

PHY307, Science and Computers I
Waves Lab, Part I, November 26 and December 3, 2002
(Parts I and II are due by end of lab Tuesday, December 3)

Waves: strings and stripes

Summary of what you will do: Waves are patterns that may vary in space or time. These patterns result from the interactions of neighboring pieces of physical objects: pieces of a string, parcels of moving water, or spreading chemicals, for example. These interactions lead to the formation and propagation of waves.

In the first part of this lab (Nov. 26), you will look at linear waves. The waves will be caused solely by the forces that are created when a piece of an elastic medium (such as a string or surface of a drum) is stretched. (The pieces of the medium also have inertia.) The forces act to bring the medium to an undistorted state. You will use both spatial displacements and colors to visualize waves.

In the second part of the lab, you will look at pattern formation in a more complex model. In this model, neighboring pieces of the substance communicate via chemical signals. As realized by Turing, models of such chemical waves can lead to patterns quite reminiscent of stripes and spots in living creatures.

LAB REPORT

1. Get set up. Start a lab report entitled “*MyLastNameWaves.doc*”, where *MyLastName* is your actual last name.
2. Download the program that we created in class last Thursday. It is available on the Codes page as *WaveInClass.py*. There have been only 2 changes made:
 - a. `scene.range = 0.6 * n` has been added, so that the string fits onto the screen (`scene.range` gives the distance from the camera to the center of the view.)
 - b. The `slinkievelocity` initialization (`slinkievelocity.append`) has been modified so that the initial velocity is a sin function, making for a smoother wave. (Before, `slinkievelocity` was a constant, initially.) Note that the *y*-component of the velocity depends on **i**, which is the number of the segment of the string.
3. Run the program. What do you see? How does varying the constant **k** affect what you see (try making **k** significantly smaller or larger)?
4. Review how the dynamics works in the code: the key is where the force is calculated. Each portion of the string moves according to the usual rules: rate of change of position is velocity and rate of change of velocity is acceleration. Acceleration is force divided by mass. The force is related to the difference between the positions of the neighboring bits of string (or slinkie.)
5. To see this a bit more clearly, make the string “coarser”. Reduce the number of the pieces of string to 5 and rerun the program. Describe what you see.
6. You can also visualize the forces on the string. Try the following, taking snapshots and including them in your report (continued next page):

- a. Attach little arrows to the curve. When you set up `slinkieposition` and `slinkievelocity`, also set up a new list of arrows:
 1. Call it **arrowlist**. Set it to be empty, initially, like the other lists.
 2. Then in the first **for** loop, append arrows at position **(i,0,0)**, with **shaftwidth=0.1**, **axis=(0,0,0)** and **color=color.green**.
 3. In the first **for** loop in the **while** loop, set the **pos** attribute of the *i*th arrow, **arrowlist[i]**, to be equal to **slinkieposition[i]**. This will make the arrow track the string/slinkie position.
 4. In the second **for** loop inside of the **while** loop, set the **axis** attribute of **arrowlist[i]** to be equal to **force * n * 0.8** (we multiply by **n** as the forces are smaller in a longer wave.)
 5. Now the arrows show the direction of the force on that bit of string and they have a position starting at that bit of string. Run your program, taking snapshots and summarizing in words what you see.
 6. Increase **n** to a larger, smoother value.
7. Try two other shapes for the initial velocity profile:
 - a. Add in a **z**-component to the velocity. Set the **z** part of the velocity to be some function of **i**, like **2*sin(i/(n-1))*4.*3.1415926535**).
 - b. Set the **z**-component back to zero and set the **y** component to have value **(i-n/2.)*5.*exp(-(i-n/2.)/n*16.))**2)**. Try more points – like **n=100**. Also reduce the size of the arrow axes relative to the force (replace 0.8 with 0.1). Here, report on how the speed of the pulses change as you change **k**. Describe the shapes of the pulses and how they interact.
8. Finally, try a two dimensional simulation. Load **coolwall2.py** from the Codes page and run it. Try clicking on the wall. This program is essentially like the wave on a string module, but keeping track of the positions of the elements requires keeping track of lists of lists.
 - a. Run this program.
 - b. Try different values of **N** that run in reasonable amounts of time.
 - c. What happens when you vary **k**?