Introduction to Maple

The purpose of this document is to help you become familiar with some of the tools the Maple software package offers for visualizing curves and surfaces in $\mathbb{R}^2$ and $\mathbb{R}^3$. Maple can also be used to visualize vectors and vector fields, solve equations, integrate and differentiate functions, and much more, but we'll get to these later. We'll start by describing the basic syntax of Maple commands and then move on to the specific commands that we'll find most useful. In every case, we'll only be scraping the surface. For much more detailed information on the commands and their syntax and options you can take a look Maple's extensive "help" documentation. As you read this, you might find it useful to have Maple running on your computer so that you can try things out as you read about them.

1 Getting started - the basics

There are various options for displaying input and output in Maple and you should feel free to use the layout that you prefer. However, the format of examples that we'll give here assume that you're using the "classic" view. If you choose to use a fancier format you don't need to change the syntax of these examples, just how they're typed in.

The command prompt in Maple is the greater than symbol: `>` (although in some fancy display formats it's absent). Everything entered by you will follow one of these prompts. You'll never have to actually type in the command prompt: Maple does that part for you. After typing in a command, to end and execute it, type a semicolon (`;`) [1] and hit "enter". For example

```
3 + 4;
```

followed by "enter" will ask Maple to add 3 to 4, producing 7 on the following line. You can also end a command with a colon (`:`). The difference is that the colon tells Maple to suppress its output. This is useful when you want Maple to perform a computation but when you don't want to see the results right away (or at all). We'll see some instances of its use later.

Addition, subtraction and division all use the usual symbols: `+`,`-`, `/`. Multiplication is done using the asterisk: `*`. Maple *does not* interpret concatenation as multiplication. So, to enter $3y$ you need to type `3 * y` or `y*3` and *not* `3y` or `y3`. Exponentiation is achieved by using the caret: `^`. So, for example, to compute 18 multiplied by 2 to the seventh power, you would type

```
18*2^7;
```

followed by "enter".

[1] The most recent versions of Maple no longer seem to require the semicolon. And even in most current versions that do require it, omitting it typically only results in a warning.

Overuse of parentheses is never a bad idea when you aren't sure how Maple might choose to associate your operations. For example, if we wanted to enter

$$\frac{4x^2+5}{y-7}$$

we'd type

```
(4*x^2)/(y-7).
```

Note that we have enclosed the numerator and denominator in parentheses, with the division symbol `/` between them. Any complex fraction should be entered this way, otherwise Maple will get confused and give you funny results. Parentheses should be used in a similar way with exponents. y^{2x-4z} would be entered

```
y^(2*x - 4*z).
```

Such a use of parentheses is necessary any time the exponent consists of more that a single symbol. Another thing to watch out for is that Maple *only* uses the standard parentheses `(` and `)` to parse expressions. Brackets and braces like { or `[` have special uses that we'll come to later.

Maple knows most of your basic algebraic and transcendental functions. It knows how to compute values of the exponential function, natural log, sine, cosine, etc. . Most of these use the usual names you'd see in a math book or on a programmable calculator. For example, to compute $\cos 8$ you'd type `cos(8);` and hit "enter." To find the natural log of $\sqrt{2}$ you'd type `ln(2^(1/2));`. Note that we have used exponential notation for the square root. We could also have typed `sqrt(2)`. Be aware, however, that to evaluate the exponential function e^x you *do not* simply enter `e^x` as you might expect. You actually type `exp(x)`. So to compute $e^{4/3}$ you would type `exp(4/3);`. If you were to try this out, Maple would give you the output

$$e^{\left(\frac{4}{3}\right)}$$

which makes it look like Maple didn't do anything! If you want to know the decimal expansion of this or any other "symbolic" number you can use the command `evalf`. So, continuing with our exponential example, typing in `evalf(exp(4/3))` gives the output 3.793667893.

Okay, now for a few more interesting (and perhaps less obvious) commands. To assign values to variables you use the assignment operator `:=`. So, to create a variable named x and set it equal to 8, you type

```
x := 8;
```

end hit "enter". To create a variable name *george* and give it the value of x, you'd type

```
george := x;
```

and hit "enter". Variables that have numerical values assigned to them can be used in any expression the same way a number would be. For example, you could ask Maple to find the natural log of the variable *george* the same way you would ask it to find the natural log of 8: by typing `ln(george)`.

