January 30, 2024

I managed to crash maple about 2/3 of the way through the class and lost everything I did. I need to save more often.

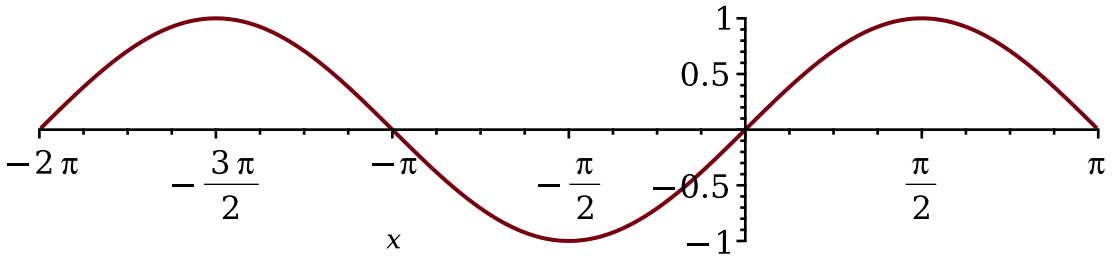
Below, I tried to reconstruct what I did in class, but may not be completely accurate. Sorry.
After that is what was actually done in the lattter part of the class.

Attempt at reconstruction of beginning of class.

First, we talked a bit about plotting. Let's start with a regular plot. I added *scaling = constrained* in order to ensure that the 1 vertical unit and 1 horizontal unit are the same length.

$$scaling = constrained \qquad (1.1)$$

```
> plot(sin(x), x = -2 Pi..Pi, scaling = constrained)
```

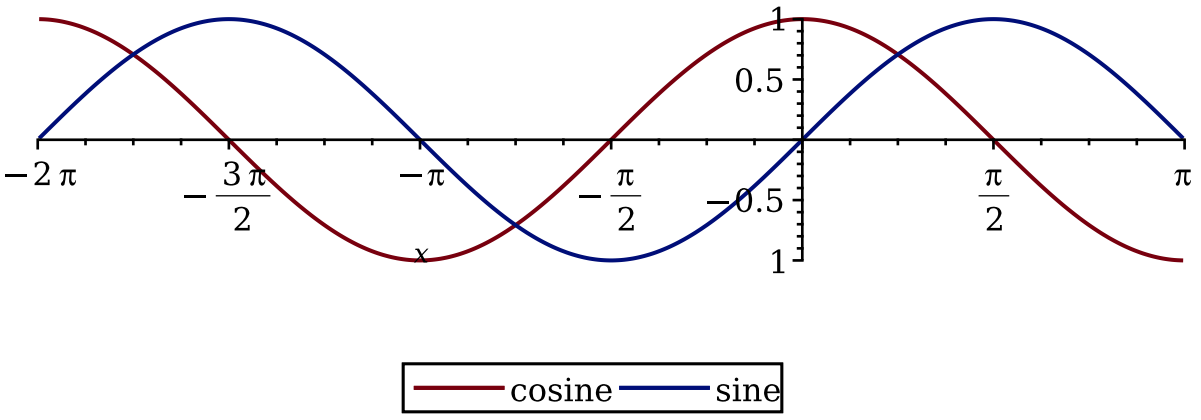


Suppose I want to see both the sine and the cosine on the same axes. The following fails because the first argument to plot wants the function or expression to be plotted.

```
> plot(cos(x), sin(x), x = -2*Pi .. Pi, scaling = constrained)
Error, (in plot) unexpected options: [sin(x), x = -2*Pi .. Pi]
```

We have to group the cos(x) and sin(x) functions together in a list (that is, plot a list of functions), which plot is happy to do.

```
> plot([cos(x), sin(x)], x = -2*Pi .. Pi, scaling = constrained,
  legend = ["cosine", "sine"])
```



Lists, sequences, etc.

