Simulation of the Two-Body Gravitational Problem with Relativistic Perturbation

Benjamin Payne

1 Software and Installation Instructions

This project was created using the pre-installed PHYS 129L software stack on the Raspberry Pi. The simulation was written entirely in Python and does not require any additional software beyond what is already provided.

Required Software:

- Python 3.9 or later
- NumPy
- Matplotlib
- JSON
- sys

2 External Hardware

No external hardware is required to run this project. All data, simulations, and plots are generated entirely using the Raspberry Pi.

3 Project Description

The goal of this project is to simulate and analyze the classical two-body gravitational problem using numerical integration techniques, and to validate the numerical results through comparison to known analytical solutions. The program allows the user to specify the physical parameters of the two bodies, including their masses, radii, and relative initial positions and velocities.

The system evolves in time using the Velocity-Verlet integration method, which is symplectic, making it well-suited for long-term orbital simulations. The orbital motion is computed in the center-of-mass frame using user-inputted relative coordinates. From this relative motion, the individual trajectories of both bodies are reconstructed.

At each time step, the program computes key orbital quantities such as the specific energy, specific angular momentum, and orbital eccentricity. These quantities are monitored throughout the simulation to verify numerical stability and physical accuracy based on conservation laws.

The simulation automatically classifies the orbit as bound (elliptical), marginally bound (parabolic), or unbound (hyperbolic) based on the calculated eccentricity. Collision detection is implemented by checking whether the separation between the two bodies falls below the sum of their radii. If a collision occurs, the simulation reports the event and terminates.

In addition to the purely Newtonian simulation, the program includes an optional general relativistic perturbation term proportional to $1/r^3$, which is added to the gravitational acceleration. This perturbation breaks the closed-orbit symmetry of Newtonian gravity and produces a measurable periapsis precession. A comparison script is included that directly contrasts Newtonian and relativistic orbits.

Finally, in the purely Newtonian case, the numerical solution is validated against the analytic Kepler solution by directly comparing trajectories and measuring the accumulated position error over time.

4 Results and Program Output

4.1 Running the Program

The main simulation is executed from the terminal using a JSON input file that specifies all physical and numerical parameters:

```
python main.py input_file.json
```

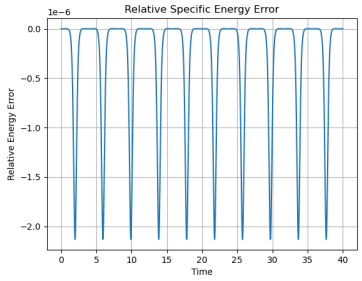
The input file includes:

- Masses and radii of both bodies
- Initial relative position and velocity vectors
- Gravitational model (Newtonian or relativistic)
- Time step and total simulation time

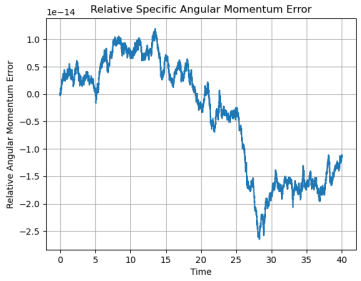
Example JSON files are provided for bound, marginally bound, and unbound orbital configurations.

4.2 Newtonian Orbit Simulation

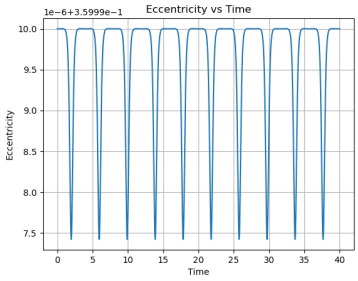
For purely Newtonian gravity, the simulation produces closed elliptical orbits for bound systems, consistent with Kepler's laws. The total specific energy, magnitude of angular momentum, and orbital eccentricity remain constant throughout the simulation, up to small numerical oscillations arising from the discrete time stepping. The plots below, and all of the following, use the values in the elliptical orbit example file (without relativistic perturbation).



(a) Relative specific energy error versus time.



