.988	0.00001	-0.01243	-0.01000	2.00000	-0.00210	0.01703
	3	100.00	2.717	0.00000	-0.00099	-0.01000	3.00000	-0.00196	0.01343
	4	100.00	2.791	0	-2E-05	-0.01	4	-0.00194	0.01287
final solution →	5	100.00	2.793	0	-9.7E-09	-0.01	5	-0.00194	0.01286

***J* =**

$$\begin{pmatrix} \left.\dfrac{\partial f_1}{\partial x_1}(x_1,x_2)\right|_{\text{old}} & \left.\dfrac{\partial f_1}{\partial x_2}(x_1,x_2)\right|_{\text{old}} \\ \left.\dfrac{\partial f_2}{\partial x_1}(x_1,x_2)\right|_{\text{old}} & \left.\dfrac{\partial f_2}{\partial x_2}(x_1,x_2)\right|_{\text{old}} \end{pmatrix}$$

Using Matlab

- Solving this example using Matlab maybe obtained with the help of symbolic derivation of the Jacobian matrix, followed by its evaluation and resolution of the system **J . δ = -f**
- The following Matlab M-file contains all the commands leading to a formatted solution in the form of summarized table of iterations
- Note that the text in green color and preceded by % contain comments and explanation
- Black text contain the Matlab executable commands or program statements

Implementing the “fsolve” command

Consider a system of non-linear equations to be solved using MATLAB

$$\begin{cases} f_1(x_1, \dots x_k)=0 \\ f_2(x_1, \dots xk)=0 \\ \vdots \quad \vdots \qquad \vdots \\ f_k(x_1, \dots. x_k)=0 \end{cases}$$

- The Matlab “fsolve” function maybe simply used by converting the system of equation into a function-script, then execute the “fsolve” command externally.
- The syntax is

```
solution = fsolve (@TheSystem,TheSatrtingGuess,options)
```

Writing the system and calling the *fsolve*

```
function F=TheSystem(x);
% This script will contain the components of the system of
% non-linear equations to be solved
%
F(1)=……………”write here equ. 1 f(1)”…………………….;
F(2)=……………”write here equ. 1 f(2)”…………………….;
:
F(k)=……………”write here equ. 1 f(k)”…………………….;
```

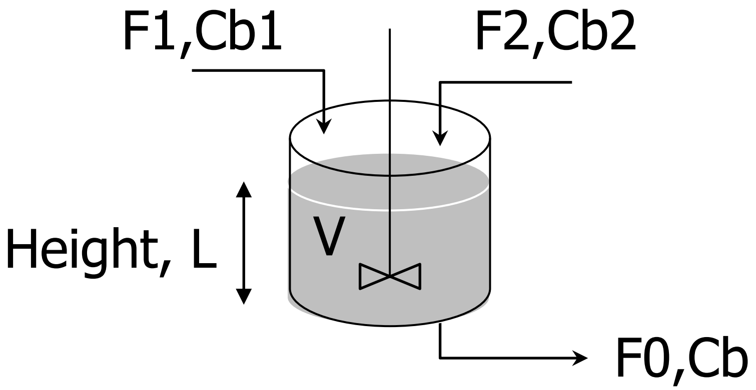
```
% The main program
:
:
% use the fsolve to get the solution
solution=fsolve(@TheSystem,xo)
```

Illustrative example

It is desired to estimate the steady state values of the height (L_∞) of the solution in CSTR and the final concentration (Cb_∞)

→ Use the overall conservation of mass and component balances to obtain a system of two equations as a function of L and Cb

→ Write the steady state system of equations and solve



Using fsolve with numerical evaluation of Jacobian

```
%  The Main program
clear all
% Step 1. Define given data
global alpha cb1 cb2 k1 k2 u1 u2 F1 F2
cb1=24.9;cb2=0.1; % Feed Concentration of components B1 and B2
k1=1;k2=1;          % Reaction rate constants k1 and k2
A=1;                % CSTR cross-sectional area
alpha = 0.2         % valve coeffecient
F1=1;F2=1;          % Feed volumetric flow rate of B1 and B2
u1=F1/A;u2=F2/A;
% Step 2. initial estimates and parameters initializations
x0=[85;1.6];
% Step 3. use “fsolve” to compute the solution
solution=fsolve(@CSTRSystem,x0)

options = optimoptions('fsolve','Display','iter');
%options = optimoptions('fsolve','Display','iter');
[x,fval,exitflag,output] = fsolve(@CSTRSystem,x0,options);
x,fval,exitflag
```

Using fsolve with numerical evaluation of Jacobian

```
function F=CSTRSystem(x)
global alpha cb1 cb2 k1 k2 u1 u2 F1 F2
F(1)=F1+F2-alpha *sqrt(x(1));
F(2)=u1/x(1)*(cb1-x(2))+u2/x(1)*(cb2-x(2))-k1*x(2)/(1+k2*x(2))^2;
```

Using fsolve with analytical evaluation of Jacobian

```
%  The Main program with analatyical jacobian
clear all
% Step 1. Define given data
global alpha cb1 cb2 k1 k2 u1 u2 F1 F2
cb1=24.9;cb2=0.1; % Feed Concentration of components B1 and B2
k1=1;k2=1;          % Reaction rate constants k1 and k2
A=1;                % CSTR cross-sectional area
alpha = 0.2 ;
F1=1;F2=1;          % Feed volumetric flow rate of B1 and B2
u1=F1/A;u2=F2/A;
% Step 2. initial estimates and parameters initializations
x0=[85;1.6];
% Step 3. use “fsolve” to compute the solution
options = optimoptions('fsolve','Display','iter','Jacobian','on');
%solution=fsolve(@JCSTRSystem,x0)
[x,fval,exitflag,output] = fsolve(@JCSTRSystem,x0,options);
x,fval,exitflag
```

Using fsolve with analytical evaluation of Jacobian

```
function [F,Jac] =JCSTRSystem(x)
global alpha cb1 cb2 k1 k2 u1 u2 F1 F2
F(1)=F1+F2-alpha *sqrt(x(1));
F(2)=u1/x(1)*(cb1-x(2))+u2/x(1)*(cb2-x(2)) -
k1*x(2)/(1+k2*x(2))^2;

% Evaluate the Jacobian matrix
Jac = zeros(2,2);
Jac(1,1) = -0.5 * alpha / sqrt(x(1));
Jac(2,1) = - 1/x(1)^2 *(u1* (cb1-x(2)) + u2* (cb2-x(2)));
Jac(2,2) = -u1/x(1) - u2/x(1)-(k1/(1+k2*x(2))^2 -
2*k1*k2*x(2)/(1+k2*x(2))^3);
return;
```

Using Matlab

The Matlab output is

```
solution =
        100
      2.7925
```