Physics 465: Computing Project 3

October 2, 2012

Computing Probabilities in time for the Infinite Square Well

This project capitalizes on the first two projects by introducing a “real” system for which we can find exact solutions to the Schrödinger equation and simulate the behavior of the system over time. Two beneficial side effects of completing this project are that you’ll learn how to make real time graphs in vpython and you’ll learn some tricks for computing probabilities numerically with discretized (sampled) wave functions.

The Problem

Problem 2.8 (page 40, Griffiths, 2nd ed.) reads: A particle of mass m in the infinite square well (of width a) starts out in the left half of the well, and is (at $t = 0$) equally likely to be found at any point in that region.

(a) What is the inital wave function $\Psi(x, 0)$? (Assume that it is real. Don’t forget to normalize it.)

(b) What is the probability that a measurement of the energy would yield the value $\pi^2\hbar^2/2ma^2$?

We worked out the normalization and coefficients for HW 2.8 in class. If you’re careful you should find the wave function is:

$$\Psi(x,0) = \begin{cases} \sqrt{(2/a)} & \text{if } 0 \le x \le a/2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

and the expansion coefficients for the stationary state expansion:

$$\Psi(x,0) = \sum_{n=1}^{\infty} c_n \psi_n(x) \tag{2}$$

are:

$$c_n = \frac{2}{n\pi}\left(1 - \cos(\frac{n\pi}{2})\right) \tag{3}$$

where the $\psi_n(x)$ are the stationary state solutions

$$\psi_n(x) = \sqrt{(2/a)}\sin(\frac{n\pi}{a}x) \tag{4}$$

It turns out the "probability of being in state n" (see Griffiths comments about this phrase) is just $|c_n|^2$, so we have the answer to part b as well.

At later times we saw that each stationary state component of the wavefunction evolves simply with an $e^{-i\omega_n t}$ phase factor according to the energy $\hbar\omega_n$ of each state (where ω_n is determined by the kinetic energy in the nth state, $p_n^2/2m = (\hbar k_n)^2/2m = n^2\hbar\omega_1$.), so that the net wavefunction is simply:

$$\Psi(x,t) = \sum_{n=1}^{\infty} c_n\psi_n(x)e^{-i\omega_n t} \tag{5}$$

or, upon substitution (watch out! It gets ugly...)

$$\Psi(x,t) = \sum_{n=1}^{\infty} \frac{2}{n\pi}\left(1 - \cos(\frac{n\pi}{2})\right)\sqrt{(2/a)}\sin(\frac{n\pi}{a}x)e^{-in^2\omega_1 t} \tag{6}$$

So.. what are we supposed to do?

Your mission is to produce three deliverables:

1) A 3D representation of the complex wavefunction over a time interval corresponding to two full cycles of the ground state component of the wavefunction. At the beginning your 3D representation should look something like Fig. 1.

Later on your display should look something like this Fig. 2.

Please include at least one screen capture of your wavefunction and include it in your report.

2) A graph of the probability of finding the particle in the range $x < (a/2)$ as a function of the phase of ψ_1 for a period of time equivalent to two full cycle of the ground state component of the wavefunction. With my setup I get a graph that looks like Fig. 3.

3) A graph of the expectation value of $\langle x\rangle$ as a function of time, for the same time period as (2) above. If you really want to impress me you can include graphs of $\langle x\rangle \pm \sigma$ as well! Your graph might look something like Fig. 4.

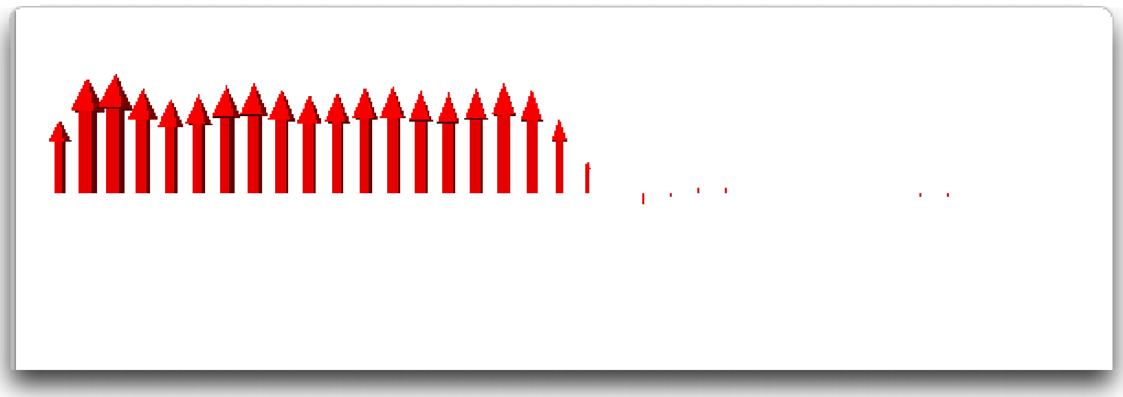
Figure 1: The initial wavefunction, 40 arrows, a=6.0

