# 2D Plotting and Curve Fitting

**黃世強 (Sai-Keung Wong)**

*National Chiao Tung University, Taiwan*

# Curve Plotting and Fitting

- Visualizing plots of large sets of data can help analyze the trends of data.

- Assist to interpret the data.

- Curve fitting of data is useful to find an approximated mathematical model.

- Useful in interpolation of data and also extrapolation of data.

# subplot

Create axes in tiled positions.

H = subplot(m,n,p), or subplot(mnp),

breaks the Figure window into an m-by-n matrix of small axes, selects the p-th axes for the current plot, and returns the axes handle.

The axes are counted along the top row of the Figure window, then the second row, etc.

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |

# subplot

Create axes in tiled positions.

H = subplot(m,n,p), or subplot(mnp),

breaks the Figure window into an m-by-n matrix of small axes, selects the p-th axes for the current plot, and returns the axes handle.

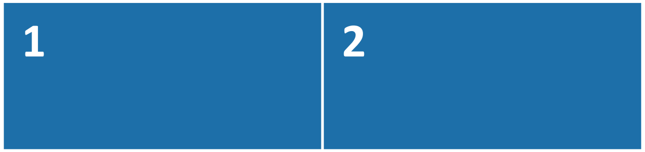
The axes are counted along the top row of the Figure window, then the second row, etc.

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |

# Examples

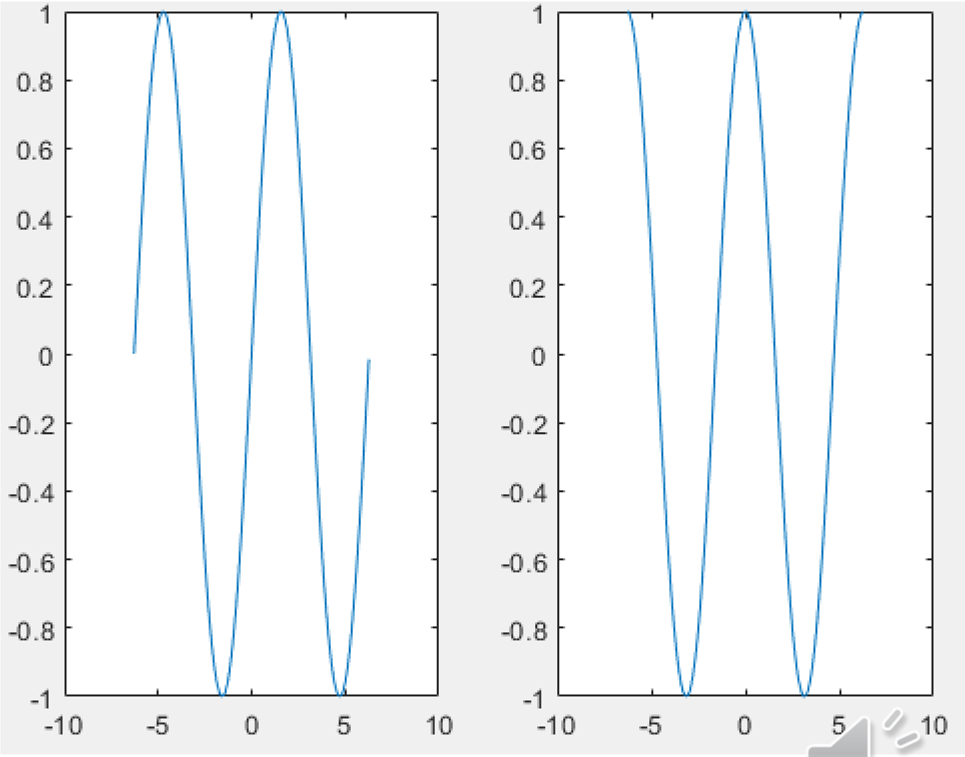
- To plot sin(x) and cos(x) on the same figure, side-by-side.

```
x = -2*pi:0.05:2*pi;
subplot(1,2,1);
plot(x,sin(x));
subplot(1,2,2);
plot(x,cos(x));
```



1

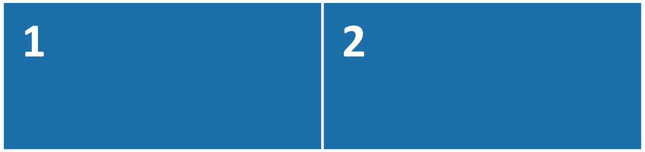
2



# Examples

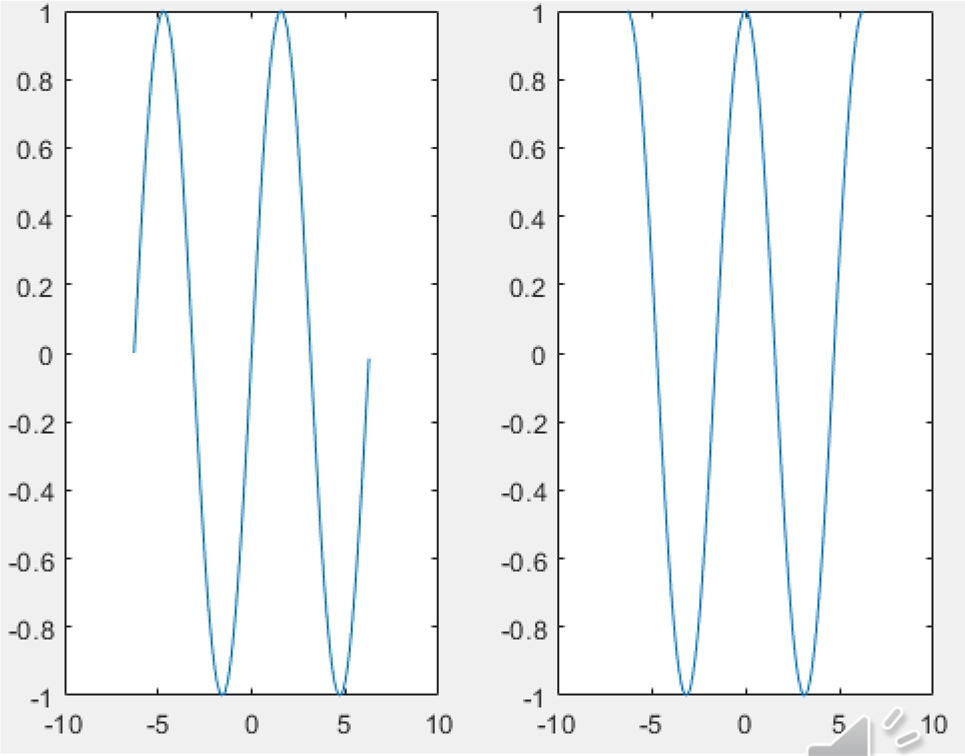
- To plot sin(x) and cos(x) on the same figure, side-by-side.

```
x = -2*pi:0.05:2*pi;
subplot(1,2,1);
plot(x,sin(x));
subplot(1,2,2);
plot(x,cos(x));
```



1

2



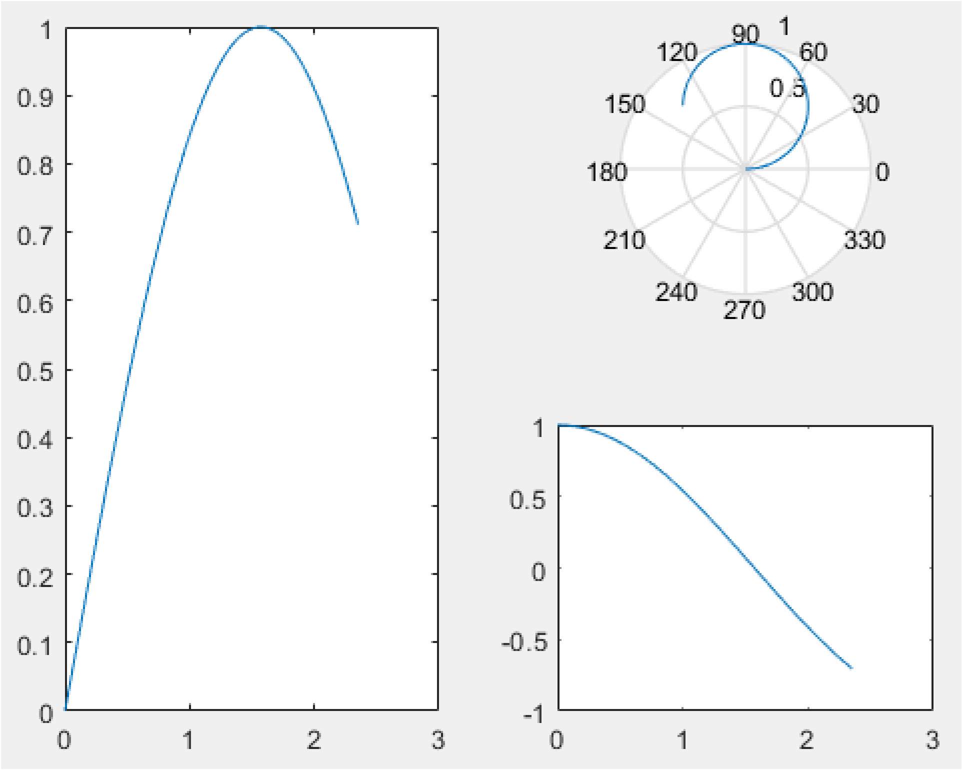
# Other functions

2-D plotting utilities

- Polar plots
- Logarithmic plots
- Bar charts (or graphs)
- Pie charts

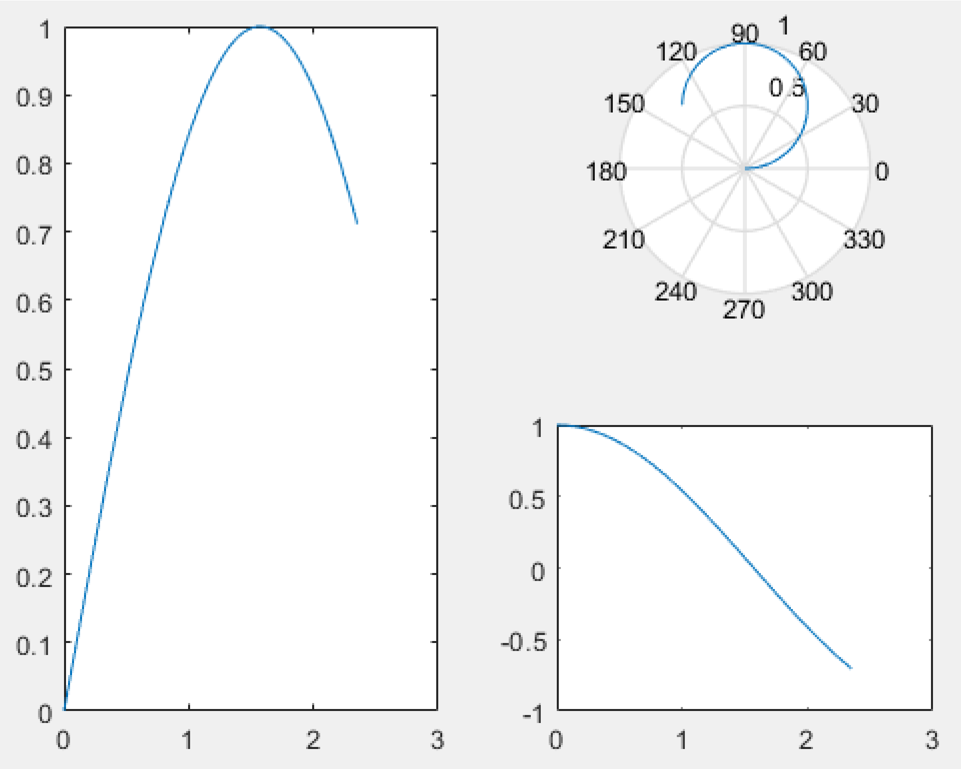
# Polar plots
# Plotting data in polar coordinates

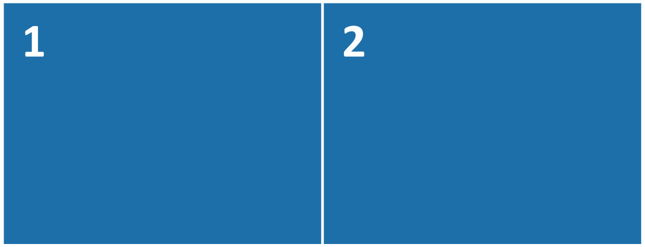
```
x = 0:0.01:pi*0.75;
r = sin(x);
subplot(1,2,1);
plot(x,r);
subplot(2,2,2);
polar(x,r);
subplot(2,2,4);
plot(x,cos(x));
```

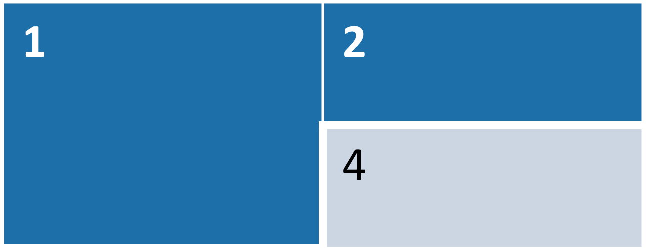


# Polar plots
# Plotting data in polar coordinates

```
x = 0:0.01:pi*0.75;
r = sin(x);
subplot(1,2,1);
plot(x,r);
subplot(2,2,2);
polar(x,r);
subplot(2,2,4);
plot(x,cos(x));
```

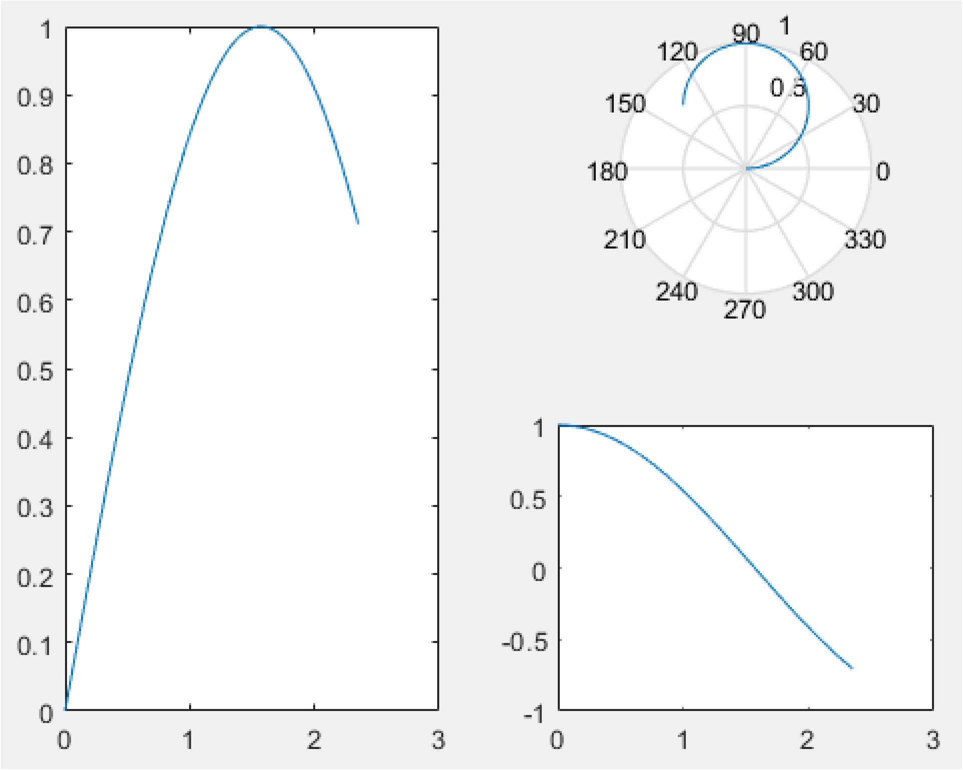


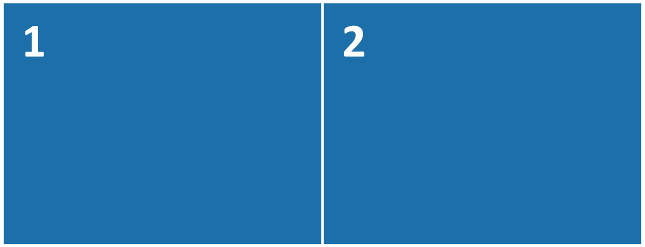


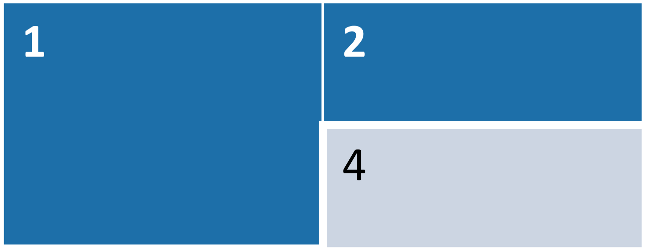


# Polar plots
# Plotting data in polar coordinates

```
x = 0:0.01:pi*0.75;
r = sin(x);
subplot(1,2,1);
plot(x,r);
subplot(2,2,2);
polar(x,r);
subplot(2,2,4);
plot(x,cos(x));
```

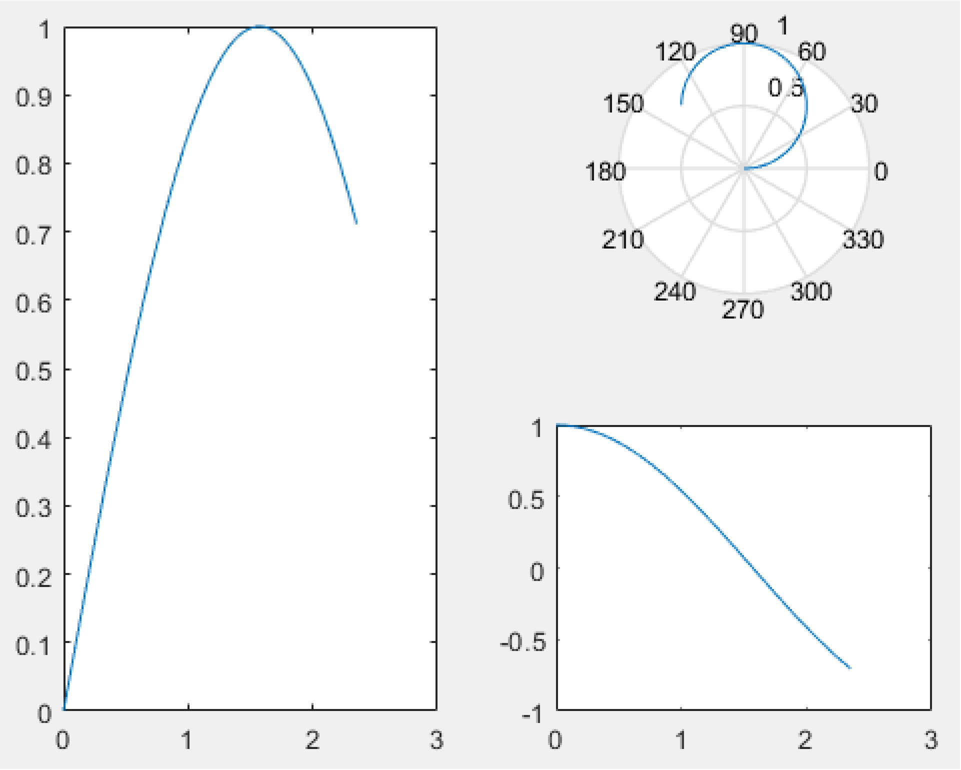


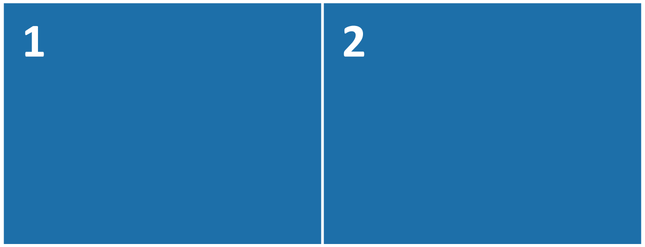


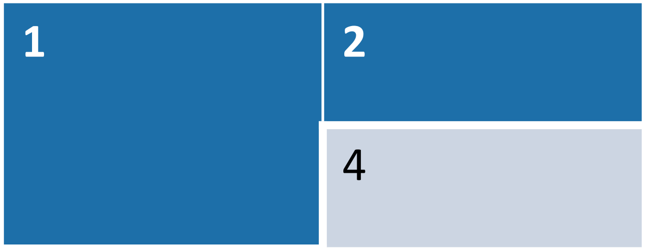


# Polar plots
# Plotting data in polar coordinates

```
x = 0:0.01:pi*0.75;
r = sin(x);
subplot(1,2,1);
plot(x,r);
subplot(2,2,2);
polar(x,r);
subplot(2,2,4);
plot(x,cos(x));
```

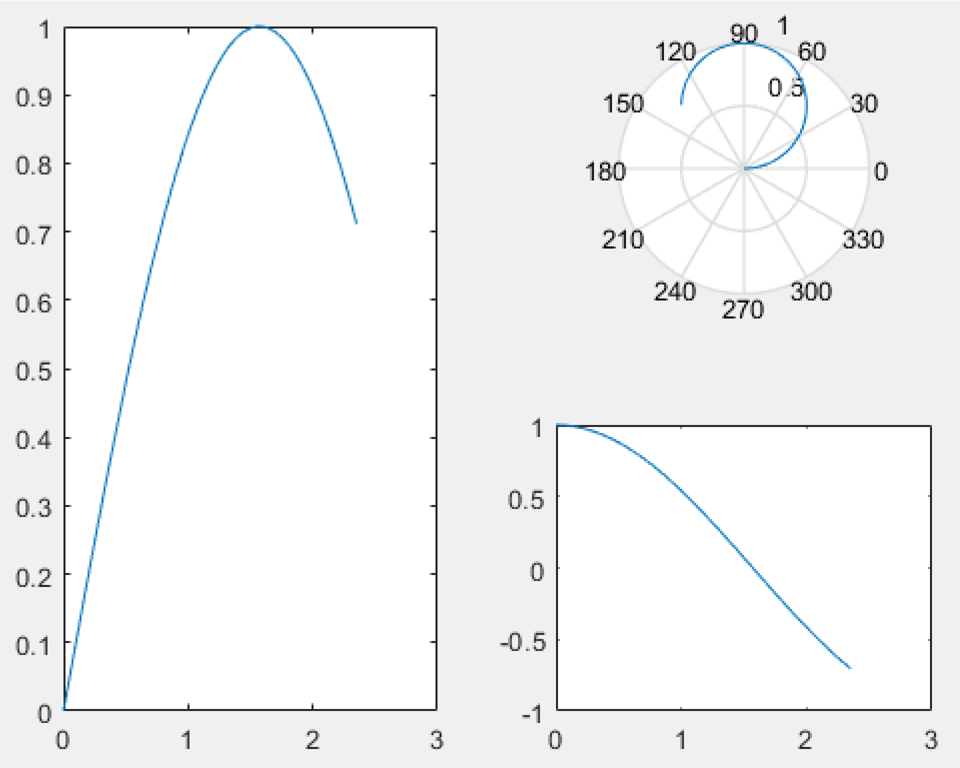


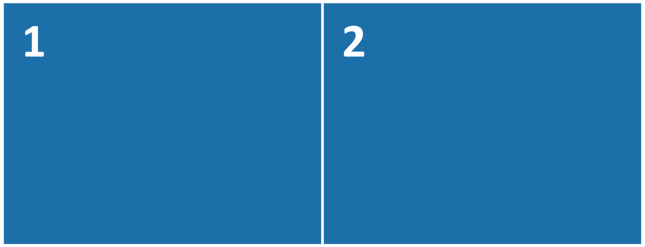


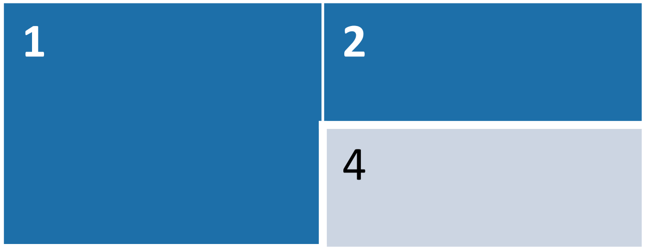


# Polar plots
# Plotting data in polar coordinates

```
x = 0:0.01:pi*0.75;
r = sin(x);
subplot(1,2,1);
plot(x,r);
subplot(2,2,2);
polar(x,r);
subplot(2,2,4);
plot(x,cos(x));
```

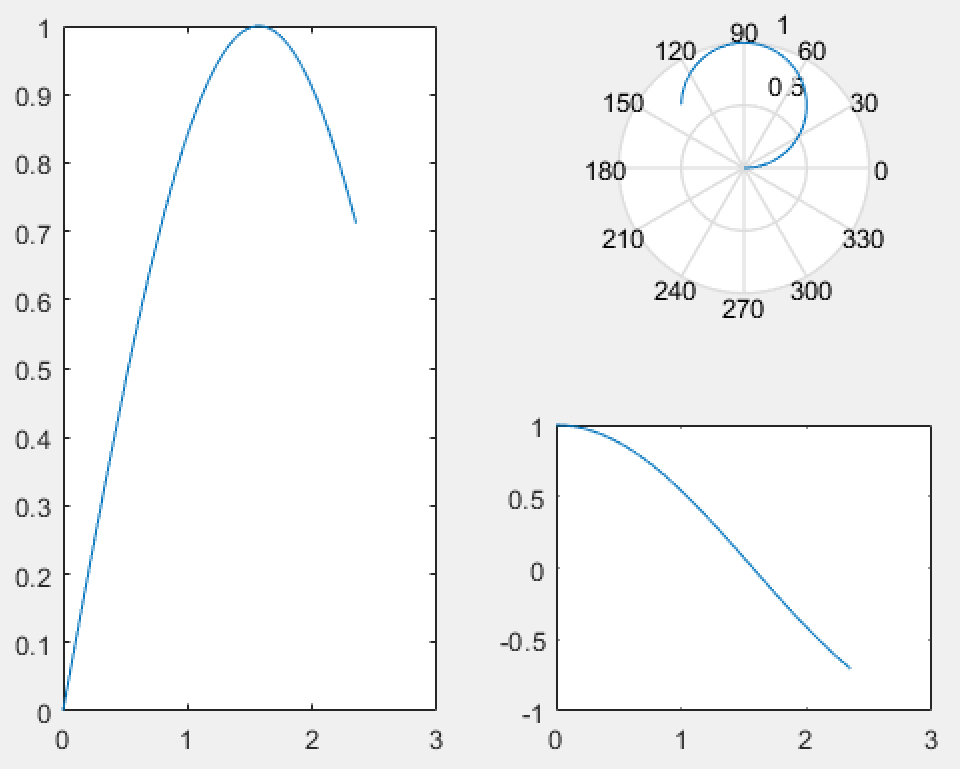


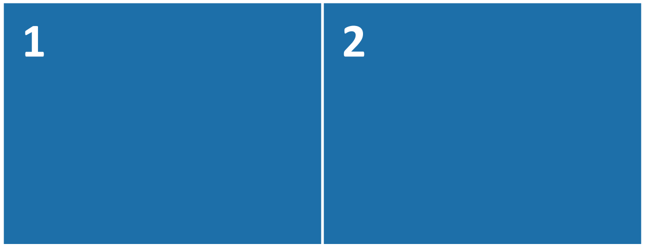


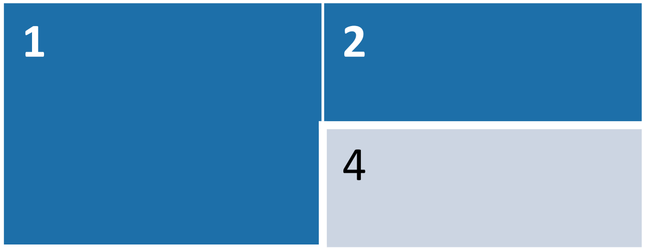


# Polar plots
# Plotting data in polar coordinates

```
x = 0:0.01:pi*0.75;
r = sin(x);
subplot(1,2,1);
plot(x,r);
subplot(2,2,2);
polar(x,r);
subplot(2,2,4);
plot(x,cos(x));
```

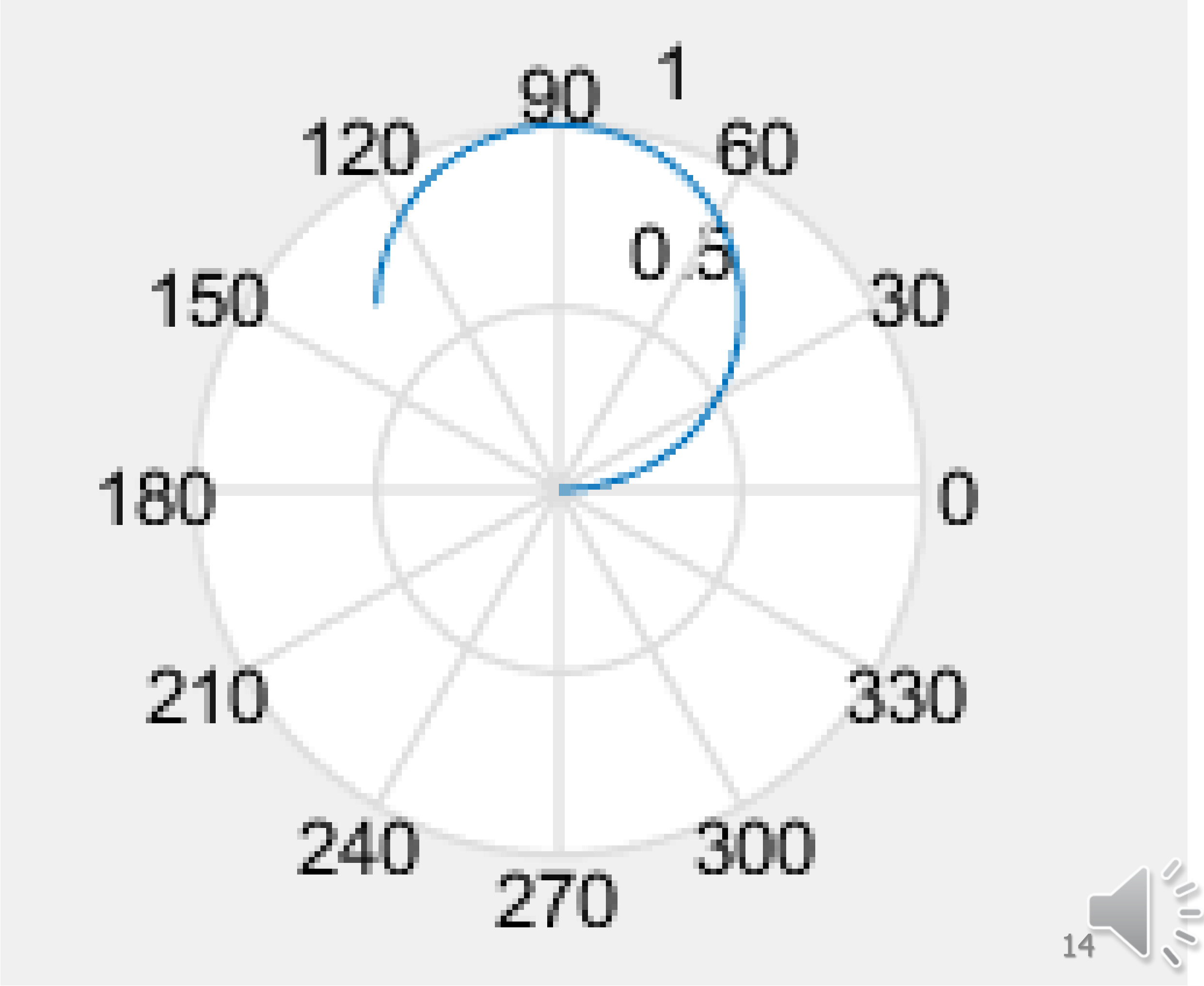






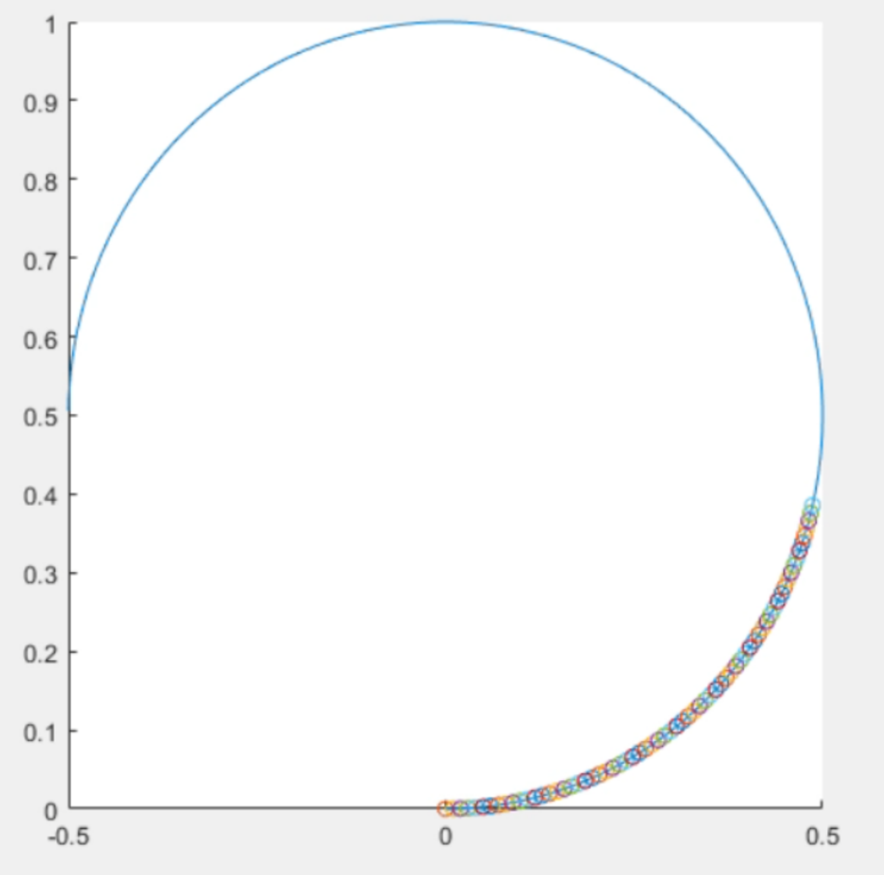
# Polar plots

```
x = 0:0.01:pi*0.75;
r = sin(x);
polar(x,r);
```



# Demo: Polar plots
# Tracing a long a circular arc

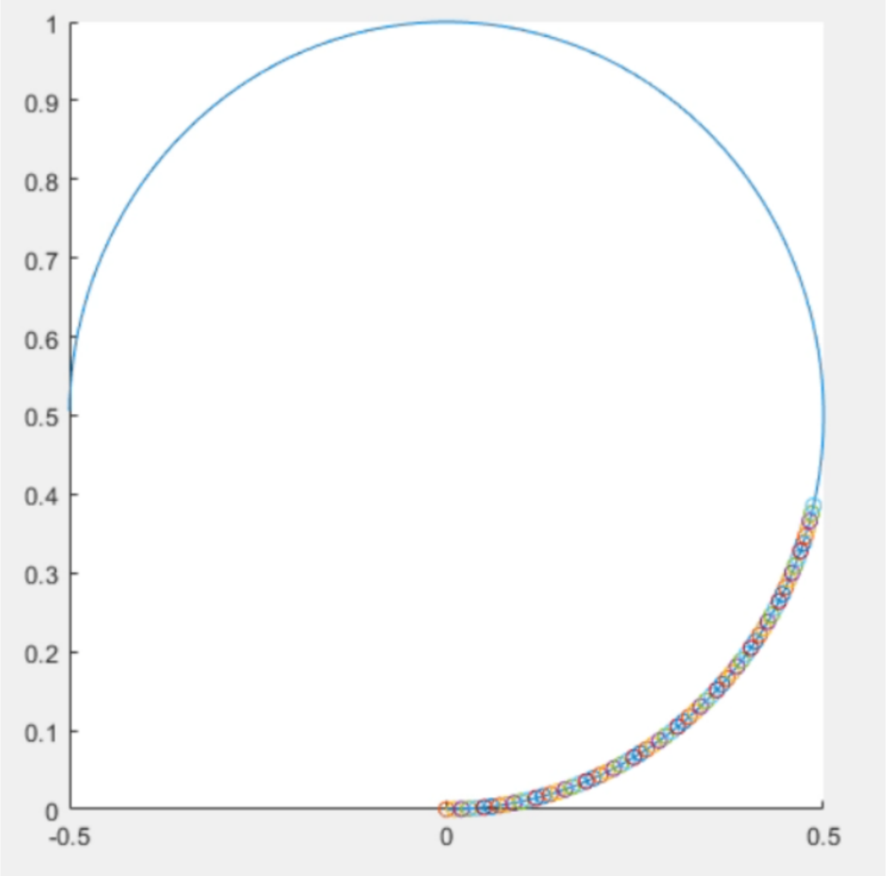
```
clear;close all;
hold on;
x = 0:0.01:pi*0.75;
r = sin(x);
polar(x,r);
for x0 = x
    r0 = sin(x0);
  polar(x0,r0,'o');
  pause(0.033);
end
hold off;
```



Euclidean space

# Demo: Polar plots
# Tracing a long a circular arc

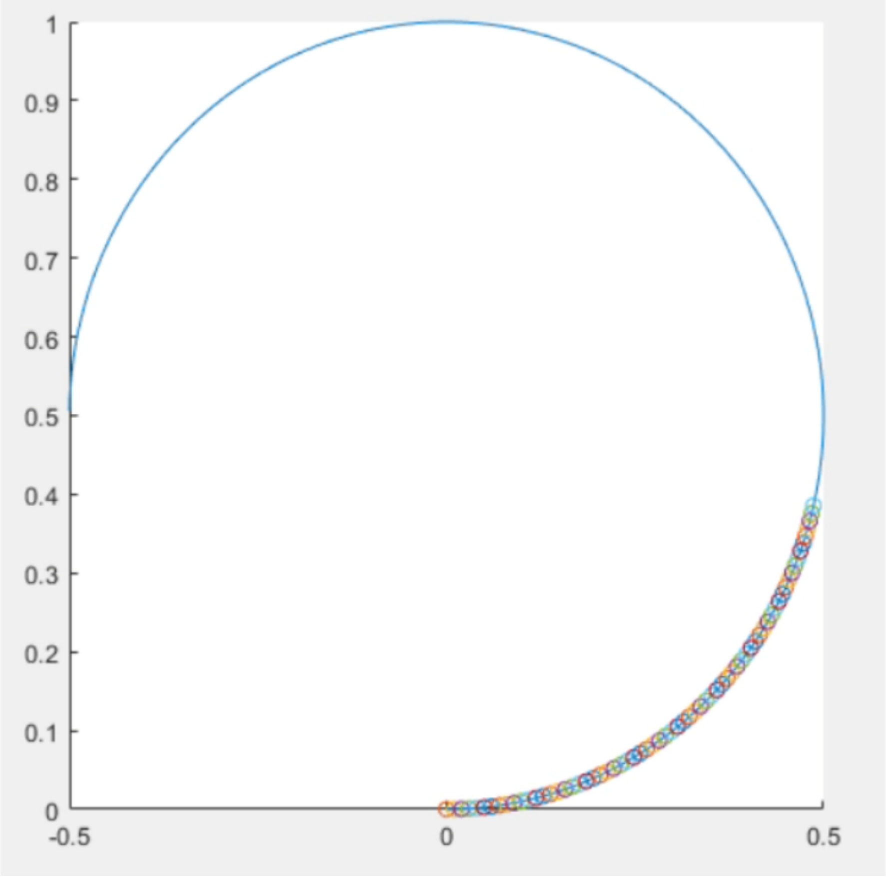
```
clear;close all;
hold on;
x = 0:0.01:pi*0.75;
r = sin(x);
polar(x,r);
for x0 = x
    r0 = sin(x0);
  polar(x0,r0,'o');
  pause(0.033);
end
hold off;
```



Euclidean space

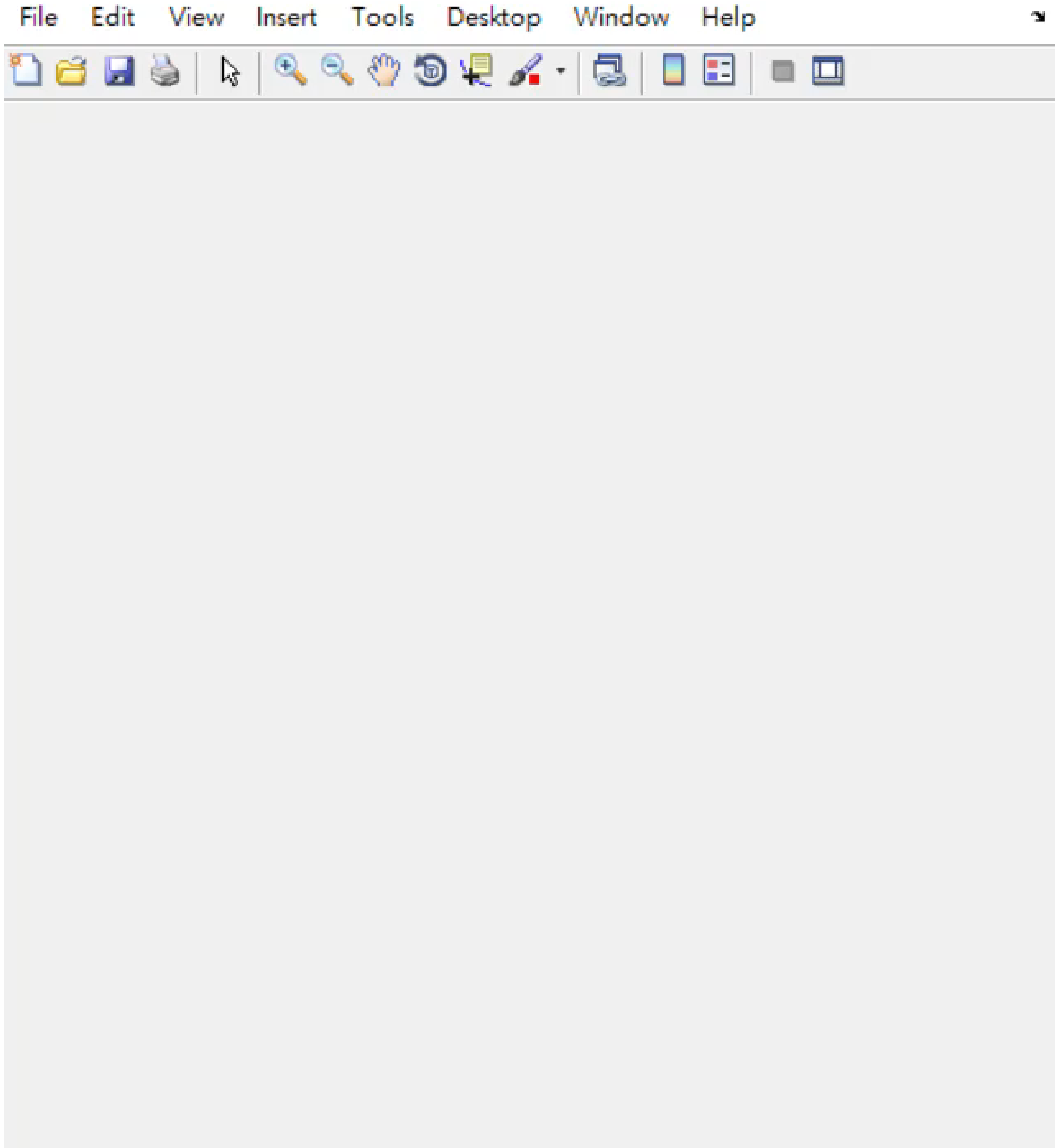
# Demo: Polar plots
# Tracing a long a circular arc

```
clear;close all;
hold on;
x = 0:0.01:pi*0.75;
r = sin(x);
polar(x,r);
for x0 = x
    r0 = sin(x0);
  polar(x0,r0,'o');
  pause(0.033);
end
hold off;
```