A sequence is just a bunch of things separated by commas:

> $stuff := 26, \mathrm{Pi}/6, rabbit, \text{"rabbit"}$

$$stuff := 26, \frac{\pi}{6}, rabbit, \text{"rabbit"} \quad \textbf{(1.2)}$$

> $stuff[3]$

$$rabbit \quad \textbf{(1.3)}$$

> $floppy := stuff[4]$

$$floppy := \text{"rabbit"} \quad \textbf{(1.4)}$$

> $flippy := \textbf{(1.2)}[3]$

$$flippy := rabbit \quad \textbf{(1.5)}$$

Here *flippy* is assigned to be the 3rd element of the sequence *stuff* and floppy is the 4th one. Note that *flippy* is assigned the value *rabbit* (no quotes) while floppy is assigned "rabbit". *flippy*'s value is another name (ie, name of a variable), while *floppy* has the value of a string of characters. They behave differently:

> **flippy[4]**

$$rabbit_4 \quad \textbf{(1.6)}$$

> **floppy[4]**

$$\text{"b"} \quad \textbf{(1.7)}$$

>

Recall that earlier we put our pair of functions inside square brackets to make it one object (a list). We can also use curley braces {}. Both are single objects with sub-parts, but a list has a fixed order (think of it like a -tuple), but a set does not. Furthermore, lists can have repeated values, but in a set, the elements are unique.

> $aSet := \{1, 2, 4, 3\}$

$$aSet := \{1, 2, 3, 4\} \quad \textbf{(1.8)}$$

> $aList := [1, 2, 4, 3]$

$$aList := [1, 2, 4, 3] \quad \textbf{(1.9)}$$

> $\{1, 2, 1\}$

$$\{1, 2\} \quad \textbf{(1.10)}$$

> $[1, 2, 1]$

$$[1, 2, 1] \quad \textbf{(1.11)}$$

members of lists or sets can be other sets or lists or any other object:

> $\{aSet, aList\}$

$$\{[1, 2, 4, 3], \{1, 2, 3, 4\}\} \quad \textbf{(1.12)}$$

recall that *flippy* had the value *rabbit*, and *flippy[4]* was $rabbit_4$ but if we assign a value to rabbit, flippy takes on that value, as does it's 4th element:

>

> $rabbit := \left[1, 2, hat, cat, idk, \frac{\mathrm{Pi}}{6}\right]$

$$rabbit := \left[1, 2, hat, cat, idk, \frac{\pi}{6}\right] \quad \textbf{(1.13)}$$

