# Matlab Fun

## Contents



## What? Matlab can be fun

This is an example using the Matlab publishing feature. You may use this feature when turning in any homework that has a computational aspect.

Here we will go over some basic commands in Matlab. Again if you are unfamiliar with Matlab, I highly recommend searching for tutorials online or making an appointment with me. Furthermore, I always recommend that you use the Matlab "help" command or go to help>Product Help and keep the help window open where you can look up the uses and definitions of all Matlab internal functions.

First we will play around with the plotting command.

## Plot the Sine Function

To plot the sine function, we will use the command plot(x,y) where x and y are vectors.

First, we define a vector of points where we will evaluate the sine function

```
x=0:.01:2*pi;
```

This creates a vector of points that starts at 0 and continues in increments of .01 until it reaches the value 2*pi. This is equivalent to using a for loop in the following way

```
dx=.01; N=floor(2*pi/dx)+1; x=zeros(N,1);
for j=2:N
    x(j)=(j-1)*dx;
end
```

Where dx is the increment, N is the total number of points from 0 to 2*pi in increments of dx, and the zeros(N,1) command creates an Nx1 vector of zeros that we fill in with the subsequent for loop.

Note that in Matlab, any command that ends with a ; is suppresed

```
r=1;
```

Not including the ; will cause Matlab to output the command

```
r=1
```

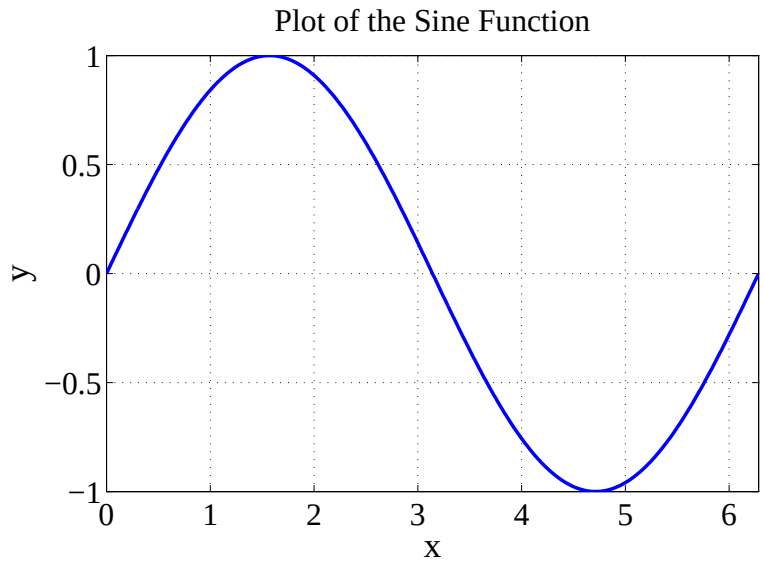
```
r =

     1
```

We can evaluate the sine function at the values in the vector x by

```
y=sin(x);
```

Now, we plot sine function

```
plot(x,sin(x),'linewidth',2)
axis([0 2*pi -1 1]) % Set the plot to display x values from 0 to 2*pi and
                    % y values from -1 to 1
grid % plots a grid in the background (not always necessary but can look cool)
xlabel('x','Fontsize',20)
ylabel('y','Fontsize',20)
title('Plot of the Sine Function') % Title the plot
```




Note that you should always label your axes using the xlabel and ylabel commands.

