MATH2221
Mathematics Laboratory II

Lecture 5: Remarks on Conditional Statements and Visualization Using MATLAB

Gary Choi

February 11, 2025

Recall: *for* statements (*for* loop)

- ***for*** **loop: loop a set of statements repeatedly**

```
for i = (a given vector)
        statements;          % Do this for every element i in the vector
        …
end
```

Example: Compute $1^2 + 3^2 + 5^2 \ldots + 99^2$

```
s = 0;
for i = 1:2:99
        s = s + i^2;
end
s
```

Example: Compute $1^1 \cdot 2^{1/2} \cdot 3^{1/3} \cdot \cdots \cdot 100^{1/100}$

```
s = 1;
for n = 1:100
        s = s*n^(1/n);
end
s
```

Example (recurrence):
If $F_1 = 1$, $F_2 = 1$ and $F_n = F_{n-1} + F_{n-2}$ for all $n \geq 3$, find F_{20}.

```
F = zeros(20,1);
F(1) = 1;
F(2) = 1;
for n = 3:20
   F(n) = F(n-1)+F(n-2);
end
F(20)
```

Recall: *while* statements (*while* loop)

- ***while*** **loop: repeat the statements until the condition is NOT satisfied**

```
while condition
    statements  % repeating the statements while the condition is true (1)
end
```

  - It is easy to get into an endless loop if the while loop is set up improperly!
  - Remember to update all relevant variables in each while loop
  - Use Ctrl+C to stop the iterations if needed

Example: Find the smallest n such that $1 + 2 + \cdots + n > 1000$

```
s = 0;
n = 0;
while s <= 1000
        n = n + 1;
        s = s + n;
end
n
```

Example: If $F_1 = 1$, $F_2 = 1$ and $F_n = F_{n-1} + F_{n-2}$ for all $n \geq 3$, find the smallest n such that $F_n \geq 1000$.

```
F = [1,1];
n = 2;
while F(n) < 1000
        n = n + 1;
        s = F(n-1)+F(n-2);
        F = [F, s];
end
n
```

Recall: Other considerations about loops

- *for* loops vs *while* loops:
  - *for* loops are usually used when you need to repeat some operations for a fixed number of times
  - *while* loops are usually used when you don't know how many times you need to repeat but have a known stopping criterion

- Nested loops

- Efficiency issue:
  - Loops are usually slower than vector-based operations! Important to optimize your code for better performance

Example:

A = (1:10000).^2 + 1; vs

```
A = zeros(10000,1);
for i = 1:10000
        A(i) = i.^2+1;
end
```

  - You can use tic; … (your statements); toc; to evaluate the computational time

The *break* command

- *break* terminates the execution of a *for* or *while* loop
  - Statements in the loop after the break statement do not execute.
  - In nested loops, *break* only break one loop

```
for ...
        …
        while …
                …
                if condition
                        break;
                end
                …
        end
        …
end
```

The *break* command

- Example (determine when the balance is tripled under compound interest of 8% annually):

```
balance = 1000;      % the initial balance
for year = 1:10000  % just set an arbitrarily large number here
        balance = balance * 1.08;
        if balance >= 3000
                % break the loop once the balance is tripled
                break;
        end
end

year                % output the year for which the balance is tripled
```

The *continue* command

- *continue* forces the program to skip the remaining part of the current *for/while* loop and continue with the next iteration

- Example:

```
a = 0;
while a < 6
        a = a + 1;
        if a == 3
                % skip the remaining part of the current loop
                % and continue with the next iteration
                continue;
        end
        disp(a); % display a
end
```

Result:

1

2

4

5

6

Recursion

- Idea: Create a function that calls itself during its own execution

- The function should consist of:
  - A base case for the final step/lowest level (otherwise it may loop endlessly)
  - Some way to proceed to the next level

- Example: Compute $n! = n \cdot (n-1) \cdot \cdots \cdot 2 \cdot 1$

```
function s = myFactorial(n)
if n <= 1
        s = 1; % the base case
else
        % proceed to the (n-1)! problem
        s = n * myFactorial(n-1);
end
end
```

If you run myFactorial(5), then it will do:

```
 myFactorial(5)
= 5 * myFactorial(4)
= 5 * 4 * myFactorial(3)
= 5 * 4 * 3 * myFactorial(2)
= 5 * 4 * 3 * 2 * myFactorial(1)
= 5 * 4 * 3 * 2 * 1 (the base case)
= 5!
```

Multiple ways to perform the same task

- In many cases, there are multiple ways to perform the same task
  Example: Compute $1^1 \cdot 2^{1/2} \cdot 3^{1/3} \cdot \dots \cdot 100^{1/100}$

Method 1 (*for* loop):

```
s = 1;
for n = 1:100
        s = s*n^(1/n);
end
s
```

Method 2 (*while* loop):

```
s = 1;
n = 1;
while n <= 100
        s = s*n^(1/n);
        n = n+1;
end
s
```

Method 3 (vectorized):

```
v = 1:100;
s = prod(v.^(1./v))
```

Method 4 (recursion):

First create a function:

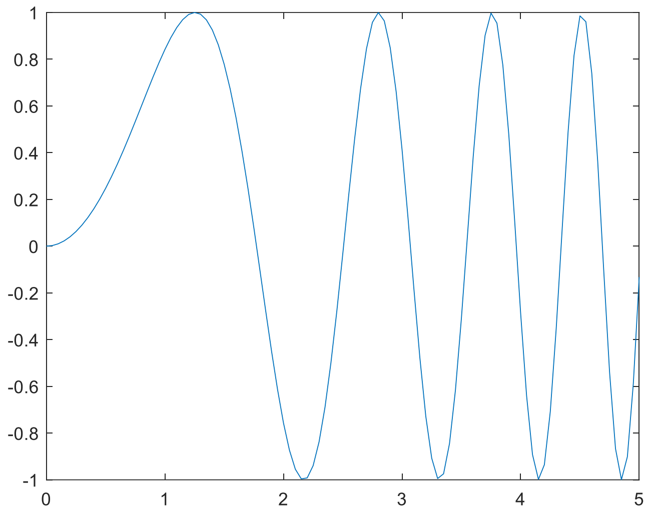
```
function s = myproduct(n)
if n <= 1
        s = 1;
else
        s = n^(1/n) * myproduct(n-1);
end
end
```

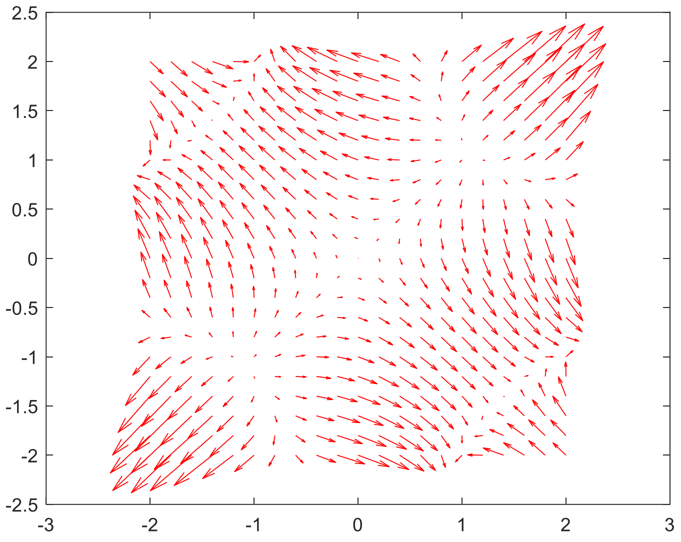
Then we can run the following command:

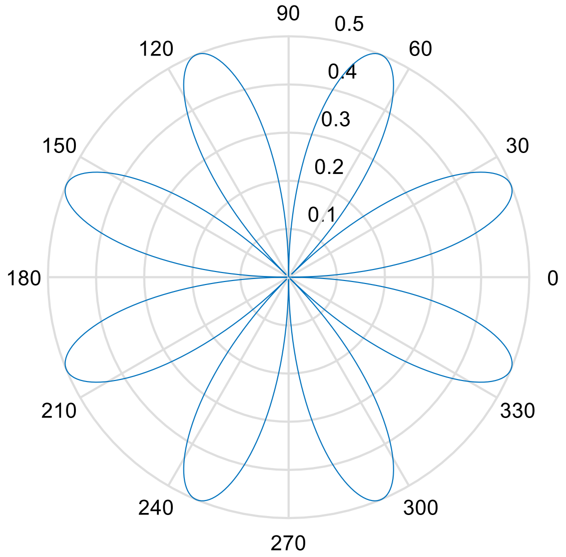
```
>> myproduct(100)
```

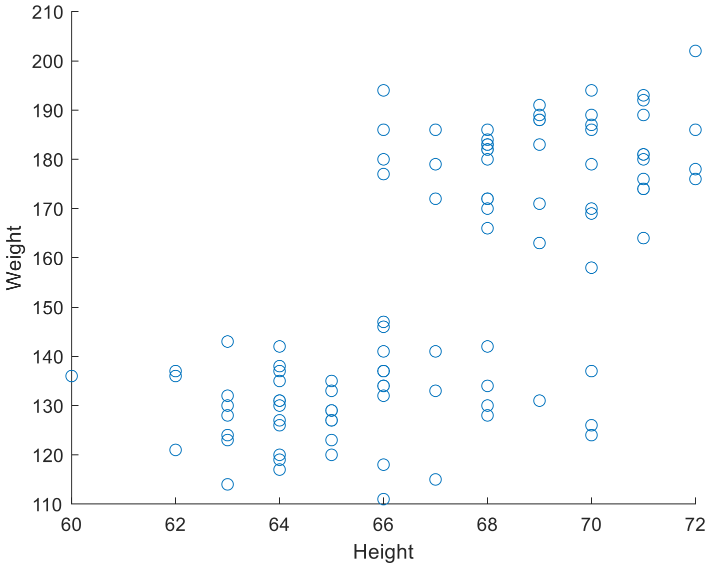
Next topic: Visualization using MATLAB

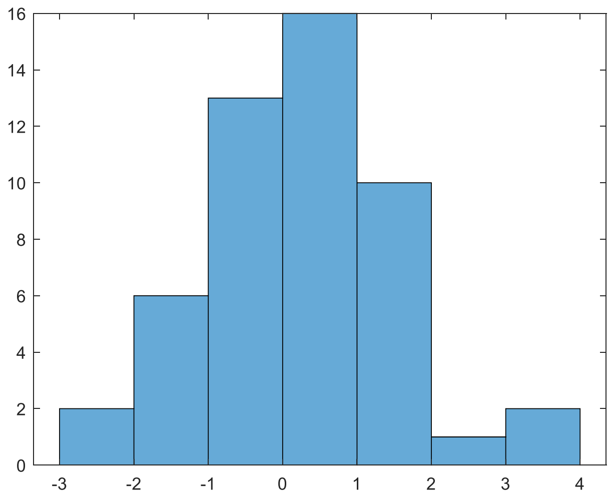
- How can we create plots using MATLAB?











2D visualization using MATLAB

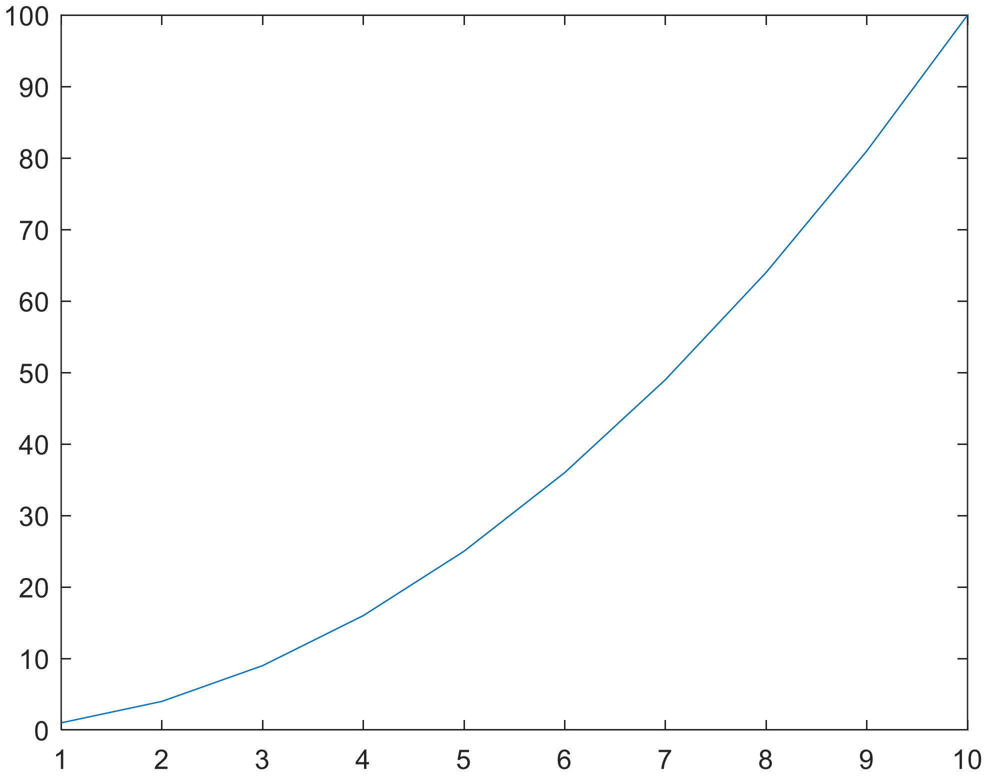
- Basic commands:
  - figure: create a new figure window
  - plot(x,y): plot the points x,y (can be vectors) in the current figure window
  - Use cursor to zoom in/out, drag etc.

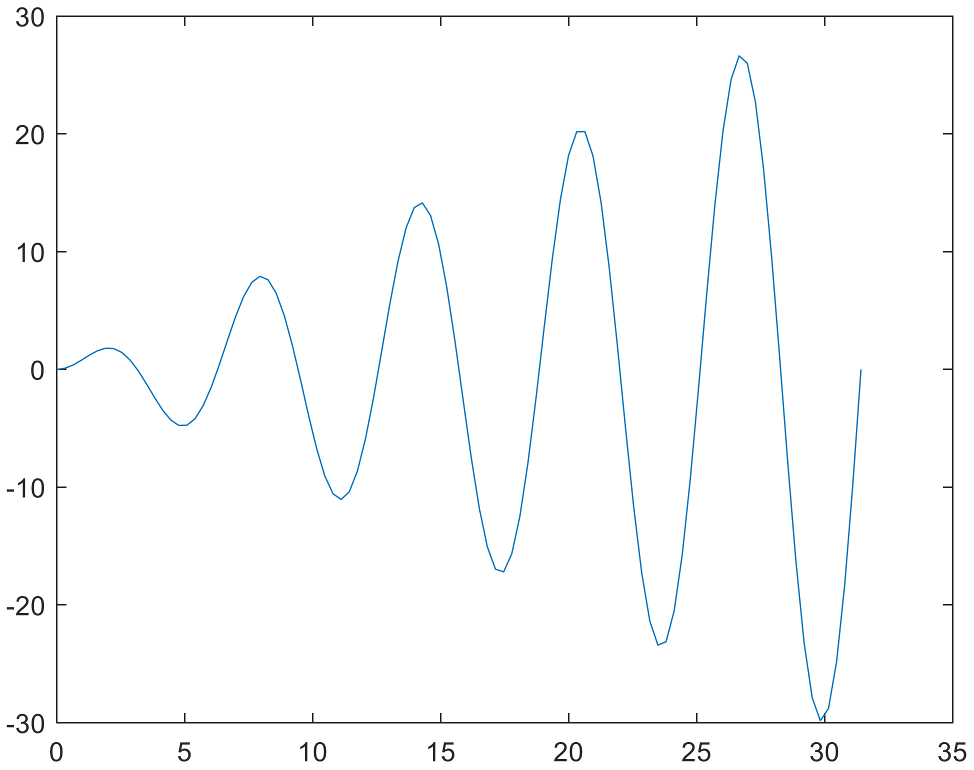
Example:

```
t = linspace(0,10*pi,100);
figure;
plot(t, t.*sin(t));
```

Example:

```
x = 1:10;
y = x.^2;
figure;
plot(x,y);
```





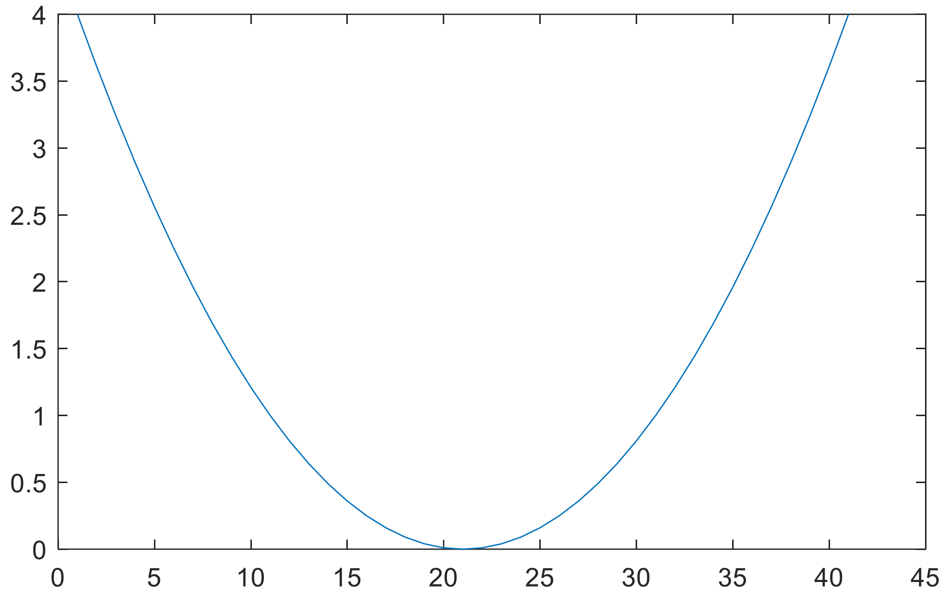
An overview of useful plotting commands

- Graph generation commands
  - Plot different types of graphs
  - Standard xy plot: plot
  - Semilog plots: semilogx, semilogy
  - loglog plots: loglog
- Management commands
  - Manage the figure windows
  - Create a new window: figure
  - Divide a window into subplots: subplot
  - Plot multiple curves on the same plot: hold (on/off)
- Annotation commands
  - Format the graphs
  - Add title and axis labels: title, xlabel, ylabel
  - Add text and figure legend: text, legend
  - Add box and grid lines: box (on/off), grid (on/off)