> $flippy$

$$\left[1, 2, hat, cat, idk, \frac{\pi}{6}\right] \quad \textbf{(1.14)}$$

> $flippy[4]$

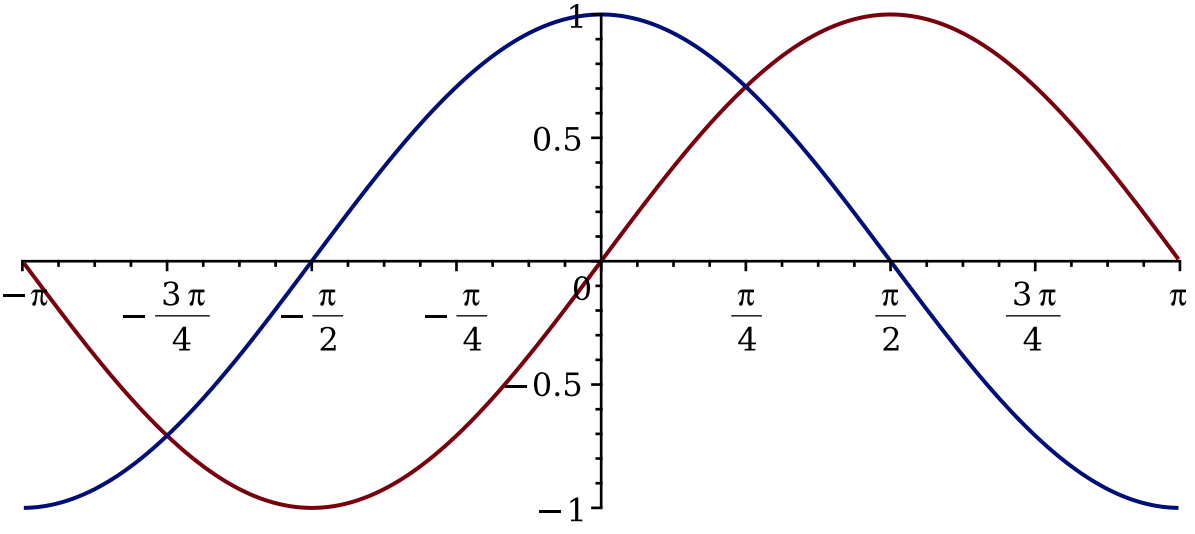
$$cat \quad \textbf{(1.15)}$$

Also, when plotting, we can have either expressions or functions, as in

> $myfuncs := [\sin, \cos]$

$$myfuncs := [\sin, \cos] \quad \textbf{(1.16)}$$

> $plot(myfuncs, -\mathrm{Pi}..\mathrm{Pi})$



Note that above, I couldn't do the following

> $plot(myfuncs, x = -\mathrm{Pi}..\mathrm{Pi})$

Error, (in plot) expected a range but received x = -Pi .. Pi

because the things in the list are functions, not expressions in x, so plugging

in values for x does not work. Maple lets us give the domain as a range, rather than saying that the variable is within the range.

We might want to generate a sequence via some expression, for example, gives the integers from -5 to 10

```
> seq(i, i=-5..10)
```

$$-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \quad \textbf{(1.17)}$$

or we can ge the squares of the first 10 intergers 1..10

```
> seq(i^2, i=1..10)
```

$$1, 4, 9, 16, 25, 36, 49, 64, 81, 100 \quad \textbf{(1.18)}$$

```
> Squares := %
```

$$Squares := 1, 4, 9, 16, 25, 36, 49, 64, 81, 100 \quad \textbf{(1.19)}$$

(here % means "the last answer")

```
> Squares[5]
```

$$25 \quad \textbf{(1.20)}$$

```
> Squares[15]
```

Error, invalid subscript selector

The above gave an error since we only defined the list with 10 elements, so referring to the 15th one doesn't make sense.

As mentione before, certain contstants are predefined, like the square root of negative 1 (I) or the ratio of the circumference of a circle to its diameter (Pi). But others (like the base of the natural logarithm) aren't built in. We can define them ourselves if we want:

```
> I
```

$$I \quad \textbf{(1.21)}$$

```
> I^2
```

$$-1 \quad \textbf{(1.22)}$$

```
> Pi
```

$$\pi \quad \textbf{(1.23)}$$

```
> evalf(Pi, 15)
```

$$3.14159265358979 \quad \textbf{(1.24)}$$

E or e is not predefined, but we can do it if we like:

```
> E
```

$$E \quad \textbf{(1.25)}$$

```
> E := exp(1);
  evalf(E)
```

$$E := e$$

$$2.718281828 \quad \textbf{(1.26)}$$

Sometimes we might want to apply a function to every element of a list (or set, or seqence) that we already have dealt with. For example

```
> f(x) := sqrt(x-3)
```

(1.27)

$$f := x \mapsto \sqrt{x-3} \quad \textbf{(1.27)}$$

> $f(Squares[3])$

$$\sqrt{6} \quad \textbf{(1.28)}$$

We could generate a new sequence of f applied to the i-th element of *Squares*

> $seq(f(Squares[i]), i = 1..10)$

$$I\sqrt{2}, 1, \sqrt{6}, \sqrt{13}, \sqrt{22}, \sqrt{33}, \sqrt{46}, \sqrt{61}, \sqrt{78}, \sqrt{97} \quad \textbf{(1.29)}$$

but if we try to just plug Squares directly into f, we don't get what we want (since f is a function of a single number, not a list of them.

> $f(Squares)$

$$I\sqrt{2} \quad \textbf{(1.30)}$$

> $f([Squares])$

$$\sqrt{[1, 4, 9, 16, 25, 36, 49, 64, 81, 100] - 3} \quad \textbf{(1.31)}$$

Another thing we can do is tell Maple to explicitly apply f to each element of squares, using map. But we have to make sure Squares is a list (not a sequence)

> $map(f, [Squares])$

$$[I\sqrt{2}, 1, \sqrt{6}, \sqrt{13}, \sqrt{22}, \sqrt{33}, \sqrt{46}, \sqrt{61}, \sqrt{78}, \sqrt{97}] \quad \textbf{(1.32)}$$