## Plot something more complicated

Lets plot

$$y = e^x \frac{\sin(x)}{x}$$

Notice that you can also display "nice" equations in publish mode. The way to do so is to use LaTex commands. For example, the above equation was generated by

"$$ y=e^x \frac{\sin(x)} {x} $$"

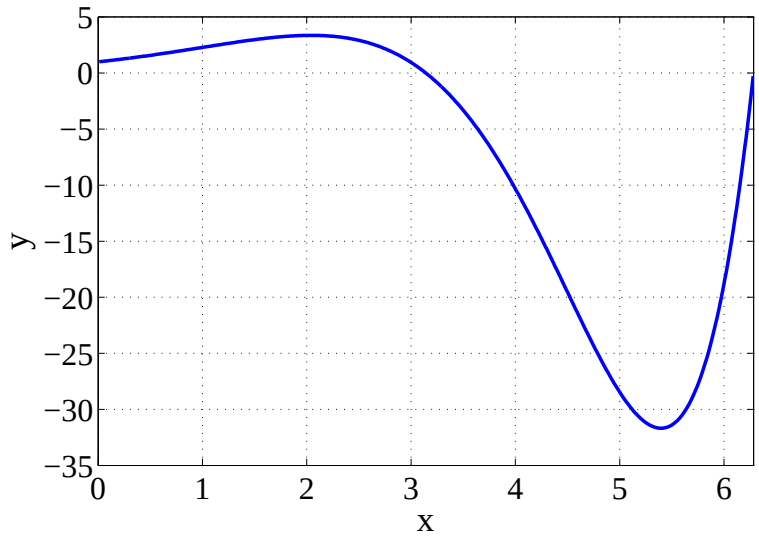
For more on LateX commands see the internet.

The above function can be plotted in Matlab by

```
x=[0:.01:2*pi];

y=exp(x).*sin(x)./x;
```

Note that when doing pointwise operations with vectors, use the . before the operation, i.e. poinwise multiplication of two vectors z and w is z.*w

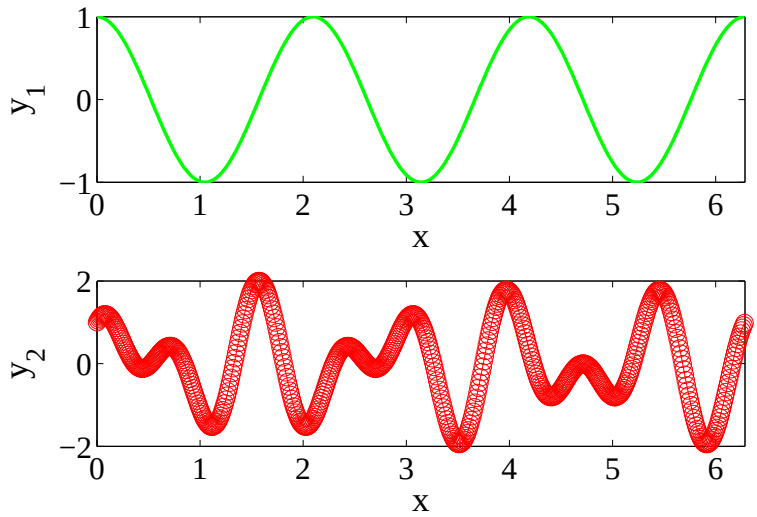
```
plot(x,y,'linewidth',2)
axis([0 2*pi -35 5])
grid
xlabel('x','Fontsize',20)
ylabel('y','Fontsize',20)
```




