

ENGR 0012 – Engineering Problem Solving

Goals for this week:

- Create functions
- Solve systems of equations
- Use stats commands
 - Create plots
- Find line of best fit


Please submit your HW!

Please submit through both new and old
submission systems


With functions, we don't need to use the same variable names

- To create a function, include this as the first line in the script you want to call and be able to use over and over:

```
function[list of variables]=function_name(list of variables)
```



Variables being passed
from the subprogram
to the main program –
use [] and separate
with commas



Variables being passed
from the main program
to the subprogram –
use () and separate with
commas

Let's try this!

Functions and linear equations

- Create a program that will call the following functions:
 - **Loading**: Function will ask user for name of file, and error check indefinitely. Then, it will load the file and get matrix A and array b from the data.
 - **Axb**: Function will use A and b to solve for x
 - From the main, display the resulting array x

Functions

Main

```
%Solve system of equations
clear
clc

%Call loading function
[A,b]=loading;

%Call Axb function
x=Axb(A,b);

%Display
disp('The resulting array x is: ');
disp(x);
```

```
function [MatrixA,Arrayb]=loading

%Get file name and error check
filename='0';

while exist(filename)==0
    filename=input('Please enter name of file: ','s');
end

%Load data
data=load(filename);

%Get MatrixA and Arrayb
matrix_dim=size(data);
rows=matrix_dim(1);
cols=matrix_dim(2);

MatrixA=data(:,1:rows);
Arrayb=data(:,cols);
```

Note that variable names can be different in the main and in the function; MATLAB looks at the order of variables, not the names!

```
%Solve for x
function xxx=Axb(AAA,bbb)
    xxx=inv(AAA)*bbb;
```

Practice Problem 1

- Write a MATLAB program that will compute a monthly loan payment (A). The main program should display the purpose of the program. It will then call two functions (so you will have three m-files).
- ***user_inputs*** should not receive any variables but should return the user-entered values for a principle amount of money to be borrowed (P), the number of months to pay the loan back (N), and a monthly interest rate (i)
 - Make sure the user is able to enter the interest value as a percent and have the program divide it by 100 for the calculation
- ***calculate_A*** should receive the three values entered above and return the payment (A) to the main program.
- The main program should then display the payment (A).

Try your program with the following values:

P=\$3000 N=30 i=1%

P=\$100,000 N=360 i=.5%

P=\$20,000 N=48 i=.2%

Answers you should get are:

116.24; 599.55; 437.40

$$A = P \left[\frac{i(1+i)^N}{(1+i)^N - 1} \right]$$

Submit zipped folder with .m files. Name folder

“Mena_Time_MyFunctions1Team#”

(“Mena_10am_MyFunctions1L01”), drop into Classwork folder

Practice Problem 2

Find the 5 numbers that satisfy the following:

- The sum of all the digits is 18
- The third digit is the sum of the first and second digits
- Subtracting the third digit from the fourth yields 4
- The 1st and 3rd digits added together give the fifth digit
- The first digit is twice the second digit

Create a data file (“MatData.txt”) with the data, and create an m-file that loads the data and solves this problem

Name your file Mena_time_SolveTeam#
(for example: “Mena_10am_SolveTeamL01”),
and submit to Classwork folder

How can we check for errors in the data?

- We know A should be a square matrix
 - $\text{num_rows} == \text{num_cols}$
- We know equations should be independent of each other
 - $\det(A) \neq 0 \rightarrow$ you have independent equations
- How would you include this in an m-file? (See next slide)

```
Editor - C:\Users\imena\Documents\MATLAB\solveAxb.m*
solveAxb.m* x +
1 - clear
2 - clc
3
4 %Load data
5 - load MatData.txt
6
7 %Get matrix dimensions
8 - MatDim=size(MatData);
9
10 %Get num rows and num columns
11 - numRows=MatDim(1);
12 - numCols=MatDim(2);
13
14 %Check for square matrix
15 - if numRows==numCols-1
16     %Extract data into A and b
17 -     A=MatData(:,1:numRows);
18 -     b=MatData(:,numCols);
19     %Check for independence of equations (det~=0)
20 -     if det(A)~=0
21         %Solve for x
22 -         x=inv(A)*b
23     else
24 -         disp('Error in file!')
25     end
26 else
27 -     disp('Error in file!')
28 - end
```

