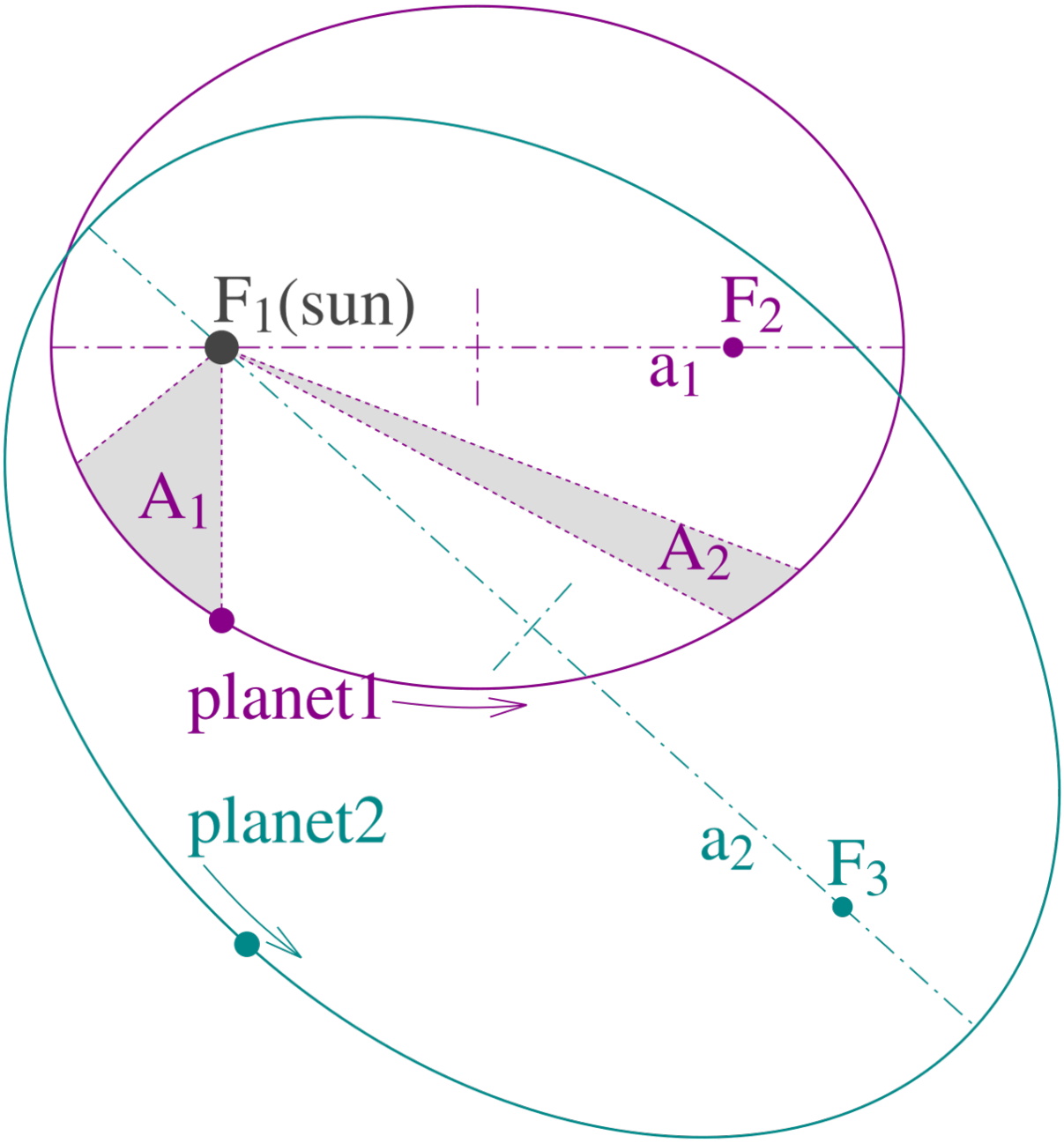
Projection Methods for the Kepler Problem

Oleksandr Samoliuk

Summer Semester, 2025

Abstract

This report provides an overview of projection methods applied to the classical Kepler problem. These methods are particularly useful in preserving geometric structures and long-term stability in the numerical integration of the system.



Contents

1 Introduction to ODEs

This is a small introduction to Ordinary Differential Equations, or ODEs for short.

Ordinary Differential Equations (ODEs) are mathematical equations that relate a function of a single independent variable to its derivatives. The term *ordinary* distinguishes them from partial differential equations (PDEs), which involve multiple independent variables. ODEs arise naturally in modeling the dynamic behavior of systems in physics, biology, economics, and engineering, such as population growth, mechanical motion, electrical circuits, and heat conduction over time.

A general n-th order ordinary differential equation can be written in the implicit form as:

$$F\left(t, u(t), \dot{u}(t), \ddot{u}(t), \ldots, u^{(n)}(t)\right) = 0, \quad \forall t \geq 0,$$

where:

- t is the independent variable,
- $u(t)$ is the unknown function,
- $\dot{u}(t), \ddot{u}(t), u^{(n)}(t)$ denote the first through n-th derivatives of u with respect to t,
- F is a given function defining the relationship among these quantities.

The highest derivative present in the equation determines the order of the ODE. In order to have a chance to solve the differential equations, we need initial conditions, i.e., we have to prescribe the values of $u(0), \ldots, u^{(n-1)}(0)$. The function u can either map to scalars (in $\mathbb{R}$) or to vectors (in $\mathbb{R}^n$).

However, in our case, it would be easier to adhere to such a formulation:

$$\begin{cases} \dot{u}(t) = f(t, u(t)) \\ u(0) = u_0 \end{cases}$$

where $f(t, y)$ is a given function and $u(0) = u_0$ is the given initial condition.

Although some ODEs can be solved analytically, many real-world systems lead to equations too complex for exact solutions. In such cases, numerical methods are employed to approximate solutions at discrete points. Common numerical approaches include Euler's method, Runge–Kutta methods, and multistep methods. Each method aims to achieve **consistency** and **discrete stability** which result in local convergence.

$$\text{Consistency} + \text{(Discrete) Stability} \Rightarrow \text{Convergence}$$

A thorough understanding of ordinary differential equations and their numerical solutions is fundamental for the simulation and analysis of dynamic systems across scientific disciplines. This article focuses on the classical Kepler problem, introducing its mathematical formulation and emphasizing the role of **projection methods**—specialized numerical techniques that preserve key geometric structures and conservation laws inherent in the physical dynamics of the system.

2 Kepler Problem Formulation

This section will elaborate on key properties of the Kepler problem and also feature its derivation.

2.1 Physical Meaning

The Kepler problem models the motion of a body orbiting a central massive object under the influence of gravity. It is based on the following assumptions:

- The central force obeys an inverse-square law (i.e., proportional to $\frac{1}{r^2}$, where r is the distance between bodies).
- The orbiting body's mass is negligible compared to that of the central object.
- The motion is confined to a plane, allowing a reduction to two dimensions.

This setup leads to a system of differential equations that govern the evolution of the particle's position and velocity over time, driven by the gravitational attraction toward the fixed center.

The system of equations can be expressed as follows:

$$\begin{cases} \ddot{x} = -\dfrac{MGx}{(x^2+y^2)^{3/2}} \\ \ddot{y} = -\dfrac{MGy}{(x^2+y^2)^{3/2}} \end{cases}$$

where:

- x is the horizontal position of the particle,
- $\ddot{x}$ is the acceleration of the particle in x direction,
- y is the vertical position of the particle,
- $\ddot{y}$ is the acceleration of the particle in y direction,
- M is the mass of the central body,
- G is the gravitational constant.

For simplicity, we will set the M and G to 1.

After obtaining solutions for different initial conditions, the resulting trajectories capture key phenomena, including elliptical orbits, conservation of angular momentum, and energy dynamics.

2.2 Derivation

The equations of motion are derived from Newton's second law and the formula of gravitational potential.

At first, we define the position and acceleration vectors:

$$p = \begin{bmatrix} x \\ y \end{bmatrix} \qquad \ddot{p} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix}$$

We begin with the expression for the gravitational potential energy:

$$U(x,y) = -\frac{MmG}{\sqrt{x^2+y^2}},$$

where m is the mass of the orbiting body.

