480/905: Activities 5

Grade: check (fix marked probs to upgrade)

*Online handouts:* eigen\_tridiagonal.cpp printout, eigen\_basis.cpp printout, harmonic\_oscillator.cpp printout, nan\_test.cpp printout, and square\_well.nb Mathematica notebook.

*Your goals for these activities:*

* Spend no more than 20 mins finishing Activities 4
* Take a first look at your shell start-up file.
* See some examples of nan's and inf's.
* Use the eigen\_tridiagonal program to look at eigenvalues of the harmonic oscillator potential.
* Find the lowest bound-state eigenvalues of two familiar potentials using the eigen\_basis program.
* Examine (and try to understand) how the accuracy of your results depends on the size of the harmonic oscillator basis and the choice of the basis parameter b.

Optional: Aliases in Your Start-up Shell

The shell is the program you type to at the command line. If you are using Linux on a Department account, you are most likely using tcsh. If you are using Cygwin or Ubuntu or a Mac, you are probably using bash. There are many good things to learn about the shells. Today we'll just learn about aliases. Just follow the corresponding instructions for bash (tcsh).

* **Bash (tcsh) aliases.** The .bashrc (or .cshrc.more) file sits in your home directory and is "sourced" when you start an interactive terminal. Take a look at the sample .bashrc (or .cshrc.more) file printout. These files, without the ".", are in the session 5 zip file. Copy any aliases of interest to your own .bashrc (or .cshrc.more) or the whole thing using cp bashrc ~/.bashrc  (or cp cshrc.more ~/.cshrc.more) if you don't have one already. Activate the changes with the command: source ~/.bashrc (or source ~/.cshrc). *Try some aliases such as*ll*(for "long listing") and*df*, which shows info about the disk file system. Do they work? Questions?*
* *Create your own alias.* E.g., you could make an alias to change from your start-up directory to your PHY480 working directory. Be creative! *What line did you add to the start-up file?*

Nan's and Inf's

Just a quickie: Take a look at nan\_test.cpp, use make\_nan\_test to create nan\_test and then run it.

* *What do you think the result of 0.\*(1./0.) will be? Predict and then modify the code to check it out. [Note: Use the variables "numerator" and "denominator" to calculate (1./0.).]*

**I predicted that the answer would be 0, but it turns out the answer is nan when I first compiled it. Then when I added 0 after the decimal point, it gave me an inf as the answer.**

Bound States by Matrix Diagonalization in Coordinate Representation

The program eigen\_tridiagonal.cpp uses the GSL library routines explored in eigen\_test.cpp (session 4) to find the eigenvalues and lowest eigenvector of the harmonic oscillator using a method described in the Session 5 notes. With the units here, the lowest eigenvalue should be (3/2)hbar-omega = 1.5 (read comments in code!).

1. Using make\_eigen\_tridiagonal, compile and run the code a few times to see how it works. Try various values of Rmax and N such as Rmax=3, N=20 or 50 (*make a chart here of Rmax, N and the two lowest eigenvalues for five pairs of Rmax,N*). *How can you verify that the code is working?*  
     
     
   **For Rmax = 3 & N = 50**

**eigenvalue 1 = 1.684067e+00**

**eigenvalue 2 = 5.113673e+00**

**For Rmax = 3 & N = 20**

**eigenvalue 1 = 1.681778e+00**

**eigenvalue 2 = 5.083030e+00**

**For Rmax = 4 & N = 60**

**eigenvalue 1 = 1.513140e+00**

**eigenvalue 2 = 3.778727e+00**

**For Rmax = 5 & N = 70**

**eigenvalue 1 = 1.499924e+00**

**eigenvalue 2 = 3.517728e+00**

**For Rmax = 6 & N = 89**

**eigenvalue 1 = 1.499647e+00**

**eigenvalue 2 = 3.498593e+00**

**The code was working because it printed out a list of multiple eigenvalues**

1. Change the code so that only the lowest few eigenvalues are printed out. Look at the output file eigen\_tridiagonal.dat and plot it with gnuplot. *Sketch your plot. What is this function? The value at r=0 is not given; what should it be?*

**The u( r ) is the wavefunction produced from the matrix eigenvalue computation and I’m assuming that the value of r = 0 is 0.**

1. You need to pick a reasonable value of Rmax. *Justify your choice based on the eigen\_tridiagonal.dat plot:*

**I choose the most reasonable value of Rmax to be 10 because if the if the Rmax value is below 10, the descent of the graph is not completely shown since when Rmax=10, the flat line of the wavefunction is shown (the completing of the trajectory).**

1. For your choice of Rmax, try N = 4,8,16,32,64,128,256,512,1024 (you could add a loop to calculate these). *How does the relative error for the lowest eigenvalue scale with N?* Sketch the plot to validate your answer.

