N-body gravitational simulation

Vilius Čepaitis, Antti Pirttikoski, Steven Schramm

April 28, 2025

1 Introduction

The topic of this project is a simple N-body simulation of gravitational interactions between objects in space, treating space as a vacuum. This will be used to simulate the behaviour of our solar system, as well as other possible stellar systems. The following description gives a brief theoretical introduction to better understand the assignment.

1.1 N-body simulation

The Sun-Earth-Moon 3-body system was the original 3-body system studied in detail, and in this case the three objects have significantly different properties (masses, sizes, distances between the objects, etc). In this simple case, the Moon orbits around the Earth, and the Earth orbits around the Sun, in a relatively simple fashion. Essentially, it is not far from a pair of 2-body systems.

While 2-body simulations can be analytically solved, 3-body systems have been shown to have chaotic behaviour, meaning that tiny changes in the initial conditions can dramatically impact the subsequent dynamics of the system. This is also true for higher dimensional systems; N-body (for N $\geq$ 3) systems have no analytic solution, with exceptions for specific configurations of the objects. In contrast, it is quite simple to numerically simulate an N-body system, at least when we make some reasonable assumptions. That is therefore the focus of this project.

In our simulation, we will make two key assumptions:

1. The objects remain sufficiently separated that we do not need to worry about any interactions other than Newtonian gravity (no space-time deformations, collisions, radiation effects, etc)
2. The timesteps that we use in the simulation are sufficiently small compared to the system evolution, meaning that we can assume each object follows linear motion for a given timestep

1.2 Gravitational interactions

Under Newtonian gravity, the force of interaction between two objects is described by the masses of the two objects, m_i and m_j for $i \neq j$, and the distance between them, r_{ij}:

$$F_G^{i,j}(t) = \frac{G\,m_i m_j}{\left(r_{ij}(t)\right)^2} \tag{1}$$

with G the gravitational constant, $G = 6.674 \times 10^{-11}\,\mathrm{m}^3/\mathrm{kg}/\mathrm{s}^2$. Note that we assume the masses remain invariant with time, thus the force's dependence on time is only on the evolving value of the distance between the two objects.

Evolving the system for a given timestep Δt therefore requires calculating the interactions between a given object and all other objects, then vectorially summing all of the individual force contributions to get the net force on the object. This has to be done for every individual object, as each object contributes to the force acting on every other object. The resulting force can then be applied to a given object by exploiting Newtonian mechanics and Euler's method to approximate the updates under linear motion:

$$F_G^i(t+\Delta t) = \sum_j^{i \neq j} F_G^{i,j}(t+\Delta t) \tag{2}$$

$$a^i(t+\Delta t) = F_G^i(t+\Delta t)/m_i \tag{3}$$

$$v^i(t+\Delta t) = v^i(t) + a^i(t+\Delta t) \cdot \Delta t \tag{4}$$

$$d^i(t+\Delta t) = d^i(t) + v^i(t+\Delta t) \cdot \Delta t \tag{5}$$

where a is the acceleration, v is the velocity, and d is the displacement/position. Note that each of these quantities is a 3-dimensional vector, but numerically it's often easier to treat as three separate components: (x, y, z), (r, θ, ϕ), or so on. In addition, most astrophysical systems involve objects orbiting in a common plane, rather than in three dimensions. We will therefore make this simplifying assumption: there are only two relevant spatial dimensions, such as (x, y). In this case, the projection of the force onto these two dimensions can be calculated as:

$$F_G^{i,j}\big|_x = F_G^{i,j} \cdot \frac{|x_i - x_j|}{r_{ij}} \qquad\qquad F_G^{i,j}\big|_y = F_G^{i,j} \cdot \frac{|y_i - y_j|}{r_{ij}} \tag{6}$$

It is important to follow the above calculations very carefully. In particular, you need to calculate all of the forces $F_G^i(t+\Delta t)$ before applying any of them, or else the system will not evolve coherently. In other words, if you calculate and immediately apply the force object by object, then the first object will receive the intended correction, the second one will receive the right force from all objects except the first one, the third from all expect the first two, and so on.