While you can name a variable almost anything in Maple, there are a few restrictions. Variable names cannot start with numbers (although they can include them otherwise), they cannot include spaces, and you should avoid special symbols like `%`, `#`, etc. since some of these have special meanings. A good piece of general advice is to only use letters, numbers and underscores in your variable names, e.g. `r_1`, `number_14`, `happy_face`. One nice thing about variable names is that if you type the name of a greek letter Maple will actually display it as a greek letter, and with the proper capitalization. So, `alpha` will be displayed in any output as α, `theta` will show up as θ, `delta` as δ and `Delta` as Δ.

Be aware that some variable names are automatically assigned values by Maple. The variable `Pi` stores the value of π, `gamma` stores the value of the Euler-Mascheroni constant γ, and there are many other examples. You don't really need to know all of the names that Maple has reserved for it's own use, since if you try to reassign a value to a protected variable you'll simply get an error message.

Variables can actually hold almost any kind of data that Maple knows how to process. In fact, we will frequently use variables to store equations or even entire plots. Encapsulating data using variables makes it easier to deal with many "large" objects at once. We'll see exactly how later.

2 Simple plots - the `plot` command

To plot functions of one variable, we use the `plot` command. The basic syntax to plot the function $f(x)$ on the domain $[a, b]$ is

```
plot(f(x),x=a..b);
```

As an example, to plot the cubic polynomial $x^3 - 3x + 5$ on the interval $[-3, 3]$ you would type

```
plot(x^3 - 3*x + 5, x=-3..3);
```
(1)

and hit enter. Figure 1 shows the resulting output. Note that the vertical axis is scaled so that it includes all values of the function on the specified domain. If you want specify your own vertical range, simply include a statement of the form `y=a..b`. For example

```
plot(x^3 - 3*x + 5, x=-3..3,y=-5..10);
```
(2)

produces the diagram shown in Figure 2.

There are lots of other options you can include in the `plot` command to customize your plots, and they are specified in a manner similar to the vertical range, that is by just tacking them on to the end of the command. To see what we mean, suppose we wanted to thicken the curve in the previous example and color it green. The appropriate syntax here would be

```
plot(x^3 - 3*x + 5, x=-3..3,y=-5..10,thickness=3,color=green);
```
(3)

and the result is shown in Figure 3. The interested reader can consult Maple's help file for a complete list of the options that can be included in the `plot` command.

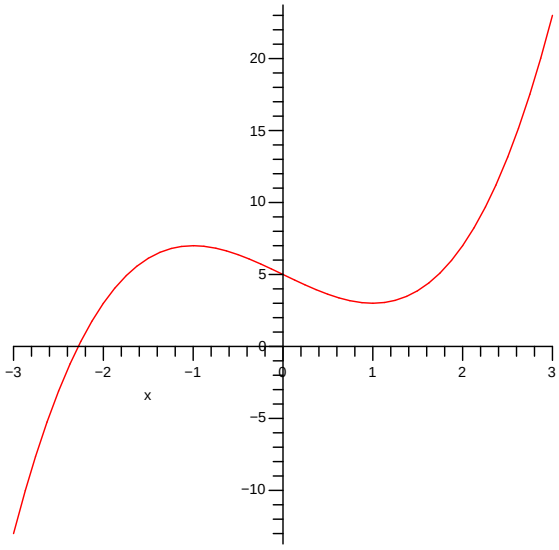

Figure 1: Output of command (1)

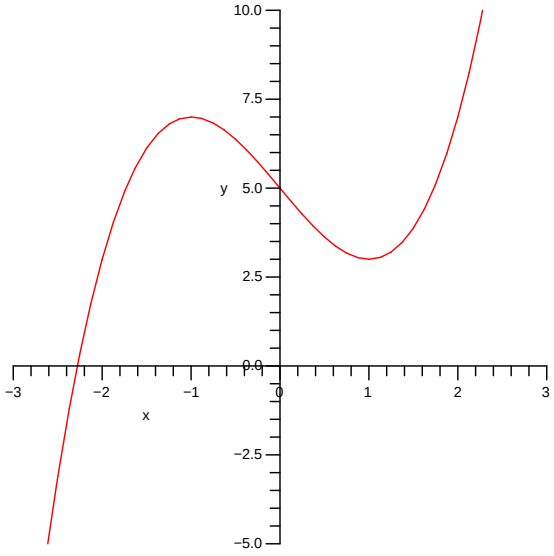

Figure 2: Output of command (2)