The gravitational force, which describes how potential energy depends on position, is derived from this potential using the negative gradient:

$$\vec{F}_{grav} = -\nabla U(x, y)$$

We compute the negative gradient of $U(x, y)$:

$$-\nabla U(x, y) = \begin{bmatrix} \frac{\partial}{\partial x} U(x, y) \\ \\ \frac{\partial}{\partial y} U(x, y) \end{bmatrix} = \begin{bmatrix} -\frac{MmGx}{(x^2+y^2)^{\frac{3}{2}}} \\ \\ -\frac{MmGy}{(x^2+y^2)^{\frac{3}{2}}} \end{bmatrix}$$

Now, by Newton's second law:

$$\vec{F}_{grav} = m\vec{a} = m\ddot{p}$$

Hence:

$$m\ddot{p} = \begin{bmatrix} -\frac{MmGx}{(x^2+y^2)^{\frac{3}{2}}} \\ \\ -\frac{MmGy}{(x^2+y^2)^{\frac{3}{2}}} \end{bmatrix}$$

We achieve the standard form of the problem by setting m, M, G to one:

$$\begin{cases} \ddot{x} = -\dfrac{x}{(x^2+y^2)^{3/2}} \\ \ddot{y} = -\dfrac{y}{(x^2+y^2)^{3/2}} \end{cases} \qquad \text{or using } p \text{ notation} \qquad \ddot{p} = -\frac{p}{(||p||_2)^3}$$

The derivation begins by combining Newton's second law, which relates force to mass and acceleration, with the formula for gravitational potential energy, which describes how potential energy depends on position. Recognizing that the force acting on a particle in a gravitational field is the negative gradient of this potential energy, we substitute the expression $-\nabla U$ for the force in Newton's law. This leads to a system of differential equations that describe how the position of the particle changes over time under the influence of gravity, producing precise expressions for the accelerations $\ddot{x}$ and $\ddot{y}$ in terms of the particle's coordinates.

The resulting equations tell us how the planet accelerates toward the central mass, producing elliptical orbits depending on initial conditions. In summary, the Kepler problem describes the motion of a particle of a certain under the influence of an inverse-square central force, for instance, gravity.

2.3 Task Reformulation

Numerical integrators presented in the following section are used to solve first-order equations of the form:

$$\dot{u}(t) = f(t, u(t))$$

However, the Kepler Problem is naturally formulated as a system involving second-order derivatives. To apply our numerical methods, we reformulate the problem into an equivalent system of first-order differential equations, making it compatible with the integrators.

Since the first derivative of position is velocity ($v_x = \dot{x},\ v_y = \dot{y}$), we can introduce new variables for the velocities. This reduces the second-order system into a larger system of first-order equations:

$$\begin{cases} \ddot{x} = -\dfrac{x}{(x^2+y^2)^{3/2}} \\ \ddot{y} = -\dfrac{y}{(x^2+y^2)^{3/2}} \end{cases} \quad \Rightarrow \quad \begin{cases} \dot{v}_x = -\dfrac{x}{(x^2+y^2)^{3/2}} \\ \dot{v}_y = -\dfrac{y}{(x^2+y^2)^{3/2}} \\ \dot{x} = v_x \\ \dot{y} = v_y \end{cases}$$

This system can be expressed concisely in matrix-vector notation, which is especially useful when applying numerical methods:

$$\dot{u} = f(u)$$

where:

$$u = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}, \quad f(u) = \begin{bmatrix} v_x \\ v_y \\ -\dfrac{x}{(x^2+y^2)^{3/2}} \\ -\dfrac{y}{(x^2+y^2)^{3/2}} \end{bmatrix}$$

2.4 The Picard–Lindelöf Theorem

Theorem 2.1 (Picard–Lindelöf *(global variant)*)**.** *Let $u_0 \in \mathbb{R}^N$ and $f : [0,T] \times \mathbb{R}^N \to \mathbb{R}^N$ be continuous and uniformly Lipschitz continuous with respect to the second argument, i.e., there exists a constant $L > 0$ such that*

$$\|f(t,z) - f(t,\tilde{z})\|_{\ell_2} \leq L\|z - \tilde{z}\|_{\ell_2}, \quad \forall t \in [0,T],\ z, \tilde{z} \in \mathbb{R}^N.$$

Then

$$\begin{cases} \dot{u}(t) = f(t,u(t)), & t \in (0,T] \\ u(0) = u_0 \end{cases}$$

has a unique solution $u \in C^1([0,T];\mathbb{R}^N)$.

Theorem 2.2 (Picard–Lindelöf *(local variant)*)**.** *Let $\Omega \subset [0,T] \times \mathbb{R}^N$ be an open set, and let $f : \Omega \to \mathbb{R}^N$ be continuous. Suppose that f satisfies a local Lipschitz condition with respect to the second argument, that is: for every compact subset $\mathcal{K} \subset \Omega$, there exists a constant $L_{\mathcal{K}} > 0$ such that*

$$\|f(t,z) - f(t,\tilde{z})\|_{\ell_2} \leq L_{\mathcal{K}}\|z - \tilde{z}\|_{\ell_2}, \quad \forall (t,z), (t,\tilde{z}) \in \mathcal{K}.$$

Then, for every initial value $(t_0, u_0) \in \Omega$, there exists a subinterval $I_0 \subset \mathbb{R}$ containing t_0, and a unique continuously differentiable function $u \in C^1(I_0;\mathbb{R}^N)$ solving the initial value problem

$$\begin{cases} \dot{u}(t) = f(t,u(t)), & t \in I_0 \\ u(t_0) = u_0 \end{cases}$$

such that $(t,u(t)) \in \Omega$ for all $t \in I_0$. Moreover, the solution $u(t)$ can be uniquely extended up to the boundary of the domain Ω.

The Lipschitz condition ensures that small changes in y do not result in wildly different outcomes, which guarantees that the solutions do not cross or behave erratically. It is also possible to check whether the Lipschitz continuity is satisfied by understanding if the function f:

- is continuously differentiable,

- has a bounded derivative.

Let us now check whether this holds for the Kepler problem case. We begin by taking a look at u and $f(u)$.

$$f(u) = \begin{bmatrix} v_x \\ v_y \\ \frac{x}{(x^2+y^2)^{\frac{3}{2}}} \\ \frac{y}{(x^2+y^2)^{\frac{3}{2}}} \end{bmatrix} \qquad u = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}$$

Next, we move on to calculating the Jacobian of f.

$$J_f = \begin{bmatrix} \nabla f_1(u) \\ \nabla f_2(u) \\ \nabla f_3(u) \\ \nabla f_4(u) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{1}{(x^2+y^2)^{\frac{3}{2}}} - \frac{3x^2}{(x^2+y^2)^{\frac{5}{2}}} & -\frac{3xy}{(x^2+y^2)^{\frac{5}{2}}} & 0 & 0 \\ -\frac{3xy}{(x^2+y^2)^{\frac{5}{2}}} & \frac{1}{(x^2+y^2)^{\frac{3}{2}}} - \frac{3y^2}{(x^2+y^2)^{\frac{5}{2}}} & 0 & 0 \end{bmatrix}$$

Let us take a look at the components of the computed Jacobian. At first we should change variables to polar coordinates:

$$x = r\cos\theta, \quad y = r\sin\theta, \quad \text{and} \quad x^2 + y^2 = r^2$$

Now consider, for instance, the component:

$$J_{f,3,1}(x,y) = \frac{1}{(x^2+y^2)^{3/2}} - \frac{3x^2}{(x^2+y^2)^{5/2}}$$

Substituting in polar coordinates produces the following result:

$$J_{f,3,1}(r,\theta) = \frac{1}{r^3} - \frac{3r^2\cos^2\theta}{r^5} = \frac{1}{r^3} - \frac{3\cos^2\theta}{r^3} = \frac{1-3\cos^2\theta}{r^3}$$

We notice that as $r \to 0$, the denominator $r^3 \to 0$, so the entire expression approaches infinity. However, the numerator $1 - 3\cos^2\theta$ is not sign-definite for all directions. For example:

- Along the x-axis ($\theta = 0$):
$$\cos^2\theta = 1 \quad \Rightarrow \quad 1 - 3 = -2$$
- Along the y-axis ($\theta = \frac{\pi}{2}$):
$$\cos^2\theta = 0 \quad \Rightarrow \quad 1 - 0 = 1$$

So, depending on the direction of approach, the function diverges to $\pm\infty$ or stays 0 (if $1 - 3cos^2\theta = 0$), which means that the limit **does not exist**.

However, despite the fact that the limit

$$\lim_{(x,y)\to(0,0)} \left(\frac{1}{(x^2+y^2)^{3/2}} - \frac{3x^2}{(x^2+y^2)^{5/2}} \right)$$

does not exist (and in fact diverges), this does *not* imply that the function is not Lipschitz continuous away from the origin.

Indeed, define the domain

$$\Omega_\epsilon = \left\{ (t,x,y,v_x,v_y) \in [0,T] \times \mathbb{R}^4 : x^2 + y^2 \geq \epsilon \right\} \qquad (*)$$

for some fixed $\varepsilon > 0$. Then on this punctured domain, the function

$$J_{f,3,1}(x,y) = \frac{1}{(x^2+y^2)^{3/2}} - \frac{3x^2}{(x^2+y^2)^{5/2}}$$

is smooth (infinitely differentiable), and bounded on any compact subset of Ω_ε. The same can be said about other non-zero terms of the Jacobian.

