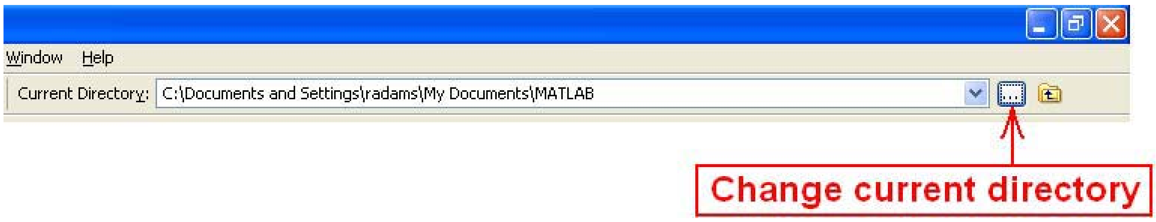
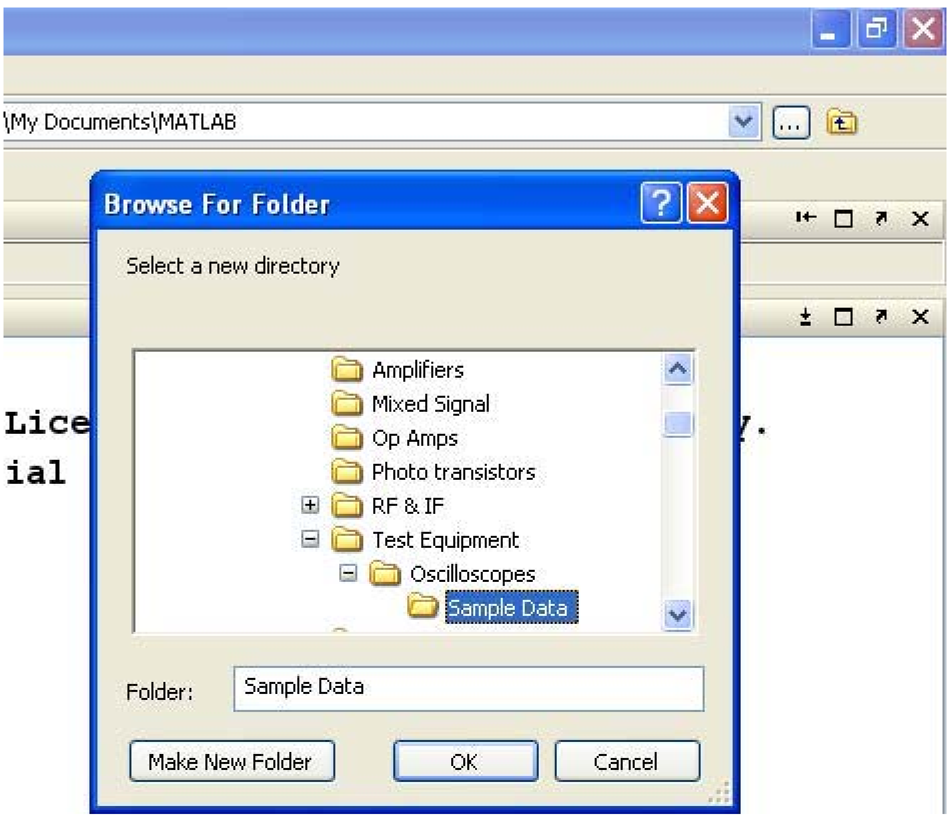
How to import the X-Y data to MATLAB

1. Open MATLAB.

2. Click on the ... button to change the current directory:



3. This will bring up the **Browse For Folder** window. Navigate to the location of the file containing the Oscilloscope XY data, then click OK



4. In the MATLAB workspace, type **dir**, then hit the ↵Enter key. This will provide a list of files in the current directory, as shown below:

```
>> dir

.
..
I&Q Modulator at 1.8 Mbps with BPF.csv
I&Q Modulator at 1.8 Mbps.csv
I&Q Modulator at 1.8 Mbps.xls
Screen Shot - I&Q Modulator at 1.8 Mbps with BPF.png
Screen Shot - I&Q Modulator at 1.8 Mbps.png
Thumbs.db


>>
```

The **xlsread** command is used to access the data in the Excel file. The format of the command is

```
XYdata = xlsread('file name',Worksheet,Range)
```

where **'file name'** is the name of the file including the extension. Both .xls and .csv files are allowable.

