MOON ORBIT

ANTON SJÖSTRÖM

Abstract. This paper presents a detailed overview of the moon orbit project, examining how the Hamiltonian-Jacobi-Bellman (HJB) Equation and Pontryagin's Maximum Principle (PMP) can be applied to optimally change a spaceship's orbit for a given cost function. The main motivation is to test my understanding of HJB and PMP. Although this work is not intended as any type cutting-edge research, it may be useful to others learning about these concepts. Additionally, I chose to write this project in Typst to explore its capabilities.

1. Introduction

The Hamiltonian-Jacobi-Bellman (HJB) Equation and Pontryagin's Maximum Principle (PMP) are two fundamental methods for solving optimal control problems. The HJB equation is a nonlinear partial differential equation that can be computationally expensive for high-dimensional systems and often requires linearization. In contrast, PMP results in an ordinary differential equation (ODE), which is generally easier to solve and less demanding computationally. In optimal control, PMP provides an open-loop solution, while HJB yields a closed-loop solution. An open-loop solution follows a predefined action without feedback, whereas a closed-loop solution uses feedback to adjust actions based on the system's state. This distinction will become important later in the project.

2. Theory

2.1. **Orbital Mechanics.**

This problem is most easily simulated using polar coordinates. This means that:

$$\begin{aligned} x &= r\cos(\theta) \\ y &= r\sin(\theta) \end{aligned} \tag{1.}$$

The forces acting on a spaceship orbiting a moon can be described by the following system of equations, assuming GM = 1:

$$\begin{aligned} \dot{r} &= v_r \\ \dot{\theta} &= \omega \\ \dot{v}_r &= u_r - \frac{1}{r^2} + r\omega^2 \\ \dot{\omega} &= \frac{u_\theta - 2v_r w}{r} \end{aligned} \tag{2.}$$

Here, v_r is the radial velocity, ω is the angular velocity, u_r is the acceleration produced by the spaceship's engine in the radial direction, and u_θ is the acceleration in the angular direction.

To maintain a stable orbit, the following conditions must be satisfied:

$$\begin{aligned} r &= r \\ v_r &= 0 \\ \omega &= \sqrt{\frac{1}{r^3}} \end{aligned} \qquad 3.$$

To simulate these orbital dynamics, I use the Runge-Kutta 4 (RK4) algorithm. It is stable for these types of systems, unlike the standard Euler-Forward method.

2.2. **Pontryagin's Maximum Principle.**

Pontryagin's Maximum Principle (PMP) is a mathematical framework for solving optimal control problems. It provides necessary conditions for an optimal control and trajectory by introducing the concepts of the Hamiltonian and costate variables. According to PMP, for a system described by differential equations, the optimal control must maximize (or minimize) the Hamiltonian at each instant, subject to the system dynamics and boundary conditions. This principle transforms the original control problem into a boundary value problem involving both the state and costate equations, which can then be solved to find the optimal control strategy.

Let the system dynamics be given by:

$$\dot{x}(t) = f(x(t), u(t), t) \qquad 4.$$

where $x(t)$ is the state, $u(t)$ is the control, and f describes the system.

We can define a cost function:

$$J = \Psi(x(T)) + \int_0^T L(x(t), u(t)) \qquad 5.$$

where $\Psi(x(T))$ is the terminal cost and $L(x(t), u(t))$ is the running cost.

Define the Hamiltonian as:

$$H(x, u, \lambda, t) = L(x, u, t) + \lambda(t) f(x, u, t) \qquad 6.$$

where λ is the costate variable.

Lastly, we define the costate equation as:

$$\begin{aligned} -\dot{\lambda}(t) &= \frac{\partial H}{\partial x} \\ \lambda(T) &= \frac{\partial \Psi(x(T))}{\partial x} \end{aligned} \qquad 7.$$

From above, we know that the dynamics, given GM = 1, are:

$$\begin{aligned} \dot{r} &= v_r \\ \dot{\theta} &= \omega \\ \dot{v}_r &= u_r - \frac{1}{r^2} + r\omega^2 \\ \dot{\omega} &= \frac{u_\theta - 2v_r w}{r} \end{aligned} \qquad 8.$$

