Interactive N-Body Solar System Simulator: A Parameter-Editable Gravitational Model

Welcome to My N-Body Simulation Playground

Congratulations! You’ve stumbled upon my orbital-dynamics code repository.
Whether you arrived here intentionally, accidentally, or because your PHYS 5300 homework hijacked your sanity—welcome.

If your only goal is to **run the code without thinking too hard** feel free to skip everything and go straight to the **How to Use** section at the bottom.
It contains all the essential instructions, warnings.

On the other hand, if you're genuinely curious about the details—or if you're a fellow **PHYS 5300 student mining GitHub for inspiration**—please enjoy browsing through the notebook.
You’ll find orbit plots, phase-space diagrams, energy evolution, and a widget that lets you tweak parameters carefully.

Have fun.

Project Description

In this project, I investigate the gravitational dynamics of the inner Solar System using a fully interactive N-body simulator.
The system contains five bodies: the Sun, Mercury, Venus, Earth, and Mars.

Users may freely modify planetary masses and initial velocities, allowing exploration of orbital stability, gravitational perturbations, resonance behavior, and chaotic divergence in multi-body systems.

The objective of this project is to study how changes in fundamental parameters influence orbital trajectories, radial phase-space dynamics, and long-term stability.

Because no closed-form analytical solution exists for general N-body motion ($N \geq 3$), numerical integration of Newtonian gravitation is required, making this a sufficiently complex physics problem for advanced mechanics.

System Description

The system consists of $N = 5$gravitationally interacting point masses.

Each body has a position vector:

$$r_i(t) = \langle x_i(t), y_i(t) \rangle$$

and a velocity vector:

$$v_i(t) = r_i{}'(t)$$

Distances and velocities in the simulation use meters (m) and meters per second ($m \cdot s^{-1}$).
For visualization, all trajectories are converted into astronomical units (1 AU = 1.496×10^{11} m).

The interaction between any two bodies $i$and $j$follows Newton's gravitational potential:

$$U_{ij} = -Gm_i m_j / | \ r_i - r_j \ |$$

The force on body i is computed by summing contributions from all $j \neq i$.

The N-body system is expressed using the Lagrangian:

$$L = T - V$$

Kinetic energy:

$$T = ½ \sum_n m_i \ | \ \dot{r}_i \ |^2$$

Gravitational potential energy:

$$V = -\sum_{(i<j)} Gm_i m_j / | r_i - r_j |$$

Thus:

$$L = ½\sum m_i \mid \dot{r}_i \mid^2 + \sum_{(i<j)} m_i m_j / | r_i - r_j | G$$

Using the Euler–Lagrange equation:

$$d/dt(\partial L/ \partial \dot{r}_i) - \partial L/ \partial r_i = 0$$

we obtain the equations of motion:

$$m_i \ddot{r}_i = G \sum_{(j \neq i)} m_i m_j (r_j - r_i)/| r_j - r_i |^3$$

which is exactly the formula used in the simulator's numerical integrator.

Numerical solution uses the method (RK45), implemented by the SciPy routine.

Symmetries and Assumptions

Translational Symmetry → Linear Momentum Conserved

The Lagrangian depends only on relative coordinates $r_i - r_j$.
Because absolute position never appears, total linear momentum of the system is conserved.

Rotational Symmetry → Angular Momentum Conserved

The gravitational potential is central:

$$U_{ij} = U(| r_i - r_j |)$$

Therefore, the system is invariant under rotations.
This implies conservation of total angular momentum:

$$L_{tot} = \sum r_i \times p_i$$

Time-Translation Symmetry → Total Energy Conserved

The Lagrangian has no explicit dependence on time:

$$\partial L / \partial t = 0$$

Thus the Hamiltonian:

$$H = T + V$$

is conserved.