Worksheet is the worksheet number. This will be 1 for XY data imported from the oscilloscope.

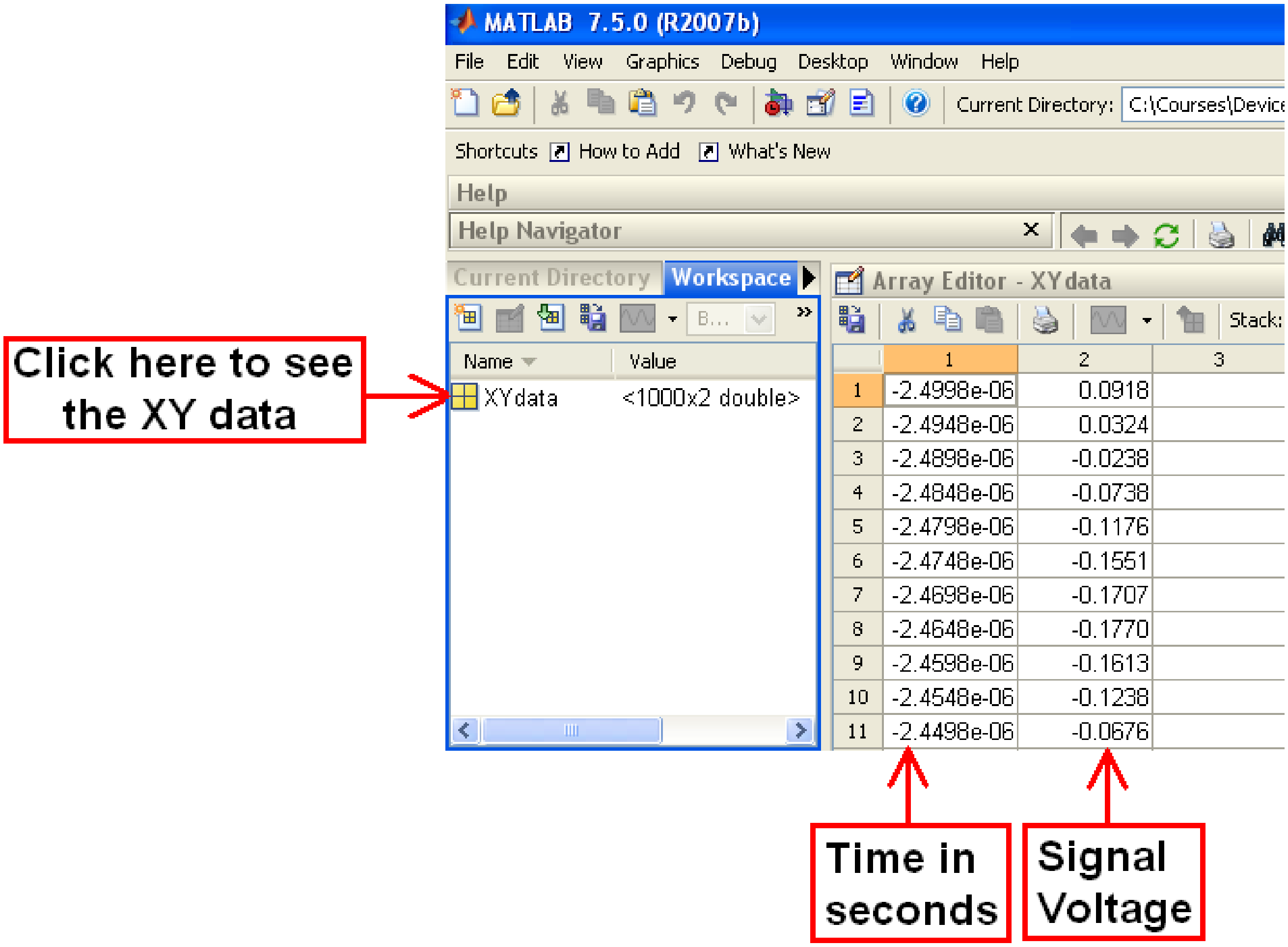
Range is the cell range to gather the data. . This will be 'a3:b1002' for XY data imported from the oscilloscope.