1.3 Reducing calculation dimensionality

Given that the calculation of the force acting on a given object depends on all other objects, a simple brute-force calculation of the forces acting on N objects scales as $\mathcal{O}(N^2)$. While such scaling doesn't sound too bad, recall that the Solar System contains nearly 300 moons, more than a million asteroids (mostly between Mars and Jupiter), and innumerable other objects beyond the orbit of Neptune. It thus is impossible to calculate the force for a single timestep if we consider the full Solar System, never mind a larger entity (our galaxy, the Universe, etc). Moreover, we need to perform this calculation repeatedly: we need Δt to be small enough that the linear motion approximation holds, and thus the number of timesteps in our simulation will be very large.

The complexity of calculating the force at each timestep, coupled with the required number of timesteps, can quickly become computationally challenging; it is therefore important in such simulations to also consider means of simplifying the calculations. For example, the net impact of the million asteroids in the Solar system should exert a negligible force on the Sun: their mass is individually so small, and they are also distributed roughly isotropically (uniformly) around the Sun, thus their force contributions cancel out to first order.

This principle can be extended: as a first order approximation, each "category" of object doesn't care much about anything from lower categories. The Sun will not be significantly impacted by planets, planets are not significantly impacted by their moons, moons are not significantly impacted by asteroids, etc. This is not strictly speaking true, but for the baseline example, we will start with this assumption to simplify the calculation. Concretely:

$$F_G^i(t+\Delta t) = \sum_j^{i\neq j,\ C_j \geq C_i} F_G^{i,j}(t+\Delta t) \tag{7}$$

where C_i and C_j are the "category" of the object, and where the category value of a star would be larger than that of a planet.

This can be further extended, as we know that distance is also an important factor. For example, while the moons of Jupiter may feel the pull of Saturn, they don't really feel the pull of Saturn's moons. As such, we can assume that objects of a given category only feel the pull of other objects of the same category if they orbit the same parent object.

In order to perform this simplification, we will make use of different classes. You should create separate classes for each of the different categories of objects, and check if the type of a given object j has a category that is of relevance to the object under study i; if so, calculate and add the force from that object, otherwise skip that instance of j and move to the next entry.

2 Baseline project requirements

The code should follow the object-oriented programming (OOP) paradigm and respect the following structure; the exact naming to use is left as a choice for the author.

Object

This is the abstract base class that represents an object in the N-body simulation. The class must be able to describe its own state in terms of mass, position, and velocity. It should also be able to be aware of the object(s) that orbit it, such as moons orbiting a planet, and of the object(s) that it orbits, such as a planet orbiting a star. This association information is useful for the dimensionality reduction procedure. The following methods should be implemented:

- A method to get the mass of the object
- A method to get the position components of the object, such as (x, y)
- A method to get the velocity components of the object, such as (v_x, v_y)
- A method to get the kinetic energy of the object, using $E_{\text{kin}} = \frac{1}{2}m|v|^2$
- A method to get the parent(s) of the object, which the current object is orbiting [if any]
- A method to get the children of the object, which orbit the current object [if any]
- A way to update and/or evolve the state of the object
- A method to print the object, convert it to string, or similar as appropriate
- An abstract method to get the relevant objects to use for calculating the force (for dimensionality reduction)

Star, Planet, Moon, etc

The `Object` is the abstract base class, which requires concrete implementations. Natural candidates to represent the Solar System are `Star`, `Planet`, `Moon`, and so on, but the naming and division is ultimately up to you. These derived classes must implement the method that provides the relevant objects to use when calculating the force, and may also have other functionality as you deem useful.

System

The `Star` when simulating a Solar System could be the top-level object, but when you are considering a binary star system, you may need something even larger. A `System` is one candidate name for the top-level representation. Note that even if you start to simulate pairs of colliding galaxies, the two individual galaxies could each be `System` instances, and those instances could both be part of a larger `System`. The name and implementation here can differ significantly, so this may or may not inherit from `Object` as you deem best for your implementation.