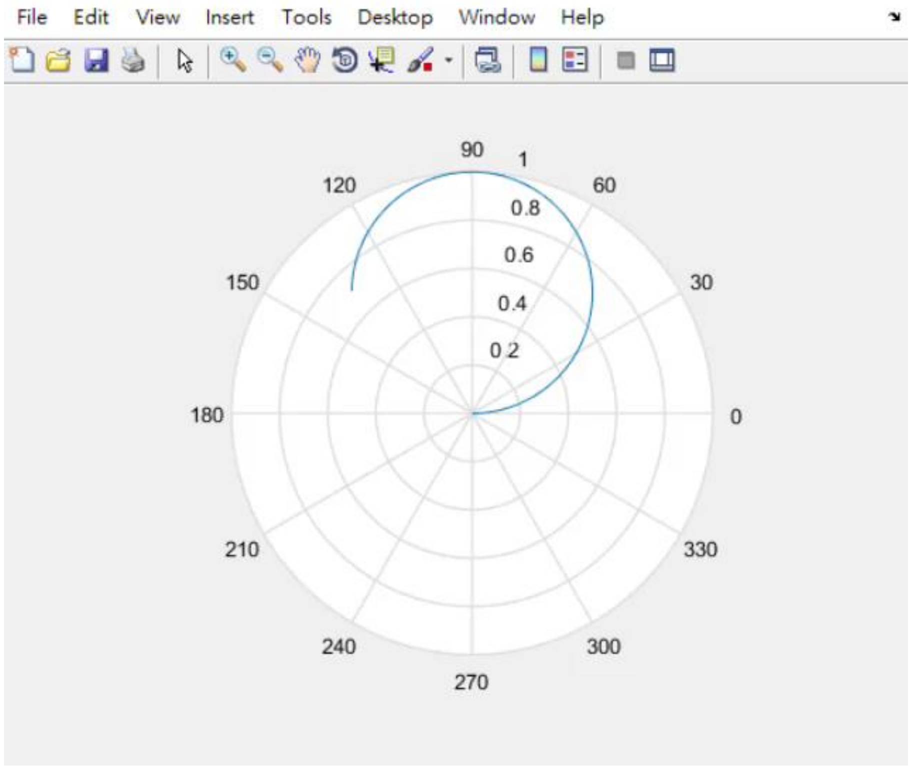
Euclidean space

# Demo: Polar plots



# Demo
# Polar plots

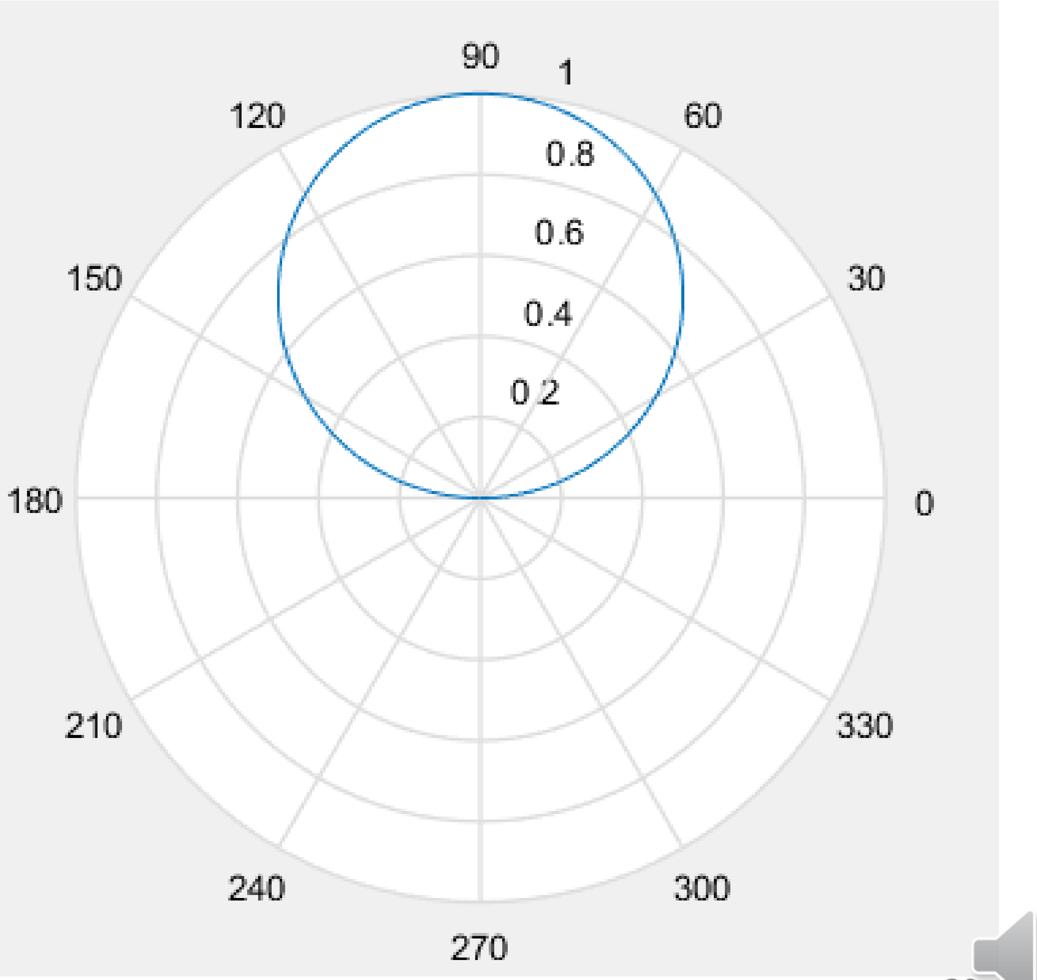
```
clear;close all;
x = 0:0.01:pi*0.75;
r = sin(x);
polar(x,r); %show it
input('Press ENTER to
start.');
hold on; %for x0 = x
  r0 = sin(x0);
  polar(x0,r0,'o');
  pause(0.033);
end
hold off;
then hold on
```



# Polar plots

```
x = 0:0.01:pi;
r = sin(x);
polar(x,r);
```

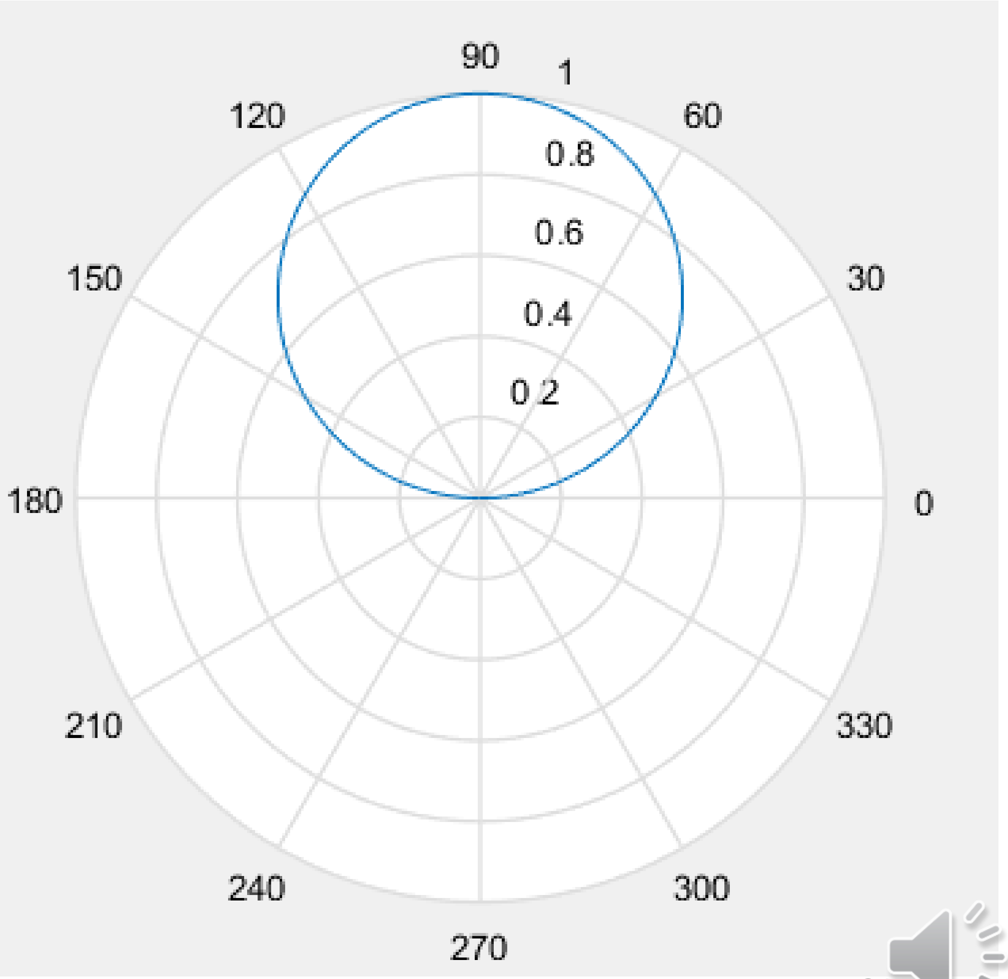
Draw a circle.



# Polar plots

```
x = 0:0.01:2*pi;
r = sin(x);
polar(x,r);
```

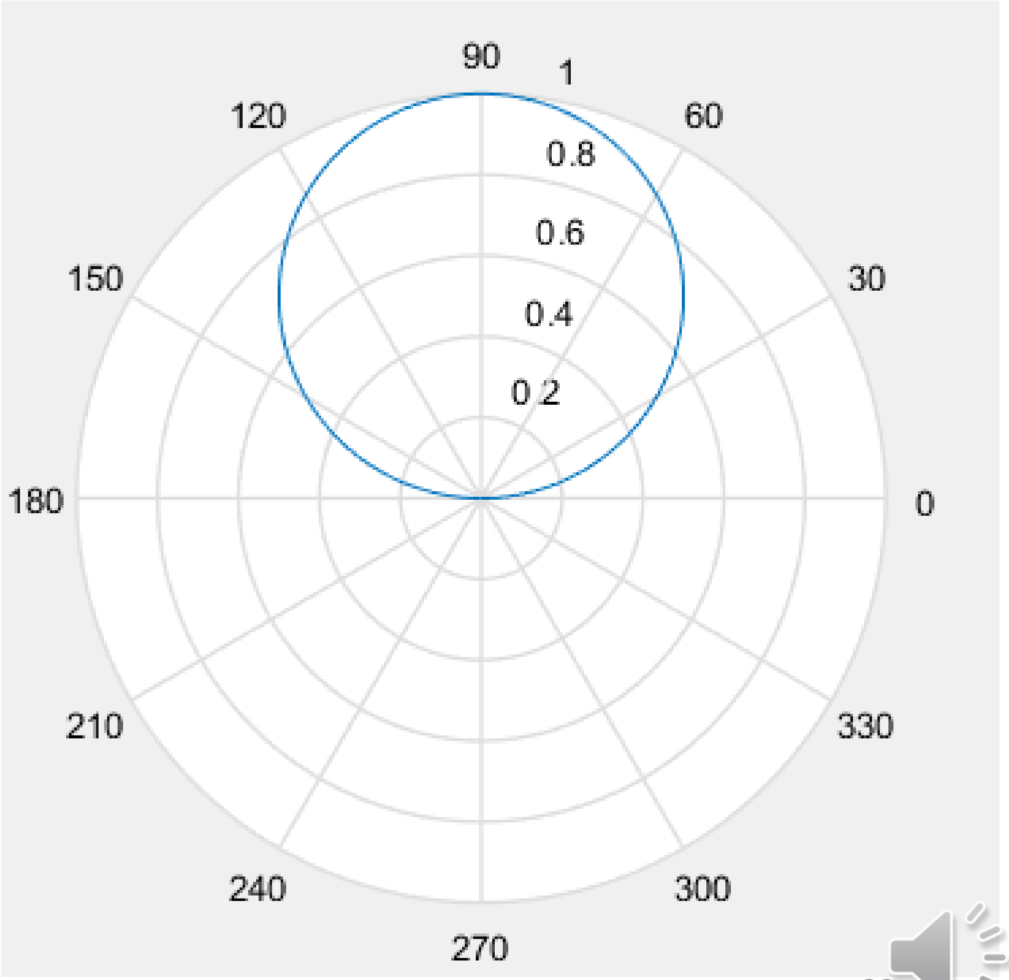
```
Only one circle?
Why not two circles?
```



# Polar plots

```
x = 0:0.01:2*pi;
r = sin(x);
polar(x,r);
```

Only one circle?
Why not two circles?

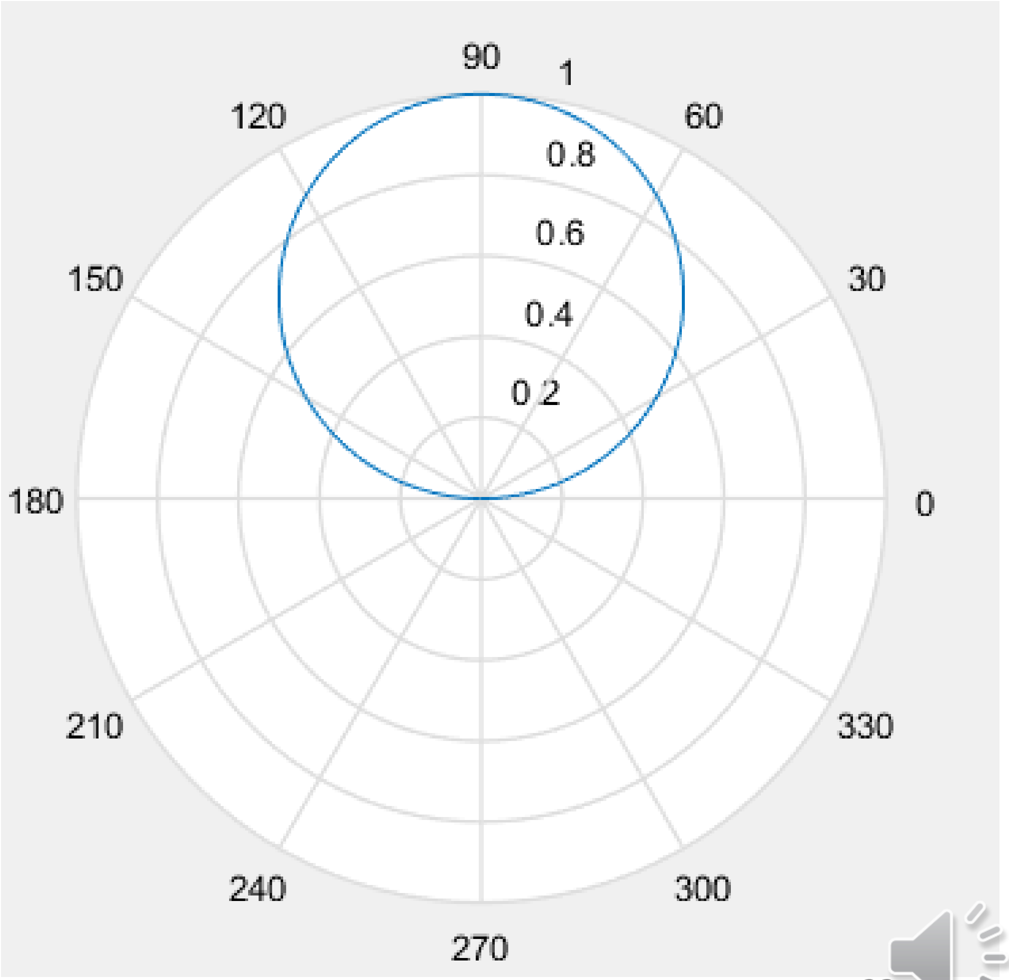


# Polar plots

```
x = 0:0.01:2*pi;
r = sin(x);
polar(x,r);
```

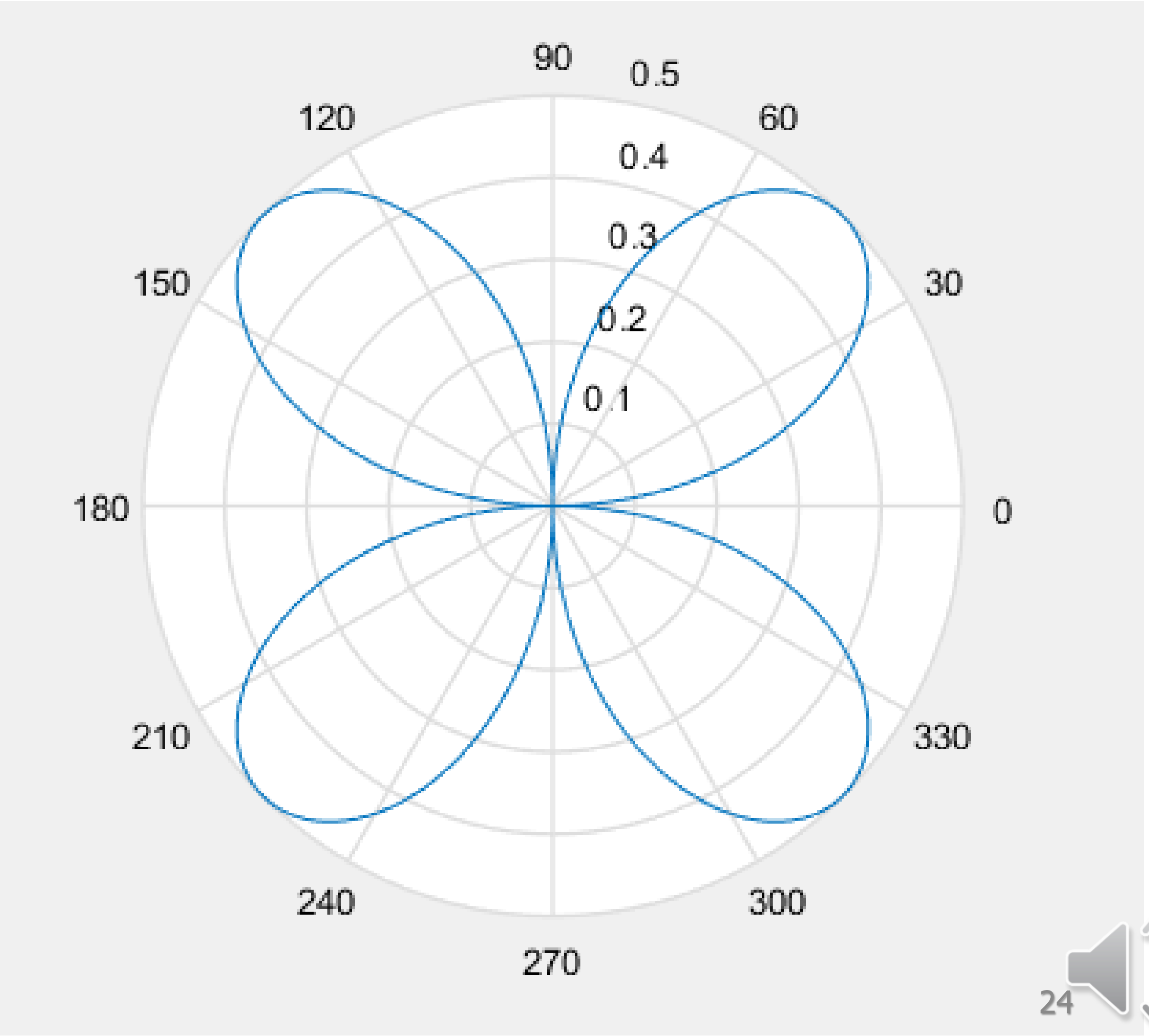
Only one circle? Why not two circles?

We produce two circles but they are the same.



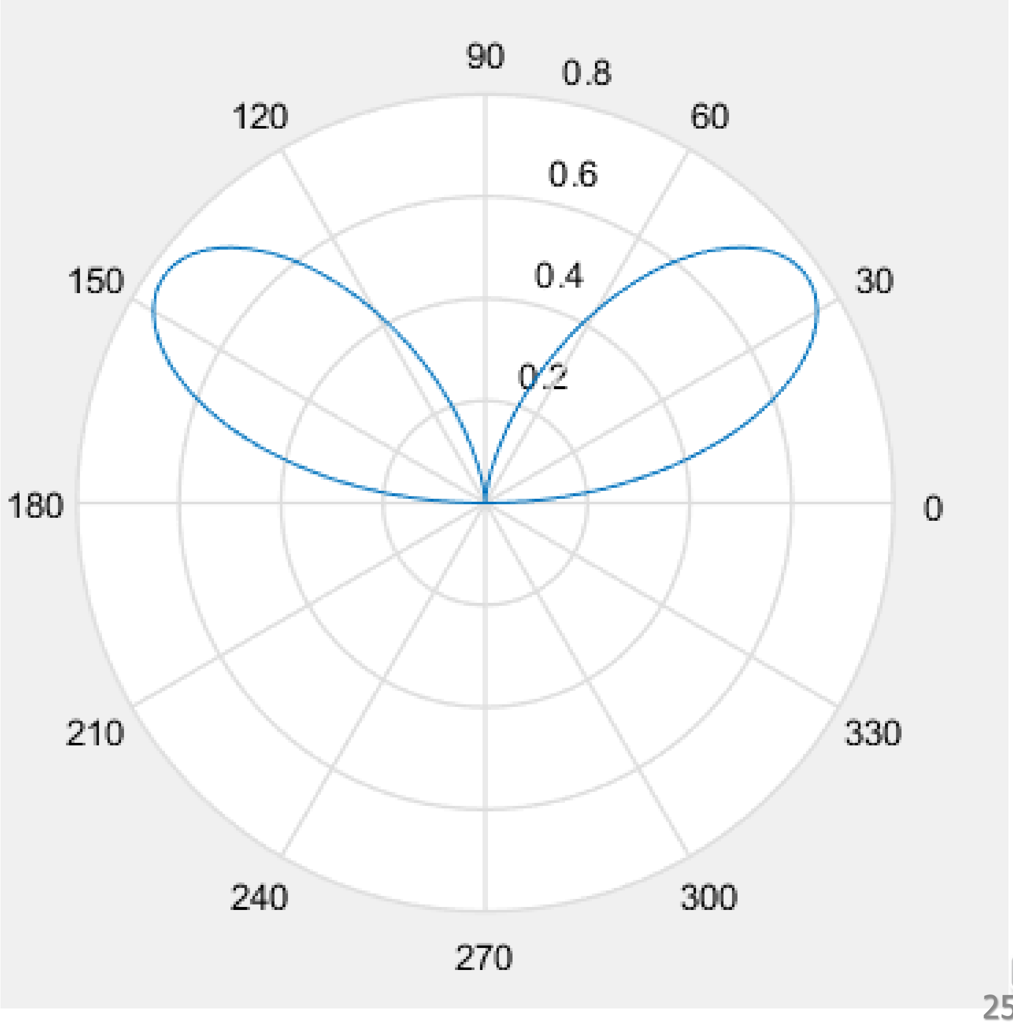
# Polar plots

```
x = 0:0.01:pi*2;
r = cos(x).*sin(x);
figure
polar(x,r);
```



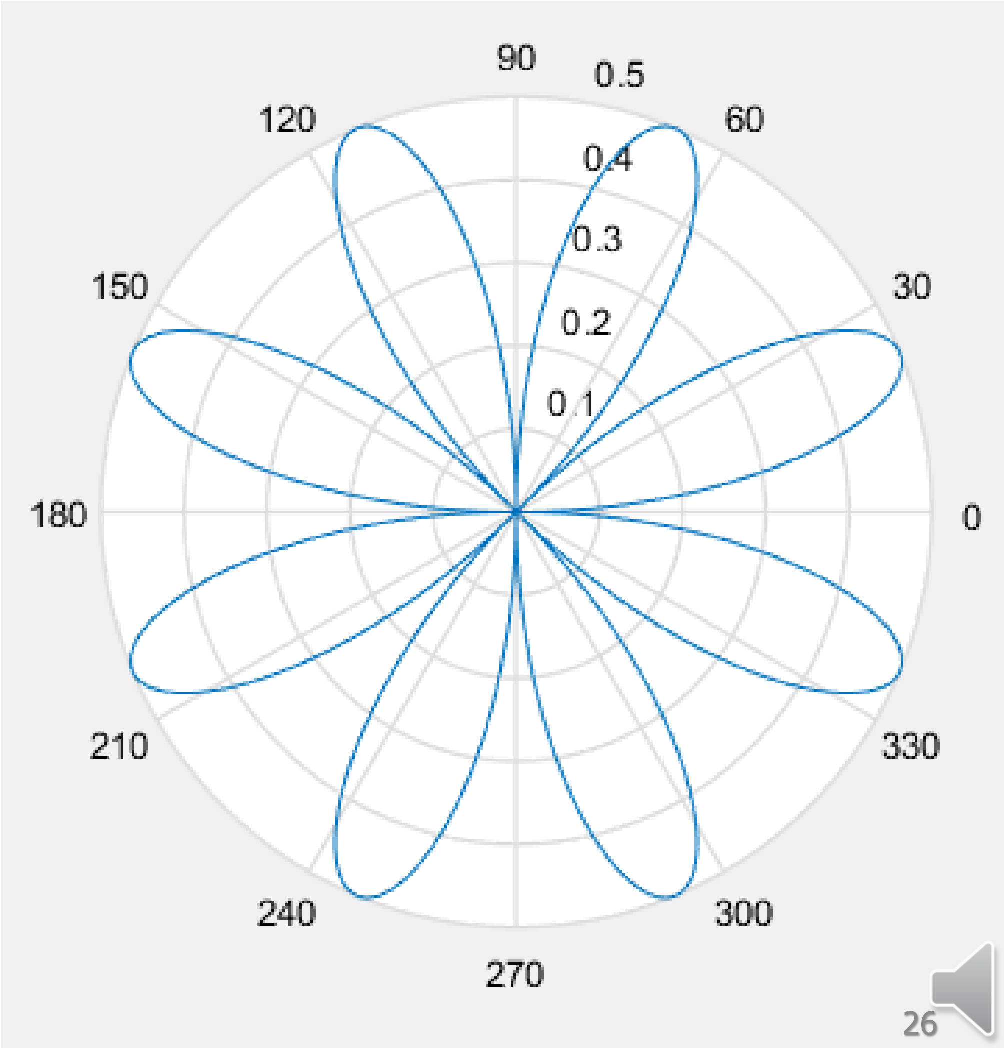
# Polar plots

```
x = 0:0.01:pi*2;
r = cos(x).*sin(2*x);
figure
polar(x,r);
```



# Polar plots

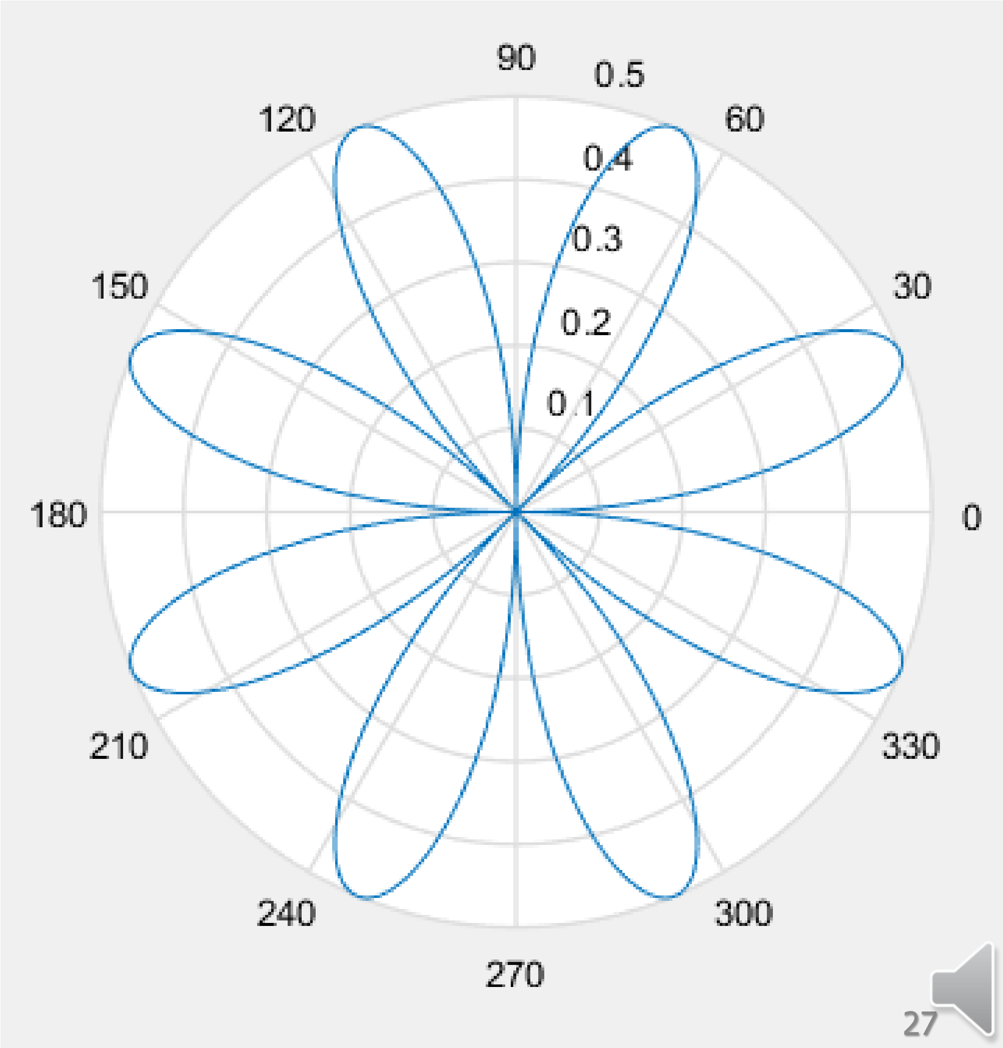
```
x = 0:0.01:pi*2;
r = cos(2*x).*sin(2*x);
figure
polar(x,r);
```



# Polar plots

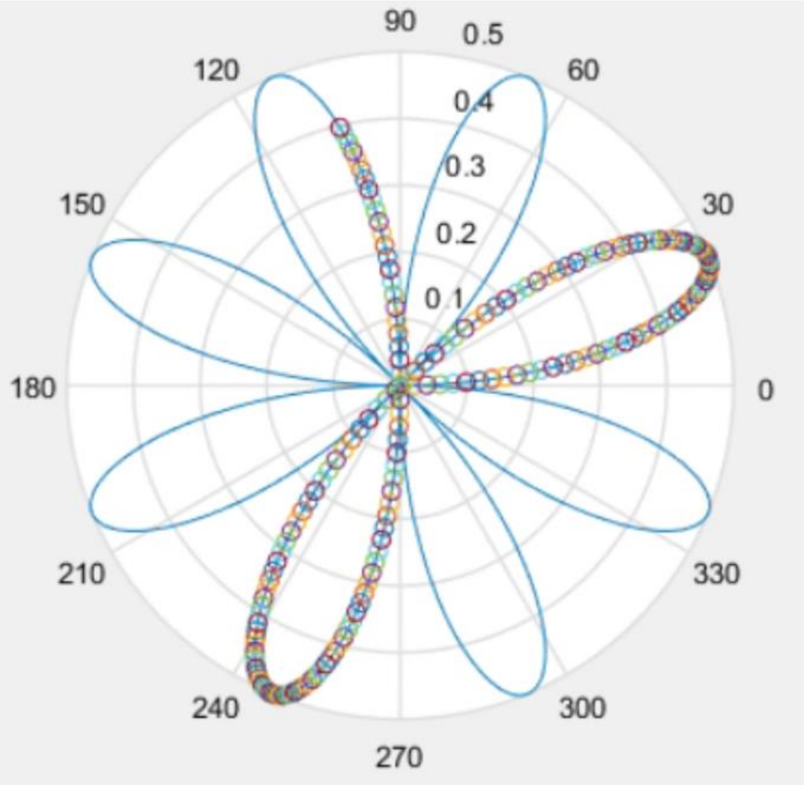
```
x = 0:0.01:pi*2;
r = cos(2*x).*sin(2*x);
figure
polar(x,r);
```

How is it drawn?



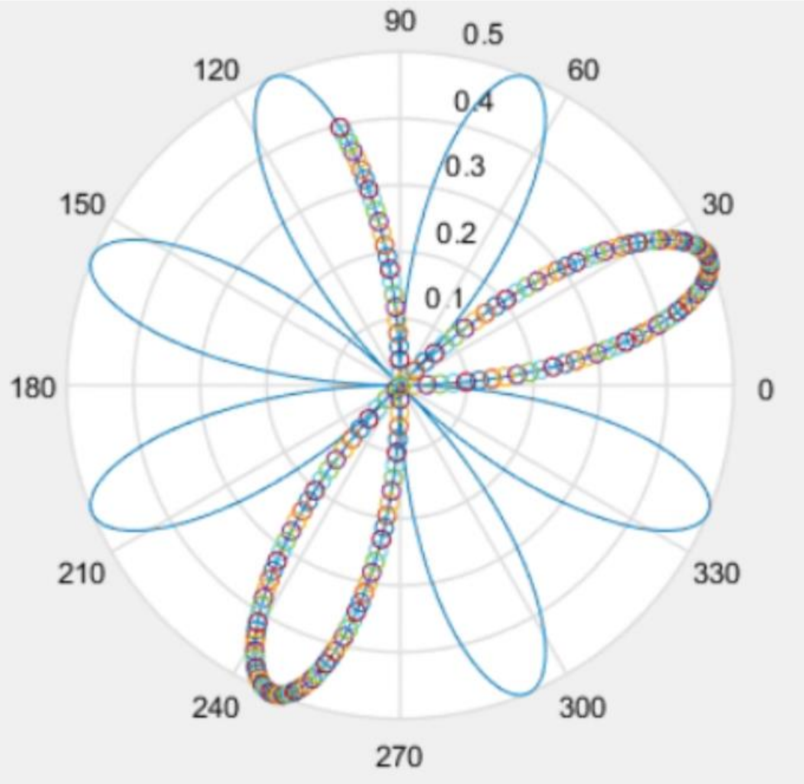
# Demo
# Polar plots. Flower

```
clear;close all;
x = 0:0.01:pi*2;
r = cos(2*x).*sin(2*x);
polar(x,r);
input('Press ENTER to
start.');
hold on;
for x0 = x
  r0 = cos(2*x0) .*
sin(2*x0);
  polar(x0,r0,'o');
  pause(0.033);
end
hold off;
```



# Demo
# Polar plots. Flower

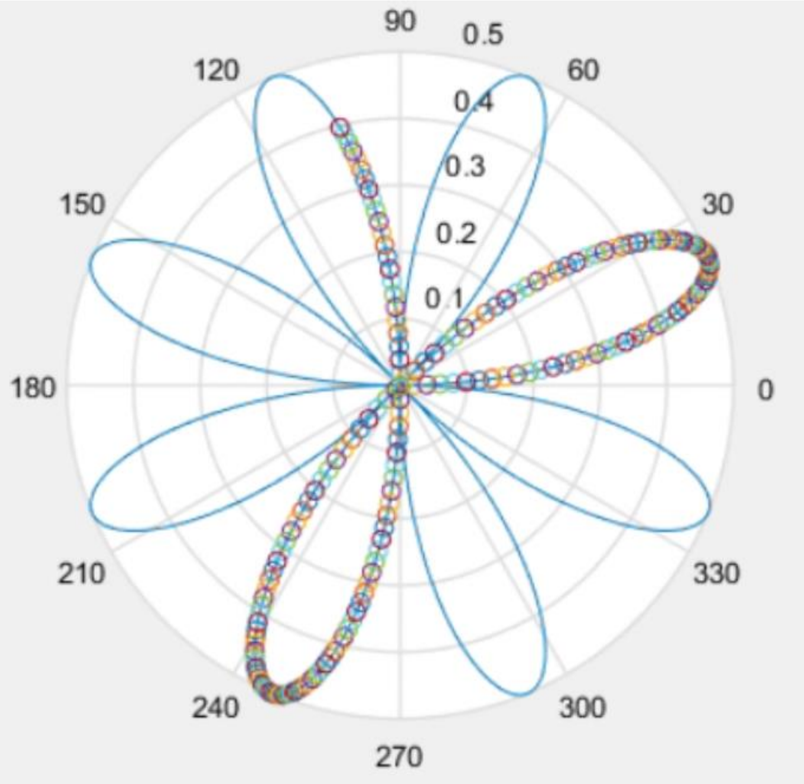
```
clear;close all;
x = 0:0.01:pi*2;
r = cos(2*x).*sin(2*x);
polar(x,r);
input('Press ENTER to start.');
hold on;
for x0 = x
  r0 = cos(2*x0) .* sin(2*x0);
  polar(x0,r0,'o');
  pause(0.033);
end
hold off;
```



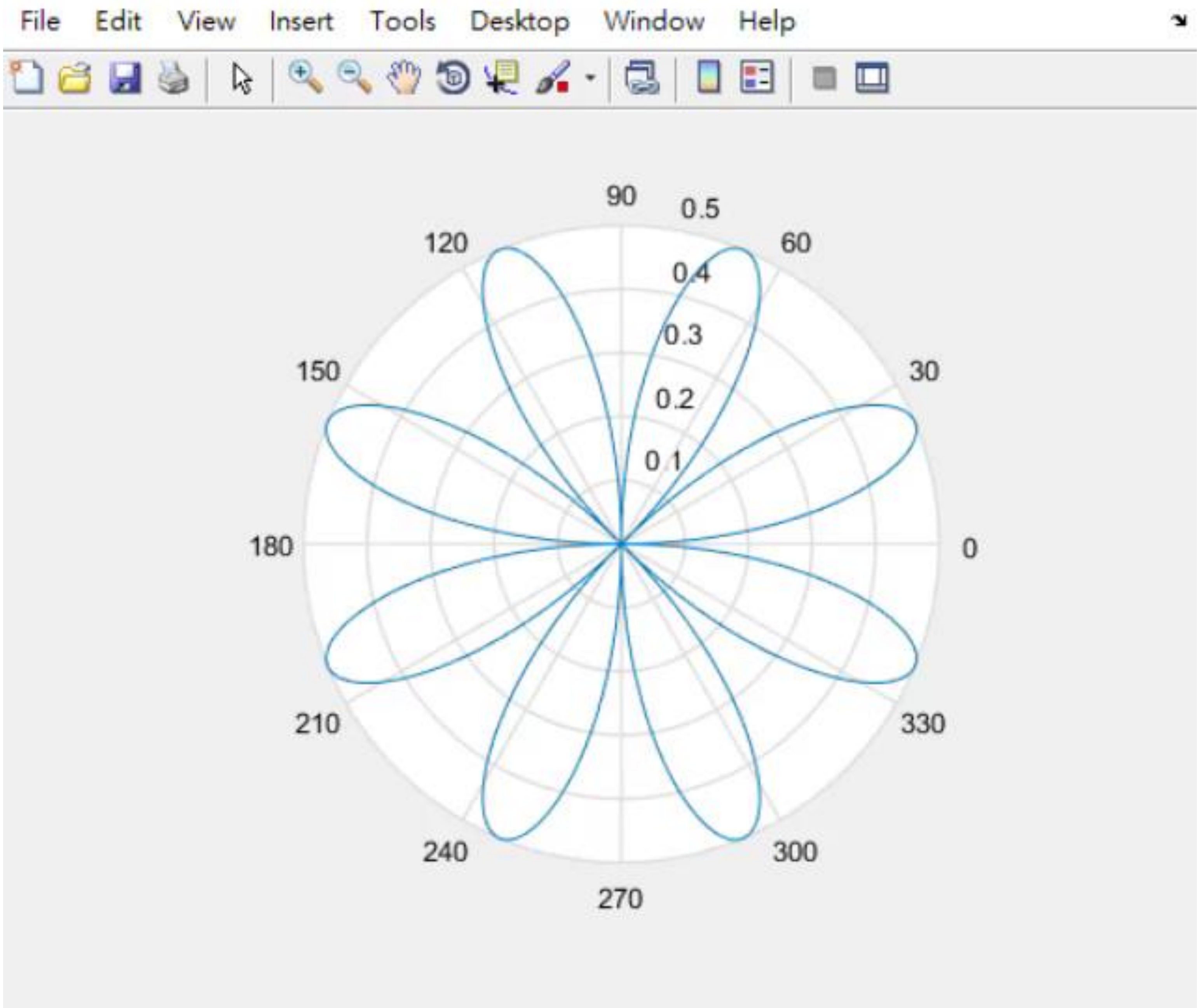
# Demo: Polar plots. Flower

```
for x0 = x
  r0 = cos(2*x0) .*  sin(2*x0);
  polar(x0,r0,'o');
  pause(0.033);
end
hold off;
```

```
s = size(x);
index = 1:s(2)
for i = index
  x0 = x(i);
  r0 = r(i);
  polar(x0,r0,'o');
  pause(0.033);
end
hold off;
```

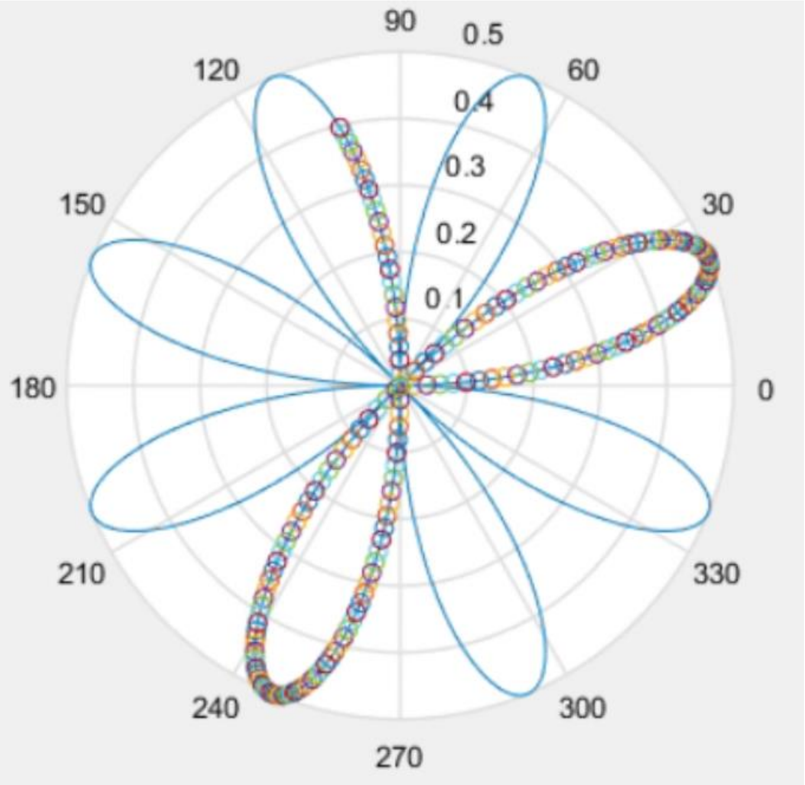


# Demo: Polar plots. Flower



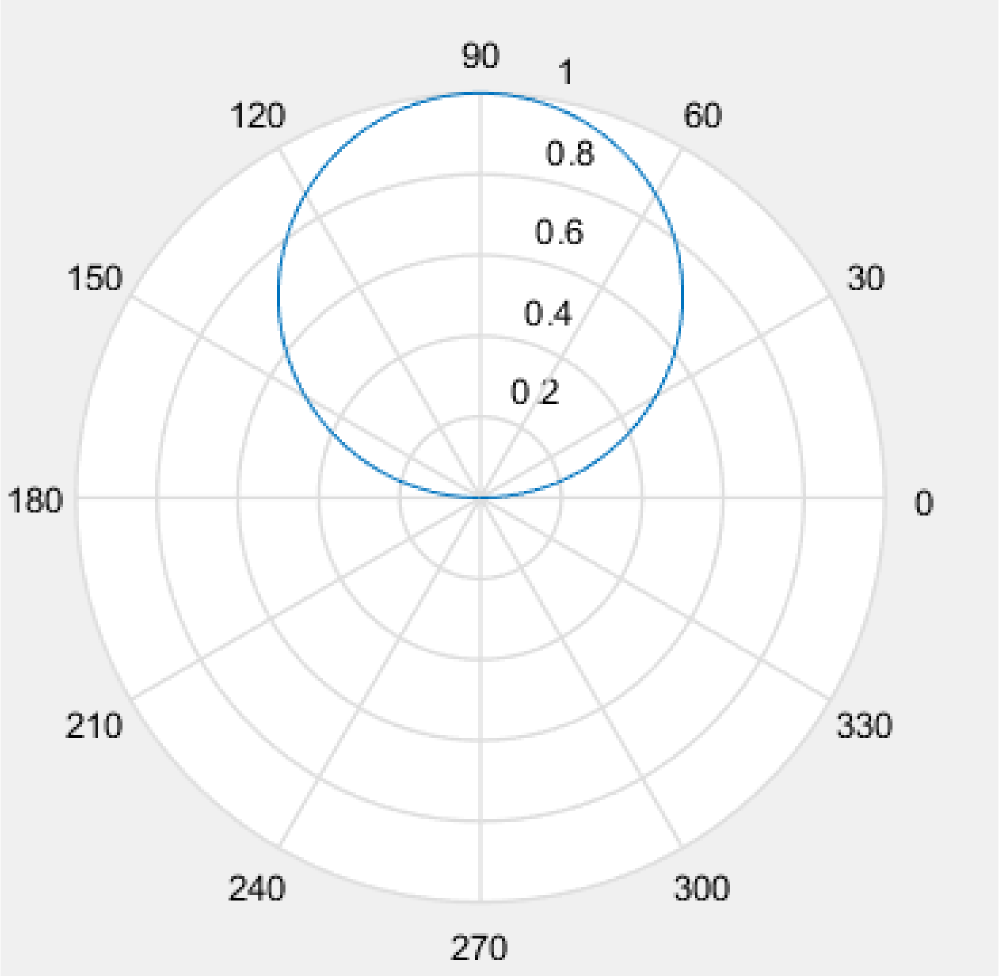
# The Problems: Show only one point?

```
clear;close all;
x = 0:0.01:pi*2;
r = cos(2*x).*sin(2*x);
polar(x,r);
input('Press ENTER to
start.');
hold on;
for x0 = x
  r0 = cos(2*x0) .*
sin(2*x0);
  polar(x0,r0,'o');
  pause(0.033);
end
hold off; %modify the program
to draw one circle only.
```

