BIOE 198: Introduction to Computing with Matlab
Emphasis on Visualizing Data
1.15.2020 mfi

A. Creating 1-D arrays from 1-D functions. Computing and plotting results.

1a. Compute and plot <u>two cycles</u> of a sinewave, $x(t) = A\sin(2\pi u_0 t + \varphi)$ where A = 2 Volts (amplitude), u_0 = 0.2 Hz (temporal frequency), and φ = 0 (phase).

Method:

a. Create a time-axis array (independent variable t). To decide on the time duration and sampling interval for the time axis, do the following. Since u_0 = 0.2 Hz, then $T_0 = 1/u_0$ = 5 s. Two cycles has a total time duration of 10 s. Let's acquire 100 samples over 10 s, so that the sampling interval is 0.1 s.

b. Define parameters (constants).

c. Create voltage signal as $x = f(t)$.

d. Plot results using line and points.

s = square	b = blue (default)
* = star	k = black
+ = plus	r = red
^ = up triangle	g = green (ugh)
> = right triangle	m = magenta
< = left triangle	c = cyan
o = circle	y = yellow

t is an **independent** variable represented as a 1D array

address of *t*	1	2	3	4	5	6	7	8	9	10	11	. . .
value of *t*	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	. . .

x is a **dependent** variable represented is a 1D array

address of *x*	1	2	3	4	5	6	7	8	9	10	11	. . .
value of *x*	0	0.25	0.50	0.74	0.96	1.18	1.37	1.54	1.69	1.81	1.92	. . .

1b. The plot in 1a is smooth, so let's reduce the sampling rate by a factor of 10 and try again. This time label the axes.

2a. Repeat 1a, but now add a second sinewave on top of the first for $\varphi = \pi/2$. That is, first plot $x(t) = A\sin(2\pi u_0 t)$ and then plot $x(t) = A\sin(2\pi u_0 t + \pi/2)$.

2b. <u>(On your own)</u> Compute and plot $y(t) = B\tan(2\pi u_0 t)$ for B = 1, u_0 = 0.2, and $-1.2 \le t \le 1.2$. Label the axes and use the command `grid on` after plotting the curve. What happens if time axis extends to ±1.25? To explore try typing into the command window `tan(2*pi*0.2*1.24)` and `tan(2*pi*0.2*1.25)`. Remember that $\tan x = \sin x / \cos x$. At t = 1.25, we have $2\pi u_0 t = \pi/2$. What happens at $\tan \pi/2$?

3. <u>(On your own)</u> Create and plot $x(t) = t^2$ where $-1 \le t \le 1$. Assume something about axis labels and apply them. I will introduce .* and .^ notation for arrays.

4. From the arrays generated in 3, create new arrays t' (I will use the Matlab variable name `tp`) and x' (use `xp`) that includes only those plot points for which $t \geq 0$. Here we introduce statements like

```
tp = t(t>0);
```

5. (on your own) Repeat 4, but for $t < 0$.

6. (on your own) Repeat 4, but for $-0.5 \leq t \leq 0.5$ by combining the methods from 4 and 5.

7a. Introduction to `for` loops. In this example, notice that $y(t)$ and `y(j)` are very different. ***Make sure you know the difference.***

7b. Introduction to `if, elseif, else` statements. Also, note `disp` output.

7c. Combining `for` loops with `if` statements to select values within an array.

8. (on your own) Do the same task at 6 but now using `for` loops and `if` statements. That is, compute and plot $x(t) = t^2$ where $-1 \leq t \leq 1$. Then use `for` loops and `if` statements to select values of t and x between $-0.5 \leq t \leq 0.5$.

Begin with

```
t=-1:0.1:1;N=length(t);
x=t.^2;
icount=1;
for j=1:N
      …
end
```

********** Handy Commands **********

`close all`	Closes all open plots.
`clear all`	Clears everything from the workspace
`save 'file.mat'`	Saves all variables currently in workspace into `file.mat`
`load 'file.mat'`	Recovers the workspace saved in `file.mat`
`clc`	clears the command window.