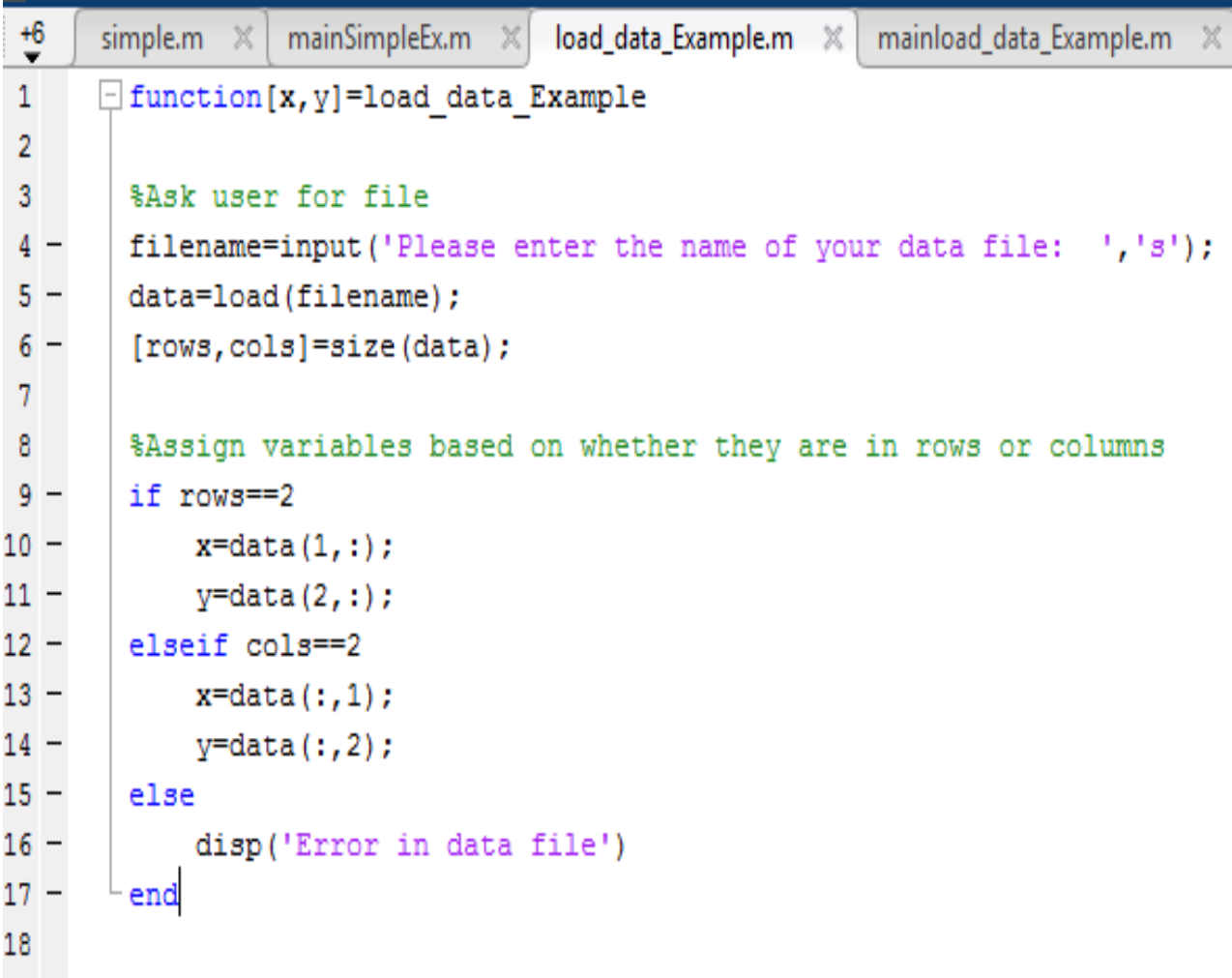

Some commands to plot in MATLAB:

