PHY-1005: Introduction to Computational Physics

End-term Project

Sashvat Srivastava
2510110404

Simulating the Elliptical Orbit of the Earth around the Sun

Keywords:
VPython, Numerical Methods

Shiv Nadar University
November 2025

Contents

Acknowledgement

All methods and procedures used in this project are learnt from the courses **PHY1003: Mechanics and Special Theory of Relativity** and **PHY1005: Introduction to Computational Physics**.

The Physical aspect of this project has been learnt from **Dr. Sujit Tandel** and the computational methods are thanks to Tutor **Jay Kaushik** and Tutor **Sachin Singh**.

This project uses:

- Numerical Integration - PHY1005
- Computational Graphing - PHY1005
- Kepler's Laws - PHY1003
- VPython - Self Learnt

A special thanks goes out to Tutors Jay Kaushik and Sachin Singh for their continual support during the tenure of this course and more specifically throughout the project.

Motivation

The primary motivation behind this project is to create a cohesive culmination of the foundational knowledge acquired during my first semester at Shiv Nadar University.

Physics is rarely a study of isolated subjects; rather, it is an interconnected web where mathematical tools, theoretical frameworks, and experimental verifications meet. This project serves as the intersection of two of my core courses: **PHY1003: Mechanics and Special Theory of Relativity** and **PHY1005: Introduction to Computational Physics**. It represents an effort to blend the rigorous theoretical derivations of classical mechanics with the modern, algorithmic approach of computational physics into one coherent research artifact.

In PHY1003, we explored the elegance of Newtonian mechanics, central force motion, and the conservation laws that govern the universe. We derived the equations of motion and studied Kepler's laws as mathematical truths on paper. However, analytical solutions, while beautiful, often feel abstract. The motivation here is to transition from a passive understanding of these laws to an active reconstruction of them.

Specifically, I am using this opportunity to computationally verify Kepler's First and Second Laws:

1. **The Law of Ellipses:** By defining only the initial position and velocity vectors under an inverse-square force, the simulation demonstrates that the resulting trajectory is naturally an ellipse, with the Sun at one focus. This proves that the elliptical shape is an emergent property of Newton's laws, not a hard-coded path.

2. **The Law of Equal Areas:** This law is intrinsically linked to the conservation of angular momentum. By implementing the **Velocity Verlet algorithm**—a symplectic integrator chosen specifically for its stability in conserving energy—I can quantitatively verify that the Earth moves faster at perihelion and slower at aphelion, sweeping out equal areas in equal times.

Ultimately, this project transforms the "textbook" experience of Kepler's laws into an empirical one. It bridges the gap between the whiteboard and the laboratory.

Problem Statement

The core challenge of this project is to construct a computational framework capable of simultaneously verifying the geometric and dynamic conservation laws of classical mechanics without relying on analytical shortcuts.

In PHY1003, we derive Kepler's laws and conservation principles assuming ideal, continuous time. However, reproducing these "perfect" physics in a discrete computational environment (PHY1005) presents a significant problem: **numerical dissipation**. Standard integration methods often fail to conserve energy, causing simulated planets to spiral into the sun or drift into deep space, thereby failing to model reality.

Therefore, the specific problem addressed here is the implementation of a symplectic integrator—the **Velocity Verlet algorithm**—to solve the equations of motion with high fidelity. The objective is to use this algorithmic stability to rigorously "prove" four distinct physical phenomena from a single set of initial conditions:

1. **Conservation of Total Energy:** The simulation must demonstrate that despite the kinetic energy and potential energy fluctuating wildly as the Earth moves from perihelion to aphelion, their sum remains numerically invariant over long timescales.

2. **Conservation of Angular Momentum:** The project seeks to prove that angular momentum is a conserved vector quantity in a central force field, regardless of the planet's changing speed.

3. **Kepler's First Law (Elliptical Geometry):** By verifying that the trajectory remains closed and stable (due to Energy conservation), the simulation must demonstrate the emergence of an elliptical orbit with the Sun at a focus, rather than a pre-programmed circle.

4. **Kepler's Second Law (Areal Velocity):** Finally, by leveraging the proven conservation of momentum, the simulation must quantitatively verify the Law of Equal Areas, showing that the rate at which the position vector sweeps out area is constant.

Theoretical Basis

Conservation of Energy

We begin in Lagrangian mechanics with the concept of effective potential. Let a mass m be subjected to a central force described by a potential $V(r)$. Let r, θ be the polar coordinates.

