MATLAB Review

Using MATLAB to solve basic linear systems of equations and optimization problems

Mehsam Khan

Mehsam Khan

<u>*Background:*</u>

Ph.D. Candidate in Mechanics, Materials and Structures
M.S. in Structural Engineering
B.S. in Civil Engineering

mehsam@u.northwestern.edu

MATLAB (**mat**rix **lab**oratory)

(1) for numerical/symbolic, scientific computations and other apps.

(2) shorter program development and debugging time than traditional programming languages such as FORTRAN and C.

(3) slow (compared with FORTRAN or C) because it is interpreted.

(4) automatic memory management.

(5) intuitive, easy to use.

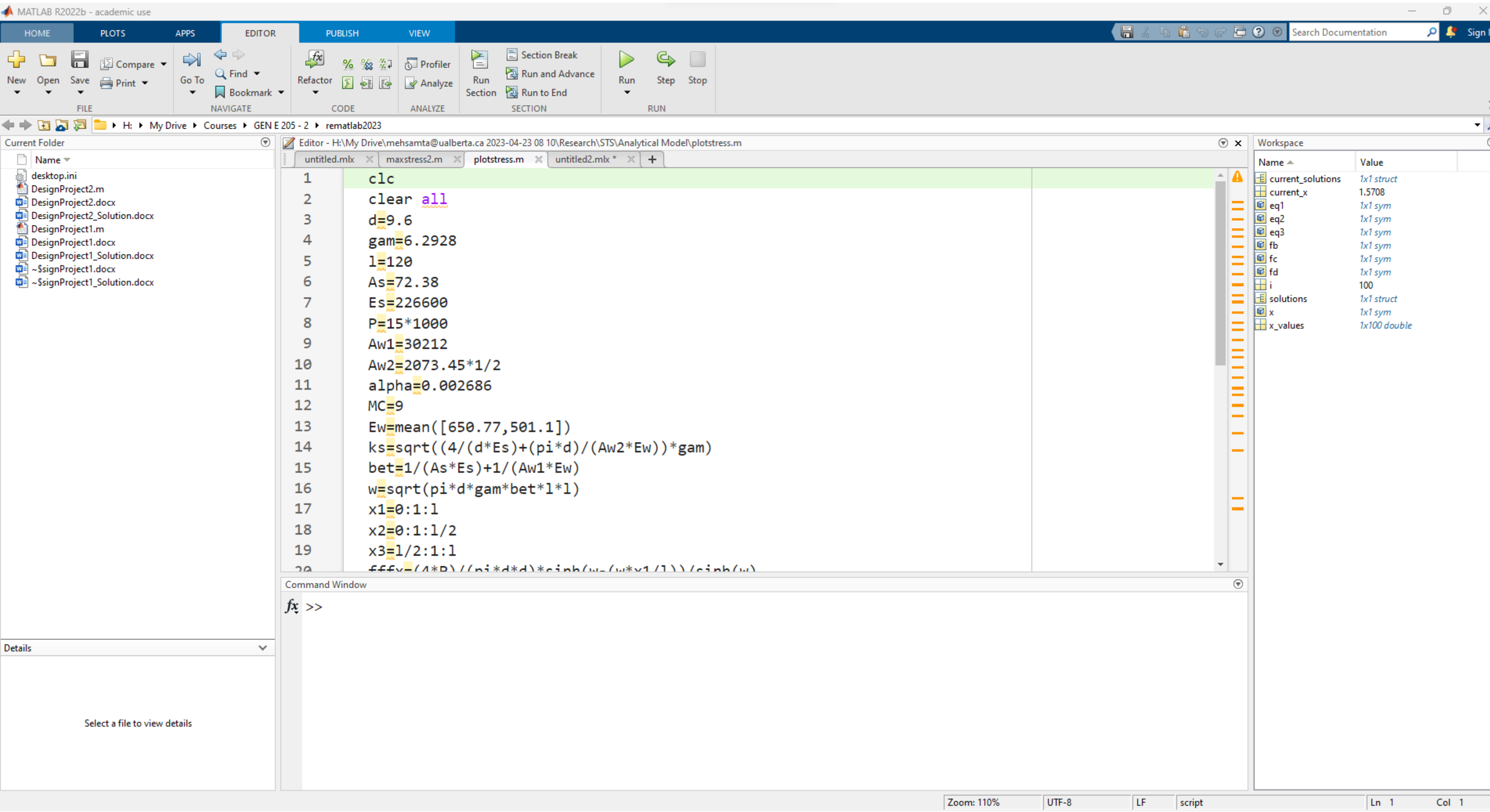
(6) compact notations.