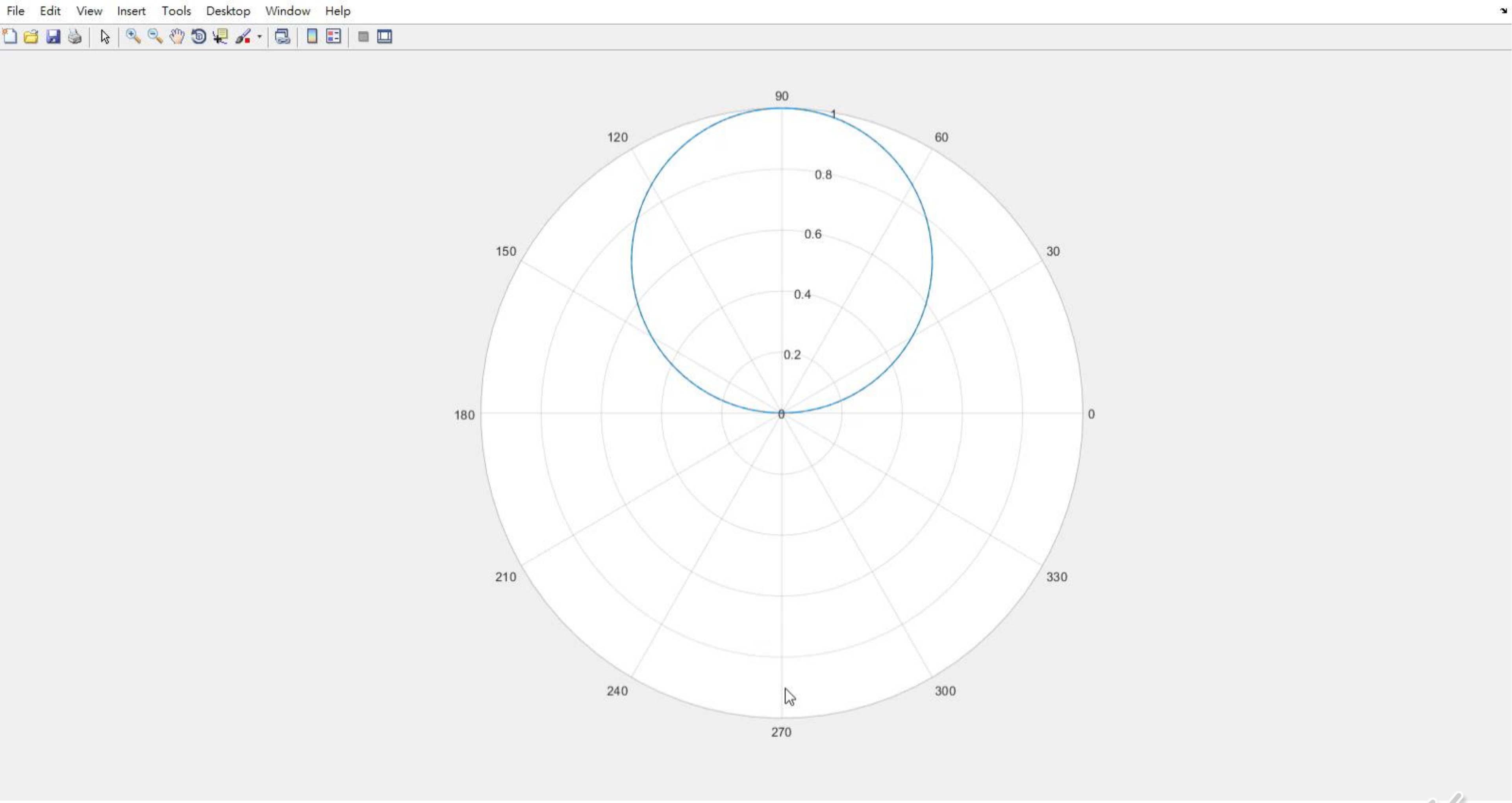


# Trace a point along the curve

```
x = 0:0.01:2*pi;
r = sin(x);
polar(x,r);
```

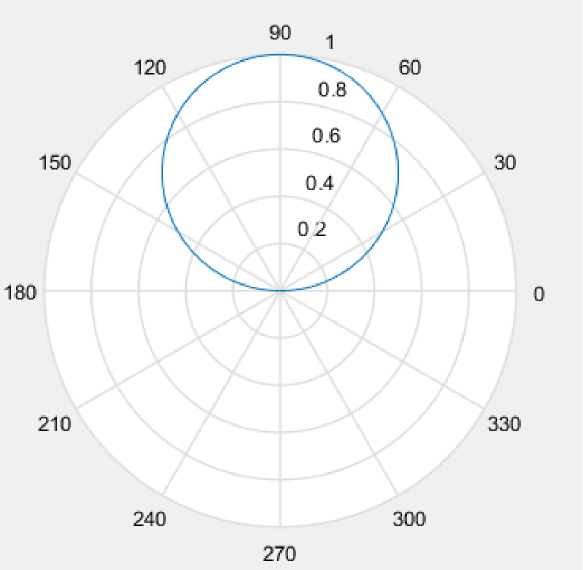


# Trace a point along the curve

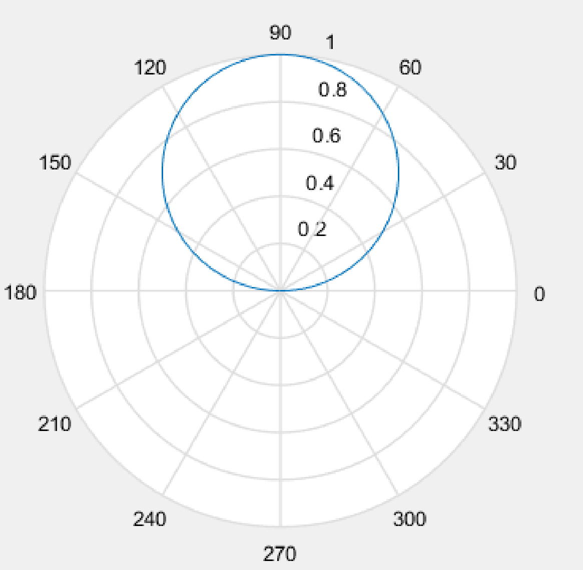


# Trace a point along the curve

```
x = linspace(0, pi*2); % 100 samples
r = sin(x);
h1 = figure          % figure handle
polarplot(x,r);
input('Press ENTER to start.');
polarplot(x,r);

for x0 = x
    clf
    % must draw a polar plot first before calling hold on;
    polarplot(x,r);
    hold on
    r0 = sin(x0);
    polarscatter(x0,r0,'filled', 'b');
    pause(0.033);
end
hold off;
```

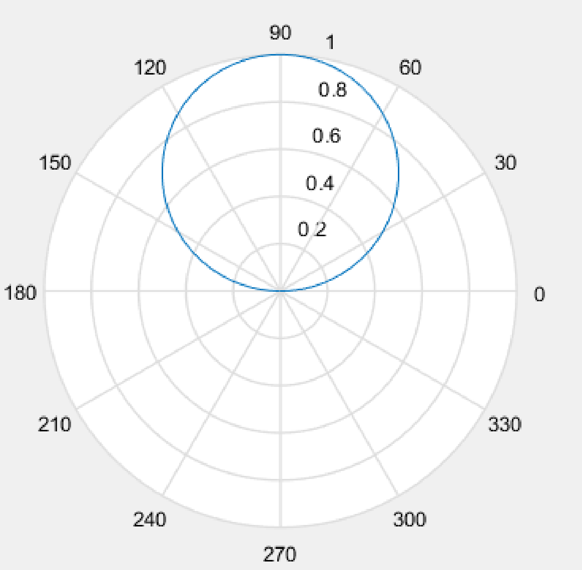


# Trace a point along the curve



```
x = linspace(0, pi*2); % 100 samples
r = sin(x);
h1 = figure
polarplot(x,r);
input('Press ENTER to start.');
polarplot(x,r);

for x0 = x
    clf             % clear figure content
    % must draw a polar plot first before calling hold on;
    polarplot(x,r);
    hold on
    r0 = sin(x0);
    polarscatter(x0,r0,'filled', 'b');
    pause(0.033);
end
hold off;
```

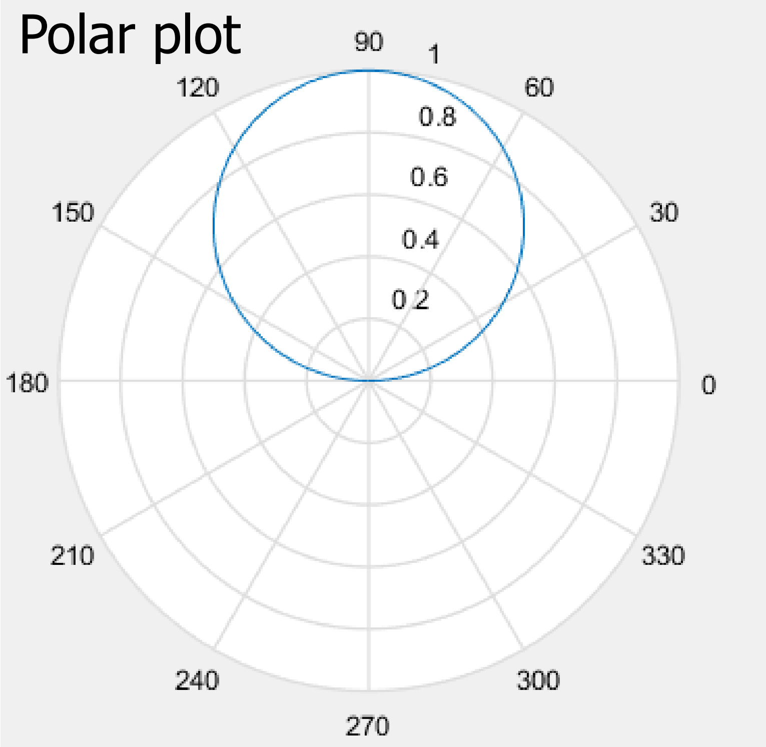
# Trace a point along the curve



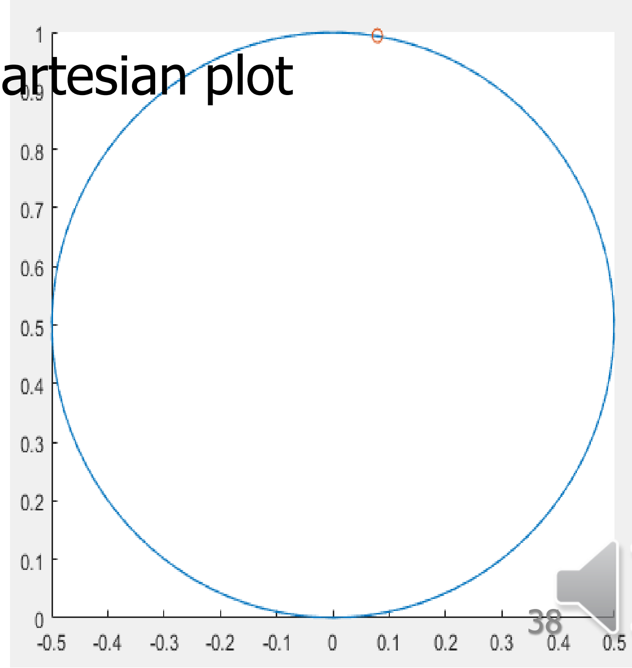
```
x = linspace(0, pi*2); % 100 samples
r = sin(x);
h1 = figure
polarplot(x,r);
input('Press ENTER to start.');
polarplot(x,r);

for x0 = x
    clf             % clear figure content
    % must draw a polar plot first before calling hold on;
    polarplot(x,r);
    hold on
    r0 = sin(x0);
    polarscatter(x0,r0,'filled', 'b');
    pause(0.033);
end
hold off;
%if clf is not used, the figure will contain
all the data that have been drawn.
```

# Trace a point along the curve

```
x = linspace(0, pi*2);
r = sin(x);
h1 = figure
polarplot(x,r);
input('Press ENTER to start.');
polarplot(x,r);

for x0 = x
    clf         % change to Cartesian
    hold on
    polar(x,r);
    r0 = sin(x0);
    polar(x0,r0, 'o');
    pause(0.033);
end
hold off;
```



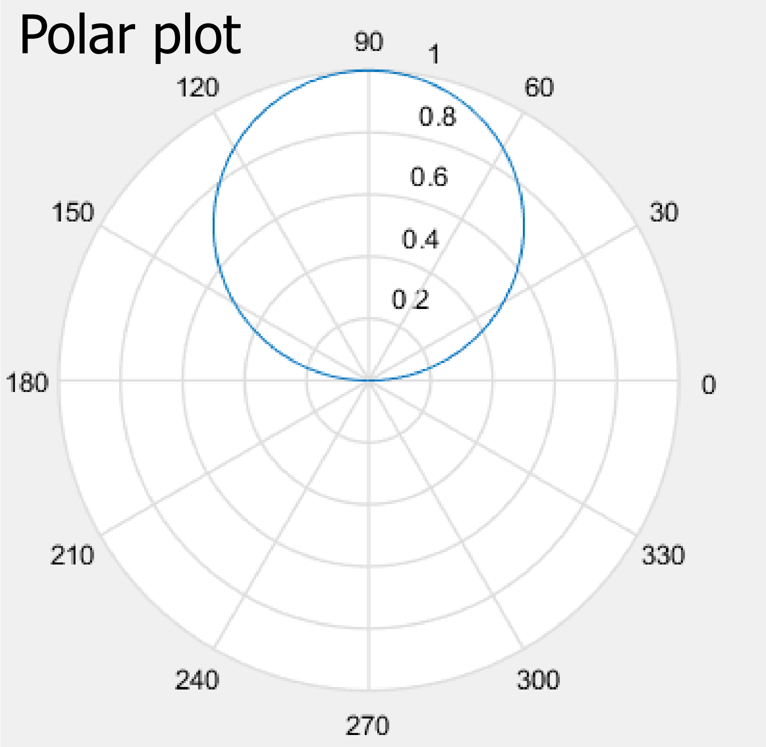
Polar plot



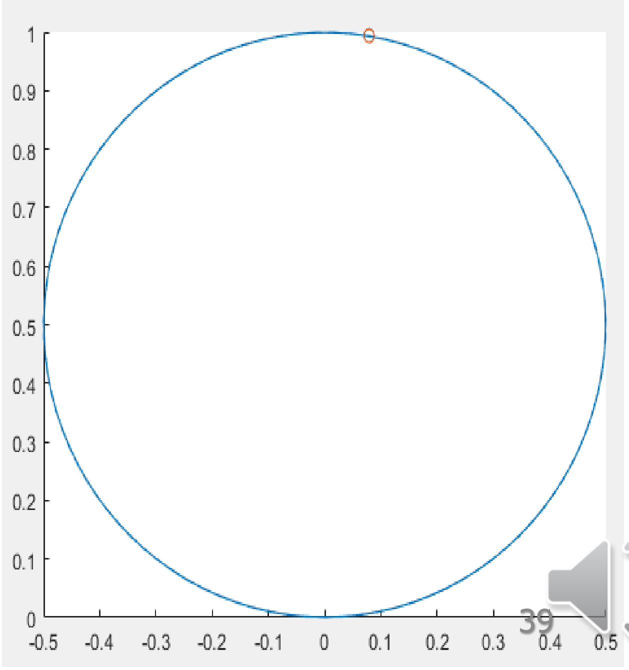
Cartesian plot

# Trace a point along the curve

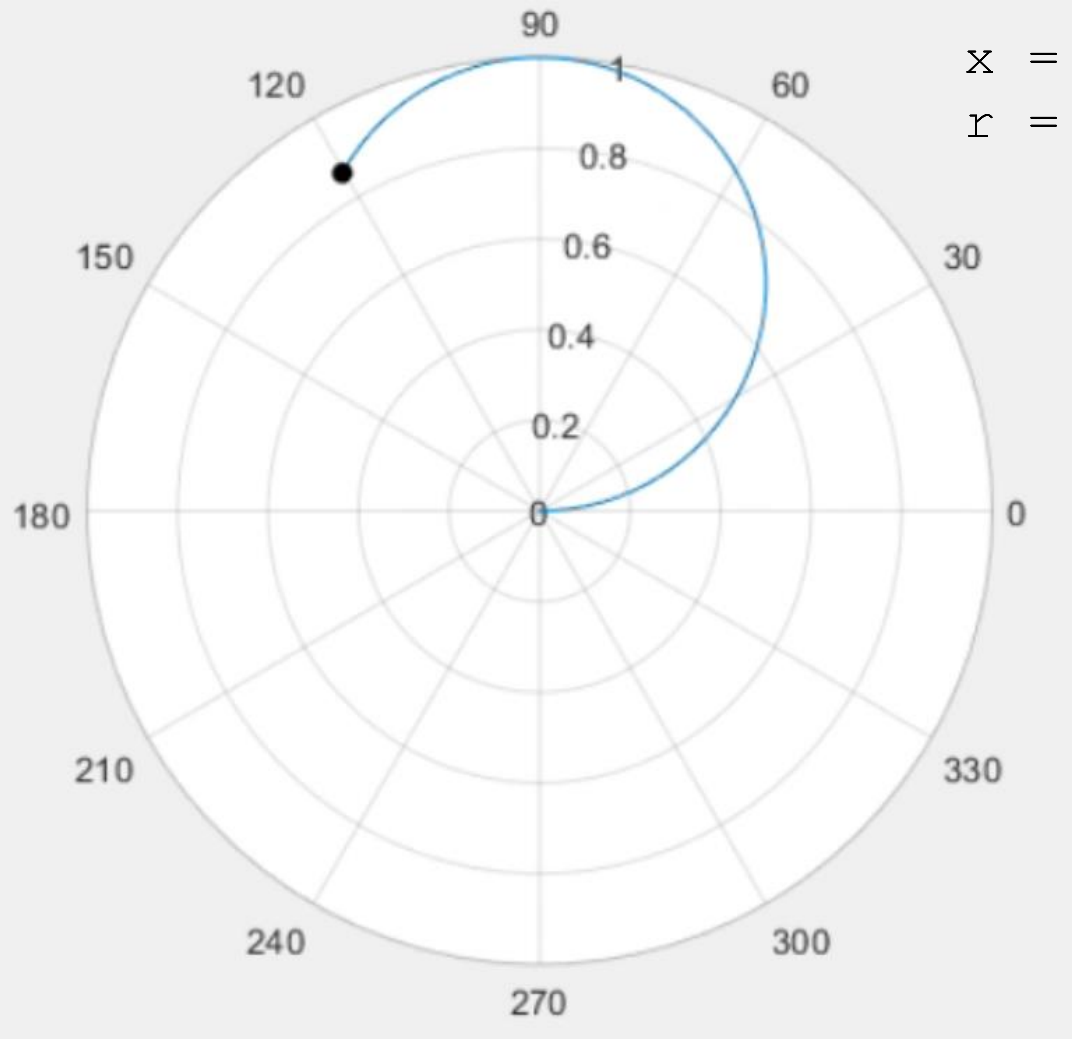
```
x = linspace(0, pi*2);
r = sin(x);
h1 = figure
polarplot(x,r);
input('Press ENTER to start.');
polarplot(x,r);

for x0 = x
    clf         % change to Cartesian
    hold on
    polar(x,r);
    r0 = sin(x0);
    polar(x0,r0, 'o');
    pause(0.033);
end
hold off;
```


Polar plot

Cartesian plot

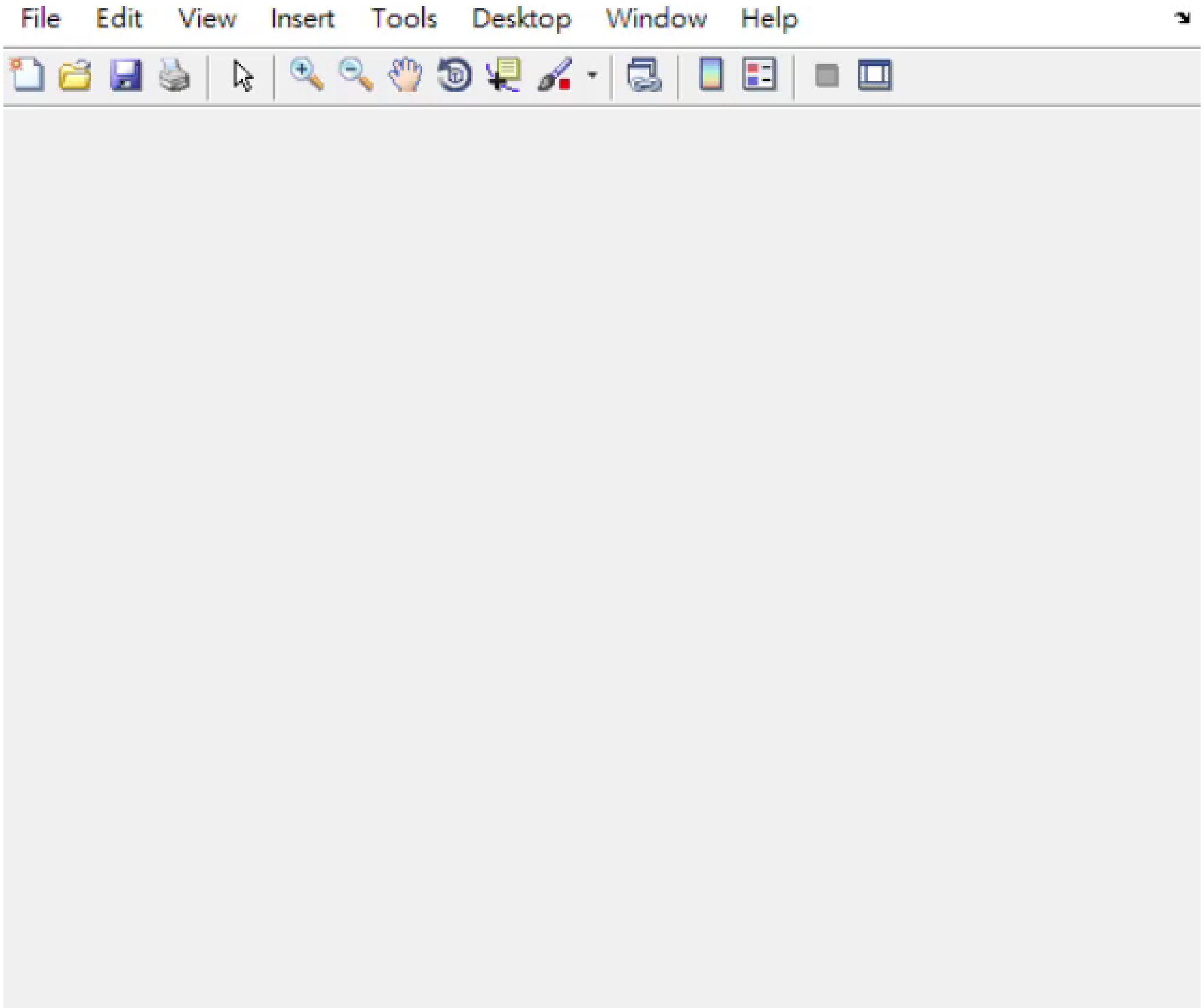


# Draw a curve interactively while tracing a point along the curve



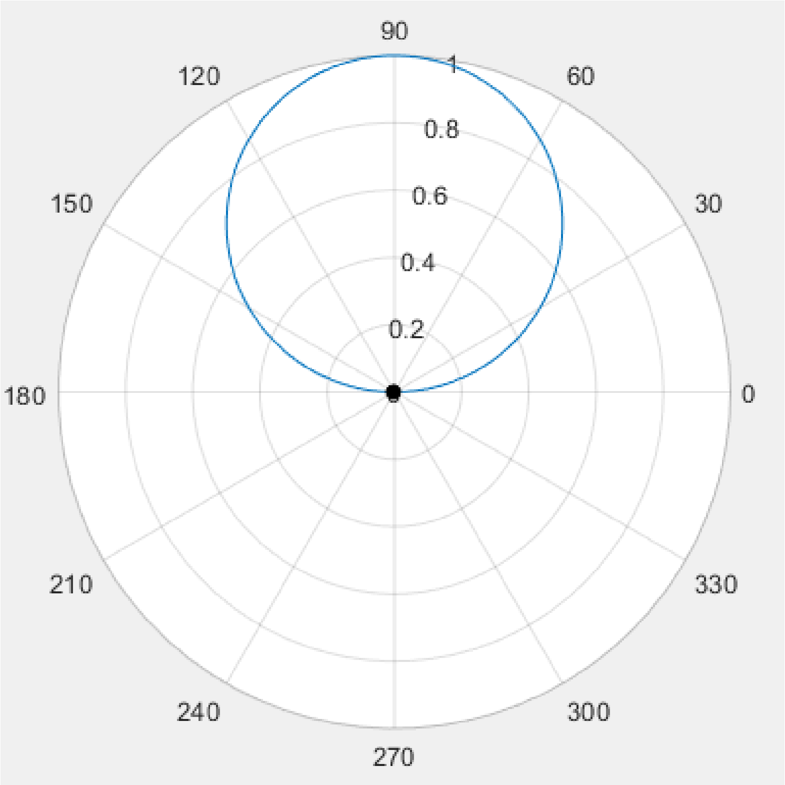
```
x = linspace(0, pi*2);
r = sin(x);
```

```
x = linspace(0, pi*2);
r = sin(x);
```



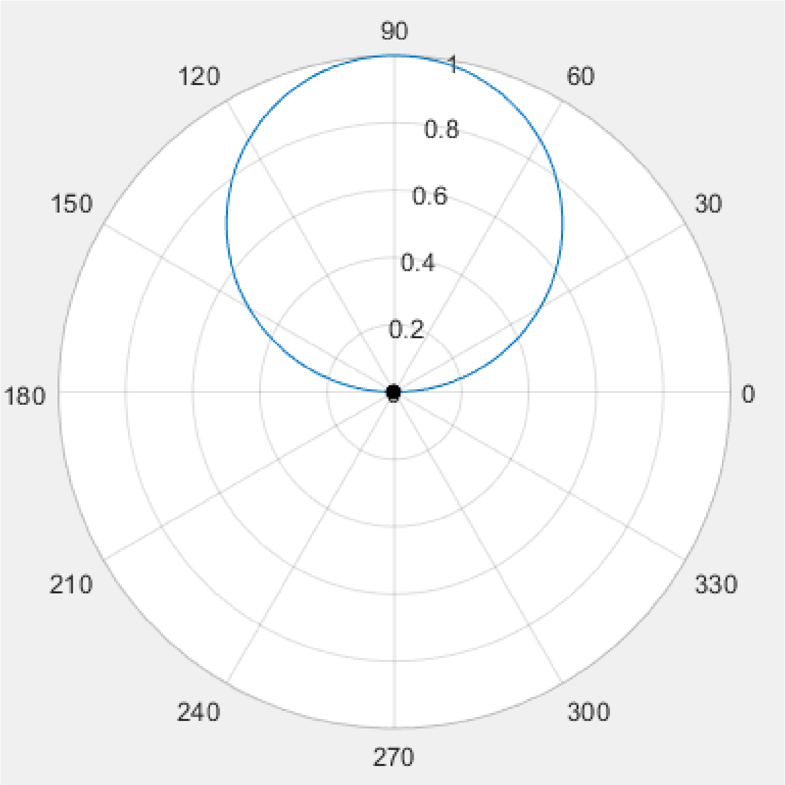
# Draw a curve interactively while tracing a point along the curve

```
X = 0:0.01:pi*2;
h1 = figure
input('Press ENTER to start.');

x = zeros(size(X));
r = zeros(size(X));
i = 1;
for x0 = X
    clf
    r0 = sin(x0);
    x(i) = x0; r(i) = r0;
    polarplot(x(1:i),r(1:i));
    hold on
    polarscatter(x0,r0,'filled','k');
    pause(0.033);
    i = i + 1;
end
hold off;;
```



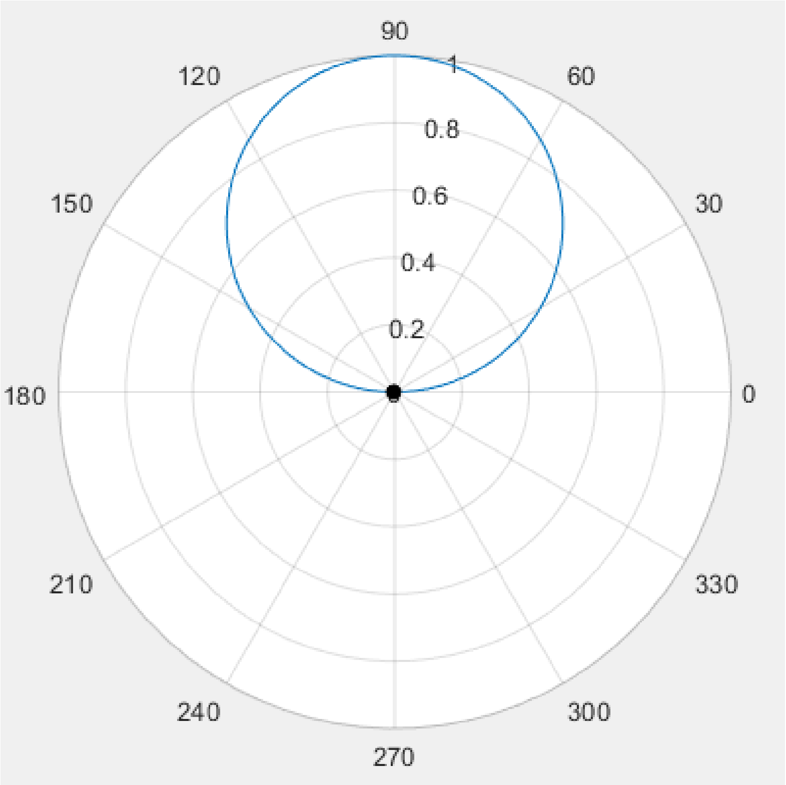
# Draw a curve interactively while tracing a point along the curve

```
X = 0:0.01:pi*2;
h1 = figure
input('Press ENTER to start.');

x = zeros(size(X));
r = zeros(size(X));
i = 1;
for x0 = X
    clf
    r0 = sin(x0);
    x(i) = x0; r(i) = r0;
    polarplot(x(1:i),r(1:i));
    hold on
    polarscatter(x0,r0,'filled','k');
    pause(0.033);
    i = i + 1;
end
hold off;;
```

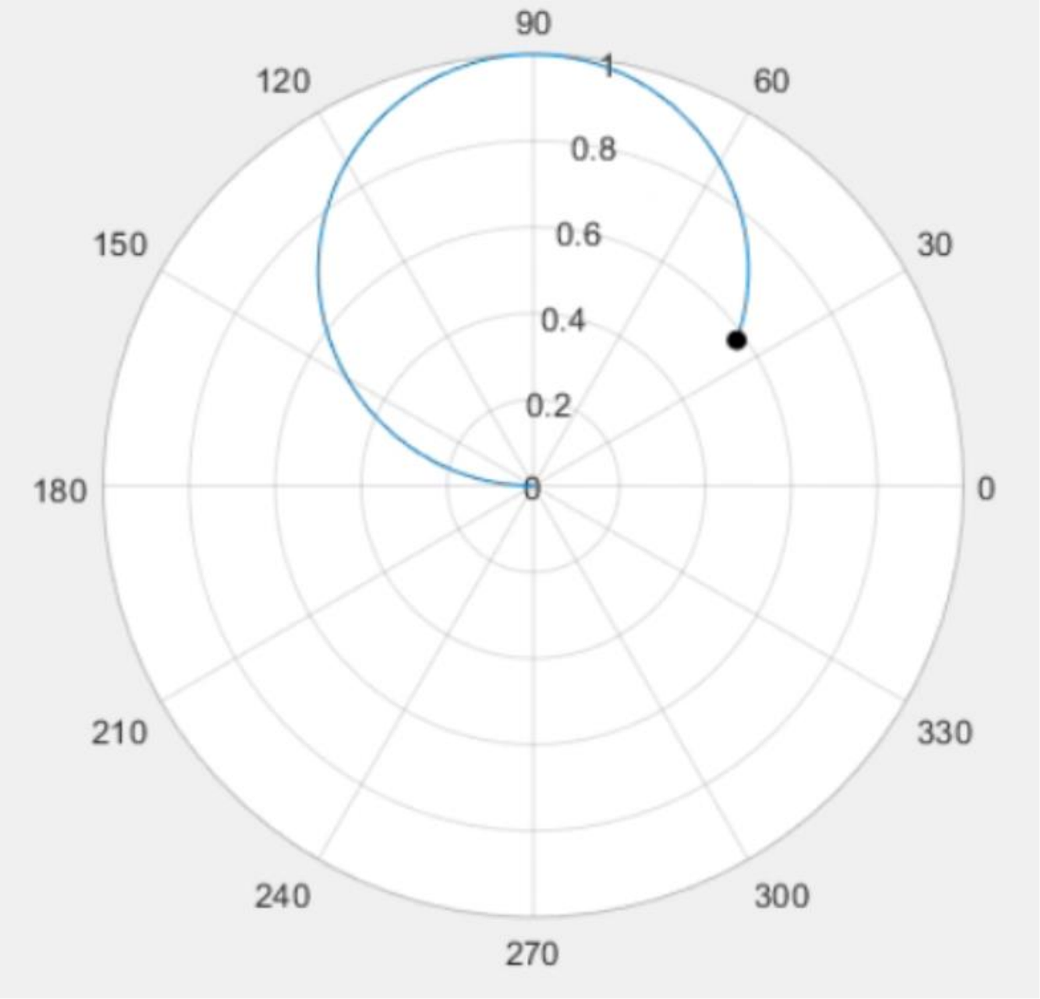


# Draw a curve interactively while tracing a point along the curve

```
X = 0:0.01:pi*2;
h1 = figure
input('Press ENTER to start.');

x = zeros(size(X));
r = zeros(size(X));
i = 1;          // point index
for x0 = X
    clf
    r0 = sin(x0);
    x(i) = x0; r(i) = r0;
    polarplot(x(1:i),r(1:i));
    hold on
    polarscatter(x0,r0,'filled','k');
    pause(0.033);
    i = i + 1;
end
hold off;;
```

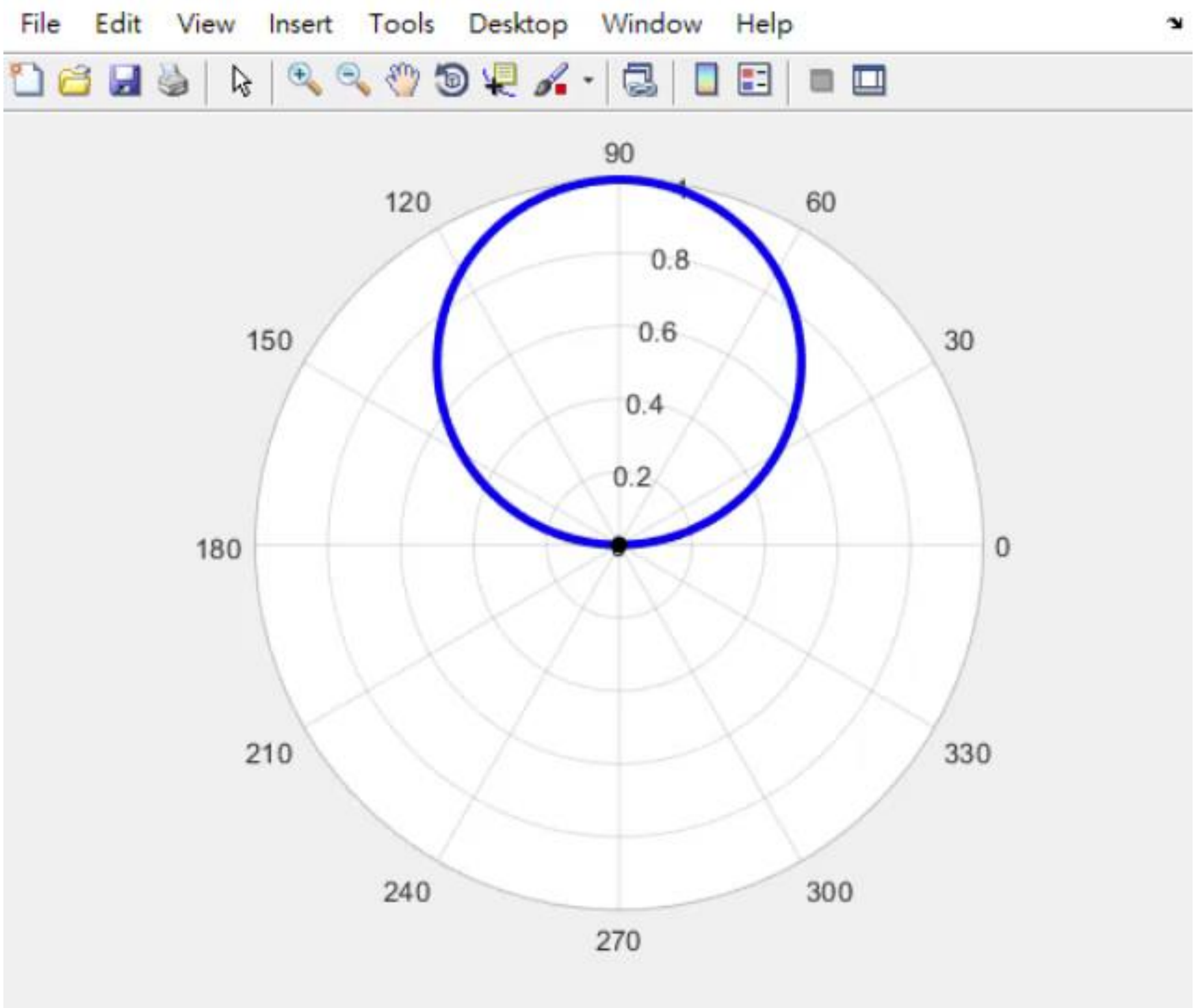


# Erase a curve interactively while tracing a point along the curve



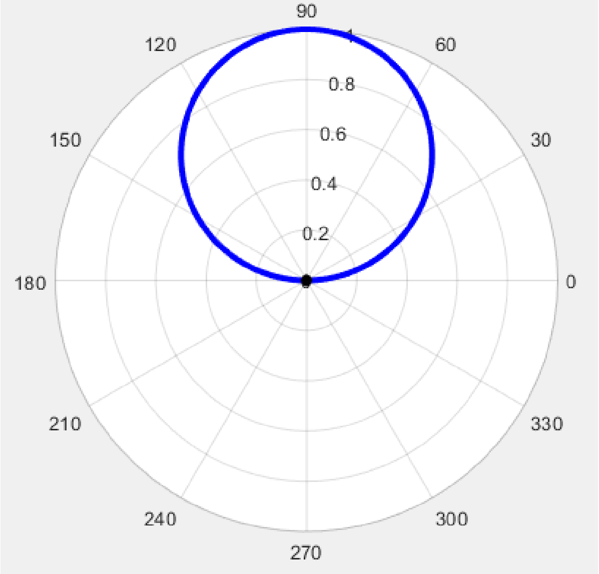
```
= linspace(0, pi);
= sin(x);
```

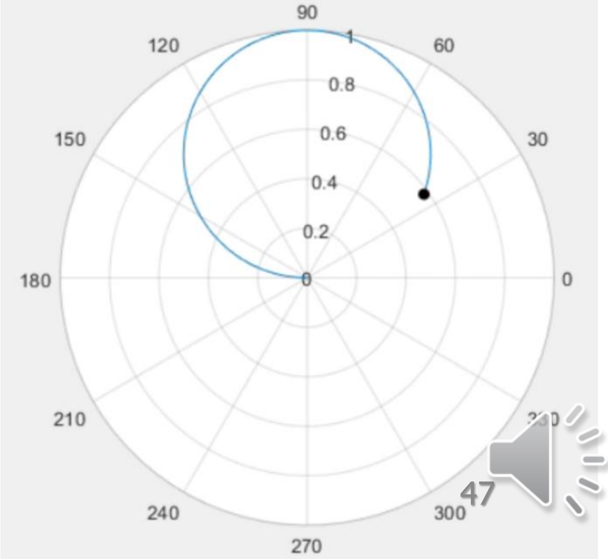
```
x = linspace(0, pi);
r = sin(x);
```



# Erase a curve interactively while tracing a point along the curve

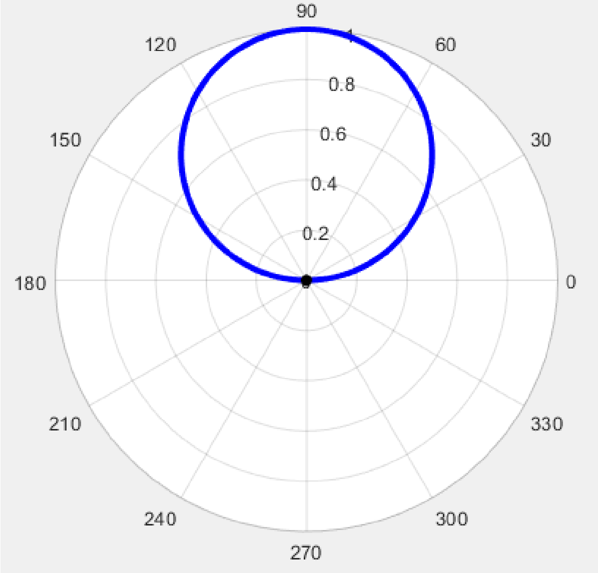
```
x = 0:0.01:pi; r = sin(x);
h1 = figure
polarplot(x(1:end),r(1:end), ...
    'LineWidth', 3, 'color', 'b');
hold on
ps = polarscatter(0,0,'filled','k');
input('Press ENTER to start.');
                    % (x(1), r(1), …
i = 1;
for x0 = x
    clf
    r0 = sin(x0);
    polarplot(x(i:end),r(i:end));
    hold on

    polarscatter(x0,r0,'filled','k');
    pause(0.033);
    i = i + 1;
end
```

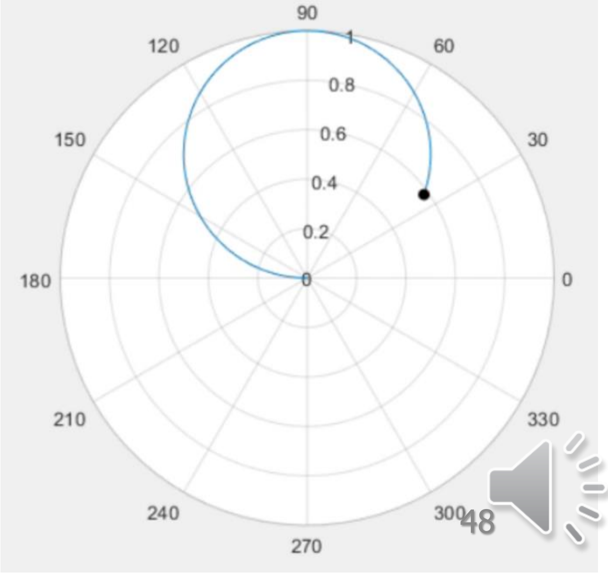




# Erase a curve interactively while tracing a point along the curve

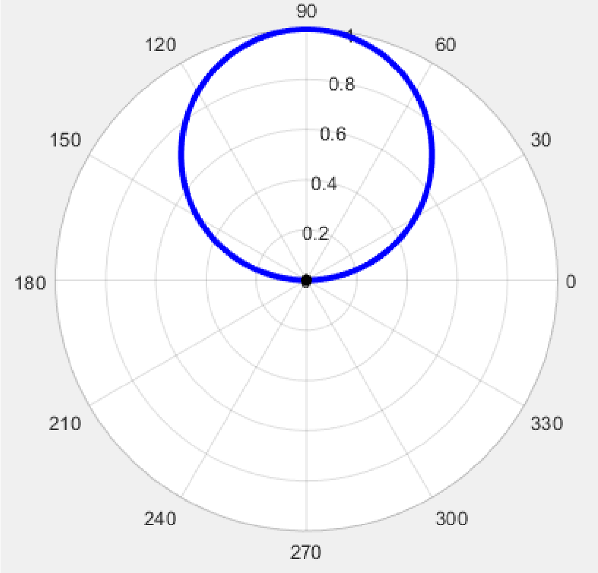
```
x = 0:0.01:pi; r = sin(x);
h1 = figure
polarplot(x(1:end),r(1:end), ...
    'LineWidth', 3, 'color', 'b');
hold on
ps = polarscatter(0,0,'filled','k');
input('Press ENTER to start.');

i = 1;                    // start point index
for x0 = x
    clf
    r0 = sin(x0);
    polarplot(x(i:end),r(i:end));
    hold on

    polarscatter(x0,r0,'filled','k');
    pause(0.033);  // before a new frame
    i = i + 1;
end
```

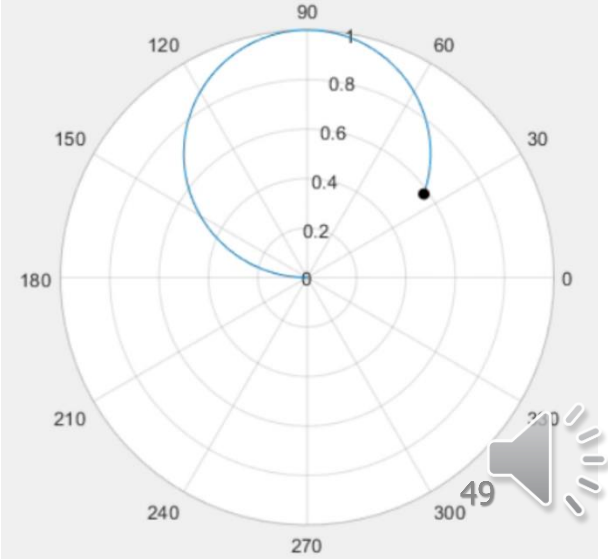




# Erase a curve interactively while tracing a point along the curve

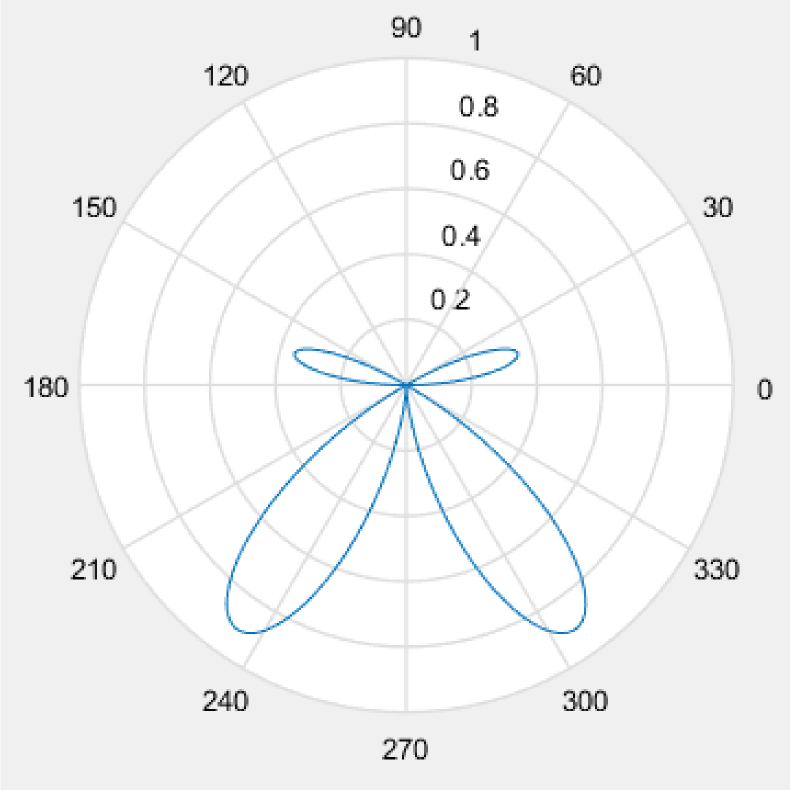
```
x = 0:0.01:pi; r = sin(x);
h1 = figure
polarplot(x(1:end),r(1:end), ...
    'LineWidth', 3, 'color', 'b');
hold on
ps = polarscatter(0,0,'filled','k');
input('Press ENTER to start.');
                    % (x(1), r(1), ...
i = 1;
for x0 = x
    clf
    r0 = sin(x0);
    polarplot(x(i:end),r(i:end));
    hold on

    polarscatter(x0,r0,'filled','k');
    pause(0.033);
    i = i + 1;
end
```





# Polar plots. Exercise
# Do you know how the curve is drawn?

```
x = 0:0.01:pi*2;
r = cos(3*x).*sin(2*x);
figure
polar(x,r);
```



Tracing a point along the curve?