Surrounding a sequence with brackets turns it into a list. We can undo this using op (which you can think of as meaning "open" although it really stands for operands)

> $Sqlist := [Squares]$

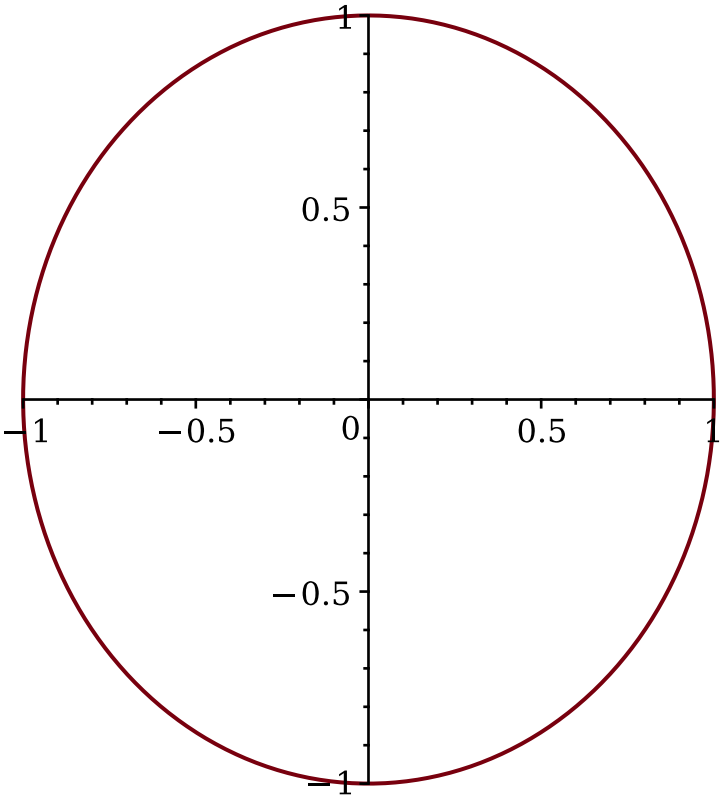
$$Sqlist := [1, 4, 9, 16, 25, 36, 49, 64, 81, 100] \quad \textbf{(1.33)}$$

> $op(Sqlist);$

$$1, 4, 9, 16, 25, 36, 49, 64, 81, 100 \quad \textbf{(1.34)}$$

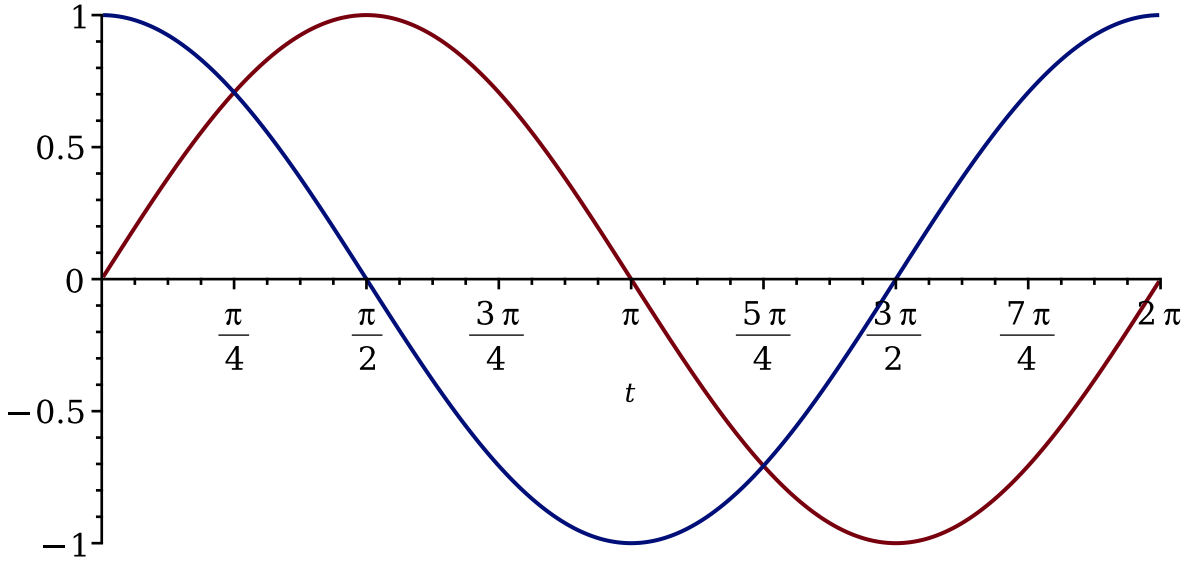
If we want to refer to a parametric equation as a list where the last entry is a variable and a range, as in the example below