L-shaped membrane logo

How to download your own MATLAB?

https://www.mccormick.northwestern.edu/it-resources/computer-software/matlab-support/#access

MATLAB Graphical User Interface (GUI)



Basics

- Scalar

$$\boldsymbol{VariableName = Value}$$

e.g. $\boldsymbol{A = 5}$

```
>> A=5

A =

     5
```

- Do not need to declare data type
- The variable name is always on the left
- Do not give variables the same names as MATLAB functions (MATLAB won't always stop you from doing this so be careful)
    - e.g. $\boldsymbol{sqrt}$ is the square root function so do not type $\boldsymbol{sqrt} = \boldsymbol{sqrt(VariableName)}$
    - If this happens you will lose the $\boldsymbol{sqrt}$ function
- Clear variables by typing $\boldsymbol{clear\ VariableName}$ or clear all variables by typing $\boldsymbol{clear\ all}$

Basics

▪ Vector

Vectors can be defined several ways

- Manually: $\boldsymbol{VariableName} = [\boldsymbol{Value1} \quad \boldsymbol{Value2} \quad \boldsymbol{Value3}]$

- Fixed interval: $\boldsymbol{VariableName} = \boldsymbol{start} : \boldsymbol{interval} : \boldsymbol{end}$
  - e.g. $\boldsymbol{A} = \mathbf{1} : \mathbf{0.25} : \mathbf{2} \quad \rightarrow \quad [\mathbf{1} \;\; \mathbf{1.25} \;\; \mathbf{1.5} \;\; \mathbf{1.75} \;\; \mathbf{2}]$
  - If no interval is defined then an interval of 1 is assumed (e.g. $\boldsymbol{A} = \mathbf{1}{:}\,\mathbf{5} \quad \rightarrow \quad [\mathbf{1} \;\; \mathbf{2} \;\; \mathbf{3} \;\; \mathbf{4} \;\; \mathbf{5}]$)

- Fixed number of elements $\boldsymbol{VariableName} = \boldsymbol{linspace}(\boldsymbol{start}, \boldsymbol{end}, \boldsymbol{number\ of\ elements})$
  - e.g. $\boldsymbol{A} = \boldsymbol{linspace}(\mathbf{1}, \mathbf{3}, \mathbf{5}) \quad \rightarrow \quad [\mathbf{1} \;\; \mathbf{1.5} \;\; \mathbf{2} \;\; \mathbf{2.5} \;\; \mathbf{3}]$

- Use single quote (‘) to transpose between row and column vectors
  - e.g. $\boldsymbol{A} = [\mathbf{1} \;\; \mathbf{2} \;\; \mathbf{3}] \quad \rightarrow \quad \boldsymbol{A}' = \begin{bmatrix} \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \end{bmatrix}$

Basics

▪ Matrix

Matrices are defined by typing a semicolon (;) between rows

- e.g. $\boldsymbol{A} = [\mathbf{1}\ \ \mathbf{2}\ \ \mathbf{3}; \mathbf{4}\ \ \mathbf{5}\ \ \mathbf{6}] \quad \rightarrow \quad \boldsymbol{A} = \begin{bmatrix} \mathbf{1} & \mathbf{2} & \mathbf{3} \\ \mathbf{4} & \mathbf{5} & \mathbf{6} \end{bmatrix}$