- `plot`
- `xlabel`
- `ylabel`
- `title`
- `text`
- `gtext`

```
Editor - C:\Users\imena\Documents\MATLAB\PlottingExample.m
Untitled* x PlottingExample.m x +
1 - x=[1:10];
2 - y=[linspace(5,50,10)];
3
4 %Plot x and y - default is blue line
5 - plot(x,y)
6
7 %Plot x and y - red stars
8 - plot(x,y,'r*')
9
10 %Add labels and title
11 - xlabel('X Value')
12 - ylabel('Y Value')
13 - title('My Plot')
14
15 %Add text
16 - text(5,10,'Adding text')
17 - gtext('Adding more text')
```

Data can be stored “in rows” or “in columns” – ideally, our program should work either way!

Data can be stored “in rows” or “in columns” – ideally, our program should work either way!

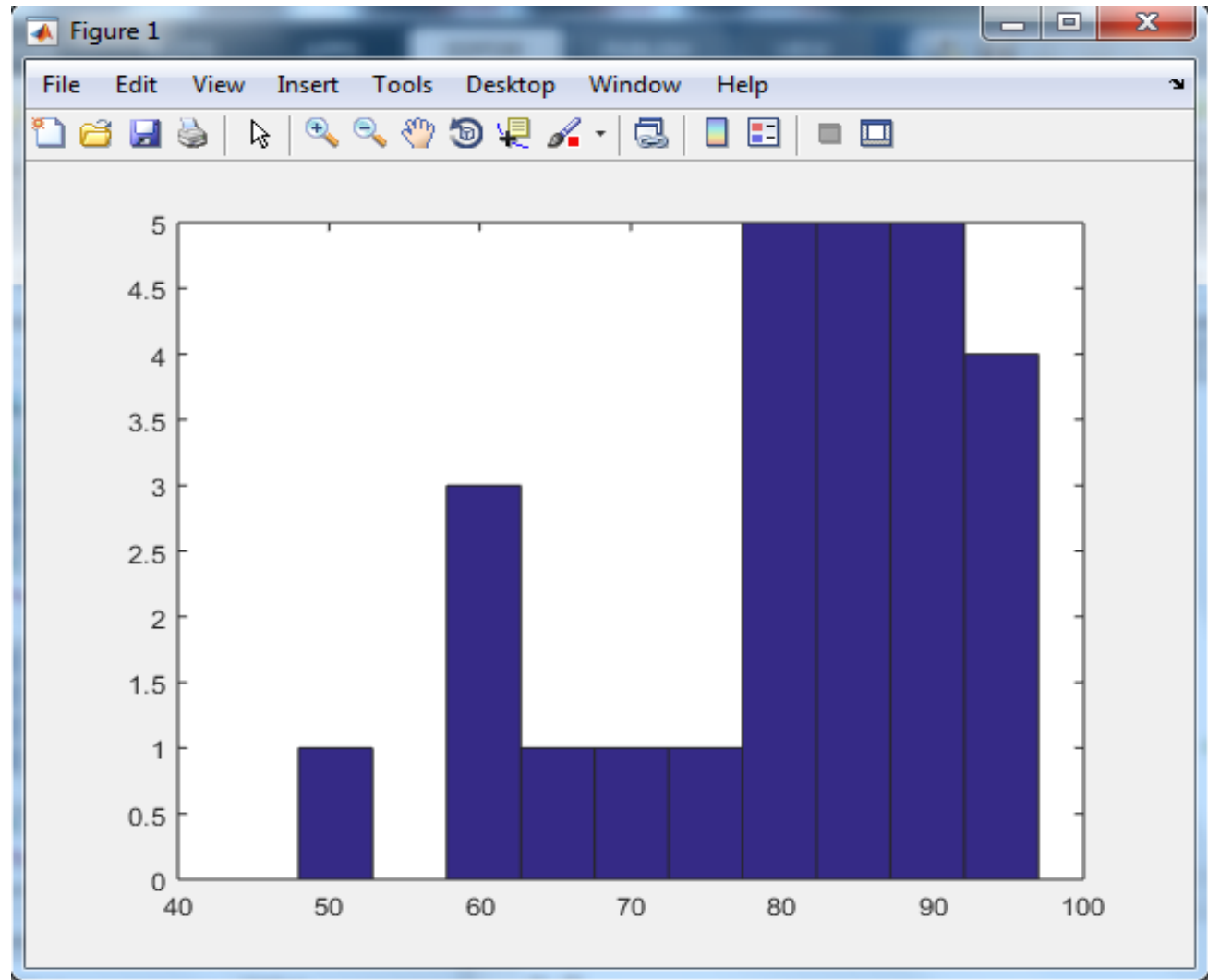


```
+6 simple.m x mainSimpleEx.m x load_data_Example.m x mainload_data_Example.m x
1 function[x,y]=load_data_Example
2
3 %Ask user for file
4 filename=input('Please enter the name of your data file: ','s');
5 data=load(filename);
6 [rows,cols]=size(data);
7
8 %Assign variables based on whether they are in rows or columns
9 if rows==2
10     x=data(1,:);
11     y=data(2,:);
12 elseif cols==2
13     x=data(:,1);
14     y=data(:,2);
15 else
16     disp('Error in data file')
17 end
18
```