Draw the curve interactively while tracing a point?

Erase the curve interactively while tracing a point?

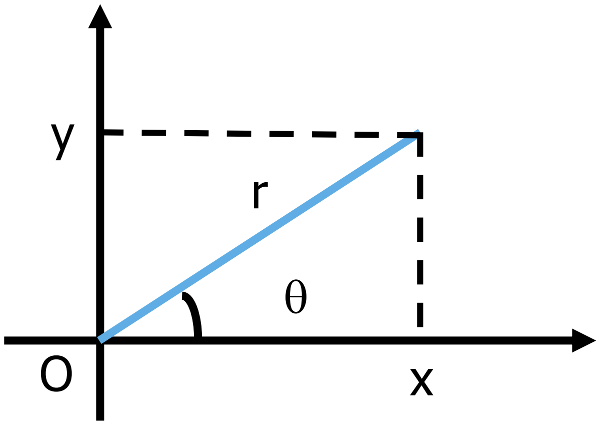
# Polar coordinates and Cartesian coordinates conversion

Polar coordinates: (r,θ)

Cartesian coordinates: (x,y)

r = $\sqrt{x^2 + y^2}$

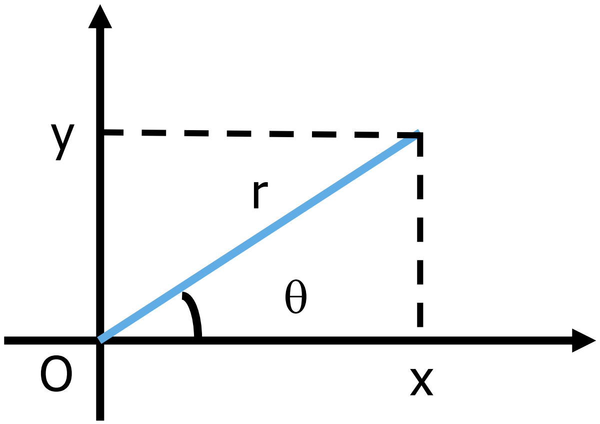
θ = $\tan^{-1}(y/x)$



# Polar coordinates and Cartesian coordinates conversion

Polar coordinates: $(r,\theta)$

Cartesian coordinates: $(x,y)$



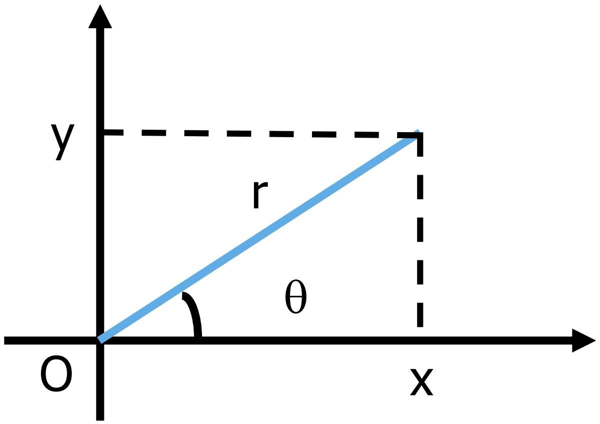
r = $\sqrt{x^2 + y^2}$

$\theta = \tan^{-1}(y/x)$

# Polar coordinates and Cartesian coordinates conversion

Polar coordinates: $(r,\theta)$

Cartesian coordinates: $(x,y)$



$$r = \sqrt{x^2 + y^2}$$

$$\theta = \tan^{-1}(y/x)$$

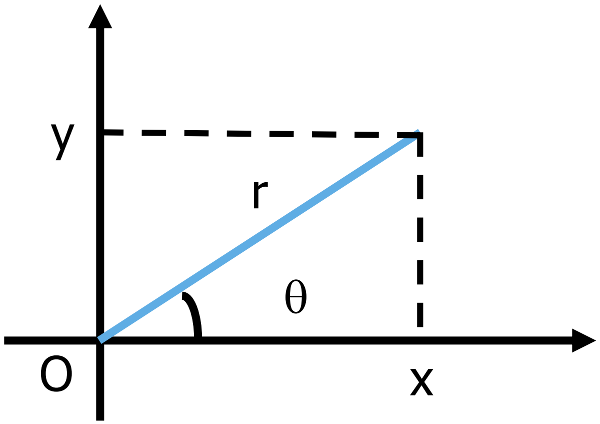
# Polar coordinates and Cartesian coordinates conversion

Polar coordinates: $(r,\theta)$

Cartesian coordinates: $(x,y)$

r = $\sqrt{x^2 + y^2}$

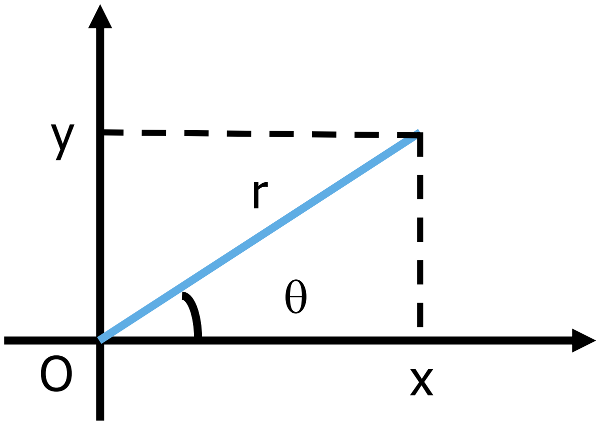
$\theta = \tan^{-1}(y/x)$



# Polar coordinates and Cartesian coordinates conversion

Polar coordinates: $(r,\theta)$

Cartesian coordinates: $(x,y)$



r = $\sqrt{x^2 + y^2}$

$\theta = \tan^{-1}(y/x)$

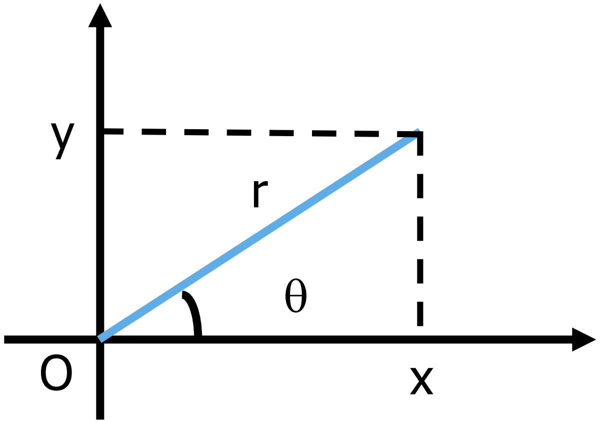
$x=r\cos\theta$

$y=r\sin\theta$

# Polar coordinates and Cartesian coordinates conversion

Polar coordinates: (r,θ)

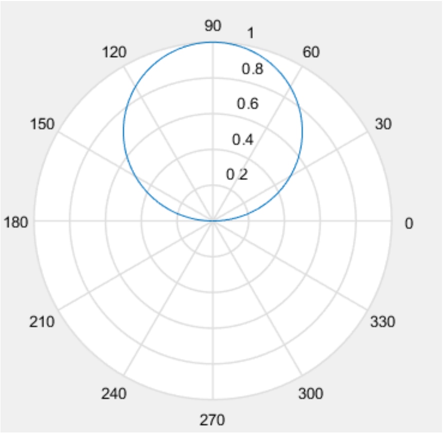
Cartesian coordinates: (x,y)



$$r = \sqrt{x^2 + y^2}$$

$$\theta = \tan^{-1}(y/x)$$

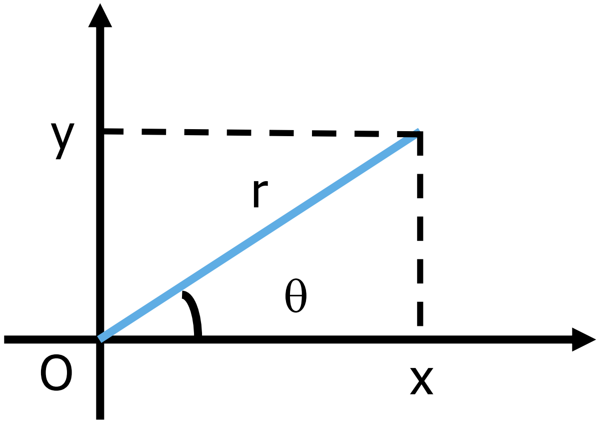
```
x = 0:0.01:pi;
r = sin(x);
polar(x,r);
%don’t confuse about x.
%Here, x is the angle
```



# Polar coordinates and Cartesian coordinates conversion

Polar coordinates: $(r,\theta)$

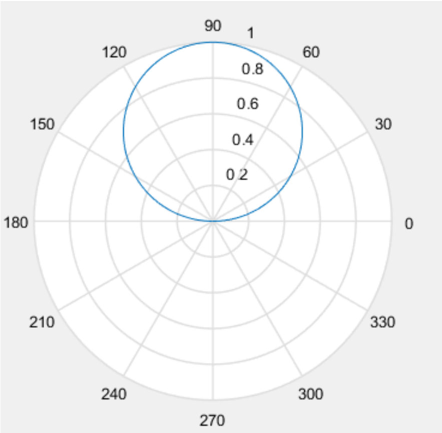
Cartesian coordinates: $(x,y)$



$$r = \sqrt{x^2 + y^2}$$

$$\theta = \tan^{-1}(y/x)$$

```
θ = 0:0.01:pi;
r = sin(θ);
polar(θ,r);
```



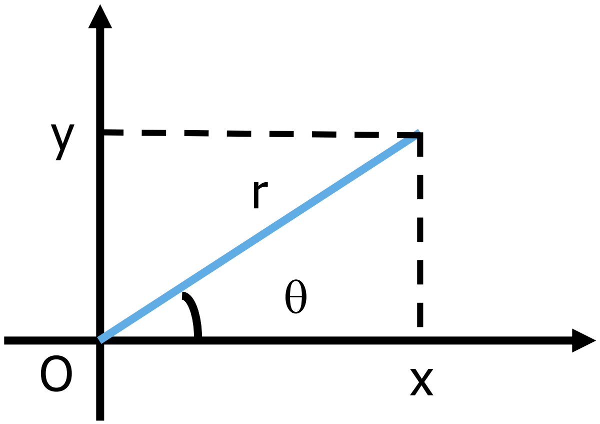
# Polar coordinates and Cartesian coordinates conversion

Polar coordinates: $(r,\theta)$

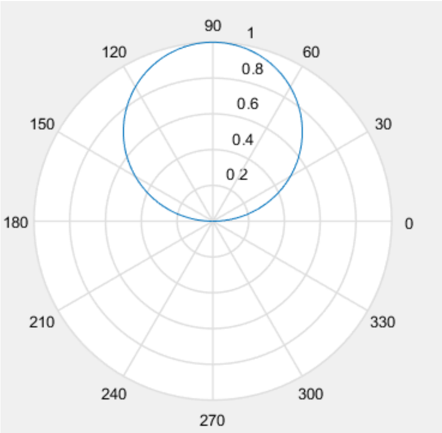
Cartesian coordinates: $(x,y)$

$$r = \sqrt{x^2 + y^2}$$

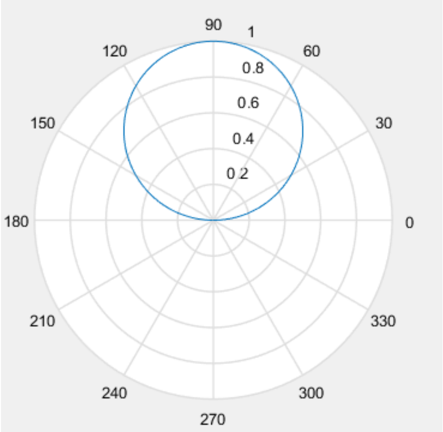
$$\theta = \tan^{-1}(y/x)$$



```
θ = 0:0.01:pi;
r = sin(θ);
polar(x,r);
x = r cosθ
y = r sinθ
```



# Polar coordinates and Cartesian coordinates conversion



Polar coordinates: $(r,\theta)$

Cartesian coordinates: $(x,y)$

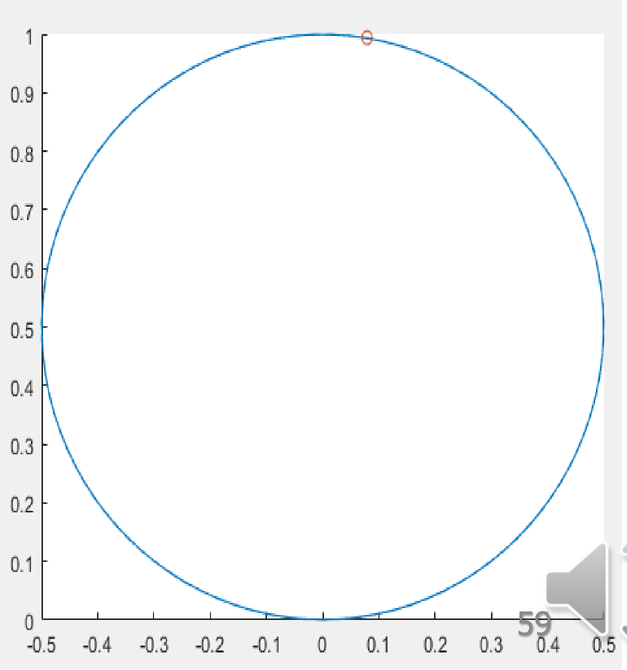
$$x = \sin\theta \cos\theta$$

$$y = \sin\theta \sin\theta$$

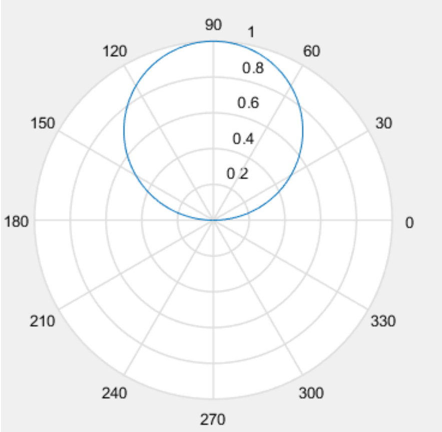
$$\Rightarrow x^2 = \sin^2\theta \cos^2\theta$$

$$\Rightarrow y^2 = \sin^2\theta \sin^2\theta$$

```
θ = 0:0.01:pi;
r = sin(θ);
polar(x,r);
x = r cosθ
y = r sinθ
```



# Polar coordinates and Cartesian coordinates conversion



Polar coordinates: $(r,\theta)$

Cartesian coordinates: $(x,y)$

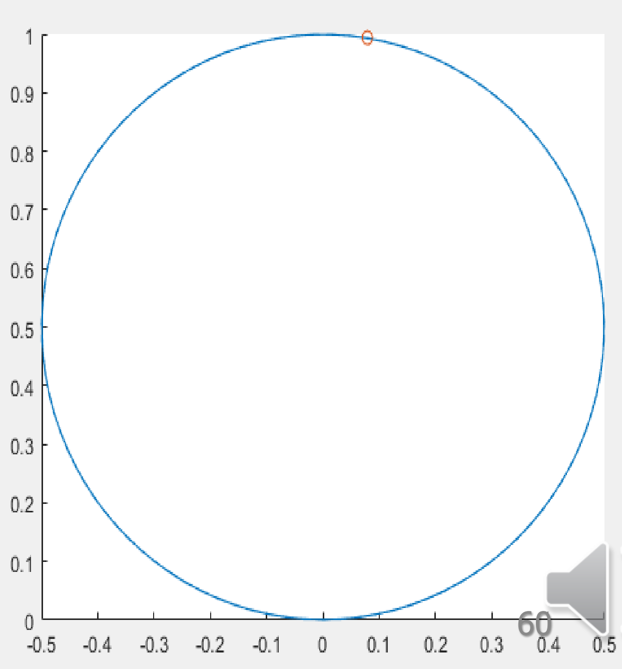
$$x = \sin\theta \cos\theta$$

$$y = \sin\theta \sin\theta$$

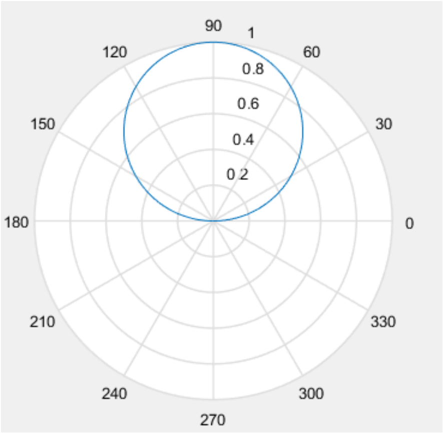
$$\Rightarrow x^2 = \sin^2\theta \cos^2\theta$$

$$\Rightarrow y^2 = \sin^2\theta \sin^2\theta$$

$$\Rightarrow x^2+y^2 = \sin^2\theta$$



# Polar coordinates and Cartesian coordinates conversion

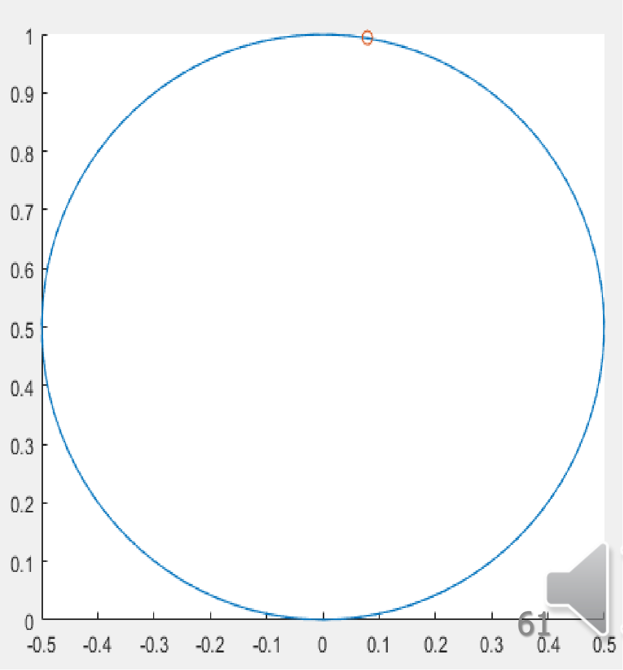


Polar coordinates: $(r,\theta)$

Cartesian coordinates: $(x,y)$

$$x = \sin\theta \cos\theta$$
$$y = \sin\theta \sin\theta$$
$$\Rightarrow x^2 = \sin^2\theta \cos^2\theta$$
$$\Rightarrow y^2 = \sin^2\theta \sin^2\theta$$
$$\Rightarrow x^2+y^2 = \sin^2\theta$$

$$x^2+y^2 = y$$
$$\Rightarrow x^2+(y^2 - y + 0.5^2) = 0.5^2$$
$$\Rightarrow x^2+(y-0.5)^2 = 0.5^2$$



# Polar coordinates and Cartesian coordinates conversion

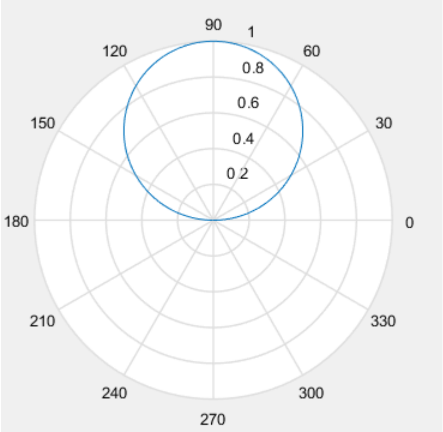
$$x = \sin\theta \cos\theta$$

$$y = \sin\theta \sin\theta$$

$$\Rightarrow x^2 = \sin^2\theta \cos^2\theta$$

$$\Rightarrow y^2 = \sin^2\theta \sin^2\theta$$

$$\Rightarrow x^2+y^2 = \sin^2\theta$$

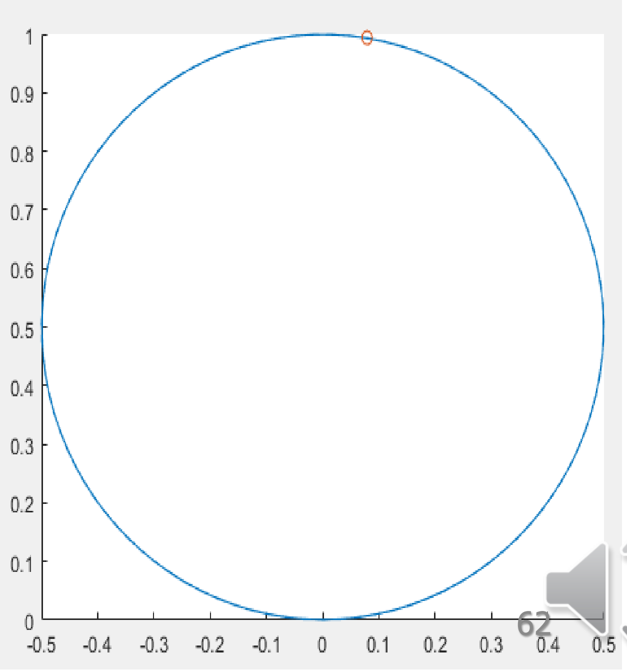


$$x^2+y^2 = y$$

$$\Rightarrow x^2+(y^2 - y + 0.5^2) = 0.5^2$$

$$\Rightarrow x^2+(y-0.5)^2 = 0.5^2$$

Note: we only show that the complete curve is a circle! If the range of the angle does not not fully cover for the curve, we only have an arc of the circle.



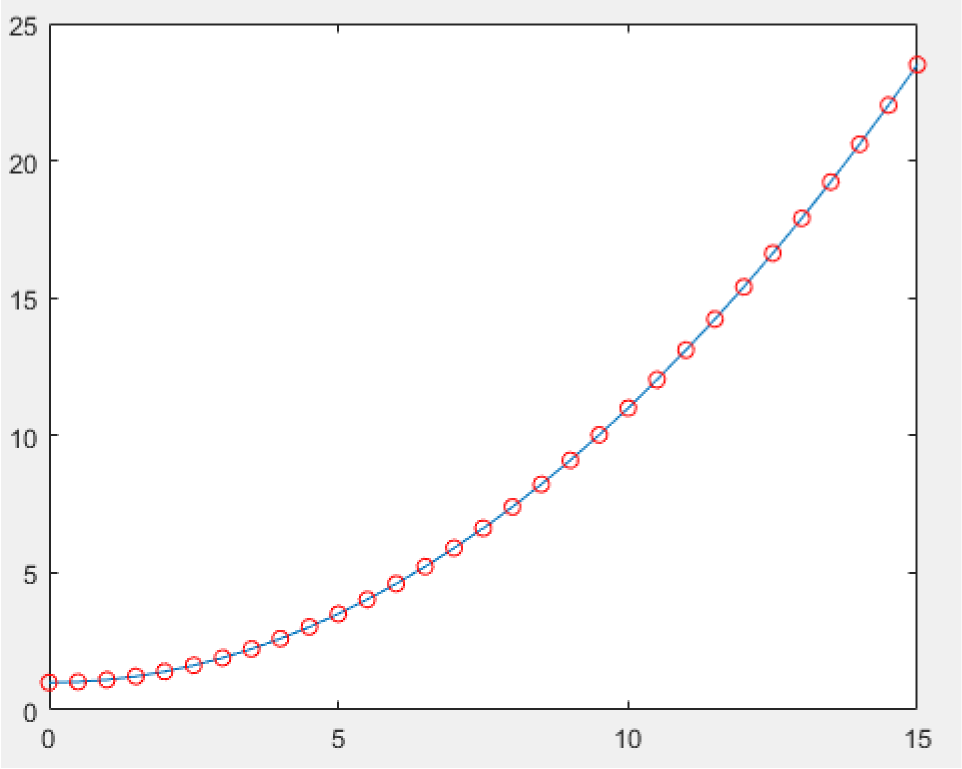
# Drawing a region

# Exercise

Plot a curve for x between [0, 15] similar to the one shown below.

The sampling interval is 0.5.

$y = x^2 /10 + 1$

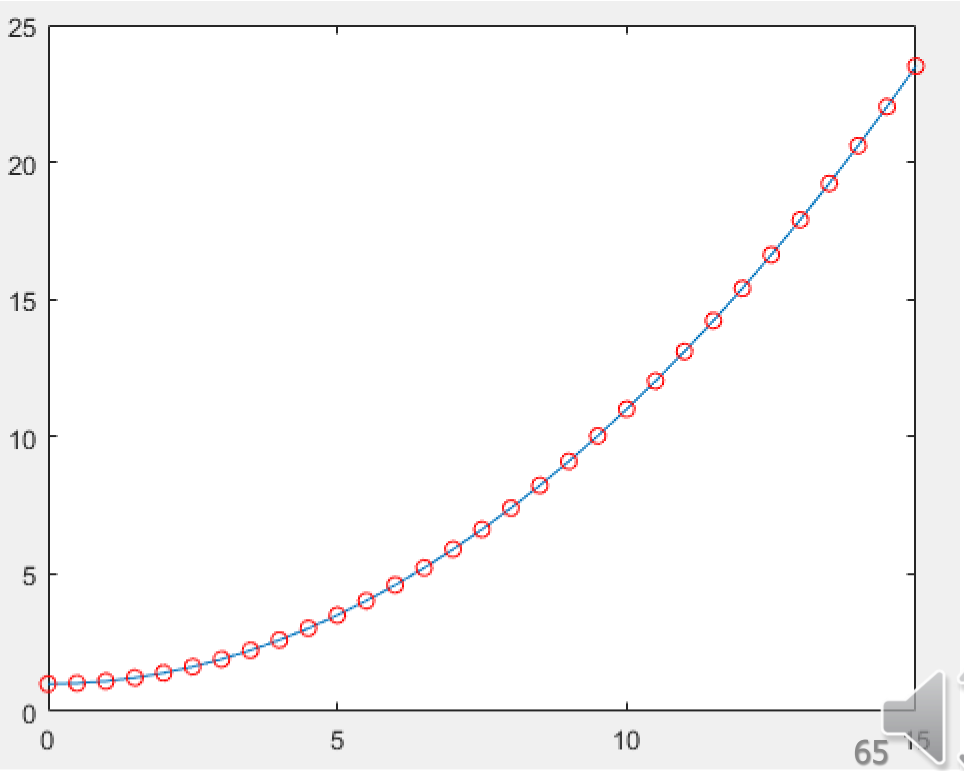


# Exercise

Plot a curve for x between [0, 15] similar to the one shown below.

The sampling interval is 0.5.

$y = x^2 /10 + 1$



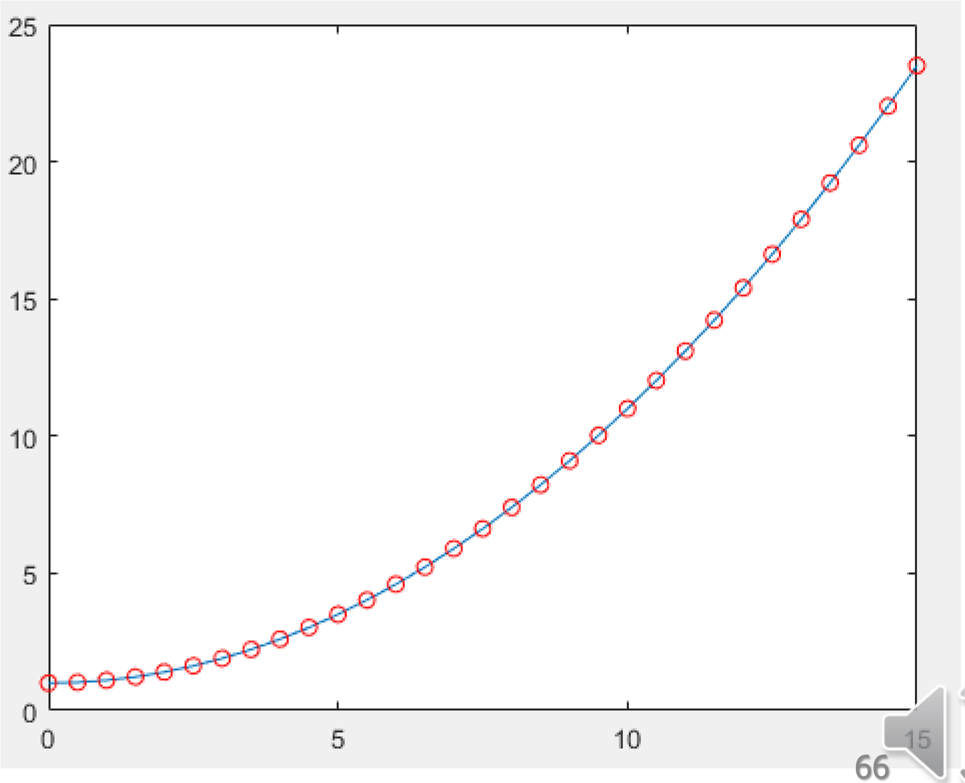
# Exercise

Plot a curve for x between [0, 15].

The sampling interval is 0.5.

$y = x^2 / 10 + 1$

```
figure
x = 0:0.5:15
y = x.^2./10+1;
plot(x,y);
hold on

plot(x,y,'ro');
hold on
```



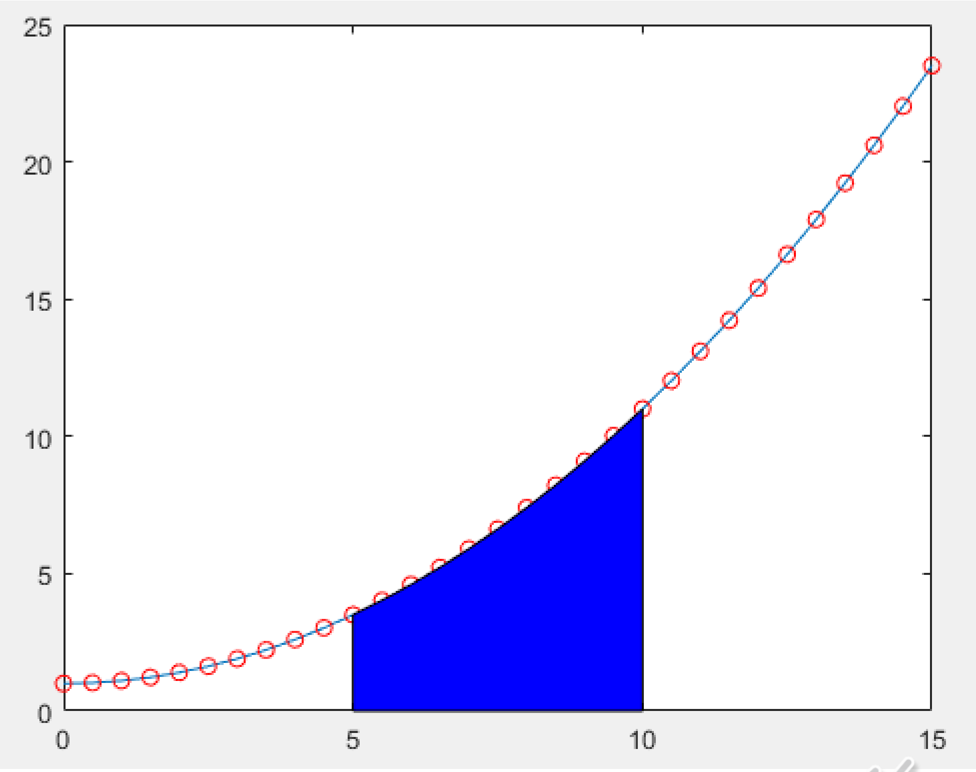
# fill

Plot a curve for x between [0, 15].

The sampling interval is 0.5.

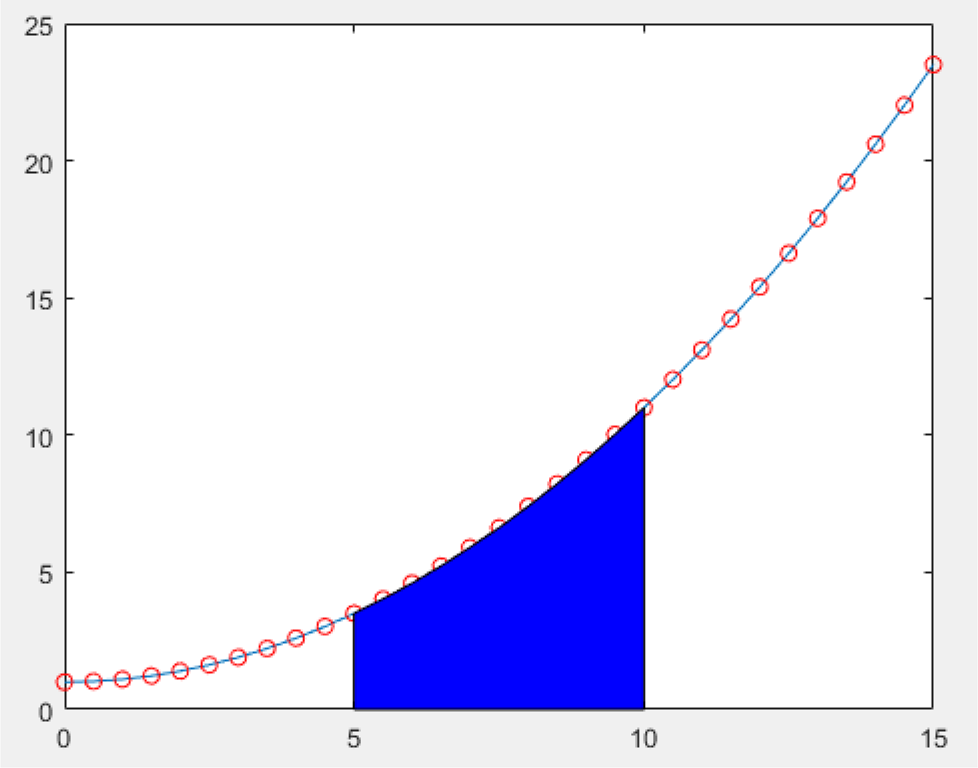
$y = x^2 /10 + 1$

Fill the region bounded by the x-axis and the curve inside the interval [5, 10]?



# fill

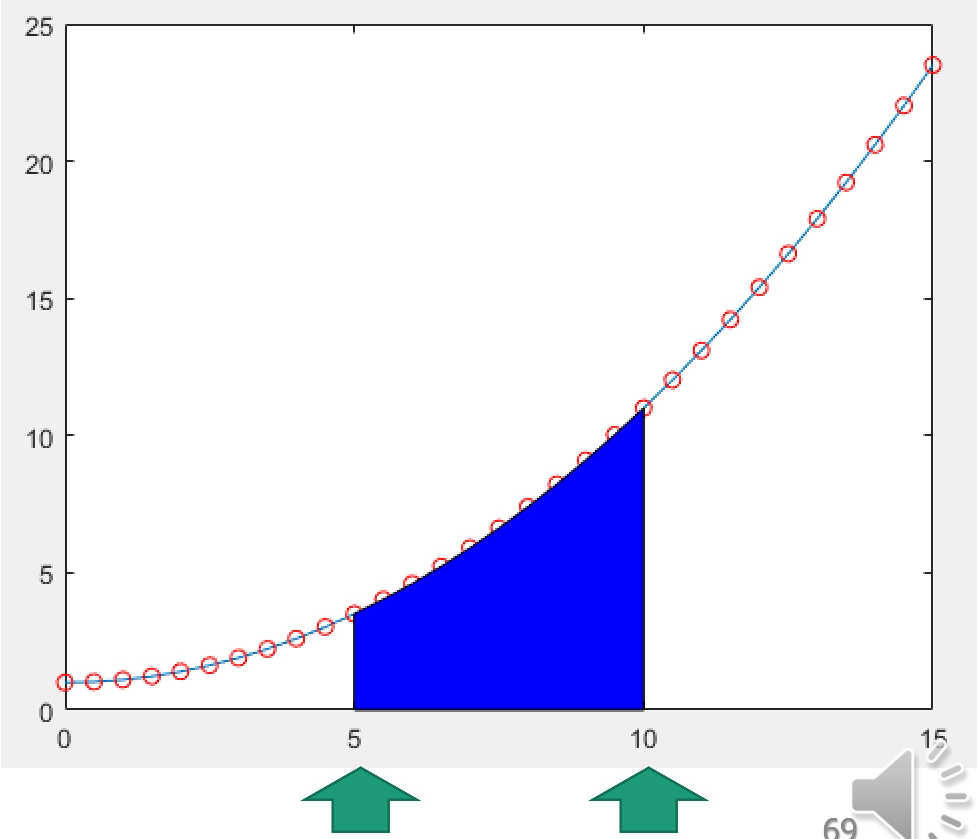
fill(X,Y,C) creates filled polygons from the data in X and Y with vertex color specified by C.



# fill

fill(X,Y,C) creates filled polygons from the data in X and Y with vertex color specified by C.

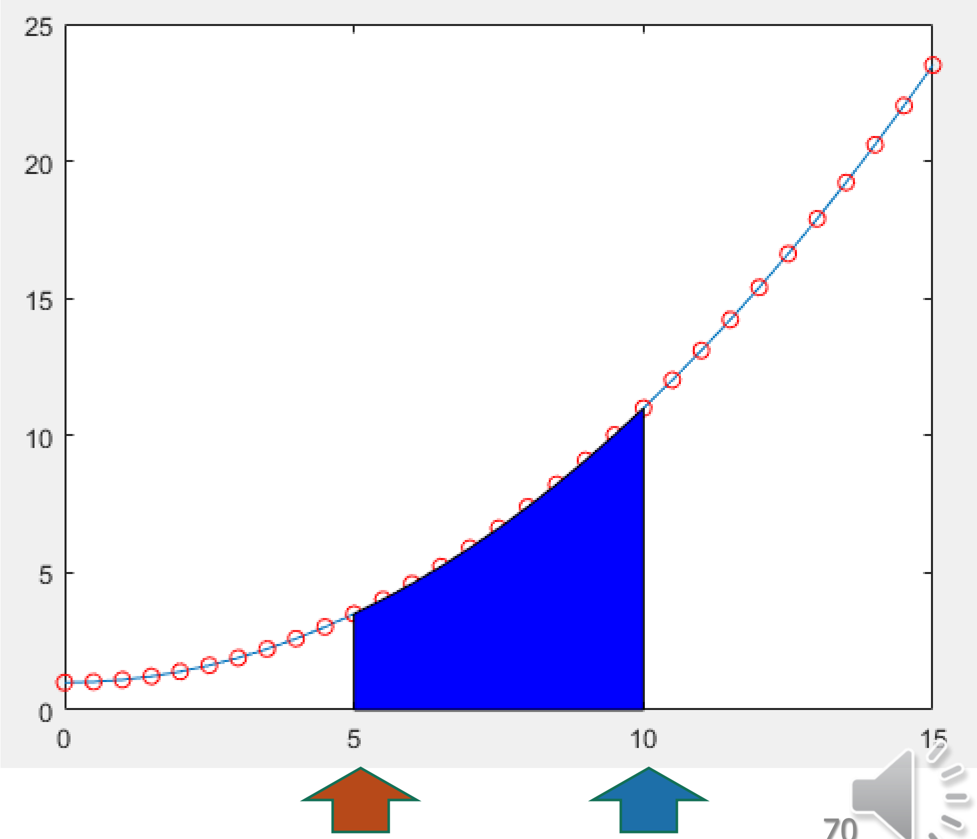
```
figure
x = 0:0.5:15
y = x.^2/10+1;
plot(x,y,x,y,'ro');
hold on

x = 5:0.5:10
y = x.^2/10+1;
x = [x,10];
y = [y,0];
x = [x,5];
y = [y,0];
fill(x,y,'b');
```



# fill

fill(X,Y,C) creates filled polygons from the data in X and Y with vertex color specified by C.

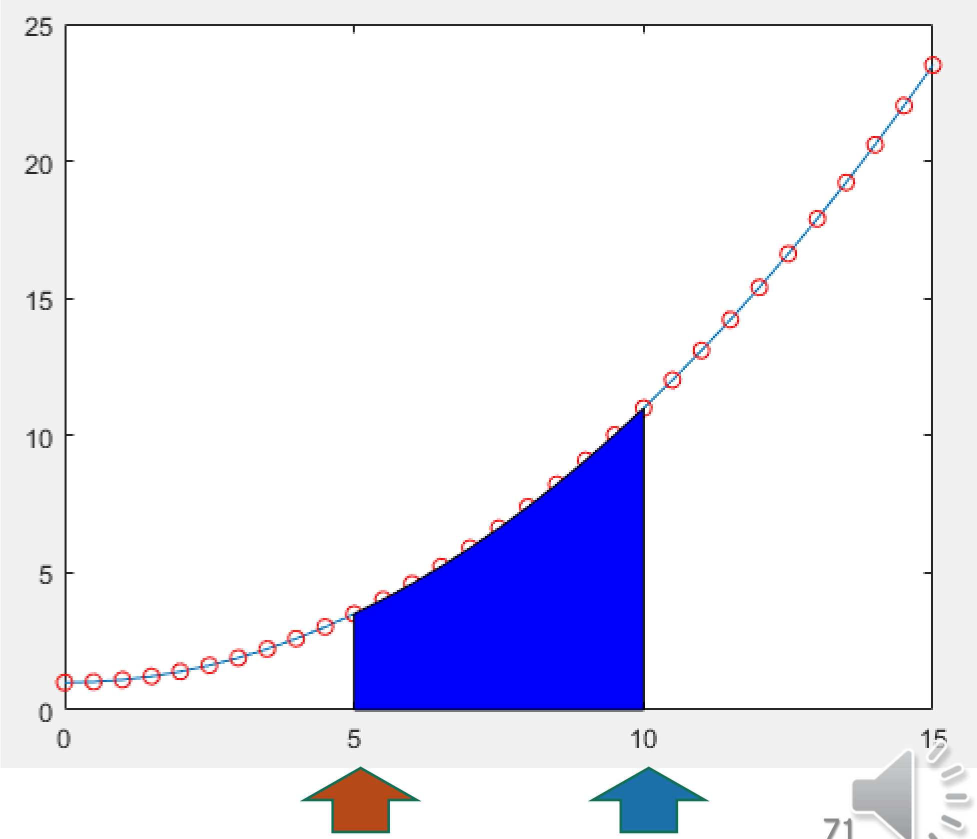
```
figure
x = 0:0.5:15
y = x.^2/10+1;
plot(x,y,x,y,'ro');
hold on

x = 5:0.5:10
y = x.^2/10+1;
x = [x,10];
y = [y,0];
x = [x,5];
y = [y,0];
fill(x,y,'b');
```



# fill

fill(X,Y,C) creates filled polygons from the data in X and Y with vertex color specified by C.

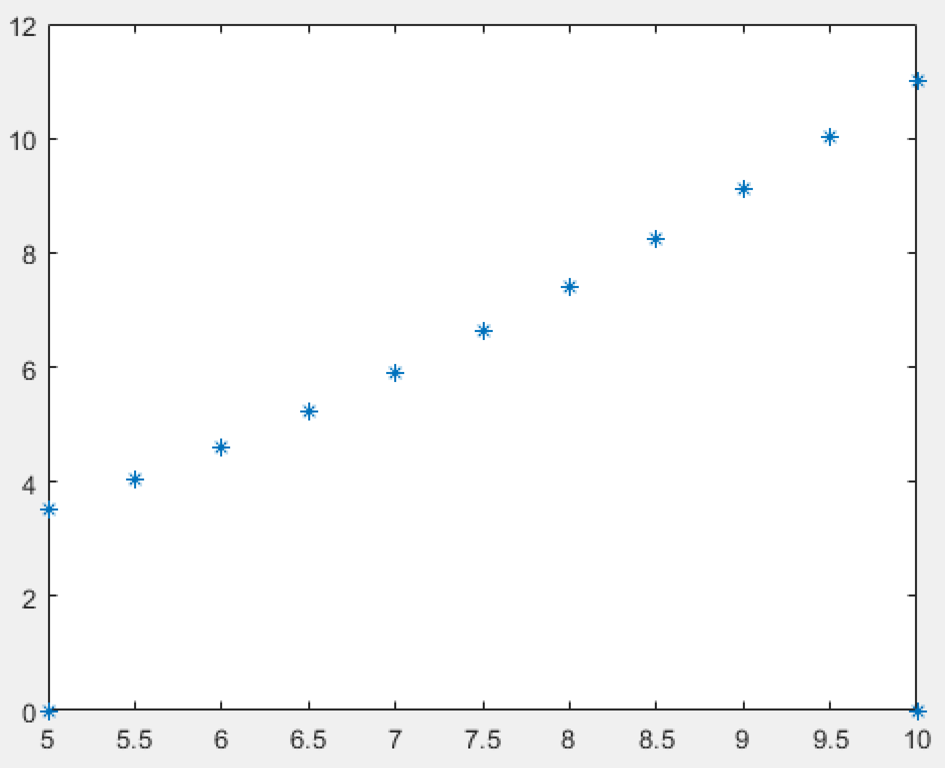
```
figure
x = 0:0.5:15
y = x.^2/10+1;
plot(x,y,x,y,'ro');
hold on

x = 5:0.5:10
y = x.^2/10+1;
x = [x,10];
y = [y,0];
x = [x,5];
y = [y,0];
fill(x,y,'b');
```



# fill

fill(X,Y,C) creates filled polygons from the data in X and Y with vertex color specified by C.

