Phys 512 Problem Set 1

Due on github Friday Sep 17 at 11:59 PM. Please place all materials in a subdirectory problem_sets/ps1 off of your main course repository so the TAs can find things more easily. You may discuss problems, but everyone must write their own code.

Problem 1: We saw in class how Taylor series/roundoff errors fight against each other when deciding how big a step size to use when calculating numerical derivatives. If we allow ourselves to evaluate our function f at four points ($x \pm \delta$ and $x \pm 2\delta$),
a) what should our estimate of the first derivative at x be? Rather than doing a complicated fit, I suggest thinking about how to combine the derivative from $x \pm \delta$ with the derivative from $x \pm 2\delta$ to cancel the next term in the Taylor series.
b) Now that you have your operator for the derivative, what should δ be in terms of the machine precision and various properties of the function? Show for $f(x) = \exp(x)$ and $f(x) = \exp(0.01x)$ that your estimate of the optimal δ is at least roughly correct.

Problem 2: Write a numerical differentiator with prototype:

```
def ndiff(fun,x,full=False):
```

where fun is a function and x is a value. If full is set to False, ndiff should return the numerical derivative at x. If full is True, it should return the derivative, dx, and an estimate of the error on the derivative. I suggest you use the centered derivative

$$f' \approx \frac{f(x + \mathrm{d}x) - f(x - \mathrm{d}x)}{2\mathrm{d}x}$$

Your routine should estimate the optimal dx then use that in calculating the derivative. If you're feeling ambitious, write your code so that x can be an array, not just a single number. If you do that, you may actually wish to save your code as you might use it in the future.

Problem 3: Lakeshore 670 diodes (successors to the venerable Lakeshore 470) are temperature-sensitive diodes used for a range of cryogenic temperature measurements. They are fed with a constant 10 μA current, and the voltage is read out. Lakeshore provides a chart that converts voltage to temperature, available here, or you can look at the text file I've helpfully copied and pasted (lakeshore.txt). The first column is the temperature, the second column is the corresponding voltage. If you choose to use it (you don't need to), the third column is dV/dT at that temperature. Write a routine that will take an arbitrary voltage and interpolate to return a temperature.

You should also make some sort of quantitative (but possibly rough) estimate of the error in your interpolation as well (this is a common situation where you have been presented with data and have to figure out *some* idea of how to get error estimates).

Your prototype should be:

```
def lakeshore(V,data):
```

where data is the output of

```
dat=np.loadtxt('lakeshore.txt')
```

Your code should support V being either a number or an array, and it should return the interpolated temperature and your uncertainty on the temperature.

Problem 4: Take $\cos(x)$ between $-\pi/2$ and $\pi/2$. Compare the accuracy of polynomial, cubic spline, and rational function interpolation given some modest number of points, but for fairness each method should use the same points. Now try using a Lorentzian $1/(1+x^2)$ between -1 and 1.

What should the error be for the Lorentzian from the rational function fit? Does what you got agree with your expectations when the order is higher (say n=4, m=5)? What happens if you switch from np.linalg.inv to np.linalg.pinv (which tries to deal with singular matrices. For sufficiently small eigenvalues in matrix A, pinv sets them to be 0 in A^{-1}, while they would be something very large when using regular inv)? Can you understand what has happened by looking at p and q? As a hint, think about why we had to fix the constant term in the denominator, and how that might generalize.