MA1710: Week 4

Creating two-dimensional plots

4.1 Introduction to the week 4 session

In this session we describe how to use Matlab to create two-dimensional plots and we take as some of the examples the curve sketching exercises in your `MA1710` sessions. There will also be a need to plot things in the latter part of your first project. As we describe, the basic plot command just needs a vector of x values and a vector of y values of the same length and thus the material described in the previous session about creating vectors is used again in this session. Often we want more than just one x–y plot, e.g. we wish to show two or more curves together, we may want labels and we may wish to include a version of the plot in a paper document. In this session we consider some of the things that can be done to 'fine tune' what is created. There will be few key commands to remember such as `figure`, `plot` and `print` with many other commands being things that you can look-up depending on your particular application.

4.2 The basic `plot` command and the `figure` command

We start with an example of plotting $f(x) = \sin(\pi x)$, $-2 \leq x \leq 2$. Create a script called `sess4a.m` which contains the following commands and run the script.

```
x=linspace(-2, 2);
y=sin(pi*x);
plot(x, y);
```

Here the `linspace` command creates 100 equally spaced points which creates `x`, the statement for `y` uses the `sin` function which acts in a vectorised way and the `plot` command creates a figure window showing a polygonal path joining the points `(x(1), y(1))`, `(x(2), y(2))`, ..., `(x(end), y(end))` with all the default settings. You should get a window

with a graph similar to that shown in Figure 4.1. (Figure 4.1 is actually the PDF version created by statements which are described in section 4.7 and the version in this format is not always identical in every detail to the the display in the figure window.)

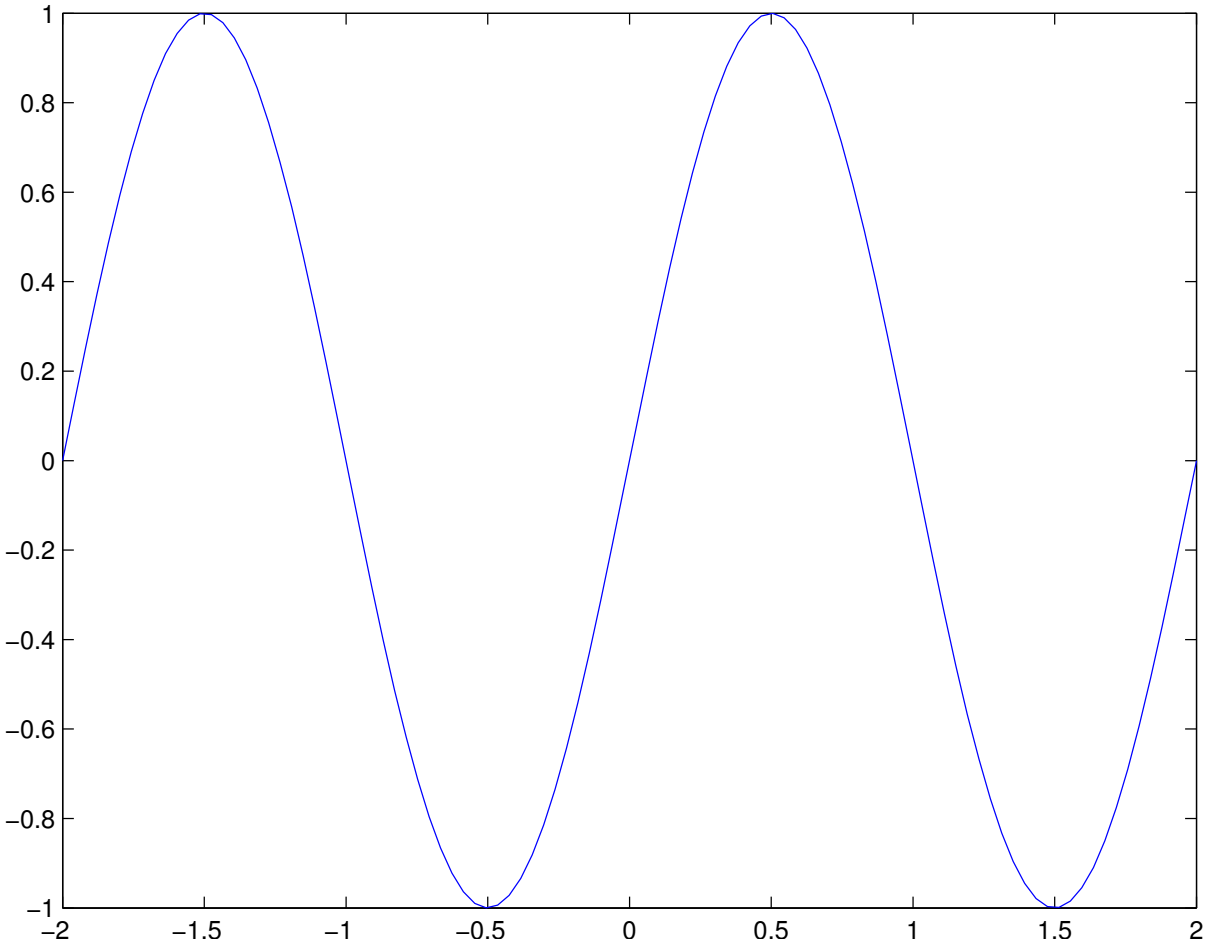

Figure 4.1: Basic plot of $y = \sin(\pi x)$, $-2 \leq x \leq 2$.

On the PC's the figure window is usually one of several windows that you may have active and to ensure that the figure window always appears when the script is run, instead of remaining hidden behind other windows, it is recommended that you add 1 line to the previous script so that it becomes the following.

```
x=linspace(-2, 2);
y=sin(pi*x);

figure(10)
plot(x, y);
```

The choice of 10 here is arbitrary and what it means is that every time the script is run a figure with this label is created or is overwritten if one already exists.