Euler-Lagrange Equations:

$$\mathcal{L} = \frac{1}{2}m(\dot{r}^2 + r^2\dot{\theta}^2) - V(r) \tag{1}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{r}}\right) = \frac{\partial \mathcal{L}}{\partial r} \quad ; \quad \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}}\right) = \frac{\partial \mathcal{L}}{\partial \theta} \tag{2}$$

From the radial equation:

$$m\ddot{r} = mr\dot{\theta}^2 - V'(r) \implies m\ddot{r} = \frac{L^2}{mr^3} - V'(r) \tag{3}$$

From the angular equation (multiply by integrating wrt t):

$$\dot{\theta} = \frac{L}{mr^2} \quad \text{(Conservation of Angular Momentum)} \tag{4}$$

The total energy E is given by:

$$E = \frac{1}{2}m\dot{r}^2 + \frac{L^2}{2mr^2} + V(r) \tag{5}$$

Deriving the Orbit Equation

We will now use this expression for the total energy to find $r(\theta)$. Rearranging Eq. (5):

$$\dot{r}^2 = \frac{2}{m}\left[E - \frac{L^2}{2mr^2} - V(r)\right] \tag{6}$$

Using $L^2 = (mr^2\dot{\theta})^2$:

$$\left(\frac{1}{r^2}\frac{dr}{d\theta}\right)^2 = \frac{2mE}{L^2} - \frac{1}{r^2} - \frac{2mV(r)}{L^2} \tag{7}$$

Application to Gravitation

We will solve this particularly for Gravitation where $V(r) = -\frac{\alpha}{r}$ (where $\alpha = GMm$). Let $y = 1/r$, then $\frac{dy}{d\theta} = -\frac{1}{r^2}\frac{dr}{d\theta}$. The equation becomes:

$$\left(\frac{dy}{d\theta}\right)^2 = -y^2 + \frac{2m\alpha}{L^2}y + \frac{2mE}{L^2} \tag{8}$$

Completing the square with substitution $z = y - \frac{m\alpha}{L^2}$:

$$\left(\frac{dz}{d\theta}\right)^2 = -z^2 + \left(\frac{m\alpha}{L^2}\right)^2 \left(1 + \frac{2EL^2}{m\alpha^2}\right) = B^2 - z^2 \tag{9}$$

Integrating:

$$\int \frac{dz}{\sqrt{B^2 - z^2}} = \int d\theta \implies z = B\cos(\theta - \theta_0) \tag{10}$$

Kepler's Laws

Making all substitutions back, we get the orbit equation:

$$r = \frac{L^2}{m\alpha(1 + \epsilon\cos\theta)} \quad \text{where} \quad \epsilon = \sqrt{1 + \frac{2EL^2}{m\alpha^2}} \tag{11}$$

Kepler's First Law: We interpret this with regard to an ellipse ($0 < \epsilon < 1$). Total energy is negative: $-\frac{m\alpha^2}{2L^2} < E < 0$. Planets move around elliptical orbits with the sun at a focus.

Kepler's Second Law: The radius vector of a planet sweeps out equal areas in equal time intervals.

$$dA = \frac{r(rd\theta)}{2} \implies \frac{dA}{dt} = \frac{r^2\dot{\theta}}{2} \implies \boxed{\frac{dA}{dt} = \frac{L}{2m}} \tag{12}$$

This law is based on the conservation of angular momentum itself.