## Subplots and Legends

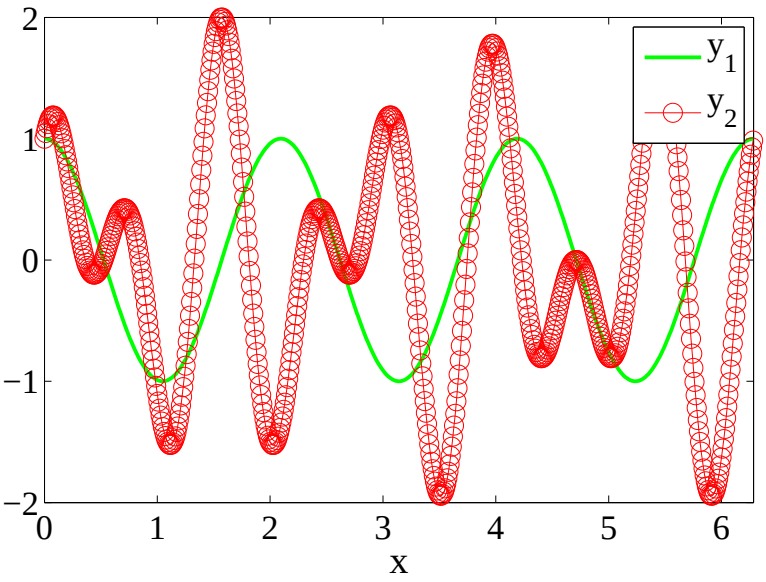
To save space (and trees) Matlab also has a subplot command that allows you to display multiple plots in one window. For example

```
x=0:.01:2*pi;
y1=cos(3*x);
y2=sin(5*x)+cos(8*x);
subplot(2,1,1)  % subplot(n,m,p) means you want a grid of nxm plots and p
                % is the number of the plot
plot(x,y1,'linewidth',2,'color','g')
axis([0 2*pi -1 1])
xlabel('x','Fontsize',20)
ylabel('y_1','Fontsize',20)
subplot(2,1,2)
plot(x,y2,'marker','o','markersize',10,'color','r')
axis([0 2*pi -2 2])
xlabel('x','Fontsize',20)
ylabel('y_2','Fontsize',20)
```




You can also plot two traces on the same window using the "hold on" command and use different colors or linestyles and use the "legend" command to let the viewer know what they are seeing.

```
figure
plot(x,y1,'linewidth',2,'color','g')
hold on
plot(x,y2,'marker','o','markersize',10,'color','r')
axis([0 2*pi -2 2])
xlabel('x','Fontsize',20)
legend('y_1','y_2')
```




## Matrix Operations

An important aspect of Matlab is the ability to perform matrix computations.

For example, let

$$A = \begin{pmatrix} 2 & 4 & 6 \\ 8 & 3 & 4 \\ 3 & 2 & 1 \end{pmatrix}$$

And

$$\vec{b} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

.

We can compute the product

$$\vec{y} = A\vec{b}$$

in Matlab by

```
A=[2 4 6; 8 3 4; 3 2 1];
b=[1;0;0];
y=A*b
```

```
y =

     2
     8
     3
```

Suppose instead that we know

$$\vec{y} = \begin{pmatrix} 2 \\ 8 \\ 3 \end{pmatrix}$$

and

$$\vec{b}$$

is an unknown that we want to solve for.

In Matlab

$$\vec{b} = A^{-1}\vec{y}$$

can be implemented two ways.

Note that to there seems to be a bug in the publish mode in getting a '-' symbol to show up in the .pdf generated by the publish mode. I had to use the hack "\ ^_ \ " in place of the "-" in the above equation, so that the code for the above equation is

"$$ \vec{b} = A^{\ ^\_ \ 1} \vec{y} $$"

This seems to work.

We solve the above matrix equation first, by using the \, or right matrix divide command

```
y=[2;8;3];
b=A\y
```

```
b =

     1
     0
     0
```

Or, by using the inv() command

```
b=inv(A)*y
```

```
b =

     1
     0
     0
```