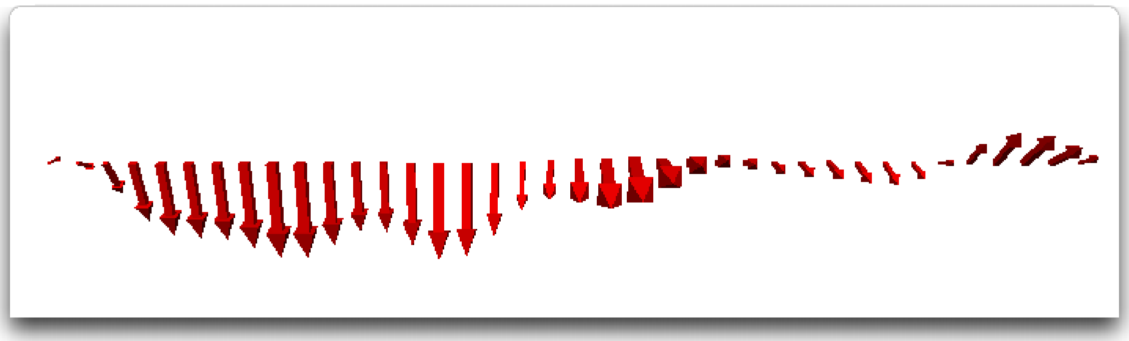
Figure 2: The wavefunction later, 40 arrows, a=6.0

(Note: If you set ω_1=1, then your time units will essentially be equal to the phase of ψ_1. You can always choose other time units, but this will be simplest.)

The questions that need to be answered as part of the report require you to reflect on this graph, so it's important to get it right. If you graph doesn't look pretty close to mine, please ask!

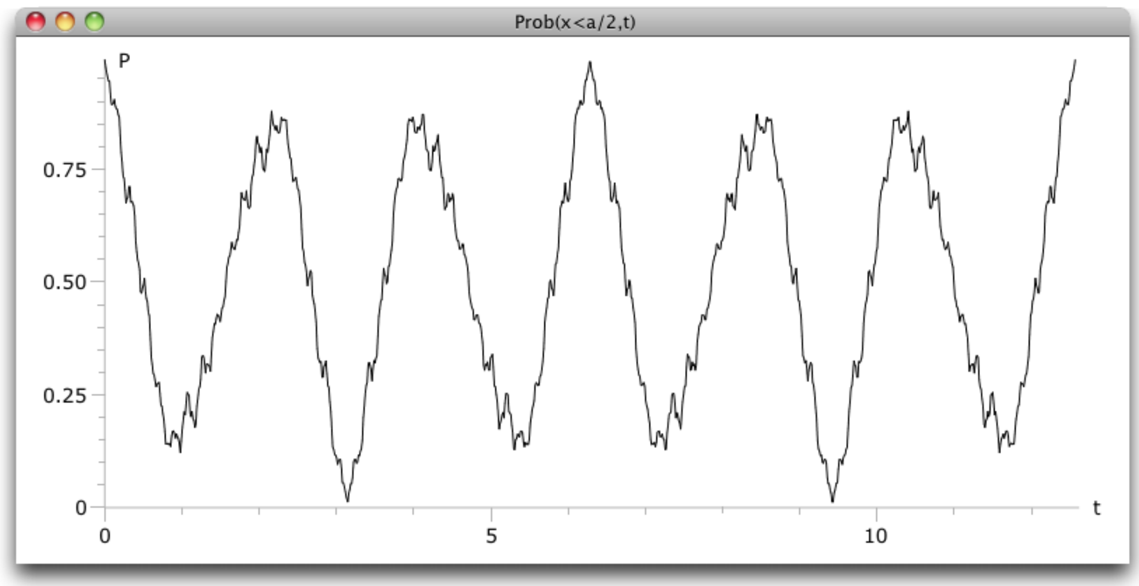

Figure 3: The probability of $x < (a/2)$ (as a function of phase of ψ_1)

Graphing in python

There are lots of ways to make graphs in python, but when using vpython (the visual module) the easiest way is to use the gdisplay and gcurve. At the beginning of your program you can just add:

```
from visual.graph import *

gd = gdisplay(title="Prob(x<a/2,t)", xtitle="t", ytitle="P")
gr = gcurve(color=color.white)
```