Therefore, we state that the function $f(u)$ is **locally Lipschitz continuous away from the origin**, as it is differentiable and the derivative itself is bounded, but not globally Lipschitz continuous on any set containing the origin. And it is natural, as we cannot start the movement of a planet inside of a star it is orbiting.

Conclusion

The analysis shows that the function $f(u)$, which defines the vector field in the Kepler problem, fails to be Lipschitz continuous at the origin $(x, y) = (0, 0)$. Specifically, certain components of the Jacobian diverge as $(x, y) \to (0, 0)$, which implies that f is not Lipschitz continuous on any domain containing the origin.

Therefore, the conditions of the **global** Picard–Lindelöf theorem (Theorem 2.1) are *not* satisfied, and we cannot guarantee global existence and uniqueness of solutions on the entire domain $[0, T] \times \mathbb{R}^4$.

However, by restricting the domain to avoid the singularity at the origin, i.e., considering a punctured domain (see *) we observe that $f(u)$ is smooth and hence **locally** Lipschitz continuous (Theorem 2.2) on any compact subset of this region.

Summary: The **global** Picard–Lindelöf theorem does *not* apply to the Kepler problem due to the singularity at the origin. However, the **local** version of the Picard–Lindelöf theorem *does* apply for any initial condition with $(x, y) \neq (0, 0)$, guaranteeing the existence and uniqueness of a local solution to the Kepler problem. This reflects the physical intuition that a planet cannot be initialized at the exact center of the gravitational source it orbits.

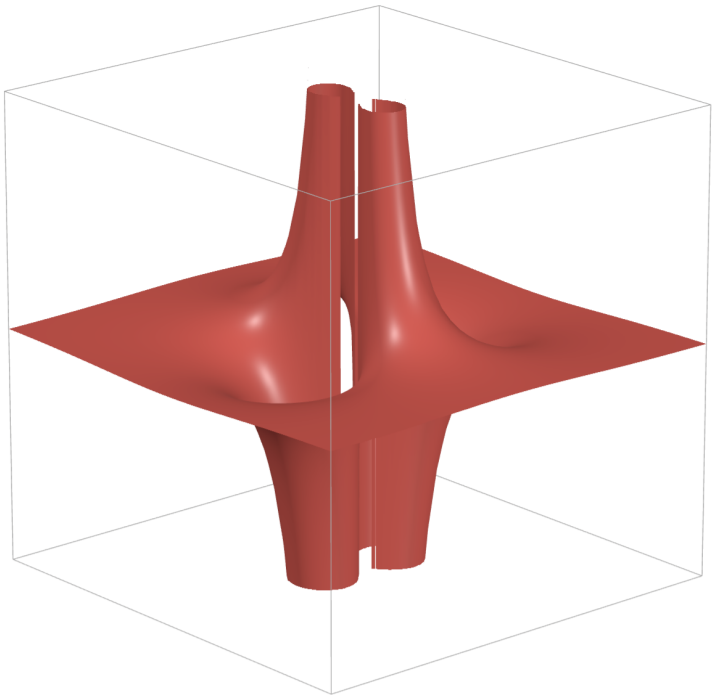


Figure 1: function $J_{f,3,1}(x, y)$

3 Numerical Methods

In this section, we will analyze different numerical methods and their application to the Kepler problem.

3.1 Forward Euler Method

The Forward Euler method is a foundational numerical scheme for approximating solutions to ODEs. As an explicit, first-order integration technique, it computes the solution at the next time step using the derivative information evaluated at the current state, assuming local linearity over the time increment.

The formula can be expressed as follows:

$$u_{n+1} = u_n + h\, f(t_n, u_n)$$

where:

- t_n is the current time step,
- u_n is the approximation of the solution at t_n,
- t_{n+1} is the next time step,
- u_{n+1} is the approximation of the solution at t_{n+1},
- $h = t_{n+1} - t_n$ is the step size.

3.1.1 Derivation

We start with the differential equation:

$$\dot{u}(t) = f(t, u(t)), \quad u(t_0) = u_0$$

Integrating both sides from t_0 to $t_0 + h$:

$$\int_{t_0}^{t_0+h} \dot{u}(t)\, dt = u(t)\Big|_{t_0}^{t_0+h} = u(t_0 + h) - u(t_0)$$

So we have:

$$u(t_0 + h) = u(t_0) + \int_{t_0}^{t_0+h} \dot{u}(t)\, dt$$

By approximating the integral with a **left Riemann sum** we get the following.

$$u(t_0 + h) \approx u(t_0) + h\, f(t_0, u(t_0))$$

Or using standard notation:

$$u_{n+1} = u_n + h\, f(t_n, u_n)$$

3.1.2 Error Derivation

Local Error

The local error is the difference between the exact solution expanded via Taylor series around $t_0 + h$ and the numerical approximation after one step:

$$\begin{aligned} u_1 &= u(t_0) + hf(u(t_0)) \\ u(t_1) &= u(t_0 + h) \\ &= u(t_0) + h\dot{u}(t_0) + \frac{1}{2}h^2\ddot{u}(t_0) + O(h^3) \\ d_h &= u(t_0 + h) - u_1 = \frac{1}{2}h^2\ddot{u}(t_0) + O(h^3) \end{aligned}$$

Thus, the **local error** satisfies:

$$d_h = O(h^2)$$

Global Error

Before moving on to the global error we have to define the local truncation error τ_h which is linked to d_h in such a manner:

$$d_h(t_n + h, u) = h\tau_h(t_n + h, u)$$

So, we now have $\tau_h = O(h)$. Each time step we have $\tau_n = \tau_h(t_n + h, u)$

We will now begin the derivation of the global error by stating the difference between the true solution and its approximation:

$$e_n = u(t_n) - u_n$$

where: $u(t_n)$ is the true solution evaluated at t_n and u_n is the approximation.

We can now expand e_{n+1} by approximating the true solution with Taylor series and accounting for their truncation with τ_n:

$$\begin{aligned} e_{n+1} &= u(t_{n+1}) - u_{n+1} \\ &= [u(t_n) + hf(t_n, u(t_n)) + h\tau_n] - [u_n + hf(t_n, u_n)] \\ &= e_n + h(f(t_n, u(t_n)) - f(t_n, u_n)) + h\tau_n \end{aligned}$$

We know that f is Lipschitz continious with a constant L, so the following holds:

$$\begin{aligned} \|e_{n+1}\| &\leq \|e_n\| + hL\|e_n\| + h\|\tau_n\| \\ &= \|e_n\|(1 + hL) + h\|\tau_n\| \end{aligned}$$

By applying recursion, we derive the following expression:

$$\|e_n\| \leq (1 + hL)^n\|e_0\| + h\sum_{j=0}^{n-1}(1 + hL)^{n-1-j}\|\tau_j\|$$

We know that $\|e_0\| = 0$ as both the approximation and true solution begin from the same initial condition. As for the local truncation error, such an inequality is satisfied: $\|\tau_n\| \leq Ch$ for some constant $C > 0$ independent of the step size h and n. So the inequality can be rephrased as follows:

$$\|e_n\| \leq hCh\sum_{j=0}^{n-1}(1 + hL)^{n-1-j} = Ch^2 \cdot \sum_{k=0}^{n-1}(1 + hL)^k$$

We can also notice a geometric series in the expression, so the following can be done:

$$\sum_{k=0}^{n-1}(1+hL)^k = \frac{(1+hL)^n - 1}{hL}$$

Utilizing the Taylor expansion of the exponential function, we obtain the following:

$$1 + hL \leq e^{hL}$$

and therefore also

$$\frac{(1+hL)^n - 1}{hL} \leq \frac{e^{nhL} - 1}{hL}$$

We also know that the total time of the simulation is defined as follows: $T = nh$, so:

$$\|e_n\| = \|u(t_n) - u_n\| \leq Ch^2 \frac{e^{nhL} - 1}{hL} = Ch\frac{e^{TL} - 1}{L} = C'h$$

So:

$$\|e_n\| = O(h)$$

Hence, the Forward Euler method is **first-order accurate** globally.

3.1.3 Simulation Results

From the results (Fig.2), we can notice that a larger step size ($h = 0.05$) causes instability, leading to an incorrect outward spiral trajectory. The planet leaves the star over time. However, by choosing a smaller step size ($h = 0.0005$), we obtain a more accurate solution that closely follows the expected elliptic trajectory. It does indeed spiral out but much slower.