****

9. The general method for the six assignments is to (a) define parameters (b) define the independent variable, (c) compute the dependent variable, (d) plot the results, and (e) add labels and titles as requested. For each problem, sample independent variables t or x on interval $T = 0.01$. Also remember when to use `a*b` and `a.*b`. (on your own)

- Create and plot $h(t) = A\ \exp(-at) \quad A = 10;\ a = 0.5;\ 0 \leq t \leq 10$ (9.1)
  Find t_0 at which $h(t_0) = h(1)/2$. Indicate the value of t_0 on the plot and in the plot title. Hint: use `for` loop and `if` statement. To add a title that also indicates a number computed in the script, use the form `title(['t_{1/2} = ',num2str(t(k))])`

- Use results from (9.1) to plot $h'(t) = A\ \exp(-a|t|)$ for same parameters, $-10 \leq t \leq 10$. (9.2)

Notice the absolute value sign! To plot this function, generate a new t variable and then figure out how to use $h(t)$ from (9.1) to generate the new dependent variable $h'(t)$.
The assembly tool is `hp = [h1 h];` See below for a demonstration of array assembly.
Try this: Set `a=[1 2 3 4 5];` and `b=[6 7 8 9 10];`
Then compute: `A=[a b]  B=[a;b]  C=[a' b'] and D=[a';b']` and see the effects.

- From the results of (9.2), find and print $h'(t)$ from $1.1 \le t \le 1.2$. (9.3)
  My code combines (9.2) and (9.3).

- Create and plot $p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$ $\sigma = 1\,;\ -5 \le x \le 5$ (9.4)

- $I = \int_{-\infty}^{\infty} dx\, p(x)$ Add this result to the plot of (9.4); print result on screen (9.5)
  Remember that a simple way to integrate a discrete function is just `I = sum(p)*dx` so long as `dx` is small enough to approximate a differential.
  To integrate $I = \int_{x_1}^{x_2} dx\, p(x)$, we translate from values x_1 and x_2 to their respective addresses `x(j1)` and `x(j2)`, and then use `I = sum(p(j1:j2))*dx`
  Place the code for (9.4) and (9.5) in the same segment.

To print text to plots or to print to the screen:

Title of a plot:
```
title('The integral of p(x) = 3.0')
```
if I know the answer w/o computing it.
Otherwise use
```
title(['The integral of p(x) = ' ,num2str(I)])
```

Print to screen:
```
formatSpec = 'The integral of p(x) is %4.2f \n';
fprintf(formatSpec,I)
%type 'help fprintf' to find list of escape characters line \n
```

Create and plot (9.6)

$$q(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \sin(2\pi u_0 x) \qquad \sigma = 0.17\,;\ u_0 = 5\,;\ -5 \le x \le 5$$

Be sure to label the axes.

B. Creating, plotting, and imaging 2-D functions and their arrays

10. Create and plot (a) $f(x,y) = xy$ for $-1 \le x,y \le 1$ where $\Delta x = \Delta y = 0.1$

(b) $f(x,y) = x^2y^2$ for $-1 \le x,y \le 1$ where $\Delta x = \Delta y = 0.1$

(c) $f(x,y) = \sqrt{x^2+y^2}$ for $-1 \le x,y \le 1$ where $\Delta x = \Delta y = 0.1$

Use `mesh(x,y,f)` to plot $f(x,y)$. Introduce `imagesc`, `surf`, and combinations.

11. Apply the following code to plot a sphere

```
[x,y,z]=sphere; %create a sphere within 3 spatial variables
surf(x,y,z);xlabel('x');ylabel('y')     %let's see it
```

(On your own). The code above generates a unit sphere (r = 1) with N = 20 surfaces, centered at the origin, and rendered using `surf`.

(a) On the same plot, place another sphere centered at (0, 1, -3)

(b) Plot a third sphere centered at (3, -2, 0) with 40 surfaces and rendered using `mesh`.

(c) Plot an ellipsoid centered at (2, -2, -3) with semi-axis lengths (1, 2, 2) and 40 facets using `surf`. Use square axes via `axis square` to be sure the shapes are not distorted.

12. Compute and a 2-D Gabor pulse and plot it in 3D. A Gabor pulse is a Gaussian amplitude-modulated sinewave $Q(x,y)$ from equation below. Assume spatial frequency 0.5 cycles/mm along x-axis. Assume "pulse width" along x is given by σ= 1.2 mm and along y-axis, σ= 1.5. Assume $\Delta x = \Delta y = 0.01$ mm.

$$Q(x,y) = \frac{1}{2\pi\sigma_x\sigma_y}\left[\exp\left(-\left(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2}\right)\right)\right] \sin 2\pi u_0 x \quad \text{for } -5 \le x,y \le 5$$

Because this function is separable, that is, $Q(x,y) = Q(x)Q(y)$ we can compute it two ways. One way is much faster than the other way. I will explain using `tic` and `toc` .