## For Loops, Iterative Equations, and the Logistic Map

As a last example, we will look at simulating the logisitic map

$$x_n = x_{n-1} r(1 - x_{n-1})$$

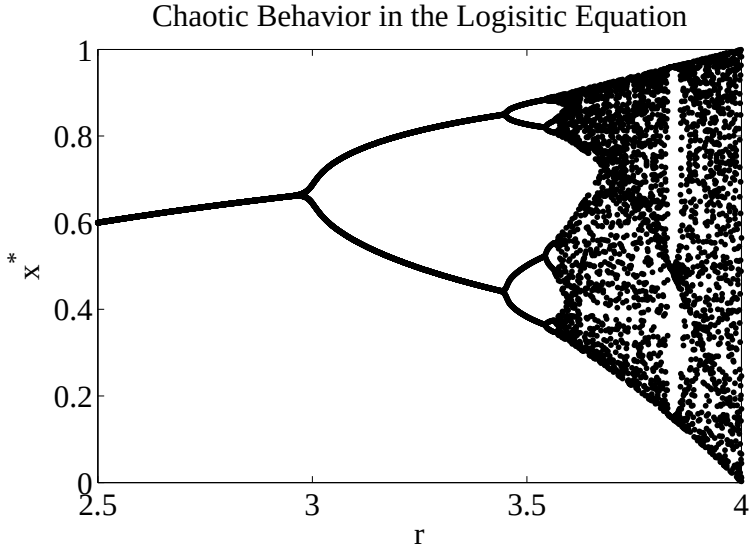
where r is a parameter. This is a simple recurrence relationship where the value of x at the current timestep is completely determined by its value one step prior. However, very interesting behavior can arise in this system as the parameter r is varied. In particular, the fixed points of the map, i.e. the points where

$$x^* = x^* r(1 - x^*)$$

undergo a series of changes known as bifurcations until the system begins to exhibit chaotic behavior. The following code shows how to simulate the map and to visualize the chaotic behavior as the parameter r is varied.

```
r=2.5:.001:4;
Nr=length(r);
Nx=120;
fp=zeros(Nr,8);
xp=0:.01:1;
% figure(1)
% hFig = figure(1);
% set(hFig, 'Position', [100, 200, 700, 500]);
for oi=1:Nr
    x=zeros(Nx,1);
    x(1)=.2;
    for ii=2:Nx
        x(ii)=x(ii-1)*r(oi)*(1-x(ii-1));
%         if mod(oi,10)==0
%             subplot(1,2,2)
%             plot([x(ii-1),x(ii-1)],[x(ii-1),x(ii)],'r')
%             hold on
%             plot([x(ii-1),x(ii)],[x(ii),x(ii)],'r')
%         end
    end
    for k=1:8
        fp(oi,k)=x(end-(k-1));
    end
%     if mod(oi,10)==0
%         str=['r value is ' num2str(r(oi))];
%         subplot(1,2,1)
%         plot(x,'.')
%         xlabel('iteration (n)')
%         ylabel('x_{n}')
%         title(str)
%         axis([0 Nx 0 1])
%         subplot(1,2,2)
%         plot(xp,r(oi)*xp.*(1-xp),'k','linewidth',3)
%         plot(fp(oi,:),fp(oi,:),'r.','markersize',20)
%         plot(xp,xp,'k--')
%         axis([0 1 0 1])
%         xlabel('x_{n}')
%         ylabel('x_{n+1}')
%         hold off
%         pause(.01)
%         clf
%     end
end

figure
plot(r,fp(:,1),'k.','markersize',10);
```

```
hold on
for k=2:8
    plot(r,fp(:,k),'k.','markersize',10);
end
axis([r(1) r(end) 0 1])
xlabel('r')
ylabel('x^*')
title('Chaotic Behavior in the Logisitic Equation')
```




The figure shows how the fixed points of the system change as r is varied. Notice that the fixed points undergo a series of changes until they begin to vary wildly. That is, for r>3.6, the system behaves chaotically, meaing that, even though the system is completely deterministic (in that the dynamics are completely described by the above equation in a predictable manner) the system can display unpredictable behavior. For more information, see the Wikipedia page on the logistic map. To see this chaotic behavior in real time, remove the percent signs in the above code.

Use this code to familiarize yourself with using for loops, plotting functions, and other commands in Matlab. If there are commands in the code that are unfamiliar to use, please use the Matlab help command to find out more about the function.

At this point you should be a complete master of Matlab, but even masters need help from time to time so do make use of the "help" command whenever necessary.

# 1 Matlab Code

The code in Matlab used to generate the above document is contained in the file

`Publish_ex.m`

. Open the file in Matlab and go to

`file>Publish Configuration for filename.m>Edit Publish Configurations for filename.m`

and you can have it publish to a .pdf file or generate the LaTeX code which you can then compile using LaTeX as I did here.