Some statistics commands in MATLAB:

- mean
- std
- var
- min
- max
- hist

A histogram is a vertical bar graph used to make a picture of the data based on the frequency in various intervals



Let's try this!

- Create data file with exam grades
- Create a program that will plot a histogram with the default 10 bins, and a histogram with a user-provided number of bins, side by side
- The program should display the number of elements in each bin, for each histogram

```
clear
clc

%Load data
load exam_grades.txt

%To create histogram (default: 10 bins)
%hist(exam_grades)

%To find the number of elements in each bin, make it equal to a variable
n_10=hist(exam_grades)|

%Ask user how many bins
bin_num=input('How many bins would you like? ');

%Create histogram with number of bins user selected
%hist(exam_grades, bin_num)

%Find number of elements in each bin
n_user=hist(exam_grades,bin_num)

%Plot histograms side by side
subplot(1,2,1)
hist(exam_grades)
subplot(1,2,2)
hist(exam_grades,bin_num)
```


Practice Problem

- Create a program that will do the following:
- Call function **load_file**, which will ask the user to enter a data file. This function will check if data is in rows or columns, and extract variables x and y.
- Call function **x_stats**, which will find the min, max, and mean of x, and display these results from the function
- Call function **xy_plots**, which will create a 2x1 subplot, with the scatter plot (`scatter`) of x and y in the first plot, and a histogram with 10 bins of y in the second plot. Ask the user to provide a title for each subplot, and display the title
- Use data file practice1.txt – in my Unix public folder

Submit zipped folder with .m files. Name folder
"Mena_Time_StatsTeam#" ("Mena_10am_StatsL01"),
drop into Classwork folder

Main

```
%Practice
clear
clc

%Call function to load data
[x,y]=load_file;

%Call function for x stats
x_stats(x);

%Call function to plot
xy_plots(x,y)
```

Functions

```
function [x,y]=load_file

%Get file name
filename='0';

while exist(filename)==0
    filename=input('Please enter the name of the data file: ', 's');
end

data=load(filename);
[rows,cols]=size(data);

%Get x and y, check if in rows or cols
if rows==2
    x=data(1,:);
    y=data(2,:);
elseif cols==2
    x=data(:,1);
    y=data(:,2);
else
    disp('Error in data file!')
end
```

```
function x_stats(x)

%Finds stats
min_x=min(x);
max_x=max(x);
mean_x=mean(x);

disp(['The min of x is ', num2str(min_x)])
disp(['The max of x is ', num2str(max_x)])
disp(['The mean of x is ', num2str(mean_x)])
```

```
function xy_plots(x,y)

titleP=input('Title for plot: ', 's');
titleH=input('Title for histogram: ', 's');
%Create subplots
subplot(2,1,1)
scatter(x,y)
title(titleP)
subplot(2,1,2)
hist(y)
title(titleH)
```

When asking for user input, how can we make it error proof?

