10.34, Numerical Methods Applied to Chemical Engineering
Prof. William Green
Lecture 2: Solving Systems of Linear Equations

Sample Function and Computing Tips

```
function k = rate(T, params)
% computes rate constant given temperature and Arrhenius parameters
% Bill Green 9/8/06
% inputs:
% T [=] Kelvin
% params = [A; n; Ea]
% A [=] 1/second
% n unitless exponent
% Ea [=] kj/mole
%
% output:
% k [=] 1/second
%
% unpack params
A=params(1);
N=params(2);
Ea=params(3);
R = 8.314; % gas constant J/mole-Kelvin
Ea=1000.*Ea;
K=A.*(T.^n).*exp(-Ea./(R.*T));
```

One additional feature is to include input/output example at bottom of code:

```
%Tvec = linspace(300,1200);
%params = [1e9;0.5;82];
%kvec=rate(Tvec,params);
%kvec(3)
%ans = 6.1551e-004
```

Use a lot of '%' comments for

1) The graders to give you partial credit
2) To help you understand your programs when you review
3) For your classmates if they need to operate your program

TEST your program in pieces!!
Otherwise you write a long program and you have no idea where the problem is.
"If you're going to build a laboratory apparatus, you check the power supply, you check if the tubes leak, if the safety features are in place, etc, before you run experiments. It's the same thing with software."

In MATLAB you don't have to describe the dimension of each array.
This can be used to your advantage by setting up the function as follows:
`function k = rate(T, params)` or using a semi-colon: f(x; p) or f(x; θ)

Matrix Algebra

P-norm of a vector:

$$\| v \|_p = \left[\sum_{k=1}^{N} | v_k |^p \right]^{1/p}$$

p = 1 city-block norm
p = 2 length, Euclidean norm
p = ∞ largest element, useful for error tolerances

Triangle Inequality:
$\| v + w \|_p \leq \| v \|_p + \| w \|_p$ for any p; good for proving bounds

Matrix times a vector:
M * v
1) Weighted sum of cols of M

$$M * v = \sum v_i \vec{M}_i^{col}$$

2) Column of dot products

$$\begin{pmatrix} \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{pmatrix} \begin{pmatrix} \vdots \\ \vdots \\ \vdots \\ \vdots \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \Rightarrow \begin{pmatrix} \vec{M}_1^{row} \bullet \vec{v} \\ \vec{M}_2^{row} \bullet \vec{v} \\ \vdots \\ \vec{M}_n^{row} \bullet \vec{v} \end{pmatrix}$$

3) Rotate and Grow/Shrink
4) Linear Mapping

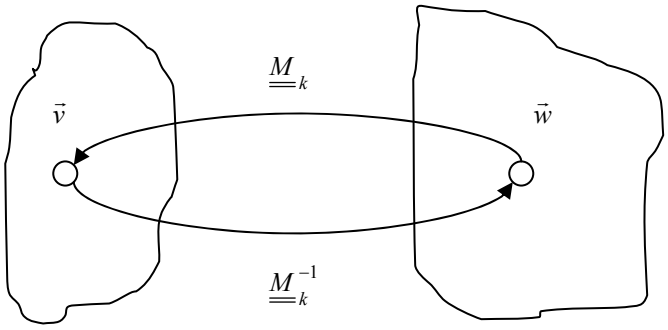

Figure 1. A linear map.

All four of these are going on and you can use whichever you want in the current application.

"I teach little tidbits of information. You have to read the textbook if you want details."