Extend the script further so that it contains the following and run the script.

```
% ...100 points case
x=linspace(-2, 2);
y=sin(pi*x);

figure(10)
plot(x, y);

% ...12 points case
x=linspace(-2, 2, 12);
y=sin(pi*x);

figure(11)
plot(x, y);

% ...401 points case
x=linspace(-2, 2, 401);
y=sin(pi*x);

figure(12)
plot(x, y);
```

Now 3 separate figure windows are created and you can compare the effect of using less points or more points than was done in the first case. It is difficult to visually distinguish the 100 point version and the 401 point version but using only 12 points is clearly too small to attempt to represent a smooth curve. The PDF version of the 12 point case is shown in figure 4.2.

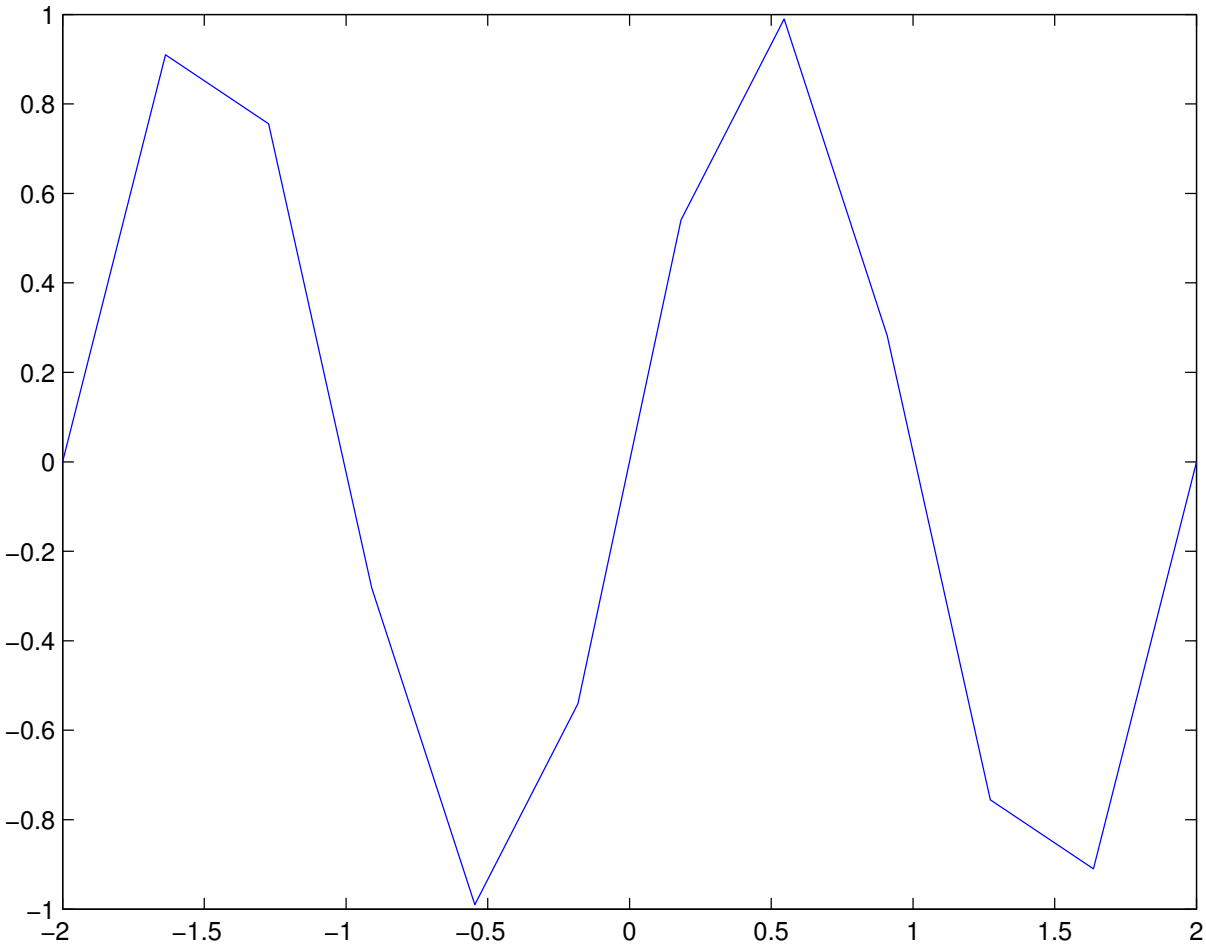

Figure 4.2: Polygonal path of $y = \sin(\pi x)$, $-2 \leq x \leq 2$ only using 12 points.

4.3 Adjusting the scales

We now consider how to plot a circle and we do this by using the parametrisation

$$(\cos(t), \sin(t)), \quad 0 \leq t \leq 2\pi$$

in the case of the unit circle.

Create a script file called `sess4b.m` which contains the following lines and run the script file.

```
t=linspace(-pi, pi, 400);
x=cos(t);
y=sin(t);

figure(13)
plot(x, y);
```

The paper version with the default settings is shown in figure 4.3 and the appearance is that of an ellipse as the x-values and the y-values are not scaled in the same way. To get a plot which looks like a circle add one line to the previous script so that it now is as follows.

```
t=linspace(-pi, pi, 400);
x=cos(t);
y=sin(t);

figure(14)
plot(x, y);
axis equal
```

The `axis equal` statement ensures that the circle plot does look like a circle and the paper version that it creates is shown in figure 4.4.

4.4 Plotting several things together

In all the examples given earlier in this session we used `plot` with just 2 arguments to plot one curve. By appropriately adding more arguments we can plot any number of curves and we start with an example of plotting two curves in the same figure. Using the editor create a script file containing the following lines and run it.

```
x=linspace(-pi, pi, 400);
y1=cos(x)+2*sin(x);
y2=sqrt(5)*cos(x);

figure(15)
plot(x, y1, x, y2);
```