Matrices can also be built from vectors

- e.g. $\boldsymbol{A} = [\mathbf{1}\ \ \mathbf{2}\ \ \mathbf{3}], \boldsymbol{B} = [\mathbf{4}\ \ \mathbf{5}\ \ \mathbf{6}], \boldsymbol{C} = [\boldsymbol{A}; \boldsymbol{B}] \quad \rightarrow \quad \boldsymbol{C} = \begin{bmatrix} \mathbf{1} & \mathbf{2} & \mathbf{3} \\ \mathbf{4} & \mathbf{5} & \mathbf{6} \end{bmatrix}$

Matrix elements, rows, and columns can be addressed individually

- e.g. $\boldsymbol{C}(\mathbf{1}, \mathbf{3}) = \mathbf{3}$ (returns the first row, third column element of C)
- $\boldsymbol{C}(\mathbf{1}, :) = [\mathbf{1}\ \ \mathbf{2}\ \ \mathbf{3}]$ (returns the first row of C. The colon (:) alone means all)
- $\boldsymbol{C}(:, \mathbf{2}) = \begin{bmatrix} \mathbf{2} \\ \mathbf{5} \end{bmatrix}$ (returns the second column of C)
- $\boldsymbol{C}(\mathbf{1}, \mathbf{2}: \mathbf{3}) = [\mathbf{2}\ \ \mathbf{3}]$ (returns the first row, second through third column elements of C)

Basics

▪ Matrix

Basic operations (+ - * / ^) are supported by MATLAB

- * and ^ will perform matrix multiplication on matrices
- Preceding the operation with a period (.) will perform the operation element by element
- e.g. $\boldsymbol{A} = \begin{bmatrix} \mathbf{1} & \mathbf{2} \\ \mathbf{3} & \mathbf{4} \end{bmatrix} \quad \rightarrow \quad \boldsymbol{A} * \boldsymbol{A} = \begin{bmatrix} \mathbf{7} & \mathbf{10} \\ \mathbf{15} & \mathbf{22} \end{bmatrix}, \quad \boldsymbol{A} .* \boldsymbol{A} = \begin{bmatrix} \mathbf{1} & \mathbf{4} \\ \mathbf{9} & \mathbf{16} \end{bmatrix}$

Useful functions

- $\boldsymbol{zeros(n, m)}$ creates a n x m matrix with all zeros
- $\boldsymbol{ones(n, m)}$ creates a n x m matrix with all ones
- $\boldsymbol{eye(n)}$ creates a n x n identity matrix
- $\boldsymbol{length(VectorName)}$ returns the number of elements in the vector
- $\boldsymbol{size(MatrixName)}$ returns the dimensions of the matrix
  - Note that this function outputs a vector

Basics

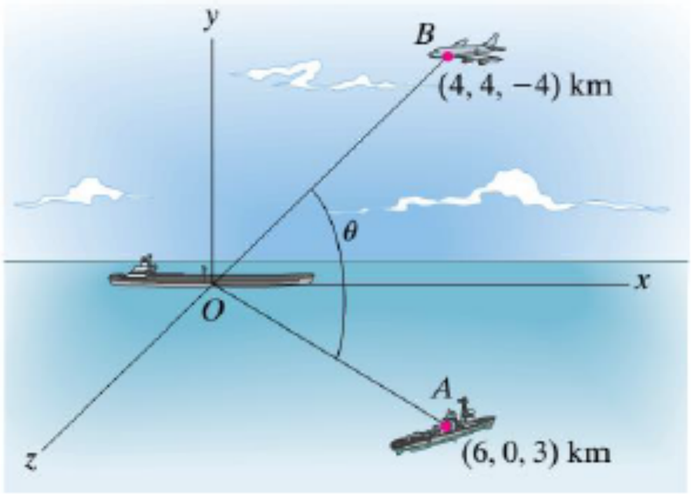
$$\cos\theta = \frac{\vec{r}_A \vec{r}_B}{\left|\vec{r}_A\right|\left|\vec{r}_B\right|}$$

- Operation for linear algebra
  - norm(A) % Magntiude of vector A
  - dot(A,B) % Scalar dot product of A and B
  - cross(A,B) % Cross product of A and B