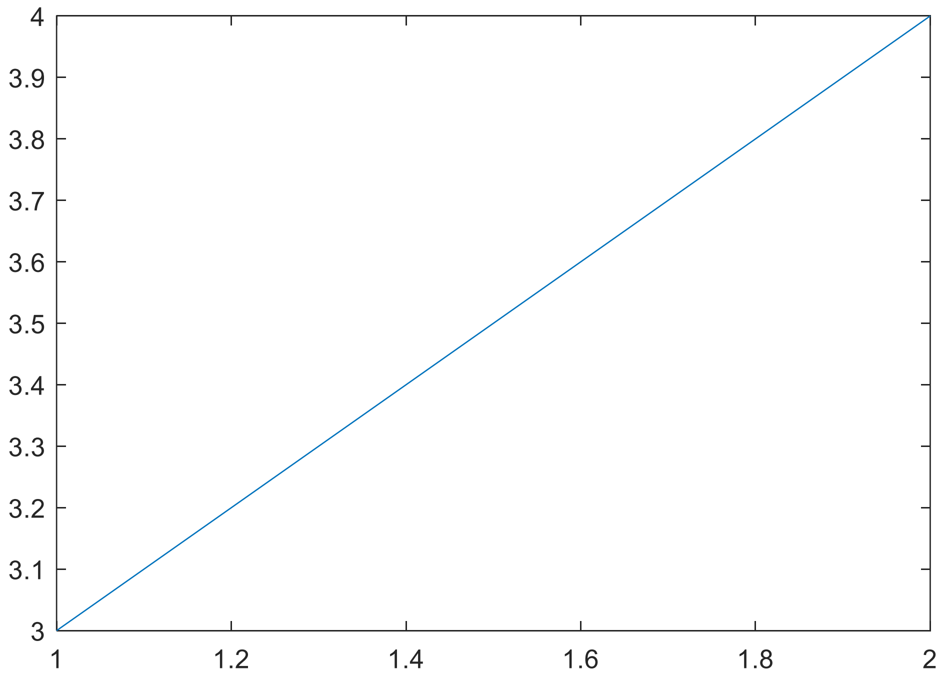
Graph generation	Management	Annotation
plot	figure	title
semilogx semilogy	hold on hold off	xlabel ylabel
loglog	subplot	text
polar	close	grid
fill	axis	legend
histogram	view	box
pie	rotate	set

The *plot* function

- plot(y) (where y is a vector):
  - **Plot every element** y(i) **versus their index number** i
  - Example:
    figure;
    plot((-2:0.1:2).^2);



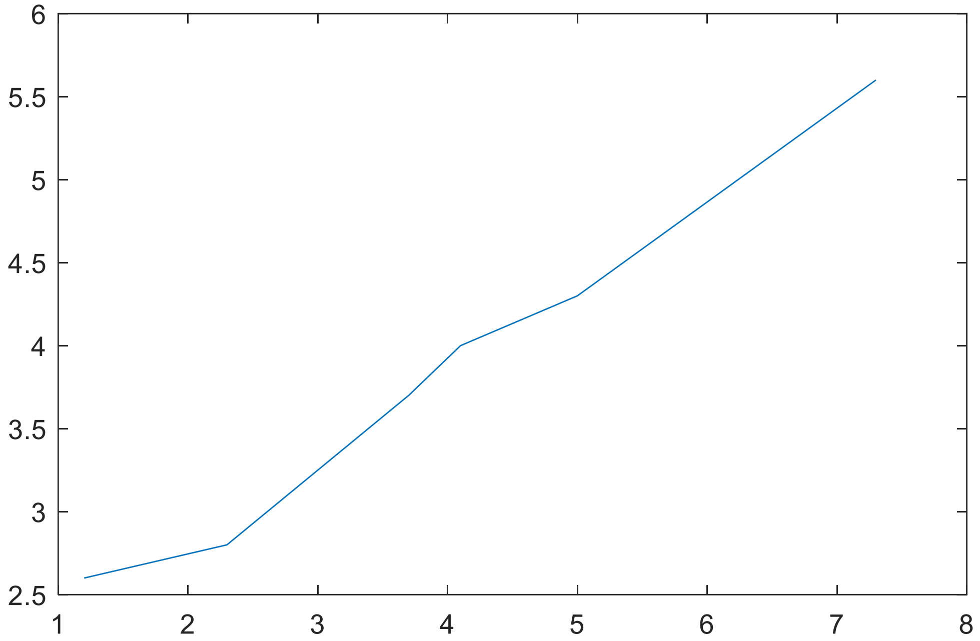
- plot([x1, x2] , [y1, y2]) (where x1, x2, y1, y2 are scalar):
  - **Plot a straight line** from (x_1, y_1) to (x_2, y_2)
  - Example:
    figure;
    plot([1,2], [3,4])



The *plot* function

- plot(x,y) (where x and y are vectors):
  - **plots a set of straight lines** connecting $(x_i\,, y_i)$ and $(x_{i+1}\,, y_{i+1})$ for all i
  - Example:

```
x = [1.2, 2.3, 3.7, 4.1, 5.0, 7.3];
y = [2.6, 2.8, 3.7, 4.0, 4.3, 5.6];
figure;
plot(x,y);
```



- **More general command**: plot(x,y, 'string')
  - The string encodes the color, marker and line type, e.g. 'r*-' means red, asterisk at each data point, and joining the points by a solid line.
  - See: https://www.mathworks.com/help/matlab/ref/plot.html

The *plot* function: some common line and marker styles

Command	Description	Result
'-'	Solid line	———
'--'	Dashed line	- - - - -
':'	Dotted line	···········
'-.'	Dash-dotted line	-·-·-·-
'o'	Circle	○
'+'	Plus sign	+
'*'	Asterisk	*
'. '	Point	·
'x'	Cross	×
'_'	Horizontal line	−
'|'	Vertical line	|

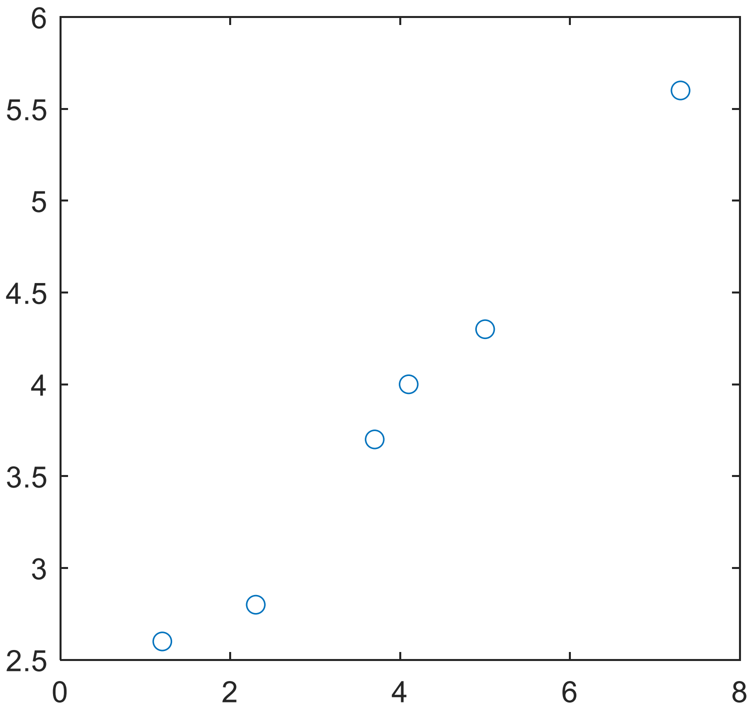
Command	Description	Result
'square'	Square	□
'diamond'	Diamond	◇
'^'	Upward-pointing triangle	△
'v'	Downward-pointing triangle	▽
'>'	Right-pointing triangle	▷
'<'	Left-pointing triangle	◁
'pentagram'	Pentagram	☆
'hexagram'	Hexagram	✡

See:
https://www.mathworks.com/help/matlab/creating_plots/specify-line-and-marker-appearance-in-plots.html

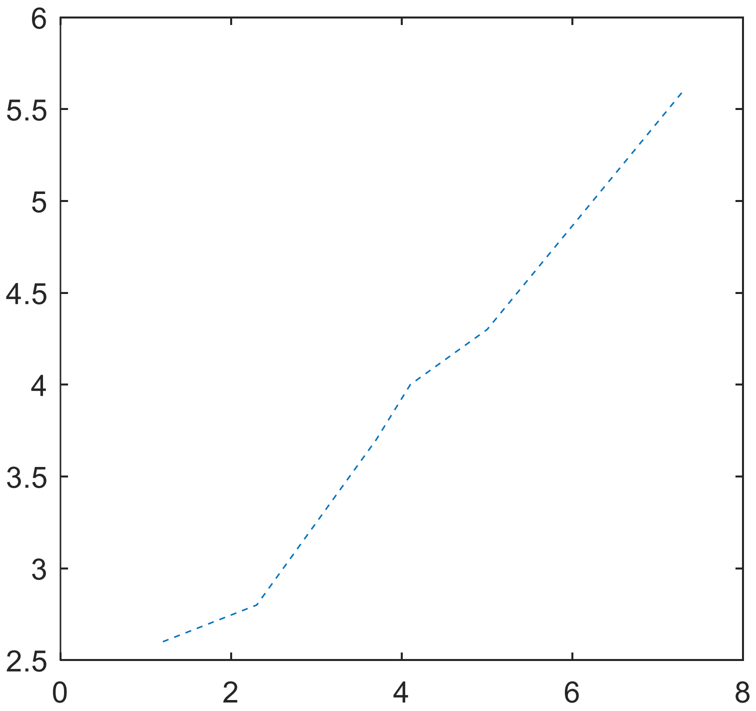
The *plot* function: some common line and marker styles