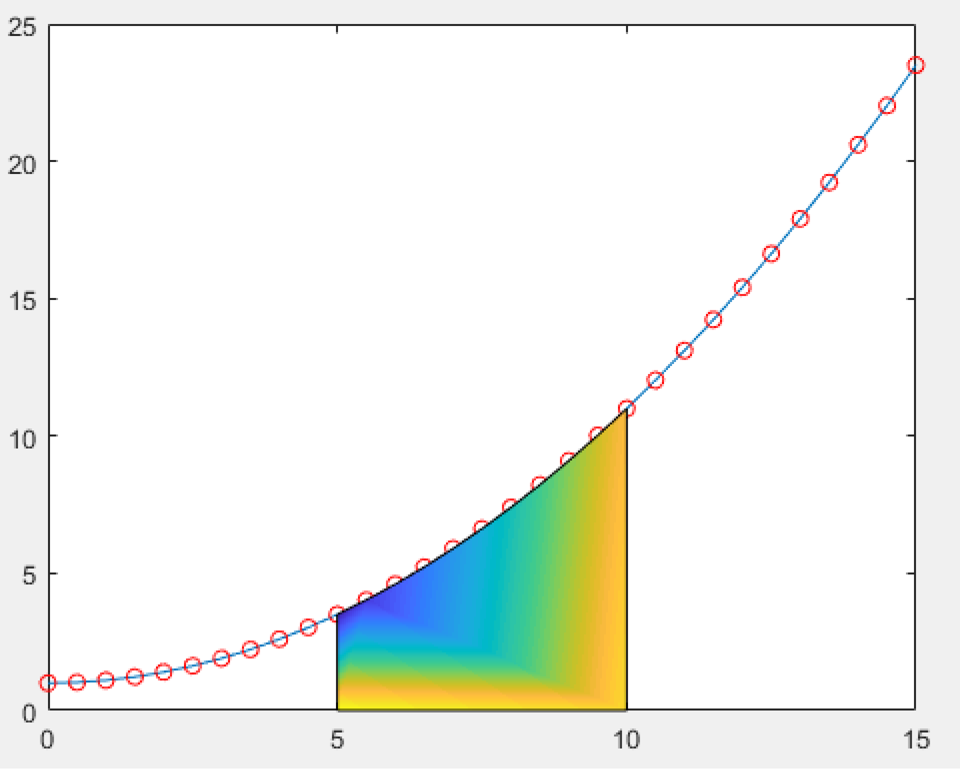
```
x = 5:0.5:10
y = x.^2/10+1;
x = [x,10];
y = [y,0];
x = [x,5];
y = [y,0];
fill(x,y,'*');
```



# fill(X,Y,C)

C is a vector or matrix used as an index into the colormap.

```
figure
x = 0:0.5:15
y = x.^2/10+1;
plot(x,y,x,y,'ro');
hold on
x = 5:0.5:10
y = x.^2/10+1;
x = [x,10];
y = [y,0];
x = [x,5];
y = [y,0];
C = 1:13;        %color index
fill(x,y,C);
```

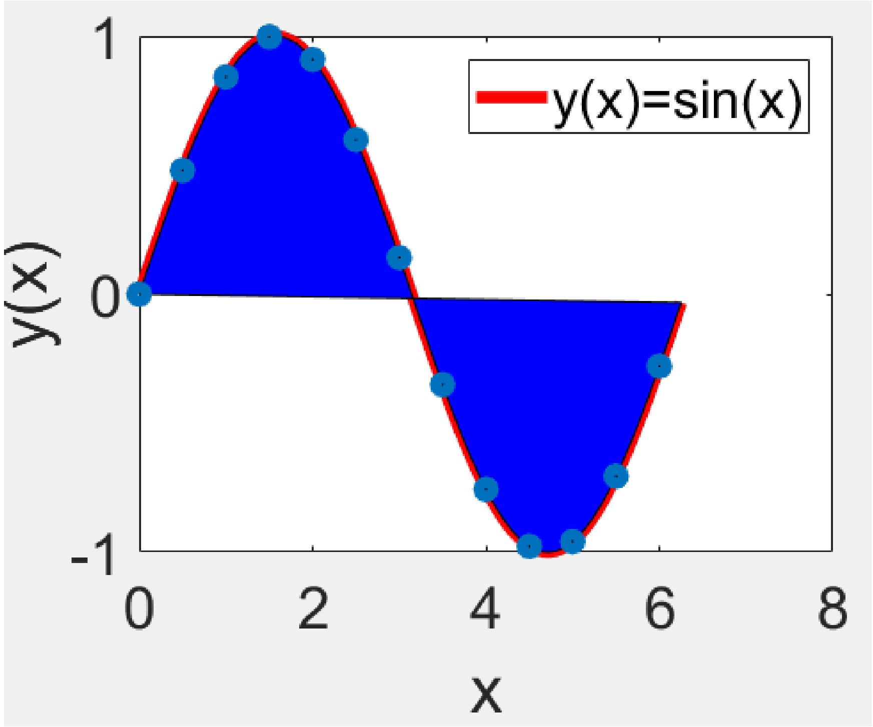


# Exercise

Let y(x) = sin(x). Write a program to fill a region with blue color and to draw the curve on the same figure. The region is bounded by y(x), the axis, and the x-interval [0, 2π]. Beautify the figure with legend(s), labels, proper fontsize, etc. Also, draw some uniform samples of the curve in circles.

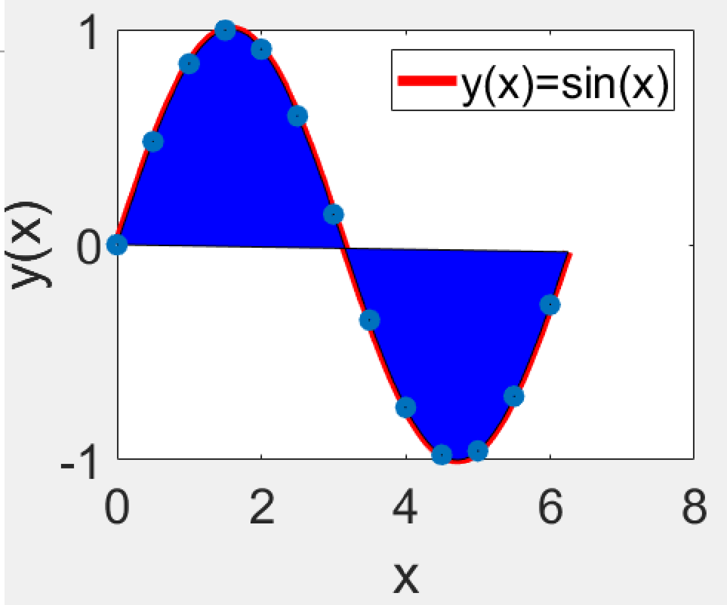
# Exercise (1 min)

Let y(x) = sin(x). Write a program to fill a region with blue color and to draw the curve on the same figure. The region is bounded by y(x), the axis, and the x-interval [0, 2π]. Beautify the figure with legend(s), labels, proper fontsize, etc. Also, draw some uniform samples of the curve in circles.



# fill

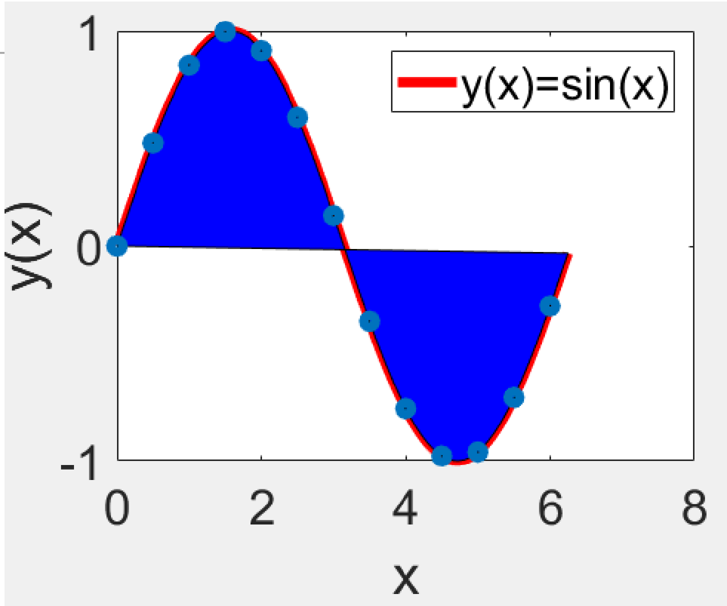
```
x = 0:0.05:(2*pi)
y = sin(x);
plot(x,y,'r', ...
    'linewidth', 5);
hold on
fill(x,y,'b');
x = 0:0.5:(2*pi)
y = sin(x);
plot(x,y,'o', ...
    'linewidth', 5);
set(gca, 'fontsize', 25);
legend('y(x)=sin(x)');
xlabel('x');
ylabel('y(x)');
```



No extra points are required.

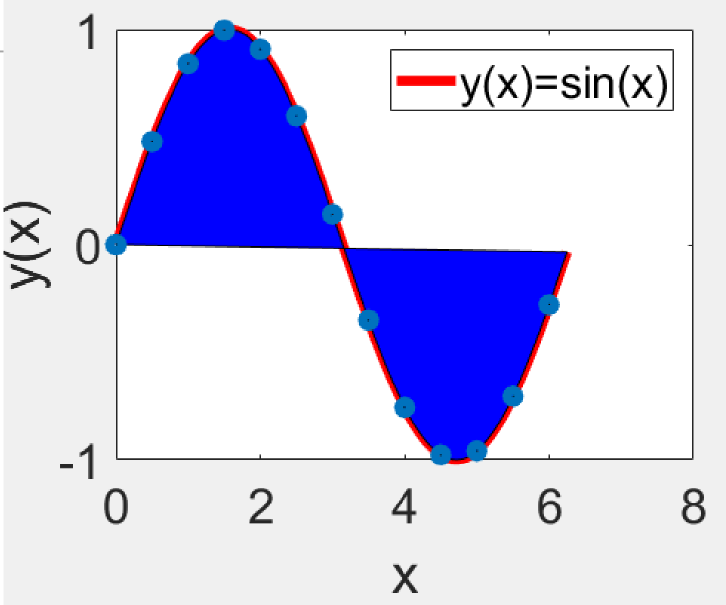
# fill

```
x = 0:0.05:(2*pi)
y = sin(x);
plot(x,y,'r', ...
    'linewidth', 5);
hold on
fill(x,y,'b');
x = 0:0.5:(2*pi)
y = sin(x);
plot(x,y,'o', ...
    'linewidth', 5);
set(gca, 'fontsize', 25);
legend('y(x)=sin(x)');
xlabel('x');
ylabel('y(x)');
```



# fill

```
x = 0:0.05:(2*pi)
y = sin(x);
plot(x,y,'r', ...
    'linewidth', 5);
hold on
fill(x,y,'b');
x = 0:0.5:(2*pi)
y = sin(x);
plot(x,y,'o', ...
    'linewidth', 5);
set(gca, 'fontsize', 25);
legend('y(x)=sin(x)');
xlabel('x');
ylabel('y(x)');
```



# Exercise

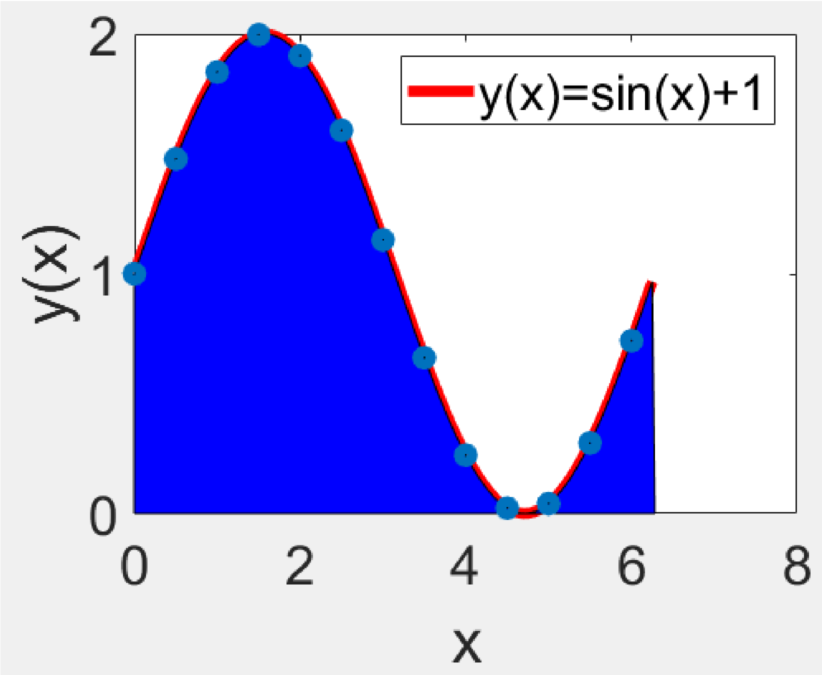
Let y(x) = sin(x) + 1. Write a program to fill a region with blue color and to draw the curve on the same figure. The region is bounded by y(x), the axis, and the x-interval [0, 2π]. Beautify the figure with legend(s), labels, proper fontsize, etc. Also, draw some uniform samples of the curve in circles.

**What do you expect to see?**
**Make a guess!**

# Exercise ( 1 min)

Let y(x) = sin(x) + 1. Write a program to fill a region with blue color and to draw the curve on the same figure. The region is bounded by y(x), the axis, and the x-interval [0, 2π]. Beautify the figure with legend(s), labels, proper fontsize, etc. Also, draw some uniform samples of the curve in circles.

Write down the structure plan. Sketch the program.

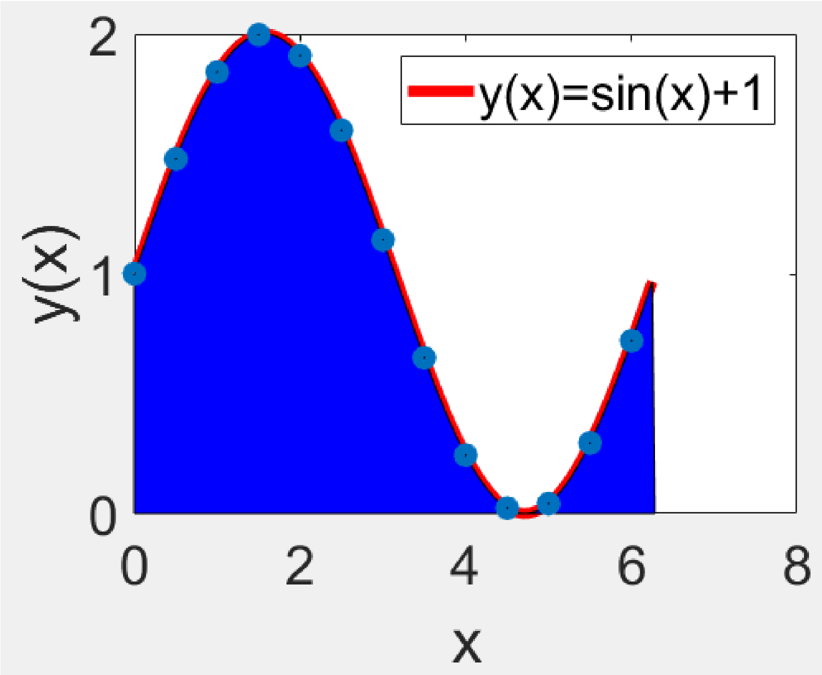


# Exercise ( 1 min)

Let y(x) = sin(x) + 1. Write a program to fill a region with blue color and to draw the curve on the same figure. The region is bounded by y(x), the axis, and the x-interval [0, 2π]. Beautify the figure with legend(s), labels, proper fontsize, etc. Also, draw some uniform samples of the curve in circles.

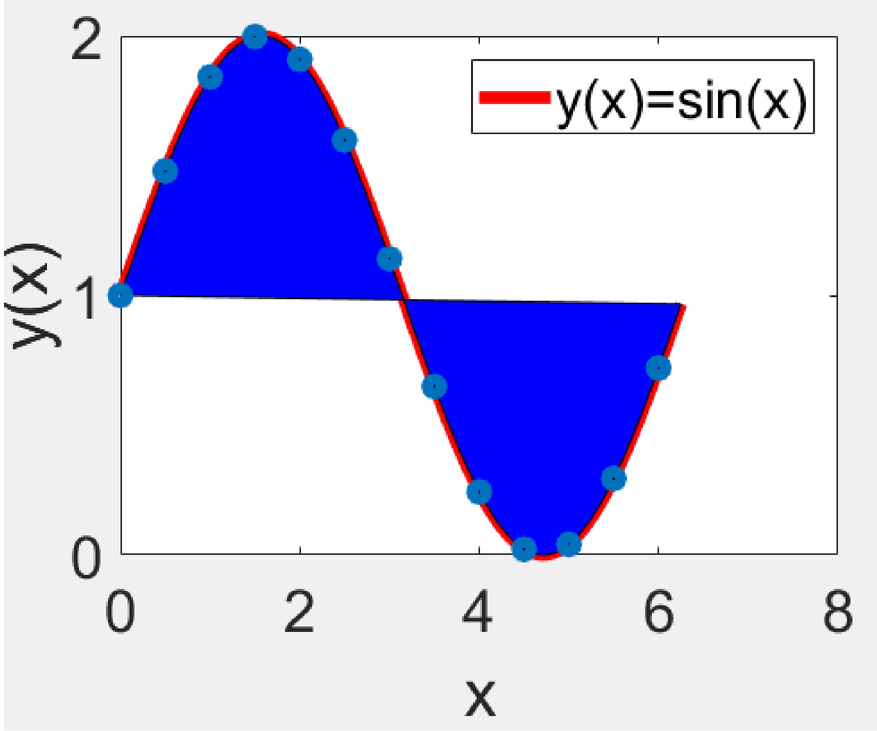
Draw the curve and collect sample points.
Compute the polygon.
Add extra points to bound the region when necessary.

Call *fill* to fill the region. …….



# Exercise. A wrong approach

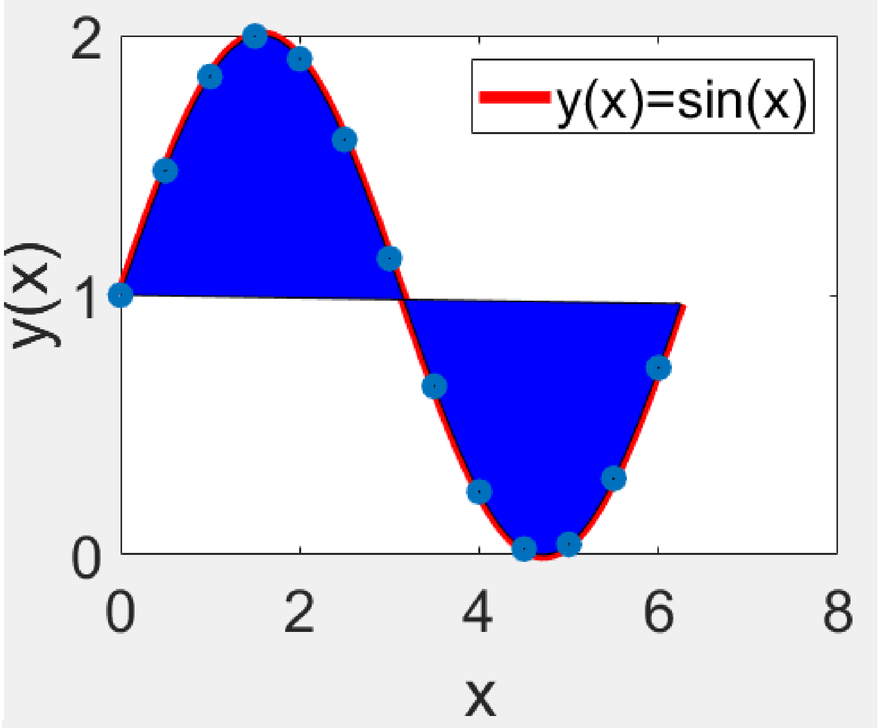
```
close all; clear;clf
x = 0:0.05:(2*pi)
y = sin(x) + 1;
plot(x,y,'r', 'linewidth', 5);
hold on

fill(x,y,'b');
x = 0:0.5:(2*pi)
y = sin(x) + 1;
plot(x,y,'o', 'linewidth', 5);
set(gca, 'fontsize', 25);
legend('y(x)=sin(x)');
xlabel('x');
ylabel('y(x)'); % wrong?!
```

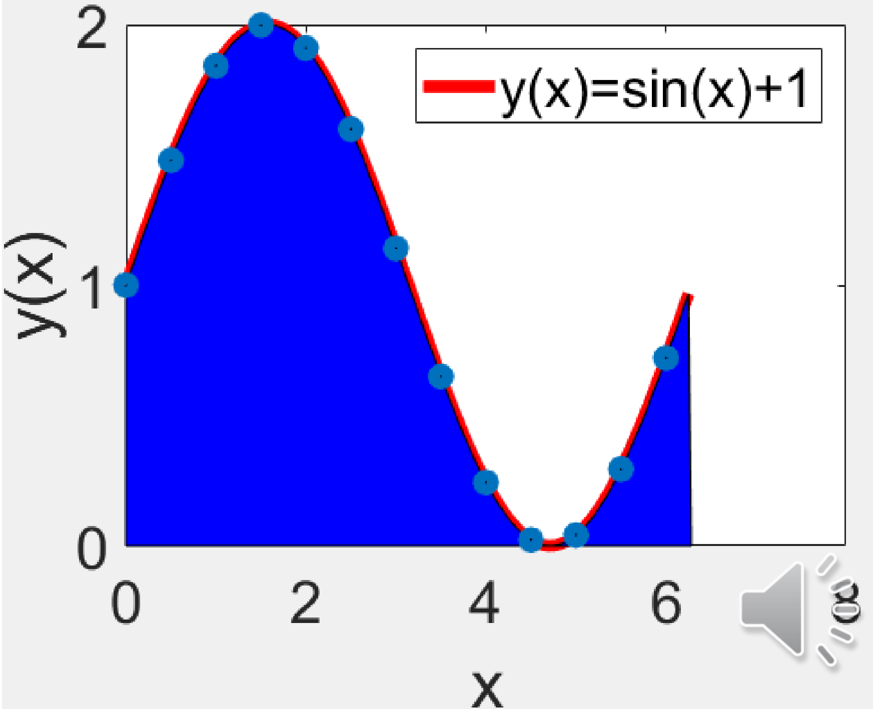


Do not obtain the desirable result.

# Exercise. A wrong approach

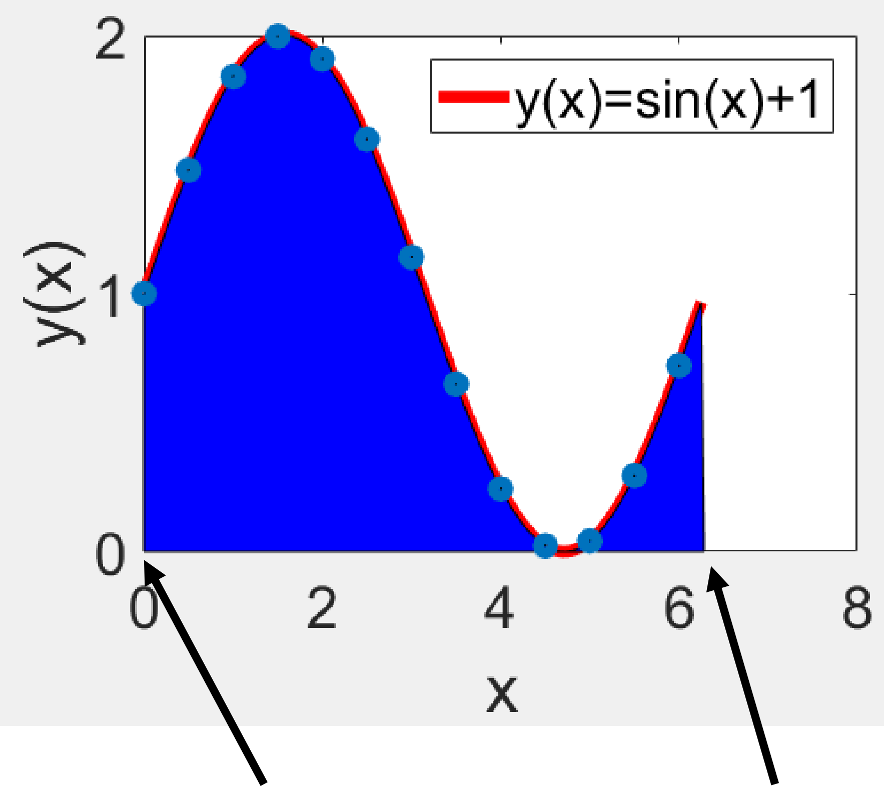
```
close all; clear;clf
x = 0:0.05:(2*pi)
y = sin(x) + 1;
plot(x,y,'r', 'linewidth', 5);
hold on

fill(x,y,'b');
x = 0:0.5:(2*pi)
y = sin(x) + 1;
plot(x,y,'o', 'linewidth', 5);
set(gca, 'fontsize', 25);
legend('y(x)=sin(x)');
xlabel('x');
ylabel('y(x)'); % wrong?!
```





# Exercise

```
close all; clear;clf
x = 0:0.05:(2*pi)
y = sin(x) + 1;
plot(x,y,'r', 'linewidth', 5);
hold on
x = [x 2*pi 0]
y = [y 0    0]
fill(x,y,'b');
x = 0:0.5:(2*pi)
y = sin(x) + 1;
plot(x,y,'o', 'linewidth', 5);
set(gca, 'fontsize', 25);
legend('y(x)=sin(x)');
xlabel('x');
ylabel('y(x)');
```



(0, sin(0)+1)

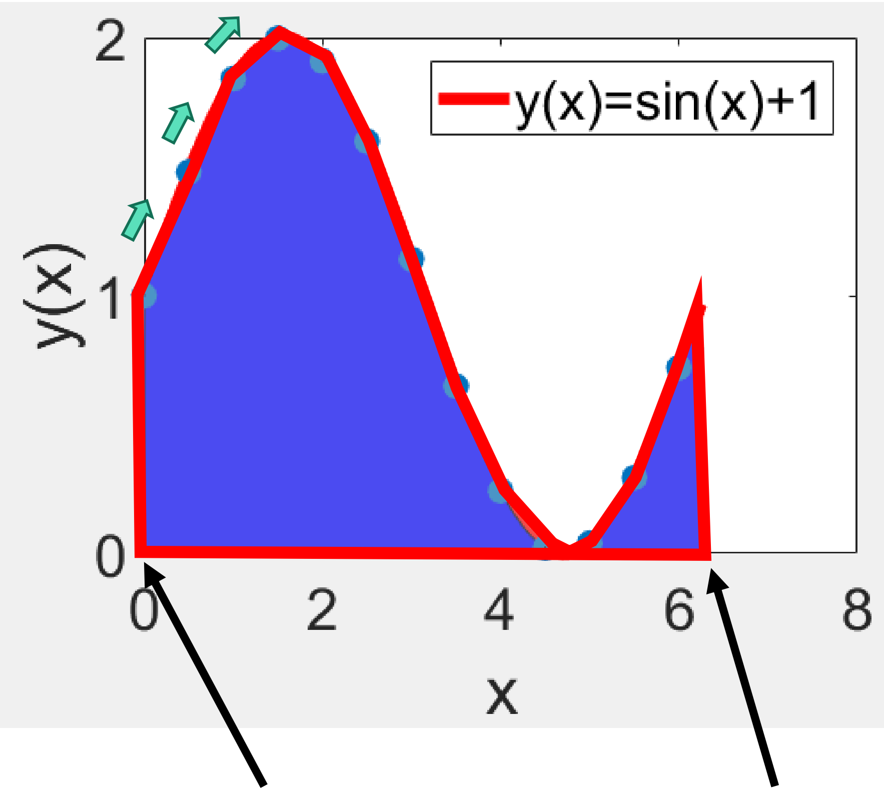
projected
(0, 0)

(2π, sin(2π)+1)

projected
(2π, 0)

# Exercise

```
close all; clear;clf
x = 0:0.05:(2*pi)
y = sin(x) + 1;
plot(x,y,'r', 'linewidth', 5);
hold on
x = [x 2*pi 0] % organized
y = [y 0     0] % in clockwise
fill(x,y,'b');
x = 0:0.5:(2*pi)
y = sin(x) + 1;
plot(x,y,'o', 'linewidth', 5);
set(gca, 'fontsize', 25);
legend('y(x)=sin(x)');
xlabel('x');
ylabel('y(x)');
```



(0, sin(0)+1)

(2π, sin(2π)+1)

# Exercise

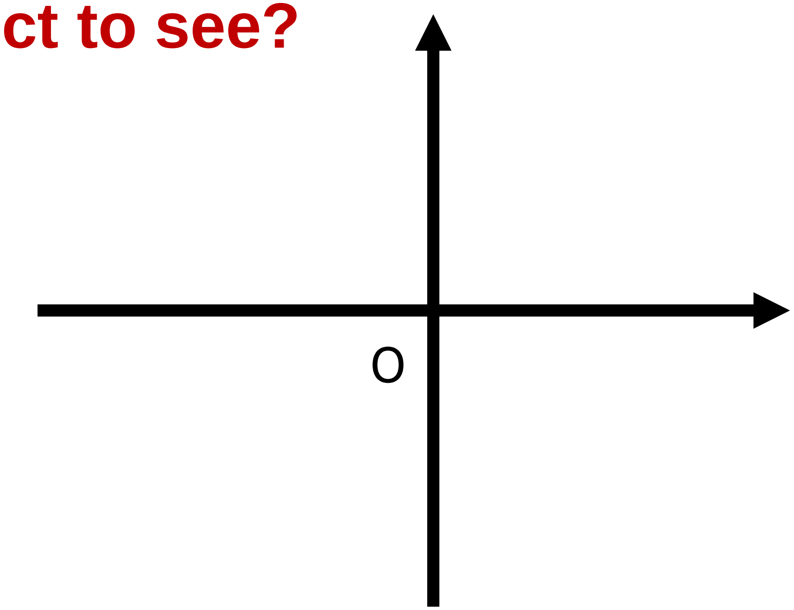
Let y(x) = 2cos(x) + 1. Write a program to fill a region with blue color and to draw the curve on the same figure. The region is bounded by y(x), the axis, and the x-interval [π/4, 7π/4]. Beautify the figure with legend(s), labels, proper fontsize, etc. Also, draw some uniform samples of the curve in circles.

**What do you expect to see?**
**Make a guess!**

# Exercise

Let y(x) = 2cos(x) + 1. Write a program to fill a region with blue color and to draw the curve on the same figure. The region is bounded by y(x), the axis, and the x-interval [π/4, 7π/4]. Beautify the figure with legend(s), labels, proper fontsize, etc. Also, draw some uniform samples of the curve in circles.

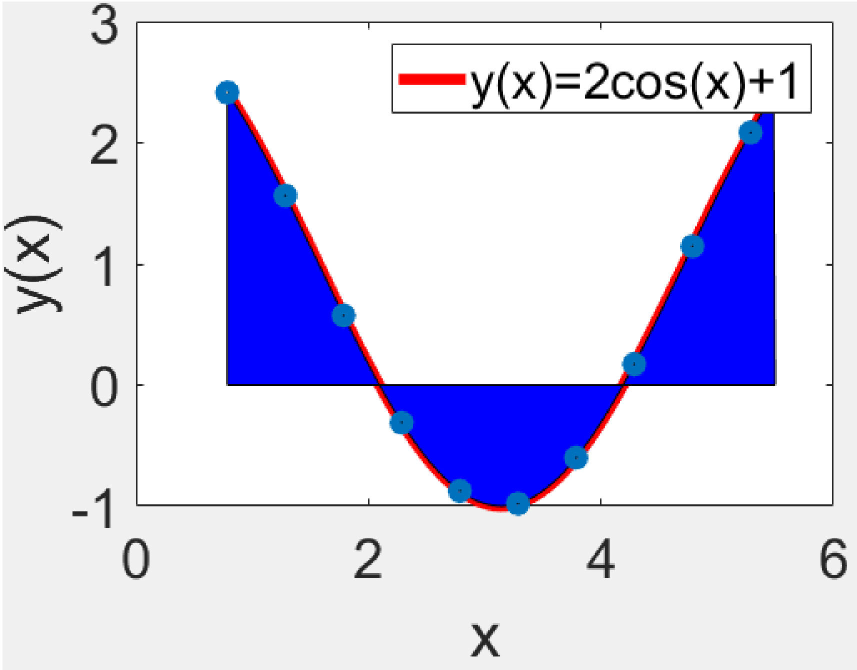
**What do you expect to see? Make a guess!**



# Exercise

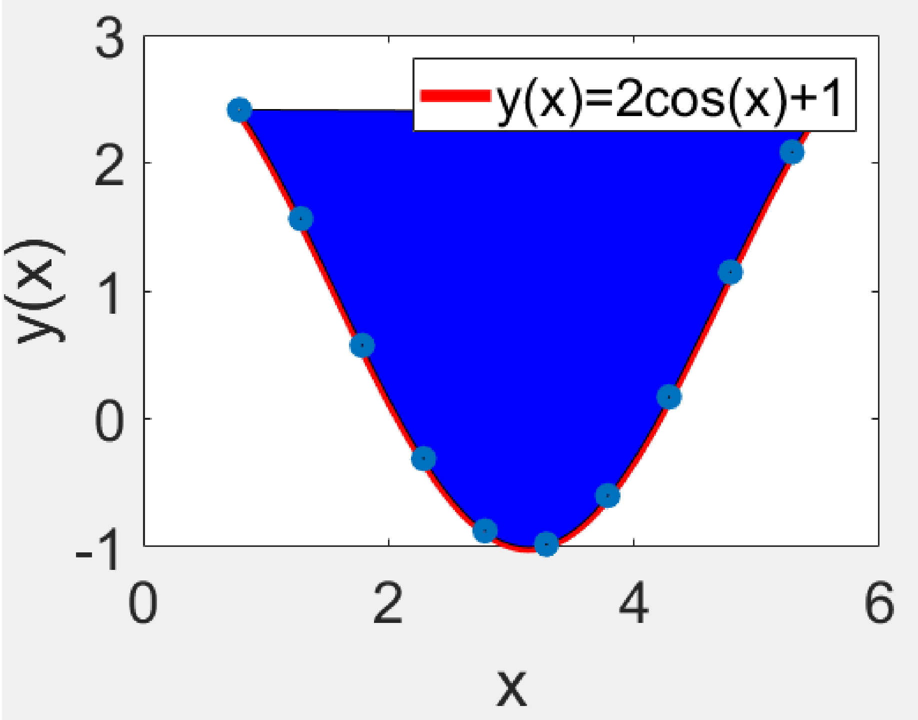
Let y(x) = 2cos(x) + 1. Write a program to fill a region with blue color and to draw the curve on the same figure. The region is bounded by y(x), the axis, and the x-interval [π/4, 7π/4]. Beautify the figure with legend(s), labels, proper fontsize, etc. Also, draw some uniform samples of the curve in circles.

**What do you expect to see?**
**Make a guess!**



# Exercise. A wrong approach

```
close all; clear;clf
x0 = pi/4;
x1 = 7*pi/4;
x = x0:0.05:x1
y = 2*cos(x) + 1;
plot(x,y,'r', 'linewidth', 5);
hold on
y0 = y(1);
y1 = y(end);
fill(x,y,'b');
x = x0:0.5:x1
y = 2*cos(x) + 1;
plot(x,y,'o', 'linewidth', 5);
set(gca, 'fontsize', 25);
legend('y(x)=2cos(x)+1');
xlabel('x');
ylabel('y(x)');
```

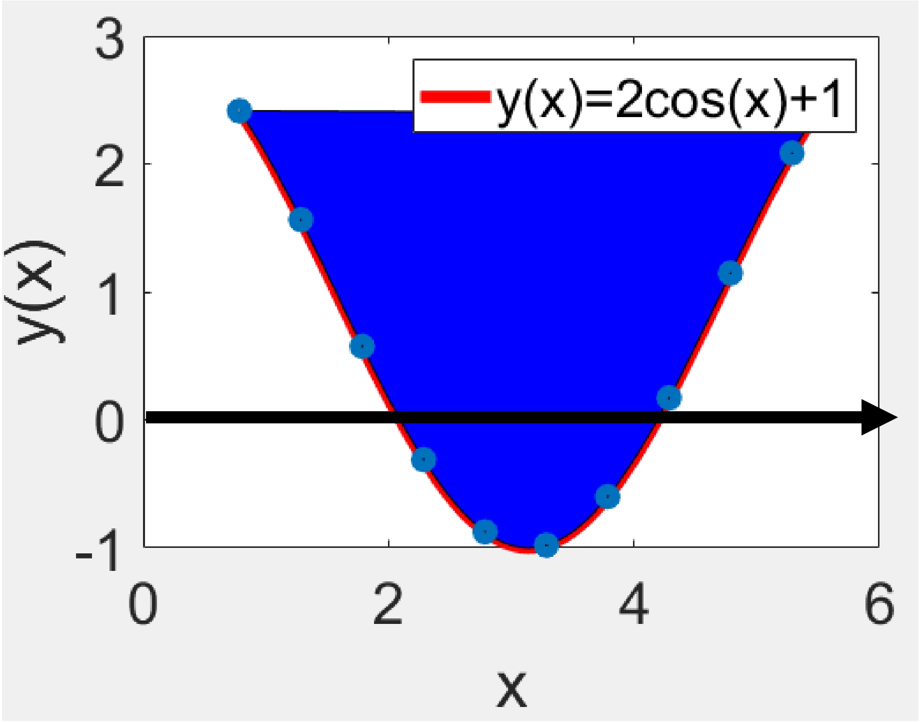


Note: the legend overlaps with the region and the curve.

The filled region is not right too.

# Exercise. A wrong approach

```
close all; clear;clf
x0 = pi/4;
x1 = 7*pi/4;
x = x0:0.05:x1
y = 2*cos(x) + 1;
plot(x,y,'r', 'linewidth', 5);
hold on
y0 = y(1);
y1 = y(end);
fill(x,y,'b');
x = x0:0.5:x1
y = 2*cos(x) + 1;
plot(x,y,'o', 'linewidth', 5);
set(gca, 'fontsize', 25);
legend('y(x)=2cos(x)+1');
xlabel('x');
ylabel('y(x)');
```

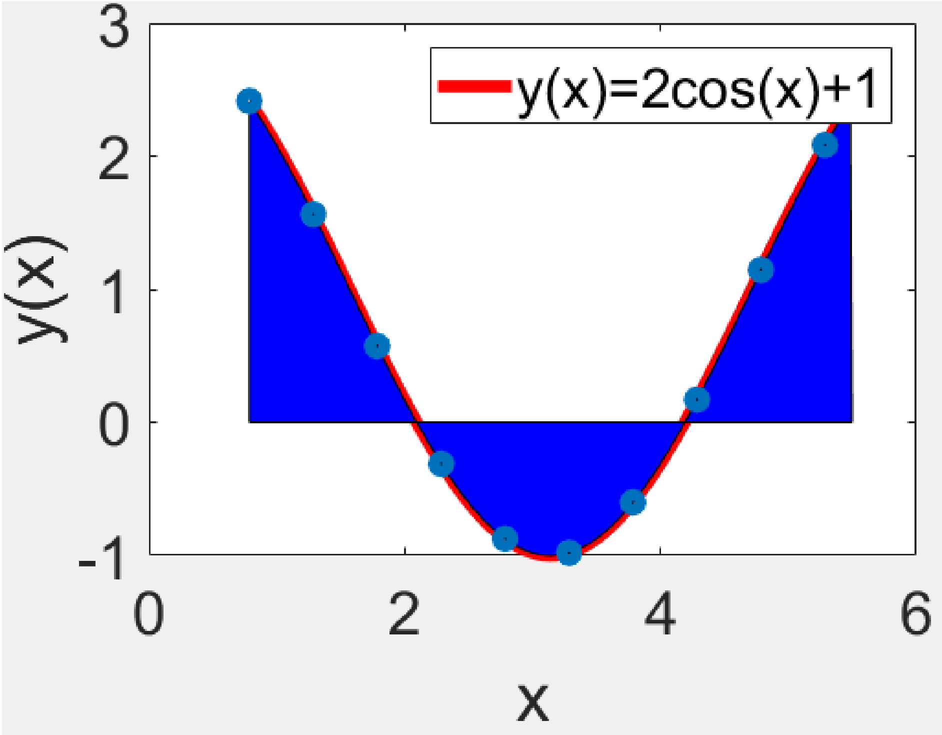


Note: the legend overlaps with the region and the curve.

The filled region is not right too.

