2D Plots

```
% Clears variables, command window, and closes all figures
clear, clc,close all

% Defines value of x and two functions
x = 0: .1 : 2*pi;
y1 = cos(x);
y2 = sin(x);

% Plots two functions with different style, and wider lines
plot(x,y1,'b', x, y2, 'r-.', 'linewidth', 2)

% Activates the grid
grid on

% Defines limits for x and y axes, and sets title, labels and legends
axis([0 2*pi -1.5 1.5])
title('2D plots', 'fontsize', 12)
xlabel('angle')
ylabel('f1(x), f2(x)')
legend('cos(x)', 'sin(x)')

% Keeps figure on screen, in order to add a third function
hold on

% Defines another function
y3 = 0.5 * x;

% Plots over the previous figure
plot(x, y3, 'm')
```

Horizontal Lines and Vertical lines

```
% Initial vertical line
a = linspace(L_min, L_min, 10);
if d == 1
   b = linspace(y-r*.03, y, 10);
else
   b = linspace(y, y+r*.03, 10);
end
plot(a,b,'k-','linewidth',1.5)
hold on

% Final vertical line
a = linspace(L_max, L_max, 10);
if d == 1
   b = linspace(y-r*.03, y, 10);
```

```
else
   b = linspace(y, y+r*.03, 10);
end
plot(a,b,'k-','linewidth',1.5)
```

Now, we'll test our function with this script.

```
clear; clc; close all

x = 0 : 2*pi/360 : 2*pi;
y = sin(x);

plot(x,y)
grid on
xlabel('angle')
ylabel('sin(x)')
title('Plot showing horizontal and vertical lines')
hold on
plot_limit([0.8, 2.5, 0.4, 1, 2])
plot_limit([4, 5.4, -0.6, -1, 2])
```

Pie Plots

```
% Clears variables, command window, and closes all figures
clc; clear; close all

% These are the names of the slices
names = char('Region 1', 'Region 2', 'Distr. 3', 'Distr. 4');

% These are the numbers to be plotted
data = [1200, 500, 300, 120];
pie(data)

% gtext('string') displays the graph window, puts up a
% cross-hair, and waits for a mouse button or keyboard
% key to be pressed.
for i=1:4
   gtext(names(i,:));
end
title('Sales', 'fontsize', 15
```

Histograms in Matlab

```
y = [1 1 2 2 2 3]
hist(y)
```

```
a = randn(15,1)
hist(a)
```

```
a =
   -0.8468
   -0.2463
    0.6630
   -0.8542
   -1.2013
   -0.1199
   -0.0653
    0.4853
   -0.5955
   -0.1497
   -0.4348
   -0.0793
    1.5352
   -0.6065
   -1.3474
```

Comet Plot

```
% Clears variables, command window, and closes all figures
clc; clear; close all

% Generates 300 linearly spaced points from 0 to 8*pi
x = linspace(0, 8*pi, 300);

% Creates the formula to be plotted
% (it's a multiplication between vector 'x' and vector 'cos(x)')
y = x .* cos(x);

% Plot it!
comet(x, y, .6)
```

stem

```
% Avoid superimposed operations and close previous figs.
clc; clear; close all

% First, we define 51 values of our independent variable
x = 0 : 2*pi/50 : 2*pi;

% Second, we define the function to graph
y = exp(-x/3) .* sin(x);

% Third, we use the 'stem' function to plot discrete values
stem(x,y)
```

```
% We can add title and labels (as strings in arguments)
title('Demonstration of the -stem- function')
xlabel('angle x')
ylabel('f(x)')

x = 0 : 2*pi/20 : 2*pi;
```

logarithmic plot

```
clear; clc; close all

% Define your independent variable
t = 0 : 2*pi/360 : 2*pi;

% Define values along your x-axis
x = exp(t);
% Define values along your y-axis
y = 50 + exp(3*t);

% Plot your function with a wider line and grid the figure
loglog(x, y, 'LineWidth', 2)
grid

% Use a title for the figure
title('Demonstration of logarithmic plots')

% Label your x-axis with a double line.
% Note the special characters
xlabel([{'e^{t}'}; {'0 \leq t \leq 2\pi'}])

% Label your y-axis
ylabel('50 + e^{3t}')
```

Polar Plots

```
t = 1 : 5;
r = t .* exp(i * t * 36 * (pi/180));
compass(r)

t = 0 : 2*pi/100 : 2*pi;
r = 1 - sin(t);
polar(t, r)

t = 0 : 2*pi/100 : 2*pi;
r = sqrt(abs(sin(3*t)));
polar(t,r)

angle_vector = angle(exp(i*randn(1, 500))) + pi/2;
rose(angle_vector)
```

