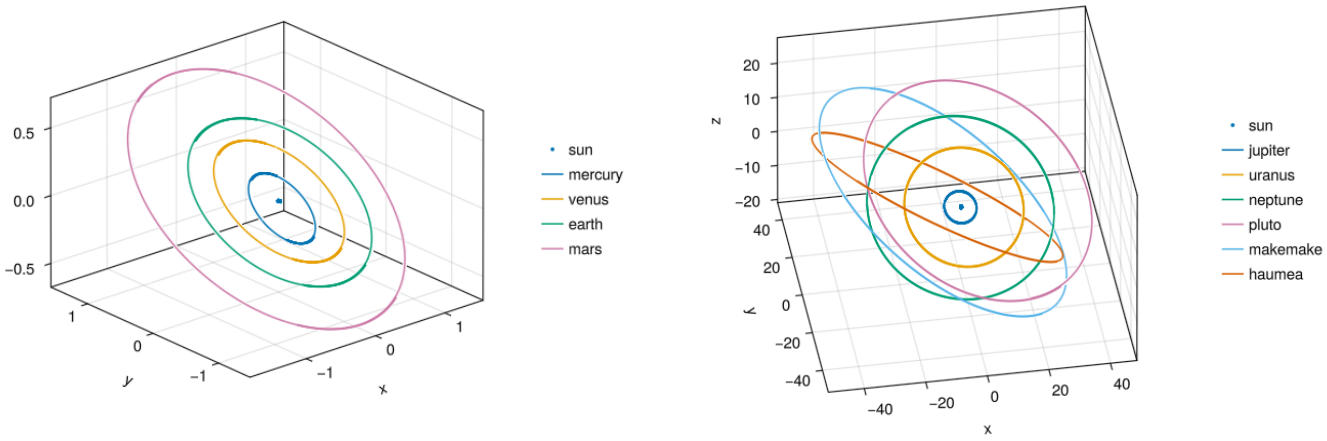
Problem Set 4 – Simple Hamiltonian Integrators for the Solar system

Our goal is to write Julia code that solves numerically the equations of motion for our solar system and visualizes the solution similar to the following figures:



(a) The inner solar system

(b) Some objects of the outer solar system

Figure 1: A Simulation of the solar system

1 Hamilton mechanics in phase space

- The mechanical state of N bodies moving in space is described by $3N$ coordinates q_i and $3N$ momenta[1] p_i where $i = 1, ..., 3N$.
- The *phase space* of a N-body system ist $\mathbb{R}^{6N}$ with coordinates $\mathbf{z} = (\mathbf{q}, \mathbf{p})$.
- Time evolution is a curve $\mathbf{z}(t) = (\mathbf{q}(t), \mathbf{p}(t))$ in phase space.
- The *Hamiltonian* $H(\mathbf{q}, \mathbf{p})$ is the total mechanical energy of the system written as function of $\mathbf{q}$ and $\mathbf{p}$. Given an initial point in phase space, the future evolution of the system is completely determined by Hamilton's equations:

$$\frac{d}{dt} q_i(t) = \frac{\partial H(\mathbf{q}, \mathbf{p})}{\partial p_i} \tag{1}$$

$$\frac{d}{dt} p_i(t) = -\frac{\partial H(\mathbf{q}, \mathbf{p})}{\partial q_i} \tag{2}$$

  This is a system of $6N$ first-order ordinary differential equations[2] (ODEs) defining the flow of time in phase space.

[1] In our case the momentum is just velocity multiplied by mass, $\vec{p} = m\vec{v}$. So you can picture the momentum vector as a velocity vector, scaled by the mass of the body.

[2] The partial derivatives on the RHS only tell us how to calculate the RHS from the function H. They don't make it a PDE.

Notation. *We will denote vectors in $\mathbb{R}^3$ (position, momentum of one body) by an arrow, $\vec{q}$. Higher-dimensional vectors are denoted by bold symbols like $\mathbf{p}$ and $\mathbf{q}$. Indices a, b will enumerate the N bodies and indices i, j denote the components of a vector. The index n will always enumerate time steps.*

The total energy is the sum of kinetic energy and potential energy due to gravitational

attraction:

$$H(\mathbf{p},\mathbf{q}) = T(\mathbf{p}) + V(\mathbf{q}) \tag{3}$$

$$H(\mathbf{p},\mathbf{q}) = \sum_{a=1}^{N} \frac{\vec{p}_a^2}{2m_a} - \frac{G}{2}\sum_{a=1}^{N}\sum_{b\neq a}^{N} \frac{m_a m_b}{|\vec{q}_a - \vec{q}_b|} \tag{4}$$

Here, m_a is the mass of the a-th body, G is the gravitational constant and $|\vec{q}_a - \vec{q}_b|$ is the distance between bodies a and b. The resulting Hamilton's equations are

$$\frac{d}{dt}q_{a,i}(t) = \frac{\partial T(\mathbf{p})}{\partial p_{a,i}} = \frac{p_{a,i}}{m_a}; \qquad a = 1,\dots,N; \quad i = 1,\dots,3; \tag{5}$$

$$\frac{d}{dt}p_{a,i}(t) = -\frac{\partial V(\mathbf{q})}{\partial q_{a,i}} = G\sum_{b\neq a} \frac{m_a m_b}{|\vec{q}_a - \vec{q}_b|^3}\left(q_{b,i} - q_{a,i}\right). \tag{6}$$