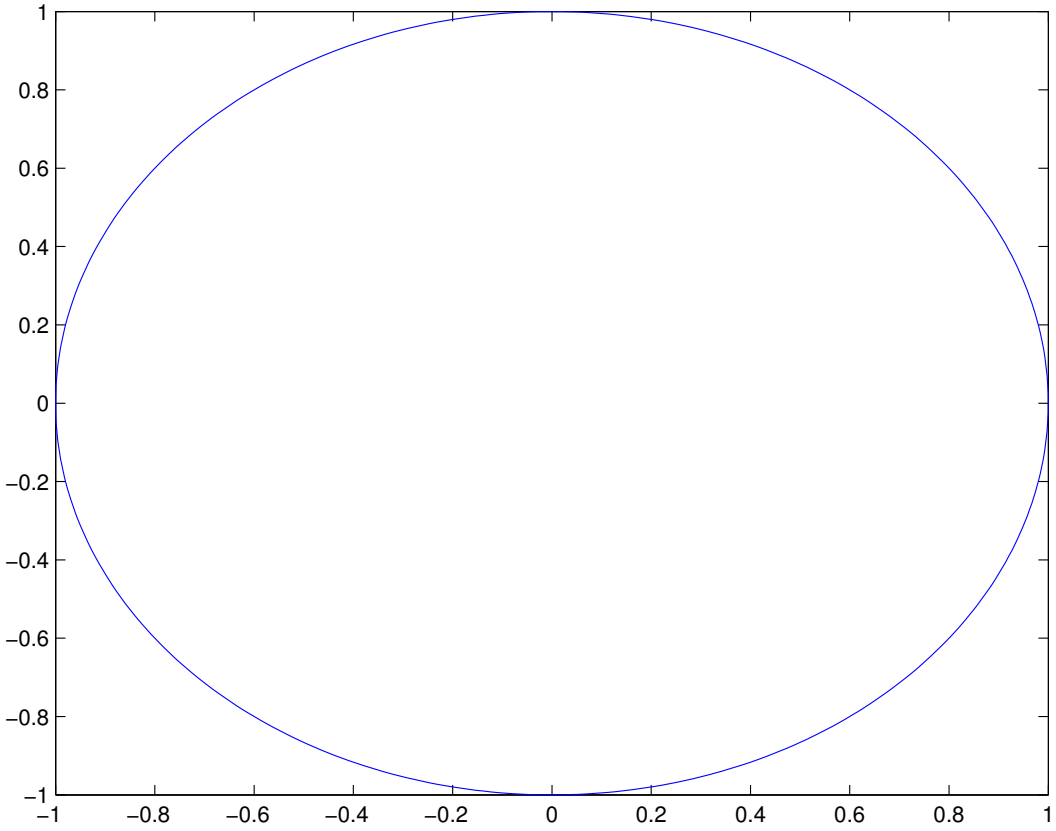

Figure 4.3: Attempt to show unit circle but the plot looks like an ellipse.

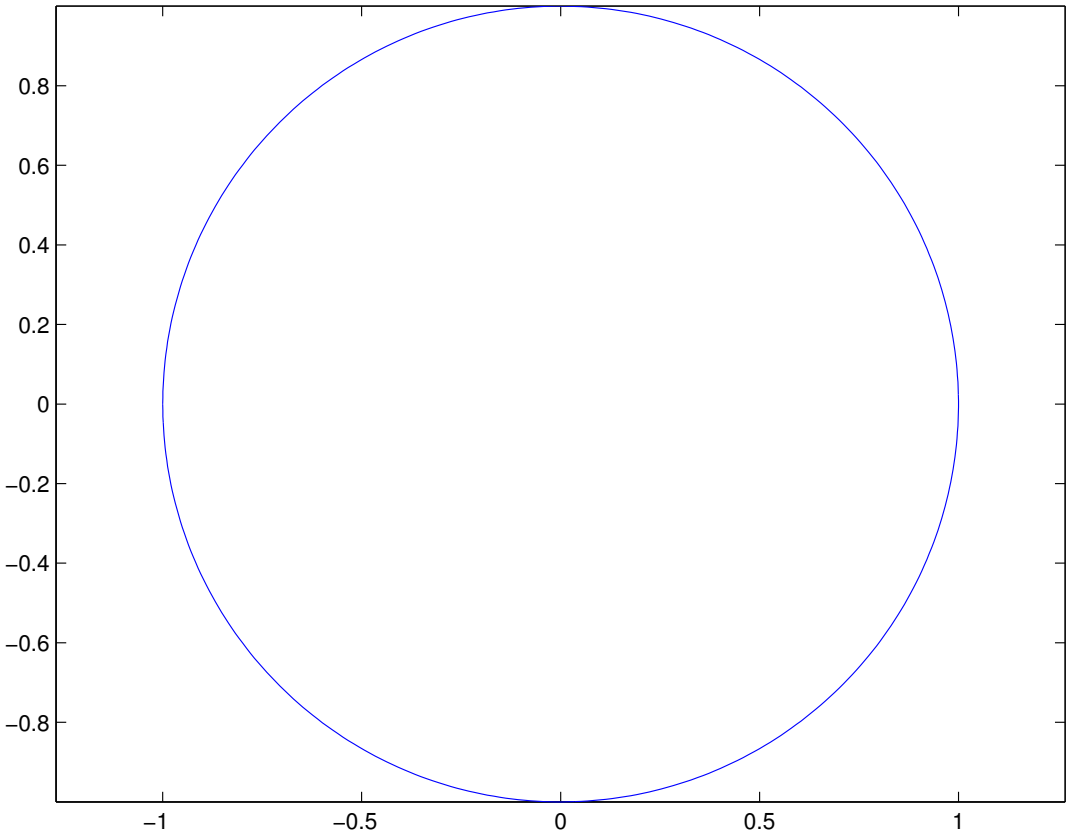

Figure 4.4: The unit circle.

To explain what is shown, note that from the `MA1710` sessions we have

$$\cos(x) + 2\sin(x) = \sqrt{5}\cos(x - \alpha)$$

with $\alpha = \tan^{-1}(2) \in (0, \pi/2)$. Thus we can get one curve from the other curve by a translation in the x-direction. (In Matlab we get $\tan^{-1}(2)$ by typing `atan(2)`.)

To be able to identify which curve is for `y1` and which is for `y2` change the plot instruction so that it is now as follows.

```
plot(x, y1, x, y2, '--');
```

Matlab knows what type of arguments are being used and if a part such as `'--'` is given then it specifies the line type of the part that it immediately follows, i.e. just the `x, y2` part is drawn with a dashed line. The paper version of the plot is shown in figure 4.5.