Consider:

1. Clearly specify possible responses to user

```
clear
clc
%Checking for user input error
%(1) Clearly specify possible responses to user:
a=input('Provide a number between 1 and 5 for a: ');
b=3;
```

When asking for user input, how can we make it error proof?
Consider:

2. Use AND/OR operators in if statements to account for different possible ways of entering response

```
clear
clc
%Checking for user input error
%(1) Clearly specify possible responses to user:
a=input('Provide a number between 1 and 5 for a: ');
b=3;

%(2) Use AND/OR operators
while (a<1 || a>5)
    disp('Error!');
    a=input('Provide a number between 1 and 5 for a: ');
end

c=a+b
```

When asking for user input, how can we make it error proof?
Consider:

3. Use a menu with a switch-case

```
Editor - C:\Users\imena\Documents\MATLAB\SwitchCaseExample.m
HistogramExample.m x PlottingExample.m x SwitchCaseExample.m x +
1 - x=[1:10];
2 - y=[linspace(5,50,10)];
3
4 %Use an if statement to get user input and plot accordingly
5 - color = input('What color of data points would you like (Type r for red...)? ', 's')
6 - if color=='r'
7 -     disp('Your plot will use the color red.')
8 -     plot(x,y,'r*')
9 - elseif color=='g'
10 -     disp('Your plot will the color green.')
11 -     plot(x,y,'g*')
12 - else
13 -     disp('Your plot will use the color yellow')
14 -     plot(x,y,'y*')
15 - end
```

```
Editor - C:\Users\imena\Documents\MATLAB\SwitchCaseExample.m
HistogramExample.m x PlottingExample.m x SwitchCaseExample.m x +
1 - x=[1:10];
2 - y=[linspace(5,50,10)];
3
4
5 %Using switch-case-otherwise
6 %Doesn't check for true or false; checks if something is an exact match
7 %Cannot use < or >
8 - datacolor=menu('What color?', 'yellow', 'magenta', 'red')
9 - switch datacolor
10 -     case 1
11 -         plot(x,y,'y*')
12 -     case 2
13 -         plot (x,y,'m*')
14 -     otherwise
15 -         plot (x,y,'r*')
16 - end
17 %Note that the output to a menu is always a number, which corresponds to the list of possible answers
18
19
```

2.54 3.175 3.81 5.08 6.35 7.62 8.89 10.16 12.7 15.24 20.32

Practice Problem

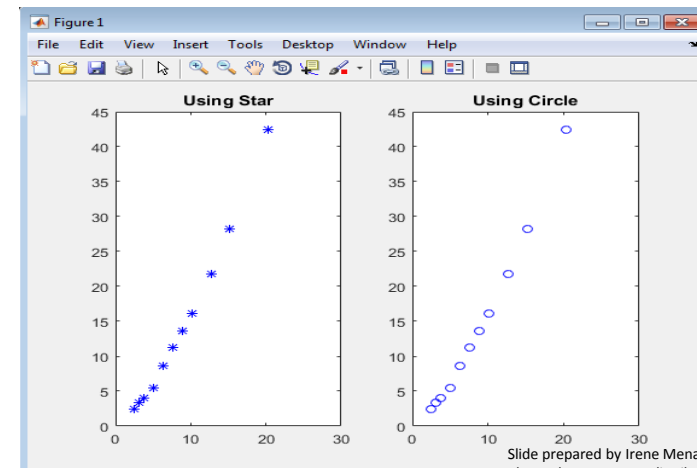
2.50 3.38 4.05 5.43 8.62 11.28 13.56 16.06 21.75 28.22 42.48

(Store the data in two separate rows)

Write a MATLAB script to plot the data above. The script must:

- Allow the user to enter the name of a data file
- Assign variables `x` and `y`
- Allow the user to pick one of five colors for the plot symbol (use menu and switch-case)
- Use `subplot` command to plot:
 - `x` vs. `y` using a star symbol and the user's chosen color
 - `x` vs. `y` using a circle symbol and the user's chosen color
- Add titles: "Using Star" and "Using Circle"