Let's define the cost function as:

$$J = (r(T) - r^*)^2 + (\omega(T) - \omega^*)^2 + {v_r}^2 + \int_0^T \alpha({u_r}^2 + {u_\theta}^2)\,\mathrm{d}t \qquad 9.$$

Using this, we can define the Hamiltonian:

$$H = \alpha({u_r}^2 + {u_\theta}^2) + \lambda_r v_r + \lambda_\theta \omega + \lambda_{v_r}\left(u_r - \frac{1}{r^2} + r\omega^2\right) + \lambda_\omega \frac{u_\theta - 2v_r w}{r} \qquad 10.$$

We wish to find u_r and u_θ such that this Hamiltonian is minimized, which can be done by setting the derivative of H to zero:

$$\begin{aligned} \frac{\partial H}{\partial u_r} &= 0 \longrightarrow u_r = -\frac{\lambda_{v_r}}{2\alpha} \\ \frac{\partial H}{\partial u_\theta} &= 0 \longrightarrow u_\theta = -\frac{\lambda_\omega}{2\alpha r} \end{aligned} \qquad 11.$$

The costate equation becomes:

$$\begin{aligned} \dot{\lambda}_r &= -\frac{\partial H}{\partial r} = -\lambda_{v_r}\left(\frac{2}{r^3} + \omega\right) - \lambda_\omega \frac{2v_r\omega - u_\theta}{r^2} \\ \dot{\theta} &= -\frac{\partial H}{\partial \theta} = 0 \\ \dot{\lambda}_{v_r} &= -\frac{\partial H}{\partial v_r} = \frac{2\omega\lambda_\omega}{r} - \lambda_r \\ \dot{\lambda}_\omega &= -\frac{\partial H}{\partial \omega} = \frac{2\lambda_\omega v_r}{r} - \lambda_\theta - 2\lambda_{v_r} r\omega \end{aligned} \qquad 12.$$

Now, all the necessary equations are in place to solve for an optimal transition between orbits using the fully non-linear dynamics of the system. This boundary value problem (BVP) can be solved numerically using Scipy in Python to obtain the optimal control variables u_r and u_θ.

2.3. **Hamiltonian-Jacobi-Bellman.**

If we instead want to find a closed-loop controller that keeps the spaceship on a desired orbit, we can use the HJB equation to achieve this.

The HJB equation has a similar structure to PMP, but instead of using costate variables, it relies on a different optimality condition. Consider the value function $V(t)$, which describes the remaining cost at time t. It follows the equation:

$$\frac{\partial V(x,t)}{\partial t} + \min\left(\frac{\partial V(x,t)}{\partial x} f(x,u) + L(x,u)\right) = 0 \qquad 13.$$

subject to the terminal condition:

$$V(x,T) = \Psi(x) \qquad 14.$$

In our case, we want to stay on orbit for an infinite time, so there is no terminal condition as $T \to \infty$:

$$\begin{aligned} J = \int_0^\infty & q(r(t) - r^*)^2 + q(r(t) - r^*)^2 + 0.1q(\omega(t) - \omega^*)^2 \\ & + 0.1q v_r(t)^2 + \alpha({u_r}^2 + {u_\theta}^2)\,\mathrm{d}t \end{aligned} \qquad 15.$$

We can rewrite this in matrix form by letting x be the difference from the goal state:

$$J = \int_0^\infty x^T Q x + u^T R u \,\mathrm{d}t \tag{16.}$$

The dynamics can also be rewritten in matrix form if we linearize them around the goal state:

$$f(x, u) = Ax + Bu \tag{17.}$$

By linearizing, we fix some state variables to the goal state and only keep the first-order terms as variables. Since we have an infinite horizon, we look for a stationary solution, which means:

$$\frac{\partial V}{\partial t} = 0 \tag{18.}$$

and

$$\min(\nabla V^T (Ax + Bu) + x^T Q x + u^T R u) = 0 \tag{19.}$$

We minimize this by taking the derivative with respect to u:

$$\frac{\partial}{\partial u}(\nabla V^T (Ax + Bu) + x^T Q x + u^T R u) = 0 \tag{20.}$$

This leads to:

$$u^*(x) = -\frac{1}{2} R^{-1} B^T \nabla V \tag{21.}$$

Assuming $V(x) = x^T P x$, then:

$$u^*(x) = -R^{-1} B^T P x = -Kx \tag{22.}$$

If we substitute this u into the HJB equation and perform some algebra, we obtain:

$$x^T (Q + A^T P + PA - PBR^{-1} B^T P) x = 0 \tag{23.}$$

Since this must hold for all x:

$$Q + A^T P + PA - PBR^{-1} B^T P = 0 \tag{24.}$$

This is known as the continuous-time Riccati equation and can be solved using the built-in solver in Scipy.