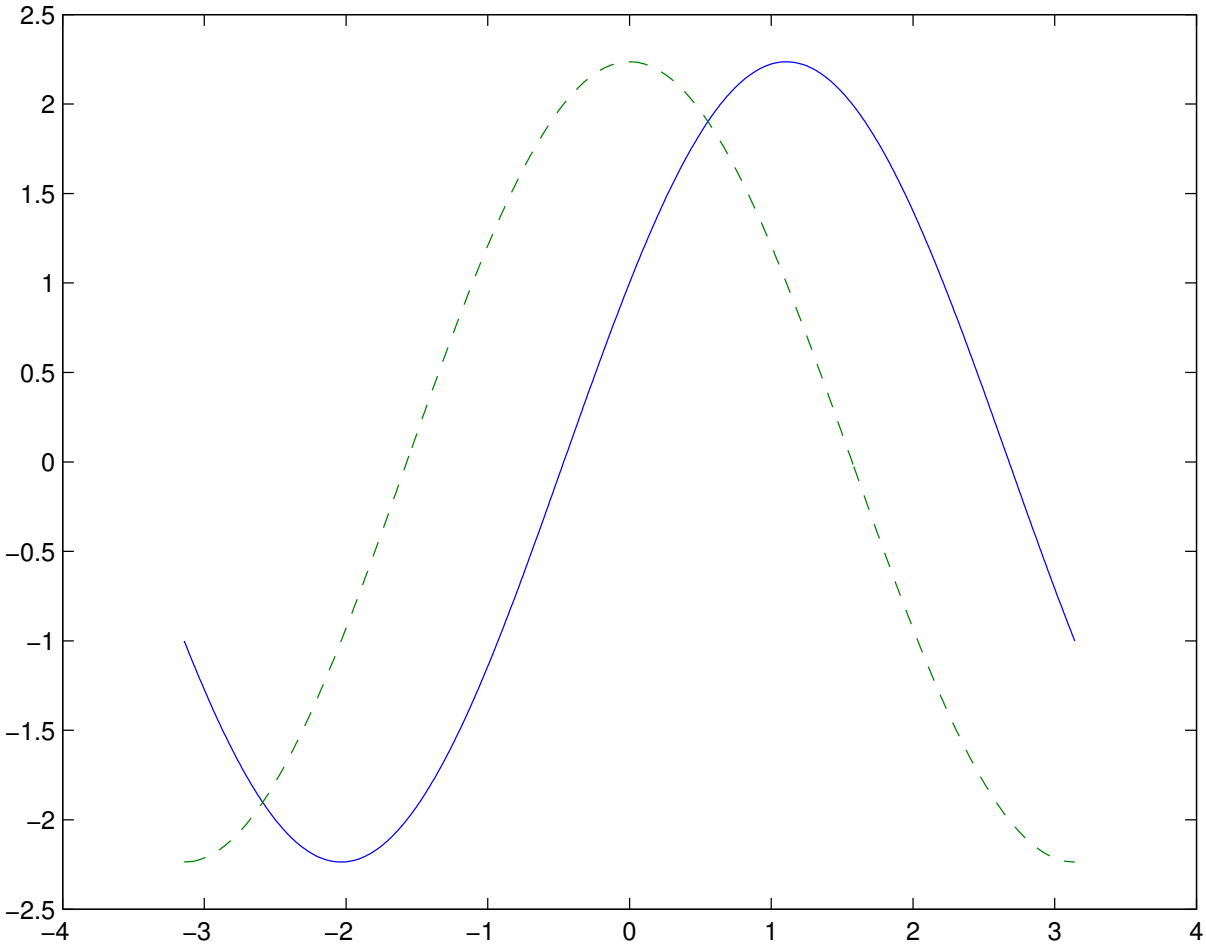

Figure 4.5: Plot of $\cos(x) + 2\sin(x)$ (solid line) and $\sqrt{5}\cos(x)$ (dashed line).

4.5 Adding labels, the fontsize, the line thickness ...

You can add labels to the axis, you can add a title, you can add a legend, you can adjust the fonts used and you can adjust the line thickness, i.e. you can adjust most things. (An example of a legend is not done until section 4.6.) We consider here the case of showing the graph of a function and its inverse function are mirror images in the line $y = x$.

In the following we consider how to plot $y = e^x$, $y = x$ and $y = \log(x)$ in the same figure. There is some thought needed in choosing suitable domains for each function. As e^x grows rapidly with x when $x > 0$ we restrict x to $(-2, 2)$ so that the range is (e^{-2}, e^2). The domain and the range of $\log(x)$ is the reverse of this and thus for both curves together we have a box with $-2 < x, y < e^2$ and the corners of this are used to show $y = x$. Note that the use of `axis equal` is appropriate here. With the above comments in mind, create a script file containing the lines shown below and run the script file. A paper version of the output created is shown in figure 4.6. As a comment, when the font size is increased for the numbers shown on the axis there are less numbers shown. This is a typical difference between what is shown in a figure window and what is shown when the plot is saved in other formats. (The term `gca` contains information about the axis on the current plot and the function `set` is re-setting one of the parameters.)

```
%% create the vectors for exp(x), log(x) and x
x1=linspace(-2, 2);
y1=exp(x1);

x2=linspace(exp(-2), exp(2));
y2=log(x2);

x3=[-2, exp(2)];

%% plot all the curves together with
%% an increased line width
figure(17)
plot(x1, y1, x2, y2, x3, x3, '--', 'LineWidth', 3)
axis equal

%% use 14pt for the numbers on the axis
set(gca, 'FontSize', 14);

%% add other things at 16pt
xlabel('x-axis', 'FontSize', 16)
ylabel('y-axis', 'FontSize', 16)
title('Plot of exp(x) and log(x)', 'FontSize', 16)
```