Example 1

Problem 2.109 The ship *O* measures the positions of the ship *A* and the airplane *B* and obtains the coordinates shown. What is the angle θ between the lines of sight *OA* and *OB*?

```
OA=[6, 0, 3];  % position vector OA
OB=[4, 4, -4]; % position vector OA
costheta=dot(OA,OB)/(norm(OA)*norm(OB));
% From Eq. (2.24)
theta=acosd(costheta)
```



Be careful! sind v.s. sin; cos v.s cosd

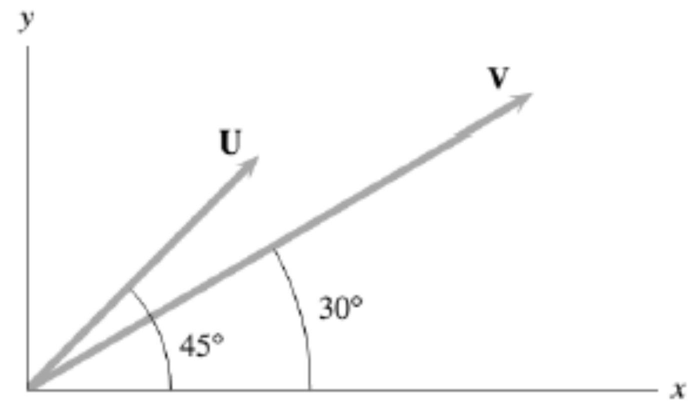
Basics

Example 2

Problem 2.130 The magnitudes $|\mathbf{U}| = 10$ and $|\mathbf{V}| = 20$.

(a) Use the definition of the cross product to determine $\mathbf{U} \times \mathbf{V}$.
(b) Use the definition of the cross product to determine $\mathbf{V} \times \mathbf{U}$.
(c) Use Eq. (2.34) to determine $\mathbf{U} \times \mathbf{V}$.

(d) Use Eq. (2.34) to determine $\mathbf{V} \times \mathbf{U}$.




```
U=[10*cosd(45),10*sind(45),0]; % Vector U
V=[20*cosd(30),20*sind(30),0]; % Vector V
cross(U,V) % U × V
cross(V,U) % V × U
```

Programming and Scripts

- Scripts (.m files)
  - Scripts allow you to run many commands in sequence
  - Write scripts
    - Click “New Script” to open the editor window
    - Terminate lines with semicolon (;) to suppress their output to command window
    - Use percent sign (%) to comment
  - Run scripts
    - In command window, type “run scriptname”
    - Click *Run* buttom on *Editor*

Programming and Scripts

- Conditional structures

```
if  expression 1
          sentence 1
elseif  expression 2
          sentence 2
elseif  expression 3
          sentence 3
……
else
          sentence n
end
```

```
a=3;b=4;
if a == b,
   fprintf('a is equal to b\n');
elseif a > 0 && b > 0
   fprintf('both positive\n');
else
   fprintf('other case\n');
end
```

Logical operators

&	logical AND
|	logical OR
~	logical NOT

<	less than
>	larger than
<=	less than or equal to
>=	less than or equal to
==	equal
~=	not equal

Programming and Scripts

- Loops

```
for index = initVal:step:endVal
        sentences
end
```

- ❑ Execute statements specified number of times
- ❑ The number of iteration depends on the length of *index*
- ❑ MATLAB allows to use one loop inside another loop (nested loop)
- ❑ While loop

```
while expression
        sentences
end
```

Example1: find 1+2+..100

```
sum = 0;
for i = 1: 100
   sum = sum + i;
end
```

Example2: sum all elements of Matrix M

```
M = rand(4,4); suma = 0;
for i = 1:4
   for j = 1:4
      suma = suma + M(i,j);
   end
end
```