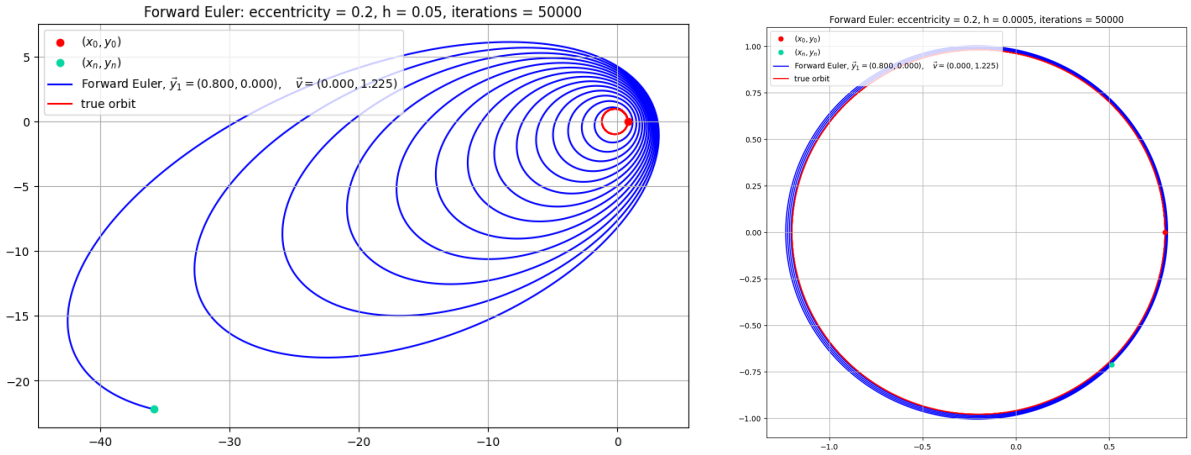


Figure 2: Two simulations with varying h values

3.2 Backward Euler Method

The Backward Euler method is a numerical technique for solving ODEs by stepping forward in time. It is an implicit, first-order method, meaning that each new value depends on solving an equation that includes the unknown future state. This characteristic often makes the method more stable, especially for stiff systems, because it evaluates the system's behavior at the upcoming time step rather than the current one.

Using the same notation as previously, the method can be expressed as follows:

$$u_{n+1} = u_n + h\, f(t_{n+1}, u_{n+1})$$

3.2.1 Derivation

We start with an ordinary differential equation (ODE):

$$\dot{u}(t) = f(t, u(t)), \quad u(t_0) = u_0$$

To evolve the solution over time, integrate both sides from t_0 to $t_0 + h$:

$$\int_{t_0}^{t_0+h} \dot{u}(t)\, dt = \int_{t_0}^{t_0+h} f(t, u(t))\, dt$$

By the Fundamental Theorem of Calculus:

$$u(t_0 + h) - u(t_0) = \int_{t_0}^{t_0+h} f(t, u(t))\, dt$$

We can now approximate the integral using the **right Riemann sum**:

$$\int_{t_0}^{t_0+h} f(t, u(t))\, dt \approx h \cdot f(t_0 + h, u(t_0 + h))$$

Thus, the implicit update becomes:

$$u(t_0 + h) \approx u(t_0) + h \cdot f(u(t_0 + h))$$

In general notation:

$$u_{n+1} = u_n + hf(t_{n+1}, u_{n+1})$$

3.2.2 Error Analysis

The error analysis follows the same general procedure as in the Forward Euler method: we expand the exact solution in a Taylor series, compare it with the numerical scheme, and subtract to isolate the leading-order error terms. This allows us to determine the order of accuracy of the method.

Local Error

The **local error** d_h, which measures the error introduced in a single time step under the assumption that all previous values are exact, satisfies the bound:

$$d_h = O(h^2)$$

This indicates that the method is *first-order accurate in time* at the local level.

Global Error

The **global error** E_h, defined as the cumulative error over the full time interval, satisfies:

$$E_h = O(h)$$

Thus, while the local error is quadratic in the step size, the global error accumulates linearly with respect to h, reflecting the first-order nature of the overall method.

3.2.3 Simulation Results

From the results (Fig.3), we can notice that a larger step size ($h = 0.005$) causes instability, leading to an incorrect inward spiral trajectory. The planet collapses onto the star over time. However, if we choose a smaller step size ($h = 0.0005$), the method will produce a more accurate solution that closely follows the expected elliptic trajectory.

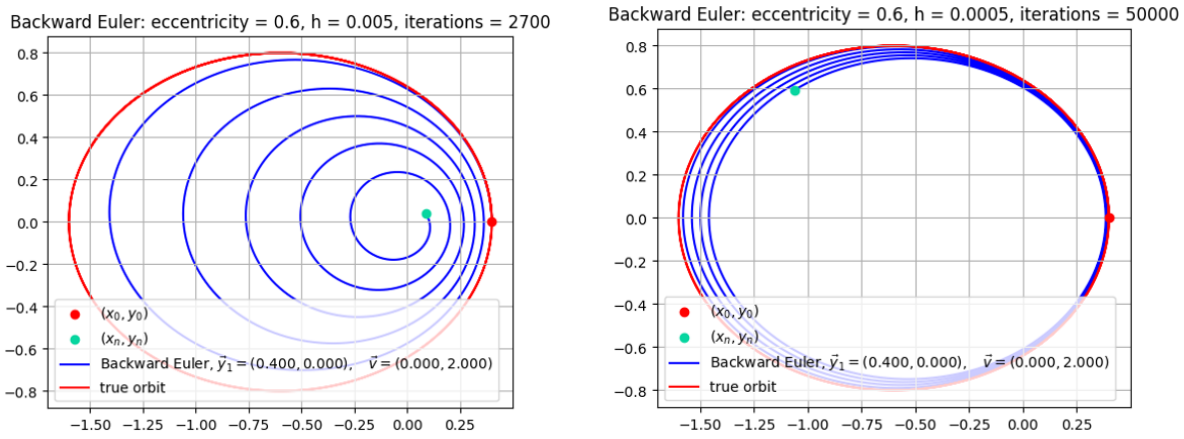


Figure 3: Two simulations with varying h values

3.3 Runge-Kutta 4 Method

3.3.1 Overview

$$\begin{cases} k_1 = f(t_n, u_n) \\ k_2 = f\left(t_n + \frac{h}{2},\ u_n + \frac{h}{2}k_1\right) \\ k_3 = f\left(t_n + \frac{h}{2},\ u_n + \frac{h}{2}k_2\right) \\ k_4 = f\left(t_n + h,\ u_n + hk_3\right) \\ u_{n+1} = u_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \end{cases}$$

The Runge-Kutta method is a powerful and versatile numerical technique for solving differential equations. It builds on the basic idea of the Euler method by taking intermediate steps to improve accuracy. This makes it especially useful for simulating complex systems, where precise solutions are needed but analytical ones are hard to obtain.

Local Error: $d_h = O(h^5)$
Global Error: $E_h = O(h^4)$

3.3.2 Simulation Results

In contrast to the Euler methods, the Runge-Kutta 4th order method (RK4) achieves improved accuracy even at moderate step sizes (Fig.4). For the same dynamical system (except for eccentricity e, which is set to 0.8 to increase the 'difficulty of convergence'), RK4 with $h = 0.01$ produces a stable and elliptical orbit, closely matching the theoretical solution. This demonstrates its superior convergence properties: RK4 resolves the non-linearities and curvature of the system with higher precision per time step, reducing the cumulative error without requiring extremely small h. The fourth-order nature ensures that local truncation error is minimized, resulting in qualitatively correct dynamics even over extended simulations. However, as an explicit integrator, RK4 tends to gradually spiral out over time (Fig.4). Despite this, it consistently exhibits superior accuracy compared to lower-order methods.