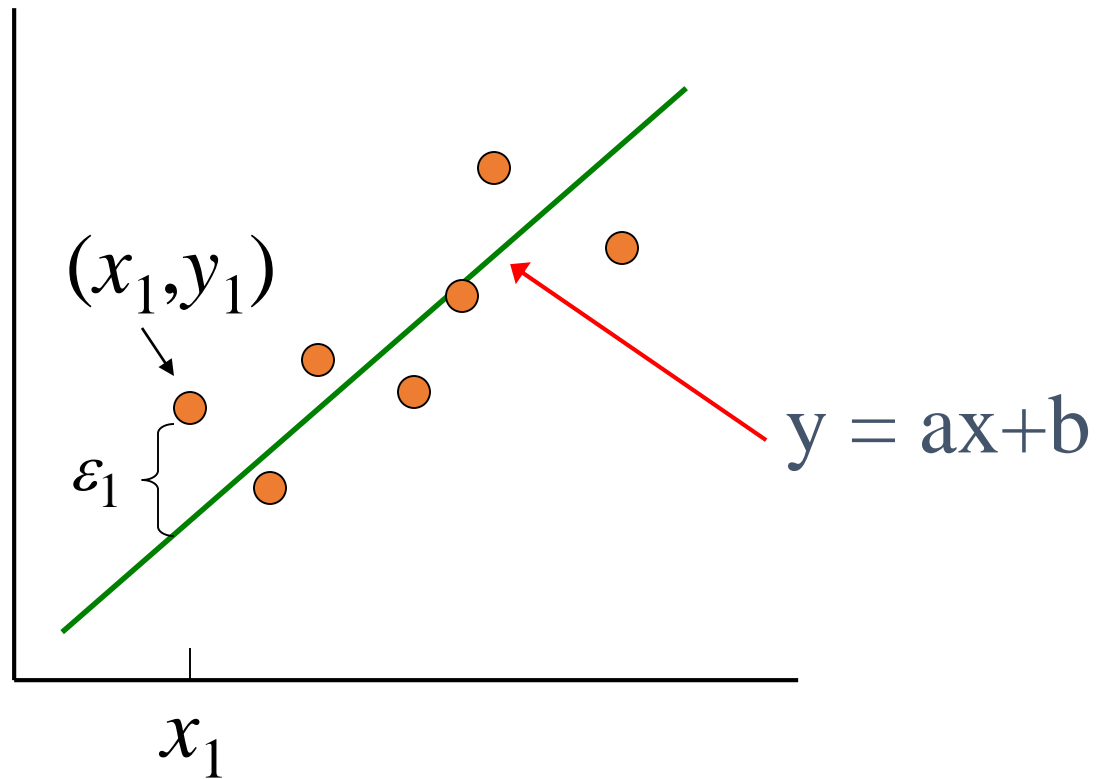
Submit .m file named "Mena_Time_MenuTeam#" ("Mena_10am_MenuL01"), drop into Classwork folder



Line fitting is a commonly-used engineering analysis tool

- You have a series of points
- You fit a line to it
- You find the line $y=ax+b \rightarrow$ this line represents the data

This is what it looks like:



Remember that the objective of regression analysis is to use information about one variable to study another

- Simple linear regression involves fitting a straight line through a sample set of points for x and y to obtain the coefficients for the equation $y = ax + b$
- x is considered the “independent variable” and y is the “dependent variable”. a is the slope and b is the y -intercept

The formulas to find the least squares estimates (for getting the “best fit” line) are derived from solving linear equations

$$a \sum_{i=1}^n x_i + bn = \sum_{i=1}^n y_i$$

$$a \sum_{i=1}^n x_i^2 + b \sum_{i=1}^n x_i = \sum_{i=1}^n x_i y_i$$

If a and b are our unknowns, can you convert this to a system of equations and to the format $Ax=b$?