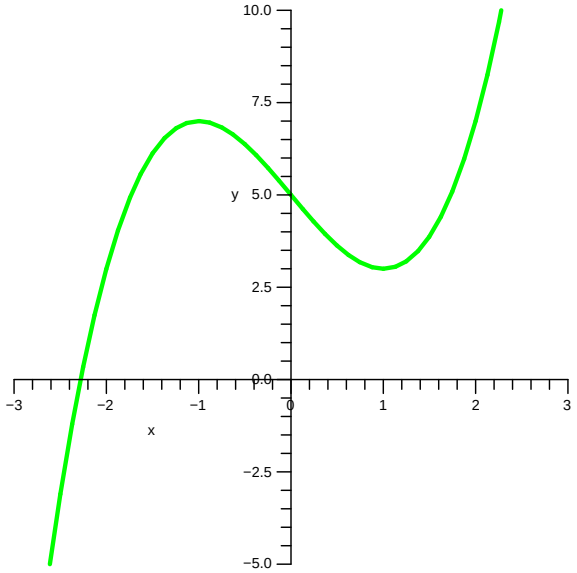

Figure 3: Output of command (3)

The simplest way to plot two functions on the same graph is to use the `plot` command with a slightly modified first argument. Rather than just enter a single function you can enter any number of functions, enclosed in square brackets (`[` and `]`) and separated by commas. Say we wanted to plot $\sin x$ and $\cos x$ on the same set of axes. We could type

```
plot([sin(x),cos(x)], x=0..4*Pi);
```
(4)

and hit enter to get the graph shown in Figure 4. Notice that Maple automatically gives each function a different color. If you'd rather have all of your functions be the same color, say blue, then you would add the option `color = blue` to the command above. And if you want to specify the color of each graph individually, say orange and yellow, you would add `color = [orange, yellow]`. Similar comments apply to the `thickness` option and many others.

The `plot` command can also be used to plot vector functions/parametric curves in $\mathbb{R}^2$. To plot the vector function $\mathbf{r}(t) = \langle x(t), y(t) \rangle$ for $a \leq t \leq b$, replace the single function we entered previously with the expression `[x(t), y(t), t=a..b]`. If we want to graph $\mathbf{r}(t) = \langle \cos(2t), \sin(6t) \rangle$ for $0 \leq t \leq \pi$ we'd type

```
plot([cos(2*t), sin(6*t), t=0..Pi]);
```
(5)

and after hitting enter we'd get the output shown in Figure 5. Notice that we didn't have to specify the ranges of either the horizontal or vertical axes. Maple takes care of this for you. But if you'd like to specify your own ranges, just add in `x=` and/or `y=` options like we did above. Likewise, options like thickness, color, etc. can be specified just as they were for the case of plotting functions. And if you want multiple parametric curves on the same plot, just enclose the expressions `[x(t), y(t), t=a..b]` for each curve in square brackets, separated by commas.

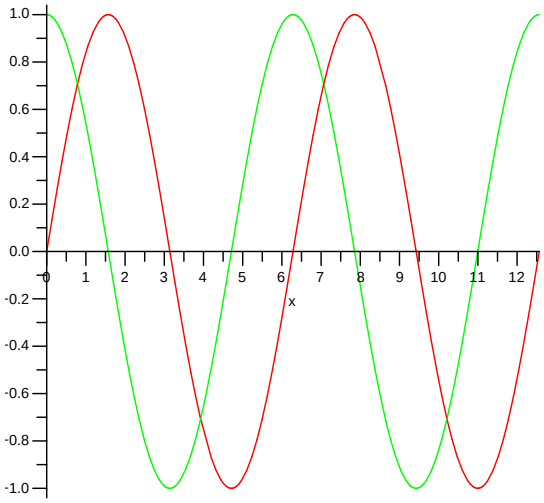

Figure 4: Output of command (4)