- The `System` must be able to calculate the total kinetic energy at the current point in time, summing up the individual kinetic energy of each object in the system
- The `System` must be able to calculate the total potential energy at the current point in time, summing up the potential energy of all pairs of objects in the system, using $E_{\text{pot}}^{i,j} = F_G^{i,j} \cdot r_{i,j}$
  - Make sure not to double-count the energy; each pair should be considered once, not twice.

main

The main method should do the following, either directly within the main method or in additional classes/methods that you may define:

- Read an input file specifying the parameters of all of the `Object` instances to be created, and the relations between them. See Tables 1 and 2 for the parameters to use; it is up to you to encode them as you prefer in an input file to be read by your code.
- Create the top-level `System`, creating/adding all of the above-mentioned `Object` instances to it, as appropriate.
- Run the simulation, for one million timesteps, writing every 1000^{th} timestep to an output file. This difference allows for a fine-detail simulation to mitigate the limitations of the approximations we have made, while avoiding creating enormous output files with data points that are too close together to be useful in plotting.
  - Print out the state of each object, plus the current time, the total kinetic energy, and the total potential energy.

3 Baseline expected results

The baseline requirement of the project is to perform a simulation of both the inner part of the Solar System and of an artificial binary star system, demonstrating the utility of N-body simulations of gravitational interactions. Moreover, we will study the impact of our simplifying assumption of linear motion on energy conservation.

- Implement the required classes to run the simulations, together with a means of simplifying the N-body force calculations by defining which objects are relevant for a given calculation.
- Run the simulations for the parameters given in Tables 1 (Solar System) and 2 (binary system).
- Plot the results of these simulations, which should look similar to Fig. 1 and Fig. 2.
- Try varying the step size from the nominal simulation, always keeping a total simulation duration of one Earth-year, but changing the number of time slices that is divided into. Study the impact of timestep size on energy conservation, and plot the results.
  - In a perfect simulation, the total energy $E_{\text{tot}} = E_{\text{kin}} + E_{\text{pot}}$ should be stable, which you are unlikely to see in your implementation.

  - Note: you can check your implementations of E_{pot} and E_{kin} by comparing with the expectation of the Virial theorem: $E_{\mathrm{pot}} \approx -2E_{\mathrm{kin}}$

- For the Solar System simulation, study the impact of entirely removing Mars on the motion of its two moons, over one Earth-year of simulation (in other words, use the same simulation parameters). Remake the bottom-left plot in Fig 1. What do the moons do?

Quantity	**Value to use**
The Sun	Star
Mass	1.98855×10^{30} kg
(x, y)	(0,0) meters
(v_x, v_y)	(0,0) m/s
Mercury	Planet
Mass	3.3011×10^{23} kg
(x, y)	$(5.790905 \times 10^{10}, 0)$ meters
(v_x, v_y)	$(0, 4.7362 \times 10^{4})$ m/s
Venus	Planet
Mass	4.8675×10^{23} kg
(x, y)	$(1.08208 \times 10^{11}, 0)$ meters
(v_x, v_y)	$(0, 3.502 \times 10^{4})$ m/s
Earth	Planet
Mass	5.97237×10^{24} kg
(x, y)	$(1.49598023 \times 10^{11}, 0)$ meters
(v_x, v_y)	$(0, 2.978 \times 10^{4})$ m/s
The Moon	Moon (Earth)
Mass	7.342×10^{22} kg
(x, y)	$(3.84399 \times 10^{8}, 0)$ meters, relative to Earth
(v_x, v_y)	$(0, 1.022 \times 10^{3})$ m/s, relative to Earth
Mars	Planet
Mass	6.4171×10^{23} kg
(x, y)	$(2.279392 \times 10^{11}, 0)$ meters
(v_x, v_y)	$(0, 2.4007 \times 10^{4})$ m/s
Phobos	Moon (Mars)
Mass	1.0659×10^{16} kg
(x, y)	$(9.376 \times 10^{6}, 0)$ meters, relative to Mars
(v_x, v_y)	$(0, 2.138 \times 10^{3})$ m/s, relative to Mars
Deimos	Moon (Mars)
Mass	1.4762×10^{15} kg
(x, y)	$(-2.34632 \times 10^{7}, 0)$ meters, relative to Mars
(v_x, v_y)	$(0, -1.3513 \times 10^{3})$ m/s, relative to Mars
Comet	TransientObject
Mass	2.2×10^{14} kg
(x, y)	$(5 \times 10^{11}, -9 \times 10^{10})$ meters
(v_x, v_y)	$(-7 \times 10^{4}, 1 \times 10^{4})$ m/s
Simulation	Number of timesteps: 10^6 How often to print timesteps to output file: $1/10^3$ Timestep size: 31.5576 seconds ($1/10^6$ of a year)

Table 1: Parameter values to use for simulating the inner part of the Solar System. These values are what was used to make the plots shown in the baseline results section.

