Math 4500/6500 Homework #7

This homework assignment covers our notes on the Romberg integration algorithm and on adaptive integration. You are welcome to look at the code from the *Mathematica* notebooks, but when the problems say "write a piece of code to" they mean "write your own code from scratch", not "modify the code in the notebook" or "find a piece of code on the web". If you find the algebra lengthy or irritating (which is pretty likely), you are encouraged to use *Mathematica* to do it.

1. Using the Romberg integration scheme, find $R(1,1)$ in the approximation

$$R(1,1) \sim \int_0^1 e^{-(10x)^2}\, dx$$

   You can do the arithmetic in *Mathematica*.

2. Sometimes, you can change the accuracy of a numerical result by making a clever symbolic transformation of the problem to be solved before feeding it to the computer. For instance, we can modify an integral by u-substitution. First, use the substitution $u = x^2$ to verify that

$$\int_0^1 \frac{e^x}{\sqrt{x}}\, dx = 2\int_0^1 e^{x^2}\, dx.$$

   Which integral do you think is more likely to produce a more accurate result using the Romberg integration scheme? Why? Use the Romberg integrator from the class demonstration notebooks to check your argument. Which form actually does produce better results with Romberg integration?

3. How many evaluations of the integrand are required to fill the Romberg array with n rows and n columns?

4. Suppose that the function $f(x)$ is smooth (it has continuous derivatives of any order). Find a bound on the error in the $R(n,m)$ term in the Romberg array

$$\left| \int_a^b f(x)\, dx - R(n,m) \right|$$

   in terms of h.

5. Use *Mathematica* to test the Romberg integration scheme on the *terrible, horrible, no-good very bad* function $\sqrt{x}$, integrated from $[0,1]$. How does the Romberg answer compare to the true value of the integral? *Why* is the function terrible, horrible, no-good, and very bad?

6. Use Romberg integration to approximate π by proving that

$$8\int_0^{\frac{1}{\sqrt{2}}} \left(\sqrt{1-x^2} - x\right)\, dx = \pi$$

and computing the integral numerically. It will probably help to remember that π is the area of the unit circle, and to prove the identity above by showing that the integral is really a formula for this area.

7. (Challenge) Numerical integration is a great candidate for parallel computing. Use *Mathematica*'s `ParallelMap` function to parallelize your implementation of the trapezoid rule by splitting the domain of integration into pieces and computing the integral over each in parallel. Give examples which show that your parallel implementation is faster.

8. (Challenge + Extra Credit) Really impress your professor and prepare for your future career by repeating the above exercise on the GPU with `OpenCLLink`. Give timing examples which show that the OpenCL implementation of the trapezoid rule is ridiculously fast compared to both the straight *Mathematica* version and the `ParallelMap` version.