Homework # 3

PHY-905-005
Computational Astrophysics and Astrostatistics
Spring 2023

This assignment is due by 11:59 p.m. on Friday April 14, 2023.

Instructions: Turn in all materials via GitHub. Include your code, plots, and answers to any questions asked in your assignment. Your code must adhere to the class coding standards and use text files rather than Jupyter notebooks. The solutions to individual parts of the assignment should be saved in separate, clearly-named subdirectories named `part_1`, `part_2`, etc. 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 a file in each subdirectory named `ANSWERS.md` or in a LaTeX-created PDF document of a similar name (e.g., `ANSWERS.pdf`).

Part 1: You have been asked to do a spectral analysis of a time series observation from an X-ray timing instrument, which can be found in the file `part_2_files/instrument_data_output.dat`. This instrument is having some technical issues and periodically has to shut down and reboot itself in order to continue working[1], so some of the data is missing and the signal has been set to zero. To help you in identifying the parts of the observation where the errors occur the instrument engineers have helpfully set an error code in the file[2]. Inspection of the signal shows that it is likely to be comprised of a small number of time-varying components. **The goal of your analysis is to identify and report the frequencies, amplitudes, and phases of the signal components and create a synthetic (noise-free) signal that you can compare to the dataset.** In addition to reporting that information (in `ANSWERS.md` or another similar file), make sure to include a comparison between your synthetic signal and the observation and a discussion of the likely source of differences between the two. Include any plots that are necessary to demonstrate your results. Also, describe in your writeup if there any features in the dataset that are missing from your synthetic signal. If you had to adjust your synthetic signal by hand in some way to make them agree better, what did you do?

Here are some suggestions/questions that might be useful when constructing your analysis and your report:

- Remember that `scipy.fftpack.fft` returns complex numbers that encode both amplitude and phase information – in other words, recall from your pre-class readings that the "real" component of the signal is in phase with (has a 0°/ 0 radian phase shift from) a reference signal, and the "imaginary" component is 90°/0.5π radians out of phase with the reference signal. An arbitrary signal at a given frequency can have both real and imaginary components, which indicates a phase shift that is neither 0° or 90° from the reference signal. Refer to this useful Stack Overflow post for one way of extracting the amplitude and phase information from the array, and think about how that would impact the synthetic signal that you're constructing.

- If the frequencies where you expect the oscillations are relatively high compared to the temporal length of the observation (i.e., $1/f_{osc} \ll T_{obs}$), do you need to analyze the entire observation to extract that information?

- Are there types of time-varying behaviors that would not show up in a Fourier analysis? If so, what are they?

[1] The engineers blame the errors on "cosmic rays," but that's what they blame everything on so you're somewhat skeptical.
[2] The comments at the top of the file describe the columns of data.

Part 2: 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 `part_2_files/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 many trials per choice of N_S to get typical behavior!

Part 3: "Stellar population synthesis" models are a broad class of astrophysical models that are used to make predictions about (and understand observations probing) phenomena relating to stellar populations. This can include understanding the metallicities, ages, star formation histories, and initial mass functions of stars in galaxies (Evolutionary population synthesis[3]) or predicting the behavior of the population of compact objects that are the end products of these stellar populations, such as their emission of gravitational waves as a function of time due to binary black hole mergers (Binary population synthesis[4]). **Your goal is to make a simple binary population synthesis code that will predict the behavior of a population of stars over time** – in particular, the rate of Type Ia and Type II supernovae and mergers of neutron stars and black holes (and the resulting gravitational wave signatures) as a function of time.

To do this, start from the stellar initial mass function sampling code you used in class, which creates a list of stars comprising a given stellar cluster given a set of IMF parameters. After you generate that code, modify it to calculate the post-main-sequence behavior of the population as a function of time. Do that using the following assumptions[5]:

- Stars have a mass-dependent main sequence lifetime of $t(M_*) = 10(M_*/M_\odot)^{-2.5}$ Gyr, with a minimum lifetime of 2 Myr (at the highest-mass end, the most massive stars take roughly a constant amount of time to deplete their cores of fuel).
- Stars have a mass-dependent binary fraction (i.e., probability of being found in a gravitationally-bound pair). 75% of stars of ≥ 8 $M_\odot$ are found in binaries; 50% of stars with lower masses are in binaries.
- Stars are most likely to form in binary systems with companions of similar masses – to that end, assign all stars in binaries to a companion whose mass is within 50% of its mass, if possible. If that's not possible, assign it to the star with the closest mass you can.
- All stars with $m_* \geq 8$ $M_\odot$ explode as Type II supernovae, which you should assume happens promptly at the end of their main sequence lifetimes; of these supernovae, 80% of these supernovae will result in neutron stars and 20% will create black holes as compact remnants.
- All stars below 8 $M_\odot$ evolve into asymptotic giant branch (AGB) stars at the end of their main sequence lifetimes, and ultimately shed their stellar envelopes and become white dwarfs. Assume this happens promptly at the end of their main sequence lifetime.
- Any binary system that has a neutron star-neutron star, black hole-black hole, or neutron star-black hole pair is assumed to merge due to gravitational radiation. This takes time, because the orbital energy needs to be removed by gravitational radiation. Assume that the delay time for any individual pair of stars to merge is a lognormal distribution with a peak of 1 Gyr and a width of 200 Myr, with the time delay occuring after the end of the main sequence lifetime of the less massive star in the pair.
- Any binary system that has a white dwarf-white dwarf pair is assumed to explode in a Type Ia supernova promptly at the end of the main sequence lifetime of the less massive star in the pair.
- Any star that is not in a binary system, or that does not conform to one of the categories described above, is assumed to end its main sequence lifetime in one of the ways described above but do nothing else that is interesting for the purpose of this discussion.

[3] As exemplified by FSPS; see Conroy's papers – Paper 1, Paper 2, Paper 3

[4] As exemplified by the COSMIC code; see Breivik et al. 2021

[5] Note that these assumptions are significantly simpler than in standard binary population synthesis codes, but capture the spirit of what those codes can do. If you're interested, talk to your instructor about the ways that a cutting edge research-grade binary population synthesis code deviates from the model you're creating here.

Given these assumptions, assume a star cluster with a total mass of 10^3 $M_\odot$, create a list of individual stars, and probabilistically assign them to binary systems. Calculate the instances of Type Ia and Type II supernovae and compact object mergers as a function of time (keeping track of the NS-NS, BH-BH, and NS-BH categories separately). Do this for a large number of clusters. Answer the following questions, including plots as necessary to make your points:

1. When do you expect to see Type Ia vs. Type II supernovae as a function of time? Does either of these quantities vary significantly between individual star clusters, and why?
2. When do you expect to see gravitational wave signatures coming from mergers of massive compact object pairs? ("Massive" in this context means comprised of neutron stars and/or black holes.)
3. What is the typical number of massive compact object pairs in a single star cluster, and how does this vary between clusters?
4. What fraction of the stellar population as a function of number of stars and, separately, of stellar mass explodes in a supernova and/or has a binary merger of some sort? In other words, what fraction of the stars do something "interesting"?
5. As you go to larger stellar populations (say, to star clusters of total mass 10^4 or 10^5 $M_\odot$), does the cluster-to-cluster variation increase, decrease, or stay the same? Why do you think that is?

Part 4: 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 (in the directory `part_4_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?