Tutorial problems for Lectures 6/7/8. Due Monday May 15, 2017.

Problem 1: Let's write a direct-solving N-body solver. First, write a class that contains masses and x and y positions for a collection of particles. The class should also contain a dictionary that can contain options. Two entries in the dictionary should be the number of particles and G (gravitational constant). The class should also contain a method that calculates the potential energy of every particle, sum(G m1m2/r12) . (10)

Problem 2: Now lets add some methods to the class so we can actually use it in an n-body simulation.
(A): First, add a method to initialize the number of particles with random positions in 2-D (numpy.random.randn() will get gaussian random numbers). (5)
(B): Next, write a method that calculates the forces on the particles using a softened potential (10)
(C): Write a method that will update the particle positions and velocities using a timestep. (5)
(D): Finally, plot the total kinetic and potential energies as a function of time, and show that the total energy is approximately conserved (5)

Problem 3: Write linear least-squares code to fit sines and cosines to evenly sampled data. Pick the sines and cosines to have integer numbers of periods, so you pick 100 numbers, should have $\sin/\cos(2\pi n(0:50)/100)$. Compare your fit parameters to the FFT of the data. (NB - sin should go from 1 to 49 instead of 0 to 50. Why?) (10)

Problem 4: Take the mcmc sample code. Add a Lorentzian class $f(x) = \frac{a}{b+(x-c)^2}$. Run the fit, and show you get correct answers. (10)

Bonus: Modify the mcmc sample code to run a short chain, use that to estimate the parameter errors, and then run a longer chain using the error estimates. What is your accept fraction? Can you get this to ∼0.25 by rescaling the covariance? You can do this for either the Gaussian example or the Lorentzian from Problem 4.(10)