2.4. **Combination of HJB and PMP.**

It is also possible to construct a time-varying HJB controller by creating a running cost around the path proposed by PMP, rather than taking the stationary solution. Instead of linearizing around the final orbit, the linearization is performed at the position where the controller targets the spaceship to be at time t. The time-varying Riccati equation can be solved using Scipy's ODE solver, similar to the standard PMP approach.

3. Results

3.1. **Initial Stable Orbit.**

The first result demonstrates the stability of the RK4 algorithm, showing that numerical errors do not accumulate and cause the spaceship to leave its orbit. The resulting orbit with radius 1 is shown in Figure 1.

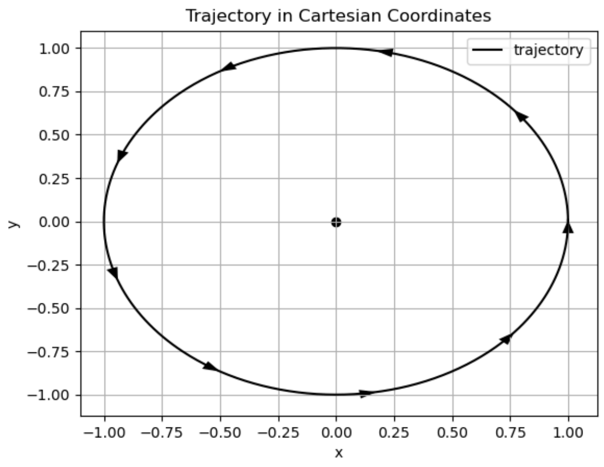

FIGURE 1. A stable simulation of the orbital dynamics.

3.2. **Perturbations.**

If we introduce perturbations to the acceleration of the spaceship, for example due to errors in the motor, the orbit becomes unstable as shown in Figure 2. From the previous result, we know that this instability is caused by the perturbations and not by numerical errors.

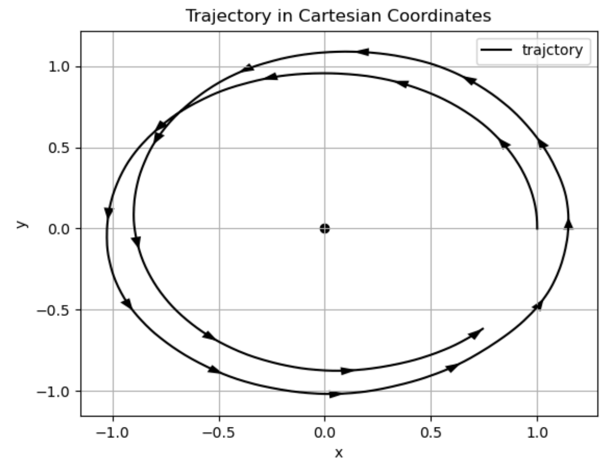

FIGURE 2. An unstable simulation of the orbital dynamics due to perturbations.

To maintain control of the orbit even in the presence of perturbations, we can use the controller defined in Equation 22.

When applying the controller from Equation 22, the resulting orbit, shown in Figure 3, is nearly indistinguishable from the stable orbit in Figure 1.

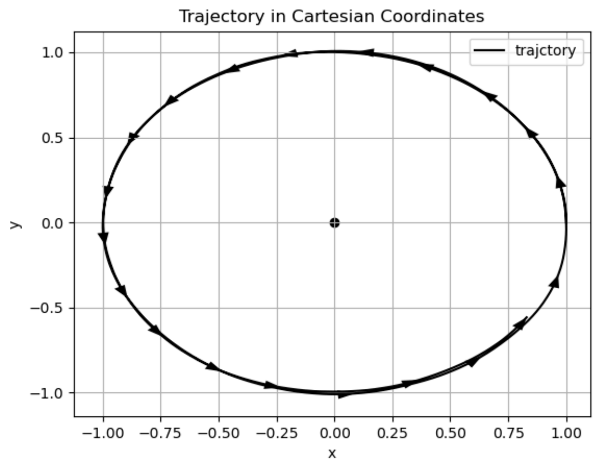

FIGURE 3. A controlled simulation using HJB to compensate for perturbations.