> $plot([\sin(x), \cos(x), x = 0..2\,\mathrm{Pi}])$



Note that putting the range of t outside the bracket gives the plot of two functions, rather than a parametric representative

> $plot([\sin(t), \cos(t)], t = 0..2\,\mathrm{Pi})$

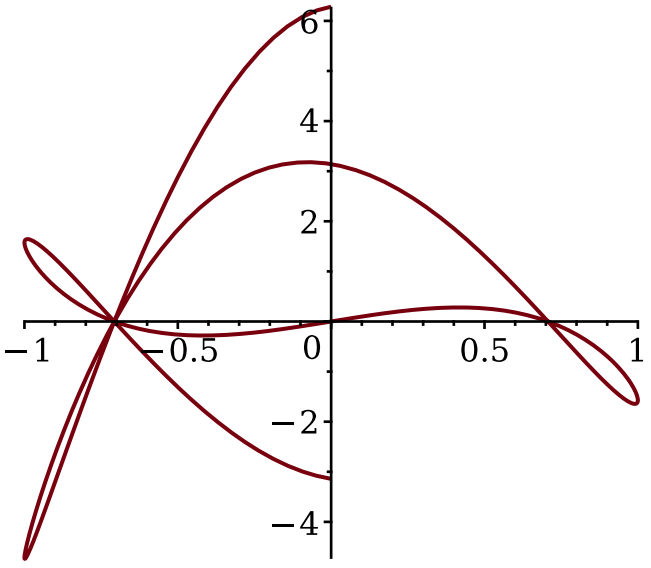


Here is where I messed up and crashed maple....

Actual second part of class meeting

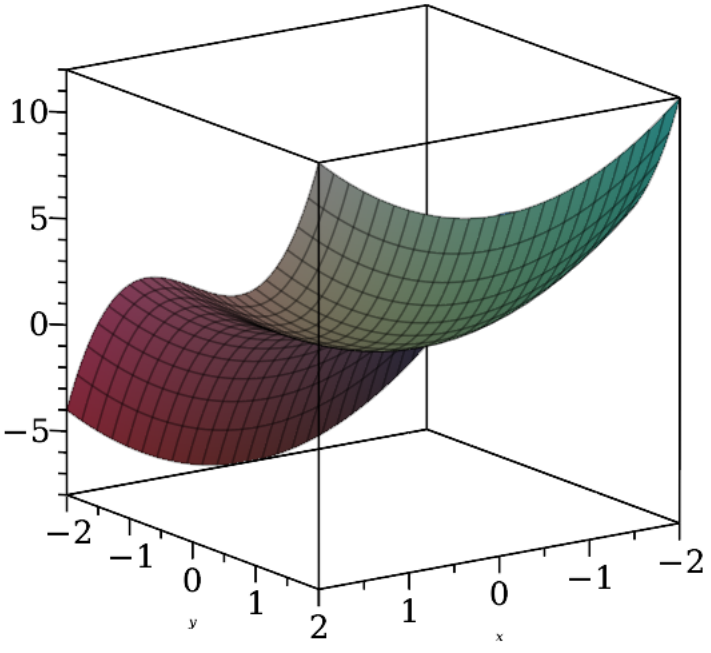
I was talking about parametric plots.