13a. Draw circle of radius a = 2 located at x_0 = 5 and y_0 = 7. See code below.

$$(x-x_0)^2 + (y-y_0)^2 = a^2$$

$$x - x_0 = a\cos(\theta) \quad \text{and} \quad y - y_0 = a\sin(\theta)$$

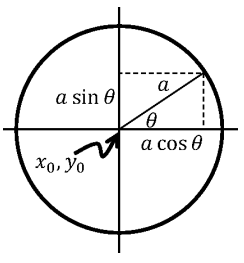

Figure: Diagram of a circle.

```
%
a=2; x0=5; y0=7;                %set the parameters
theta=linspace(0,2*pi);         %same as theta=0:2*pi/99:2*pi;
x=a*cos(theta)+x0;              %(x-x0)=a*cos(theta)
y=a*sin(theta)+y0;
plot(x,y,'linewidth',3);axis equal    %increase linewidth of plot
hold on;plot(x0,y0,'*');text(x0,y0,'\leftarrow origin')
xlabel('x=acos(\theta)');ylabel('y=asin(\theta)') %use of Latex code for theta!
```

13b. Form a 10 x 10 mm plane. Fill a circle of radius 1 mm at (*x,y*) position (3.33, 3.33) with a value different from the surroundings. This is an application of analytic geometry. You do not need to reproduce the math; *the goal is to follow code once you are shown the math so you can repurpose*.

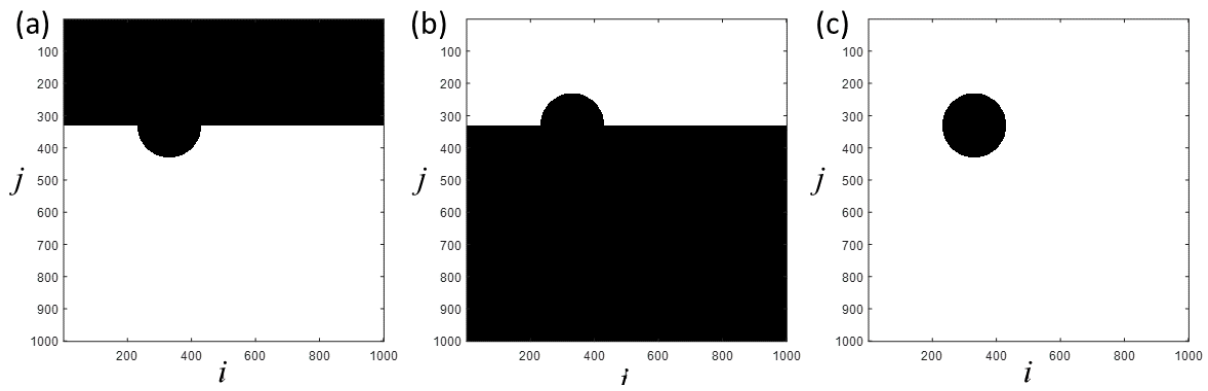

Fig. 1. Illustration of the code for problem 13b. Outside the circle, f(x,y)=1. Inside circle, f(x,y)=0.3.

The equation of a circle is $(x-x_0)^2+(y-y_0)^2=r^2$. The method I choose is to compute the black area in Fig. 1a and Fig. 1b and then `&&` the two solutions. Each is found by solving the equation of a circle for *y* as shown below.

```
dx=0.01;X=10;x=0:dx:X;N=length(x);y=x;f=ones(N); %x(i),y(j) where 1<=i,j<=N
```

$$(y-y_0)^2=r^2-(x-x_0)^2$$

$$y<y_0+\sqrt{r^2-(x-x_0)^2}$$

$$j<\left(\frac{N}{3}\right)+\sqrt{\left(\frac{N}{10}\right)^2-\left(i-\left(\frac{N}{3}\right)\right)^2}$$

Here position values *x,y* are translated to position addresses *i, j*

```
if(j<floor(N/3)+sqrt(floor(N/10)^2-(i-floor(N/3))^2));  %top half
```

The options to obtain an integer are `floor, ceil, round`.

This statement is embedded in `for` loops and `if` statement to give the result in Fig. 1a.

Similarly, we can identify the bottom half…

```
if(j>floor(N/3)-sqrt(floor(N/10)^2-(i-floor(N/3))^2));  %bottom half
```

These are combined using && (logical and) to give,

```
for j=1:N;
    for i=1:N;        %search x and y for locations within the circle
        if(j<floor(N/3)+sqrt(floor(N/10)^2-(i-floor(N/3))^2) ...
                && j>floor(N/3)-sqrt(floor(N/10)^2-(i-floor(N/3))^2));
            f(j,i)=f(j,i)*0.3;          %attenuate the circle region
        end;
    end;
end;
```

(on your own) Your job is to follow what was done above and adapt the code for similar tasks.

