AE 6102 - Parallel Scientific Computing and Visualization

Orbit Propagation

Prof. Prabhu Ramachandran

Spring 2022-23

Konda Karthikeya - 190050060
Akash Reddy G - 190050038
Kartikey Gupta - 190050044

May 1, 2023

Contents

1 Abstract

Calculating trajectories of a space object given current conditions is non trivial task involving integration , which are mostly not closed form. However can be computed numerically . In this project we try to build and visualise a simple orbit propagation simulator based on the following update equations

$\vec{v}_{t+dt} = \vec{v}_t + \vec{a} * dt$

$\vec{s}_{t+dt} = \vec{s}_t + \vec{v}_t * dt$

We also contrast the behaviour of a satellite under different initial velocity conditions, creating visualizations of the same.

2 Tools

The following tools have been used in this project

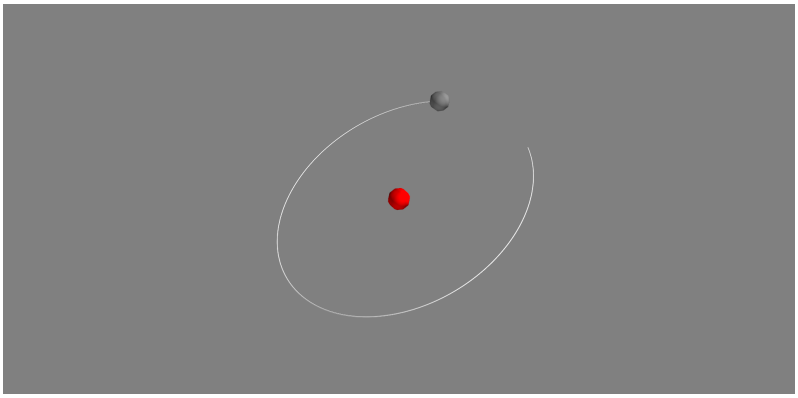
- Python3 - Our choice of programming language
- Numba, numpy - To obtain performance enhancements
- Automan - To automate generation of data for visualization
- Mayavi - to visualize orbit propogation

3 Outline

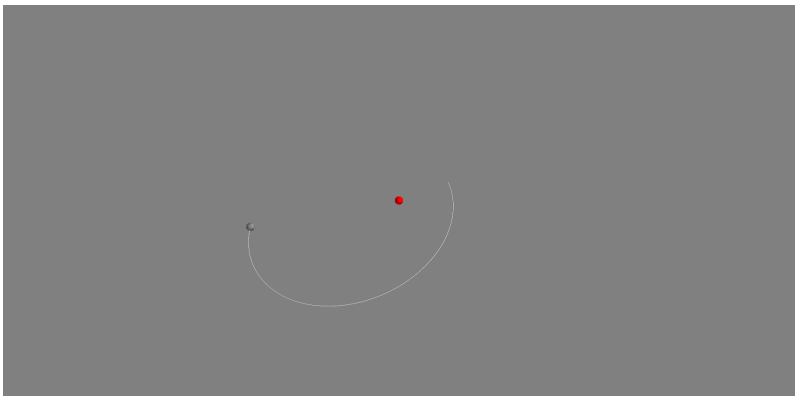
- Implemented the physics of gravity
  - Did not use any external libraries - such as scipy - and computed the various physical quantities via differential time update
  - developed custom classes so as to ease development
- Visualised our implementation by Mayavi
  - Encountered difficulties while trying to animate using the mlab.animate decorator, explored other possibility using time.sleep method.
  - Still encountered visualization difficulties where animation stops halfway without any debug errors. Used a trick of using mlab.savefig to refresh the visualization
  - This gave a neat visualization but ended up taking toll on performance
  - overall, absence of debug messages from Mayavi gave us a hard time
- Optimised the computation using numba, numpy
  - Encountered difficulty while trying to use numba optimization on user defined classes
  - Separated the computation of orbit path from user defined classes and successfully used numba to gain performance benefits!
  - Used vectorized equations to compute paths of multiple satellites at once and visualized them together
- Automating the tests
  - Used the Automan library to automate generation of data for various simulations of satellite propagation
- Some possible extensions to our work-
  - Study simulation of a binary-star system and a satellite's behaviour around this system under different initial conditions.
  - Leverage parallel computing and multiple cores for generating data of system with multiple satellites.

4 Results

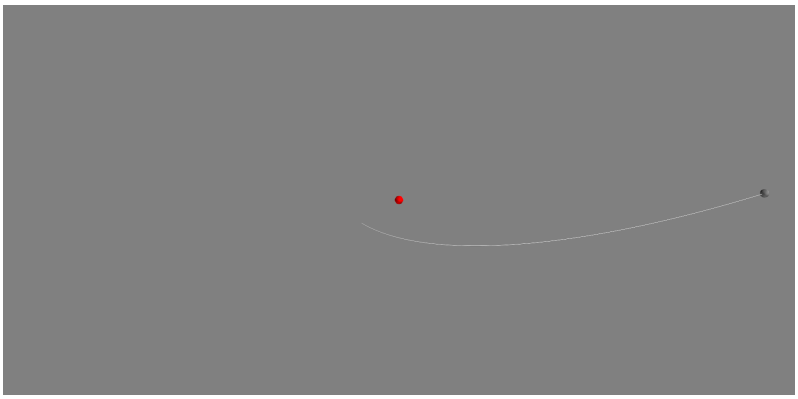
Mayavi visual :