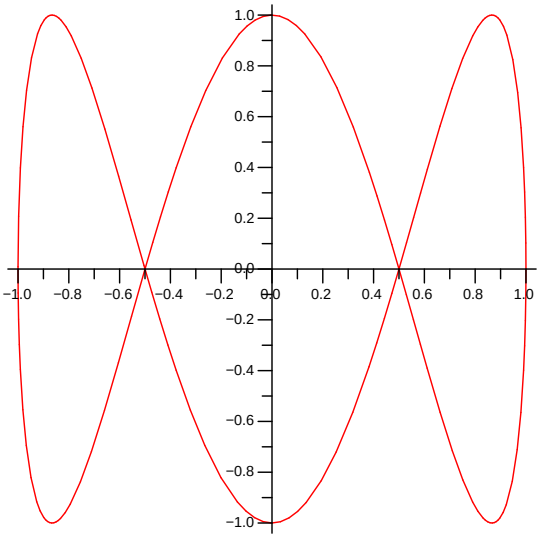

Figure 5: Output of command (5)

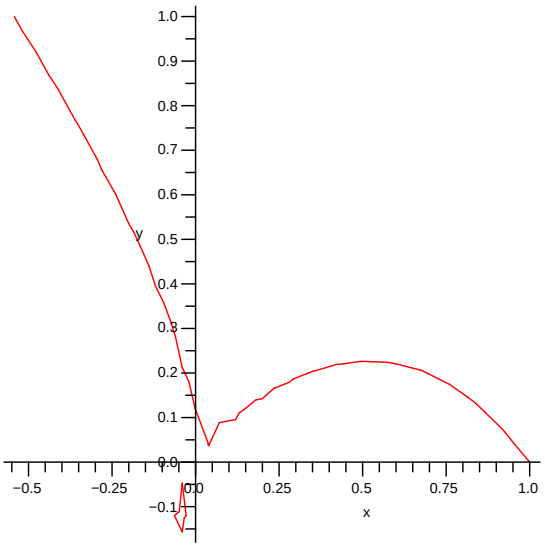

Figure 6: Output of command (6)

3 Advanced plots - the `plots` library

There are several other types of objects that we'll be interested in plotting, and these require us to load a special library of additional plotting commands. To do this just type `with(plots):` at the command prompt and hit enter (if you end this input with a semicolon instead of a colon it simply lists all of the plot commands that are now available). While we'll eventually meet quite a few of the plot commands that come with the `plots` library, for now we'll just focus on a few.

One class of objects Maple can plot using the `plots` library is curves of the form $f(x, y) = 0$, i.e. curves that are not necessarily the graphs of functions. To plot these types of curves we use the `implicitplot` command. It functions much the same way as the `plot` command, except that instead of specifying a function of a single variable we give an equation in both x and y. We also *must* specify the ranges of *both* x and y. So, as an example, if you wanted to know what the curve $x^3 + y^3 = x^2 - xy$ looks like, you might enter

```
implicitplot(x^3 + y^3 = x^2 - x*y, x=-1..1, y=-1..1);          (6)
```

The output is shown in Figure 6

Notice that the curve looks rather jagged and it seems to have a disconnected piece in the third quadrant. This commonly occurs with the `implicitplot` command due to the way that Maple produces its graphs, but is easily rectified. We include the `grid` option. If add `grid=[150,150]` to the command 6 we get the output shown in Figure 7, which is much smoother. In particular, we see that the odd piece in the third quadrant isn't disconnected from the rest of the curve after all! The larger the grid size the more accurate the resulting output will be, but Maple will also have to think a lot longer to draw the diagram. As

