Project Title: Optimizing Tardis via ASV
Name: Youssef Eweis
E-mail address: joey070@gmail.com
Github username: Youssef15015
Gitter: Youssef15015
Phone: 973-932-2479
Skype: Shikabala4

Brief background info

I grew up in Egypt and moved to the United States for my senior year of highschool. This is my last semester at Rutgers University. I am majoring in astrophysics, mechanical engineering, and french - literary studies. I generally enjoy learning, and the logic applied in coding comforts me. I am planning on starting a PhD program in physics at Iowa State in the fall, studying high energy astrophysics.
I have been working as part of a supernova research group at Rutgers since last may. My main role is extracting spectroscopic data, and working on this pipeline: "https://github.com/Youssef15015/rusalt/blob/master/rusaltD.py". I first learned python 5 years ago, up until object oriented programming. I only started applying professionally since last may. I also analyze cosmological simulations using pynbody.

Experience: Python, most used python modules: pynbody, astropy, pyraf, pyfits, and numpy
Experience needed: C, Cython, and understanding Tardis as a whole

Important note: I plan to take advantage of working remotely from Egypt. I plan on taking about a week of time off in Egypt. This time period is highly variable and will be curtailed to this program. I believe that Egypt is in good relations with the United States, adhering to the requirements of the program.

Project Statement

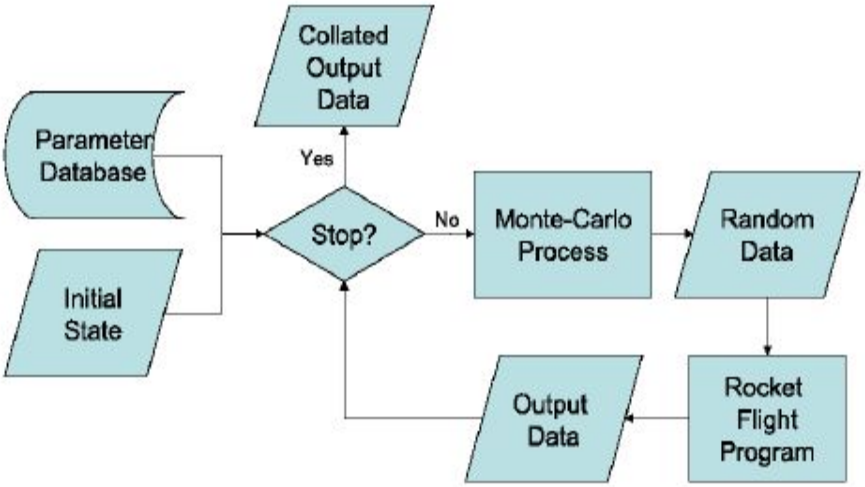
As stated in the list of project ideas, the goal of this project is to benchmark tardis to optimize its performance. ASV is a useful application to test the performance of tardis after each change, such as adding analytics of microphysics. Relative benchmarks of performance, such as time processing, will be measured using airspeed velocity. After each addition to the tardis project, we can then measure the relative performance compared to previous benchmarks.

What to build on:

Upon generating a virtual spectrum, tardis reports the time taken to achieve each step. This can be seen in: https://github.com/tardis-sn/tardis/blob/master/tardis/simulation/base.py , the goal is to use ASV to generate a more accurate and comprehensible report.

Monte Carlo wrappers are an effective method to test performance. Their implementation process has already been started in the tardis github, but commented out. Found here: "https://github.com/tardis-sn/tardis/blob/master/tardis/tests/test_montecarlo.py"

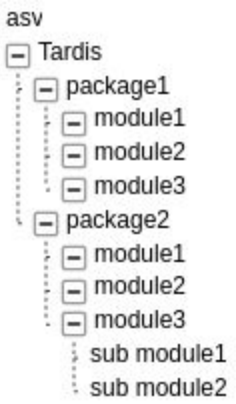
This is a diagram by Simon Box, illustrating the implementation of a montecarlo wrapper. The idea is testing the time it takes to match a desired output through a Monte-Carlo Process. The Monte-Carlo process is inherent to the application of tardis, generating spectrum.



(Box, researchgate)

Requirements

Since the goal is to optimize tardis, it is necessary to understand tardis in its entirety, and understand how each small portion of code affects the whole project. In other words, it is necessary to establish core dependencies, and then the modules and packages that are based off of them. Creating an illustration of modules and packages is generally a useful practice, if not done already.