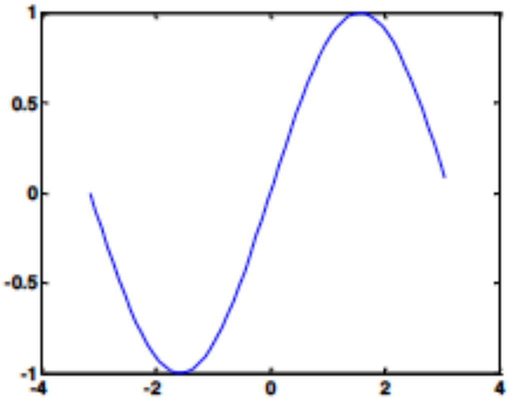
Plotting

- 2D line plot

plot(X1,Y1,...,Xn,Yn)

e.g. $y = \sin(x)$

```
x = -pi:.1:pi;
y = sin(x);
plot(x,y)
```



- 3D plot (functions of two variables)

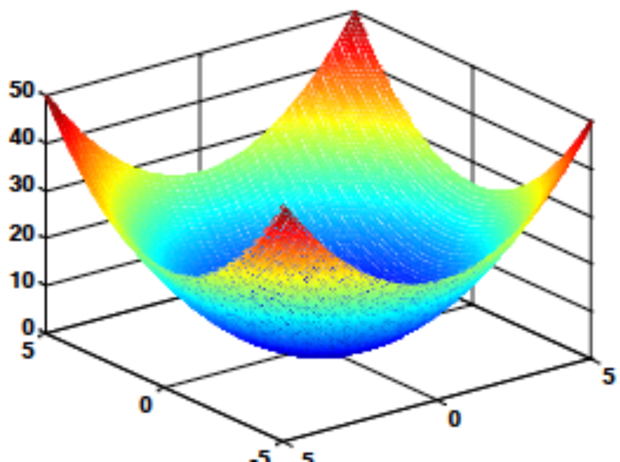
surf(X, Y, Z) **mesh**(X, Y, Z)

contour(X, Y, Z)

e.g. $f = x_1^2 + x_2^2$

```
clear;clc;
x1=-5:0.1:5;
x2=-5:0.1:5;
f=zeros(length(x1),length(x2));
for i=1:length(x1)
    for j=1:length(x2)
        f(i,j)=x1(i)^2+x2(j)^2;
    end
end
surf(x1,x2,f)
```

Pay attention to a common mistake.



Plotting

Example
Find: min $f(x_1, x_2) = x_1^2 + x_2^2$ at domain $-5 < x_1 < 5, -5 < x_2 < 5$

```
x1 = linspace(-5, 5, 100);
x2 = linspace(-5, 5, 100);

[X1, X2] = meshgrid(x1, x2);

f = X1.^2 + X2.^2;

figure;

surf(x1, x2, f);
xlabel('x1');
ylabel('x2');
zlabel('f(x1, x2)');
title('Plot of f(x1, x2) = x1^2 + x2^2');
```

```
x = 1:3;
y = 1:5;
[X,Y] = meshgrid(x,y)
```

```
X = 5×3

     1     2     3
     1     2     3
     1     2     3
     1     2     3
     1     2     3

Y = 5×3

     1     1     1
     2     2     2
     3     3     3
     4     4     4
     5     5     5
```