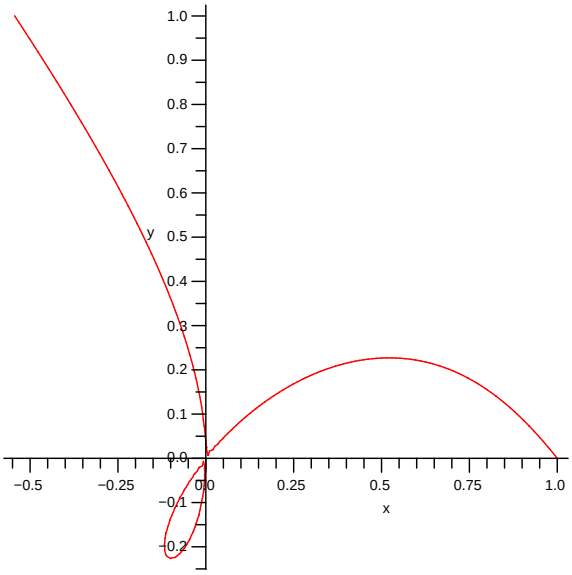

Figure 7: Output of command (6) with the option `grid=[150,150]` added.

with the ordinary `plot` command, options like thickness, color etc. can be specified in the `implicitplot` command, too.

The `plots` library also allows us to graph vector functions in $\mathbb{R}^3$ with the `spacecurve` command. Let's say that we want to see what the curve $\mathbf{r}(t) = \langle \sin t, \cos 2t, \sin 2t \cos 2t \rangle$, $0 \leq t \leq 2\pi$ looks like. As a first attempt we'd enter

```
spacecurve([sin(t),cos(2*t),sin(2*t)*cos(2*t)],t=0..2*Pi);        (7)
```

and get the diagram shown in Figure 8. Notice that if you hold down the button on your mouse and move your cursor while it's over the graph you can rotate it in various ways. This is nice, and should help you get a feel for the general shape of the curve, but where, exactly, is it in space? There are no axes for reference! This is easily resolved by adding the `axes` option. If we add in `axes = normal` to command 7 we get the plot shown in Figure 9, which has a nice set of 3 dimensional axes.

Another nice axis option is `axes=boxed` which puts your curve in a box with edges labeled with coordinates. We'll leave it to the reader to experiment with this option, as well as changing the thickness, color, viewing window, etc. of the curve to suit your preferences.

As a final example of advanced plots, let's talk about the `implicitplot3d` command. This functions much the same as the `implicitplot` command, but allows us to visualize solutions to equations of the form $f(x, y, z) = 0$, which are surfaces. For a basic example, let's get Maple to plot a sphere of radius 1. We know points on the sphere satisfy the equation $x^2 + y^2 + z^2 = 1$, and so the command we want is

```
implicitplot3d(x^2 + y^2 + z^2 = 1, x=-1..1,y=-1..1,z=-1..1);        (8)
```

which results in the graph shown in Figure 10. As with the `implicitplot` command you

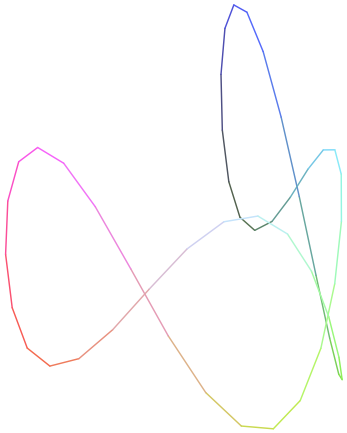

Figure 8: Output of command (7)

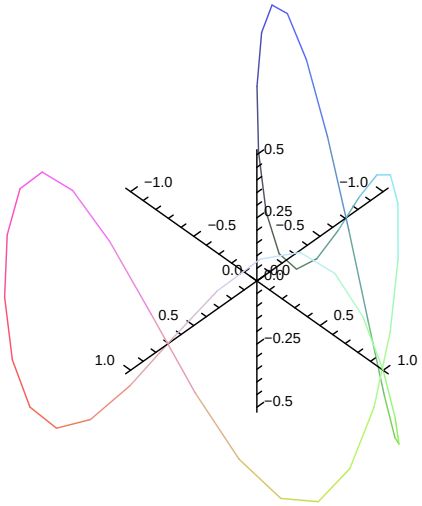

Figure 9: Output of command (7) with the option `axes=normal`

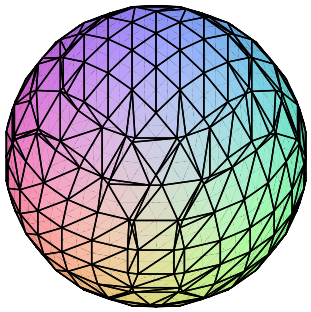

Figure 10: Output of command (8)

must specify the range of *every* variable, and like `spacecurve` you don't get any axes unless you ask for them using the `axes` option. Like `spacecurve`, too, you can grab the plot with your mouse and rotate it. If the sphere looks a little coarse to you, you can use the `grid` option, being sure to give 3 numbers instead of just 2 (as we did for `implicitplot`). Try `grid=[20,20,20]` and see what you think.