The formulas to find the least squares estimates (for getting the “best fit” line) are derived from solving linear equations

a and b can be found by using these formulas and matrix multiplication

$$a \sum_{i=1}^n x_i + bn = \sum_{i=1}^n y_i$$

$$a \sum_{i=1}^n x_i^2 + b \sum_{i=1}^n x_i = \sum_{i=1}^n x_i y_i$$

$$\begin{matrix} A & & X & & B \\ \left[\begin{array}{cc} \sum_{i=1}^n x_i & n \\ \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i \end{array} \right] & \begin{bmatrix} a \\ b \end{bmatrix} & = & \begin{bmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{bmatrix} \end{matrix}$$

$$X = A^{-1} * B$$

X is a 2x1 array containing the values of a and b (slope and intercept of the best-fit line)

Let's try this!

Given these data points, what is the line?

$x=[0, 1, 2, 3, 4];$

$y=[0.2, 1.1, 3.9, 8.8, 16.4];$

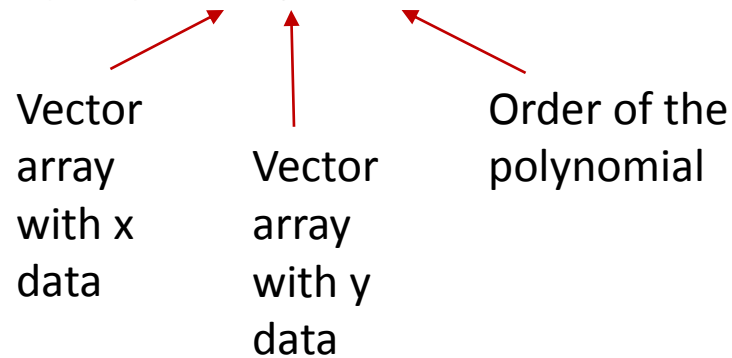
$$\begin{array}{ccc} \text{A} & \text{x} & \text{B} \\ \left[\begin{array}{cc} \sum_{i=1}^n x_i & n \\ \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i \end{array} \right] & \left[\begin{array}{c} a \\ b \end{array} \right] & = \left[\begin{array}{c} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{array} \right] \end{array}$$

(The long way of finding
the coefficients!)

```
clear
clc
%declare variables
x=[0,1,2,3,4];
y=[0.2 1.1 3.9 8.8 16.4];
%length of x
n=length(x);
%Initiate running sums
xisum=0;
yisum=0;
xisquaresum=0;
xiyisum=0;
%Use loops to get sums
for i=1:n
    xisum=xisum+x(i);
    yisum=yisum+y(i);
    xisquaresum=xisquaresum+x(i)^2;
    xiyisum=xiyisum+(x(i)*y(i));
end
%Create A and b
A=[xisum n;xisquaresum xisum];
b=[yisum; xiyisum];
%Solve for a and b
abSolution=inv(A)*b;
%Display results
disp(['The slope is ',num2str(abSolution(1))]);
disp(['The intercept is ',num2str(abSolution(2))]);
```

MATLAB finds the coefficients using the command `polyfit`

- The syntax is: `polyfit(x,y,order)`



- For linear regression, the order = 1
 - `polyfit(x, y, 1)` ← A line is a first order polynomial

`polyfit` returns two values in a 1x2 array:
[slope,intercept]

- You can assign the output coefficient array to a variable:
`LineCoeffs = polyfit(x, y, 1)`

- The equation which best fits the data is thus
 $y = ax + b = \text{LineCoeffs}(1)x + \text{LineCoeffs}(2)$

$$a = \text{slope} = \text{LineCoeffs}(1) \quad b = \text{y-intercept} = \text{LineCoeffs}(2)$$

Now, we can plot the line

- We need an array of **x** values and corresponding **y** values based on the best fit equation

- Try this:

```
yfit = LineCoeffs(1)*x + LineCoeffs(2)
```

```
plot(x, y, 'r*', x, yfit, '—')
```

Two things were plotted!