- Example:
  x = [1.2, 2.3, 3.7, 4.1, 5.0, 7.3];
  y = [2.6, 2.8, 3.7, 4.0, 4.3, 5.6];
  figure;

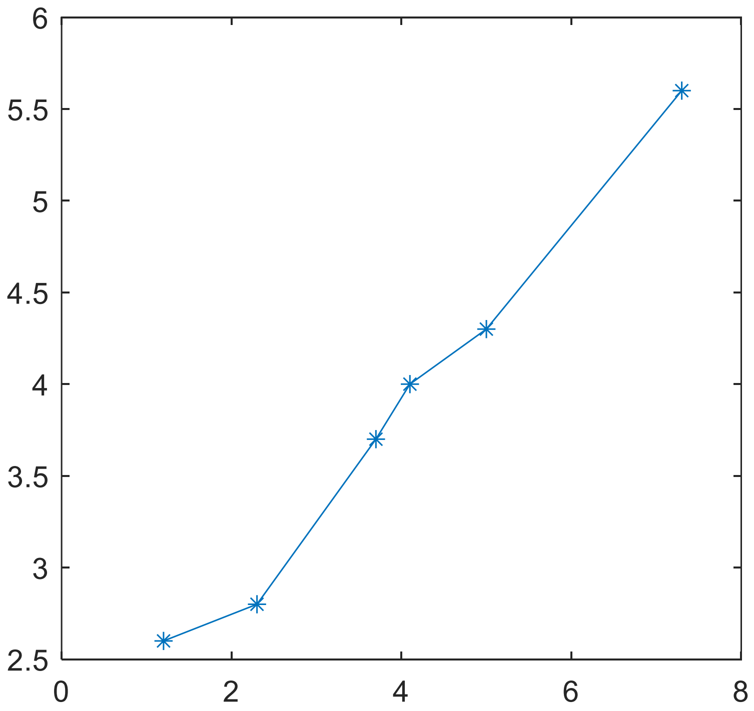
plot(x,y,'o')



plot(x,y,'--')



plot(x,y,'*-')



The *plot* function: changing the marker/line color

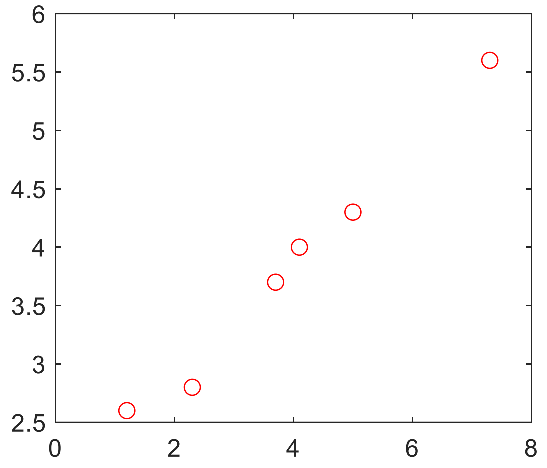
- There are 8 default color options in MATLAB:

Option	Color	RGB	Result
'r'	Red	[1,0,0]	
'g'	Green	[0,1,0]	
'b'	Blue	[0,0,1]	
'c'	Cyan	[0,1,1]	
'm'	Magenta	[1,0,1]	
'y'	Yellow	[1,1,0]	
'k'	Black	[0,0,0]	
'w'	White	[1,1,1]	

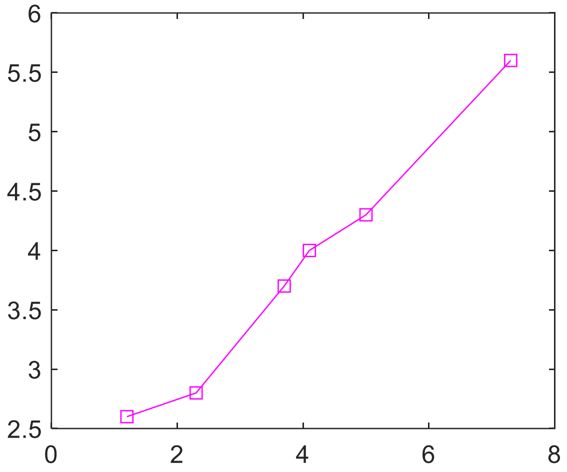
- We can also use other colors by specifying the RGB values (next page)

Example:

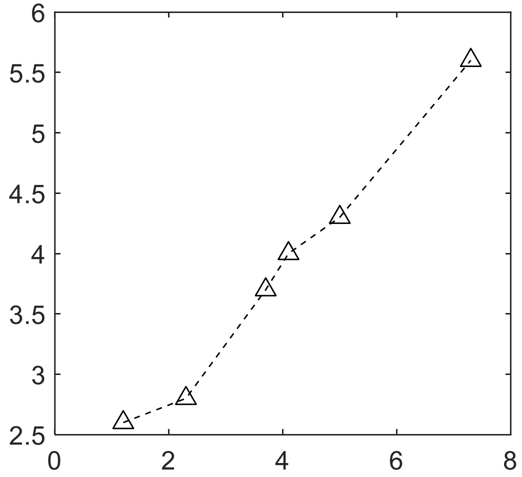
plot(x,y,'ro')



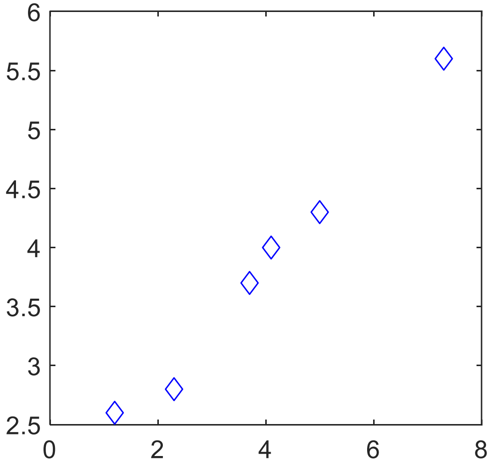
plot(x,y,'ms-')



plot(x,y,'k^--')

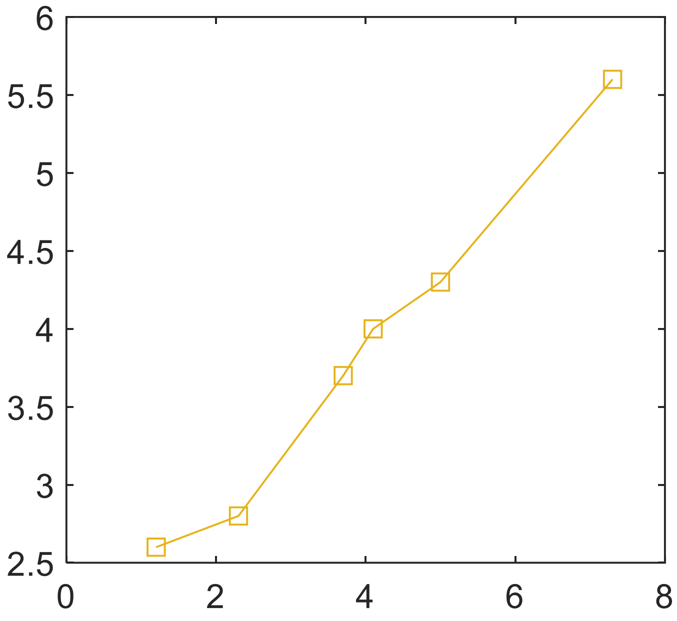


plot(x,y,'bdiamond')

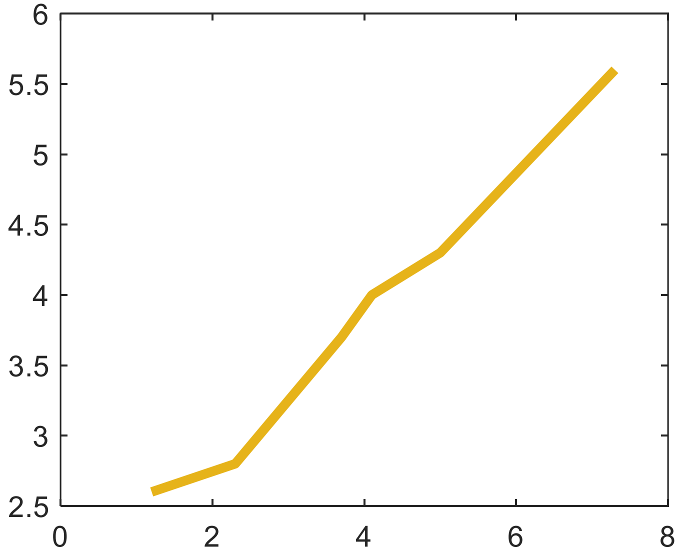


The *plot* function: more style options

- **Using customized color**: plot(x,y,'...',$\underbrace{\text{'Color',[R,G,B]}}$)
  - Include 'Color',[R,G,B]
  - [R,G,B] should have value within 0 and 1
  - Example:
    plot(x,y,'s-','Color',[0.9,0.7,0.1]);