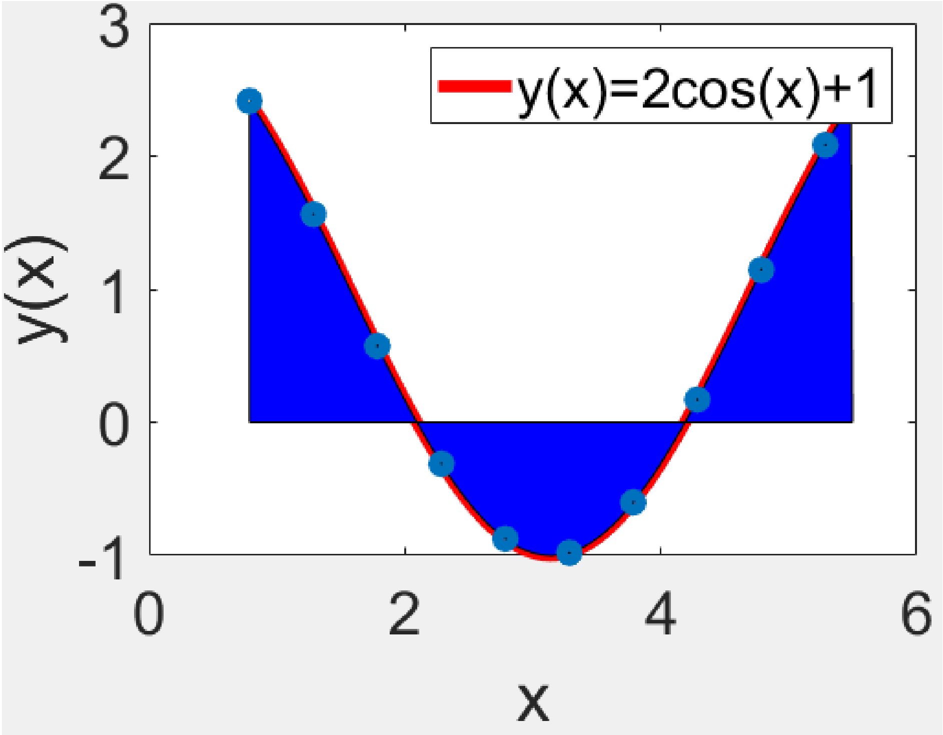
# Exercise. An ok approach

```
close all; clear;clf
x0 = pi/4;   //x-co start point
x1 = 7*pi/4; //x-co end point
x = x0:0.05:x1
y = 2*cos(x) + 1;
plot(x,y,'r', 'linewidth', 5);
hold on
y0 = y(1);
y1 = y(end);
x = [x x1 x0]
y = [y 0 0]
fill(x,y,'b');
x = x0:0.5:x1
y = 2*cos(x) + 1;
plot(x,y,'o', 'linewidth', 5);
set(gca, 'fontsize', 25);
legend('y(x)=2cos(x)+1');
xlabel('x');
ylabel('y(x)');
```



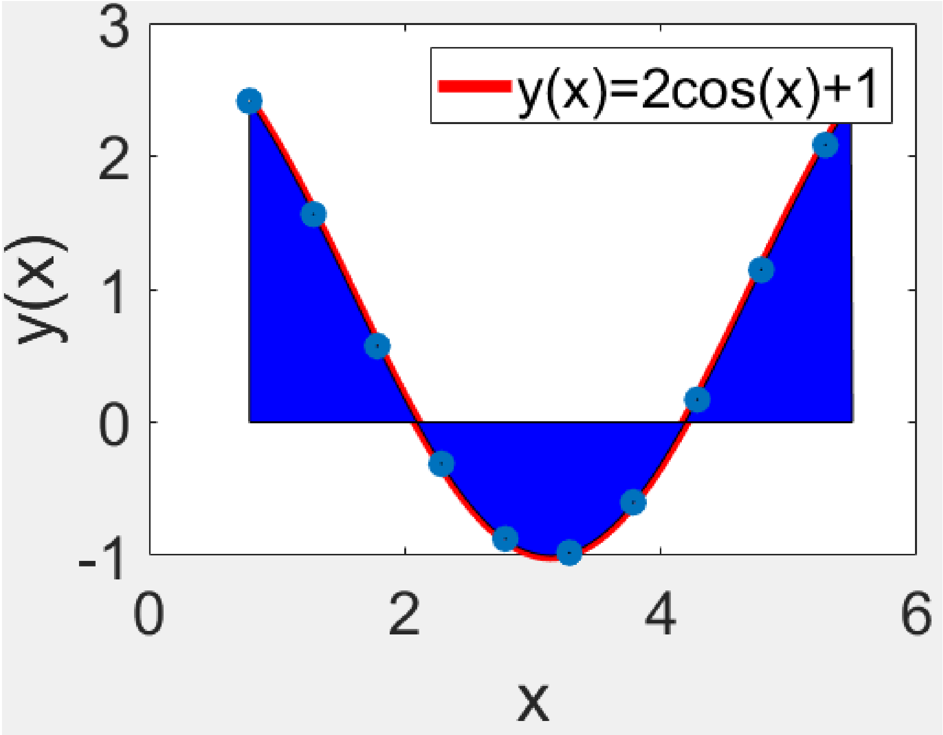
# Exercise. An ok approach

```
close all; clear;clf
x0 = pi/4;   //x-co start point
x1 = 7*pi/4; //x-co end point
x = x0:0.05:x1
y = 2*cos(x) + 1;
plot(x,y,'r', 'linewidth', 5);
hold on
y0 = y(1);
y1 = y(end);
x = [x x1 x0]
y = [y 0 0]
fill(x,y,'b');
x = x0:0.5:x1
y = 2*cos(x) + 1;
plot(x,y,'o', 'linewidth', 5);
set(gca, 'fontsize', 25);
legend('y(x)=2cos(x)+1');
xlabel('x');
ylabel('y(x)');
```



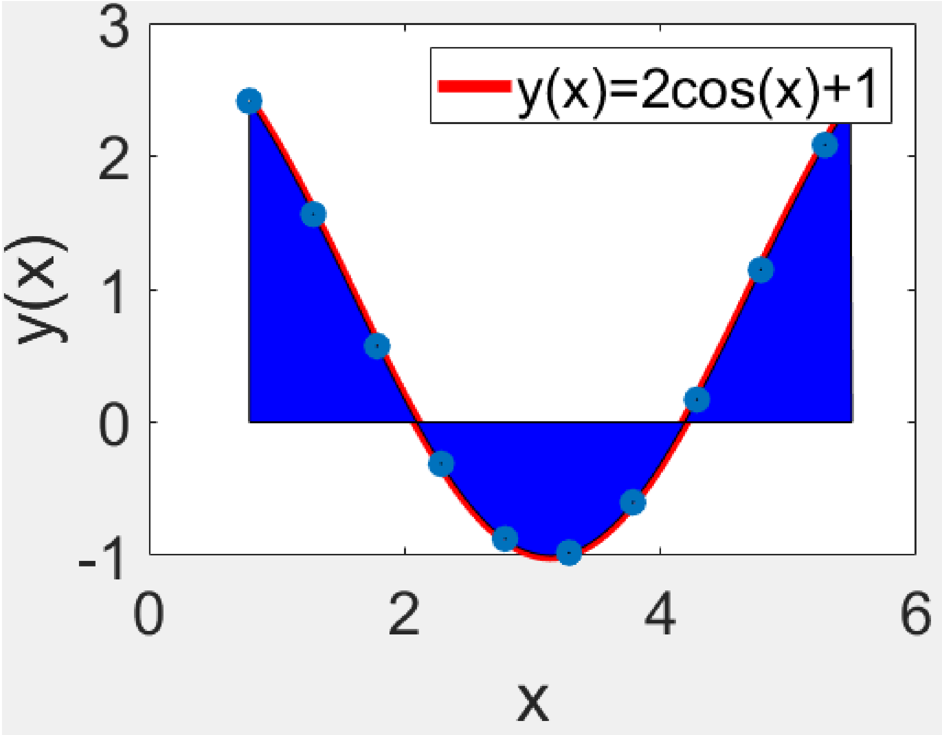
# Exercise. An ok approach

```
close all; clear;clf
x0 = pi/4;   //x-co start point
x1 = 7*pi/4; //x-co end point
x = x0:0.05:x1
y = 2*cos(x) + 1;
plot(x,y,'r', 'linewidth', 5);
hold on
y0 = y(1);
y1 = y(end);
x = [x x1 x0]
y = [y 0 0]
fill(x,y,'b');
x = x0:0.5:x1
y = 2*cos(x) + 1;
plot(x,y,'o', 'linewidth', 5);
set(gca, 'fontsize', 25);
legend('y(x)=2cos(x)+1');
xlabel('x');
ylabel('y(x)');
```



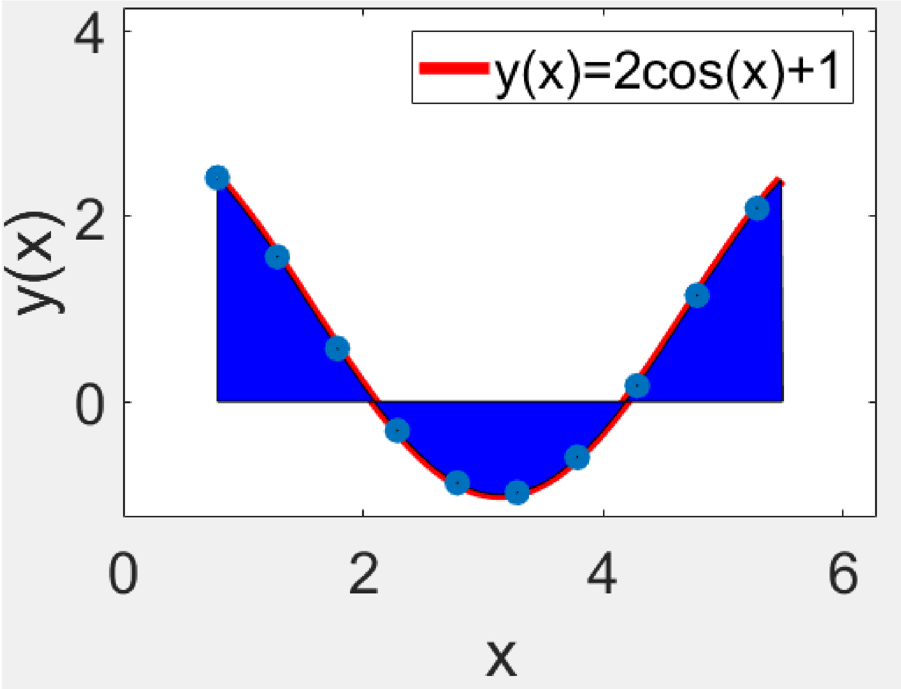
# Exercise. An ok approach

```
close all; clear;clf
x0 = pi/4;   //x-co start point
x1 = 7*pi/4; //x-co end point
x = x0:0.05:x1
y = 2*cos(x) + 1;
plot(x,y,'r', 'linewidth', 5);
hold on
y0 = y(1);
y1 = y(end);
x = [x x1 x0]
y = [y 0 0]
fill(x,y,'b');
x = x0:0.5:x1
y = 2*cos(x) + 1;
plot(x,y,'o', 'linewidth', 5);
set(gca, 'fontsize', 25);
legend('y(x)=2cos(x)+1');
xlabel('x');
ylabel('y(x)');
```



# Exercise. Good.

```
close all; clear;clf
x0 = pi/4;
x1 = 7*pi/4;
x = x0:0.05:x1
y = 2*cos(x) + 1;
plot(x,y,'r', 'linewidth', 5);
hold on
y0 = y(1);
y1 = y(end);
x = [x x1 x0]
y = [y 0 0]
fill(x,y,'b');
x = x0:0.5:x1
y = 2*cos(x) + 1;
plot(x,y,'o', 'linewidth', 5);
set(gca, 'fontsize', 25);
legend('y(x)=2cos(x)+1');
xlabel('x');
ylabel('y(x)');
axis( [0   2*pi   -1.25    4.25] );  % avoid overlapping
```



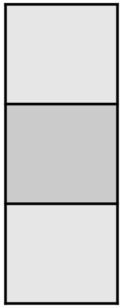
# fill (X, Y, C)

If C is a row vector, length(C) must equal size(X,2) and size(Y,2); %size(X,2) returns the number of columns of X

if C is a column vector, length(C) must equal size(X,1) and size(Y,1). %size(X,1) returns the number of rows of X

If necessary, fill closes the polygon by connecting the last vertex to the first.

The values in X and Y can be numeric, datetime, duration, or categorical values.



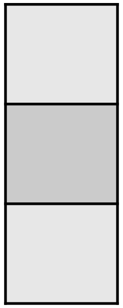
# fill (X, Y, C)

If C is a row vector, length(C) must equal size(X,2) and size(Y,2); %size(X,2) returns the number of columns of X

if C is a column vector, length(C) must equal size(X,1) and size(Y,1). %size(X,1) returns the number of rows of X

**If necessary, fill closes the polygon by connecting the last vertex to the first.**

The values in X and Y can be numeric, datetime, duration, or categorical values.



# Logarithmic plots

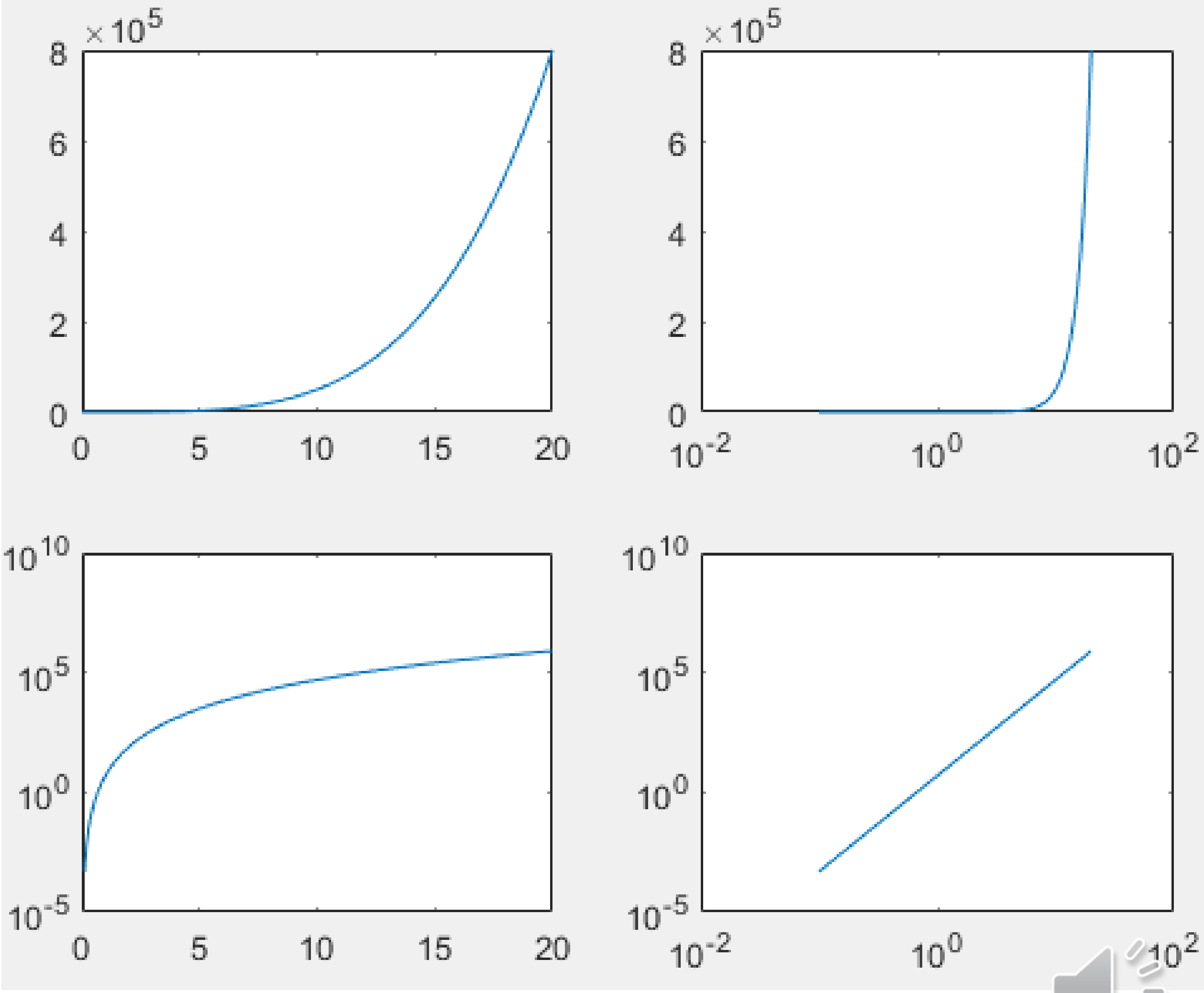
Functions:

- ➢ semilogx
- ➢ semilogy
- ➢ loglog

- Automatically replace linear scales with logarithmic scales.
- Useful for a variable ranging over many orders of magnitude.

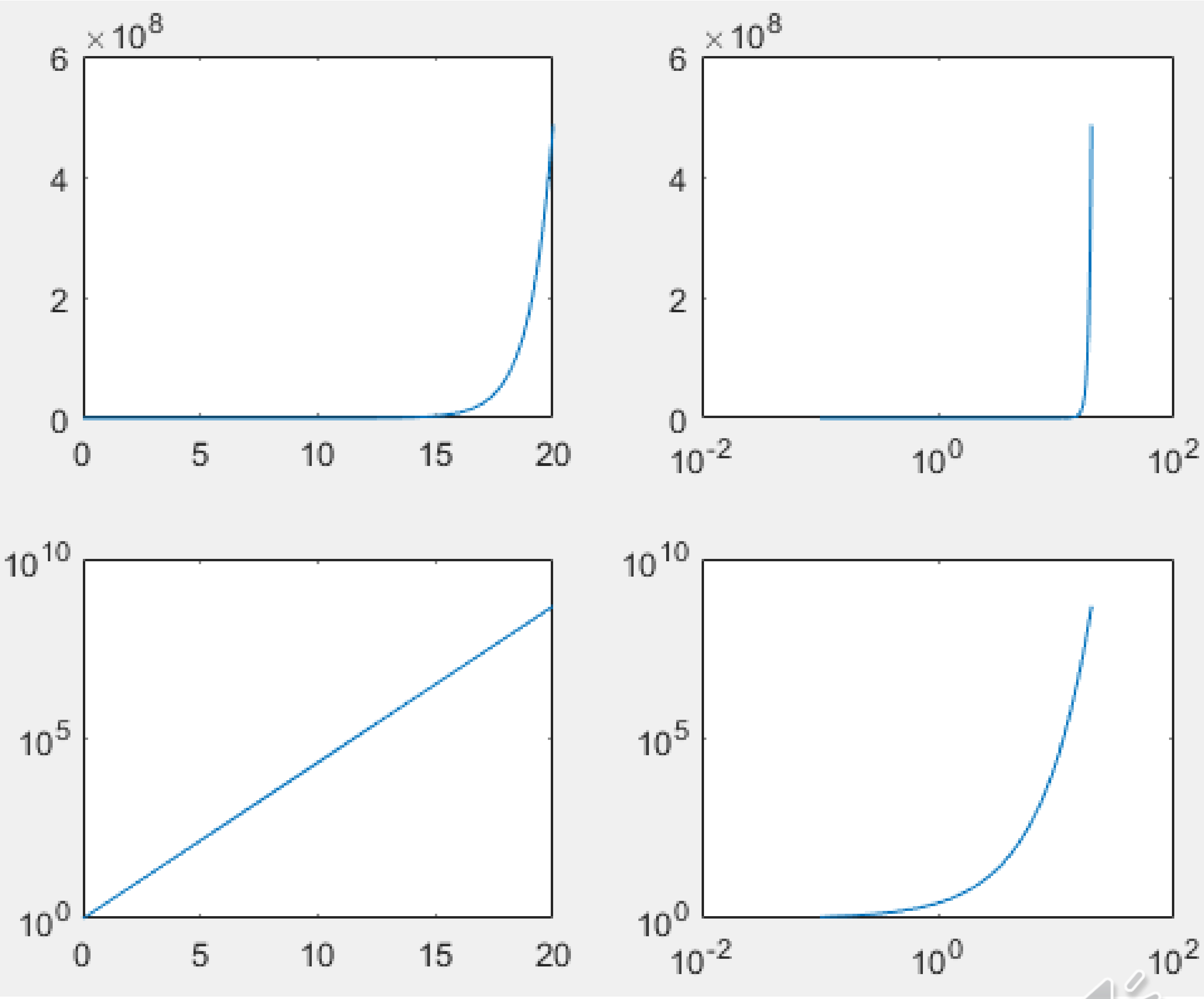
# Example
# Logarithmic plots and subplots

```
x = 0:0.1:20;
y = 5*x.^4;
subplot(2,2,1);
plot(x,y);
subplot(2,2,2);
semilogx(x,y);
subplot(2,2,3);
semilogy(x,y);
subplot(2,2,4);
loglog(x,y);
```



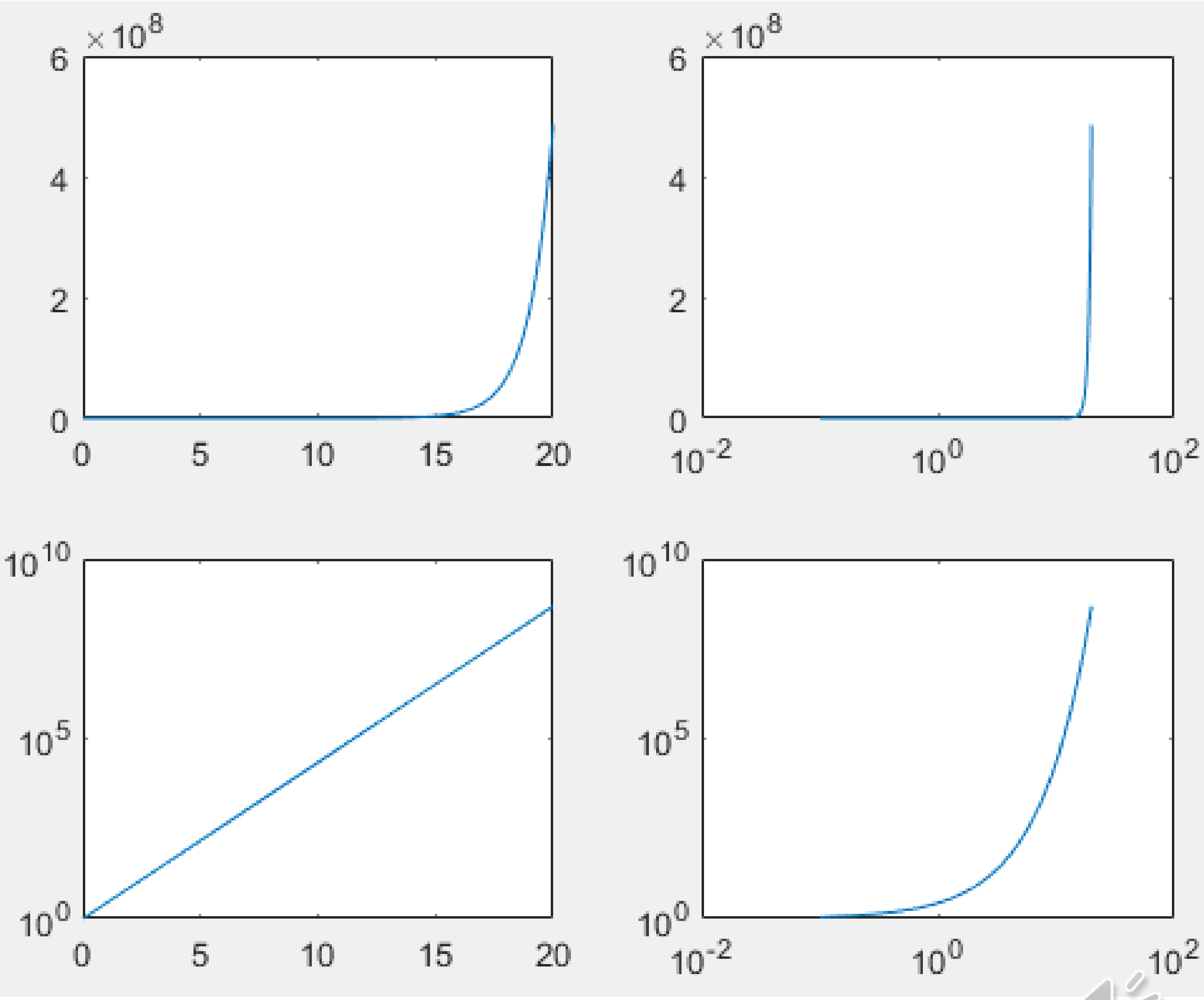
# Example: $e^x$
# Logarithmic plots and subplots

```
x = 0:0.1:20;
y = exp(x);
subplot(2,2,1);
plot(x,y);
subplot(2,2,2);
semilogx(x,y);
subplot(2,2,3);
semilogy(x,y);
subplot(2,2,4);
loglog(x,y);
```



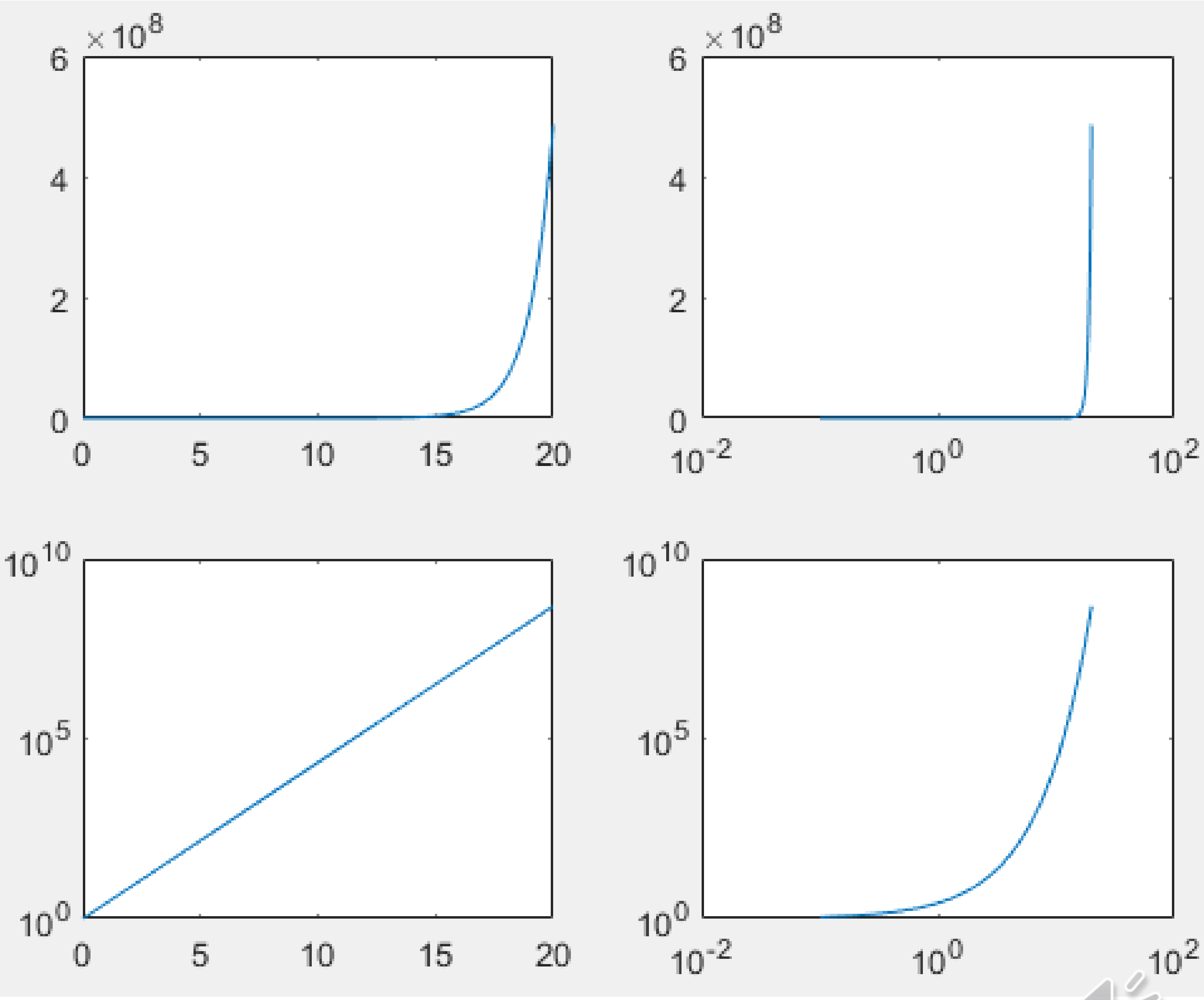
# Example
# Logarithmic plots and subplots

```
x = 0:0.1:20;
y = exp(x);
subplot(2,2,1);
plot(x,y);
subplot(2,2,2);
semilogx(x,y);
subplot(2,2,3);
semilogy(x,y);
subplot(2,2,4);
loglog(x,y);
```



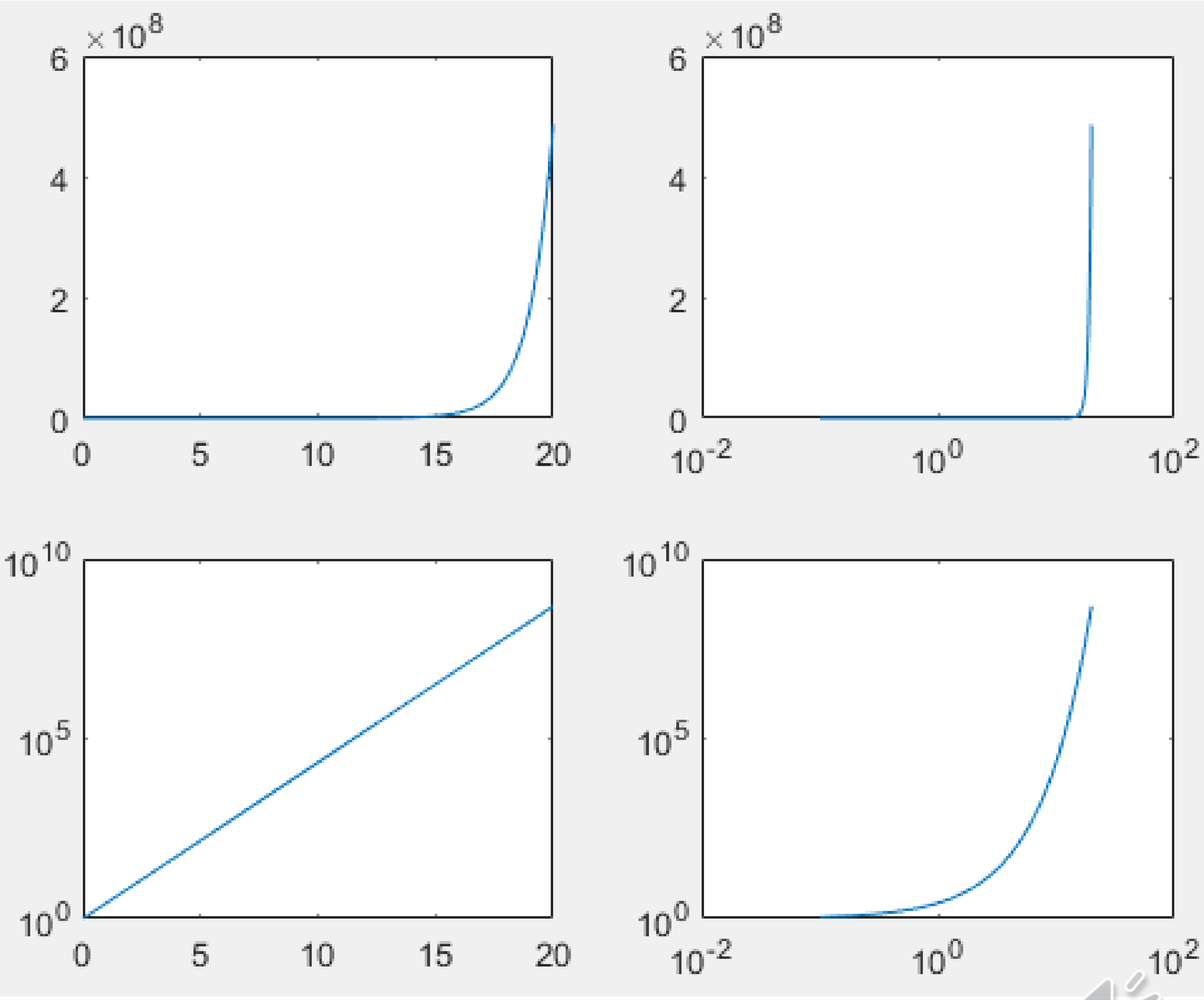
# Example
# Logarithmic plots and subplots

```
x = 0:0.1:20;
y = exp(x);
subplot(2,2,1);
plot(x,y);
subplot(2,2,2);
semilogx(x,y);
subplot(2,2,3);
semilogy(x,y);
subplot(2,2,4);
loglog(x,y);
```



# Example
# Logarithmic plots and subplots

```
x = 0:0.1:20;
y = exp(x);
subplot(2,2,1);
plot(x,y);
subplot(2,2,2);
semilogx(x,y);
subplot(2,2,3);
semilogy(x,y);
subplot(2,2,4);
loglog(x,y);
```



# Bar charts

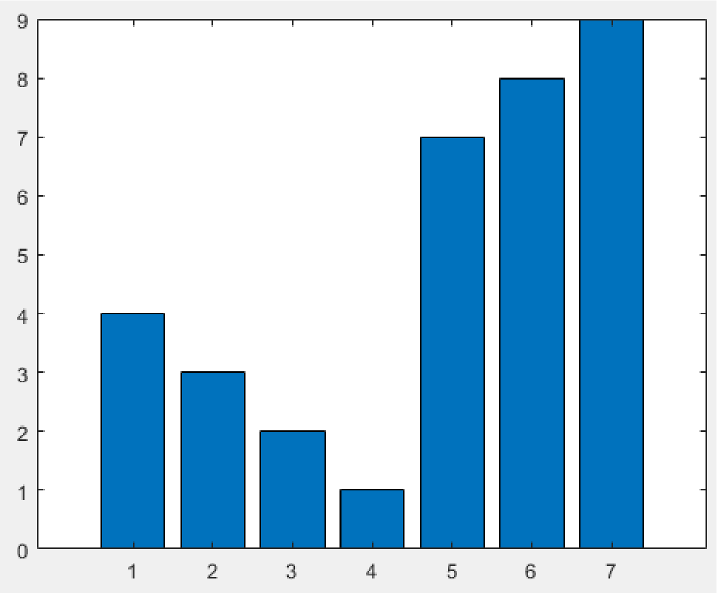
help bar

bar(X,Y) draws the columns of the M-by-N matrix Y as M groups of N vertical bars.  The vector X must not have duplicate values.

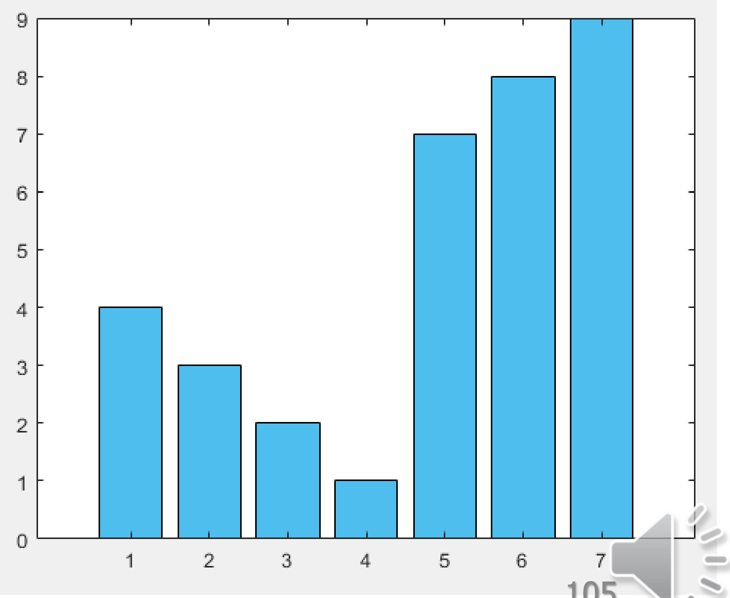
See also histogram, plot, barh, bar3, bar3h.

# Bar charts

```
clear; clf;
figure;
x = [4 3 2 1 7 8 9];
bar(x);
```

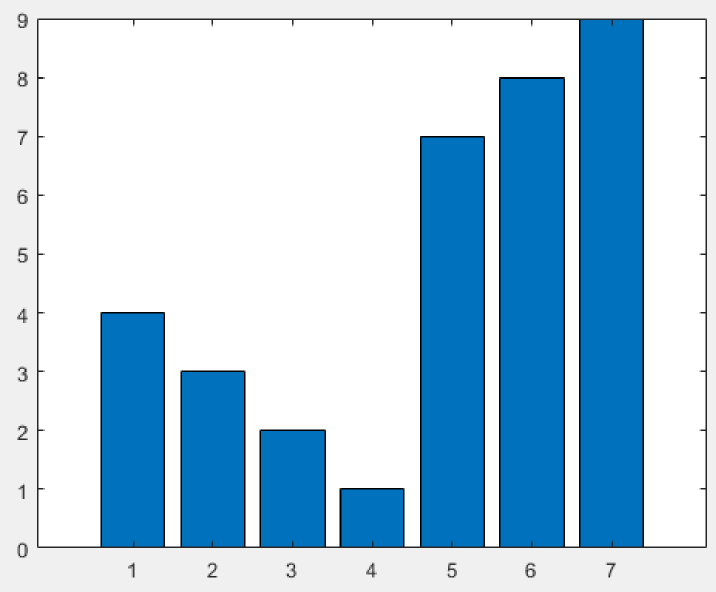


```
clear; clf;
figure;
x = [4 3 2 1 7 8 9];
bar(x, 'FaceColor',...
     [0.3010 0.7450 0.9330]);
```

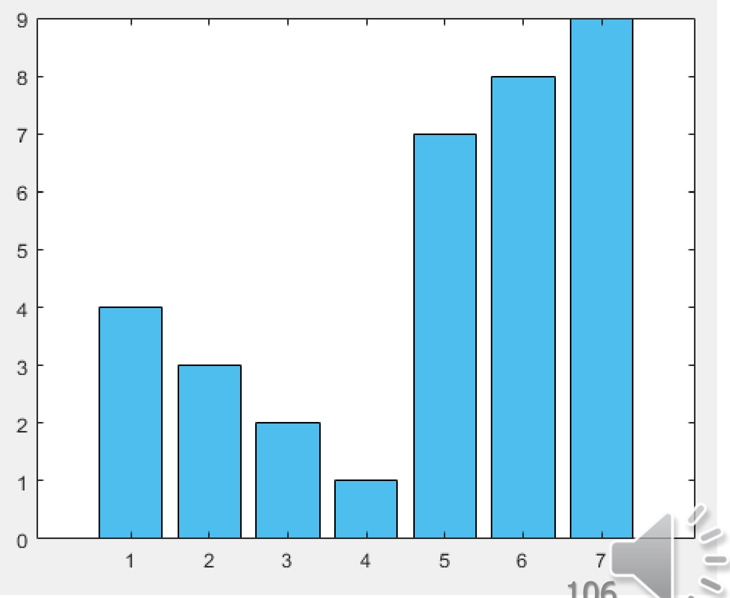


# Bar charts

```
clear; clf;
figure;
x = [4 3 2 1 7 8 9];
bar(x);
```

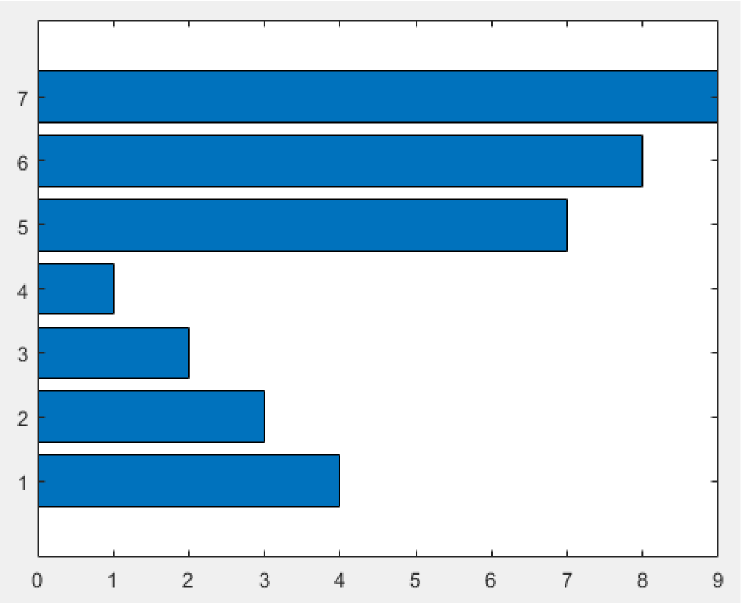


```
clear; clf;
figure;
x = [4 3 2 1 7 8 9];
bar(x, 'FaceColor',...
    [0.3010 0.7450 0.9330]);
```

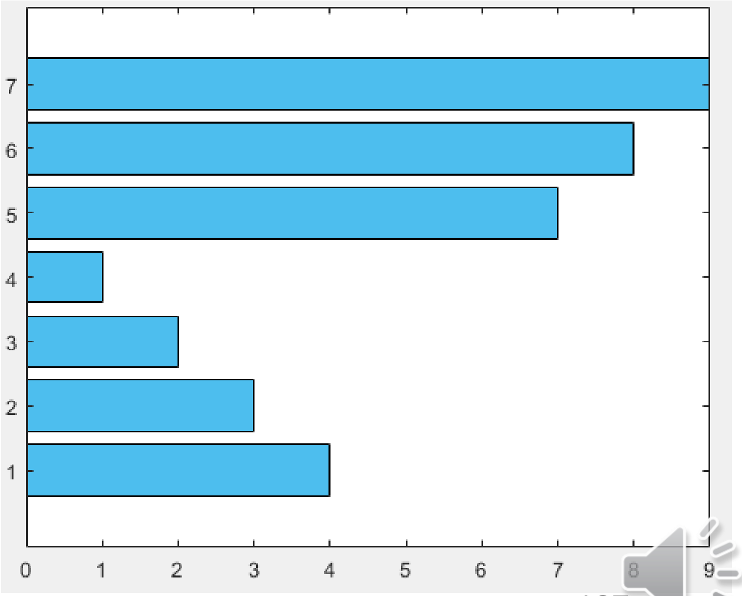


# Bar charts

```
clear; clf;
figure;
x = [4 3 2 1 7 8 9];
barh(x);
```

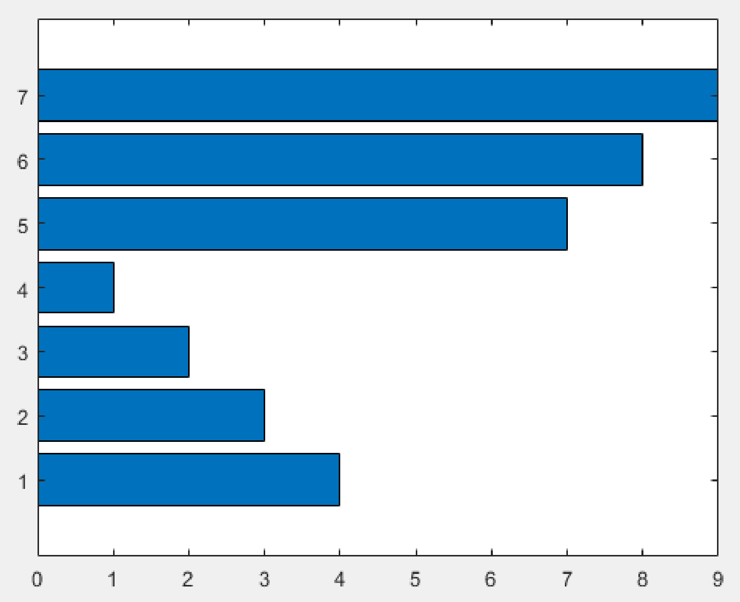


```
clear; clf;
figure;
x = [4 3 2 1 7 8 9];
barh(x, 'FaceColor',...
     [0.3010 0.7450 0.9330]);
   % red     green   blue
```

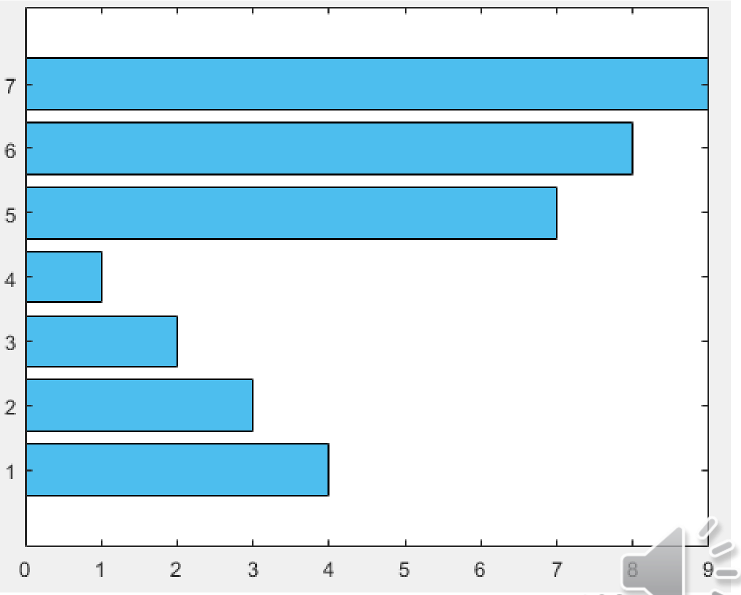


# Bar charts

```
clear; clf;
figure;
x = [4 3 2 1 7 8 9];
barh(x);
```

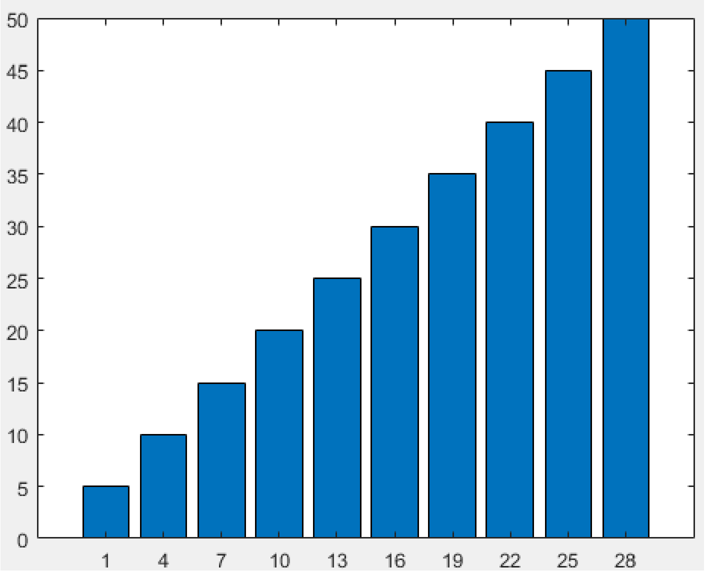


```
clear; clf;
figure;
x = [4 3 2 1 7 8 9];
barh(x, 'FaceColor',...
     [0.3010 0.7450 0.9330]);
   % red      green    blue
```

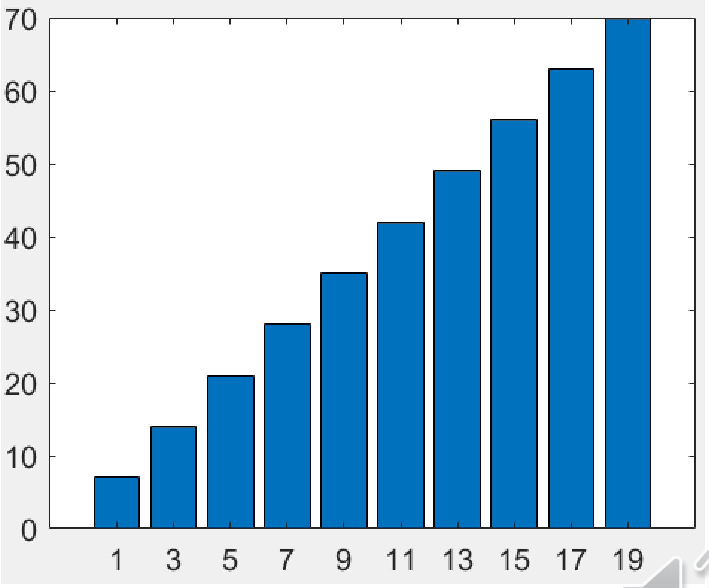


# Bar charts
# Specify Bar Locations

```
clear; clf;
figure;
x = [1:3:30];
y = [5:5:50];
bar(x,y);
```

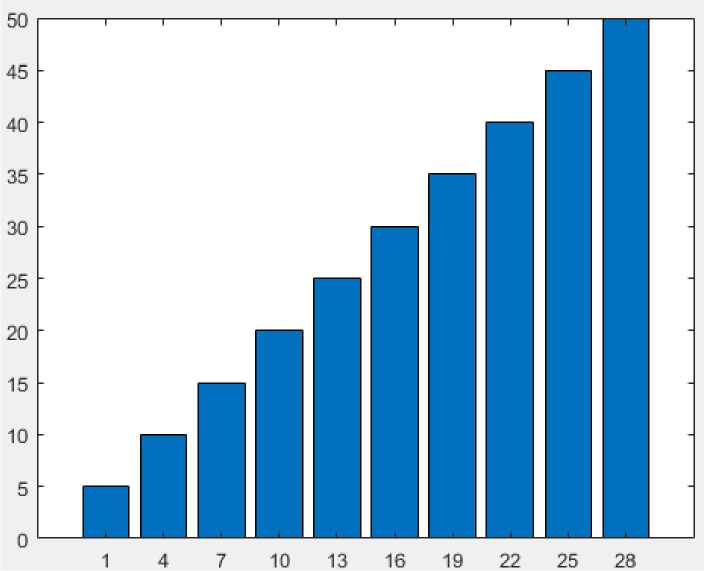


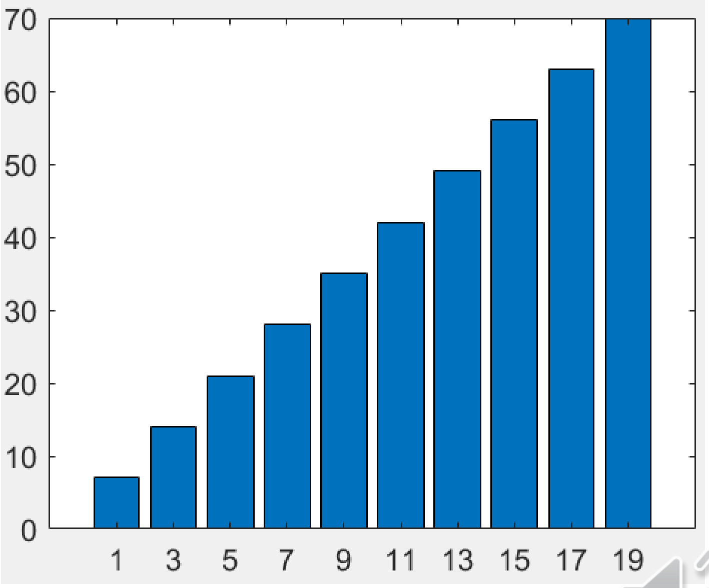
```
clear; clf;
figure;
x = [1:2:20];
y = [7:7:70];
bar(x,y);
set(gca,'FontSize',15);
```