2 Simple numerical solver for ODEs

[3] Autonomous: the f_i do not depend explicitely on t

Our equations of motion (1,2) are first order, autonomous[3] in a higher-dimensional linear space $\mathbb{R}^n$ (with $n = 6N$ in our case):

$$\frac{d}{dt}z_1(t) = f_1(z_1(t),\dots,z_n(t)) \tag{7}$$

$$\frac{d}{dt}z_2(t) = f_2(z_1(t),\dots,z_n(t)) \tag{8}$$

$$\vdots$$

$$\frac{d}{dt}z_n(t) = f_n(z_1(t),\dots,z_n(t)). \tag{9}$$

with initial conditions $z_i(0) = z_{0,i}$ We express this concisely using vector notation as:

$$\frac{d}{dt}\mathbf{z}(\mathbf{t}) = \mathbf{f}(\mathbf{z}(\mathbf{t})); \qquad \mathbf{z}(0) = \mathbf{z}_0. \tag{10}$$

A numerical method generates approximations $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \dots$ to the solution $\mathbf{z}(t)$ at various values of time t. We will only consider integrators with fixed step size h; so $\mathbf{z}_n$ is an approximation to $z(t = nh)$.

2.1 Explicit Euler method (Euler 1786)

Choose a small enough value h for your time steps.

$$t_{n+1} = t_n + h$$
$$\mathbf{z}_{n+1} = \mathbf{z}_n + h\mathbf{f}(\mathbf{z}_n)$$

2.2 Heun's method

Heun's is the simplest of a family called predictor-corrector methods. First we make a preliminary Euler step (prediction), calculate the slope $\mathbf{f}$ at this point and then make our (corrected) step using the average of the slopes at $\mathbf{z}_n$ and at $\tilde{\mathbf{z}}_{n+1}$:

$$t_{n+1} = t_n + h$$
$$\tilde{\mathbf{z}}_{n+1} = \mathbf{z}_n + h\mathbf{f}(\mathbf{z}_n)$$
$$\mathbf{z}_{n+1} = \mathbf{z}_n + \frac{h}{2}\left(\mathbf{f}(\mathbf{z}_n) + \mathbf{f}(\tilde{\mathbf{z}}_{n+1})\right)$$

2.3 Størmer-Verlet method

Euler and Heun are general methods. Symplectic integrators are adapted to the special structure of Hamiltonian equations. One of the simplest symplectic integrators is the Størmer-Verlet or leapfrog method:

$$t_{n+1} = t_n + h \tag{11}$$

$$p_{i,n+\frac{1}{2}} = p_{i,n} - \frac{1}{2} h \frac{\partial V}{\partial q_i}(\mathbf{q}_n) \tag{12}$$

$$q_{i,n+1} = q_{i,n} + h \frac{\partial T}{\partial p_i}(\mathbf{p}_{n+\frac{1}{2}}) \tag{13}$$

$$p_{i,n+1} = p_{i,n+\frac{1}{2}} - \frac{1}{2} h \frac{\partial V}{\partial q_i}(\mathbf{q}_{n+1}), \qquad i = 1, \dots, 3N \tag{14}$$

3 Implementation in Julia

Your task is to write Julia code that numerically integrates the equations of motions for the solar system using different integrators.

- The file `ephemerides.jl` contains initial data (positions and velocities) and masses for the Sun, the 8 planets and 3 dwarf planets (Pluto, Makemake, Haumea).

  ```
  include("ephemerides.jl")
  using .Ephemerides
  ```

  The initial data correspond to 2025-Jan-01, 00:00 UTC. Distances are given in astronomical units (1au = 149 597 870.7km, the average Earth-Sun distance), times in Earth days (1d = 86 400s), masses as multiples of the solar mass $\mathrm{M}_\odot$. In these units, the gravitational constant is

  $$G = 2.95912208286 \times 10^{-4} \frac{\mathrm{au}^3}{\mathrm{d}^2 \mathrm{M}_\odot} \tag{15}$$

  which is also provided by `ephemerides.jl`.
- It is recommended to use the package `Distances.jl` for the fast calculation of all pairwise distances. If, e.g, the coordinates are available in a $3 \times N$-matrix `q`, then

  ```
  using Distances
  ...
  D = pairwise(Euclidean(), q, dims=2)
  ```

  returns a symmetric $N \times N$ matrix where `D[i,j]` is the distance between body `i` and body `j`.
- Do not use packages like `DifferentialEquations.jl`; write your own numerical integrator! (You may use them to compare and check your code.)
- Use a plot package, e.g., `Plots.jl` or `GLMakie.jl` to visualize the orbits similar to Figure 1.
- Calculate the total energy H of the system every 20 steps. Check the conservation of energy for the different methods (Euler, Heun, Størmer-Verlet). Plot the energy over time.
- Suggested parameters:
  (a) inner solar system: step size $h = 0.2d$, 10^4 time steps
  (b) outer solar system: step size $h = 10d$, 10^5 time steps