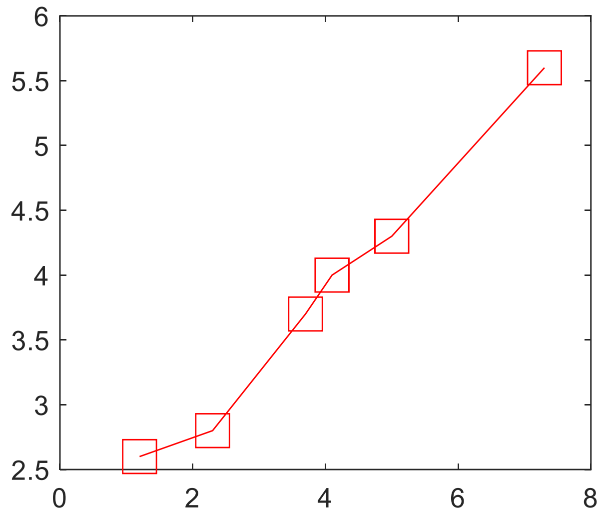


- **Adjusting line width**: plot(x,y,'...', $\underbrace{\text{'LineWidth',w}}$)
  - Include 'LineWidth',w
  - w is the desired width (default = 0.5)
  - Example:
    plot(x,y,'-','Color',[0.9,0.7,0.1],'LineWidth',3);

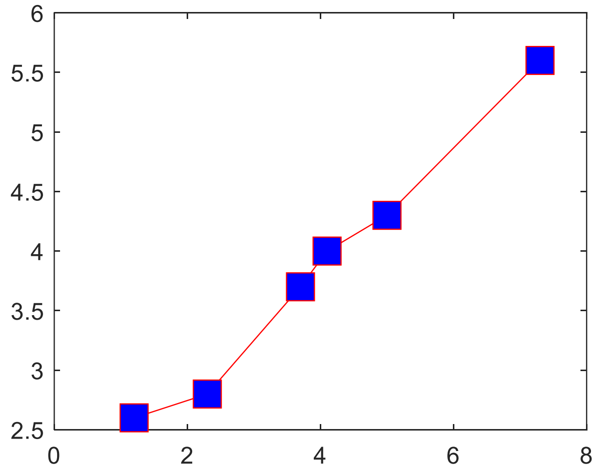


The *plot* function: more style options

- **Adjusting marker size**: plot(x,y,'...', 'MarkerSize',s)
  - Include 'MarkerSize',s
  - s is the desired marker size (default = 6)
  - Example:
    plot(x,y,'rs-','MarkerSize',15);



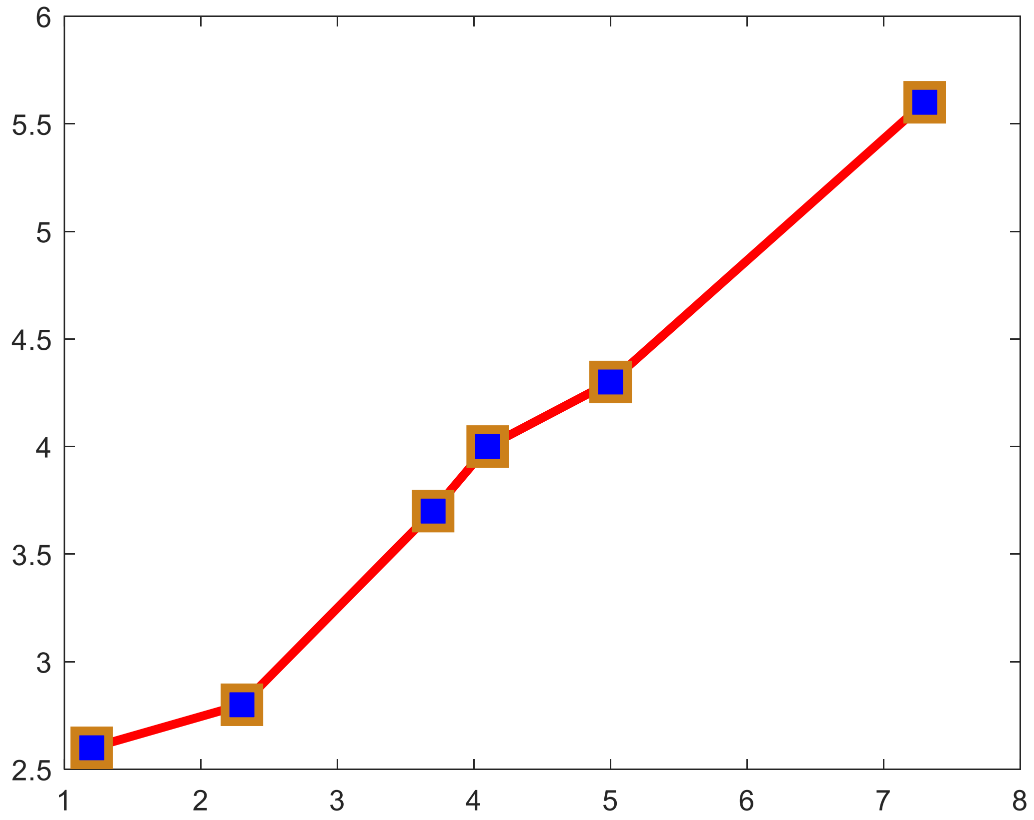
- **Adjusting marker face color**: plot(x,y,'...', 'MarkerFaceColor',c)
  - Include 'MarkerFaceColor',c
  - c is the desired color ('r', ‘b’ etc., or [R,G,B] value)
  - Example:
    plot(x,y,'rs-','MarkerSize',15,'MarkerFaceColor','b');



The *plot* function: more style options

- **Adjusting marker edge color**: plot(x,y,'...', 'MarkerEdgeColor',c)
  - Include 'MarkerEdgeColor',c
  - c is the desired color ('r', 'b' etc., or some [R,G,B] value)
  - Example:

```
plot(x,y,'rs-','MarkerSize',15, ...
    'MarkerFaceColor','b', ...
    'LineWidth', 3, ...
    'MarkerEdgeColor',[0.8,0.5,0.1]);
```



Side note:
- You can use ... to break a long line of command into several lines
- MATLAB will automatically connect them when executing the code
- Better code readability!

Plotting multiple sets of data on the same plot

- What if we want to plot multiple sets of data (points/curves) on the same plot?

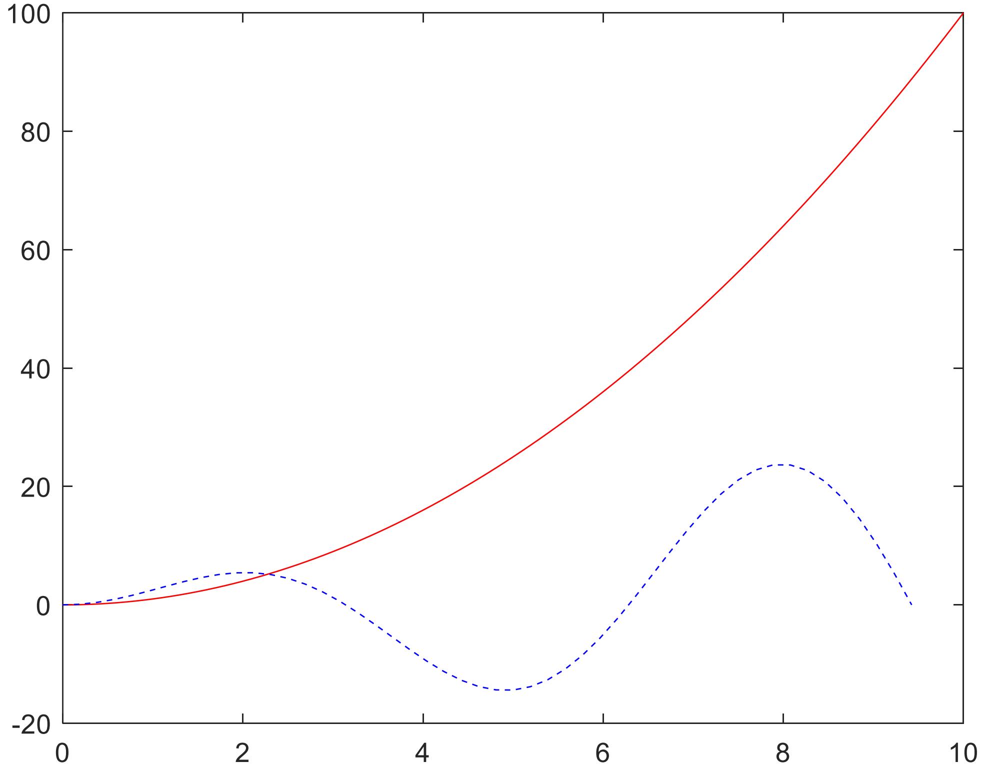
- Method 1: use plot(X1,Y1,LineSpec1,...,Xn,Yn,LineSpecn)

  - Put everything in the same plot command one by one

Example:

```
x = 0:0.1:10;
y = x.^2;
t = linspace(0,3*pi,50);
z = 3*t.*sin(t);

figure;
plot(x,y,'r-',t,z,'b--');
```