Visualization

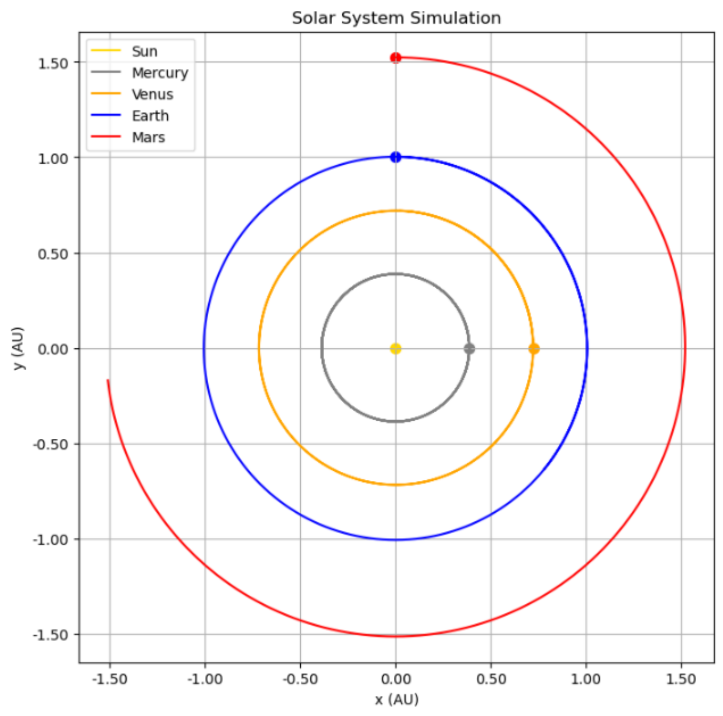


Figure 1 — Orbital Trajectories in Coordinate Space (x–y plane) . Orbital trajectories of the inner Solar System obtained through numerical integration of the N-body equations of motion. Each planet follows a distinct path in the x–y plane under Newtonian gravitation. Distances are expressed in astronomical units (AU). The trajectories reflect stable closed orbits consistent with Keplerian expectations when parameters remain unmodified**.**

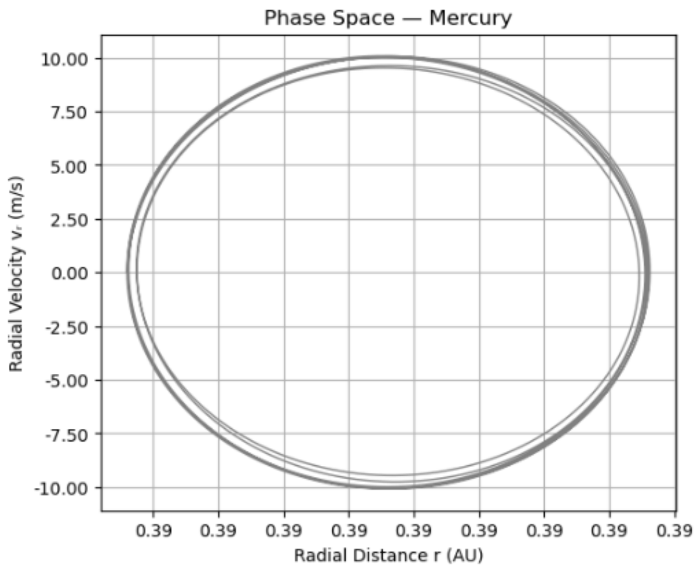


Figure 2 — Phase Space Diagram of Mercury (r vs v_r). *Radial phase-space diagram showing Mercury's radial distance r (in AU) versus radial velocity v_r over the simulation period. Near-closed loops indicate quasi-periodic motion. Deviations from perfect ellipses reflect mutual gravitational perturbations. This plot highlights stability and periodicity in orbital dynamics.*