> $plot([\sin(t), t\cos(2\,t), t = -\mathrm{Pi}..2\cdot\mathrm{Pi}])$



We can also plot surfaces y=f(x,y) with plot3d

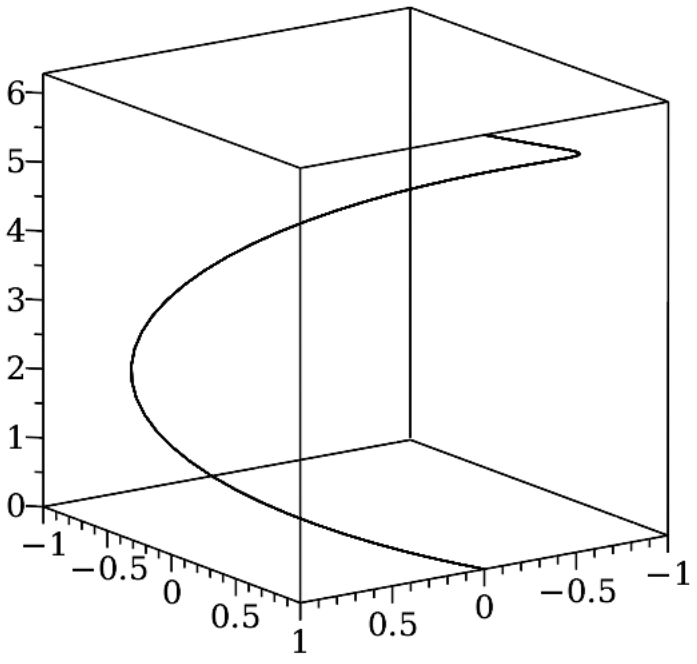
> $plot3d(x^2+y^3, x=-2..2, y=-2..2)$



>

We can plot parametric plots too

> $plot3d([\sin(t), \cos(t), t], t=0..2\,\mathrm{Pi})$



Sometimes it is useful to "thicken" the curve in 3d in order to be able to see it better. This can be done with tubeplot

> *tubeplot*([sin(*t*), cos(*t*), *t*], *t* = 0..2 Pi)

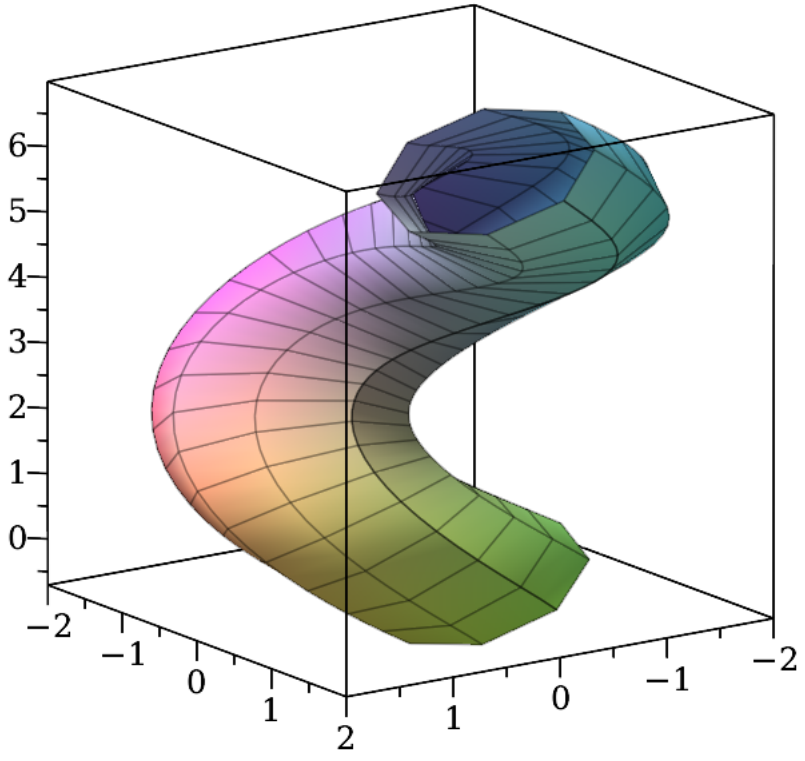
tubeplot([sin(*t*), cos(*t*), *t*], *t* = 0..2 π) **(2.1)**

But tubeplot lives in an extra library contianing many other variations of plots. We can load this using the with command:

> *with*(*plots*)

