Homework # 4

PHY-905-003, Computational Astrophysics and Astrostatistics
Spring 2017

This assignment is due on Sunday April 30, 2017. Turn in all materials via GitHub. Include your code, plots, and answers to any questions asked in your assignment. Your code must (1) be easily readable, with good use of whitespace, clear variable names, and adequate commenting (which documents design and purpose, not mechanics) and (2) use functions to break up code into logical components. The solutions to individual problems should be saved in separate, clearly-named source files or Jupyter notebooks. Plots should have easily readable and logical axis labels and titles, and the source code and data used to generate the plots should be included. Questions should be answered in the file `ANSWERS.md` or in a LaTeX-created PDF document of a similar name (e.g., `ANSWERS.pdf`).

Part 1: The Traveling Salesman Problem is a classic problem in computer science where the focus is on optimization. The problem is as follows: Imagine there is a salesman who has to travel to N_C cities. The order is unimportant, as long as he only visits each city once on each trip, and finishes where he started. The salesman wants to keep the distance traveled (and thus travel costs) as low as possible. This problem is interesting for a variety of reasons - it applies to transportation (finding the most efficient bus routes), logistics (finding the best UPS or FedEx delivery routes for some number of packages), or in optimizing manufacturing processes to reduce cost. In astrophysics, it is akin to model parameter optimization. The Traveling Salesman Problem is extremely difficult to solve exactly for large numbers of cities - testing every possible combination of cities would take $N_C!$ individual tests. As a result, methods to quickly find a "good enough" solution are of great interest.

In this problem, you are going to use the code in `traveling_salesman.py` to minimize the traveling salesman problem for $N_C = 30$ cities to find a "good enough" solution. Your code should take some number of iterations, doing the following at each step:

1. Randomly swap N_S contiguous cities in the array of cities - in other words, if you have cities (1,2,3,4,5, 6,7,8,9,10) for $N_S = 2$ you could swap two pairs of cities to get (1,2,8,9,5, 6,7,3,4,10). N_S is typically 1, but can be more than 1 if you wish.

2. Check the total distance traversed by the salesman.

3. If the new ordering results in a shorter path, keep it. If not, throw it away.

You stop when you attain convergence. Decide what that means in this context and make sure to report it! Also keep track of the steps and the minimum distance traveled as a function of number of iteration and plot out the minimum distance as a function of step. (If you're feeling ambitious, make a movie of what it looks like as your simulation evolves toward the minimum path.)

Some questions to answer/tests to do:

1. How do you decide when to stop?

2. Does varying N_S from 1-5, but keeping it constant per run, significantly affect the rate of convergence?

3. Does allowing N_S to randomly vary from 1-5 every time step in a single run significantly speed convergence?

4. Assuming you can test one model per microsecond per core on the Blue Waters supercomputer (which has approximately 3×10^5 computational cores), how long would it take to test every single combination for $N_C = 30$, and how does that compare to how long it takes to get a reasonably good solution using your implementation?

Note that you should do at least 10 trials per run to get typical behavior!

Part 2: It has been hypothesized that the stellar initial mass function may have an environmental dependence. To test this, you are being provided with two data files containing a list of measured stellar masses for two separate star clusters – one toward the galactic anticenter (`IMF_1.dat`) and the other very close to the center of the galaxy (`IMF_2.dat`). Given the distances involved, only the masses of stars with masses greater than 2 $M_\odot$ could be measured. Using the statistical techniques that you've learned this semester, (1) argue whether or not the stellar mass distributions of the two star clusters appear to have the same mass function, and (2) quantify your confidence in this result. Clearly explain the statistical techniques that you have used for both of these things! **Complicating factors:** Note that the star clusters have substantially different total masses and somewhat different (non-zero) ages. How might this affect your approach?

Part 3: We're going to examine exoplanet data obtained using two different methods – by the transit method (using, e.g., the Kepler mission), and by using radial velocity measurements of the central star. For the transit method we will use the star KIC-8554498 (in the data file `KIC-8554498_lightcurve.dat`, which contains the relative flux of the star as a function of time since the start of the Kepler mission) and for radial velocity measurements we will use 51-Pegasi (in the data file `51_Pegasi_RadVel.dat`, which contains the Julian date, the star's radial velocity, and the error in the radial velocity).

For each of the two files, do the following:

1. Plot the light curve or radial velocity data for each of the two planetary systems, zooming in as necessary to show interesting features. Can you see by eye the evidence of exoplanets in the data? If so, what do you see?

2. Use the SciPy or Astropy Lomb-Scargle periodogram routines to identify the orbital periods of the exoplanets orbiting these stars. Include your plots, zooming in on regions as necessary. Note any interesting features in the periodogram. Are the results consistent with the expected periods of the planets orbiting 51 Peg ($P \simeq 4.23$ days) and KIC-8554498 ($P \simeq 4.78$ days)? What features do you see in these graphs, and what do you hypothesize causes them?

Part 4: We're going to continue using the data for 51-Pegasi, which was the first confirmed discovery of a planet outside of our own solar system. Assuming a period of $P = 4.230785$ days, created a stacked radial velocity curve that folds all of the orbital data you have been provided into a single period. Then, create a model for the observed radial velocity curve of a single planet in a circular orbit around a star (you may wish to start here), and use a Bayesian Markov Chain Monte Carlo algorithm to determine the planet's mass (really $M \sin(i)$), the semi-major axis of the orbit, and the orbital period. Take the mass of 51 Pegasi as a given ($M_* = 1.06\ M_\odot$), and assume reasonably broad priors for the other quantities. Use a reduced sum of squares to estimate the agreement between the model and the data. Using the data and the errors provided, can you reproduce the values for the period, semi-major axis of the orbit, and mass of the planet? Show 2D histograms for various combinations of the unknowns, and identify any degeneracies between model parameters that may exist.