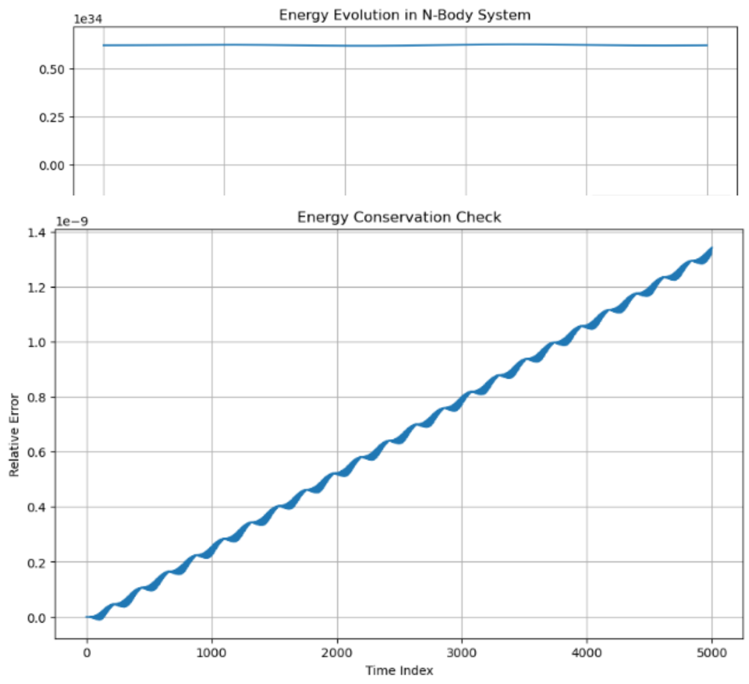


Figure 3 — Energy Evolution Over Time

Time evolution of total system energy E(t) computed from kinetic and gravitational potential contributions.

Because the Lagrangian has no explicit time dependence, the Hamiltonian (total energy) should remain conserved. Small fluctuations arise from numerical integration error.

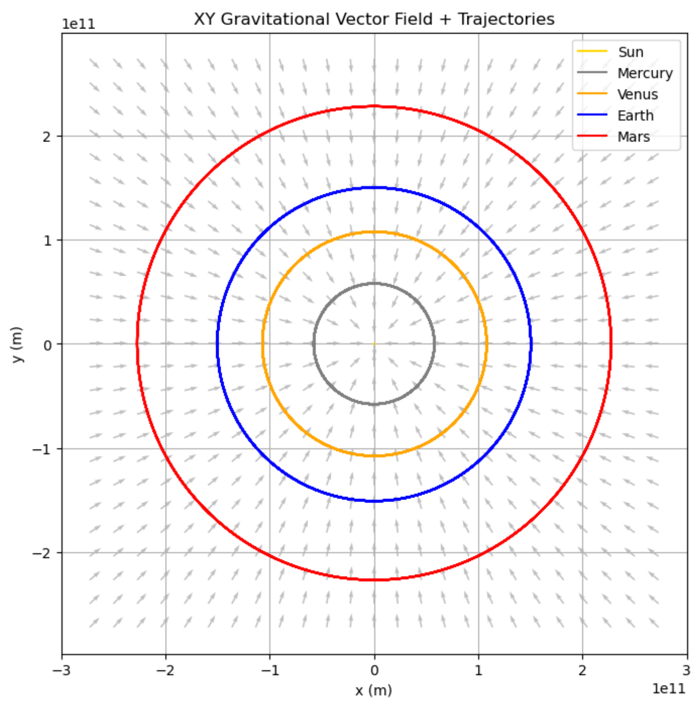


Figure 4. Vector field representation of the gravitational acceleration in the xy-plane.
Each gray arrow corresponds to the normalized acceleration vector $\mathbf{a}(\boldsymbol{x},\boldsymbol{y})$derived from Newton's law of gravitation.

This plot visualizes how the underlying differential equations— $\dot{\mathbf{x}} = \mathbf{v}$ and $\dot{\mathbf{v}} = \sum_j \boldsymbol{G}\,\boldsymbol{m}_j \mathbf{r}_{ij}/|\,\mathbf{r}_{ij}\,|^3$— generate the observed orbital motion.