4.6 Using `hold on` and `hold off`

In all the examples given so far there has been just one plot command. If everything can be done easily with just one command then this is usually the best way of creating the graphs. However, if arranging things to be done in one go is more difficult than doing this separately or if you do not want the same line thickness for every curve then you need to use the `hold on` and `hold off` mechanism. To illustrate what can be done create a script file containing the following lines and run the script file for the case of plotting $y = x^3$ and its inverse in the same figure. The paper version of the plot is shown in figure 4.7.

```
%% create the vectors for x^3 and x^{1/3}
x1=linspace(-1.5, 1.5);
y1=x1.^3;

x2=y1;
y2=x1;

x3=[x1(1)^3 x1(end)^3];

%% start the figure and ensure it is clear
figure(18)
clf

%% now start the hold mechanism to plot separately and
%% add a legend
hold on
plot(x1, y1, 'LineWidth', 3)
plot(x2, y2, '-.', 'LineWidth', 2)
plot(x3, x3, '--');
leg=legend('y=x^3', 'y=x^{1/3}', 'y=x', ...
           'Location', 'Best');
set(leg, 'FontSize', 14);

axis equal

%% add labelling at enlarged size
set(gca, 'FontSize', 14);

%% add other things at 16pt
xlabel('x', 'FontSize', 16)
ylabel('y', 'FontSize', 16)
title('Plot of x^3 and x^{1/3}', 'FontSize', 16)

%% finish the hold mechanism
hold off
```

A few comments are needed to explain some of the parts. Firstly, if `hold on` is used then it is likely to be appropriate to first start with an empty figure as the subsequent plots are "added" to what already exists. It is the instruction `clf` which clears the figure window. Related to this comment it is advisable to have the `hold off` command later to stop further things being added. Secondly, Matlab knows the order in which you have done the plots for the `legend` statement to know what line type to show for each curve, thus the 3 descriptions corresponds to the 3 plots in the order that they are done. The last part of the `legend` statement controls how it appears with the `'Location'` and `'Best'` parts being to attempt to ensure that the legend is obscuring as little as possible of the plots. In the listing given above the use of `...` at the end of the line starting with `legend` means that the statement continues onto the next line.

4.7 Saving the plot to include in a document

To get a high quality paper copy of the graphics as a stand-alone item or to save the graphics in some way so that it can be included in a document can be achieved by using the `print` command. In the case of using Matlab on the Windows operating system with the program Word also available, as in our labs, there is also the option of 'copy figure' and pasting it directly into a Word document. We consider briefly the possibilities with comments about what should be done to retain reasonable quality throughout.

Using the `print` command and `epstopdf`

Unless otherwise stated, all the figures shown in this document were created using the `print` command to create what is known as an encapsulated postscript file which is vector format which is scalable. A PDF version was then created using a free program known as `epstopdf`. In the case of creating figure 4.7 this just involved adding the following lines to the script file.

```
print('x_cubed_and_inverse.eps', '-depsc')
system('epstopdf x_cubed_and_inverse.eps');
```

The `print` command creates `x_cubed_and_inverse.eps` and the `epstopdf` command creates `x_cubed_and_inverse.pdf` . Add such lines to your script, re-run the script and view the PDF created.

The `print` command can create many different formats with several of the formats actually being created by the program `ghostscript`. For example, in the case of plotting x^3 and its inverse you might also further add the following lines and re-run the script.

```
print('x_cubed_and_inverse.jpg', '-djpeg')
print('x_cubed_and_inverse.png', '-dpng')
```

This now creates versions as images in the `jpeg` and `png` format. These are suitable formats for web pages but they are not good formats for a paper version.

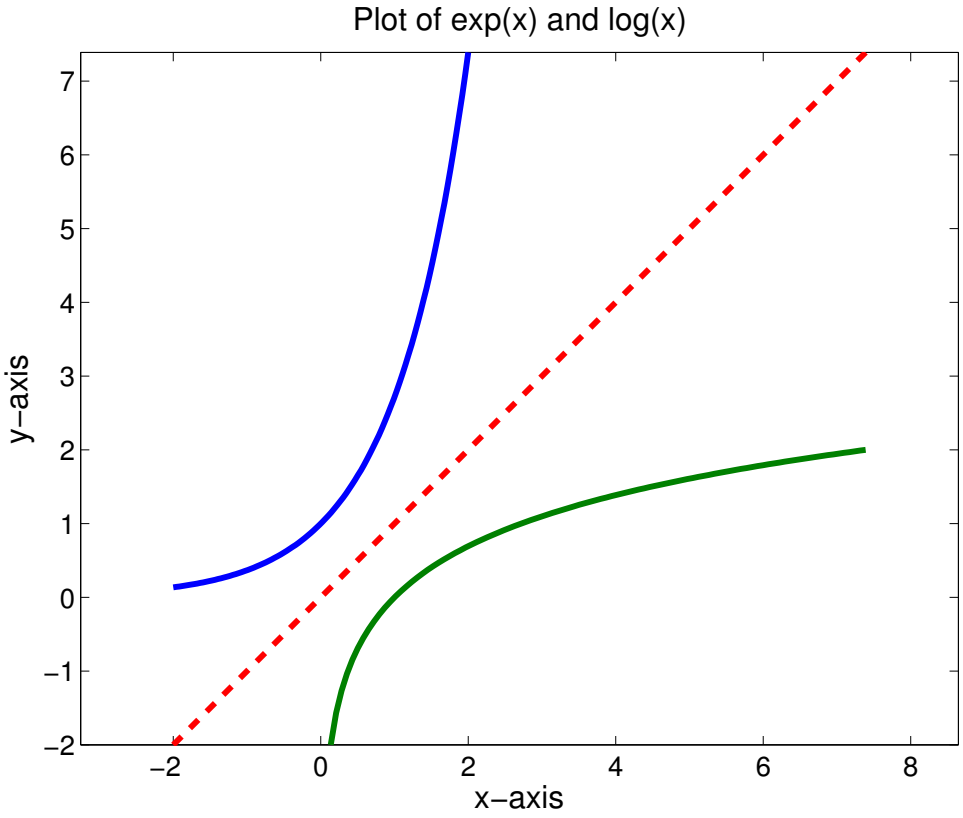

Figure 4.6: Plot of $\exp(x)$ and the inverse function $\log(x)$.