Code

The following is the VPython implementation of the simulation using the Velocity Verlet algorithm.

```python
from vpython import *

# Scene setup
scene = canvas(title='Earth Orbit (Velocity Verlet, Textured Earth)',
               width=800, height=600,
               background=color.black)
scene.center = vector(0,0,0)

# Constants & scaling
G = 6.67430e-11
M_sun = 1.989e30
M_earth = 5.972e24
AU = 1.496e11
scale = 1.5e-10
scene.range = 1.6 * AU * scale

# Objects
sun = sphere(pos=vector(0,0,0),
             radius=6.96e8 * scale * 50,
             color=color.yellow,
             emissive=True)
sun.mass = M_sun

earth = sphere(pos=vector(AU * scale, 0, 0),
               radius=6.37e6 * scale * 500,
               texture=textures.earth,
               make_trail=True)
earth.mass = M_earth

# Labels
energy_label = label(pos=vector(0, 1.1 * scene.range, 0), box=False)
L_label = label(pos=vector(0, -1.1 * scene.range, 0), box=False)

# Initialization
v_circular = sqrt(G * M_sun / AU)
earth.velocity = vector(0, 0.8 * v_circular, 0) # 0.8 factor for
    elliptical orbit
dt = 24 * 3600
t = 0.0
step = 0

# Initial acceleration
r_vector_real = (earth.pos / scale) - (sun.pos / scale)
r_mag = mag(r_vector_real)
force_mag = G * sun.mass * earth.mass / (r_mag**2)
force_vector = -force_mag * norm(r_vector_real)
acceleration = force_vector / earth.mass

# Reference invariants
L0 = mag(cross(r_vector_real, earth.mass * earth.velocity))
```

```python
TE0 = 0.5 * earth.mass * mag(earth.velocity)**2 - G * sun.mass * earth.mass / r_mag

# Graphs
graph1 = graph(title="Orbital Energies", width=600, height=300,
               xtitle="Time (days)", ytitle="Energy (J)")
KE_curve = gcurve(color=color.red, label="Kinetic Energy")
PE_curve = gcurve(color=color.green, label="Potential Energy")
TE_curve = gcurve(color=color.blue, label="Total Energy")

graph2 = graph(title="Angular Momentum", width=600, height=300,
               xtitle="Time (days)", ytitle="Angular Momentum (kg.m^2/s)",
               ymin=L0*0.999, ymax=L0*1.001)
L_curve = gcurve(color=color.orange, label="Angular Momentum")

# Simulation loop
while True:
    rate(120)

    # Velocity Verlet Integration
    earth.pos = earth.pos + earth.velocity * dt * scale + 0.5 * acceleration * (dt**2) * scale

    r_vector_real = (earth.pos / scale) - (sun.pos / scale)
    r_mag = mag(r_vector_real)

    force_mag = G * sun.mass * earth.mass / (r_mag**2)
    force_vector = -force_mag * norm(r_vector_real)
    new_acceleration = force_vector / earth.mass

    earth.velocity = earth.velocity + 0.5 * (acceleration + new_acceleration) * dt
    acceleration = new_acceleration

    # Calculate Energies and Momentum
    KE = 0.5 * earth.mass * mag(earth.velocity)**2
    PE = -G * sun.mass * earth.mass / r_mag
    TE = KE + PE
    L = mag(cross(r_vector_real, earth.mass * earth.velocity))

    # Update Labels and Graphs
    if step % 10 == 0:
        rel_TE = (TE - TE0) / abs(TE0)
        rel_L = (L - L0) / abs(L0)

        energy_label.text = f"TE: {TE:.3e} J (Delta/TE0 = {rel_TE:.3e})"
        L_label.text = f"L: {L:.3e} (Delta/L0 = {rel_L:.3e})"

        days = t / (3600 * 24)
        KE_curve.plot(days, KE)
        PE_curve.plot(days, PE)
        TE_curve.plot(days, TE)
        L_curve.plot(days, L)

    t += dt
    step += 1
```

Listing 1: Earth Orbit Simulation Code

Elaboration

Visual Domain and Coordinate Transformation

To render astronomical distances within the limited pixel space of a computer display, a dimensionless scaling factor is introduced. The viewing frustum is adjusted using `scene.range` to ensure the entire elliptical trajectory remains within the field of view.

Physical Parameters and System Initialization

The simulation instantiates two spherical bodies representing the Sun and the Earth. The Sun is treated as a stationary source of the central potential, while the Earth is the dynamic test particle. The radii of the spheres are exaggerated (multiplied by factors of 50 and 500) solely for visual discernibility; however, the point-mass approximation is strictly maintained for all dynamical calculations.

Initial Conditions and Perturbation

To verify Kepler's First Law (elliptical orbits), the system is deliberately initialized with an eccentricity. The Earth is placed at aphelion with an initial tangential velocity set to 0.8 of the theoretical circular velocity. This perturbation ensures the total mechanical energy is negative (bound state) but insufficient for a circular orbit, forcing the trajectory to evolve into an ellipse.