# Bar charts
# Specify Bar Locations

```
clear; clf;
figure;
x = [1:3:30];
y = [5:5:50];
bar(x,y);
```

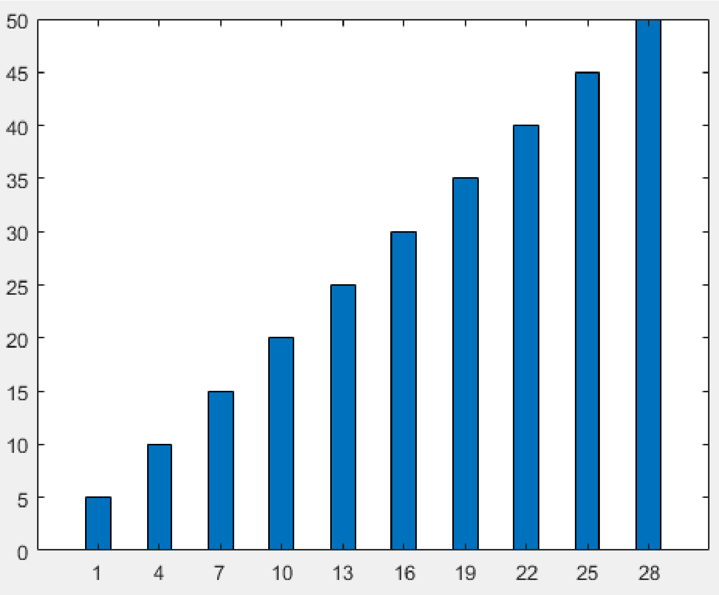




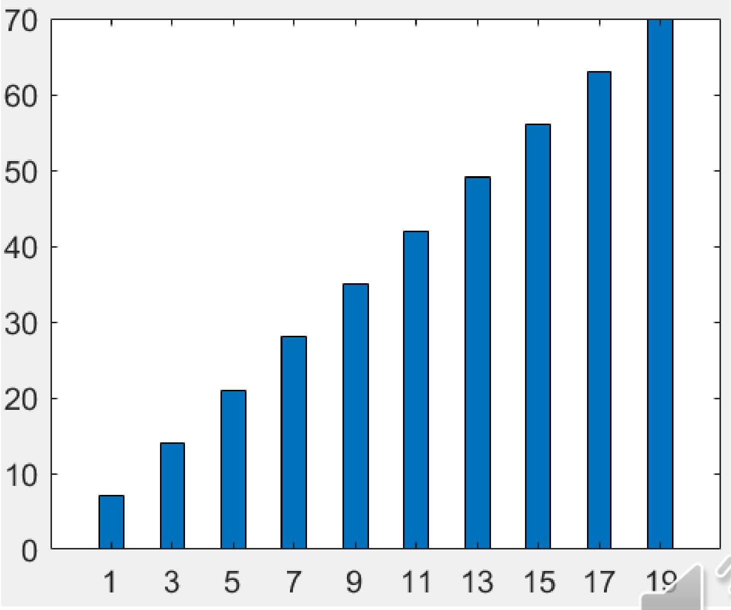
```
clear; clf;
figure;
x = [1:2:20];
y = [7:7:70];
bar(x,y);
set(gca,'FontSize',15); // axe object
```

# Bar charts
# Specify Bar Width

```
clear; clf;
figure;
x = [1:3:30];
y = [5:5:50];
bar(x,0.4);
```



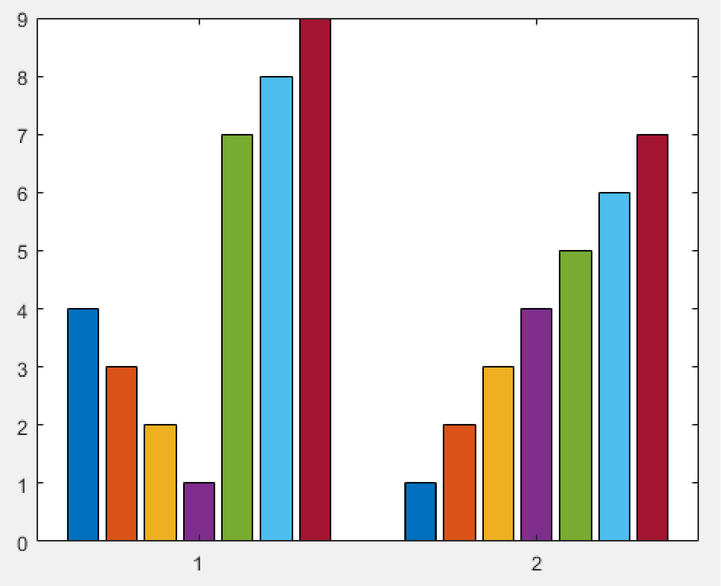
```
clear; clf;
figure;
x = [1:2:20];
y = [7:7:70];
bar(x,y,0.4);
set(gca,'FontSize',15);
```



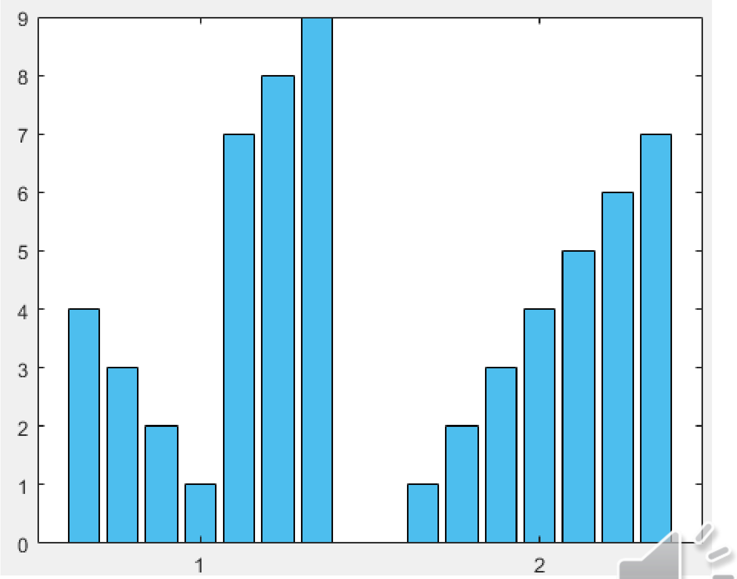
bar(X,Y,WIDTH) or bar(Y,WIDTH) specifies the width of the bars. Values of WIDTH > 1, produce overlapped bars.

# Bar charts: Display Groups of Bars

```
clear; clf;
figure;
x = [4 3 2 1 7 8 9;...
     1 2 3 4 5 6 7
];
bar(x);
```

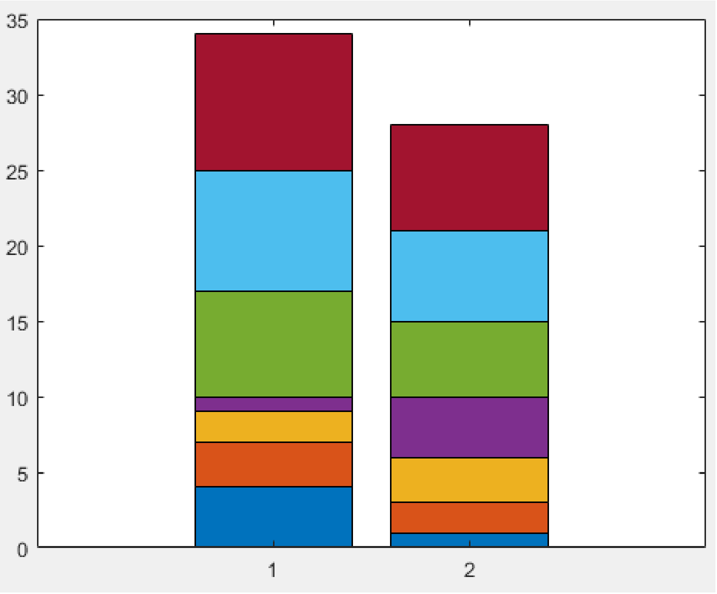


```
clear; clf;
figure;
x = [4 3 2 1 7 8 9;...
     1 2 3 4 5 6 7
bar(x, 'FaceColor',...
    [0.3010 0.7450 0.9330]);
```



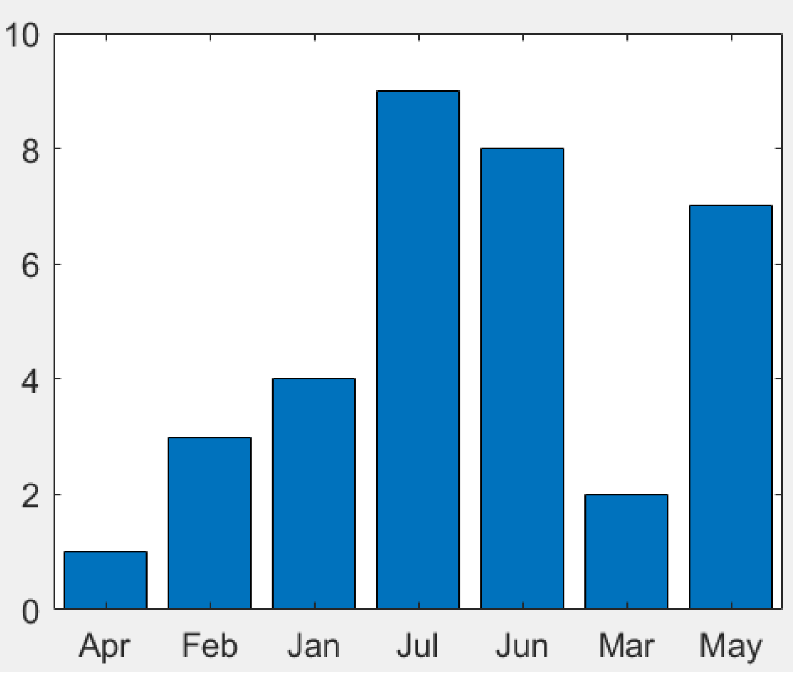
# Bar charts: Display Stacked Bars

```
clear; clf;
figure;
x = [4 3 2 1 7 8 9;...
     1 2 3 4 5 6 7
];
bar(x,'stacked');
```



# Bar charts: Create Bar Graph with Categorical Data

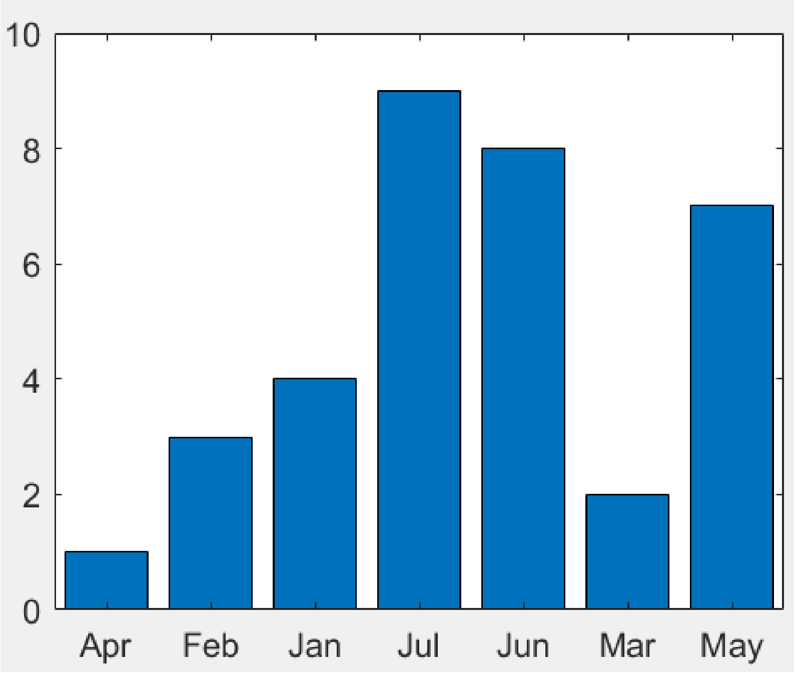
```
clear; clf;
figure;
c =
categorical({'Jan','Feb','Ma
r', 'Apr', 'May', 'Jun',
'Jul'});
x = [4 3 2 1 7 8 9 ...
    ];
bar(c,x,'stacked');
set(gca,'FontSize',15);
```



By default, the categories display in alphabetical order.

# Bar charts: Create Bar Graph with Categorical Data

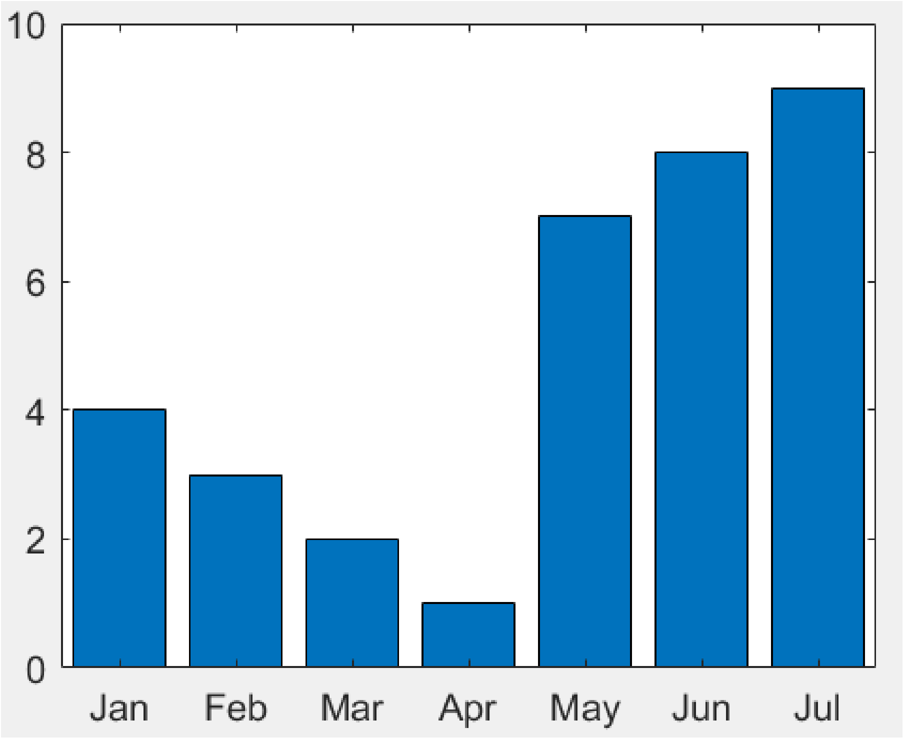
```
clear; clf;
figure;
c = categorical({'Jan','Feb','Mar', 'Apr', 'May', 'Jun', 'Jul'});
x = [4 3 2 1 7 8 9 ...
    ];
bar(c,x,'stacked');
set(gca,'FontSize',15);
```



By default, the categories display in alphabetical order.

# Bar charts: Create Bar Graph with Categorical Data

```
clear; clf;

figure;

list =
categorical({'Jan','Feb','
Mar', 'Apr', 'May', 'Jun',
'Jul'});

x = [4 3 2 1 7 8 9 ...

    ];

c =
categorical(list,list);

bar(c,x);

set(gca,'FontSize',15);
```



Preserve the order

```
c =
1×7 categorical array
Jan      Feb      Mar      Apr      May      Jun      Jul
```

```
list =
1×7 categorical array
Jan      Feb      Mar      Apr      May      Jun      Jul
```

# Bar charts: Create Bar Graph with Categorical Data

```
clear; clf;
figure;
list = categorical({'Jan','Feb','Mar', 'Apr', 'May', 'Jun', 'Jul'});
x = [4 3 2 1 7 8 9 ...
    ];
c = categorical(list,list);
bar(c,x);
set(gca,'FontSize',15);
```

**Value set**

```
categories(list)

ans =

  7×1 cell array

    {'Apr'}
    {'Feb'}
    {'Jan'}
    {'Jul'}
    {'Jun'}
    {'Mar'}
    {'May'}
```

```
categories(c)

ans =

  7×1 cell array

    {'Jan'}
    {'Feb'}
    {'Mar'}
    {'Apr'}
    {'May'}
    {'Jun'}
    {'Jul'}
```

B = categorical(A,valueset) creates one category for each value in valueset. The categories of B are in the same order as the values of valueset.

# Bar charts: Create Bar Graph with Categorical Data

```
clear; clf;

figure;

list =
categorical({'Jan','Feb','
Mar', 'Apr', 'May', 'Jun',
'Jul'}); % use a list

x = [4 3 2 1 7 8 9 ...

    ];

c =
categorical(list,list);

bar(c,x);

set(gca,'FontSize',15);
```

**Value set**

```
categories(list)

ans =

  7×1 cell array

    {'Apr'}
    {'Feb'}
    {'Jan'}
    {'Jul'}
    {'Jun'}
    {'Mar'}
    {'May'}
```

```
categories(c)

ans =

  7×1 cell array

    {'Jan'}
    {'Feb'}
    {'Mar'}
    {'Apr'}
    {'May'}
    {'Jun'}
    {'Jul'}
```

B = categorical(A,valueset) creates one category for each value in valueset. The categories

of B are in the same order as the values of valueset.

# Bar charts: Create Bar Graph with Categorical Data

```
clear; clf;

figure;

list =
categorical({'Jan','Feb','
Mar', 'Apr', 'May', 'Jun',
'Jul'}); % use a list

x = [4 3 2 1 7 8 9 ...

    ];

c =
categorical(list,list);

bar(c,x);

set(gca,'FontSize',15);
```

**Value set**

```
categories(list)

ans =

  7×1 cell array

    {'Apr'}
    {'Feb'}
    {'Jan'}
    {'Jul'}
    {'Jun'}
    {'Mar'}
    {'May'}
```

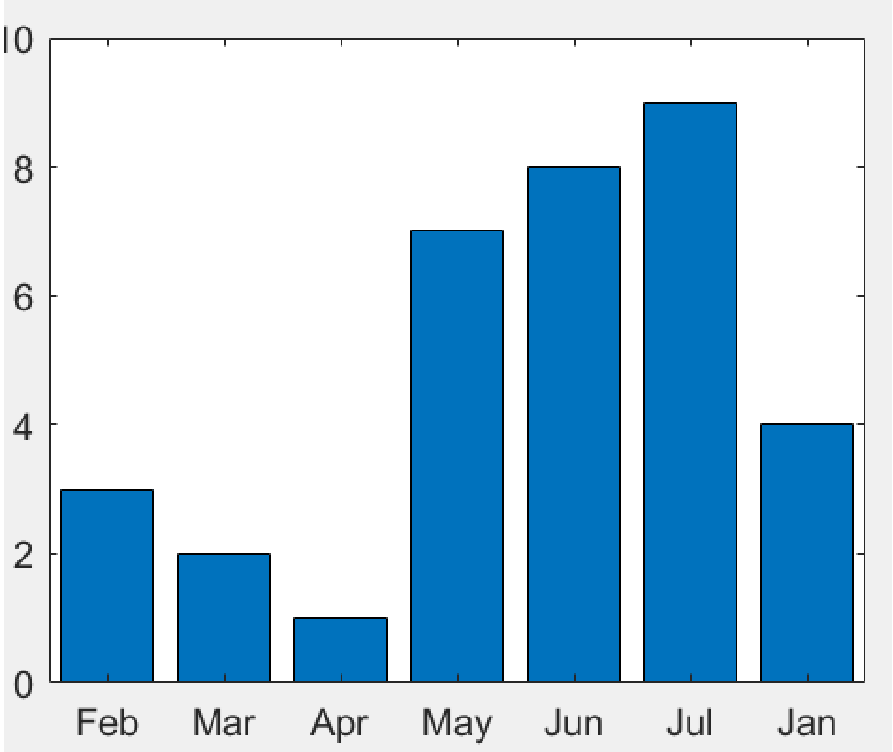
```
categories(c)

ans =

  7×1 cell array

    {'Jan'}
    {'Feb'}
    {'Mar'}
    {'Apr'}
    {'May'}
    {'Jun'}
    {'Jul'}
```

B = categorical(A,valueset) creates one category for each value in valueset. The categories

of B are in the same order as the values of valueset.

# Bar charts: Create Bar Graph with Categorical Data

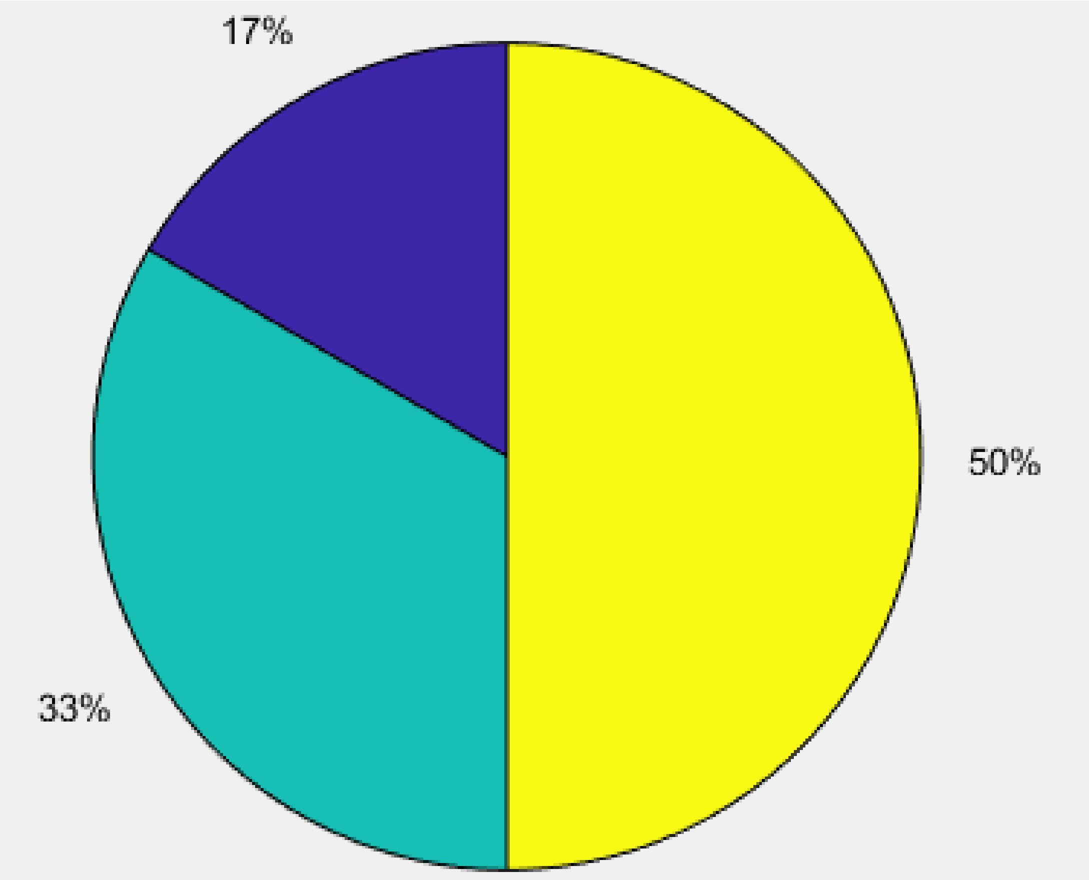
```
clear; clf; figure;

list =
categorical({'Jan','Feb','Mar
', 'Apr', 'May', 'Jun',
'Jul'});

c0 =
categorical({'Feb','Mar',
'Apr', 'May', 'Jun', 'Jul',
'Jan'});

x = [4 3 2 1 7 8 9 ...

    ];

c = categorical(list,c0);

bar(c,x);

set(gca,'FontSize',15);
```



**Value set: define the order of the categories.**

# Pie charts

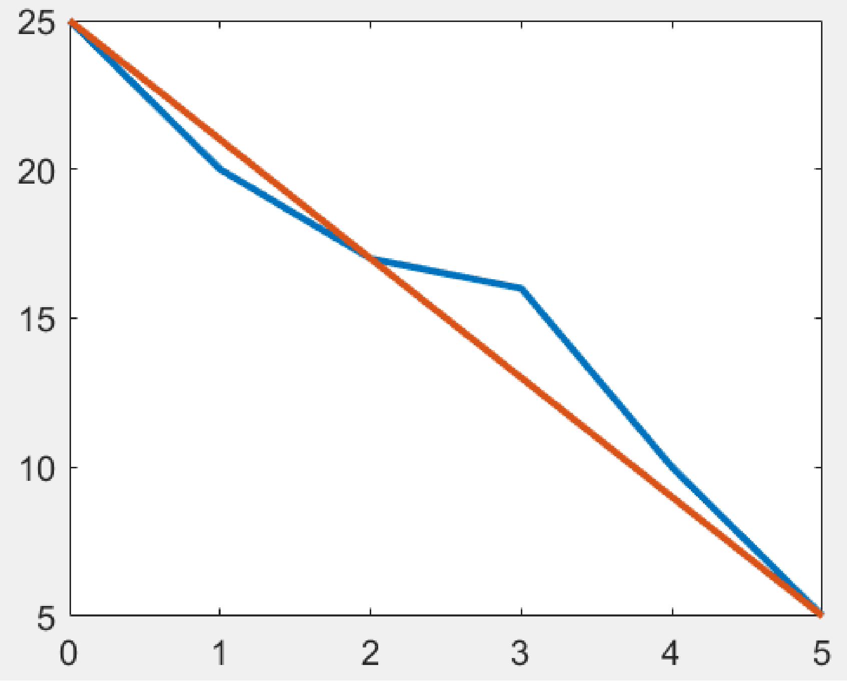
```
clear; clf;
figure;
x = [1 2 3];
pie(x);
set(gca,'FontSize',15);
```



# Curve fitting

- The process of constructing a curve, or mathematical function, that fits to a series of data points in the best way.

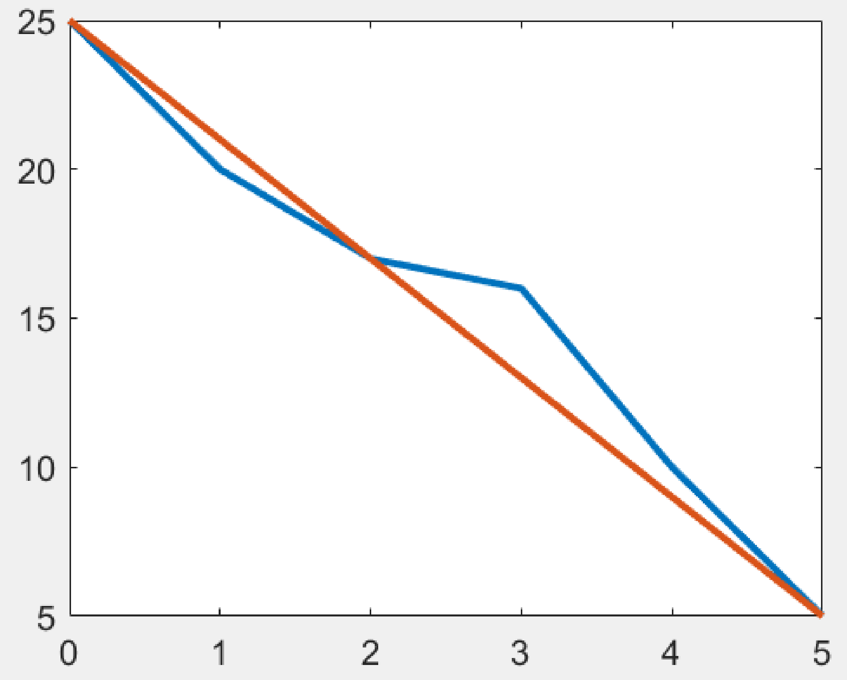
# Linear approximation



```
clear; clf;
figure;
x = [0:5]
y = [25,20,17,16,10,5]
plot(x,y, 'Linewidth',3);
set(gca,'FontSize',15);



%y = a*x + b
a = (y(end)-y(1))/(x(end) - x(1));
b = ???;
y1 = a.*(x-x(1)) + y(1);
hold on
plot(x,y1, 'Linewidth',3);
```

# Linear approximation

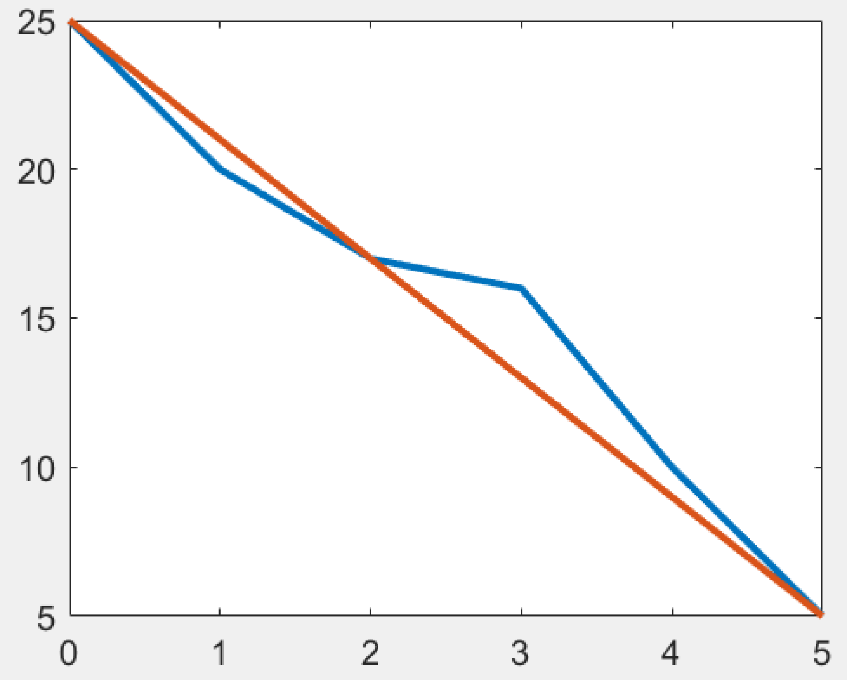
```
clear; clf;
figure;
x = [0:5]
y = [25,20,17,16,10,5]
plot(x,y, 'Linewidth',3);
set(gca,'FontSize',15);
```



```
%y = a*x + b
a = (y(end)-y(1))/(x(end) - x(1));
b = ???;
y1 = a.*(x-x(1)) + y(1);
hold on
plot(x,y1, 'Linewidth',3);
```

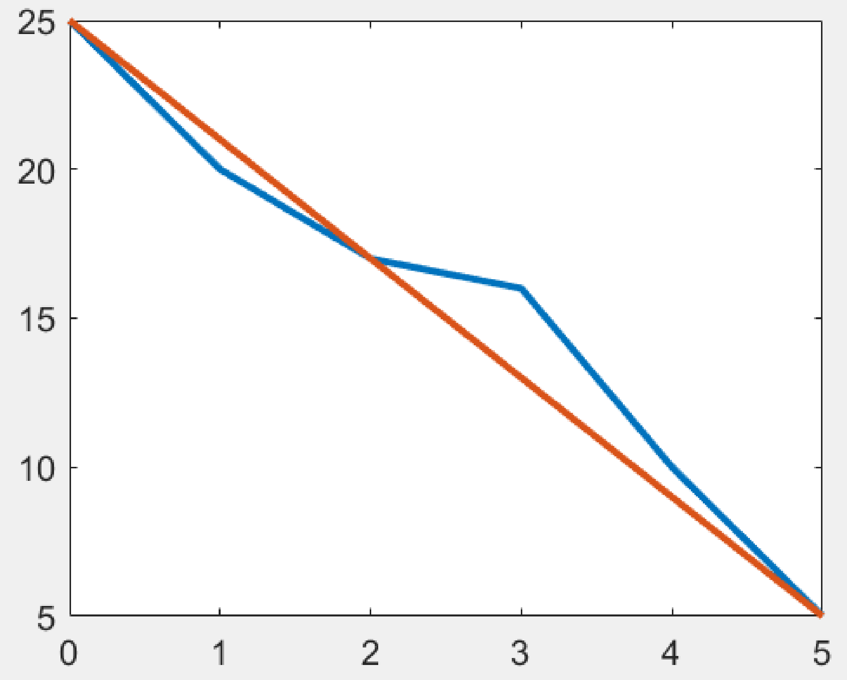
# Linear approximation

```
clear; clf;
figure;
x = [0:5]
y = [25,20,17,16,10,5]
plot(x,y, 'Linewidth',3);
set(gca,'FontSize',15);
```



```
%y = a*x + b
a = (y(end)-y(1))/(x(end) - x(1));
b = ???;
y1 = a.*(x-x(1)) + y(1);
hold on
plot(x,y1, 'Linewidth',3);
```

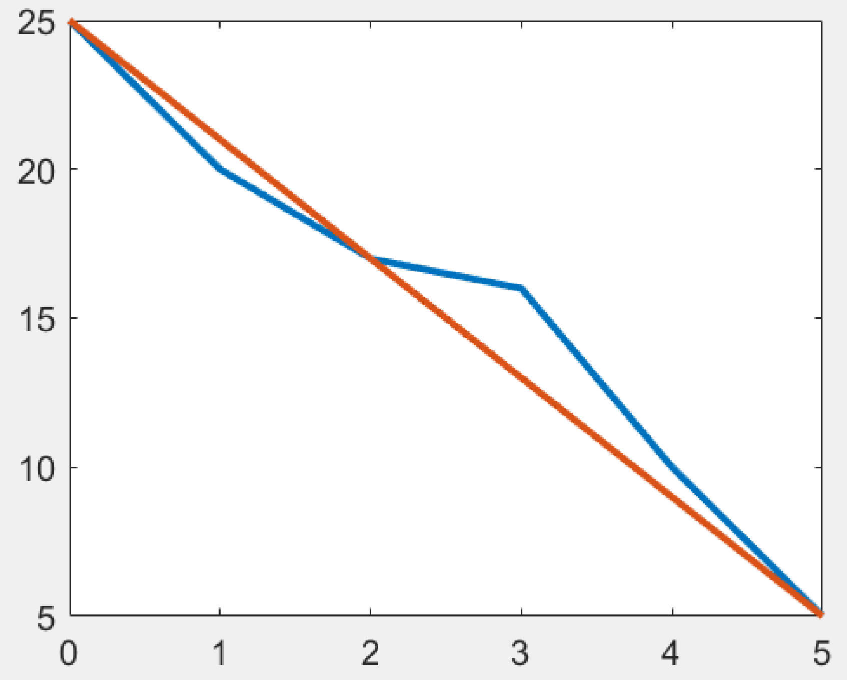
# Linear approximation



```
clear; clf;
figure;
x = [0:5]
y = [25,20,17,16,10,5]
plot(x,y, 'Linewidth',3);
set(gca,'FontSize',15);



%y = a*x + b
a = (y(end)-y(1))/(x(end) - x(1));  % slope
b = ???;
y1 = a.*(x-x(1)) + y(1);
hold on
plot(x,y1, 'Linewidth',3);
```

# Linear approximation

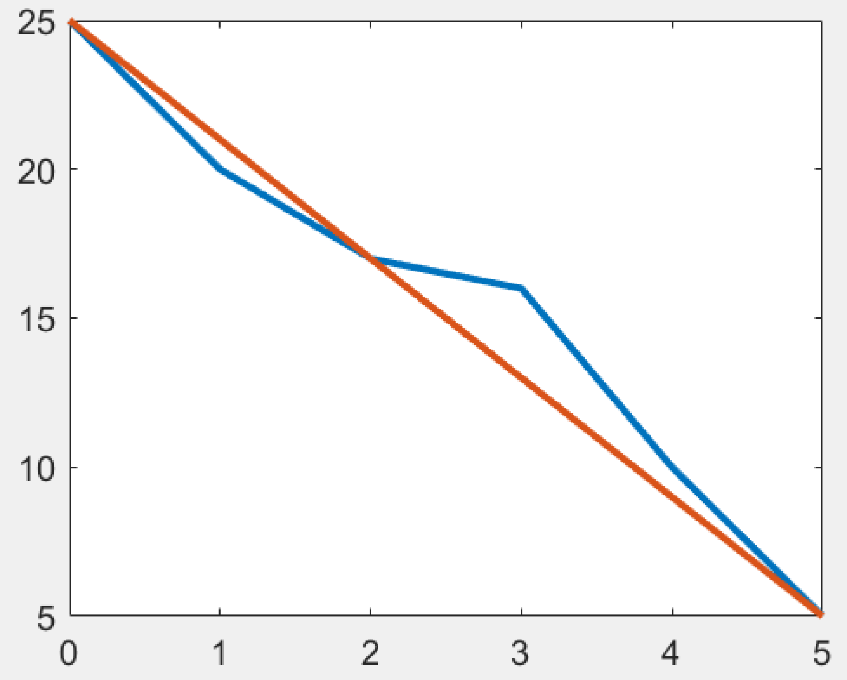
```
clear; clf;
figure;
x = [0:5]
y = [25,20,17,16,10,5]
plot(x,y, 'Linewidth',3);
set(gca,'FontSize',15);
```



```
%y = a*x + b
a = (y(end)-y(1))/(x(end) - x(1));  % slope
b = ???; // do not need to explicitly evaluate b
y1 = a.*(x-x(1)) + y(1);
hold on
plot(x,y1, 'Linewidth',3);
```

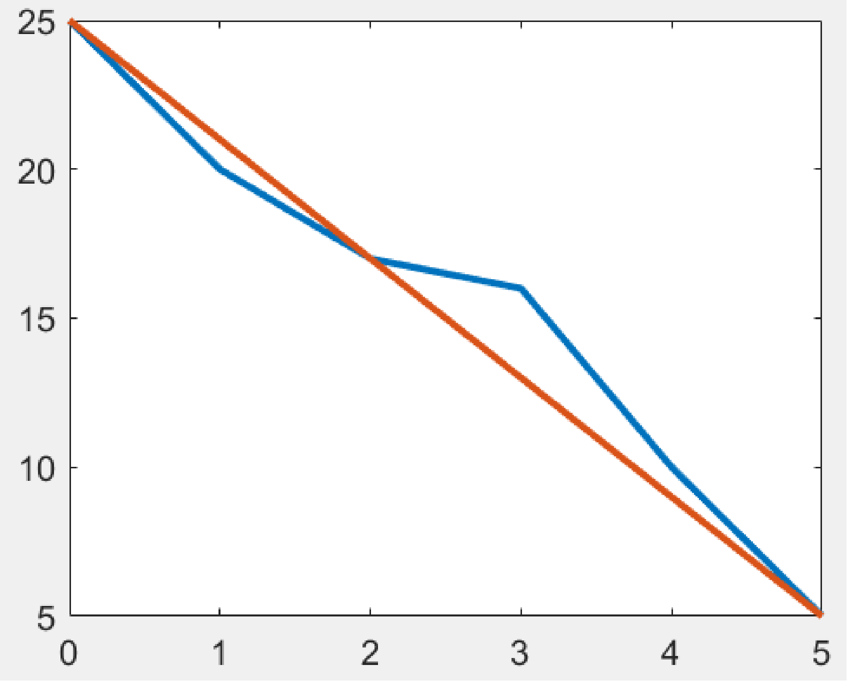
# Linear approximation

```
clear; clf;
figure;
x = [0:5]
y = [25,20,17,16,10,5]
plot(x,y, 'Linewidth',3);
set(gca,'FontSize',15);
```



```
%y = a*x + b
a = (y(end)-y(1))/(x(end) - x(1));  % slope
b = ???; // do not need to explicitly evaluate b
y1 = a.*(x-x(1)) + y(1);
hold on
plot(x,y1, 'Linewidth',3);
```

# Linear approximation



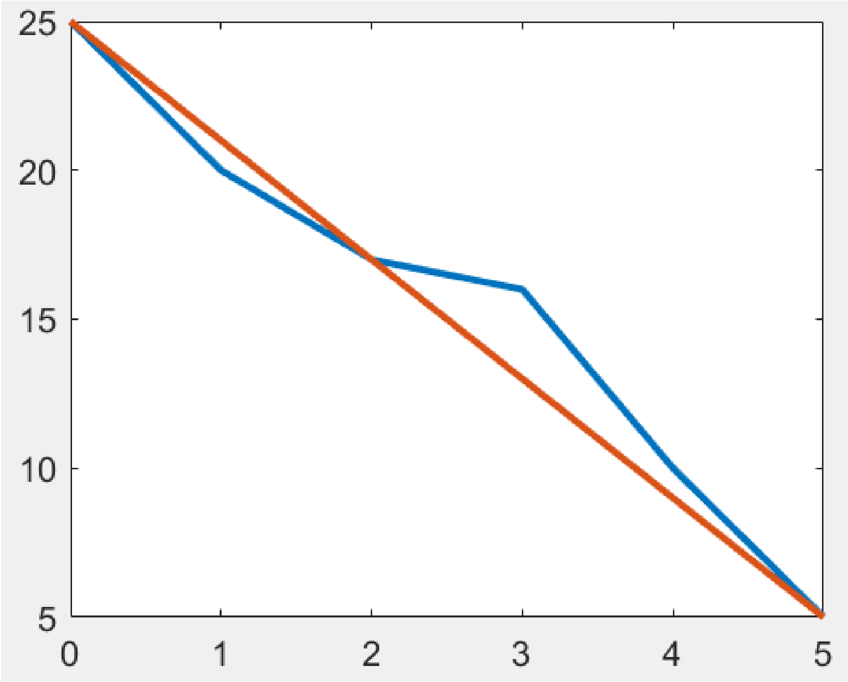
```
clear; clf;
figure;
x = [0:5]
y = [25,20,17,16,10,5]
plot(x,y, 'Linewidth',3);
set(gca,'FontSize',15);



%y = a*x + b
a = (y(end)-y(1))/(x(end) - x(1));  % slope
b = ???; // do not need to explicitly evaluate b
y1 = a.*(x-x(1)) + y(1);
hold on
plot(x,y1, 'Linewidth',3);
```

# polyfit

- Fit polynomial to data.
- P = polyfit(X,Y,N) finds the coefficients of a polynomial P(X) of  degree N that fits the data Y **best in a least-squares sense**.