(a) circular orbit



(b) Elliptical Orbit



(c) parabolic path

Figure 1: Three different paths of the satellite, based on speed

Also the animations can be viewed in our github repository.
Single satellite animations - here
Multiple Satellite animations - here

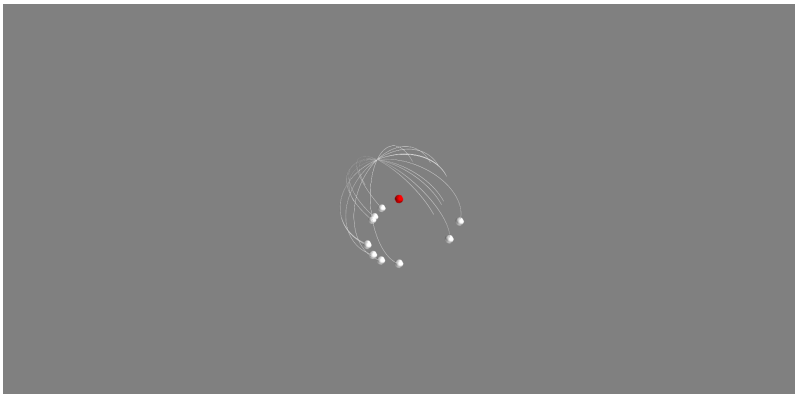


Figure 2: Multiple objects processed at once

The following graph shows the performance enhancement acheived using numba optimised code. The setup is as follows -

- We are simuulating for a fixed time frame - 1 year
- 'dt' refers to the duration of each differential time step we consider for recalculating the position, velocity and force parameters.
- As dt decereases, the computation increases as total number of frames increase. Here the numba leverage is clearly depicted.

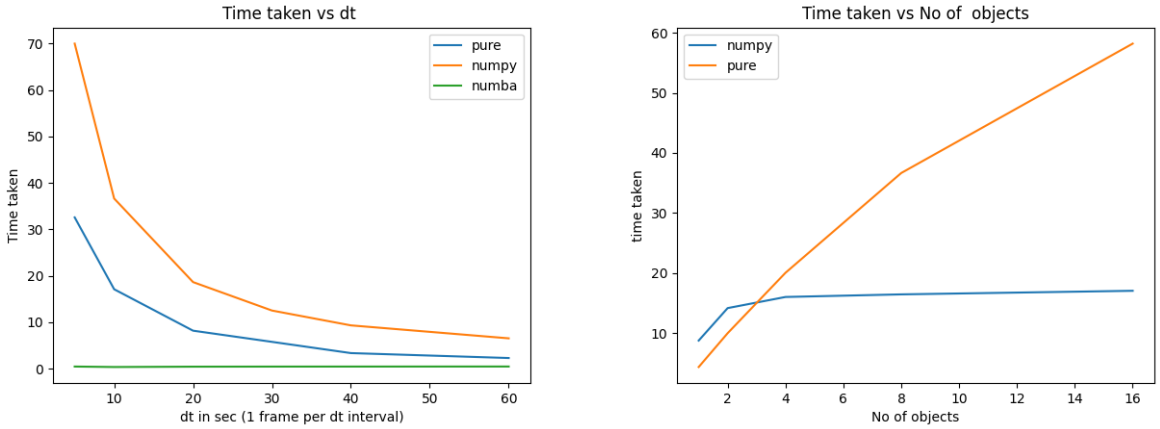


Figure 3: Performance optimisations

5 Timeline

Konda Karthikeya - Physics, Automation and Numba benefits
Akash Reddy G - Visualization through Mayavi
Kartikeya Gupta - Classes development, vectorization & Report

Update No	work
1	Exploration and other works
2	Development of physics and Basic visualization
3	Bug fixes and Final visualization
4	Optimisation, automation, Enhancing visualization

Table 1: Timeline of the project

6 Code

Link to the github repo : here

Majority of the code is organized as :

1. **driver.py, driver_2.py, driver_3.py** - run with command line arguments to generate *.npy files used for visualization

2. **visual*.ipynb** - jupyter notebooks used to generate visualizations using Mayavi and store snapsohts in a directory, which are used to prepare a animation clip

3. **Object.py** - contains the physics implementations and optimisations as well
4. **automate.py** - the automan file used to generate all the data used in this report and the repo
5. **image_to_mp4.ipynb** - notebook with code to compile all the snapshots together to generate a mp4 clip
6. **data/** - some example dataset used to generate above clips
7. **videos/** - Example clips

7 Resources

- **Mayavi Docs** - can be found at http://docs.enthought.com/mayavi/mayavi/
- **Converting images to video clip** - Stack Overflow Answer
- **Not all images were converted to clip** - I ran into a bug where all the images were not converted into the clip, turns out the images being generated were not all of the same size! This (click here) helped me identify the problem, after which I was careful not to resize the Mayavi window during image generation.
- **Automan Docs** - can be found at https://automan.rtfd.io/