In the example above, the background plane has size 10mm x 10mm and amplitude value 1.0. In the circle of radius 1.0 mm at coordinates (33.3 mm, 33.3 mm) the area amplitude is 0.3. In the following, I will give you coordinates, radii and amplitudes and ask you to add two new circles to the plane. Then display all three.

(a) Circle center coordinates (66.7 mm, 33.3 mm), radius 2.0 mm, and amplitude 0.8

(b) Circle center coordinates (50 mm x 66.7 mm), radius 0.5 mm, and amplitude 1.5

14. 1-D and 2-D histograms. Simple two-group discrimination.

(a) Two groups of 1-D data.

Begin by generating two groups of data x_1 and x_2 using code for random numbers:

```
N=500;m1=2;s1=1;m2=5;s2=2;                  %set the parameters
x1=s1*randn(N,1)+m1;x2=s2*randn(N,1)+m2;    %generate two groups of random data
```

I will explain probability later in the course. 500 data points per group.
The first group has mean `m1=5` and standard deviation `s1=2`.
The second group has mean `m2=2` and standard deviation `s2=1`. The two lines of code give two 1-D arrays, each with 500 points. We can plot the points and histogram them using the following:

```
subplot(4,1,1);plot(x1,'bs');hold on;plot(x2,'bs');hold off;xlabel('t');ylabel('x(t)')
subplot(4,1,2);histogram([x1 x2]);xlabel('x');ylabel('h(x)')
subplot(4,1,3);plot(x1,'bo');hold on;plot(x2,'rx');hold off;xlabel('t');ylabel('x(t)')
subplot(4,1,4);histfit(x1);hold on;histfit(x2);hold off;xlabel('x');ylabel('h(x)')
title('(mean,std): (2,1) and (5,2)')
```

The `subplot` command is new! The top two plots show the data are not easy to separate if we don't know anything about the two groups. Since we know group properties, the bottom two plots show how much they overlap. The bottom plots are called training data while the top plots are test data.

(b) The two groups of data overlap each other a small amount, so we must have classification errors regardless of which threshold we choose. Let's set a histogram threshold of 3 (see arrow in plot from part (a)). Any points from $x_1 > 3$ will be misclassified. Similarly, values of $x_2 < 3$ are misclassified. Use the training data to estimate the percent errors for both groups when the decision threshold is 3. I will help you do this in class. The simplest method is to use `for` loops and `if` statements. Print out the percent errors to the screen.

(c) Histogram 2-D data sets. Use a normal pdf to generate the first group of data, $A_1 = (x_1, y_1)$, and the second group, $A_2 = (x_2, y_2)$. They do not need to be placed in 2-D array. Apply methods established in parts (a) and (b). Select 10,000 samples for each group using the following mean & standard deviations:

```
N=10000;                                    %set number of samples
mx1=-3;sx1=2.5;my1=0;sy1=3;                 %set parameters for bivariate x1,y1
x1=sx1*randn(N,1)+mx1;y1=sy1*randn(N,1)+my1;  %generate x1,y1 data
mx2=5;sx2=2;my2=0;sy2=3;                    %set parameters for bivariate x2,y2
x2=sx2*randn(N,1)+mx2;y2=sy2*randn(N,1)+my2;  %generate x2,y2 data
```

Group 1 is compose of ordered pairs (x_1, y_1). Group 2 is compose of ordered pairs (x_2, y_2).