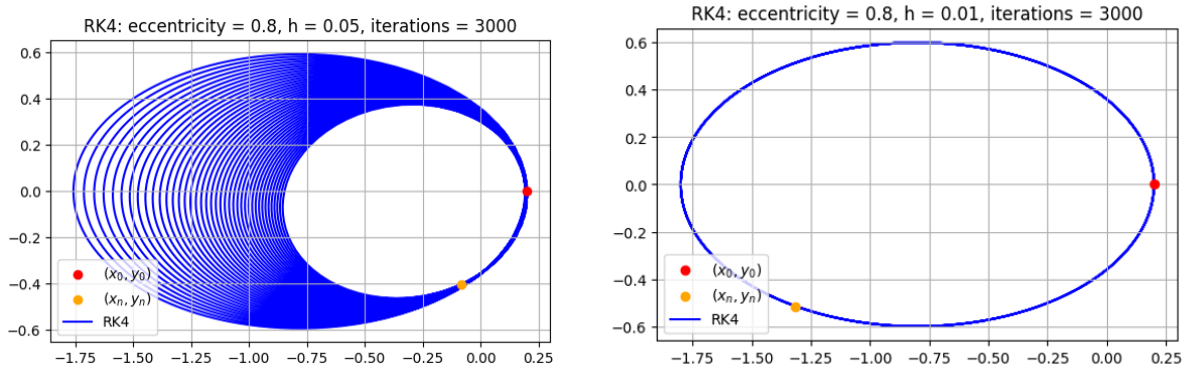


Figure 4: Two simulations with varying h values

3.4 Simulation Comparison

We can also visualize the side-by-side comparison of six solvers:

- Backward Euler Method
- Forward Euler Method
- Implicit Midpoint Method
- Störmer-Verlet Method
- RK4
- Symplectic Euler Method

and notice how the approximations converge to the expected elliptic track as the time step shrinks (h decreases from 0.005 to 0.00005). The initial condition is defined as follows:

$$u(0) = \begin{bmatrix} 1-e \\ 0 \\ 0 \\ \sqrt{\frac{1+e}{1-e}} \end{bmatrix}, \quad \text{where } e = 0.3 \text{ is the eccentricity of the orbit.}$$

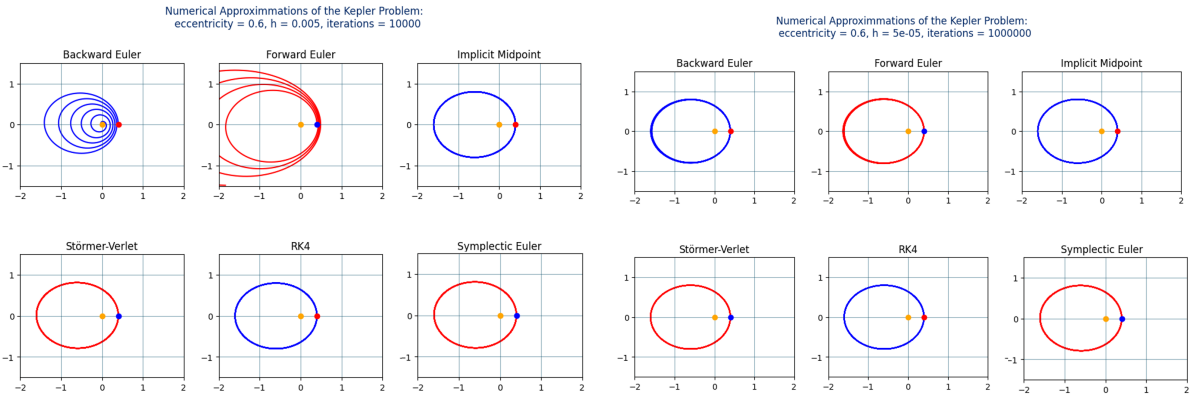


Figure 5: Two simulations with different step values

4 Conservation of Energy

4.1 Energy

By recalling the conserved quantities from classical physics, we can construct expressions for the total energy E and angular momentum L of the system:

$$E = K + P = \frac{m\|\dot{\vec{r}}\|^2}{2} - \frac{GMm}{\|\vec{r}\|} = \frac{m(v_x^2 + v_y^2)}{2} - \frac{GMm}{\sqrt{x^2 + y^2}}$$

where K is the kinetic and P is the potential energy.

$$L = \|\vec{r} \times m\dot{\vec{r}}\| = m(xv_y - yv_x)$$

To simplify the computations we set $m = M = G = 1$:

$$E = \frac{v_x^2 + v_y^2}{2} - \frac{1}{\sqrt{x^2 + y^2}} \qquad L = xv_y - yv_x$$

4.2 Noether's theorem

Theorem 4.1 (Noether's Theorem)**.** *Every continuous symmetry of the laws of physics corresponds to a conserved quantity.*

- ***Time translation symmetry*** $\longrightarrow$ ***Energy conservation***
  (Since the Kepler problem is time-independent, the total energy (kinetic + potential) remains constant.)
- ***Translation symmetry*** $\longrightarrow$ ***Momentum conservation***
  (Angular momentum conservation is a branch of momentum conservation.)
- ***Rotational symmetry*** $\longrightarrow$ ***Angular momentum conservation***
  (The problem has spherical symmetry, meaning angular momentum is conserved.)

Thus, to achieve a more accurate estimation of the trajectory, it is essential that the approximation method preserves at least one of the fundamental conserved quantities—either energy or angular momentum.

4.3 Comparison of Invariants: Loss and Gain

By taking a look at the following figures (Fig.7, Fig.8), we notice that the Implicit Euler looses energy/angular momentum as the simulation time increases, so the invariants are not conserved. Opposite can be observed in the Forward Euler method plots: it gains energy/angular momentum over time, accumulating error over time. The RK4 method looses energy/angular momentum, although extremely slowly, owing to high error order, so the accuracy of approximation is high (important: the scale is different for each plot).

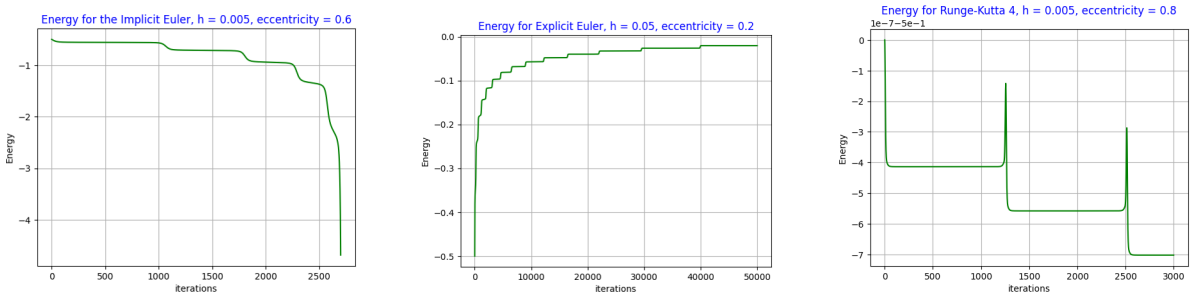


Figure 6: Energy plots for Backward Euler, Forward Euler, and RK4

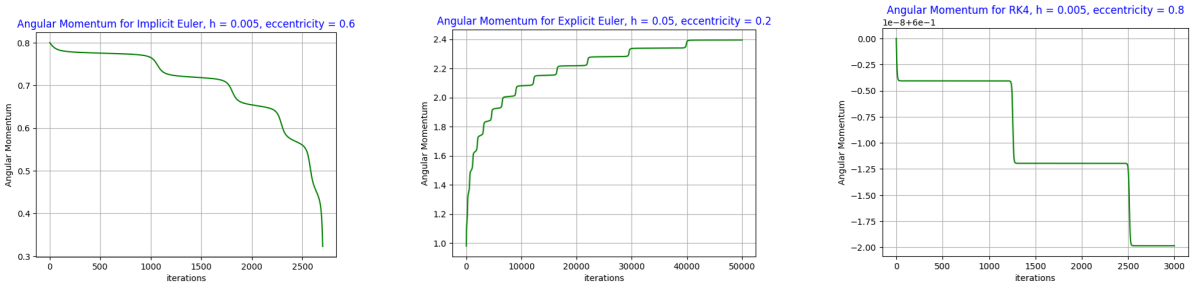


Figure 7: Angular momentum plots for Backward Euler, Forward Euler, and RK4

5 Projection Methods

In this section, we will look at possible ways to enforce the conservation of **invariants**.

5.1 Projection on Energy

We will start by defining the Lagrangian. To find the projected u_{n+1}, we would like to minimize the distance between u_{n+1} and $\tilde{u}_{n+1}$ (next step derived by the Euler method) by applying the Lagrange multipliers method and solving the following optimization problem:

$$min(\frac{1}{2}||u_{n+1} - \tilde{u}_{n+1}||^2) \quad \text{subject to} \quad E(u_{n+1}) = E(u_n)$$

The Lagrangian takes on such a form:

$$\mathcal{L}(u_{n+1}) = \frac{1}{2}||u_{n+1} - \tilde{u}_{n+1}||^2 - \lambda(E(u_{n+1}) - E(u_n))$$

where: $\frac{1}{2}||u_{n+1} - \tilde{u}_{n+1}||^2$ is the 'distance' between the two steps: projected and original, which should be minimized; $E(u_{n+1}) - E(u_n)$ is the constraint, indicating that the difference between the energy values in the current and next time steps should be equal to zero, that is, the energy remains unchanged.

