Phys 512 Problem Set 2

Due on github Monday Sep 19 at 11:59 PM. You may discuss problems, but everyone must write their own code.

Problem 1: Write a recursive variable step size integrator like the one we wrote in class that does **NOT** call $f(x)$ multiple times for the same x. The function prototype should be

```
def integrate_adaptive(fun,a,b,tol,extra=None):}
```

where extra contains the information a sub-call needs from preceeding calls. On the initial call, extra should be None, so the integrator knows it is starting off. For a gaussian integrated from -20 to 20 with an overall accuracy of 1e-6, how many function calls do you save vs. the lazy way we wrote it in class? What is your actual accuracy compared to the true answer?

Problem 2: a) Write a function that models the log base 2 of x valid from 0.5 to 1 to an accuracy in the region better than 10^{-6}. Please use a truncated Chebyshev polynomial fit to do this - you can use np.polynomial.chebyshev.chebfit. How many terms do you need? You should use many x/y values and fit to some high order, then only keep the terms you think you'll need and drop the rest. Make sure also that you rescale the x-range you use to go from -1 to 1 before calling chebfit.

Once you have the Chebyshev expansion for 0.5 to 1, write a routine called mylog2 that will take the natural log of any positive number. Hint - you will want to use the routine np.frexp for this, which breaks up a floating point number into its mantissa and exponent. Relatedly, having your routine natively produce the log base 2 will simplify life. Also feel free to use np.polynomial.chebyshev.chebval to evaluate your fit. You might ask yourself if a computer ever takes a natural log directly, or if it goes through the log base 2 (near as I can tell, it's the log base 2).

If you'd like a bonus, repeat this exercise using the Legendre polynomial (or, heaven forbid, the actual Taylor series) of the same order as the Chebyshev you used. How big is the RMS error in the log compared to Chebyshev? How big is the maximum error? This hopefully gives you a flavor as to how useful Chebyshev's can be if you have to write your own function. If you enjoy this sort of problem, I'd encourage you to look at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.5177 that describes how some of these transcendental functions are actually implemented.

Problem 3: a) Write a program to solve for the decay products of U238 (refer to slides for the decay chain). You can use the ODE solver from scipy, but you'll need to set the problem up properly. Please make sure to include all the decay prodcuts in the chain. Assume you start from a sample of pure U238 (in nature, this sort of separation happens chemically when rocks are formed). Which solver would you use for this problem?

b) Plot the ratio of Pb206 to U238 as a function of time over a region where it's interesting. Does this make sense analytically? (If you look at the decay chain, all the half-lives are short compared to U238, so you can approximate the U238 decaying instantly to lead. Now plot the ratio of Thorium 230 to U234 over a region where that is interesting. Radioactive decay is frequently used to date rocks, and these results point at how you can determine the age of a uranium-bearing rock that is anywhere from thousands to billions of years old. (Of course, in this case the starting ratio of U234 to U238 would probably have already reached its long-term average when the rock was formed, but you could still use the U234/Th230 ratio under that assumption.)