Quantity	**Value to use**
Star #1	Star
Mass	$m_{\text{sun}}/2$
(x, y)	$(+1 \times 10^{10}, 0)$ meters
(v_x, v_y)	$(0, +3 \times 10^4)$ m/s
Star #2	Star
Mass	$m_{\text{sun}}/2$
(x, y)	$(-1 \times 10^{10}, 0)$ meters
(v_x, v_y)	$(0, -3 \times 10^4)$ m/s
Planet #1	Planet
Mass	3×10^{23} kg
(x, y)	$(+5 \times 10^{10}, 0)$ meters, relative to Star #1
(v_x, v_y)	$(0, +3 \times 10^3)$ meters, relative to Star #1
Planet #2	Planet
Mass	3×10^{23} kg
(x, y)	$(-5 \times 10^{10}, 0)$ meters, relative to Star #2
(v_x, v_y)	$(0, -3 \times 10^3)$ meters, relative to Star #2
Simulation	Number of timesteps: 10^6 How often to print timesteps to output file: $1/10^3$ Timestep size: 6 seconds

Table 2: Parameter values to use for simulating the (artificially created) binary star system. These values are what was used to make the plots shown in the baseline results section.

3.1 Possible extensions

There are several ways to extend the baseline model. Here are a few possibilities to consider:

- Add a way for the user to specify the "origin" of the simulation for output purposes, such as allowing the user to switch between Sun-centric and Earth-centric reference frames, or any other reference frame of their choice.
- Simulate Jupiter and/or Saturn, together with their numerous moons of different sizes, for one Jupiter or Saturn year as appropriate.
- Modify the dimensionality reduction procedure to group smaller objects with the object that they are orbiting, rather than neglecting them entirely. For example, instead of having all of the planets (other than the Earth) ignoring the Moon, add the mass of the Moon to the mass of Earth, and use the combined mass when interacting with other objects.
- Improve the dimensionality reduction procedure, which can be done in many different ways. Then, study the differences between brute force, the simple dimensionality reduction procedure used in the baseline project, and your new dimensionality reduction procedure.
- Investigate higher-order methods to simulate object motion, going beyond the linear motion approximation based on Euler's method.
- Demonstrate the stability of "small" objects at the Earth-Sun Lagrange points.
- Extend the simulation to consider specific inelastic interactions, such as when a star consumes a planet/moon/etc.
- Create an animation (gif or similar) of the evolution of the system(s) considered.

