

## Physics 780.20: Assignment #2

This assignment is a follow-up to Sessions 3 and 4. It is intended to give you additional practice in the basic tasks from class (like coding algorithms), that we'll need repeatedly and build upon, and to verify to me that you can do them. Please ask questions! It is due at the end of the day (midnight) on Friday, February 3.

Use Mercurial to “hand in” the assignment by pushing your files to `bitbucket.org`. Put them in your repository directory (e.g., `780_furnstahl_richard`) but in a separate directory from your files for Assignment #1 (move files around as needed). Recall the sequence of commands to update your files (you don't need the comments after the #):

```
$ hg add      # add files to the repository list
$ hg commit  # take a snapshot of the current state of the files
$ hg push    # send the updates to bitbucket.org.
```

Note that if you have cloned your repository from the one on `bitbucket.org`, you can type `hg push` without any web address. If you push without committing, or commit without adding, you will get a message that there are no changes. You know that the commit has worked if an editor pops up and you leave a descriptive message. You may find that the `.hgrc` file that you made in your `$HOME` directory is not on the computer you are using now (if not, you'll get an error message like `abort: no username supplied`). Recall that you need to put in this file your own version of:

```
[ui]
username = furnstahl <furnstahl.1@osu.edu>
editor = nedit
```

Alternatively, you can add the same lines in the `hgrc` file (no period!) in the `.hg` directory in your repository (use `ls -a` to see it) and you will avoid the problem of different `$HOME` directories.

Include in your repository makefiles (one for each program), C++ programs (with the answers to any questions in the comments at the top) and a postscript file of the log-log plot (made with a gnuplot plot file). *It is required that your code have appropriate comments.* Comment your codes with your name, email, AND revision history, as in the example codes from class.

The problems here are parts of the same computational problem of calculating integrals. You can combine them in any way that is convenient (i.e., you don't need separate main programs for each sub-section). Note that the session notes have lots of help info and that a sample code for GSL integration is included in the session 4 zip archive. *Check the 780.20 webpage for suggestions and hints.* Please ask questions and give feedback early and often.

### 1. Follow-up/Completion of Session 3

- (a) Write routines to do integration of any user-supplied integrand function over a given (finite) interval in *double precision* using Simpson's rule, the Milne integration rule (see the Session 4 notes), and an appropriate integration routine from the GSL. (You may choose to base your Simpson's rule routine on the one from class. Check the hints for other suggestions.) Test your routines with a non-trivial test integral (*choose an integrand we haven't used yet that is much more complicated than a polynomial*). The test program and routines should be in separate files and you should create a makefile to compile them.
- (b) Extend your program to generate an appropriate data file and plot (with a gnuplot plot file that makes a postscript file) to do an error analysis of the integration methods from the first part.
- (c) Find *graphically* (i.e., by analyzing the plot from the last section) the optimum number of points to use for the Milne integration rule in double precision and explain how this makes sense analytically (based on the discussion of errors in the notes).
- (d) (*BONUS to get a plus*) Evaluate one of the singular integrals from the handout (1094 Session 4) directly using Simpson's or Milne and a GSL routine, and then compare to the answer found by applying a method discussed in the handout or session notes (or in Numerical Recipes).

2. **Empirical error analysis.** In real-life research problems, we may not have an exact answer to test our code with. How could you analyze your integration results if you didn't know the exact answer? Apply the method from the section on "Empirical Error Analysis" in the Session 4 background notes to analyze one of the integration rules (your choice of rule and integral!) to find the approximation error.