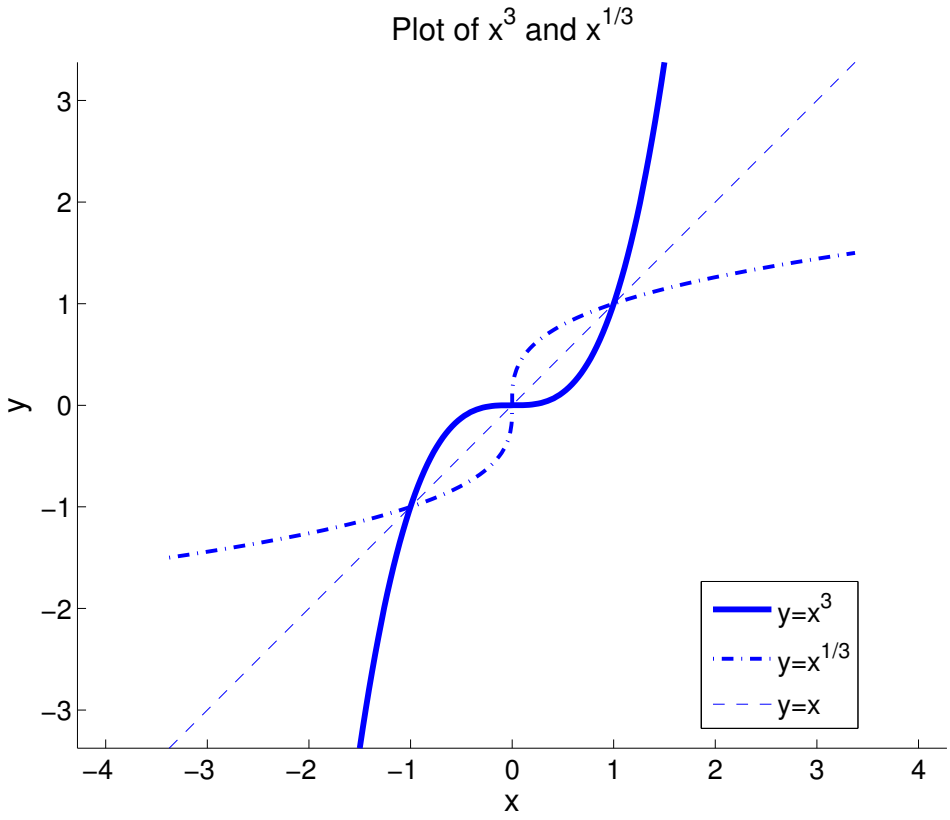

Figure 4.7: Plot of $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^3$ and the inverse function $x^{1/3}$.

Using the 'copy figure' option'

If you are creating the graphics to put in a Word document then you have the option of 'copy figure' which you get as a menu item of Edit which is on the top bar of the figure window. Try this and start a Word document and press the keys `Ctrl` and `V` together to paste the figure into the document.

In the case of Word you also have the option of creating a PDF, as explained above, and then including this as an object into the Word file. Try this with the same Word file. My attempt at doing both methods for the example of plotting the unit circle are shown in figure 4.8. In both cases shown it is probably better to adjust the Matlab script to increase the line thickness a little and to increse the size of the fonts in the labelling.

4.8 The size of the figure window

As has been stated a few times, it is possible to adjust almost every feature of what is created if you search hard enough for the command and/or the option to use. We can, for example, change the size of the figure window as the default size is perhaps a little small. Create a script containing the following lines and run the script to observe how this is done. In pixels the size of screens in the lab are 1920×1080 and hence this is the largest possible figure size.

```
x=linspace(0, 2*pi, 500);
y=sin(x)+0.1*sin(50*x);

%% plot with default size
figure(19);
plot(x, y);

%% plot in a 1000x800 screen
f=figure(20);
set(f, 'Position', [0, 0, 1000, 800]);
plot(x, y);

%% plot in a larger figure window
f21=figure(21);
fsz=get(0, 'ScreenSize')
set(f21, 'Position', 0.8*fsz);
plot(x, y);
```