Since the graph of a function of two variables $f(x, y)$ corresponds to the equation $z = f(x, y)$, you can certainly use the `implicitplot3d` command to draw it. But, as with the case of functions of a single variable, Maple has a built in plotting feature for this special case. It's called `plot3d` and it works in much the same way as the `plot` command. Suppose we want to graph the function $f(x, y) = x^2y/(x^2+y^2)$ on the rectangular domain $[-5, 5]\times[-5, 5]$. We'd type

```
plot3d(x^2*y/(x^2 + y^2),x=-5..5,y=-5..5,axes=normal,view=-2..2);      (9)
```

and press "enter," obtaining the output shown in Figure 11. The `x=` and `y=` entries are necessary and give the limits of the plot in the xy-plane. Although we've made the ranges the same in this example, this isn't necessary. As with our earlier 3-dimensional plots, we need to include the `axes` option so that the coordinate axes are visible, although we can omit it if we don't want to see the axes. The option `view` limits the z aspect of the plot to the region with $-2 \leq z \leq 2$. It's optional, but can be very useful for plotting functions that have vertical asymptotes or grow very rapidly in their domain, since without it Maple will just plot the largest and smallest possible z values it finds. This can result in a very skewed graph.

Plotting contours of a function of two variables can also be achieved with the `implicitplot` command, since every contour of a function $f(x, y)$ is just a curve defined by the implicit

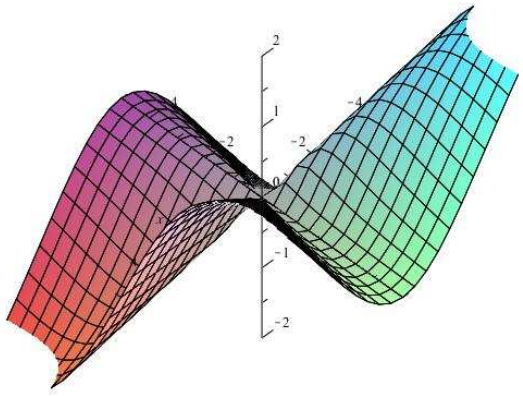

Figure 11: Output of command (9)

equation $f(x, y) = k$. While this can be very useful for visualizing specific individual contours, it's very inconvenient for producing contour plots, which would require us to assemble many implicit plots on top of one another (we address the topic of combining plots shortly). Again, Maple includes a plot command that readily produces contour plots with minimal effort. It's called `contourplot` and it works in much the same way as `implicitplot`. As an example, the input

```
contourplot((x^2+y^2-1)*x,x=-1..1,y=0..1);
```
(10)

will produce the contour diagram of $f(x, y) = (x^2 + y^2 - 1)x$ shown in Figure 12. As with the `implicitplot` command, notice that the contours produced are fairly jagged. We can get a better feeling for the true nature of the contours using the `grid` option. We can further improve the visualization by including more contours with the `contours` option. These are both implemented with the command

```
contourplot((x^2+y^2-1)*x,x=-1..1,y=0..1,contours=20,grid=[50,50]);
```
(11)

the improved output of which is given in Figure 13.

There's one final comment that should be made regarding options to the plot commands like `grid` and `contours`. Because these take whole number arguments which produce a number of subdivisions of the domain or range of a function, the parity of the arguments (whether they are even or odd) can change the detail obtained in the final diagram. We'll leave it to the reader to decipher why this is the case, but as an example, consider the following (seemingly trivial) modification of (11):

```
contourplot((x^2+y^2-1)*x,x=-1..1,y=0..1,contours=21,grid=[50,50],thickness=3);
```
(12)

whose output is shown in Figure 14. The main change in the input is the number of contours (21 now, 20 previously). The `thickness` argument was included simply to help illuminate the contour that has now manifested itself along the y-axis.

