

Engineering Analysis ENG 3420 Fall 2009

Dan C. Marinescu

Office: HEC 439 B

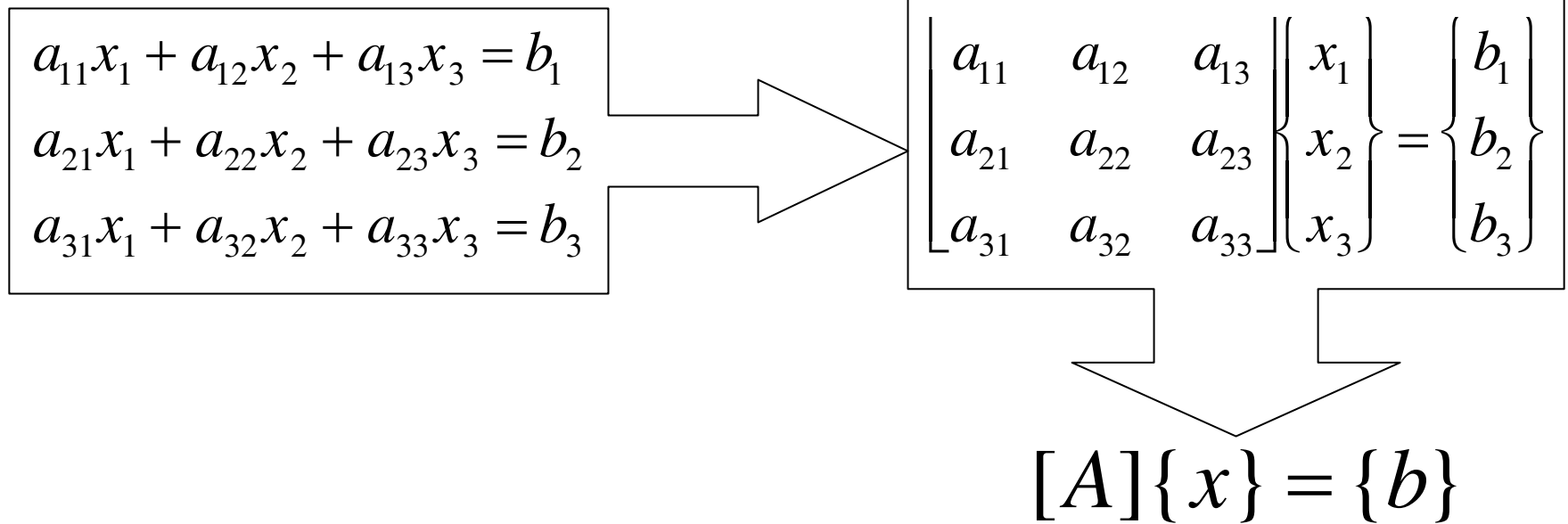
Office hours: Tu-Th 11:00-12:00

Lecture 13

- Last time:
 - Problem solving in preparation for the quiz
 - Linear Algebra Concepts
 - Vector Spaces, Linear Independence
 - Orthogonal Vectors, Bases
 - Matrices
- Today
 - Solving systems of linear equations (Chapter 9)
 - Graphical methods
- Next Time
 - Gauss elimination

Solving systems of linear equations

- Matrices provide a concise notation for representing and solving simultaneous linear equations:

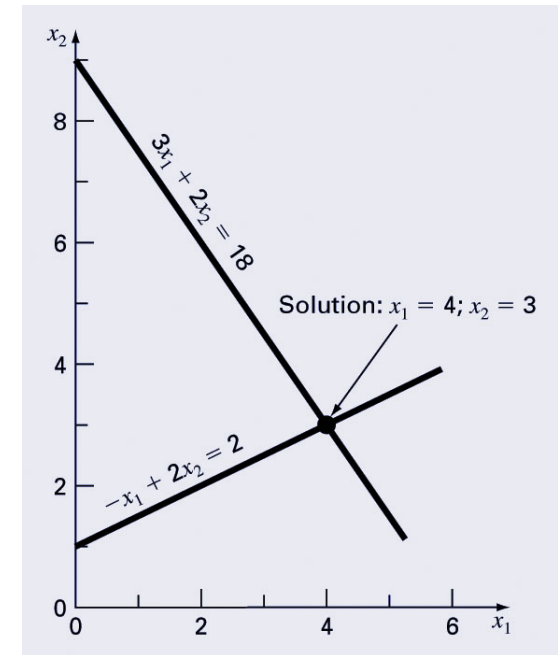
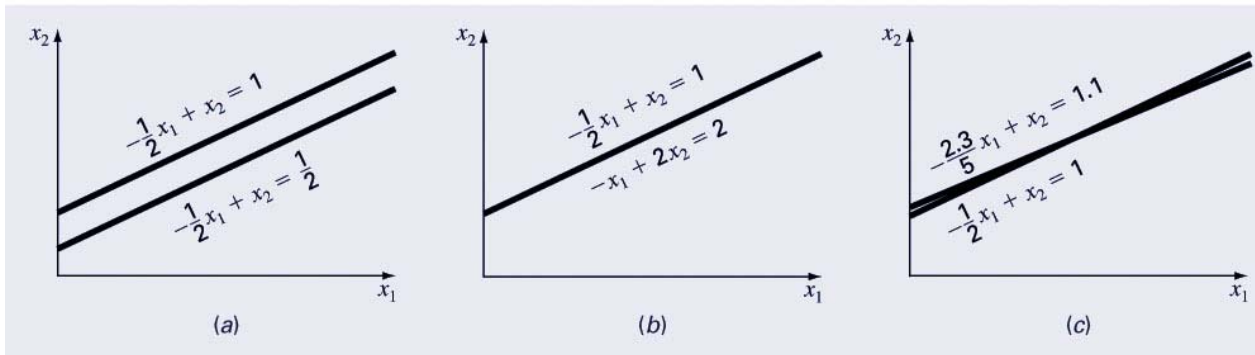


Solving systems of linear equations in Matlab

- Two ways to solve systems of linear algebraic equations $[A]\{x\}=\{b\}$:
 - Left-division
 $x = A \setminus b$
 - Matrix inversion
 $x = \text{inv}(A) * b$
- Matrix inversion only works for square, non-singular systems; it is less efficient than left-division.

Solving graphically systems of linear equations

- For small sets of simultaneous equations, graphing them and determining the location of the intersection of the straight line representing each equation provides a solution.
- There is no guarantee that one can find the solution of system of linear equations:
 - a) No solution exists
 - b) Infinite solutions exist
 - c) System is ill-conditioned



Determinant of the square matrix $A=[a_{ij}]$

$$A = [a_{ij}] = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n-1} & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n-1} & a_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & a_{n-1,n-1} & a_{n-1,n} \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

$$|A| = \det(A) = A_{i1}a_{i1} + A_{i2}a_{i2} + \cdots A_{in}a_{in}$$

- Here the coefficient A_{ij} of a_{ij} is called the cofactor of A
- A cofactor is a polynomial in the remaining rows of A and can be described as the partial derivative of A . The cofactor polynomial contains only entries from an $(n-1) \times (n-1)$ matrix M_{ij} called a “minor” obtained from A by eliminating row i and column j .

Determinants of several matrices

- Determinants for 1x1, 2x2, 3x3 matrices are:

$$1 \times 1 \quad |a_{11}| = a_{11}$$

$$2 \times 2 \quad \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

$$3 \times 3 \quad \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

- Determinants for square matrices larger than 3 x 3 are more complicated.

Properties of the determinants

- If we permute two rows of the rectangular matrix A then the sign of the determinant $\det(A)$ changes.
- The determinant of the transpose of a matrix A is equal to the determinant of the original matrix.
- If two rows of A are identical then $|A|=0$

Cramer's Rule

- Consider the system of linear equations:

$$[A]\{x\}=\{b\}$$

- Each unknown in a system of linear algebraic equations may be expressed as a fraction of two determinants with denominator D and with the numerator obtained from D by replacing the column of coefficients of the unknown in question by the vector b consisting of constants b_1, b_2, \dots, b_n .