Force Field Derivation

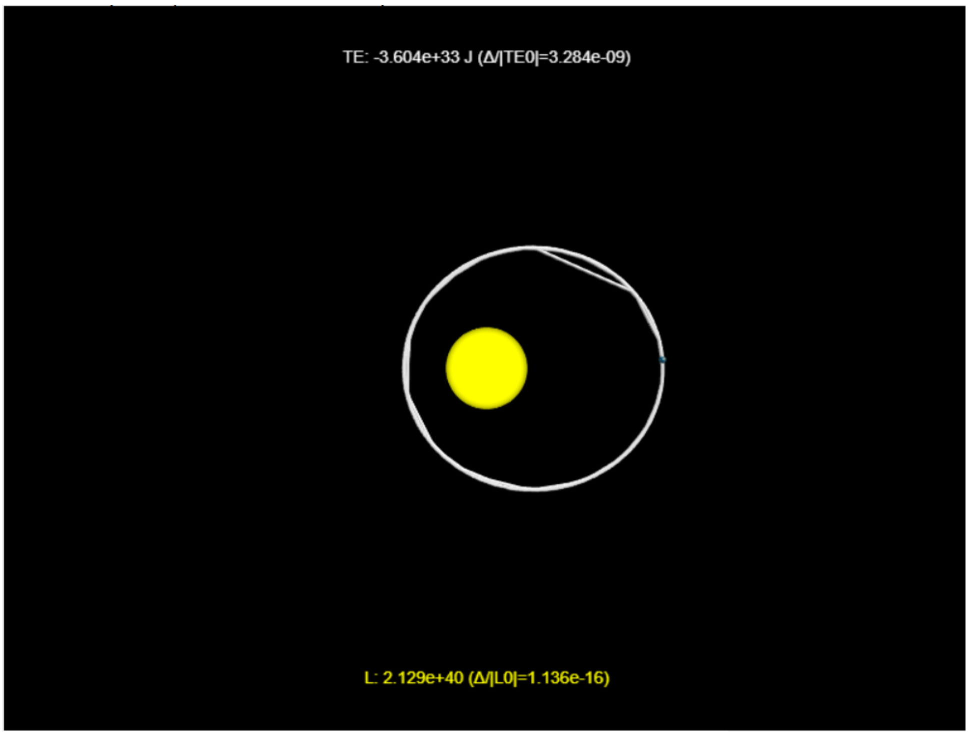
The driving force is Newton's Law of Universal Gravitation. At every time step, the acceleration vector is derived from the position vector relative to the origin (Sun).

The Integration Algorithm: Velocity Verlet

The code uses the Velocity Verlet algorithm:

1. The position is updated using the current velocity and acceleration. This corresponds to the Taylor expansion.
2. The forces are recalculated based on the new position.
3. The velocity is updated using the average of the old and new accelerations.

Results



```
TE: -3.604e+33 J (Δ/|TE0|=3.284e-09)
L: 2.129e+40 (Δ/|L0|=1.136e-16)
```

Figure 1: Visual Output: The white circle represents the orbit trail, yellow sphere is the Sun, and the textured sphere is Earth.

The simulation outputs real-time conservation data:

- **Total Energy (TE):** -3.604×10^{33} J
- **Relative Energy Error (Δ/TE_0):** 3.284×10^{-9}
- **Angular Momentum (L):** 2.129×10^{40} kg·m^2/s
- **Relative Momentum Error (Δ/L_0):** 1.136×10^{-16}