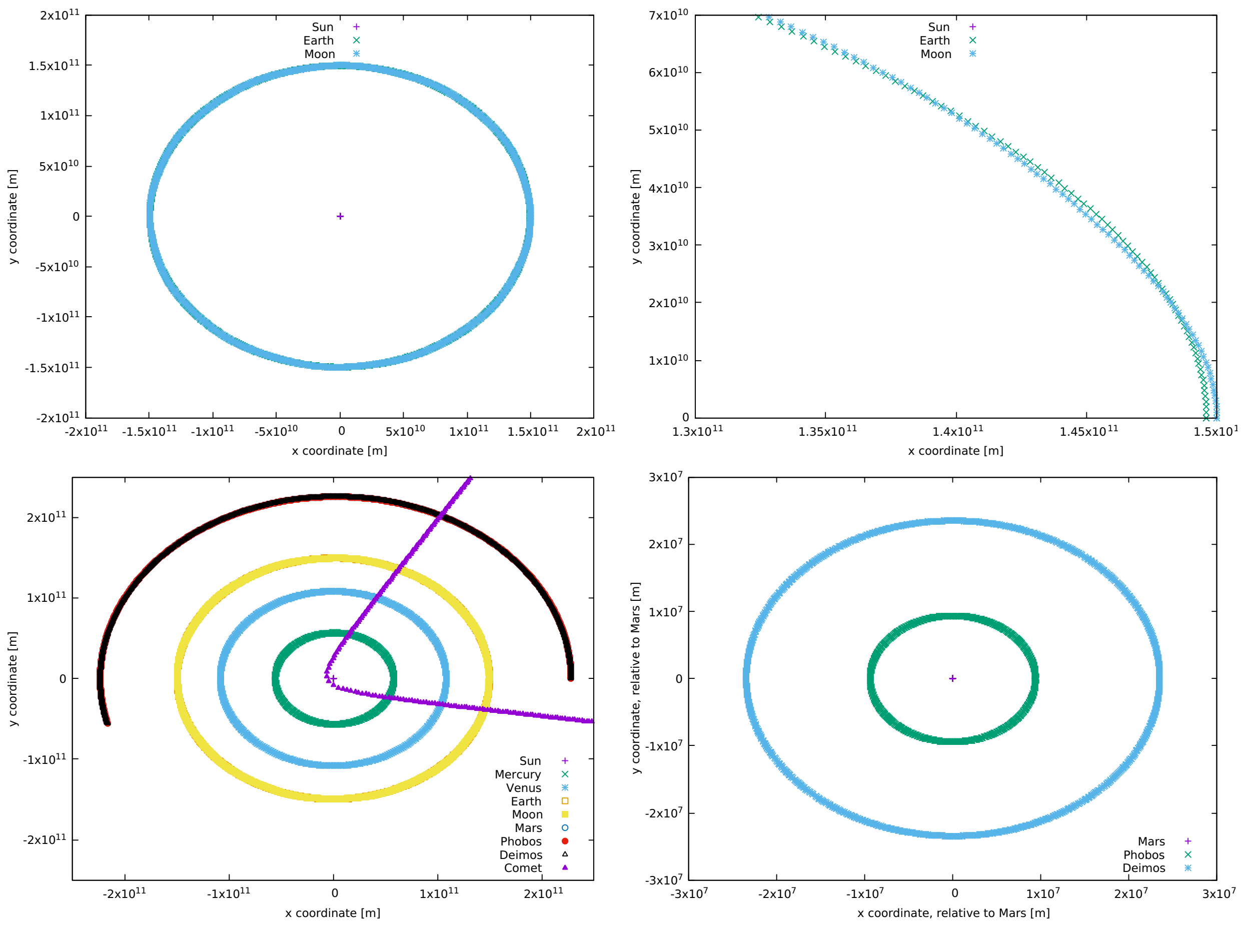

Figure 1: Top row: the motion of the Earth and Moon around the Sun (left) and a zoom-in such that the Moon's orbit around the Earth can be observed (right). Bottom row: a view of the entire inner Solar System (left) and a zoom-in of Mars as the reference point (right).

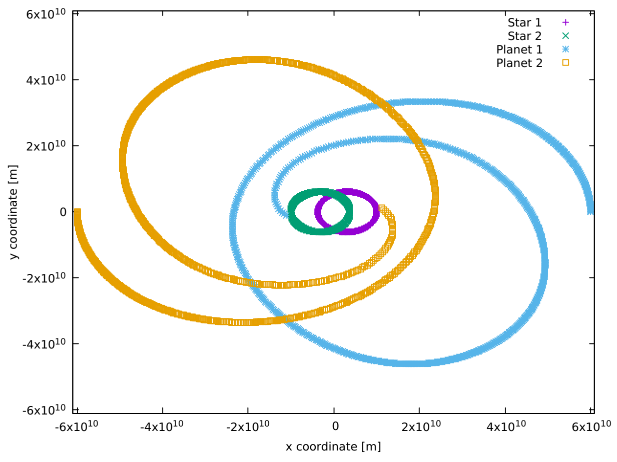

Figure 2: The evolution of a binary star system, each with half of the mass of the Sun, plus two arbitrary planets with roughly the same mass as Mercury. For this choice of initial conditions, the planets of the system decay into the stars, breaking the assumption of the simulation that the distance between objects remains sufficiently large for the gravitational force to be the only force of relevance.