- Figure out how to apply `histogram2` to histogram the two 2-D data sets
- Compute percent errors, as in (b) above by selecting a threshold (now a line)
- The sum of the errors in A_1 and A_2 is the total percent error
- Find the threshold line for `th` in the range $-1 \leq x \leq 3.5$ that gives minimum total percent error

```
% Below are a set of scripts for beginning Matlab users

% 1.a  Simple functions: plot two cycles of a sinewave, x(t)=Asin(2*pi*u0*t+phi)
A=2;u0=0.2;phi=0;               %amplitude, frequency, phase of a sinewave
t=0:0.1:10;                     %10 s of data acquired every 100 ms.
x=A*sin(2*pi*u0*t + phi);       %Generate the sinewave
plot(t,x,'o-')                  %plot results
%

%%
% 1.b Reduce the sampling rate and try again
A=2;u0=0.2;phi=0;               %amplitude, frequency, phase of a sinewave
t=0:1:10;                       %reduce the sampling rate a factor of 10
x=A*sin(2*pi*u0*t + phi);
plot(t,x,'o-')
xlabel('time (s)');ylabel('Amplitude (V)')
%

%%
% 2a. Repeat 1a but now add to the plot the same function at phi=pi/2.
A=2;u0=0.2;phi1=0;phi2=pi/2;    %amplitude, frequency, phase of a sinewave
t=0:0.1:10;                     %10 seconds of data acquired every 100ms.
x=A*sin(2*pi*u0*t + phi1);      %sinewave for phi1
plot(t,x,'o-');hold on          %plot result
x=A*sin(2*pi*u0*t + phi2);      %sinewave for phi2
plot(t,x,'ro-');hold off        %plot result
xlabel('time (s)');ylabel('Amplitude (V)')
%

%%
%2b.  Compute and plot tangent.
B=1;u0=0.2;
t=-1.2:0.01:1.2;
x=tan(2*pi*u0*t);
plot(t,x);grid on;xlabel('time (s)');ylabel('tangent')
%%

%3. Create and plot x(t) = t^2 from -1 <= t <= 1 at time internal 0.1s;
% unitless amplitude.  Plot line and points and label axes.  Verify x(t=-0.8)
t=-1:0.1:1;     %time axis
x=t.^2;
plot(t,x,'o-')
xlabel('time (s)');ylabel('Amplitude')
%%

%4. Generate from t=-1 to t=1 but plot points from t=0 to t=1 only.
% This assignment emphasizes array addresses as distinct from array values.
%
t=-1:0.1:1;     %indepndent variable array
x=t.^2;         %dependent variable array
tp=t(t>=0);     %select time-array addresses where t values exceed zero; place in tp
xp=x(t>=0);     %select the same dependent variable addresses, place in xp
plot(tp,xp,'x-')%plot the reduced range values
%
%%

%5. Select values of t,x for which t<0 in tp, xp and plot.
%
t=-1:0.1:1;
x=t.^2;
tp=t(t<0);
xp=x(t<0);
plot(tp,xp,'d-','linewidth',2)    %use diamonds for points and increased linewidth
```

```
xlabel('time (s)');ylabel('Amplitude')
title('I used diamond points and increased the linewidth')  %introduced title
%

%%
%6. Plot points in #3 between -0.5 and 0.5
%
t=-1:0.1:1;
x=t.^2;
tp=t(t>=-0.5);tpp=tp(tp<=0.5);
xp=x(t>=-0.5);xpp=xp(tp<=0.5);
plot(tpp,xpp,'rv-')    %use red downward triangles for points
xlabel('time (s)');ylabel('Amplitude')
%%

%7a. Introduction to 'for' loop.
%Here, a for loop is used to do the same thing to each element of an array
%
t=-10:10;N=length(t);       %shift function y(t)=t up by 10.
for j=1:N                   %begin for loop: N is npts in t
    y(j) = t(j)+10;         %shift each element of y by 10
end                         %end for loop
plot(t,y,'sk');xlabel('input');ylabel('output')
%

%%
%7b. Introduction to 'if' statements and 'disp' output
%
x=4; minx=2; maxx=6;
if x >= minx && x <= maxx
    disp('Value within specified range.')
elseif x > maxx
    disp('Value exceeds maximum value.')
else
    disp('Value is below minimum value.')
end
%

%%
%7c. Combining for loops with if statements to find the addresses
% for 5 < t < 10. Need to run "clear all"
%
clear all                   %Notice that Matlab warns about starting this way
t=20:-1:10;N=length(t);     %I generated a time axis backwards
icount=1;                   %see a diagram of the array values and addresses