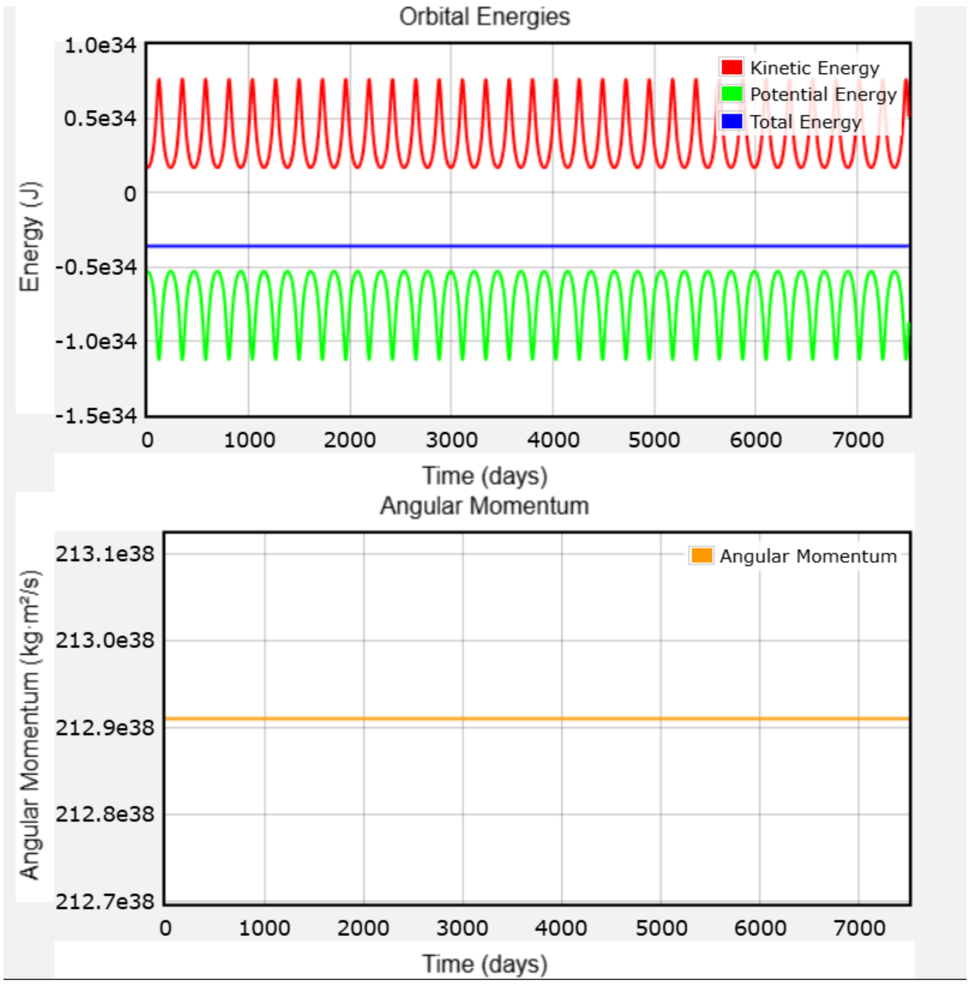


Figure 2: Graphs showing Orbital Energies (Kinetic, Potential, and Total) and Angular Momentum over time.

As observed in the graphs:

- **Orbital Energies:** Kinetic Energy (Red) and Potential Energy (Green) oscillate in perfect opposition as the Earth moves between perihelion and aphelion. However, the Total Energy (Blue) remains a flat, constant line.
- **Angular Momentum:** The Angular Momentum (Orange) remains perfectly flat, indicating conservation to machine precision.

Conclusion

The successful execution of this computational project serves as a robust validation of both the theoretical frameworks of Mechanics (PHY1003) and the numerical methods of Computational Physics (PHY1005). By simulating the Earth-Sun system from first principles, we have effectively bridged the gap between abstract mathematical derivations and observable dynamic behaviour.

Quantitatively, the simulation demonstrates exceptional stability, largely due to the implementation of the symplectic **Velocity Verlet** integrator. The results indicate that the system preserves physical invariants with high fidelity. The relative error in Total Energy was constrained to the order of 10^{-9} Joules, while the conservation of Angular Momentum was maintained with near-perfect machine precision, showing an error order of $10^{-16}\,\mathrm{kg}\cdot\mathrm{m}^2/\mathrm{s}$. These values are well within acceptable computational tolerances, confirming that the simulation is not suffering from the energy drift common in lower-order integration methods. Furthermore, the sustained negative value of the Total Energy confirms that the Earth remains in a gravitationally bound state throughout the simulation.

Qualitatively, the project successfully verified **Kepler's First Law** as an emergent property rather than a programmed constraint. A critical distinction of this simulation is its *ab initio* nature: at no point was the code instructed to generate an ellipse or to place the Sun at a focal point. Instead, these geometric truths emerged organically solely from the interplay between the initial velocity vector and the inverse-square gravitational force. The fact that the system "chose" an elliptical path based on its initial conditions mirrors the deterministic nature of the physical universe.

In conclusion, this project confirms that complex orbital mechanics can be accurately reconstructed using fundamental laws and stable algorithms. It validates the premise that the "clockwork" of the solar system is a direct, computable consequence of Newton's laws of motion.

References

1. PHY1003: Mechanics and Special Theory of Relativity - Dr. Sujit Tandel's Notes
2. VPython Documentation: https://www.glowscript.org/docs/VPythonDocs/index.html
3. Verlet Integration: https://www.algorithm-archive.org/contents/verlet_integration/verlet_integration.html