Example of the Cramer's Rule

- Find x_2 in the following system of equations:

$$0.3x_1 + 0.52x_2 + x_3 = -0.01$$

$$0.5x_1 + x_2 + 1.9x_3 = 0.67$$

$$0.1x_1 + 0.3x_2 + 0.5x_3 = -0.44$$

- Find the determinant D

$$D = \begin{vmatrix} 0.3 & 0.52 & 1 \\ 0.5 & 1 & 1.9 \\ 0.1 & 0.3 & 0.5 \end{vmatrix} = 0.3 \begin{vmatrix} 1 & 1.9 \\ 0.3 & 0.5 \end{vmatrix} - 0.52 \begin{vmatrix} 0.5 & 1.9 \\ 0.1 & 0.5 \end{vmatrix} + 1 \begin{vmatrix} 0.5 & 1 \\ 0.1 & 0.4 \end{vmatrix} = -0.0022$$

- Find determinant D_2 by replacing D 's second column with b

$$D_2 = \begin{vmatrix} 0.3 & -0.01 & 1 \\ 0.5 & 0.67 & 1.9 \\ 0.1 & -0.44 & 0.5 \end{vmatrix} = 0.3 \begin{vmatrix} 0.67 & 1.9 \\ -0.44 & 0.5 \end{vmatrix} - 0.01 \begin{vmatrix} 0.5 & 1.9 \\ 0.1 & 0.5 \end{vmatrix} + 1 \begin{vmatrix} 0.5 & 0.67 \\ 0.1 & -0.44 \end{vmatrix} = 0.0649$$

- Divide

$$x_2 = \frac{D_2}{D} = \frac{0.0649}{-0.0022} = -29.5$$

Gauss Elimination

- Gauss elimination \rightarrow a sequential process of removing unknowns from equations using *forward elimination* followed by *back substitution*.
- “Naïve” Gauss elimination \rightarrow the process does not check for potential problems resulting from division by zero.

Naïve Gauss Elimination (cont)

■ Forward elimination

- Starting with the first row, add or subtract multiples of that row to eliminate the first coefficient from the second row and beyond.
- Continue this process with the second row to remove the second coefficient from the third row and beyond.
- Stop when an upper triangular matrix remains.

■ Back substitution

- Starting with the *last* row, solve for the unknown, then substitute that value into the next highest row.
- Because of the upper-triangular nature of the matrix, each row will contain only one more unknown.

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array} \right]$$



$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ & a'_{22} & a'_{23} & b'_2 \\ & & a''_{33} & b''_3 \end{array} \right]$$

(a) Forward elimination



$$x_3 = b''_3 / a''_{33}$$

$$x_2 = (b'_2 - a'_{23}x_3) / a'_{22}$$

$$x_1 = (b_1 - a_{13}x_3 - a_{12}x_2) / a_{11}$$

(b) Back substitution

```

function x = GaussNaive(A,b)
% GaussNaive: naive Gauss elimination
%   x = GaussNaive(A,b): Gauss elimination without pivoting.
% input:
%   A = coefficient matrix
%   b = right hand side vector
% output:
%   x = solution vector

[m,n] = size(A);
if m~=n, error('Matrix A must be square'); end
nb = n+1;
Aug = [A b];
% forward 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);
    end
end
% back substitution
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

```

```

function x=GaussNaive(A,b)
ExA=[A b];
[m,n]=size(A);
q=size(b);
if (m~=n)
    fprintf ('Error: input matrix is not square; n = %3.0f, m=%3.0f \n', n,m);
End
if (n~=q)
    fprintf ('Error: vector b has a different dimension than n; q = %2.0f \n', q);
end
n1=n+1;
for k=1:n-1
    for i=k+1:n
        factor=ExA(i,k)/ExA(k,k);
        ExA(i,k:n1)= ExA(i,k:n1)-factor*ExA(k,k:n1);
    End
End
x=zeros(n,1);
x(n)=ExA(n,n1)/ExA(n,n);
for i=n-1:-1:1
    x(i) = (ExA(i,n1)-ExA(i,i+1:n)*x(i+1:n))/ExA(i,i);
end