Implementation

To create an asv environment: the two main necessities is an asv.conf.json file for the parameters of the repository being tested and a benchmark.py that executes the required benchmarking.

Although there are four types of benchmarks, my focus will be on benchmarking time. The outline for asv, via object oriented programming, requires a prefix for each object in a class. The time prefix is time.

A tutorial of writing a benchmark is illustrated here:
https://asv.readthedocs.io/en/stable/writing_benchmarks.html

Current pull request: https://github.com/tardis-sn/tardis/pull/913/
My pull request outlines the directory and outline of the asv environment, without the actual temporary environment as that contains too much data to be pushed to git.

Implementing asv in tardis. I made a time benchmark of 1 dictionary in tardis. Tardis.util.base.atomic_number2element_symbol converts numbers in the dictionary to elements.

```
class Suite:
        def time_number2element(self):
            for i in np.arange(1,100):
                base.atomic_number2element_symbol(i)
```

This is the generated report by asv of the time taken by performing this task:


```
"results": {
        "benchmarks.Suite.time_number2element": {
            "result": [
                4.974437044534412e-05
            ],
            "stats": [
                {
                    "ci_99": [
                        4.3694777327935254e-05,
                        5.8334761133603255e-05
                    ],
                    "max": 5.8334761133603255e-05,
                    "mean": 4.988159716599191e-05,
                    "min": 4.3694777327935254e-05,
                    "number": 247,
                    "q_25": 4.425990182186234e-05,
                    "q_75": 5.4243711538461556e-05,
                    "repeat": 10,
                    "std": 5.464971746521136e-06
                }
            ]
        }
    }
```


My plan, after mapping out the most important tardis packages and modules, is to implement this technique, especially to tardis/simulation/base.py, when it generates a spectrum.

There are three known python modules that have implemented asv and shared their coding.

Resources

Asv documentation:
https://asv.readthedocs.io/en/stable/

Examples of benchmarks using asv :
https://www.astropy.org/astropy-benchmarks/
https://pv.github.io/numpy-bench/
https://pv.github.io/scipy-bench/

Schedule

- (**Milestone 1)**From now until the end of the community bonding period, my schedule will be highly restrained by the semester, but I will be reviewing, C, cython, and tardis as a whole. I will be accessible via gitter within 24 hours, and have access to one of the mentor's physical office (to ask questions about tardis in general).
- From May 27th to June 1st. I will be online each day for at least 5 hours, either in the mornings or afternoons for EST.
- **(Milestone 2**) Apply montecarlo wrappers to easily test the implementation time for each module.
- From June 1st to the end of July, I will be working remotely from Egypt, and will have no other obligations (40+ hours), so I will fit my schedule according to my mentor's.
- **June**: (**Milestone 3)** Understand the best way to test the optimization of each required module/packages, have first benchmark module draft ready.
- **July:** (**Milestone 4)** Have final benchmark module (benchmark.py) ready + test it with adding and removing subtle variations in the code of the tardis directory.
- From August 1st to the end of the project, I will move to the central time zone, but will have no other obligations (40 + hours).
- August 15: I will officially start graduate school.
- Generally the deliverables will be the benchmark.py file and the generated reports.

Why us?
Given the requirements of: "Maintain a dev log", "List your milestones, and Be communicative", I want to learn to work in a professional environment with others. I am sure that the version of myself after this program will be far more knowledgeable and proficient in coding that my

current self. That alone is enough reason for me to apply to this program.
I specifically chose this project, beauce the concept of optimization is inherently important for any application. After this project is over, I would like to keep working in an open collaboration, and would specifically want to remain in the SN community, especially with Tardis.

Why you?
Despite my lack of education in formal and theoretical computer science, I find that I good and fast at learning new things. I also feel obligated to return whatever time you invest in me. I also believe that I have good communication skills.

Anything else?
Being a part of the xda-developers, during high school, I continuously extracted information for my own android device. I felt really proud the first time I contributed useful information for many people all around the world to profit from. I think more than anything, that experience makes me want to join an open-source community to continuously exchange useful information. I think my time during the semester was fairly limited, and I will be much more active during the summer.

References

Box, Simon "Code block diagram for the Monte-Carlo wrapper" https://www.researchgate.net/figure/Code-block-diagram-for-the-Monte-Carlo-wrapper_fig10_245307313
"Airspeed velocity" https://asv.readthedocs.io/en/stable/
"Tardis" https://github.com/tardis-sn/tardis
https://creately.com/app/#