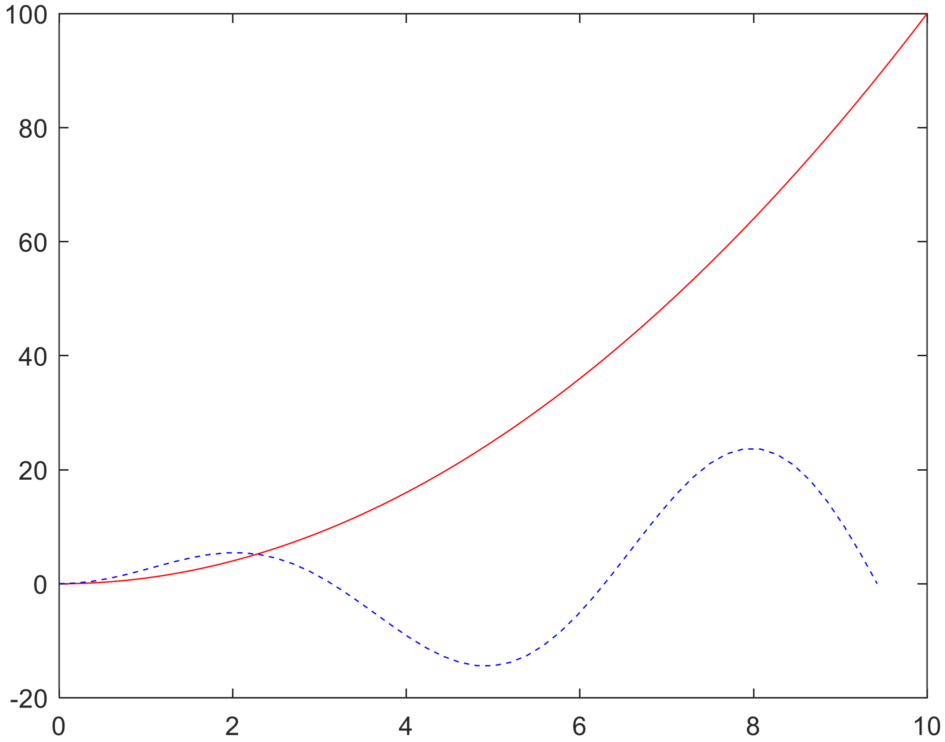


Plotting multiple sets of data on the same plot

- Method 2: use hold
  - To plot two or more graphs in one window, use hold on
  - To unhold the windows, use hold off

Example:

```
figure;
plot(x,y,'r-');
hold on;
plot(t,z,'b--');
```



- Remark:
  - More flexible than Method 1 (don’t need to plot everything at once)
  - Caution:
    If we don’t put hold on here, the second plot will replace the first plot

The *subplot* function: One figure window, multiple subplots

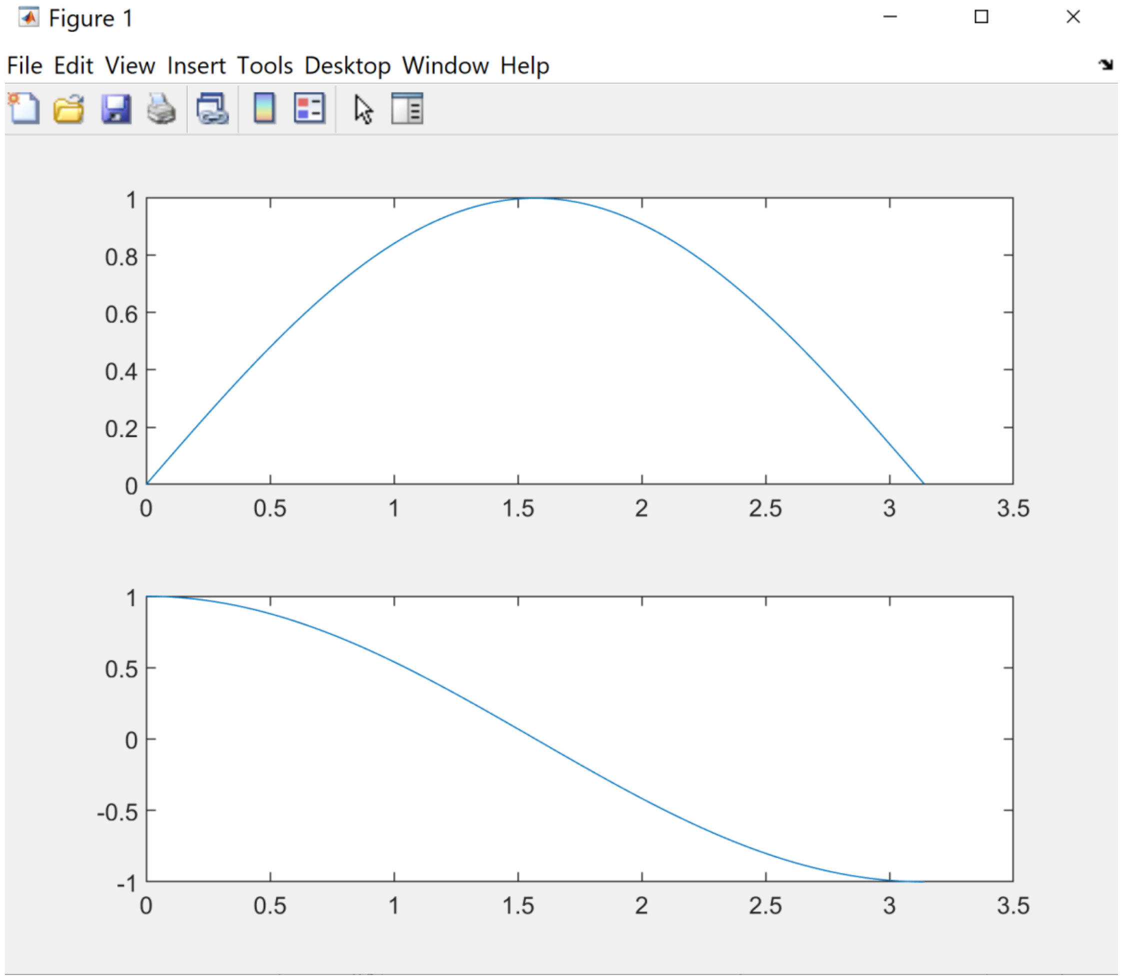
- subplot(m,n,p):
  - Divide the current figure into an m-by-n grid
  - p is the ID of the grid in which you want to put the plot

Example:

```
x = 0:0.01:pi;
figure;

subplot(2,1,1);
plot(x, sin(x));

subplot(2,1,2);
plot(x, cos(x));
```

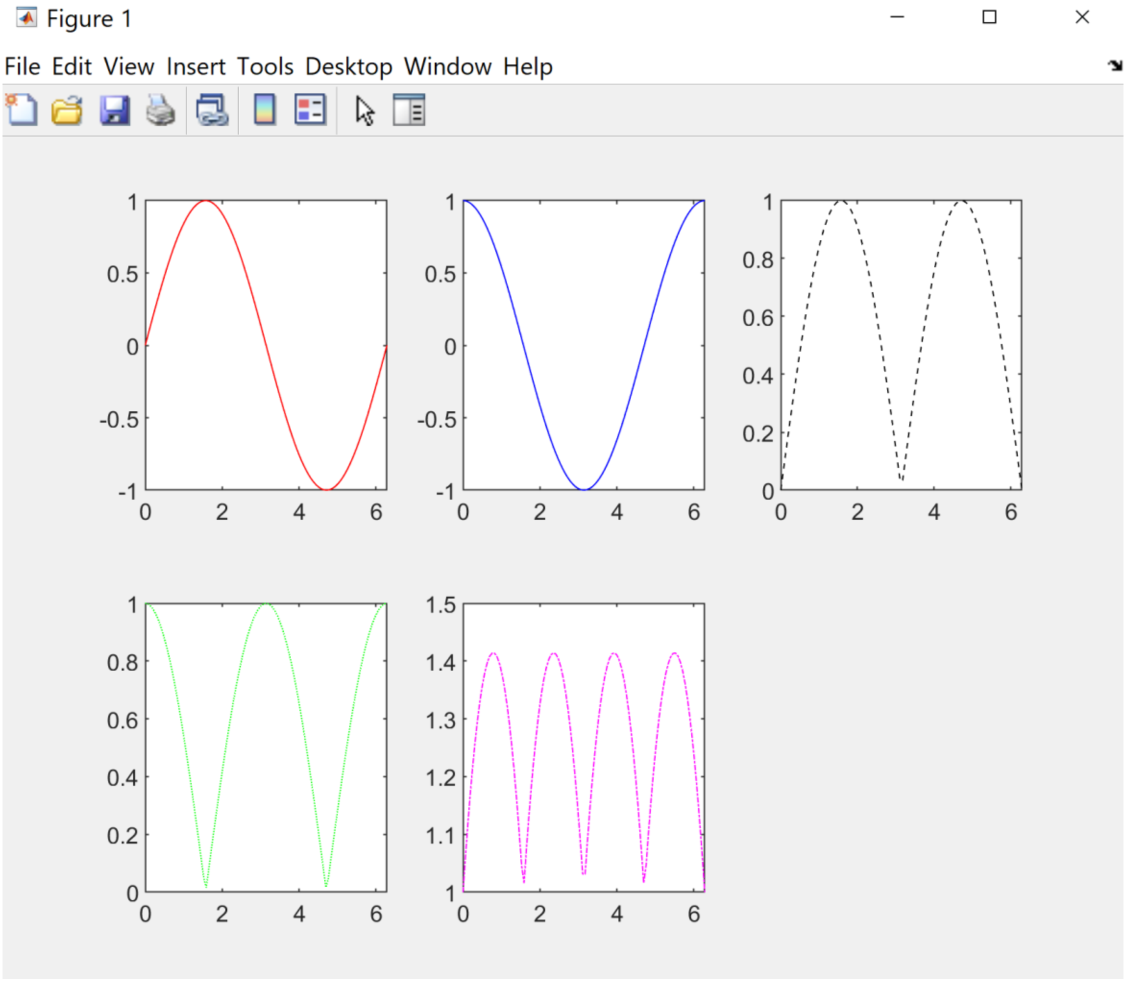


The *subplot* function: One figure window, multiple subplots

- subplot(m,n,p):
  - Divide the current figure into an m-by-n grid
  - p is the ID of the grid in which you want to put the plot