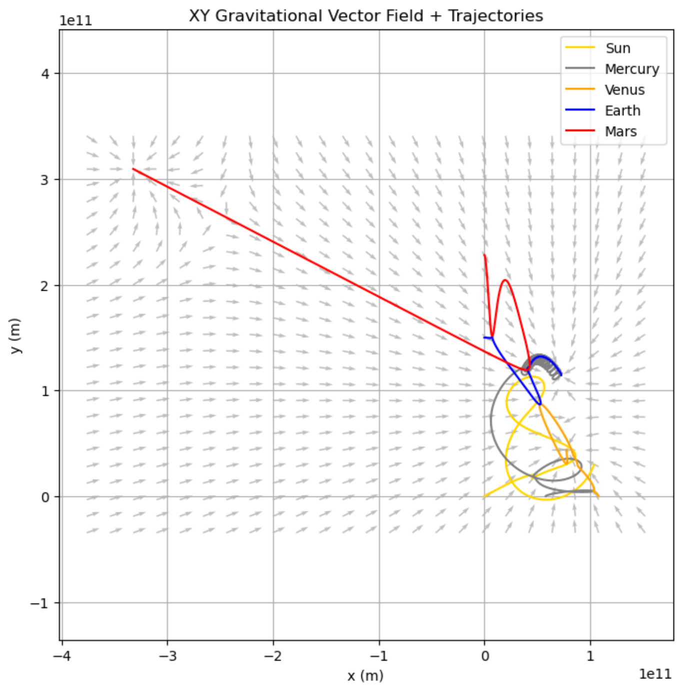
Beyond the baseline solar parameters, the simulation framework allows arbitrary modification of physical constants such as planetary masses, initial velocities, and position.

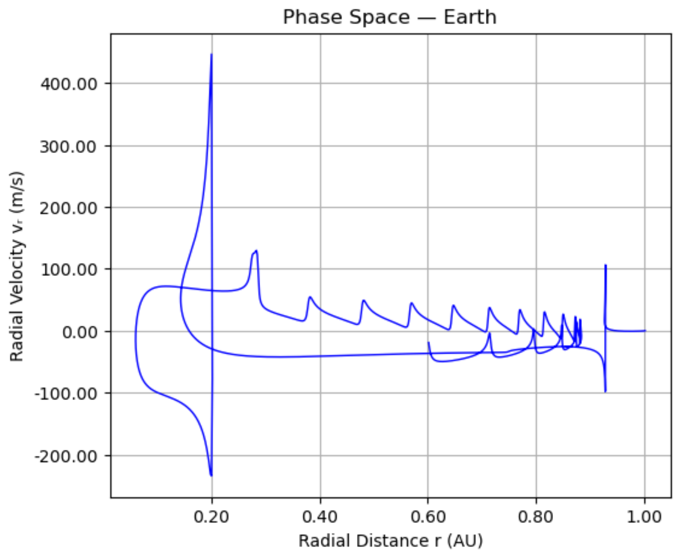
Because the equations of motion are fully general, all visualizations—including orbital trajectories, phase-space portraits, energy evolution plots, and gravitational vector-field diagrams—respond directly to these parameter changes.

When parameters are significantly altered, the system may behave in a dramatically different manner.
For example, if the Sun's mass is artificially reduced to be comparable to Earth's mass, and the initial orbital velocities of the planets are scaled downward accordingly, the system transitions

from a star–planet hierarchy into a fully interacting five-body gravitational system.
With the central mass no longer dominating the dynamics, the gravitational field becomes too weak to maintain closed Keplerian orbits.
The orbits become irregular, the trajectories no longer repeat, and close encounters between bodies can redirect motion dramatically.

In this regime, the corresponding vector-field visualization loses its previously symmetric radial structure, the phase-space loops break into chaotic patterns, and the total energy plot reflects a system that no longer resembles a stable Hamiltonian orbit around a central mass.
These extreme-parameter simulations clearly demonstrate how reducing the Sun's mass and altering initial velocities converts the model into a genuine five-body gravitational problem—illustrating both the sensitivity of Newtonian N-body dynamics and the adaptability of the numerical framework used in this study.





