Computational Physics
Tutorial Problems set 2
Due Thurs. Apr 21

sieversj@ukzn.ac.za

git clone https://github.com/ukzncompphys/lecture4_2016.git

Tutorial

- (part of previous tutorial, but result is useful). Write a python script to make a vector of n evenly spaced numbers between 0 and pi/2. i.e. x[0]=0, x[-1]=pi/2 (5)

- Use this vector to integrate cos(x) from 0 to pi/2 for a range # of points using the simple method. include 10,30,100,300,1000 points between 0 and pi/2. How does error scale with # of points? (5)

- Write a python function to integrate this vector using Simpson's rule. How does error scale with # of points? How many points did we need to use in part 2 to get same accuracy as 11 points with Simpson's rule? (10)

- Plot the errors as a function of # of points using Simpson's rule and standard sum. You will want to use a log scale here - look at logplot.py in the github distribution (5)

Tutorial Problems

- Write a function that will shift an array by an arbitrary amount using a convolution (yes, I know there are easier ways to do this). The function should take 2 arguments - an array, and an amount by which to shift the array. Plot a gaussian that started in the centre of the array shifted by half the array length. (10)

- The correlation function f★g is $\int f(x)g(x+y)$. Through a similar proof, one can show f★g=ift(dft(f)*conj(dft(g))). Write a routine to take the correlation function of two arrays. Plot the correlation function of a Gaussian with itself. (10)

- Using the results of part 1 and part 2, write a routine to take the correlation function of a Gaussian (shifted by an arbitrary amount) with itself. How does the correlation function depend on the shift? Does this surprise you? (10)

- The circulant (wrap-around) nature of the dft can sometimes be problematic. Write a routine to take the convolution of two arrays *without* any danger of wrapping around. You may wish to add zeros to the end of the input arrays. (10)

Bonus Points

- the scipy module has built in integration functions in scipy.integrate. The quad routine will do numerical integrals. quad will try to put its effort where the function changes quickly.

- Look at scipy_quad_example.py, which uses scipy to integrate our Gaussian function over two different ranges. The integrals should be (almost) identical - yet they are not. Can you figure out why? (5)

- Can you write another function that will always give the correct answer to this integral? (5) Hint - you may want to do two integrals instead of one.

Tutorial Bonus Problem

- You have a sample code that calculates an FFT of an array whose length is a power of 2. Using that routine as a guideline, write an FFT routine that works on an array whose length is a power of 3 (e.g. 9, 27, 81). Verify that it gives the same answer as numpy.fft.fft (10)