So for example the command:

```
XYdata=xlsread('I&Q Modulator at 1.8 Mbps.csv', 1, 'a3:b2005');
```

will read XY data from the .csv file named **I&Q Modulator at 1.8 Mbps**.

The data is all in one matrix. If you click on XYdata in the Workspace frame, you will see the values: ‘**XYdata**’ you will see the values:



Notice that the first column contains the time in seconds and the second column contains the signal voltage. The following command will select the first column of data and store the data into a variable called **time**.

```
time = XYdata(:, 1);
```

The following command will select the second column of data and store the data into a variable called **voltage**.

```
voltage = XYdata(:, 2);
```

Notice that the colon operator (**:**) is used here to select all rows.

To plot the oscilloscope data, you could enter the commands:

```
figure;
plot(time, voltage, 'r',  'linewidth', 2);
grid;
```

The **figure** command tells MATLAB to generate a new figure. The plot option **'linewidth'** followed by the number 2, indicates to plot the line with a medium thickness. To make the line thinner use 1, to make the line thicker use 3 or more. The plot option **'r'** tells MATLAB to use the color red in generating he plot. For more plot options, type **help plot** in the command window. The **grid** command forces MATLAB to display grid lines.

How to generate an amplitude spectrum plot in MATAB

First you need to find the sampling interval by subtracting the time between two successive time samples. The following will do this:

```
Ts = time(2) - time(1); % sampling interval in seconds per sample
```

Next you need to find the sampling rate by inverting the sampling time:

```
fs = 1/Ts; % sampling rate in samples per second
```

After that we need to specify the number of points that we want for the spectral plot:

```
Npoints = 2^13; % number of points for the amplitude spectrum plot
```

The reason we choose a power of 2 for the number of points is that MATLAB Fast Fourier Transform algorithm works most quickly when the number of data points is a power of 2.

Next we need to specify the frequency samples of the spectral plot:

```
f = -fs/2 : fs/Npoints : fs/2-fs/Npoints; % frequency samples in Hz
```

After that we need to specify generate the samples of the voltage amplitude spectrum:

```
VoltageSpectrum = (abs(fftshift(fft(voltage, Npoints)))/sqrt(Npoints)/2/pi);
```

Next, convert the voltage amplitude spectrum to power in watts assuming a 50 Ω load. Use the equation $P = \frac{V^2}{R}$

```
R = 50;  % resistance in ohms
PowerSpectrumWatts = VoltageSpectrum.^2 / R';% power in Watts
```

The **.^2** raises the voltage array to the second power. The dot (**.**) tells MATLAB to perform an element-by-element operation on every single voltage in the voltage array.
Next, convert the power amplitude spectrum from Watts to mW:

```
PowerSpectrummW= PowerSpectrumWatts*1000
```

Next, convert the power amplitude spectrum from mW to dBm:

```
PowerSpectrumdBm = 10*log10(PowerSpectrummW);% power in mW
```

Now we can plot the amplitude spectrum:

```
figure; plot(f, PowerSpectrumdBm); grid;
```

If we divide the frequency by 10^6, the plot will show frequency in MHz instead of Hz:

```
figure; plot(freq1/1e6, PowerSpectrumdBm);
```