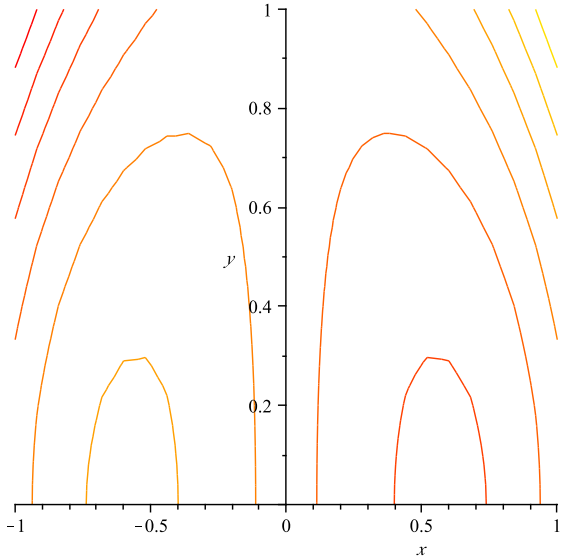

Figure 12: Output of command (10)

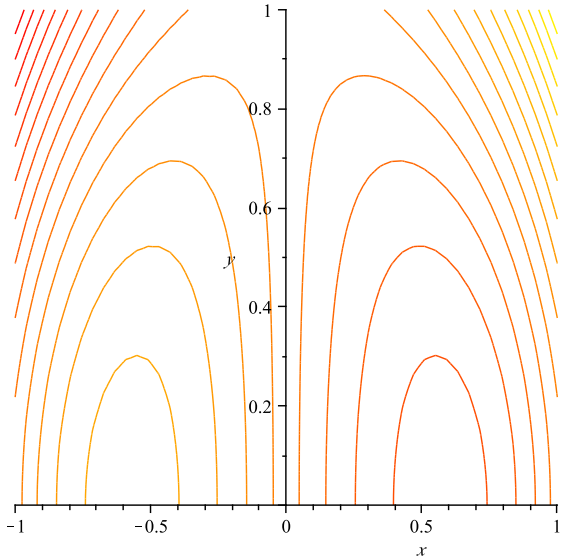

Figure 13: Output of command (11)

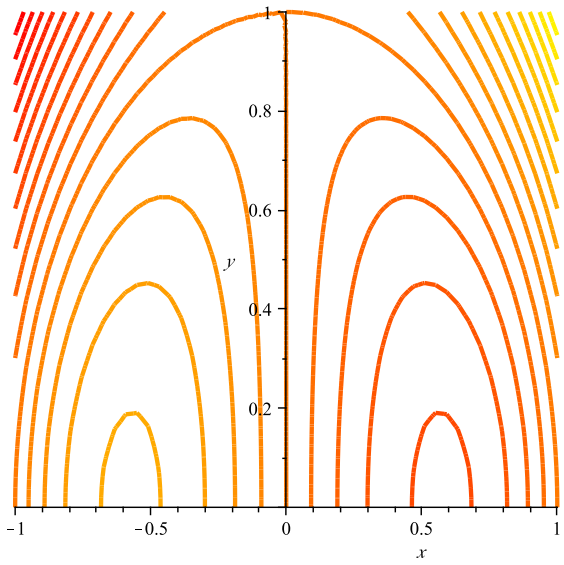

Figure 14: Output of command (12)

4 Combining Plots

So now you can plot surfaces and curves. But what if you want to put multiple plots on the same graph? For instance, to visualize the intersection of two surfaces you might like to plot them on the same set of axes. You might even like to improve your visualization by including a parametric version of the curve in a contrasting color with a relatively large thickness. These types of tasks are easily achieved by saving your plots in variables and then using the `display` command.

As with `plot` and `implicitplot`, the `display` command can only be used after loading the `plots` package (see above). The easiest way to use `display` is to compute your plots first, store them as variables and then tell `display` to show them to you.

