10.34, Numerical Methods Applied to Chemical Engineering
Professor William H. Green
Lecture #3: Matrix Factorization. Modularization.

$$\mathbf{U}\cdot\vec{x}=\vec{v} \qquad \begin{pmatrix} \bullet & \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet & \bullet \\ 0 & 0 & \bullet & \bullet \\ 0 & 0 & 0 & \bullet \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}=\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} \qquad x_1\begin{pmatrix} \bullet \\ 0 \\ 0 \\ 0 \end{pmatrix}+x_2\begin{pmatrix} \bullet \\ \bullet \\ 0 \\ 0 \end{pmatrix}+x_3\begin{pmatrix} \bullet \\ \bullet \\ \bullet \\ 0 \end{pmatrix}+x_4\begin{pmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{pmatrix}=(\vec{v})$$

```
function x = backsub(U,v)
N = length(v);     % dimension of matrix

for i = 1:(N-1)
    m = N+1-i;     % backwards from bottom matrix
    x(m) = v(m)/U(m,m);   % solve equation with one unknown
    v = v - x(m)*U(:,m);  % move terms involving known x(m) to r.h.s.
end

x(1) = v(1)/U(1,1);
% tested using: U = [1 2 3; 0 4 5; 0 0 6], v = [0 1 3]
% x = [-.75 -.375 .5]
```

} To comment a block of text: Ctrl-R

Gaussian Elimination

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix}*\vec{x}=\begin{pmatrix} 0 \\ 1 \\ 3 \end{pmatrix} \qquad \text{Augment:}\left(\begin{array}{ccc|c} 1 & 2 & 3 & 0 \\ 4 & 5 & 6 & 1 \\ 7 & 8 & 0 & 3 \end{array}\right)\xrightarrow[\text{to } 2^{\text{nd}} \text{ row}]{\text{add } -4*1^{\text{st}} \text{ row}}\left(\begin{array}{ccc|c} 1 & 2 & 3 & 0 \\ 0 & -3 & -6 & 1 \\ 0 & -6 & -21 & 3 \end{array}\right)\xrightarrow[\text{to } 3^{\text{rd}} \text{ row}]{\text{add } 2*2^{\text{nd}} \text{ row}}$$

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 0 \\ 0 & -3 & -6 & 1 \\ 0 & 0 & -9 & 1 \end{array}\right)\Rightarrow \text{Upper Triangular solution}$$

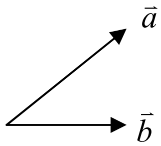
gauss.m

```
function [U,v] = gauss(A,b)
% uses Gaussian elimination to convert Ax=b to Ux=v
% where U is upper triangular and A is NxN matrix and b is Nx1 matrix

M = [A b];  N = length(b);
for i = 1:(N-1)
      for j = (i+1):N
            lambda = -M(j,i)/M(i,i);
            M(j,i) = M(j,i)+lambda*M(i,i);
      end
end
U = M(i,1:N)
v = M(i,N+1)
```

} unpacks augmented solution

Gaussian Elimination has N^3 operations. Not good when N is large (10^6). Backsub requires N^2 operations.



} can span entire 2D vector space (linear combination)

Figure 1. 2 vectors with a common origin and different directions.

$\vec{x} = c_1\vec{a} + c_2\vec{b}$ }

$\vec{x} = c_1\vec{a} + c_2\vec{b} + c_3\vec{d}$ can span entire 3D vector space (linear combination)

*Not linearly independent if: $\vec{d} = c_1\vec{a} + c_2\vec{b}$

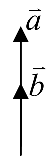


Figure 2. 2 vectors with a common origin and direction.

same direction - not independent

rank (m) = # of linearly independent column vectors = dimension of space spanned by column vectors

Existence

A * x = b

$$x_1\begin{pmatrix}\text{col}\\1\end{pmatrix} + x_2\begin{pmatrix}\text{col}\\2\end{pmatrix} + \cdots + x_N\begin{pmatrix}\text{col}\\N\end{pmatrix} = (\vec{b})$$

solution exists if:
rank(A) == rank ([A b])

if rank(A) = N and A is NxN: does not matter what $\vec{b}$ is; always a solution

If null space of A does not include any vector beyond 0, then solution unique
The null space is the set of all vectors of such that x = 0.

} Ay = 0
Ax = b
A(x+y) = b

Run ***gauss.m*** => Error!
NaN: error – not a number (divided by 0)
Need to pivot rows to avoid 0 in diagonal
* see "gausselim_pivot.m" online *

$A*x_1 = b_1$
$A*x_2 = b_2$ } Gauss elimination has nothing to do with b, only A

$$\underbrace{L^{-1} * A}_{U} * x_1 = \underbrace{L^{-1} * b_1}_{v}$$

L: lower triangular } Alternative method to Gaussian elimination

LUx = b
Lv = b } Ux=v

MATLAB:

```
[L,U,P] = lu(A)
A = P'*L*U      (factorizing A)
PAx = Pb
LU

LUx = Pb

Lv = P*b (forward sub)

Ux = v   (backward sub)
```

For more info: help lu