Physics 468: Computing Project 1

May 24, 2013

Two non-interacting particles in a square well

Last semester we studied the behavior of a single particle in a 2-D infinite square well (ISW, computing project 8). For this project I'd like to start with that code but modify it to handle *two* particles in a 1-D ISW.

Bosons, Fermions and Distinguishable particles

In CP8 we had the following time dependent solution to the SWE for a single particle in a 2-D well:

$$\Psi(x,y,t) = \langle xy| e^{-i\frac{\hat{H}}{\hbar}t} \sum_{nm} |nm\rangle \langle nm|\psi(0)\rangle = \sum_{nm} c_{nm}\psi_{nm}(x,y)e^{-i\frac{E_{nm}}{\hbar}t} \tag{1}$$

where

$$\psi_{n_x m_y}(x,y) = \frac{2}{a}\sin(\frac{n_x \pi x}{a})\sin(\frac{m_y \pi y}{a}) \tag{2}$$

Now we'd like to study *two* particles in a 1-D well. How does the situation change? As long as the particles don't interact and they are distinguishable you can just change the x, y in various places to x_1, x_2 since the SWE of two non-interacting particles in 1-D is formally equivalent to a single particle in 2-D. However, if the particles are *indistinguishable* then there is an additional complication. Indistinguishable particles come in two flavors: *bosons* and *fermions.* We'll see that their behavior is quite different. In addition to needing to solve the SWE in the usual way, boson and fermion wavefunctions also need to be symmetric and anti-symmetric WRT particle exchange respectively. In other words:

$$\psi(x_1,x_2) = \pm\psi(x_2,x_1) \tag{3}$$

where the + is for bosons and the − is for fermions. This will have an impact on our code and bookkeeping for this project! For example, in CP8 we had code like this:

```
for nx in NX:
    for my in NY:
        psinm = sin(nx*pi*x/a)*sin(my*pi*y/a)      # compute the n,m energy eigenstate
        psinm = psinm/sqrt((abs(psinm)**2).sum())  # normalize it.
        eigenstates[(nx,my)] = psinm
```

for this project the wavefunction will depend on the nature of the particles, something like this:

```
for n1 in N1:
    for n2 in N2:
        if FERMIONS:
            psinm = sin(n1*pi*x1/a)*sin(n2*pi*x2/a) - sin(n2*pi*x1/a)*sin(n1*pi*x2/a)
        elif BOSONS:
            psinm = sin(n1*pi*x1/a)*sin(n2*pi*x2/a) + sin(n2*pi*x1/a)*sin(n1*pi*x2/a)
        else:
            psinm = sin(n1*pi*x1/a)*sin(n2*pi*x2/a)

        if (FERMIONS and (n1 != n2)) or (not FERMIONS):
            psinm = psinm/sqrt((abs(psinm)**2).sum())  # normalize it.
            eigenstates[(n1,n2)] = psinm
```

Also, the code that computes the fourier coefficients will need to change a bit:

```
for nmPair in eigenstates.keys():
    n1, n2 = nmPair
    psinm = eigenstates[nmPair]                         # get nth basis
    cn1n2 = ((psi0*psinm).sum())                        # compute fourier coef.
    coefs[nmPair] = cn1n2                               # save it.
    omega = omega0*(n1**2+n2**2)                        # get omega for nmPair,
    omegas[nmPair] = omega                              # save it.
```

But.. aside from these differences, the projects are practically identical. Modify your CP8.py to model the behavior of two non-interacting particles in a 1-D well and add some constants at the beginning that determine whether the particles are bosons, fermions or distinguishable. Once you get your program working, please answer the following questions:

Questions

Please answer these questions at the end of your report.

1) Starting in a state where either particle can be found with equal probability between 0 and $a/2$ how do the distinguishable, bosonic and fermionic states compare? What is the trouble with the fermion state with this initial condition?

2) Change the initial condition so that the one particle is between 0 and $a/4$ and the other is between $a/4$ and $a/2$. Describe the behavior of the distinguishable, bosonic and fermionic systems in this case. Why does the fermionic case work better now?