Reactor System Example

Alcohol + Acid → Ester + H2O
R-OH + RCOOH <-> RCOOR + H_2O

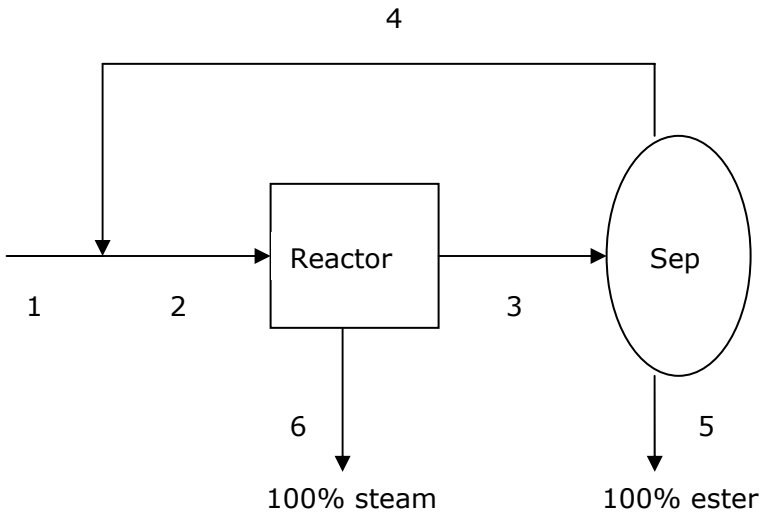

Figure 2. A reactor for alcohol and acid.

6 streams and four compounds = 24 stream variables (not counting energy balance)

Assumptions: Stream 3 has same composition Alcohol/Acid as Stream 4
Set BASIS for stream 1
Unknowns: Compositions for streams 2, 4

$$x = \begin{matrix} m_{2A} \\ m_{4A} \\ m_{2ROH} \\ m_{4ROH} \\ m_{2H2O} \\ m_{4H2O} \end{matrix}$$

$m_{1,ACID} + m_{4,ACID} = m_{2,ACID}$
$m_{1,ROH} + m_{4,ROH} = m_{2,ROH}$
$m_{1,H2O} + m_{4,H2O} = m_{2,H2O}$

IR Analysis of Streams 4 and 2:
$m_{ROH} / m_{H2O} = 0.43$
$m_{2,ROH}/m_{2,Acid} = 1.4$
$m_{2,tot} = 2.1m_{1,tot}$

Set up linear matrix of equations as:

$$\begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 & 0 & -.43 \\ -1.4 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} (x) = \begin{bmatrix} -m_{1,Acid} \\ -m_{1,ROH} \\ -m_{1,H2O} \\ 0 \\ 0 \\ 2.1m_{1,tot} \end{bmatrix}$$

Rearrange equations and use elimination to produce upper triangle matrix U
$x_{N} = v_N / U_{NN}$ gives you value of last element
Use backward substitution to solve for other unknowns

$$\begin{pmatrix} \bullet & \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet & \bullet \\ 0 & 0 & \bullet & \bullet \\ 0 & 0 & 0 & \bullet \end{pmatrix}(x) = \begin{pmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{pmatrix}$$

U **v**

$$U * x = x_1 \begin{pmatrix} \bullet \\ 0 \\ 0 \\ 0 \end{pmatrix} + x_2 \begin{pmatrix} \bullet \\ \bullet \\ 0 \\ 0 \end{pmatrix} + x_3 \begin{pmatrix} \bullet \\ \bullet \\ \bullet \\ 0 \end{pmatrix} + x_4 \begin{pmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{pmatrix} = (\underline{v})$$

$$x_4 = v_4 \big/ U_{44}$$

$$\vec{v}_{new} = \vec{v} - x_4 * U_4^{col}$$

$$x_3 = v_3^{new} \big/ U_{33}$$

Can permute the rows. Can permute the columns, if you permute the variables in the $\underline{x}$ vector.

MATLAB examples

```
function x = backsub(U,v)
N =
for i=1: (N-1)
      m = N+1 - i;
      x(m) = v(m)/U(m,m);
      v=v-x(m)*U(: ,m)
```

Use of the colon ':'

```
M =[1 2 3 4; 5 6 7 8];
v1=M(:,1)
    1
    5
m = M(:,1:2)
m =
        1   2
        5   6
```

for loop
Save as: `silly.m`

```
function sum = silly
sum = 0
for i = 2:4
       sum = sum + i;
end
```