How to Use

This repository contains **two** Jupyter notebooks:

1. **Adjustable_widget.ipynb** — the *safe*, interactive version
2. **Myfinalproject_free_adjust.ipynb** — the *no-safety-rails* version

---

Option 1: The Widget Notebook

Adjustable_widget.ipynb includes a small interactive widget that lets you:

- Pick a planet
- Adjust its mass or initial velocity
- Re-run the simulation without touching the code
- Avoid catastrophic computational black holes

The widget is intentionally limited—you can modify **one planet at a time**, and you cannot push parameters into physically absurd territory.
These limits are deliberate. They protect your laptop, and your mental health.

If you just want to play with the system or explore modest parameter variations, **this is the notebook for you**.

Option 2: The Free-Adjust Notebook (For Advanced Users)

Myfinalproject_free_adjust.ipynb has **no widget**, no guards, and no “Are you sure?” prompts.

You can modify **any** parameter manually—for any planet.
Masses, initial positions, initial velocities … everything is editable.

To change values:

- Search for the parameter you want
- Edit it directly in the code

Simple as that.

If you set parameters too “creative” the simulation might run **very** slowly.

Like “go get coffee, come back, still running” slowly.

Proceed with caution.

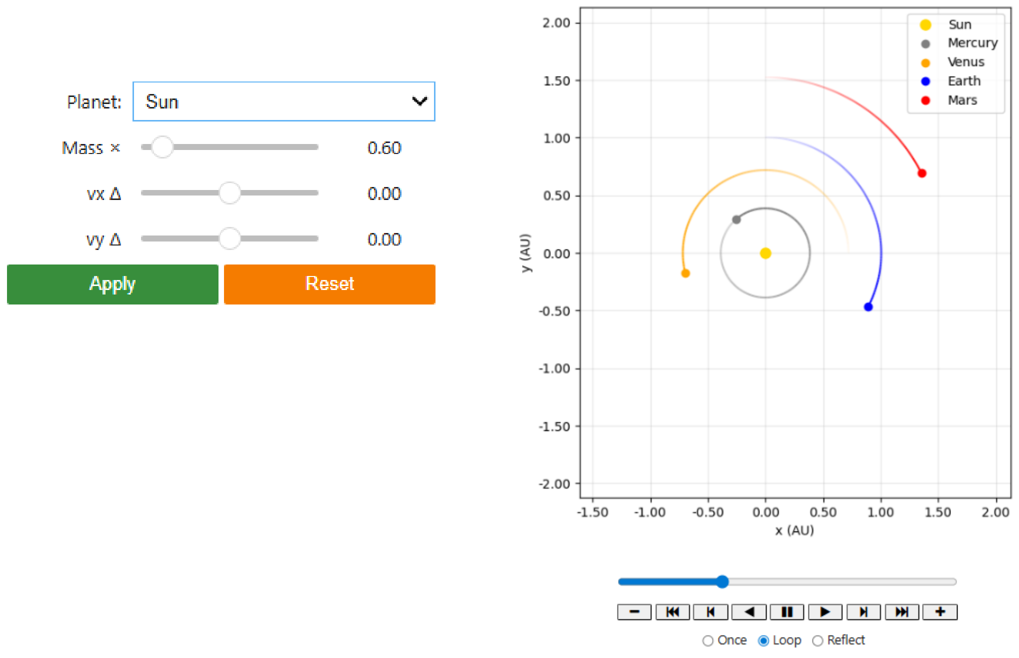
Important: Run the Notebook *Top to Bottom* Every Time

Regardless of which notebook you use:

Every time you change a parameter, you should run all cells from top to bottom in order.

This ensures:

- the new parameters propagate properly
- the solver reinitializes
- all plots update consistently



Planet: Sun

Mass × 0.60

vx Δ 0.00

vy Δ 0.00

○ Once [✓ Loop] ○ Reflect