Example:

```
figure;
x = linspace(0,2*pi,100);
subplot(2,3,1);
plot(x, sin(x), 'r-');
subplot(2,3,2);
plot(x, cos(x), 'b-');
subplot(2,3,3);
plot(x, abs(sin(x)), 'k--');
subplot(2,3,4);
plot(x, abs(cos(x)), 'g:');
subplot(2,3,5);
plot(x, abs(sin(x))+abs(cos(x)), 'm-.');
```

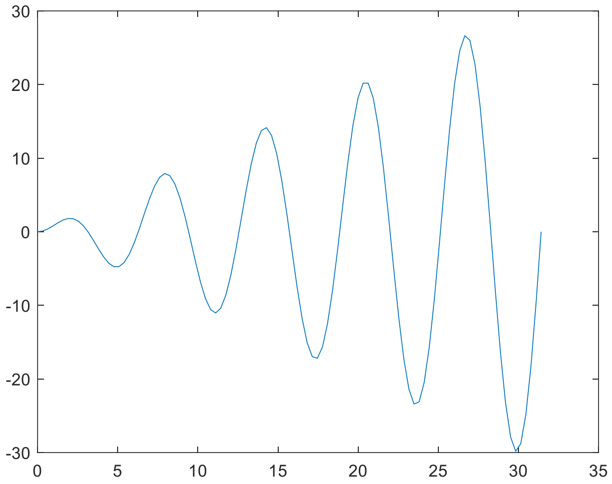


Adjusting the axes of your plot

- Some useful commands:
  - axis on: show the axes (default)
  - axis off: hide the axes
  - axis equal: make the two axes equal in ratio
  - axis tight: set axis limit as the range of the data

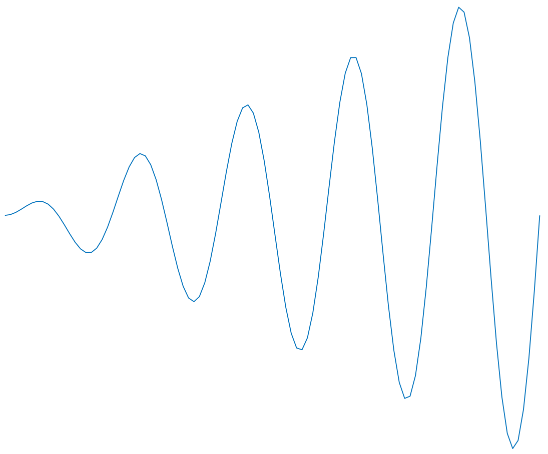
Example:

```
t = linspace(0,10*pi,100);
figure; plot(t,t.*sin(t));
axis on;
```



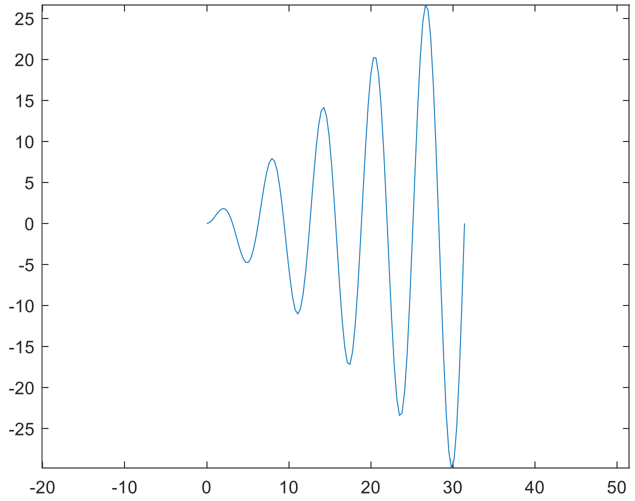
Example:

```
t = linspace(0,10*pi,100);
figure; plot(t,t.*sin(t));
axis off;
```



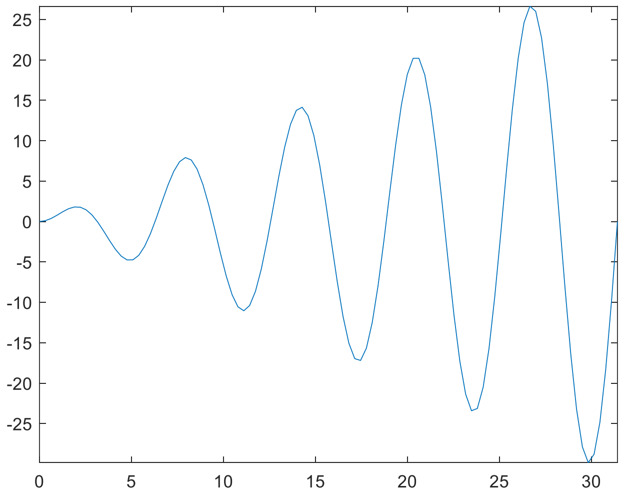
Example:

```
t = linspace(0,10*pi,100);
figure; plot(t,t.*sin(t));
axis equal;
```



Example:

```
t = linspace(0,10*pi,100);
figure; plot(t,t.*sin(t));
axis tight;
```

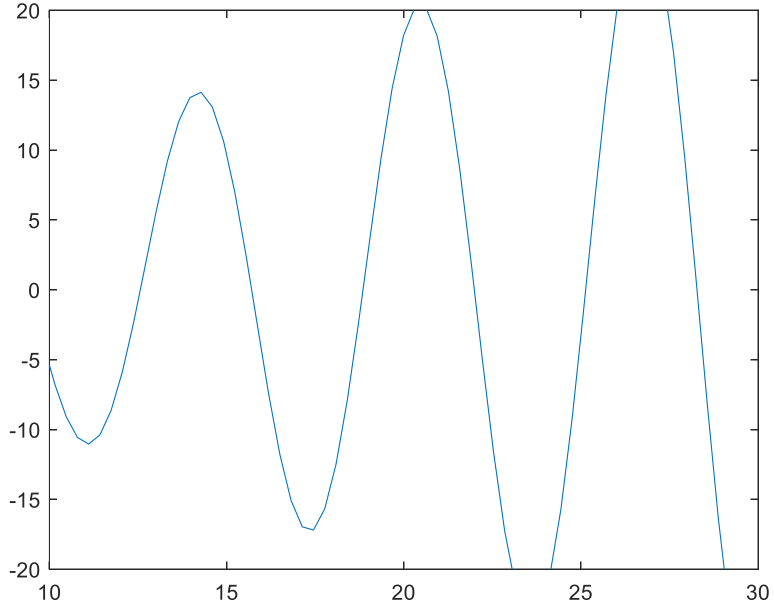


Adjusting the axes of your plot

- Some useful commands:
  - axis([xmin,xmax,ymin,ymax]): set the axis limits
  - xlim([xmin,xmax]): set the x-axis limit
  - ylim([ymin,ymax]): set the y-axis limit

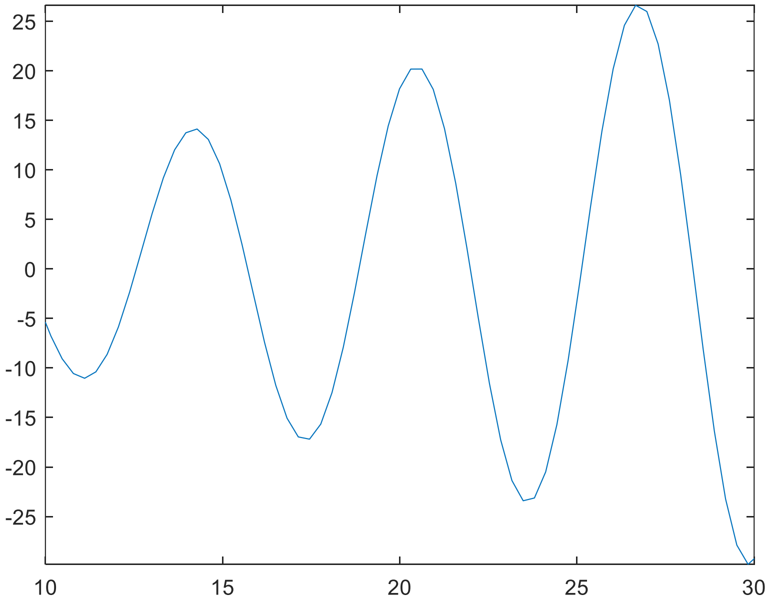
Example:

```
t = linspace(0,10*pi,100);
figure; plot(t,t.*sin(t));
axis([10, 30, -20, 20])
```