Now we turn to solving the system that originates from the application of the Lagrange theorem:

$$\begin{cases} \nabla(\frac{1}{2}||u_{n+1} - \tilde{u}_{n+1}||^2) = \lambda \nabla E(u_{n+1}) \\ E(u_{n+1}) - E(u_n) = 0 \end{cases}$$

We start by computing the gradients and beneficially rearranging the system:

$$\begin{cases} \begin{bmatrix} x-\tilde{x} \\ y-\tilde{y} \\ v_x-\tilde{v}_x \\ v_y-\tilde{v}_y \end{bmatrix} = \lambda \begin{bmatrix} \frac{\partial}{\partial x}E(u_{n+1}) \\ \frac{\partial}{\partial y}E(u_{n+1}) \\ \frac{\partial}{\partial v_x}E(u_{n+1}) \\ \frac{\partial}{\partial v_y}E(u_{n+1}) \end{bmatrix} \\ E(u_{n+1})-E(u_n)=0 \end{cases} \iff \begin{cases} x-\tilde{x} = \lambda \frac{x}{(x^2+y^2)^{\frac{3}{2}}} \\ y-\tilde{y} = \lambda \frac{y}{(x^2+y^2)^{\frac{3}{2}}} \\ v_x-\tilde{v}_x = \lambda v_x \\ v_y-\tilde{v}_y = \lambda v_y \\ \frac{v_x^2+v_y^2}{2} - \frac{1}{\sqrt{x^2+y^2}} = E(u_n) \end{cases} \rightarrow$$

We change the variables on the right-hand side of the first four equations to respective constants of known from the Euler method ($\tilde{y}_{n+1}$) to approximate the non-linear system. Then we are able to obtain an equation only with one variable - λ:

$$\rightarrow \begin{cases} x = \tilde{x}\left(\frac{\lambda}{(\tilde{x}^2+\tilde{y}^2)^{\frac{3}{2}}}+1\right) \\ y = \tilde{y}\left(\frac{\lambda}{(\tilde{x}^2+\tilde{y}^2)^{\frac{3}{2}}}+1\right) \\ v_x = \tilde{v}_x(\lambda+1) \\ v_y = \tilde{v}_y(\lambda+1) \\ \frac{v_x^2+v_y^2}{2} - \frac{1}{\sqrt{x^2+y^2}} = E(u_n) \end{cases} \rightarrow \begin{cases} K(\lambda+1)^2(\lambda+r^3) - r^2 - E(u_n)(\lambda+r^3) = 0 \\ K = \frac{\tilde{v}_x^2+\tilde{v}_y^2}{2} \\ r = \sqrt{\tilde{x}^2+\tilde{y}^2} \end{cases}$$

After solving the polynomial (for instance, using fsolve()) we would be able to obtain the vector u_{n+1} projected on the Energy manifold.

5.2 Projection on Angular Momentum

The optimization problem of the form:

$$min(\frac{1}{2}||u_{n+1}-\tilde{u}_{n+1}||^2) \quad \text{subject to} \quad L(u_{n+1}) = L(u_n)$$

must be solved. We shall follow the same strategy as previously shown. However, this time we want to conserve the angular momentum. The system takes on such a form:

$$\begin{cases} \nabla\left(\frac{1}{2}\|u_{n+1}-\tilde{u}_{n+1}\|^2\right) = \lambda \nabla L(u_{n+1}) \\ L(u_{n+1}) - L(u_n) = 0 \end{cases}$$

We then precede by computing the gradients and swapping the unknowns, highlighted in blue with the known values, computed by our chosen numerical method ($\tilde{u}_{n+1}$). Along the way we also try to restructure the system in such a way, that x, y, v_x, v_y will be on the left hand

side.

$$\begin{cases} \begin{bmatrix} x-\tilde{x} \\ y-\tilde{y} \\ v_x-\tilde{v}_x \\ v_y-\tilde{v}_y \end{bmatrix} = \lambda \begin{bmatrix} \frac{\partial}{\partial x}L(u_{n+1}) \\ \frac{\partial}{\partial y}L(u_{n+1}) \\ \frac{\partial}{\partial v_x}L(u_{n+1}) \\ \frac{\partial}{\partial v_y}L(u_{n+1}) \end{bmatrix} \\ L(u_{n+1})-L(u_n)=0 \end{cases} \iff \begin{cases} x-\tilde{x}=\lambda v_y \\ y-\tilde{y}=\lambda v_x \\ v_x-\tilde{v}_x=\lambda y \\ v_y-\tilde{v}_y=\lambda x \\ xv_y-yv_x=L(u_n) \end{cases} \rightarrow \begin{cases} x=\tilde{x}+\lambda\tilde{v}_y \\ y=\tilde{y}-\lambda\tilde{v}_x \\ v_x=\tilde{v}_x-\lambda\tilde{y} \\ v_y=\tilde{v}_y+\lambda\tilde{x} \\ xv_y-yv_x=L(u_n) \end{cases}$$

After these operations, we once again conclude with a polynomial:

$$(\lambda\tilde{v}_y+\tilde{x})(\lambda\tilde{x}+\tilde{v}_y)-(\tilde{y}-\lambda\tilde{v}_x)(\tilde{v}_x-\lambda\tilde{y})-L(u_n)=0$$

We then find the roots numerically and substitute λ back into the system to find the unknown projected step u_{n+1}.

5.3 Projection on Energy and Angular Momentum

We would like to solve the following optimization problem:

$$min(\frac{1}{2}||u_{n+1}-\tilde{u}_{n+1}||^2) \quad \text{subject to} \quad E(u_{n+1})=E(u_n) \quad \text{and} \quad L(u_{n+1})=L(u_n)$$

The procedure here is similar to the previous two cases. However, this time we will have two multiplies that have to be considered, as we have two constraints: conservation of energy and angular momentum. Let us construct the Lagrangian at first. We will use the same architecture as previously:

$$\mathcal{L}(u_{n+1})=\frac{1}{2}\|u_{n+1}-\tilde{u}_{n+1}\|^2-\mu\left(E(u_{n+1})-E(u_n)\right)-\lambda\left(L(u_{n+1})-L(u_n)\right)$$

Now we apply Lagrange multipliers theorem and obtain the system:

$$\begin{cases} \vec{\nabla}\mathcal{L}(u_{n+1})=0 \\ \frac{\partial\mathcal{L}(u_{n+1})}{\partial\lambda}=0 \\ \frac{\partial\mathcal{L}(u_{n+1})}{\partial\mu}=0 \end{cases} \iff \begin{cases} \nabla\left(\frac{1}{2}\|u_{n+1}-\tilde{u}_{n+1}\|^2\right)=\mu\nabla E(u_{n+1})+\lambda\nabla L(u_{n+1}) \\ L(u_{n+1})-L(u_n)=0 \\ E(u_{n+1})-E(u_n)=0 \end{cases} \rightarrow$$

We follow the same steps as before:

$$\rightarrow \begin{cases} \begin{bmatrix} x-\tilde{x} \\ y-\tilde{y} \\ v_x-\tilde{v}_x \\ v_y-\tilde{v}_y \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x}\left(\lambda L(u_{n+1})+\mu E(u_{n+1})\right) \\ \frac{\partial}{\partial y}\left(\lambda L(u_{n+1})+\mu E(u_{n+1})\right) \\ \frac{\partial}{\partial v_x}\left(\lambda L(u_{n+1})+\mu E(u_{n+1})\right) \\ \frac{\partial}{\partial v_y}\left(\lambda L(u_{n+1})+\mu E(u_{n+1})\right) \end{bmatrix} \\ L(u_{n+1})-L(u_n)=0 \\ E(u_{n+1})-E(u_n)=0 \end{cases} \rightarrow \begin{cases} x-\tilde{x}=\lambda\tilde{v}_y+\mu\frac{\tilde{x}}{(\tilde{x}^2+\tilde{y}^2)^{\frac{3}{2}}} \\ y-\tilde{y}=-\lambda\tilde{v}_x+\mu\frac{\tilde{y}}{(\tilde{x}^2+\tilde{y}^2)^{\frac{3}{2}}} \\ v_x-\tilde{v}_x=-\lambda\tilde{y}+\mu\tilde{v}_x \\ v_y-\tilde{v}_y=\lambda\tilde{x}+\mu\tilde{v}_y \\ \frac{v_x^2+v_y^2}{2}-\frac{1}{\sqrt{x^2+y^2}}=E(u_n) \\ xv_y-yv_x=L(u_n) \end{cases}$$

As we have two multipliers (μ and λ) we will have to solve such a system numerically:

$$\begin{cases} \dfrac{v_x^2 + v_y^2}{2} - \dfrac{1}{\sqrt{x^2+y^2}} = E(u_n) \\ xv_y - yv_x = L(u_n) \end{cases}$$

after substitution:

$$\begin{cases} (\lambda\tilde{v}_y + \tilde{x}(\mu\dfrac{1}{(\tilde{y}^2+\tilde{y}^2)^{\frac{3}{2}}} + 1))(\lambda\tilde{x} + \tilde{v}_y(\mu+1)) - (-\lambda\tilde{v}_x + \tilde{y}(\mu\dfrac{1}{(\tilde{x}^2+\tilde{y}^2)^{\frac{3}{2}}} + 1))(-\lambda\tilde{y} + \tilde{v}_x(\mu+1)) = L(u_n) \\ \dfrac{(-\lambda\tilde{v}_x + \tilde{y}(\mu+1))^2 + (\lambda\tilde{x} + \tilde{v}_y(\mu+1))^2}{2} - \dfrac{1}{\sqrt{(\lambda\tilde{v}_y + \tilde{x}(\mu\dfrac{1}{(\tilde{y}^2+\tilde{y}^2)^{\frac{3}{2}}} + 1))^2 + (-\lambda\tilde{v}_x + \tilde{y}(\mu\dfrac{1}{(\tilde{x}^2+\tilde{y}^2)^{\frac{3}{2}}} + 1))^2}} = E(u_n) \end{cases}$$

After finding the unknown μ and λ we finally obtain the projected next step.