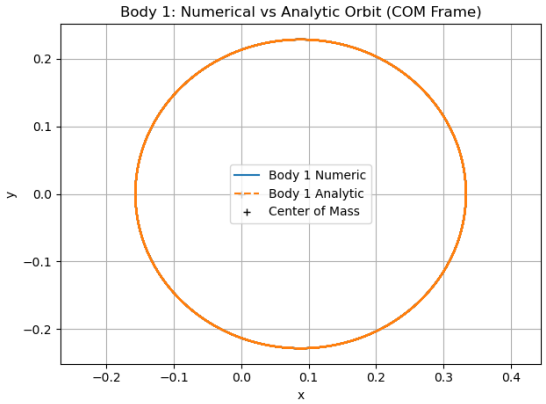
(b) Relative angular momentum error versus time.



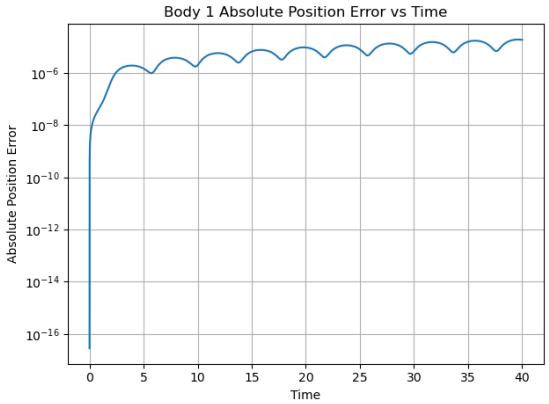
(c) Orbital eccentricity versus time.

4.3 Numerical vs. Analytical Comparison

For purely Newtonian gravity, the numerical solution is compared directly to the analytic Kepler solution for an elliptical orbit. The two trajectories are nearly identical and lie on top of each other visually. The absolute position error between the trajectories remains extremely small but grows slowly over time due to finite time-step effects.



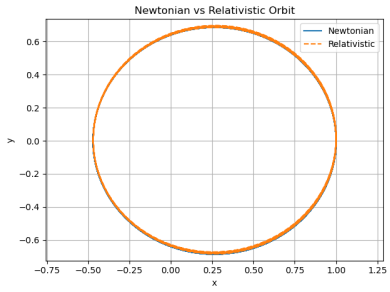
(a) Comparison of numerical and analytical trajectories.



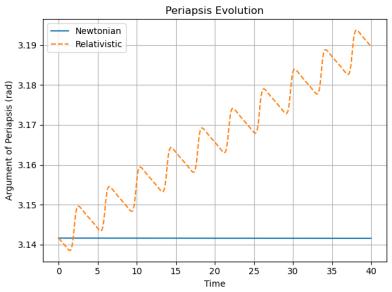
(b) Absolute position error between numerical and analytical solutions.

4.4 Relativistic Perturbation and Periapsis Precession

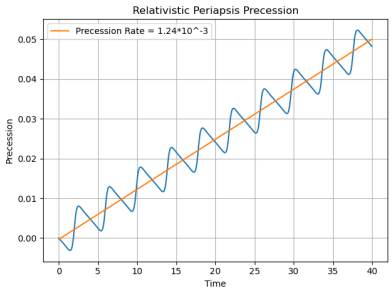
When the relativistic perturbation is enabled, the orbit is no longer closed. The periapsis advances steadily over time, producing a clear precession effect. By tracking the eccentricity vector, the total precession and precession rate are measured and compared to the purely Newtonian case.



(a) Comparison of Newtonian and relativistic orbits ($\alpha = 0.001$).



(b) Argument of periapsis versus time ($\alpha = 0.001$).



(c) Periapsis shift versus time with linear fit ($\alpha = 0.001$).

4.5 Animation Output

The program also generates a real-time animation of the two-body motion, displaying the positions of both bodies along with their motion trails. Unfortunately, due to the computing power of the Raspberry Pi, I was unable to save the animation of the orbits as a GIF while running the simulation. However, you can see for yourself by running the command from **4.1**. Collision events are visually detected when the bodies overlap. In the event of a collision, the collision time and positions are printed to the terminal and the simulation is terminated.

5 Summary

This project demonstrates the accuracy and stability of numerical integration methods for the two-body gravitational problem by validating numerical results against analytical Kepler solutions. The inclusion of a relativistic perturbation allows the simulation to capture physically observable deviations from Newtonian motion, such as periapsis precession, highlighting the power of numerical methods in exploring perturbations to classical mechanics.