for j=1:N                   %for the N addresses in t, do the following...
    if t(j) < 15 && t(j) > 11 %do time values fall in desired range?
        tp(icount)=j;       %If yes, then copy to tp and
        icount=icount+1;    %index the counter
    end                     % end the if
end                         %end the for
tp                          %Lazy-person's method of typing to screen
%

%%
%8. Plot points in #3 between -0.5 and 0.5 another way
%
t=-1:0.1:1;N=length(t);                 %time axis and its length
x=t.^2;                                 %generate dependent variable
icount=1;                               %initialize counter
for j=1:N                               %combine 'for' with 'if' to select
    if t(j) >= -0.5 && t(j) <= 0.5
        tpp(icount)=t(j);xpp(icount)=x(j);
```

```
        icount=icount+1;
    end
end
plot(tpp,xpp,'g^-')    %used green upward triangles (terrible color) for points
xlabel('time (s)');ylabel('Amplitude')
%

%%
%9.  There are six parts:
% 9.1 Plot h(t) = exp(-at) for A=10, a=0.5, and 0 <= t <= 10. Find t1/2 and print as
title
% 9.2 Use these results to plot hp(t) = exp(-a|t|) for -10 <= t <= 10.
% 9.3 Find of print out hp(t) values for 1.1 <= t <= 1.2
%
% 9.1
%
A=10;a=0.5;                              %Set parameters
t=0:0.01:10;N=length(t);                 %Independent variable and its length
h=A*exp(-a*t);                           %Dependent variable
plot(t,h);xlabel('t');ylabel('A*h(t)')   %plot and label axes
for j=1:N                                %Find the half-ampl value
    if h(j) >= h(1)/2
        k=j;
    end
end
title(['t_{1/2} = ',num2str(t(k))])      %List value in the title
hold on;plot(t(k),h(k),'r*');hold off    %and as a point in the plot
%

%%
% 9.2  Be sure to run the prior script to generate h needed here!
%
tp=-10:0.01:10;
hp=[h(end:-1:2) h];  %can use hp=[fliplr(h) h(2:end)];plot(t,hp)
                     %In either case, we need eliminate one of t=0 pts.
figure;plot(tp,hp);xlabel('t');ylabel('A*h(t)')
%
% 9.3  Find values of hp within the range 1.1<tp<1.2
%
tpp=tp(tp>=1.1);tppp=tpp(tpp<=1.2)
hpp=hp(tp>=1.1);hppp=hpp(tpp<=1.2)
%Notice in the results for 9.3 that the value at tp=1.1 is not captured.
%Likely a round off problem.  So the initial time could be changed
%from 1.1 to 1.09.
%

%%
%9.4 Compute and plot a Gaussian function with s = 1 from -5 to 5.
%
s=1;dx=0.01;X1=-5;X2=5;x=X1:dx:X2;  %parameters & independent variable array
p=exp(-x.^2/(2*s^2))/(s*sqrt(2*pi)); %dependent variable: not use of .^
plot(x,p);xlabel('x');ylabel('p(x)') %plot and label axes
%
%9.5 Integrate p(x)dx by summing values computed.
%
I = sum(p)*dx;                       %integrate p over all available values
formatSpec = 'The integral of p(x) is %4.2f \n';
fprintf(formatSpec,I)
title(['The integral of p(x) = ' ,num2str(I)]) %tell us value of I in plot title

%
%%
%9.6 Compute a Gabor pulse for s = 0.17 u0=5 over -5 <= x <= 5
```

```
%
s=0.17;u0=5;dx=0.01;X1=-5;X2=5;x=X1:dx:X2;%set up independent variable array and
parameters
q=exp(-x.^2/(2*s^2))/(s*sqrt(2*pi)).*sin(2*pi*u0*x);
plot(x,q);xlabel('x');ylabel('q(x)')
%

%%
%
% Two-dimensional arrays and 3-D plotting.
%
% 10.  Simple 2-D functions plotted in 3-D, f(x,y)
%
x=-1:0.1:1;y=x;N=length(x);f=zeros(N);  %Set parameters, independent variable array x
for j=1:N                               %Also create & zero  NxN dependent variable
array f
    for i=1:N                           %Two 'for loops' for filling f
       f(j,i)=x(i)*y(j);               %Three possible 2D functions
%       f(j,i)=x(i)^2*y(j)^2;
%       f(j,i)=sqrt(x(i)^2+y(j)^2);
    end
end
mesh(x,y,f)                             %visualize the 2D variable f
xlabel('x');ylabel('y');zlabel('f(x,y)')%label axes
figure;imagesc(f),colormap gray         %grayscale image (use 'figure' to not
overwrite)