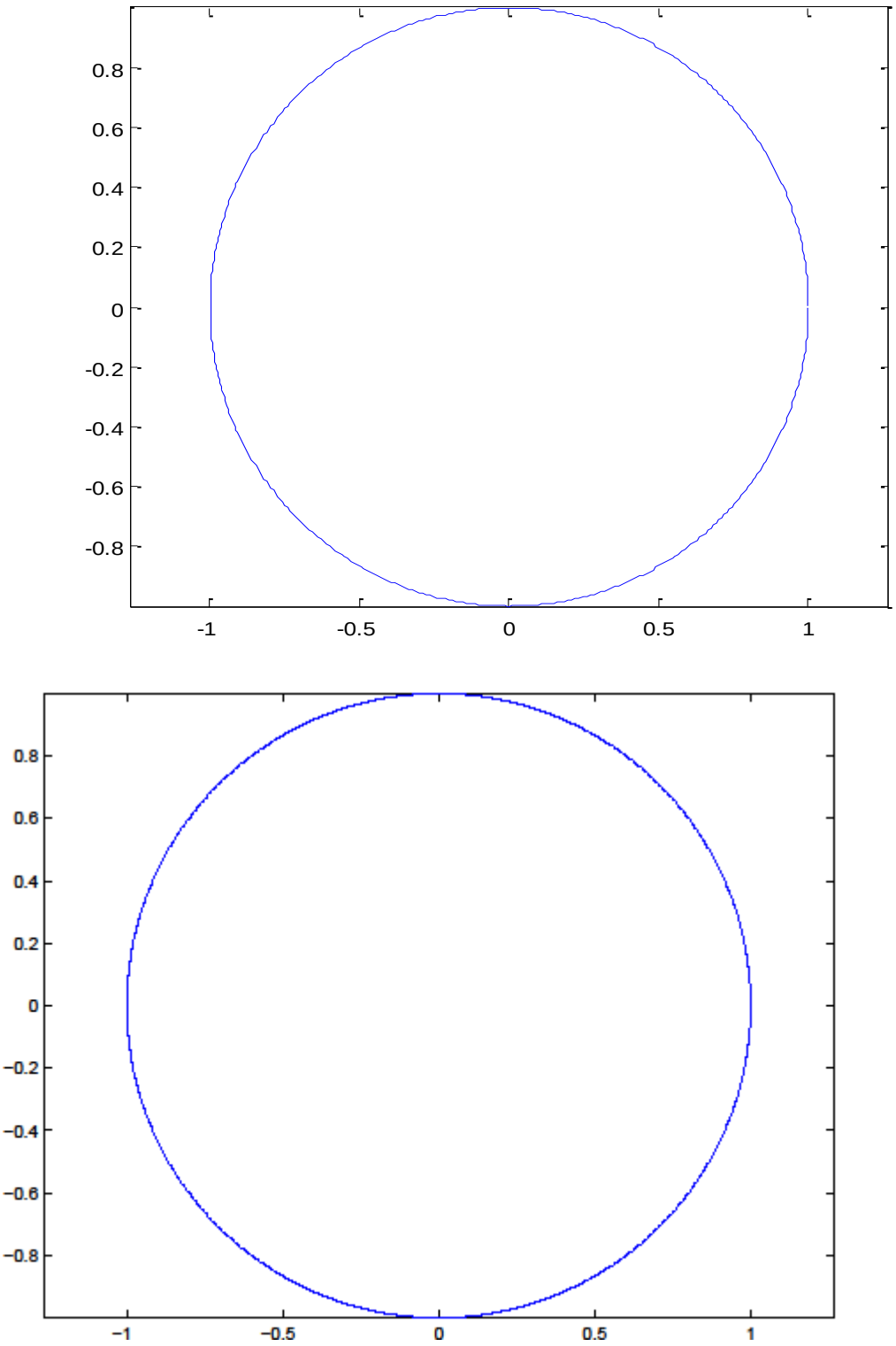

Figure 4.8: PDF of a word file of the circle example of section 4.3. The top version was created using the `edit` option in the figure window. From the options `Copy Figure` is chosen and then with word running the 'copy' is pasted into the word document by pressing together the two keys `Ctrl` and `V`. The bottom version is from first creating a PDF as described above and then including it in word by selecting `include` followed by `object`.

4.9 Exercises

From the examples that you have tried already in this session try the following exercises.

Exercise 4.9.1

Create a plot of $\tan(x)$, $-3\pi/2 \leq x \leq 3\pi/2$ so that you get a figure close to that shown in figure 4.9.

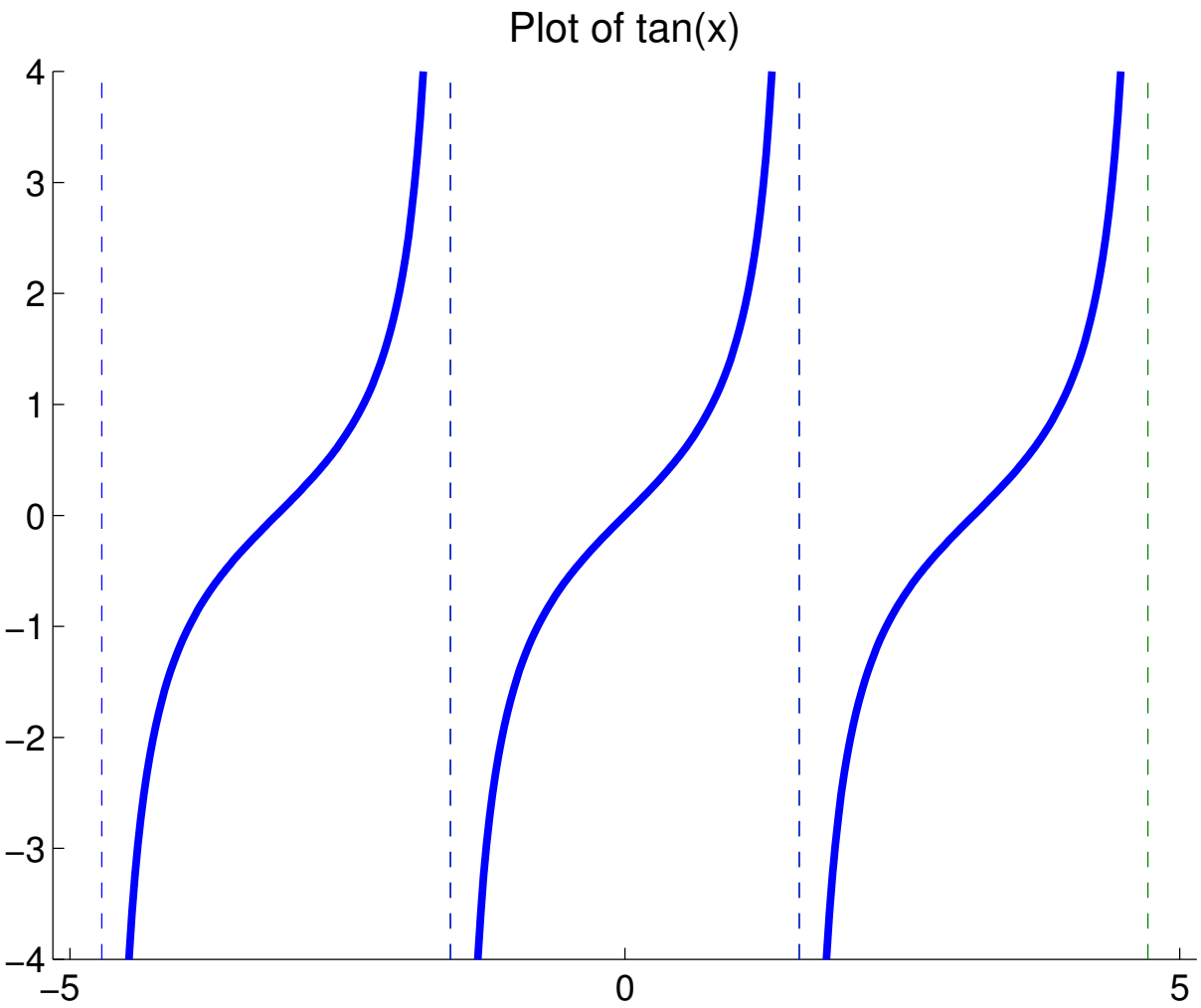

Figure 4.9: 3 periods of the function $\tan(x)$ corresponding to $-3\pi/2 \leq x \leq 3\pi/2$.

