VPython Class 7: Plotting and Standing Waves

1. Introduction

We're changing directions this week: we won't be using the 3D graphical visualization capability of Vpython. Rather, we will be using the more conventional 2D plotting capability of the python module `matplotlib` to look at the motion of standing waves.

2. Importing modules

We won't need to import the `vpython` module today, but we will use the `scipy` module to get similar mathematical features (and more!). We will also import the `matplotlib` module for plotting. The first cell in your notebook should be

```
import scipy as sp
import matplotlib.pyplot as plt
%matplotlib notebook
```

When imported this way, every command from the `matplotlib` module must be preceded by `plt.` as a prefix, and every command from the `scipy` module by `sp.` as a prefix. There are other ways to import `matplotlib` and `scipy`. For example you could use "`from scipy import *`" like we did with Vpython, but the suggested method explicitly identifies the source of all imported functions. The last line in this cell is not actually python code — it's a 'magic command' that tells Jupyter to display the plots in the notebook, rather than popping up an external window.

3. Arrays

SciPy arrays are lists of numbers. (Some of you may have heard of the NumPy module. SciPy arrays are the same thing as NumPy arrays — SciPy is an extended set of tools for scientific computation that is built on the foundation of NumPy arrays.) You can create a four-element array as follows:

```
x = sp.array([2,4,8,16])
print(x)
```

and you can reference individual elements of the array: the first element is `x[0]`, the second is `x[1]`, etc. Try

```
print(x[0], x[2])
```

You should also try

```
print(x[4])
```

There are lots of tools for generating commonly used arrays. For example, you can make an array of 6 evenly spaced numbers starting at 0 and ending at 5 with the command

```
x = sp.linspace(0, 5, 6)
print(x)
```

Look at the `print` output: evidently `x` refers to an array of values. Now modify your command above to be `x = sp.linspace(0, 5, 11)`. Do you see what is happening here? The function `linspace` has three arguments: starting value, ending value, and the number of points.

Now here is where it gets cool. Try adding two more lines at the end of the previous cell:

```
y = x**2
print(y)
```

What does it mean to square an array? It means that `y` will also be an array, and each element in `y` is the square of the corresponding element in `x`. There was nothing special about squaring. The procedure is the same if we want to find e^x for every element in our array `x`: `y = sp.exp(x)`, or we could calculate `y = x*sp.exp(x)`. Try it! The advantage of arrays is that we don't have to build a loop to step through each value. It's all implied.

4. Plotting a function

- Define a function f1(x) that returns the cube of `x`. (Recall that function definitions start with a statelment like `def f1(x):`.)
- Create an array of 201 equally spaced values for `x` between -2 and 2.
- Create 201 values for `y` using your function `f1`:

  ```
  y = f1(x)
  ```

- Create an interactive plot with the following commands:

  ```
  plt.figure()
  plt.plot(x,y)
  ```

  If you don't get a graph, talk to an instructor. Even if you do get a graph, talk to an instructor about its interactive features.

- You can add lots of stuff to your graph. For example, try adding the following lines to the cell with the `plt.figure()` and the `plt.plot(x,y)` commands. Then re-execute the cell.

  ```
  plt.xlabel('Label for horizontal axis') # Label for horizontal axis
  plt.ylabel("Label for vertical axis")   # Label for vertical axis
  plt.title("This is the title")
  plt.axhline(0)                          # Make x-axis (horiz. line)
  plt.grid(True)                          # Turn on grid
  plt.ylim(-12,12)                        # Set scale for plot, and
                                          # turn off autoscaling
  ```

5. Standing waves

Consider a string stretched between $x = 0$ and $x = 1$ that supports waves with a velocity $v = 1$.

- Write a function that will return the spatial profile of the longest wavelength standing wave mode on the string. Set the amplitude of the wave at an antinode to be 1. Make a plot of your function.
- Generalize your function so that it includes the time dependence of the standing wave. Your function should take two inputs, `x` (position along the string) and time `t` (time). Make a plot of your function at various times that you select. Is your string oscillating as you expect?
- Generalize your function so that it takes a third input: `n`, the mode number. Make some plots of modes with $n > 1$. Do they have the time dependence you expect?
- Make a graph of the spatial profile when the two lowest frequency modes (with equal amplitudes) are *both* excited at the same time on the string. Explore the time dependence of this combination of modes.