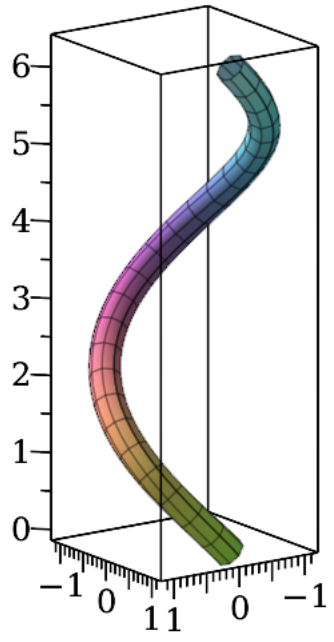
[*animate*, *animate3d*, *animatecurve*, *arrow*, *changecoords*, *complexplot*, *complexplot3d*, *conformal*, *conformal3d*, *contourplot*, *contourplot3d*, *coordplot*, *coordplot3d*, *densityplot*, *display*, *dualaxisplot*, *fieldplot*, *fieldplot3d*, *gradplot*, *gradplot3d*, *implicitplot*, *implicitplot3d*, *inequal*, *interactive*, *interactiveparams*, *intersectplot*, *listcontplot*, *listcontplot3d*, *listdensityplot*, *listplot*, *listplot3d*, *loglogplot*, *logplot*, *matrixplot*, *multiple*, *odeplot*, *pareto*, *plotcompare*, *pointplot*, *pointplot3d*, *polarplot*, *polygonplot*, *polygonplot3d*, *polyhedra_supported*, *polyhedraplot*, *rootlocus*, *semilogplot*, *setcolors*, *setoptions*, *setoptions3d*, *shadebetween*, *spacecurve*, *sparsematrixplot*, *surfdata*, *textplot*, *textplot3d*, *tubeplot*] **(2.2)**

> *tubeplot*([sin(*t*), cos(*t*), *t*], *t* = 0..2 Pi)

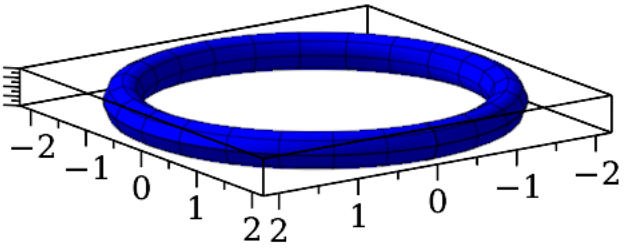


This is ugly... let's make the tube have a small radius

```
> tube := tubeplot( [sin(t), cos(t), t], t = 0..2 Pi, radius = .2, scaling
      = constrained)
```

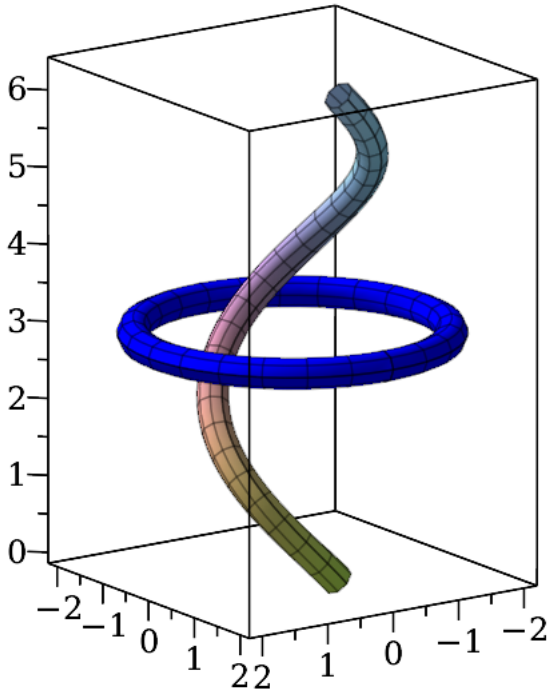



```
> circ3d := tubeplot( [2 sin(t), 2 cos(t), 3], t = 0..2 Pi, radius = .2, scaling
      = constrained, color = blue)
```