Evaluate the expression $x^2 + y^2$ over the 2-D grid.

```
X.^2 + Y.^2
```

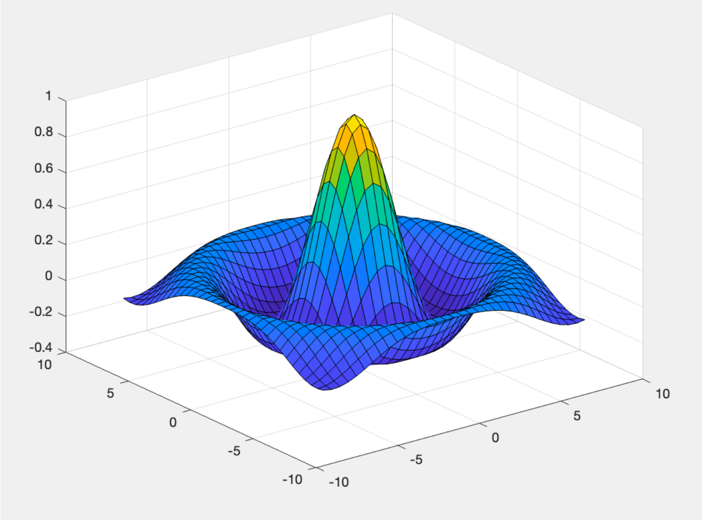
Plotting

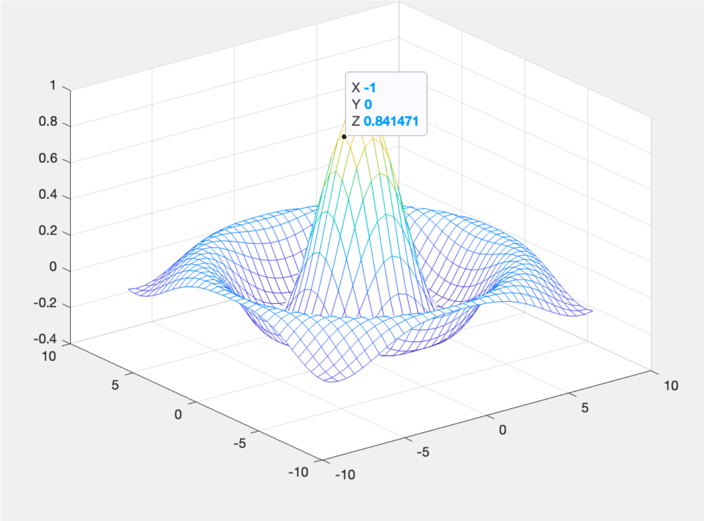
- After the plot command you can add axes labels, plot title, legend, limit for axis,etc.
  - ❑ xlabel(‘x axis label’)
  - ❑ ylabel(‘y axis label’)
  - ❑ title(‘plot title’)
  - ❑ legend(‘first line label’,’second line label’)
  - ❑ xlim([x y]) and ylim([x y])
  - ❑ Difference choices of the linetype and color (‘*, r, b, --, ‘)

More info:
http://www.mathworks.com/help/matlab/ref/plot.html?s_tid=gn_loc_drop

A MATLAB program can produce three-dimensional graphics using the functions *surf*, *plot3 (useful for points)* or *mesh*.

```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
figure
mesh(X,Y,Z)
figure
surf(X,Y,Z)
```





In mathematics, the historical **unnormalized sinc function** is defined for $x \neq 0$ by

$$\operatorname{sinc}(x) = \frac{\sin(x)}{x}.$$

Customizing Graphical Effects

Generally, MATLAB's default graphical settings are adequate which make plotting fairly effortless. For more customized effects, use the *get* and *set* commands to change the behavior of specific rendering properties.

```
>> hp1 = plot(1:5)               % returns the handle of this line plot
>> get(hp1)                      % to view line plot's properties and their values
>> set(hp1, 'lineWidth')         % show possible values for lineWidth
>> set(hp1, 'lineWidth', 2)      % change line width of plot to 2
>> gcf                % returns current figure handle
>> gca                % returns current axes handle
>> get(gcf)           % gets current figure's property settings
>> set(gcf, 'Name', 'My First Plot')       % Figure 1 => Figure 1: My First Plot
>> get(gca)           % gets the current axes'  property settings
>> figure(1)          % create/switch to Figure 1 or pop Figure 1 to the front
>> clf                % clears current figure
>> close              % close current figure; "close 3" closes Figure 3
>> close all          % close all figures
```

Solving linear system of equations

- A system of equations

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2 \end{cases}$$

- can be written as a matrix equation

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \end{Bmatrix}$$

$$\boldsymbol{A}\,\vec{\boldsymbol{x}} = \vec{\boldsymbol{b}}$$

- and the solution is

$$\vec{\boldsymbol{x}} = \boldsymbol{A}^{-1}\,\vec{\boldsymbol{b}}$$

- In MATLAB, this operation is expressed as

$$\boldsymbol{x} = \boldsymbol{A}\backslash\boldsymbol{b}$$

or $\boldsymbol{x} = \boldsymbol{inv}(\boldsymbol{A}) * \boldsymbol{b}$

or $\boldsymbol{x} = \boldsymbol{linsolve}(\boldsymbol{A}, \boldsymbol{b})$

Solving linear system of equations

Example: solve the following set of linear equations

$$x + 2y + 3z = 2$$
$$x + y - z = 4$$
$$x + 2y + z = 4$$

Matrix form:

$$\boldsymbol{A}\,\vec{\boldsymbol{x}} = \vec{\boldsymbol{b}}$$

where

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & -1 \\ 1 & 2 & 1 \end{bmatrix} \quad \vec{\boldsymbol{b}} = \begin{Bmatrix} 2 \\ 4 \\ 4 \end{Bmatrix} \qquad \vec{\boldsymbol{x}} = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$$

```
>> A=[1 2 3;1 1 -1;1 2 1];b=[2;4;4];
x=A\b
```

```
x =

     1
     2
    -1
```

Solving basic optimization problems

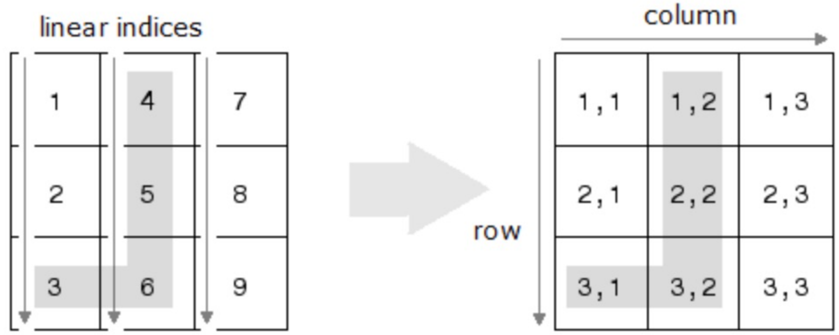
Example
Find: min $f(x_1, x_2) = x_1^2 + x_2^2$ at domain $-5 < x_1 < 5, -5 < x_2 < 5$

- Numerically
  - Look over the entire domain, calculate function value and pick up the minimum value

```
x1 = linspace(-5, 5, 100);
x2 = linspace(-5, 5, 100);
[X1, X2] = meshgrid(x1, x2);
f = X1.^2 + X2.^2;

[minValue, minIndex] = min(f(:));

figure;
surf(X1, X2, f);
hold on;
```

Solving basic optimization problems

```
[minX1, minX2] = ind2sub(size(f), minIndex);
plot3(x1(minX1), x2(minX2)
, minValue, 'ro', 'MarkerSize', 10,
'MarkerFaceColor', 'r');

xlabel('x1');
ylabel('x2');
zlabel('f(x1, x2)');
title('Plot of f(x1, x2) = x1^2 + x2^2');

fprintf('Minimum value of f(x1, x2): %f\n',
minValue);
fprintf('Corresponding x1 value: %f\n',
x1(minX1));
fprintf('Corresponding x2 value: %f\n',
x2(minX2));
```



Create input vectors and perform the conversion.

```
ind = [3 4 5 6];
sz = [3 3];
[row,col] = ind2sub(sz,ind)
```

General Tips for Projects

1. Translating the physical problem to a mathematical one, deriving a system of equations for force equilibriums.
   - Convert this system of equations into matrix form if necessary
2. Find the goal, and express in numerical form, such as finding the minimum value, matching certain criteria (a>0.001), etc.
3. Start writing code, write down all the variables, and code to solve the equations.
4. It is always better to work with parts of a code and check the outputs in the command window instead of working with the whole code in the script window
5. Use plots to display this data in a way that is easy to interpret.
6. Comment your code so the TA can easily understand your idea.
7. Write a report describing what you have found.

Additional Tutorial Materials for MATLAB

1. Introduction To Matlab For Engineering Students,
   by David Houcque at Northwestern University.

2. Experiments with MATLAB,
   by Cleve Moler, the inventor of MATLAB.
   (https://www.mathworks.com/moler/exm.html)

References
D. Houcque, Introduction to MATLAB for Engineering Students.
K. Tseng, Introduction to MATLAB.
R. Larsen and S. Hunt, Using MATLAB for Statics and Dynamics.

Prepared by
Department of Civil & Environmental Engineering, Northwestern University