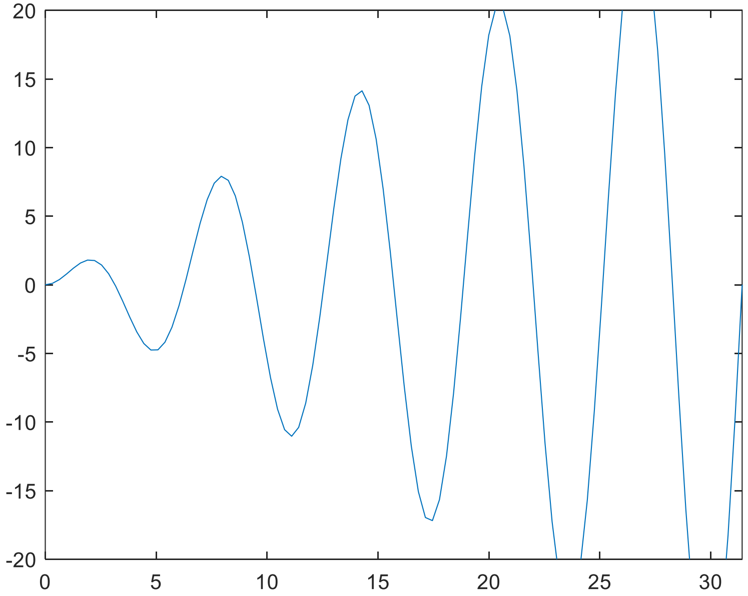
Example:

```
t = linspace(0,10*pi,100);
figure; plot(t,t.*sin(t));
xlim([10, 30]);
```



Example:

```
t = linspace(0,10*pi,100);
figure; plot(t,t.*sin(t));
ylim([-20, 20]);
```



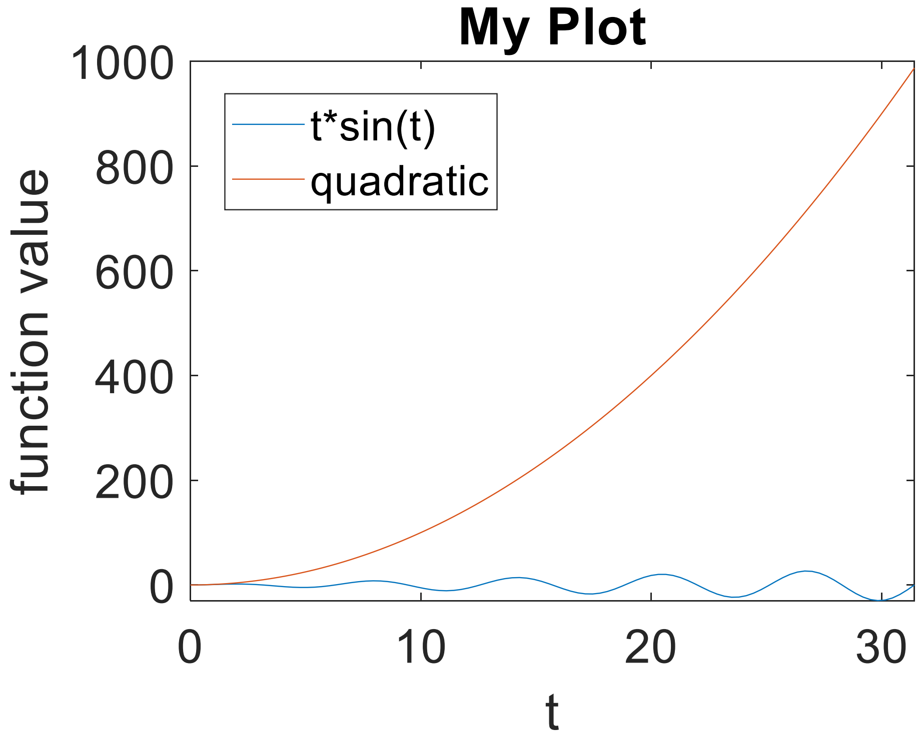
Adding title, axis labels, and figure legends

- Some useful commands:
  - title('string'): add figure title
  - xlabel('string'): label the x-axis
  - ylabel('string'): label the y-axis
  - legend('string'): label each set of data on the plot
  - set(gca,'FontSize',k): adjust the font size of the title and labels (default k = 10)

Note: gca stands for get current Axes, and set is for setting the Axes object properties. See: https://www.mathworks.com/help/matlab/ref/matlab.graphics.axis.axes-properties.html

Example:

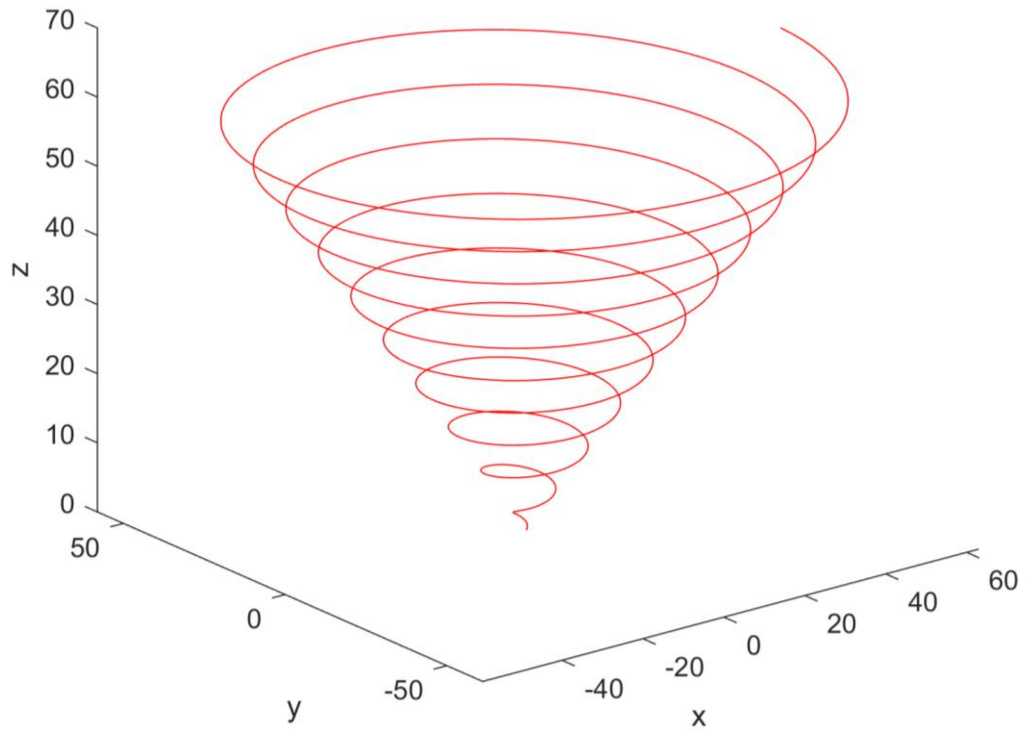
```
t = linspace(0,10*pi,100);
figure;
plot(t,t.*sin(t));              % the first curve
hold on;
plot(t, t.^2);                  % the second curve
title('My Plot')                % title
xlabel('t')                     % x-label
ylabel('function value')        % y-label
legend('t*sin(t)','quadratic')  % label both curves
set(gca,'FontSize',20)          % Increase the font size
```



3D visualization using MATLAB

- MATLAB can also do 3D visualization!

- Plotting 3D data points / curves: plot3(x,y,z,…)
  Example:
```
t = linspace(0,20*pi,1000);
figure;
plot3(t.*cos(t),t.*sin(t),t, 'r-');
xlabel('x')
ylabel('y')
zlabel('z')
```



- You can drag and rotate the 3D viewer

- Plotting surfaces:
  Need to create a meshgrid (next time)

Reminder: Lab 4 this week

January

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	[28]	[29]	[30]	[31]	

February

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						[1]
[2]	[3]	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	1

March

Sun	Mon	Tue	Wed	Thu	Fri	Sat
2	[3]	[4]	[5]	[6]	[7]	[8]
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

April

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17		

Lecture 1- Lecture 13

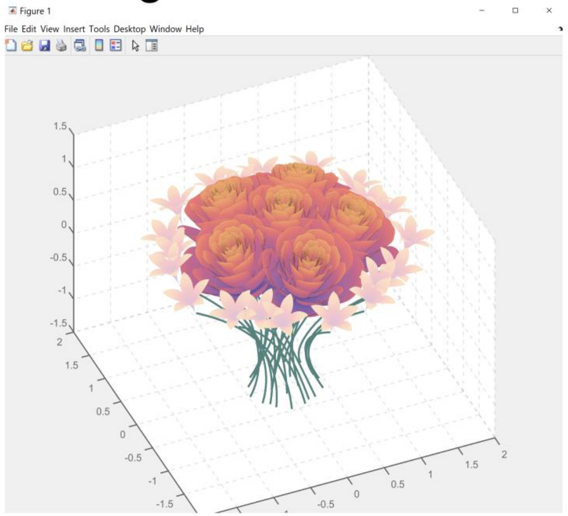
Lab 1 - Lab 10 (40%)

Test 1 (30%)
Test 2 (30%)

Tips for Valentine’s Day

Buying roses

Making roses in MATLAB



See: https://www.mathworks.com/matlabcentral/fileexchange/154496-rose-bouquet

Thank you!

Next time:

- More on 2D and 3D visualization