As usual, we'll illustrate with an example. Suppose we'd like to look at the intersection of the surfaces $y^2 = x^2 + z^2$ and $z = y^2$. We can plot the later surface using `plot3d` and the former with `implicitplot3d`. We use the usual syntax, but now we store our input to separate variables, and suppress the output:

```
p1:=plot3d(y^2,x=-4..4,y=-2..2,color=grey):
p2:=implicitplot3d(y^2=x^2+z^2,x=-4..4,y=-4..4,z=0..4,grid=[25,25,25],color=blue):
```

Notice that we haven't included the axes, and we've made each surface a different color to help us later. To put these plots together we now enter

```
display(p1,p2,view=[-2..2,-2..2,0..2],axes=normal);
```
(13)

The first two arguments are the variable names we've stored our plots in, and we can add any number of plots in this way. The next (optional) argument tells us the x, y and z ranges of the plots that we'd like to see, and good choices can be determined by trial and error. The final option asks Maple to show the coordinate axes, as we've seen before. One nice aspect of `display` is that we can create our original plots with minimal restrictions or options, and then impose whatever restrictions we'd like all at once when we want to take a look at our combined pictures. The output of (13) is shown in Figure 15. Although the default viewpoint isn't that illuminating, a quick click-and-drag should help you get a nice view of the intersection.

If we want to add the intersection curve itself, we need to parametrize it first (sorry, I won't tell you how to get Maple how to do that for you). It turns out that the curve of intersection of the surfaces we've been considering can be parametrized by

$$\begin{aligned} x(t) &= \sin(t)\cos(t), \\ y(t) &= \sin(t), \\ z(t) &= \sin^2(t), \end{aligned}$$

for $0 \leq t \leq 2\pi$, an exercise we leave to the reader. We can store this as a space curve plot using the command

```
p3:=spacecurve([sin(t)*cos(t),sin(t),sin(t)^2],t=0..2*Pi,color=red,thickness=2):
```

(the options beyond the t domain being included solely for the purposes of visualization) and display it with the previous two surfaces using

```
display(p1,p2,p3,view=[-2..2,-2..2,0..2],axes=normal);          (14)
```

The resulting output, which again is better understood by dragging it around, is shown in Figure 16.

5 Closing comments

We've barely even scratched the surface of all the things one can do with Maple, but hopefully you've seen enough to get in and play around with some of its features. You should definitely sit down and experiment with Maple to get a feel for it and to see what kind of wild graphs you can create! Eventually we'll learn how to plot functions of 2 variables, combine different types of plots, and even integrate and differentiate functions, but if you want to get a head start check out Maple's help documentation. It's an extremely valuable resource, and even if you don't understand the abstract syntax of a command it always includes lots of examples. In fact, I learned how to use Maple almost entirely by reading its help files! And don't worry: with a little practice you'll get to know all of the commands we'll need and their various options and you might even wonder how you ever got along without it.

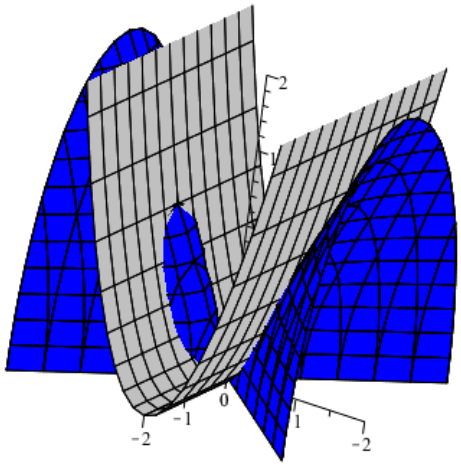

Figure 15: Output of command (13)

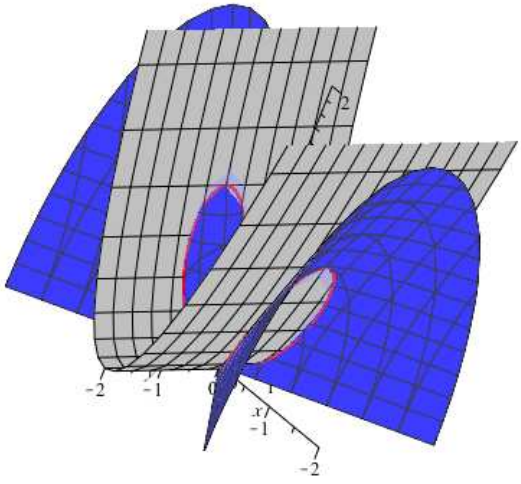

Figure 16: Output of command (14)