

Physics 780.20: Assignment #3

These exercises are follow-ups to class and background-note discussion of Richardson extrapolation (session 4), eigensystems (session 5), and adaptive numerical routines (the last one is a bonus problem). These involve modifications to the `derivative_test.cpp` code and the `eigen_basis.cpp` code. Please ask questions! It is due at the end of the day (midnight) on Saturday, February 11.

To “hand in” the assignment, email me (at `furnstahl.1@osu.edu`) a “tarball” (see below) of your program and makefile (with the answers to any questions in the comments), a postscript file of the log-log plot, and a Mathematica notebook with the command(s) and result. (Send the tarball as an attachment.) Use C++ and any plotting program you want (gnuplot recommended for now). *Check the 780.20 webpage for suggestions and hints if you get stuck.* Please give feedback early and often.

To make a tarball, create a directory with the name: `your_name_ps3` (replace “your_name” by your last name!), e.g., `mkdir furnstahl_ps3`

Copy all of the files to pack up into that directory. Then give the command:

```
tar cfvz furnstahl_ps3.tarz furnstahl_ps3
```

and you’re done! The command “`tar tfvz furnstahl_ps3.tarz`” lists the contents, so you can check that you’ve packed all the files correctly. [Note: The “c” is for “create a new archive”, the “f” is for file and goes with the archive name `furnstahl_ps3.tarz`, “v” is for verbose mode (say what is going on), and “z” means to compress the tar file with gzip (tar stands for “tape archive”).]

1. It’s time to start thinking about a 780.20 project. Please send email to `furnstahl.1@osu.edu` with a (brief!) description of your ideas for a project. They can be very vague at this point; we will refine them as the quarter progresses! See the 780 webpage for some past project descriptions.
2. Add a subroutine to take the Richardson extrapolation used in the “`extrap_diff`” subroutine one step further. That is, `extrap_diff` calls `central_diff` with two different values of h and then combines them to extrapolate to smaller h (leading to an error proportional to h^4). Now write a new routine (called `extrap_diff2`) that calls `extrap_diff` with two different values of h and combines them appropriately to get a still steeper dependence of the error on h . Verify the result by making an error plot (you may want to increase the starting value of h to 0.5). [A new version of `derivative_test.cpp` with `extrap_diff` explicitly

written with `central_diff` is available from the 780 homepage.]

3. Modify `eigen_basis.cpp` so that you can print out (to a file is best!) the approximate wave function corresponding to a given state (e.g., the ground state or the first excited state). Plot the exact ground state wave function and the approximate wave function (as a function of r) for one of the potentials (your choice; I like the Coulomb best!) with a fixed b (your choice) for basis sizes of 1, 5, 10, and 20. Comment on the nature of the convergence and speculate about choosing b based on your plots. Make sure that the wave functions are normalized.
4. (BONUS) Make the calculation using the central difference method *adaptive*. That is, you specify the function but don't specify the value of h . Instead, your program determines (or, more precisely, estimates) the optimal value of h automatically and uses that. Compare the h chosen by your program for the function described above to the value you would select based on the error plot.
5. (BONUS) Devise a measure of how close the approximate wave function is to the exact wave function and determine how this measure scales with the basis size D .