This will create an additional graphing window where your graph will appear. You can add points to the graph by calling the 'plot' method of the gcurve object. Here's a simple complete example:

```
from visual import *
from visual.graph import *
```

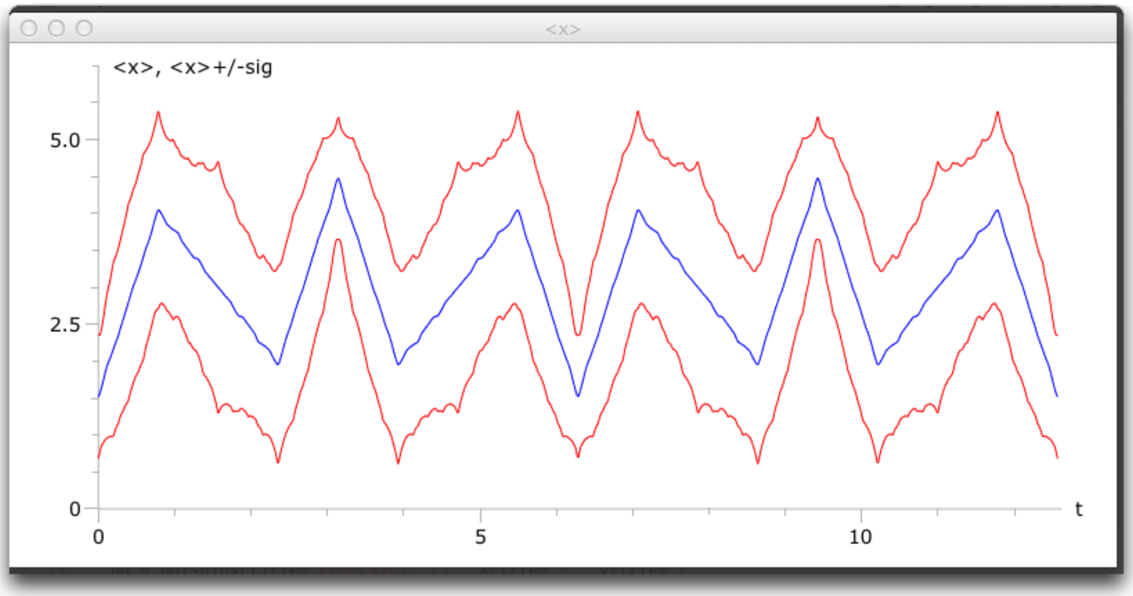

Figure 4: $\langle x \rangle \pm \sigma$ as a function of phase of ψ_1

```
gd = gdisplay(title="A demo of a graph", xtitle="t", ytitle="x")
gr = gcurve(color=color.red)

A=10.0
s = sphere(pos=(A, 0, 0), color=color.blue)
t=0.0
dt=0.01

scene.autoscale=False
while True:
    rate(100)
    t=t+dt
    s.pos.x = A*cos(t)
    gr.plot(pos=(t, s.pos.x))
```

If you run this code, you'll see that the graph simply shows the value of the x coordinate of the sphere over time. You'll need similar code in your project to graph the probability vs. time.

Calculating Probabilities

The easiest way to compute a probability is to use the power, slicing and sum features of vpython (numpy really) arrays.

If you have a real array you can add all the elements like so:

```
from visual import *

x = array([1,2,3,4])
print x.sum()       # prints out the value "10"
print x**2          # prints out "[ 1  4  9 16]"
print (x**2).sum()  # prints out the value "30"
```

Slicing is a way to get a portion of an array, so e.g., to get the first half of an array you'd say:

```
from visual import *

x = array([1,2,3,4])
print x[:2].sum()      # prints out the value "3"
print x[:2]**2         # prints out "[ 1  4]"
print (x[:2]**2).sum() # prints out the value "5"
```

In general to get the first half of an array you'd use `x[:len(x)/2]` and to get the last half `x[len(x)/2:]`. There are lots of other fancy slicing tricks, but this is all we'll need for project 3.

Finally to compute the sum of the squared magnitudes of a array of complex numbers you'd use all of these:

```
from visual import *

x = linspace(0, 6.0, 10)
psi = exp(1j*x)
print (abs(psi)**2).sum()      # prints out the value "10.0"
print (abs(psi[:5])**2).sum()  # prints out the value "5.0"
```

Questions

Please answer these questions at the end of your report.

1) Explain how it is that the probability of being on the left half of the square well returns to 1 every time the ground state component of the wavefunction completes a full cycle.

2) What happens when the ground state wavefunction completes half a cycle? Explain why this occurs.

3) Are there other initial wavefunctions you could choose that would also start the particle out on the left half of the well? Would the results be different from what you found using the initial wavefunction from Eq.1?