Visualizing Projections

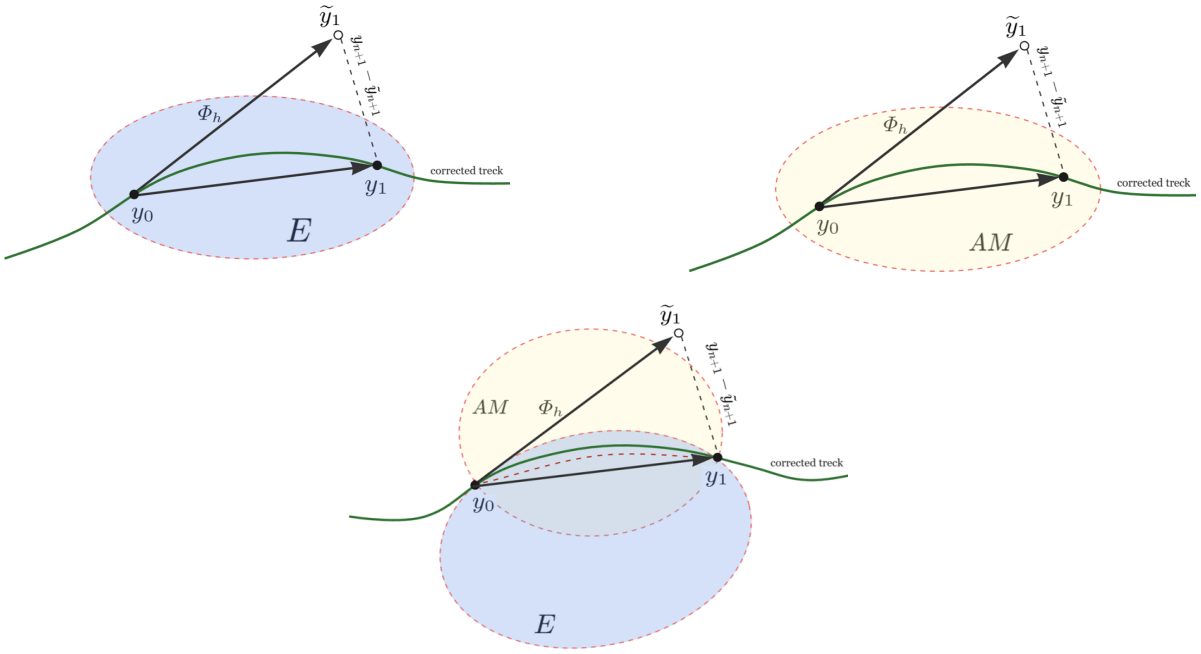


Figure 8: Visualizations for all three projections

5.4 Simulation Results: Forward Euler Method

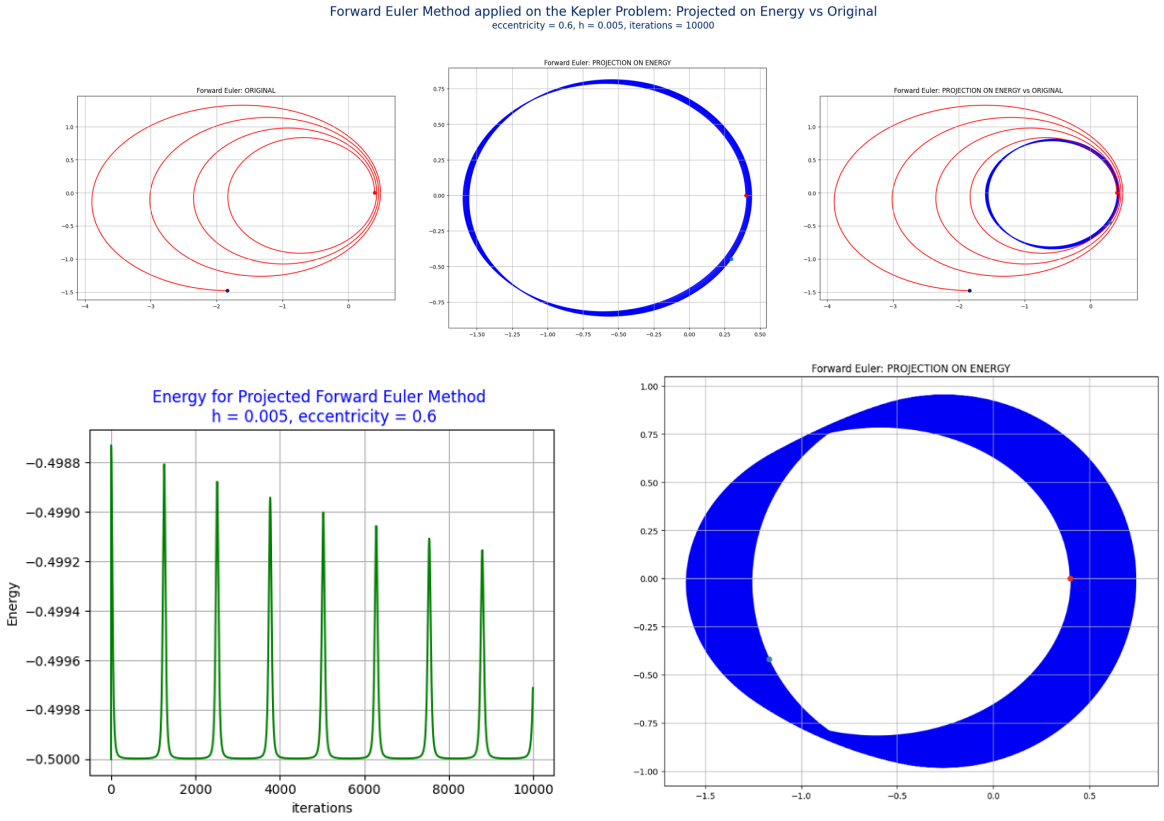


Figure 9: Comparison: Before/After projection; Energy plot; Behavior of the method when the simulation time was significantly increased

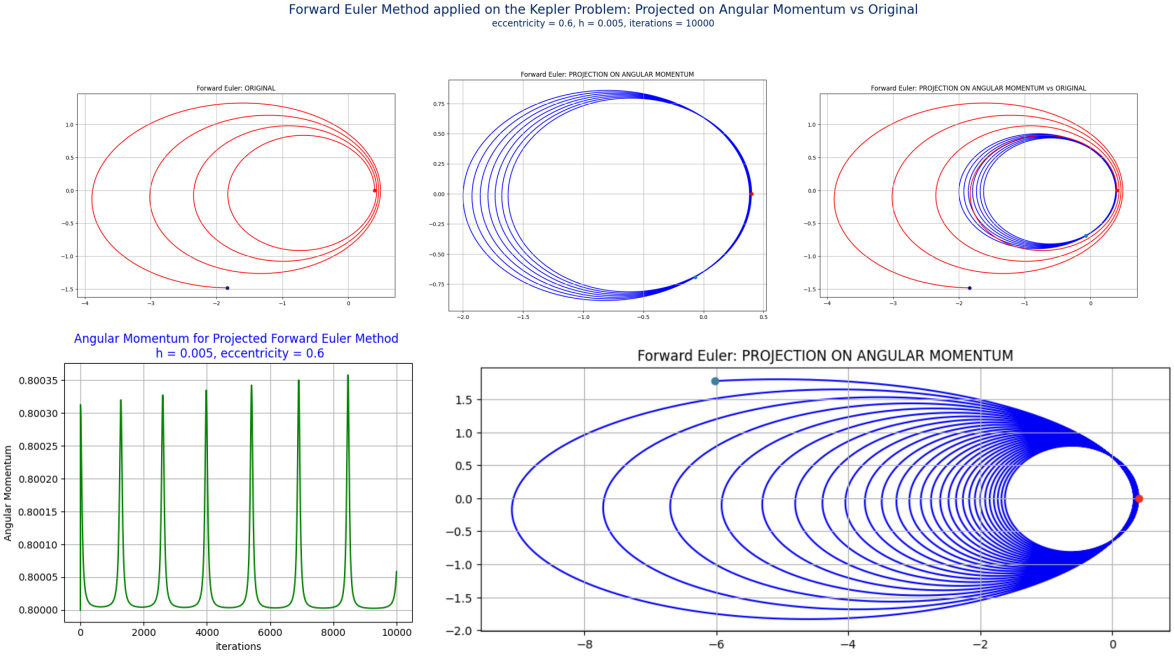


Figure 10: Comparison: Before/After projection; Angular momentum plot; Behavior of the method when the simulation time was significantly increased