figure;surf(f);hold on;imagesc(f)       %surf combined with imagesc
%

%%
% 11. Using Matlab-generated objects: Plot a sphere centered at the origin.
% Then add two more spheres located at (3,-2,0) and (0,1,-3) and an ellipsoid
% at (2,-2,-3) with semi-axis lengths (1,2,2) and 40 facets. Use: axis
% square.  What does surfl do?  A. light from angle instead of from the top.
%
[x,y,z]=sphere;                             %create a sphere within 3 spatial
variables
surf(x,y,z);xlabel('x');ylabel('y');axis square     %let's see it
%  From here on the students need to finish the code
% hold on                                   %plot on the same figure window
% surf(x,y+1,z-3)                           %sphere centered at (0,1,-3)
% [x,y,z]=sphere(40);
% mesh(x+3,y-2,z)                           %sphere centered at (3,-2,0)
% [x,y,z] = ellipsoid(2,-2,-3,1,2,2,40);    %ellipsoid centered at (2,-2,-3) with
semi-axis ...
% surf(x,y,z);axis square                   %... lengths (1,2,2) and 40 facets
%

%%
% 12.  Gabor pulse.  At frequency 0.5 MHz and increased the 1D pulse length
%  to fill the space.  Then I made a 2-D Gabor pulse two different ways.
%  In both cases the pulse is wider (in y) than long (in x).
%
%   Method 1
%
sx=1.3;sy=1.5;u0=0.5;dx=0.01;X1=-5;X2=5;x=X1:dx:X2;y=x;
%previous line sets up independent variable array and parameters
N=length(x);Q=zeros(N);     %establish the 2D space for the pulse
tic                         %start the stopwatch
for j=1:N                   %create 2D array as generally as possible
    for i=1:N
        Q(j,i)=exp(-x(i)^2/(2*sx^2))*sin(2*pi*u0*x(i)) ...
            *exp(-y(j)^2/(2*sy^2))/(2*pi*sx*sy);
```

```
    end
end
toc                          %stop the stopwatch
mesh(x,y,Q)                  %let's see it
xlabel('x');ylabel('y');zlabel('Q(x,y)')
%

%%
%   Method 2
%
sx=1.3;sy=1.5;u0=0.5;dx=0.01;X1=-5;X2=5;x=X1:dx:X2;y=x;N=length(x);
%previous line sets up independent variable array and parameters
q=exp(-x.^2/(2*sx^2))/(sx*sqrt(2*pi)).*sin(2*pi*u0*x);  %create pulse along x-axis
%plot(x,q);xlabel('x');ylabel('q(x)') %In case you want to see the x-axis result
Q=zeros(N);                                             %create and zero dependent
variable
tic                                                     %start the stopwatch
for j=1:N       %Because the function is separable, we can save time with this method
    Q(j,:)=q*exp(-y(j)^2/(2*sy^2))/(sy*sqrt(2*pi));
end
toc                                 % Method 2 is ~33x faster than Meth 1 depending on
dx,dy
figure;mesh(x,y,Q)                  % View the result
xlabel('x');ylabel('y');zlabel('Q(x,y)')
Q1=downsample(Q,50);                    %This retrospectively reduces dx, and yet...
figure;surf(Q1+0.1);hold on;imagesc(Q1) %...the alternative plot doesn't work too well
%

%%
% 13a. Draw a circle of a=2 located at x0=5 and y0=7.
%
a=2; x0=5; y0=7;                %set the parameters
theta=linspace(0,2*pi);         %same as theta=0:2*pi/99:2*pi;
x=a*cos(theta)+x0;              %(x-x0)=a*cos(theta)
y=a*sin(theta)+y0;
plot(x,y,'linewidth',3);axis equal    %this example increases linewidth of plot
hold on;plot(x0,y0,'*');text(x0,y0,'\leftarrow (x_0,y_0)')
xlabel('x=acos(\theta)');ylabel('y=asin(\theta)') %note use of Latex code for theta!
title(['Parameters: a=2, x_0=5, y_0=7'])    %title to specify parameters used
%

%%
% 13b. Create a 10x10mm plane and place a circle of radius 1mm at (3.33,3.33)
%
dx=0.01;X=10;x=0:dx:X;N=length(x);y=x; %set parameters, independent variables x,y
f=ones(N);                          %create and zero dependent variable f(x,y)
for j=1:N;
    for i=1:N;                      %search x and y for locations within the circle
%        if(j<floor(N/3)+sqrt(floor(N/10)^2-(i-floor(N/3))^2));  %top half
%        if(j>floor(N/3)-sqrt(floor(N/10)^2-(i-floor(N/3))^2));  %bottom half