Exercise 4.9.2

The following parametric description of a curve corresponds to a spiral of the form

$$x(t) = t\cos(m\pi t), \quad y(t) = t\sin(m\pi t), \quad 0 \le t \le 1.$$

The plot for a specific value of m is shown in figure 4.10. Can you guess what value m is in the figure? Attempt to get a plot which looks like that shown in figure 4.10.

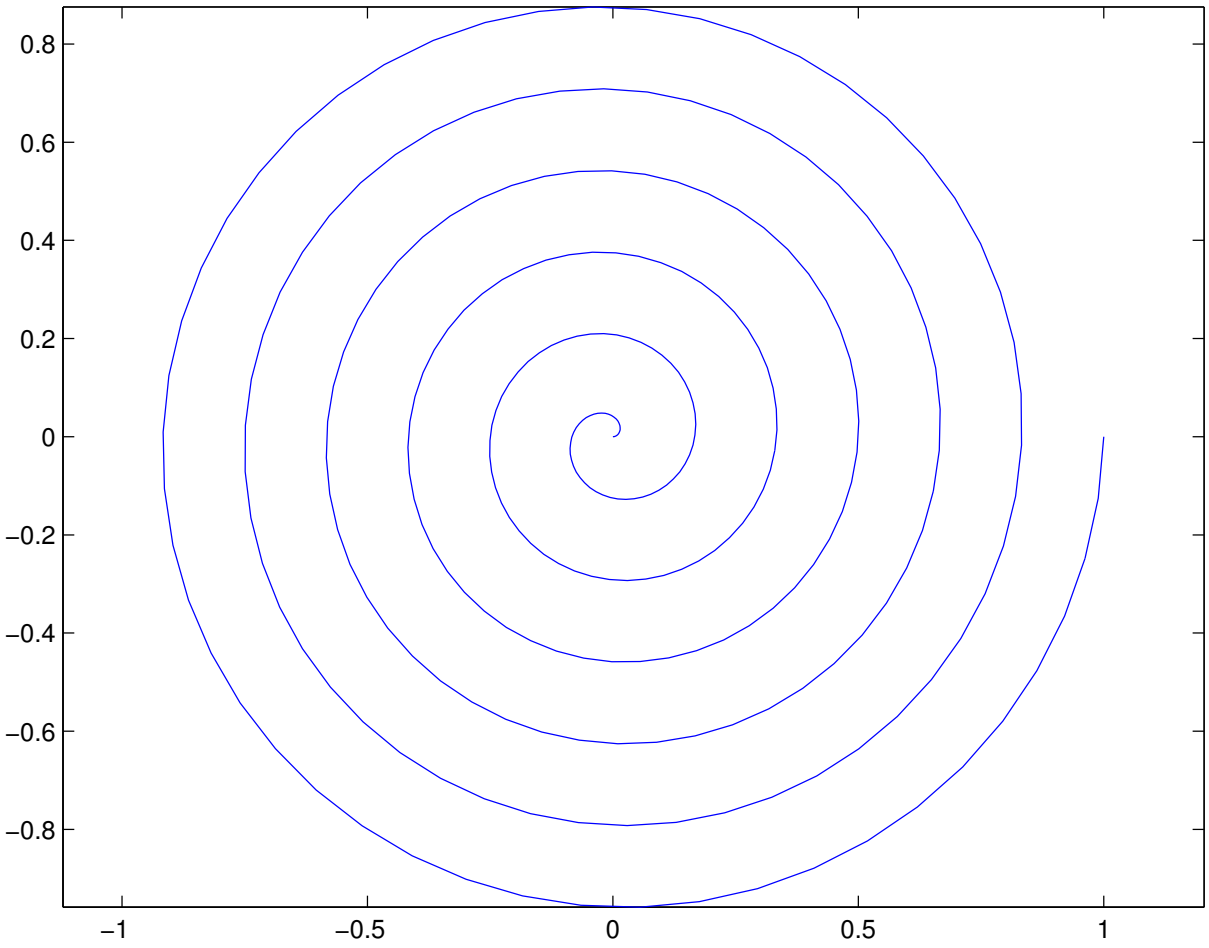

Figure 4.10: A spiral.

Exercise 4.9.3

The 16 roots of unity are represented on an argand diagram by 16 equally spaced points on the unit circle and a Matlab script to show this can be as given below. Create this and add additional statements involving loops so that a straight line segment is drawn from each point to every other point to attempt create what is shown in figure 4.11. How many straight line segments does this involve?

```
n=16;
t=linspace(-pi, pi, 400);
tp=linspace(-pi, pi, n+1);
x=cos(t);
y=sin(t);
xp=cos(tp);
yp=sin(tp);

figure(32)
clf
hold on
plot(x, y, 'LineWidth', 2);
plot(xp, yp, 'b*', 'LineWidth', 6);

axis equal;
title('16 roots of unity on the unit circle',...
      'FontSize', 18);
hold off
```

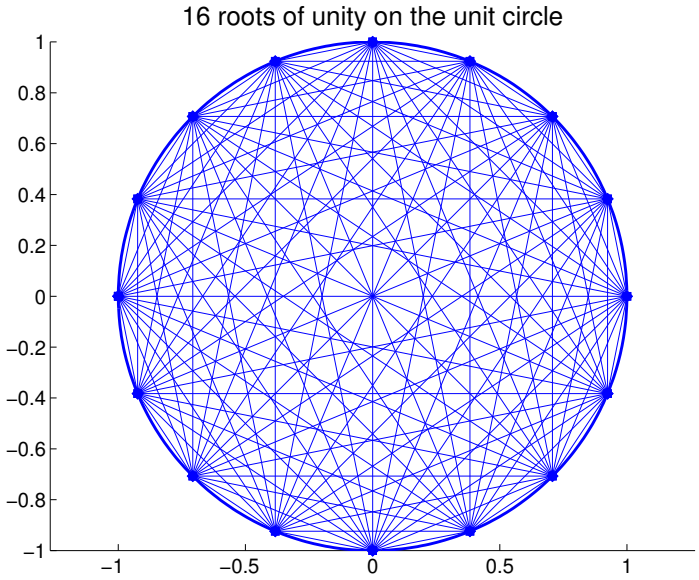

Figure 4.11: The 16 roots of unity and line segments from each point to every other point.