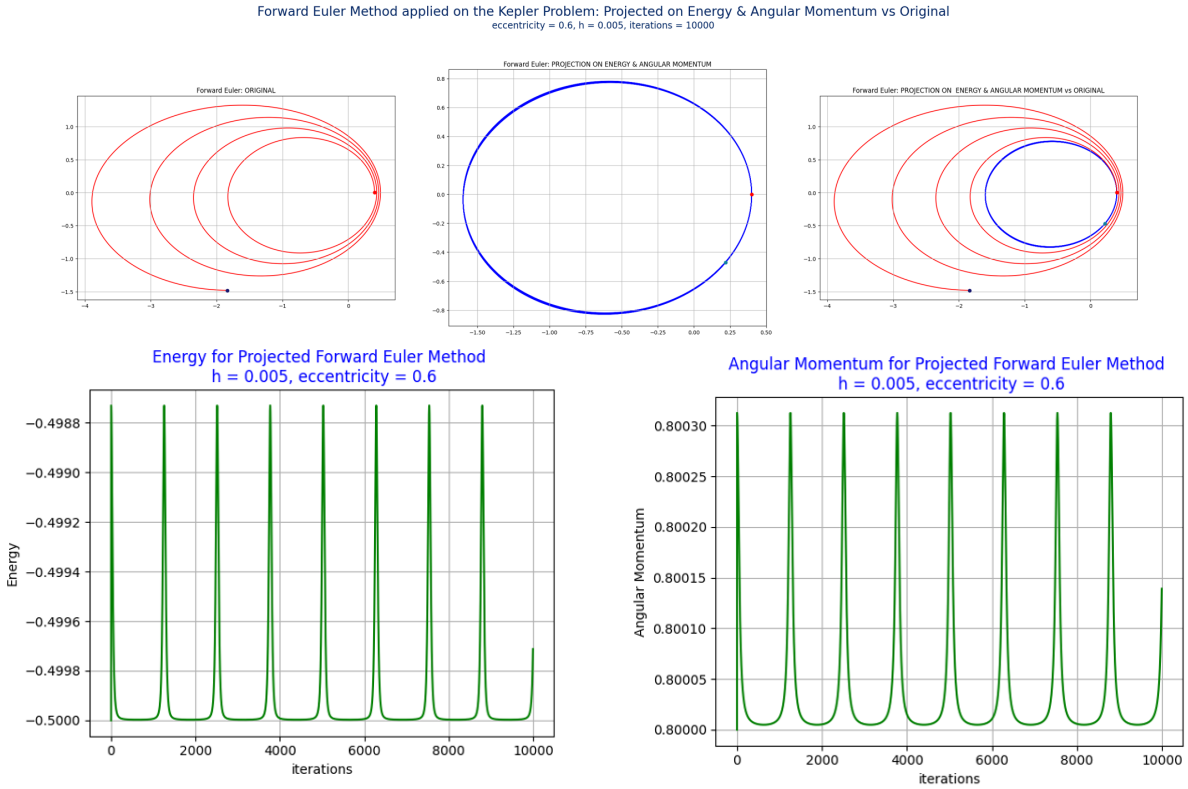


Figure 11: Comparison: Before/After projection; Angular momentum plot; Energy momentum plot

5.5 Simulation Results: Backward Euler Method

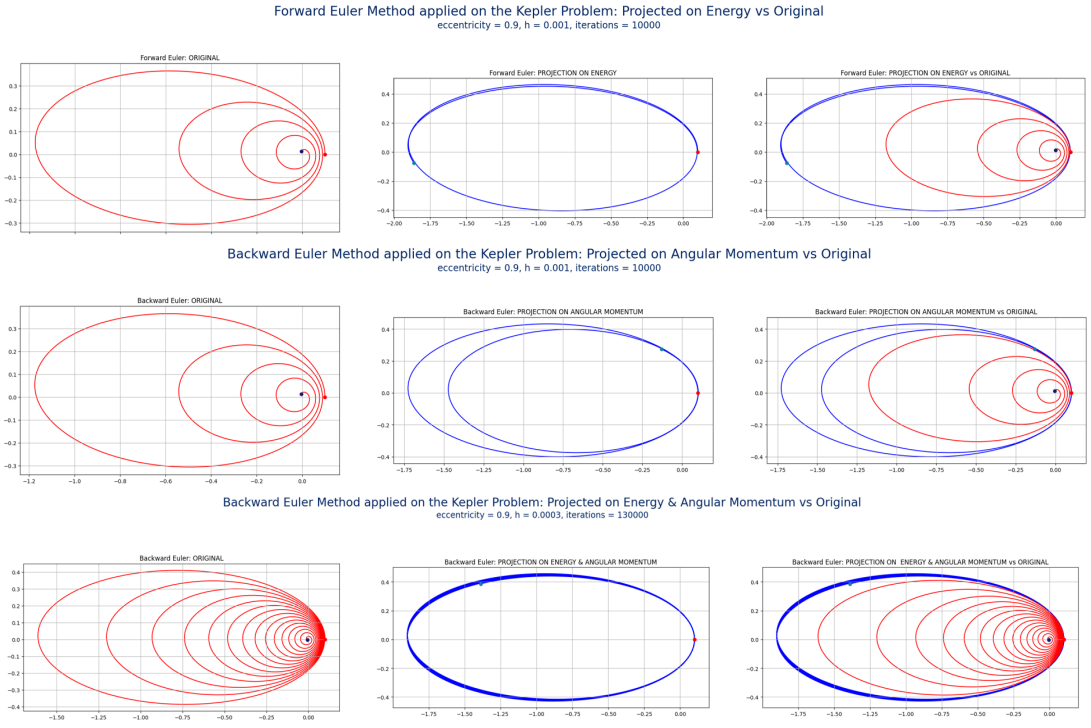


Figure 12: Comparison projection: Energy, Angular momentum, Both

6 Conclusion

To conclude, we have seen that while standard numerical integration methods, such as Forward and Backward Euler schemes, Runge-Kutta methods, are effective for many problems, they often fail to preserve conserved quantities such as energy or angular momentum when applied over long time intervals. This gradual degradation, known as numerical drift, can significantly distort the system's behavior, leading to unrealistic results such as spiraling orbits, artificial energy loss, or gain in conservative systems, and other potential issues.

Projection methods provide a reasonable solution by systematically correcting each numerical step to ensure adherence to the system's invariants. By projecting the approximate solution back onto a manifold defined by conserved quantities, these methods combat drift without requiring extremely small time steps or excessive computational power. In the context of the Kepler problem, this means maintaining **stable**, **closed** orbital paths over **long** durations. This is essential for accurate planetary simulations, spacecraft trajectory planning, and celestial mechanics research.

Ultimately, projection techniques elegantly bridge the gap between physical accuracy and computational practicality. They preserve the system's underlying geometry and conservation laws while delivering results that remain accurate over long-term simulations, making them a component of paramount importance in the modern numerical toolbox for dynamical systems.

7 References

Books

- Hairer, E., Lubich, C., Wanner, G. (2006). *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations* (2nd ed.). Springer.
- Hanke-Bourgeois, M. (2015). *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens.* Springer Vieweg.
- Székelyhidi, L. (2011). *Ordinary and Partial Differential Equations for the Beginner.* World Scientific.

Web Resources

- Wikipedia: Lagrange multipliers (accessed 12 April 2025)
- Wikipedia: The Kepler problem (accessed 25 April 2025)
- Wikipedia: Kepler orbit (accessed 8 May 2025)
- Wikipedia: Gravitational potential (accessed 2 April 2025)
- Wikipedia: Angular momentum (accessed 2 May 2025)
- Wikipedia: Noether's theorem (accessed 7 May 2025)
- Desmos 3D Plotter (accessed 21 July 2025)
- Desmos 2D Calculator (accessed 21 July 2025)