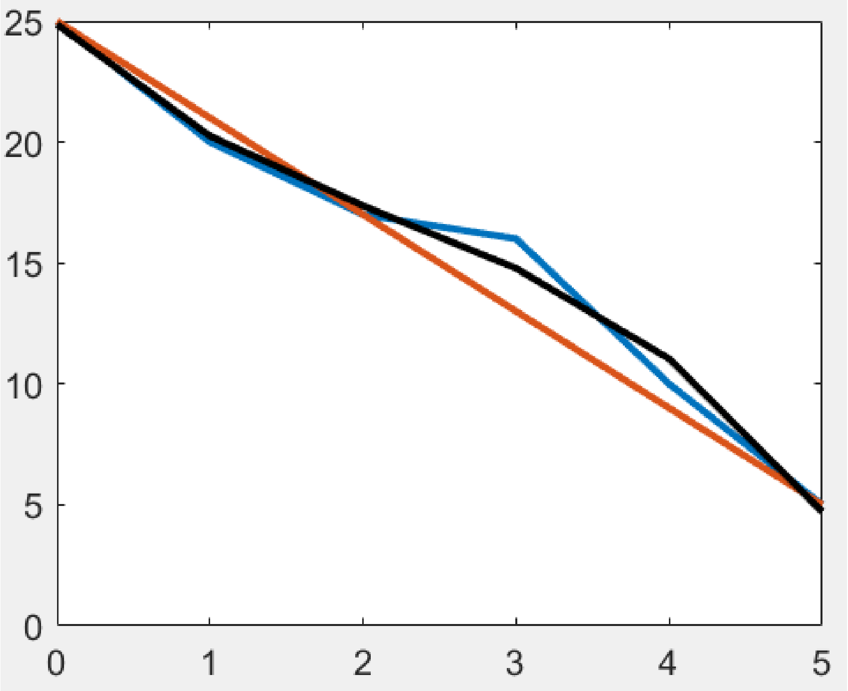
# polyfit

```
clear; clf;
figure;
x = [0:5]
y = [25,20,17,16,10,5]
plot(x,y, 'Linewidth',3);
set(gca,'FontSize',15);
```



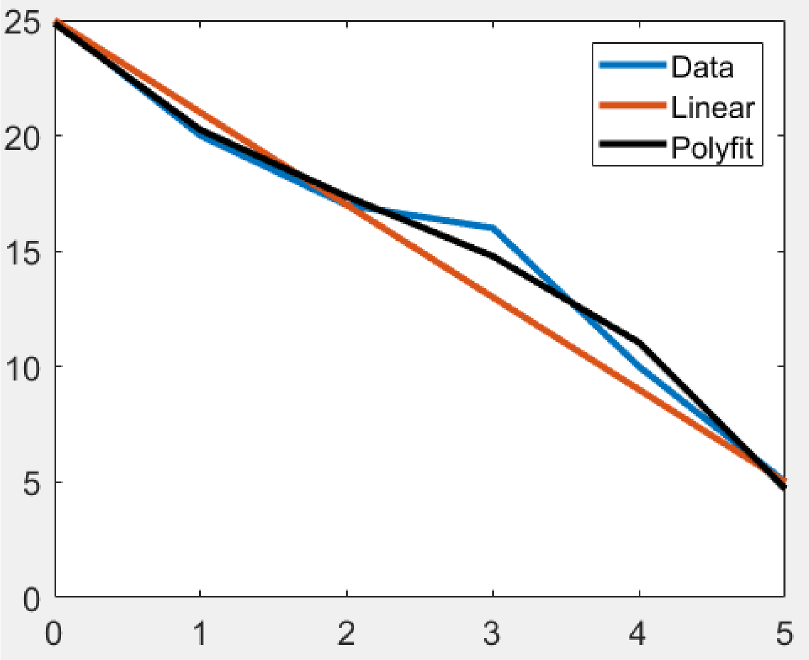
```
%y = a*x + b
a = (y(end)-y(1))/(x(end) - x(1));
b = ???;
y1 = a.*(x-x(1)) + y(1);
hold on
plot(x,y1, 'Linewidth',3);
```

# polyfit



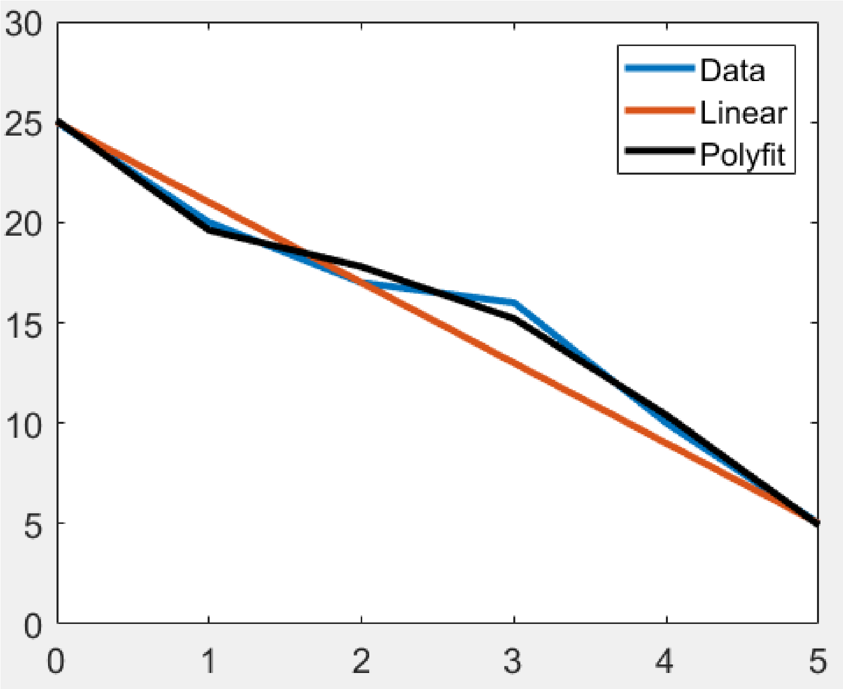
```
clear; clf; figure;
x = [0:5];
y = [25,20,17,16,10,5];
plot(x,y, 'Linewidth',3);
set(gca,'FontSize',15);


%y = a*x + b
a = (y(end)-y(1))/(x(end) - x(1));
b = ???; y1 = a.*(x-x(1)) + y(1);
hold on; plot(x,y1, 'Linewidth',3);
%%%%%%
p = polyfit(x,y,3);
y2 = polyval(p,x);
plot(x,y2, 'Linewidth',3, 'color','k'); % black
```

# polyfit



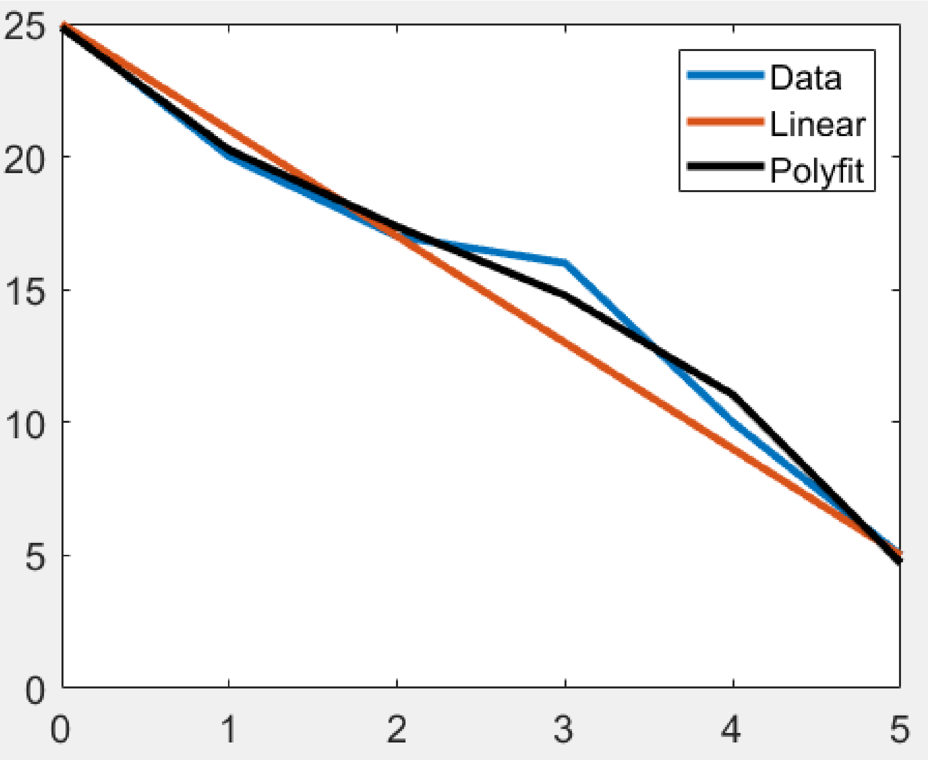
```
clear; clf; figure;
x = [0:5];
y = [25,20,17,16,10,5];
plot(x,y, 'Linewidth',3);
set(gca,'FontSize',15);


%y = a*x + b
a = (y(end)-y(1))/(x(end) - x(1));
b = ???; y1 = a.*(x-x(1)) + y(1);
hold on; plot(x,y1, 'Linewidth',3);
%%%%%%
p = polyfit(x,y,3);
y2 = polyval(p,x);
plot(x,y2, 'Linewidth',3, 'color','k');
legend({'Data' 'Linear' 'Polyfit'});
```

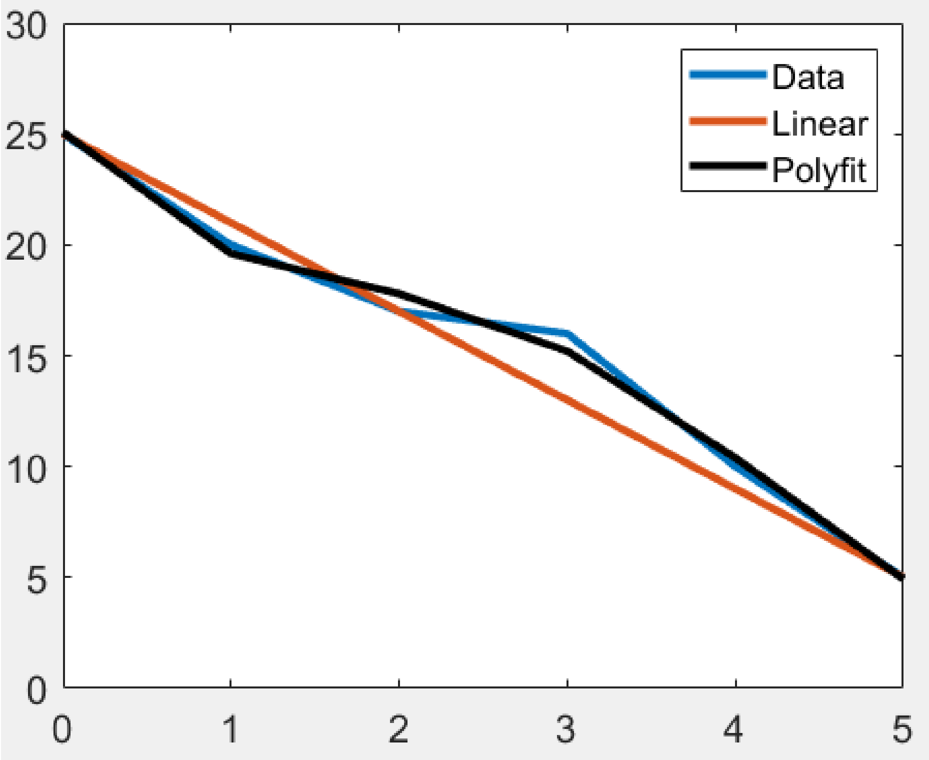
# polyfit



```
clear; clf; figure;
x = [0:5];
y = [25,20,17,16,10,5];
plot(x,y, 'Linewidth',3);
set(gca,'FontSize',15);


%y = a*x + b
a = (y(end)-y(1))/(x(end) - x(1));
b = ???; y1 = a.*(x-x(1)) + y(1);
hold on; plot(x,y1, 'Linewidth',3);
%%%%%%
p = polyfit(x,y,4);
y2 = polyval(p,x);
plot(x,y2, 'Linewidth',3, 'color','k');
legend({'Data' 'Linear' 'Polyfit'});
```

# polyfit



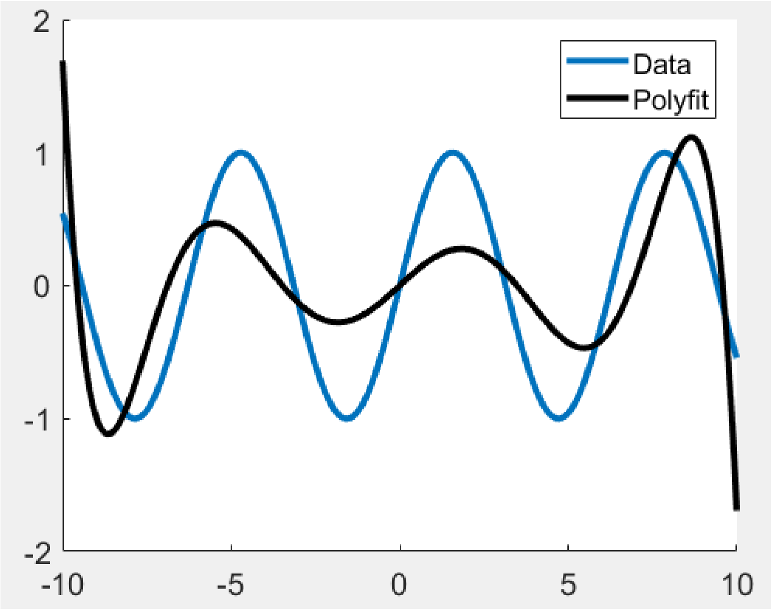


```
p = polyfit(x,y,3);
```

```
p = polyfit(x,y,4);
```

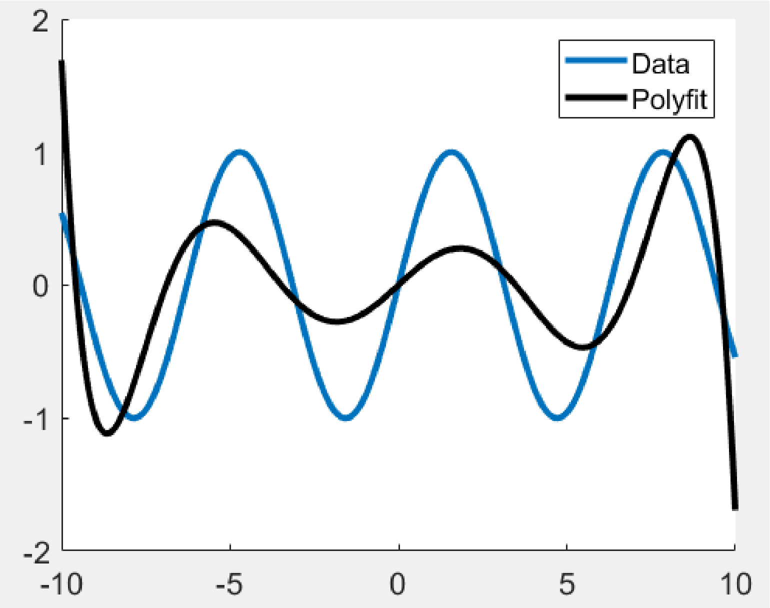
# polyfit

```
clear;close all;
hold on
x = [-10:0.05:10];
y = sin(x);
plot(x,y, 'Linewidth',3);
p = polyfit(x,y,7);
y2 = polyval(p,x);
plot(x,y2, 'Linewidth',3, 'color','k');
legend({'Data' 'Polyfit'});
set(gca,'FontSize',15);
hold off
```



# polyfit

```
clear;close all;
hold on
x = [-10:0.05:10];
y = sin(x);
plot(x,y, 'Linewidth',3);
p = polyfit(x,y,7);
y2 = polyval(p,x);
plot(x,y2, 'Linewidth',3,
'color','k');
legend({'Data'
'Polyfit'});
set(gca,'FontSize',15);
hold off
```



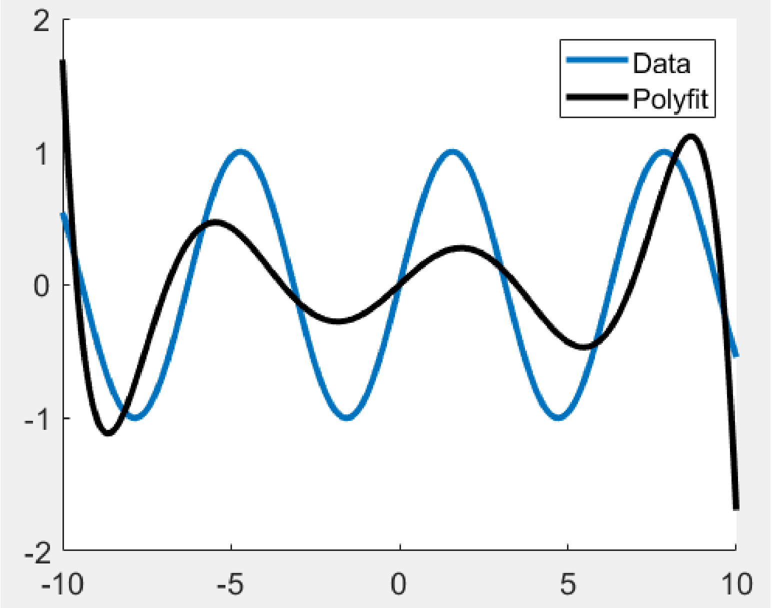
There are 6 turns.

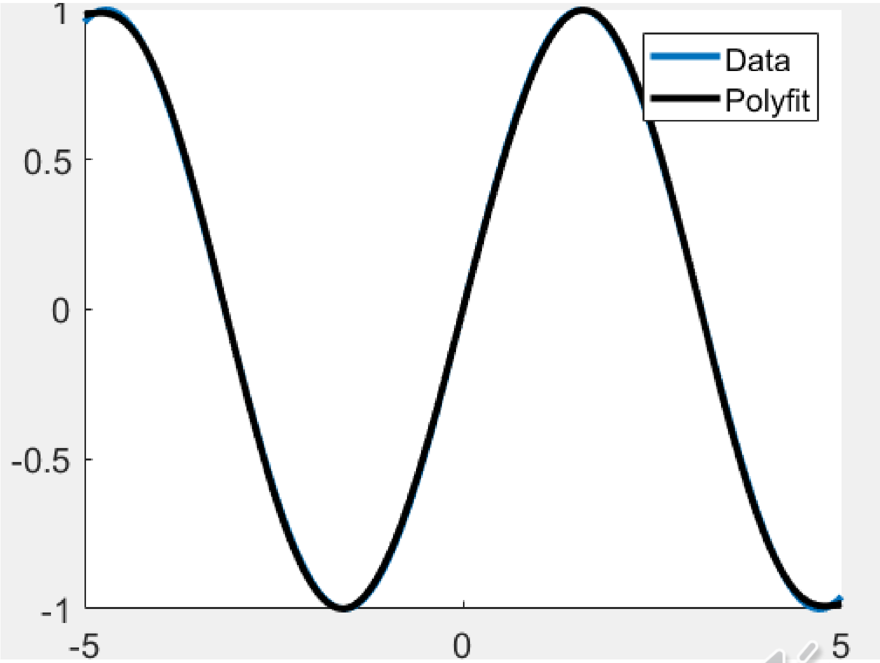
A polynomial function with degree 7.
It does not fit well to the data.

# polyfit

```
x = [-10:0.05:10];
```

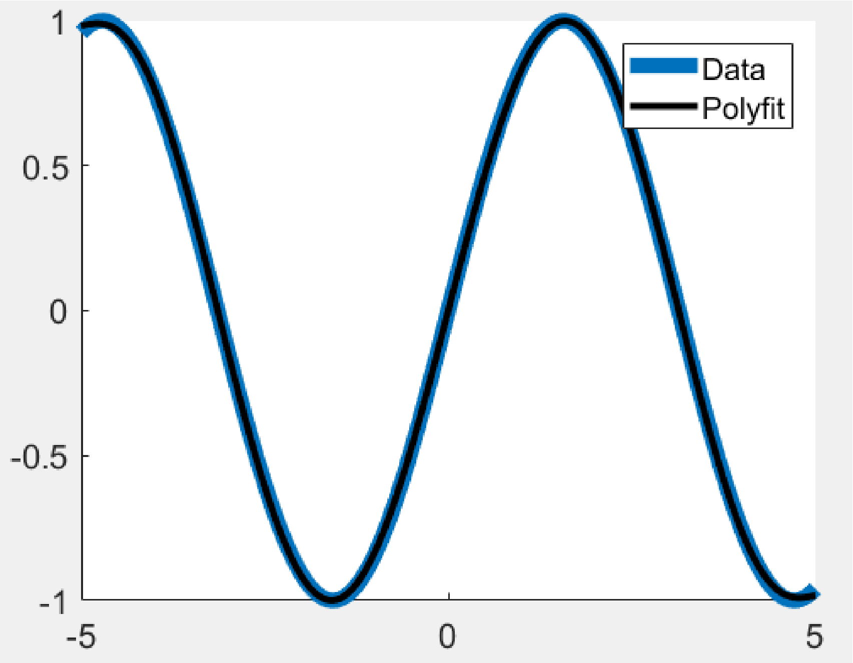
```
clear;close all;
hold on
x = [-5:0.05:5];
y = sin(x);
plot(x,y, 'Linewidth',3);
p = polyfit(x,y,7);
y2 = polyval(p,x);
plot(x,y2, 'Linewidth',3,
'color','k');
legend({'Data'
'Polyfit'});
set(gca,'FontSize',15);
hold off
```





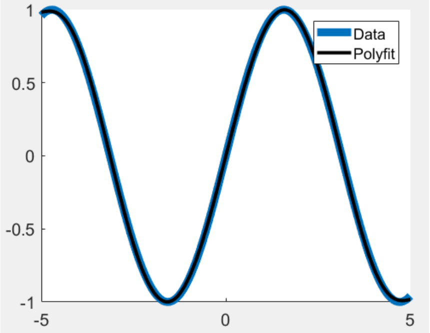
# polyfit

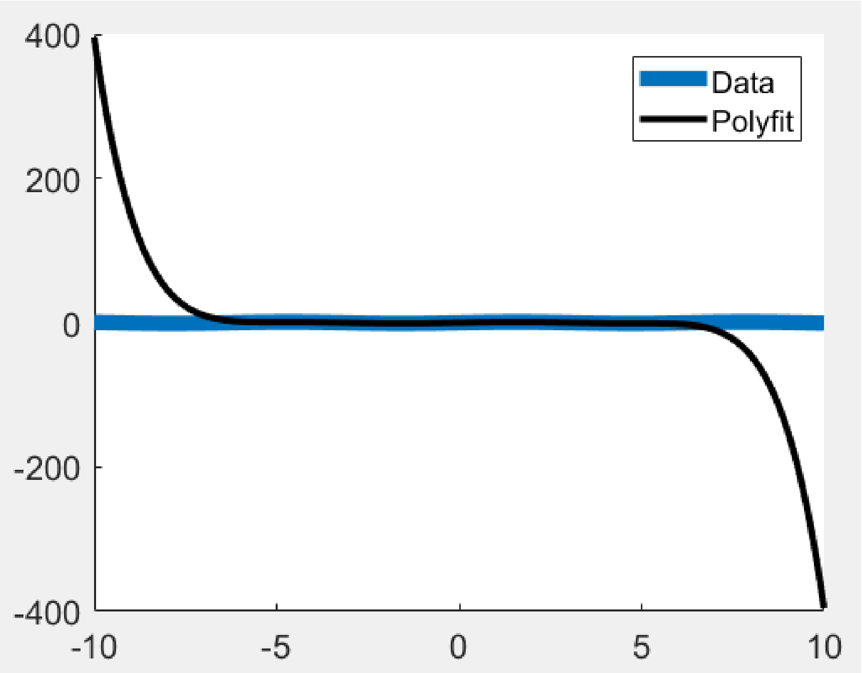
```
clear;close all;
hold on
x = [-5:0.05:5];
y = sin(x);
plot(x,y, 'Linewidth',7);
p = polyfit(x,y,7);
y2 = polyval(p,x);
plot(x,y2, 'Linewidth',3,
'color','k');
legend({'Data'
'Polyfit'});
set(gca,'FontSize',15);
hold off
```



# polyfit

```
clear;close all; hold on;
dataX = [-5:0.05:5];
dataY = sin(x);
p = polyfit(dataX,dataY,7);
%Test data
x = [-10:0.05:10];
y = sin(x);
y2 = polyval(p,x);
plot(x,y, 'Linewidth',7);
plot(x,y2, 'Linewidth',3,
'color','k');
legend({'Data' 'Polyfit'});
set(gca,'FontSize',15);
hold off
```

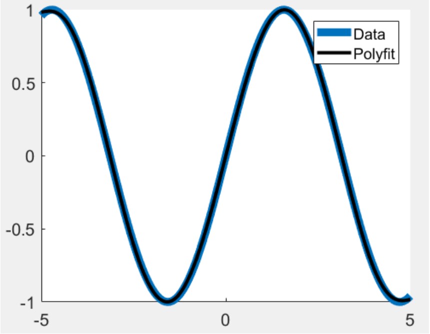


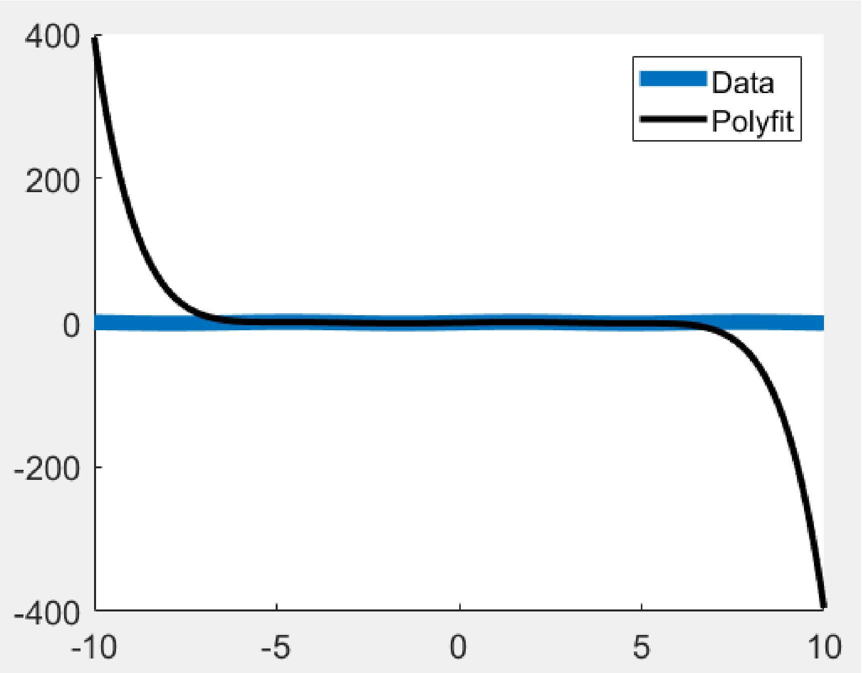


Let f(x) be a polynomial function. The number of turning points (relative extrema) of f(x) is, at most, one less than the degree of the polynomial.

# polyfit

```
clear;close all; hold on;
dataX = [-5:0.05:5];
dataY = sin(x);
p = polyfit(dataX,dataY,7);
%Test data
x = [-10:0.05:10];
y = sin(x);
y2 = polyval(p,x);
plot(x,y, 'Linewidth',7);
plot(x,y2, 'Linewidth',3,
'color','k');
legend({'Data' 'Polyfit'});
set(gca,'FontSize',15);
hold off
```

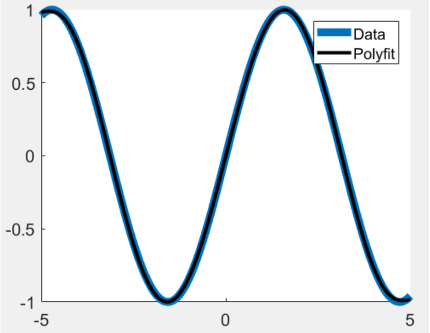


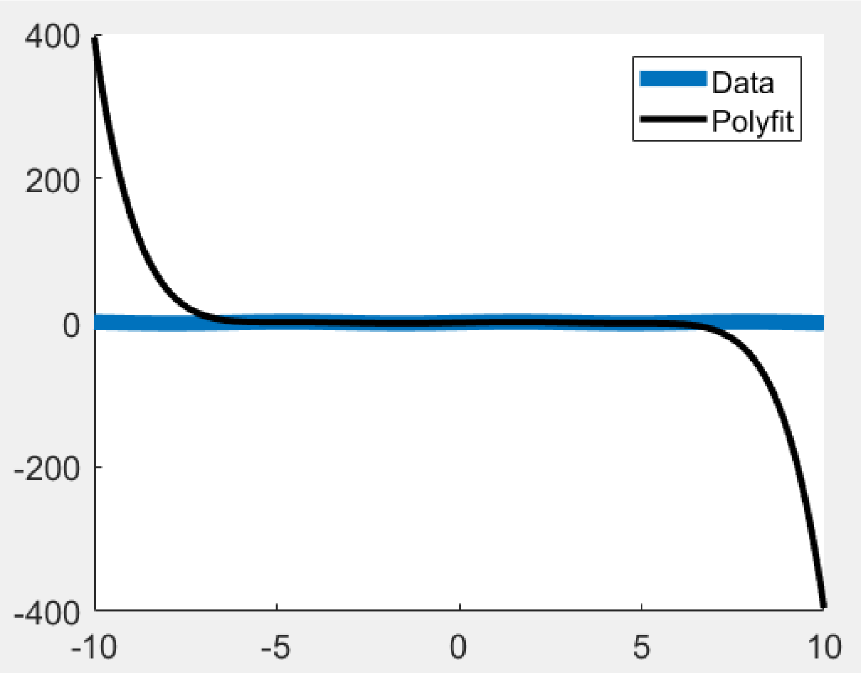


Let f(x) be a polynomial function. The number of turning points (relative extrema) of f(x) is, at most, one less than the degree of the polynomial.

# polyfit

```
clear;close all; hold on;
dataX = [-5:0.05:5];
dataY = sin(x);
p = polyfit(dataX,dataY,7);
%Test data
x = [-10:0.05:10];
y = sin(x);
y2 = polyval(p,x);
plot(x,y, 'Linewidth',7);
plot(x,y2, 'Linewidth',3,
'color','k');
legend({'Data' 'Polyfit'});
set(gca,'FontSize',15);
hold off
```

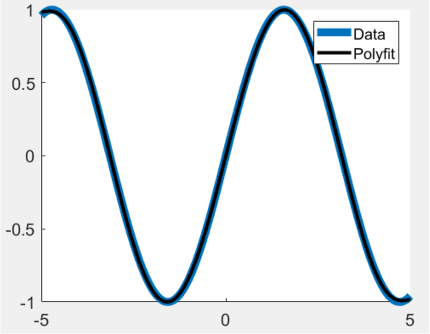


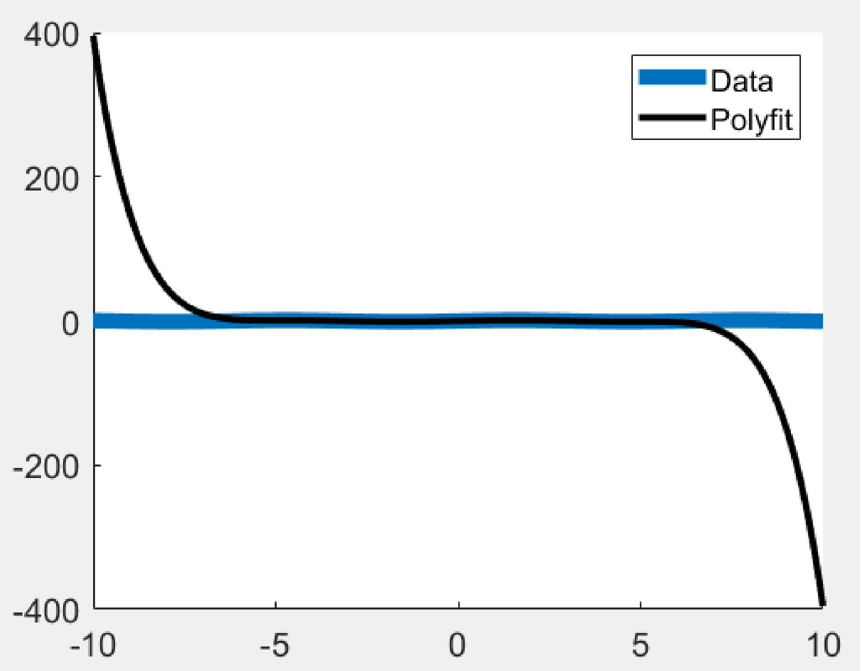


Let f(x) be a polynomial function. The number of turning points (relative extrema) of f(x) is, at most, one less than the degree of the polynomial.

# polyfit

```
clear;close all; hold on;
dataX = [-5:0.05:5];
dataY = sin(x);
p = polyfit(dataX,dataY,7);
%Test data
x = [-10:0.05:10];
y = sin(x);
y2 = polyval(p,x);
plot(x,y, 'Linewidth',7);
plot(x,y2, 'Linewidth',3,
'color','k');
legend({'Data' 'Polyfit'});
set(gca,'FontSize',15);
hold off
```

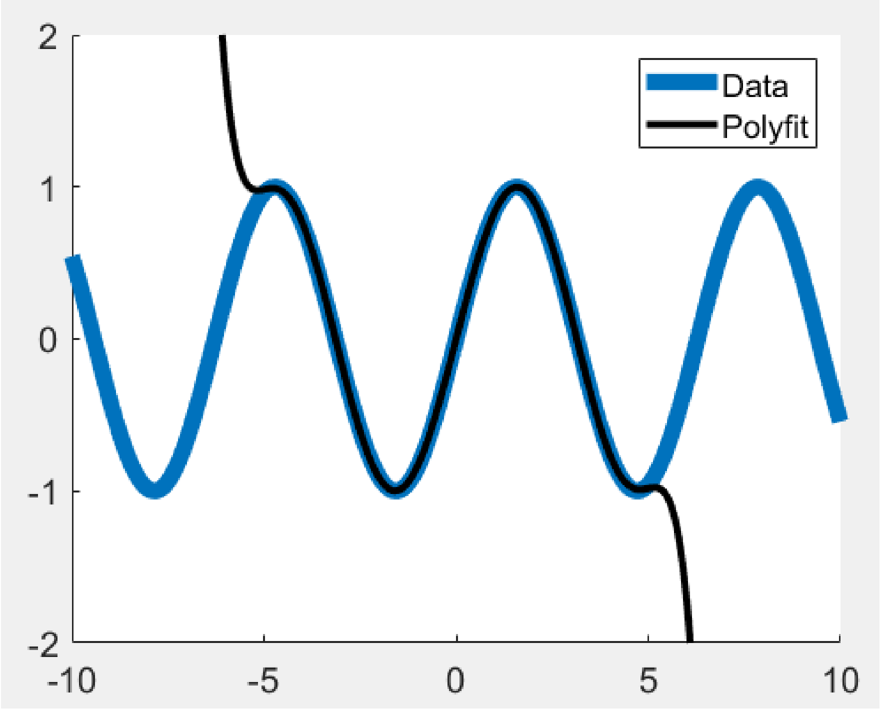




Let f(x) be a polynomial function.
The number of turning points (relative extrema) of f(x) is, at most, one less than the degree of the polynomial.

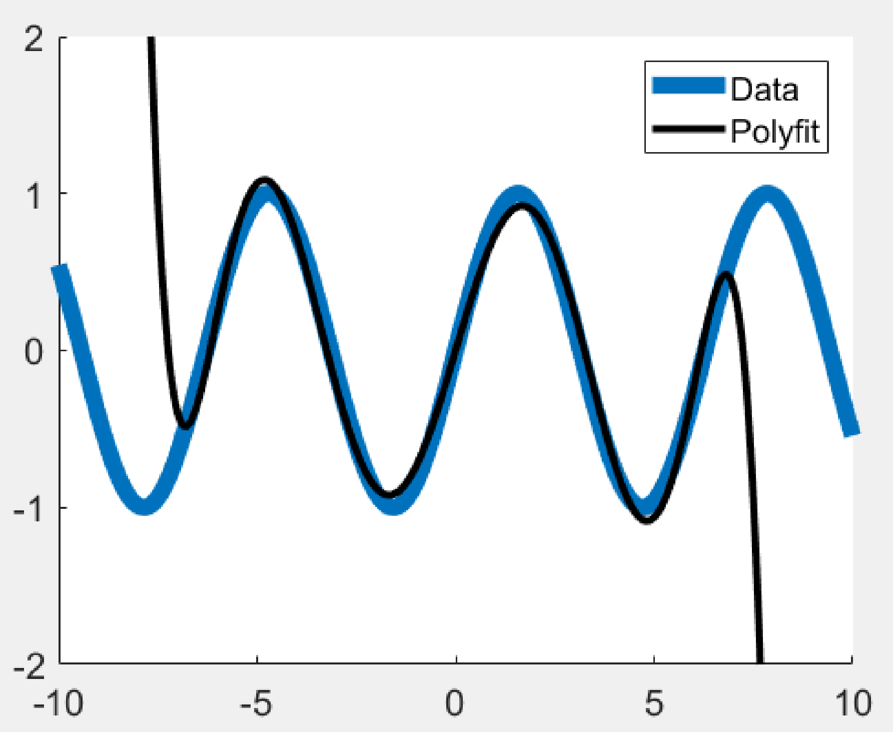
# polyfit

```
clear;close all; hold on;
dataX = [-5:0.05:5];
dataY = sin(x);
p = polyfit(dataX,dataY,7);
%Test data
x = [-10:0.05:10];
y = sin(x);
y2 = polyval(p,x);
plot(x,y, 'Linewidth',7);
plot(x,y2, 'Linewidth',3, 'color','k');
legend({'Data' 'Polyfit'});
set(gca,'FontSize',15); axis([-10 10 -2 2]);
hold off
```



# polyfit

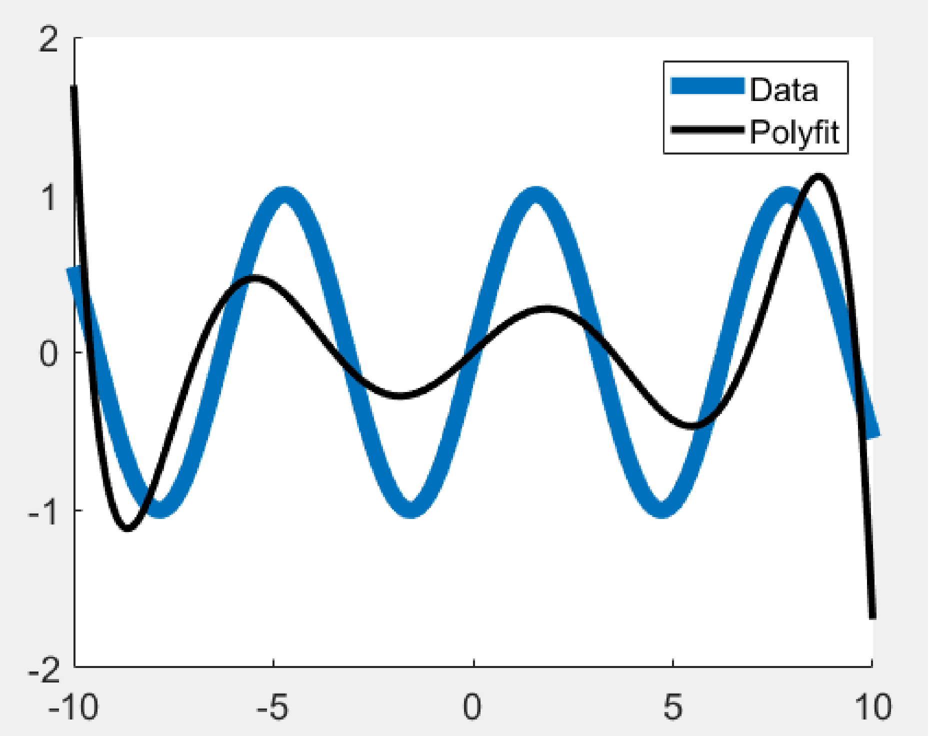
```
clear;close all; hold on;
dataX = [-7:0.05:7];
dataY = sin(x);
p = polyfit(dataX,dataY,7);
%Test data
x = [-10:0.05:10];
y = sin(x);
y2 = polyval(p,x);
plot(x,y, 'Linewidth',7);
plot(x,y2, 'Linewidth',3, 'color','k');
legend({'Data' 'Polyfit'});
set(gca,'FontSize',15); axis([-10 10 -2 2]);
hold off
```



# polyfit

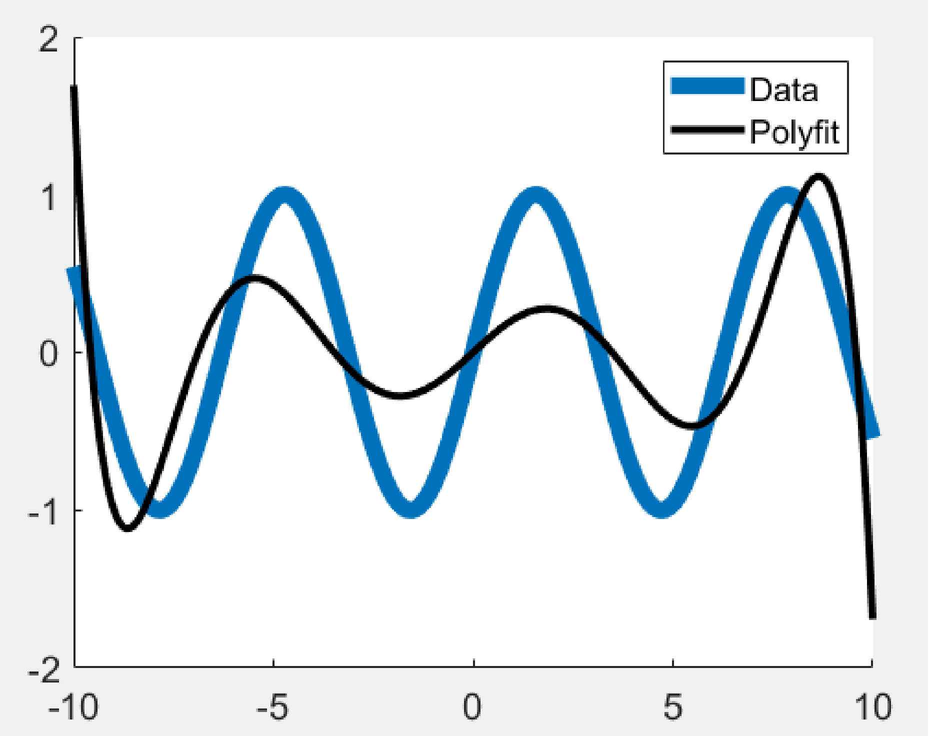
```
clear;close all; hold on;
dataX = [-10:0.05:10];
dataY = sin(x);
p = polyfit(dataX,dataY,7);
%Test data
x = [-10:0.05:10];
y = sin(x);
y2 = polyval(p,x);
plot(x,y, 'Linewidth',7);
plot(x,y2, 'Linewidth',3, 'color','k');
legend({'Data' 'Polyfit'});
set(gca,'FontSize',15); axis([-10 10 -2 2]);
hold off
```

6 turning points



# polyfit

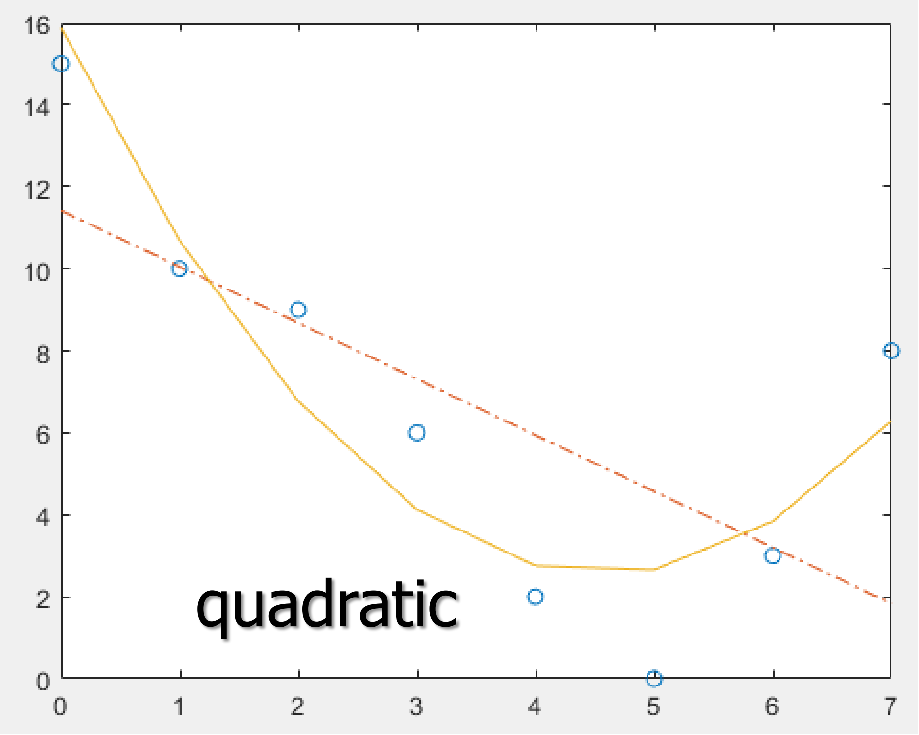
6 turning points

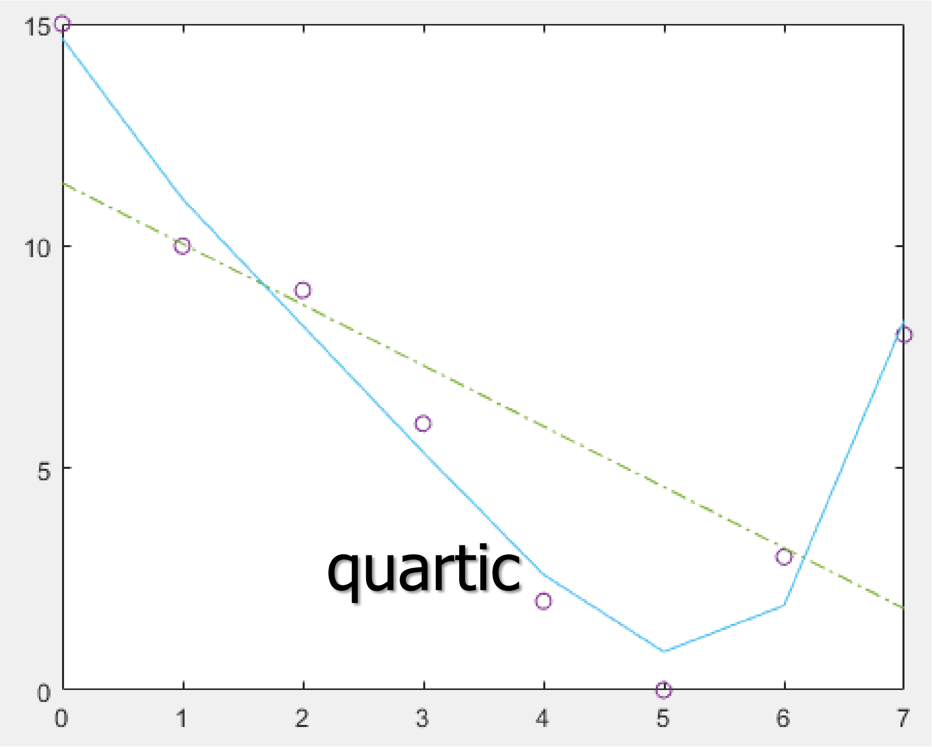


```
clear;close all; hold on;
dataX = [-10:0.05:10];
dataY = sin(x);
p = polyfit(dataX,dataY,15?)
%Test data
x = [-10:0.05:10];
y = sin(x);
y2 = polyval(p,x);
plot(x,y, 'Linewidth',7);
plot(x,y2, 'Linewidth',3, 'color','k');
legend({'Data' 'Polyfit'});
set(gca,'FontSize',15); axis([-10 10 -2 2]);
hold off
```

# Exercise
# polyfit function.  polyval(x,p)

```
c4 = polyfit(x,y,4);
y4 = c4(1) .* x.^4 + c4(2) .* x.^3 + ...
     c4(3) .* x.^2 + c4(4) .* x + c4(5);
```





$$f(x) = ax^4 + bx^3 + cx^2 + dx + e$$

# Exercises

- Draw two curves. Fill the regions by the two curves and an x-interval.
- Draw three curves. Fill the regions by the two curves and an x-interval. The regions do not contain the third curve except at the boundary of the two curves.