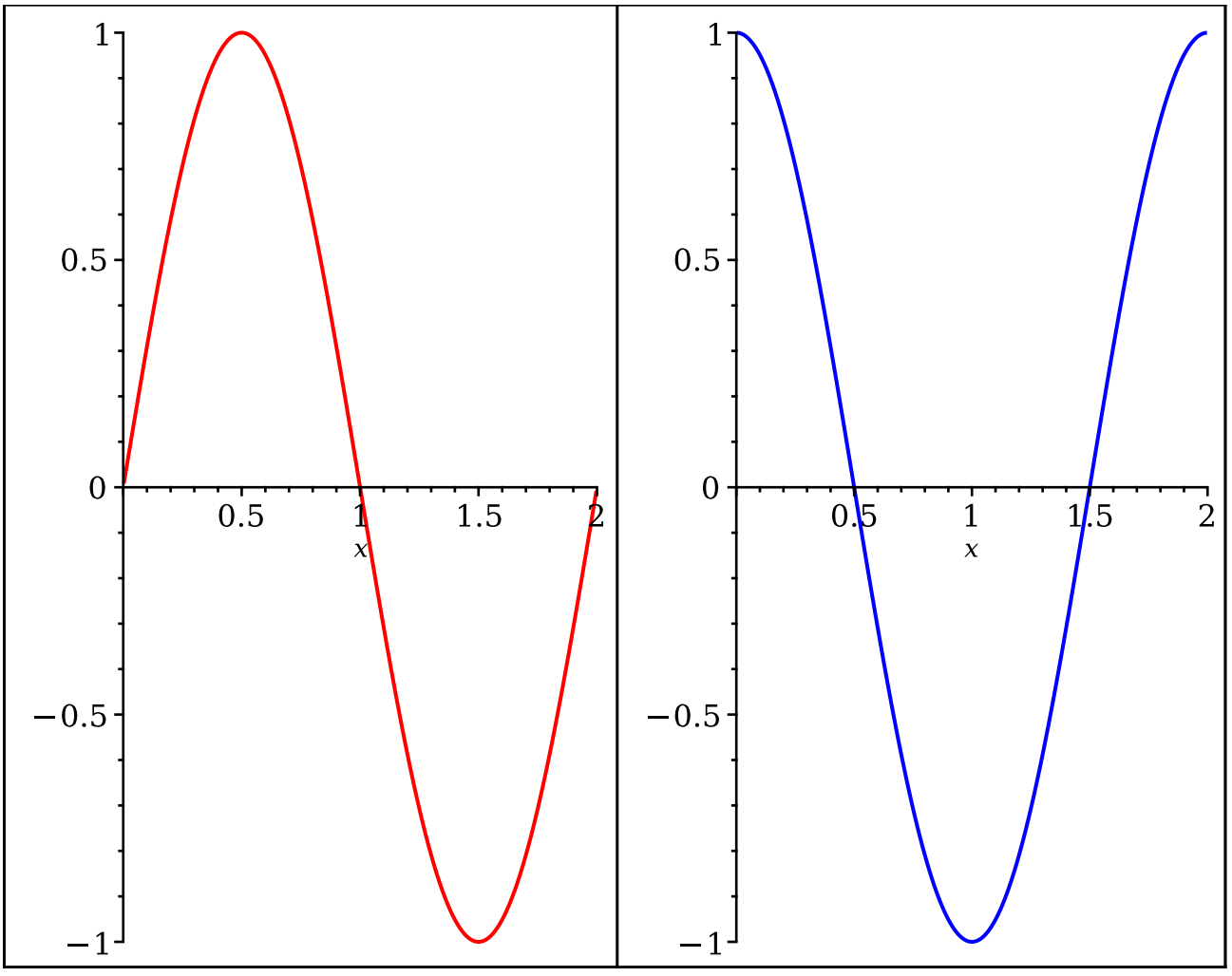
display can do many things, such as show two previously computed plots on the in the same view

```
> display([circ3d, tube], scaling = constrained)
```



the display command can also be used to show an array of plots.

```
> display(⟨plot(sin(Pi·x), x = 0..2, color = red) |
          plot(cos(Pi·x), x = 0..2, color = blue)⟩)
```



That is a row vector of two plots.

```
> vect := ⟨1, 2, 3⟩   # column vector
```

$$vect := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad (2.3)$$

```
> Rowvect := ⟨1 | 2 | 3⟩
```

$$Rowvect := \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \quad (2.4)$$

```
>
```