Gaussian distribution

```
a = -100; b = 100;
x = a + (b-a) * rand(1, 500);
m = (a + b)/2;
s = 30;

f = gauss_distribution(x, m, s);
plot(x,f,'.')
grid on
title('Bell Curve')
xlabel('Randomly produced numbers')
ylabel('Gauss Distribution')
```

Simple Animation in Matlab

```
% Define number of frames
nr_fr = 10;
% Initialize matrix using 'moviein'
frames = moviein(nr_fr);

% Generate frames with any plotting function.
% We use a cosine with variable frequency.
t = 0 : .01 : 6;
f = 1;
for i = 1 : nr_fr
   f = f * 1.25; w = 2 * pi * f;
   y = cos(w*t);
   plot(t, y);
   title('Recording movie...')
   % Get every frame with 'getframe' and load the appropriate       % matrix.
   frames(:, i) = getframe;
end

% Save the matrix so that this movie can be loaded later
save frames

% Play the movie once, 2 FPS.
title('Movie being played back...')
movie(frames, 1, 2)
```

Impulse function

```
function y = dd1(n)
% Our default value is 0
y = 0;

% The function is 1 only if the input is 0
if n == 0
   y = 1;
end
```

Let's find the appropriate output for this vector:

```
n = -2 : 2
```

We use our function above (‘dd1’) like this:

```
for i = 1 : length(n)
    f(i) = dd1(n(i));
end
stem(n, f)
axis([-3 3 -.5 1.5])
xlabel('n')
ylabel('Impulse Function')
```

Now, let’s assume another vector:

```
n = [1 2 3 0 2 5 0 1]
```

We can use our function ‘dd1’ to find the delta function output:

```
for i = 1 : length(n)
    f(i) = dd1(n(i));
end
stem(f)
```

```
function y = dd(x)
% x is a vector
% We create an output vector of only 0 (our default value)
y = zeros(1, length(x));

% We find indexes of input values equal to 0,
% and make them 1
y(find(x==0)) = 1;
```

We **don’t need a loop** now, so our process has been simplified a lot.

```
n = [1 2 3 0 2 5 0 1]
f = dd(n)
```

The result is:

```
f = 0     0     0     1     0     0     1     0
```

If we want to calculate $y = 4\delta(n) + 3\delta(n-2)$, in a range of integers that go from -10 to 10, we can do simply this:

```
n = -10 : 10
y = 4*dd(n) + 3*dd(n-2)
```

```
stem(n, y)
xlabel('n')
ylabel('Delta Function')
```

Heaviside Unit Step Function

```
% We iterate from -5 to 5 using only integers
for n = -5 : 5
    y = step_fun(n);
    stem(n, y)
    hold on
end
% We adjust our axis values just to visualize better
axis([-5 5 -1 2])

n = -5 : 5;
y = heaviside(n);
stem(n, y)
axis([-5 5 -1 2]) n = -10 : 10;
y = 4 * heaviside(n) + 3*heaviside(n-2);
stem(n, y)
axis([-15 15 -1 8]
```

Periodic function

```
clear, clc, close all

x = -8 : .01 : 8;
y = sq_w(x);
plot(x,y)
axis([-8 8 -1.5 1.5])
xlabel('x')
ylabel('Square Wave')


clear, clc, close all

x = -6 : .01 : 6;
y = sawtooth_w(x);
plot(x, y)
axis([-6 6 -1.5 1.5])
xlabel('x')
ylabel('Sawtooth Wave')
```

2D drawing

fplot('x.*cos(5*x)',

[0 4*pi])

```
x=0 : .01 : 2*pi;
semilogy(x,exp(2*x))

t = 0 : .01 : pi;
x = exp(2*t);
y = 50+exp(t);
loglog(x,y)
grid

t=0 : .01 : 2*pi;
r=sqrt(abs(sin(3*t)));
polar(t,r)


x = 0 : 10;
y = 2*x+3;
bar(x,y)


x = 1:.1:10;
y1=exp(-2*x).*cos(x);
y2=exp(2*x);
H =plotyy(x,y1,x,y2);
h1=get(H(1),'ylabel');
h2=get(H(2),'ylabel');
set(h1,'string','fun y1');
```

```
set(h2,'string','fun y2');


x = -2*pi:.01:2*pi;
y = sin(x)./x;
area(x,y)

dat = [5 5 20 30 40];
pie(dat)

y = 20*rand(1,10);
hist(y)

t = -3*pi : 3*pi;
y = cos(t)-2*sin(t);
stem(t,y)

x = 1 : 10;
y = 2*x;
stairs(x,y)

t = 0 : pi/6 : pi;
x = cos(t);
y = i*sin(t);
z = x + y;
compass(z)

X = -3 : .1 : 3;
```

```
[x,y]=meshgrid(X,X);

z = -x^2 + 2*x.*y + y^2;

pl=contour(x,y,z);

clabel(pl)
```