Another way to do this:

```
plot(x,y,'r*')
```

```
hold on
```

```
plot(x, yfit, '—')
```

```
hold off
```

```

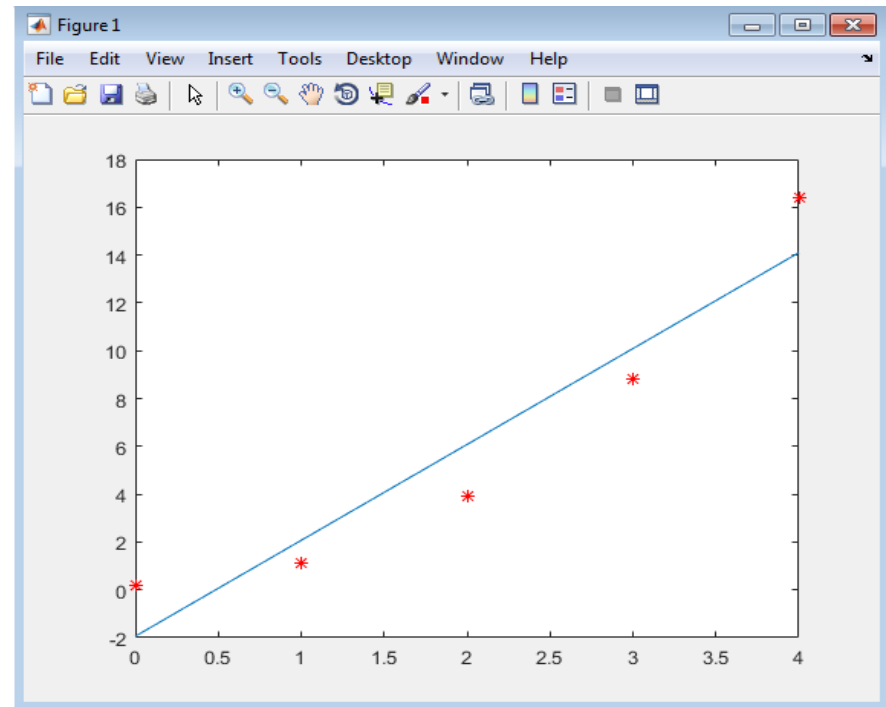
clear
clc
%declare variables
x=[0,1,2,3,4];
y=[0.2 1.1 3.9 8.8 16.4];

%Using polyfit
LineCoeffs=polyfit(x,y,1);

%Finding y values using line equation (notice vfit is an array)
vfit=LineCoeffs(1)*x+LineCoeffs(2)

%Plot the original points and the fitted line
plot(x,y,'r*',x,vfit,'-')

```



You can also use the `polyval` command to generate the best fit y array

$$y_{\text{new}} = \text{polyval}(\text{LineCoeffs}, x)$$

- The first argument is the vector array containing the coefficients of the line
- The second argument is a vector of x values at which we want to evaluate the function
- The output is a y vector of the same dimension as the x vector (equal to yfit)

You can also use the `polyval` command to calculate a value for any given x

So, if x_i represents the values of x we want to get a y for:

$$x_i = [14 \ 16]$$

We can use `polyval` to get the values of the corresponding y :

$$y_i = \text{polyval}(\text{LineCoeffs}, x_i)$$

You can add the equation to the plot as follows:

- `gtext(['y = ', num2str(LineCoeffs(1)), '*x + ', num2str(LineCoeffs(2))])`
- OR
- `string1=['y =', num2str(LineCoeffs(1)), '*x + ', num2str(LineCoeffs(2))];`
- `text(2,2,string1)`

2.54 3.175 3.81 5.08 6.35 7.62 8.89 10.16 12.7 15.24 20.32

2.50 3.38 4.05 5.43 8.62 11.28 13.56 16.06 21.75 28.22 42.48

(Store data in two separate rows)

Let's try this!

1. Load the data and extract x and y data into two vectors
2. Using `polyfit`, determine the coefficients of a line which best fits the data
3. Generate the best fit line by creating a set of data points from the coefficients determined by `polyfit`
4. Plot both the data and the best fit line on one graph
 - Use a menu to ask the user what color to plot the data points and the line (give three options and use the same color for both points and line)
5. Add a title to your plot
6. Use `gtext` to add your group number and names to the plot

Submit .m file called
"Mena_Time_PolyfitPlot_Team#"
("Mena_10am_PolyfitPlot_L01")
into Classwork folder