**It scales similar to the first graph shown in the session 5 notes**

1. *Explain the slope you found based on the approximation to the second derivative.*

**From the first to the second point the slope was a negative value of -1.512 and then from the third point, the slope becomes -2.081. The value of the slope connects with the approximation for the derivative because of the formula of the slope is equal to the method of finding the derivative.**

1. *Bonus: repeat with Rmax = 4 and explain what happens to your graph.*

**It sort of plateaus at approx. y =2**

Bound States from Diagonalizing the Hamiltonian in a Basis

The program in eigen\_basis.cpp uses the GSL library routines to diagonalize (i.e., to find the eigenvalues and eigenvectors) a Hamiltonian matrix in a basis of harmonic oscillator wave functions. You may want to refer to the GSL handout on eigensystems (there is also a printout of eigen\_basis.cpp).

The eigen\_basis program uses units with the particle mass=1 and hbar=1. The program asks you to choose

* a potential (Coulomb or square well);
* the parameter b for the harmonic oscillator basis (see harmonic\_oscillator.cpp for the definition);
* the number of basis states to use.

The parameters of the potentials are fixed in the code. The eigenvalues for the Hamiltonian matrix are written to the terminal sorted in numerical order (as opposed to absolute-value sorting, which was used in eigen\_test.cpp). The corresponding eigenvectors are generated but are not printed out (that is, the print statements are commented out).

The Coulomb potential is defined with Ze2=1, which means that the Bohr radius is also unity. This means that the exact bound energy levels are given by En = -1/2n2, with n=1,2,...  
The square well potential is defined with radius R=1 and depth V0 = 50. *You should find that there are three bound states.*

Here are some subgoals. There won't be time for everything (as usual) but you'll have a chance to finish them as part of a future assignment.

1. Run the Mathematica notebook square\_well.nb (make sure you understand what it is doing; e.g, look up FindRoot in the Help Browser). *Find the bound-state energies for the square well parameters used here (you need to change the notebook parameters!).*

**The bound-state energies are:**

**, and**

1. Compile and link the code eigen\_basis using make\_eigen\_basis. This also compiles harmonic\_oscillator.cpp. Run it a few times with each of the potentials to get familiar with it. If you try too large a basis size, the run time may be too long (so start small!). Look through the printout to see the basic idea of how the code works and find where the equation for the matrix element is implemented.
2. *Based on the "exact" results from Mathematica, which of the approximate eigenvalue(s) for the square well are most reliable? Why do you think this is?*

**The most reliable eigenvalue from the Mathematica results is because as I ran the eigen\_basis code, the eigenvalues produces are more nearer to that value**

1. *Considering all three of the lowest eigenvalues, which are calculated most effectively, those of the Coulomb potential or the square well potential? Can you explain your observation?*

**The square well potential are calculated more effectively because the states in Mathematica are more likely to be reproduced in the compiler using that method since the square well method produces more negative eigenvalues**

1. You have under your control the size of the basis (i.e., the dimension of the matrix) and the harmonic oscillator parameter b (see harmonic\_oscillator.cpp for the definition). For a fixed basis size (pick one that reproduces the ground state reasonably), how do you find the optimum b? (Hint: think gnuplot!) *Can you qualitatively (or semi-quantitatively) account for your result? (Think about the potentials and guess what the lowest wave functions will look like and what changes about the basis when the harmonic oscillator parameter b is changed.)*

**As the b value is increased, the eigenvalues will have an increase in value because when I ran 0.5, 0.7 and 1 as my b values, the eigenvalues become more likely positive in value**

1. *If you now fix b (if you have time you can consider two or three different values in turn), how can you find how the accuracy of the ground state energy scales with the basis size?* Make an appropriate plot.
2. Look at the code. *How could you make it more efficient? (What do you think is the limiting factor based on the scaling of the time with the size of the basis?) For example, could you speed it up by almost a factor of two?* (Hint, hint!)
3. for (int i = 0; i < dimension; i++)
4. {
5. for (int j = 0; j < dimension; j++)

**I could change i++ and j++ to i\*\*2 and j\*\*2**

1. *(For a later HW set, so no need to finish.) How would you find the wave function that corresponds to a given state (e.g., the ground state)?* Add code to generate the lowest wave function for the lowest bound state (hint: it involves the eigenvector).