```

```
>> C=[150 -100 0 ; -100 150 -50; 0 -50 50]
```

```
C =  
150 -100 0  
-100 150 -50  
0 -50 50
```

```
>> d= [588.6; 686.7;784.8]
```

```
d =  
588.6000  
686.7000  
784.8000
```

```
>> x = GaussNaive(C,d)
```

```
x =
```

```
41.2020
```

```
55.9170
```

```
71.6130
```

```

>> A=[1 1 1 0 0 0;
0 -1 0 1 -1 0;
0 0 -1 0 0 1;
0 0 0 1 -1;
0 10 -10 0 -15 -5;
5 -10 0 -20 0 0]
A =
    1     1     1     0     0     0
    0    -1     0     1    -1     0
    0     0    -1     0     0    -1
    0     0     0     0     1    -1
    0    10 -10     0 -15    -5
    5   -10     0 -20     0     0
b =
    0     0     0     0     0    200
>> b=b'
b =
    0
    0
    0
    0
    0
    200
>> x = GaussNaive(A,b)
x =
NaN
NaN
NaN
NaN
NaN
NaN

```

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

$$\underline{x =}$$

$$\begin{array}{r} \underline{6.1538} \\ \underline{-4.6154} \\ \underline{-1.5385} \\ \underline{-6.1538} \\ \underline{-1.5385} \\ \underline{-1.5385} \end{array}$$

$$\underline{>> x=inv(A)*b}$$

$$\underline{x =}$$

$$\begin{array}{r} \underline{6.1538} \\ \underline{-4.6154} \\ \underline{-1.5385} \\ \underline{-6.1538} \\ \underline{-1.5385} \\ \underline{-1.5385} \end{array}$$

Complexity of Gauss elimination

- To solve an $n \times n$ system of linear equations by Gauss elimination we carry out the following number of operations:

Forward Elimination	$\frac{2n^3}{3} + O(n^2)$
Back Substitution	$n^2 + O(n)$
Total	$\frac{2n^3}{3} + O(n^2)$

- Flops \rightarrow floating-point operations. Mflops/sec number of floating point operation executed by a processor per second.
- Conclusions:
 - As the system gets larger, the computation time increases greatly.
 - Most of the effort is incurred in the elimination step.

Pivoting

- If a coefficient along the diagonal is 0 (problem: division by 0) or close to 0 (problem: round-off error) then the Gauss elimination causes problems.
- Partial pivoting → determine the coefficient with the largest absolute value in the column below the pivot element. The rows can then be switched so that the largest element is the pivot element.
- Complete pivoting → check also the rows to the right of the pivot element are also checked and switch columns.

Partial Pivoting Program

```
function x = GaussPivot(A,b)
% GaussPivot: Gauss elimination pivoting
%   x = GaussPivot(A,b): Gauss elimination with pivoting.
% input:
%   A = coefficient matrix
%   b = right hand side vector
% output:
%   x = solution vector

[m,n]=size(A);
if m~=n, error('Matrix A must be square'); end
nb=n+1;
Aug=[A b];
% forward elimination
for k = 1:n-1
    % partial pivoting
    [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
% back substitution
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
```


Tridiagonal system solver

```
function x = Tridiag(e,f,g,r)
% Tridiag: Tridiagonal equation solver banded system
%   x = Tridiag(e,f,g,r): Tridiagonal system solver.
% input:
%   e = subdiagonal vector
%   f = diagonal vector
%   g = superdiagonal vector
%   r = right hand side vector
% output:
%   x = solution vector
n=length(f);
% forward elimination
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
```