        if(j<floor(N/3)+sqrt(floor(N/10)^2-(i-floor(N/3))^2) ... %both halves
                && j>floor(N/3)-sqrt(floor(N/10)^2-(i-floor(N/3))^2));
            f(j,i)=f(j,i)*0.3;                  %attenuate the circle region
        end;
    end;
end;
mesh(f);colormap autumn %other colormaps: flag lines jet cool gray bone
figure;imagesc(f);colormap gray;axis square % Notice that origins of two figures are
different
xlabel('i');ylabel('j')
%
%This is the assignment:  Place the following additional circles.
%
```

```
% for j=1:N;
%     for i=1:N;                      %search x and y for locations in the circles
%         if(j<floor(N/3)+sqrt(floor(N/5)^2-(i-floor(2*N/3))^2) ... %Circle (a)
%                 && j>floor(N/3)-sqrt(floor(N/5)^2-(i-floor(2*N/3))^2));
%             f(j,i)=f(j,i)*0.8;                %attenuate the circle region
%         end;
%                 if(j<floor(2*N/3)+sqrt(floor(N/20)^2-(i-floor(N/2))^2) ... %Circle
(b)
%                 && j>floor(2*N/3)-sqrt(floor(N/20)^2-(i-floor(N/2))^2));
%             f(j,i)=f(j,i)*1.5;                %amplify the circle region
%         end;
%     end;
% end;
% mesh(f);colormap autumn %other colormaps: flag lines jet cool gray bone
% figure;imagesc(f);colormap gray;axis square
% xlabel('i');ylabel('j')
%

%%
% 14. Histogramming 1-D and 2-D data.
%
%(a) We will use pdfs from probability to generate data without discussing pdfs.
%Probability os a topic discussed later in the course.
%
N=500;m1=2;s1=1;m2=5;s2=2;                  %set the parameters
x1=s1*randn(N,1)+m1;x2=s2*randn(N,1)+m2;    %generate two groups of random data
subplot(4,1,1);plot(x1,'bs');hold on;plot(x2,'bs');hold off;xlabel('t');ylabel('x(t)')
subplot(4,1,2);histogram([x1 x2]);xlabel('x');ylabel('h(x)')
subplot(4,1,3);plot(x1,'bo');hold on;plot(x2,'rx');hold off;xlabel('t');ylabel('x(t)')
subplot(4,1,4);histfit(x1);hold on;histfit(x2);hold off;xlabel('x');ylabel('h(x)')
title('(mean,std): (2,1) and (5,2)')
text(3,79,'\downarrow') %Here, I am marking the decision threshold in the next section
%

%%
%(b) Let's set the separation threshold at x=3.0. Now let's see if we can
%count the number of errors.
%
th=3.0;k1=0;k2=0;   %decision threshold set to th, set counters k1 and k2
for j=1:N           %let's go through all the data
    if x1(j) > th
        k1=k1+1;
    end
    if x2(j) < th
        k2=k2+1;
    end
end
formatSpec1 = 'The percent error for x1 is %4.2f \n';
fprintf(formatSpec1,100*k1/N)
formatSpec1 = 'The percent error for x2 is %4.2f \n';
fprintf(formatSpec1,100*k2/N)
%

%%
%(c) This section is the assignment for histogramming bivariate data.
%
N=10000;                                    %set number of samples
mx1=-3;sx1=2.5;my1=0;sy1=3;                    %set parameters for bivariate x1,y1
x1=sx1*randn(N,1)+mx1;y1=sy1*randn(N,1)+my1;    %generate x1,y1 data
mx2=5;sx2=2;my2=0;sy2=3;                    %set parameters for bivariate x2,y2
x2=sx2*randn(N,1)+mx2;y2=sy2*randn(N,1)+my2;    %generate x2,y2 data
histogram2(x1,y1);hold on;histogram2(x2,y2);hold off
xlabel('x');ylabel('y')
```

```
%
% I can tell from the distribution functions and from the plots that a line
% parralel to the y-axis is a good threshold line for making decisions. I'm not
% sure which value of x to select, so let's try a few between x=-1 and x=3.5.
%
th=zeros(1,10);k1=th;k2=th;  %initialize threshold and counters
for i = 1:10
    th(i)=(i-3)*0.5;         %set variable threshold to th(i)
    for j=1:N                %go through all the x1,x2 data
        if x1(j) > th(i)
           k1(i)=k1(i)+1;
        end
        if x2(j) < th(i)
           k2(i)=k2(i)+1;
        end
    end
end
er=100*(k1/N+k2/N);[mine,I]=min(er);
figure;plot(th,er,'k*-')
xlabel('x-axis threshold value');ylabel('Total Percent Error')
title(['The minimum total percent error is ',num2str(mine),' at x=',num2str(th(I))])
```