The controller can be visualized by examining the vector field generated by the matrix K. The arrows indicate the direction and magnitude of the control vector u^*. This is illustrated in Figure 4.

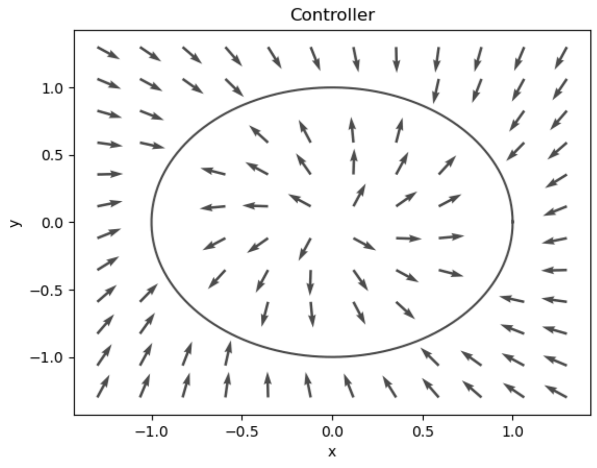

FIGURE 4. The vector field generated by the matrix K, showing the control directions.

3.3. **Change in Orbit.**

Instead of only maintaining the spaceship's orbit, it is also desirable to change the orbit. This can be achieved using the HJB controller by setting the reference orbit to a new value. If the new orbit is close to the current one, the linearization remains a good approximation of the true nonlinear dynamics. The change in orbit is shown in Figure 5.

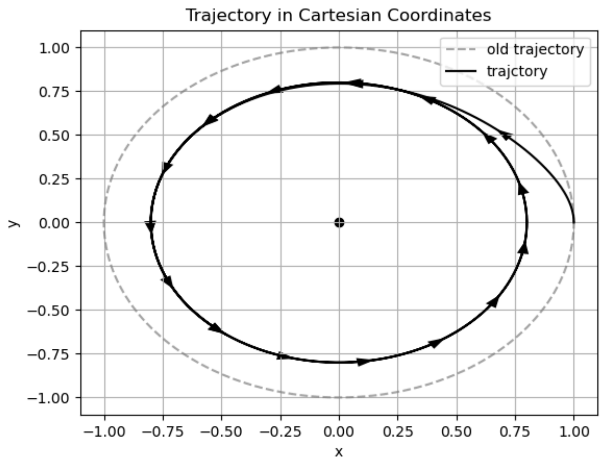

FIGURE 5. The black line shows the new trajectory of the spaceship, while the gray line indicates the original orbit.

If the desired orbit is too far from the original, the linearization becomes inaccurate and the resulting path is not optimal. This behavior is illustrated in Figure 6. Ideally, there would be a way to achieve this without relying on linearization of the true dynamics.

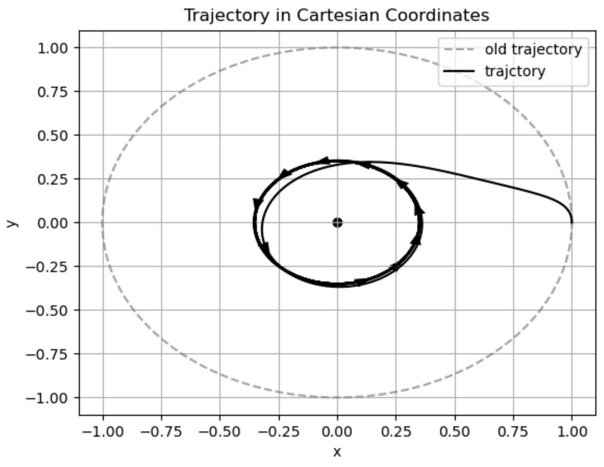

FIGURE 6. The black line shows the new trajectory of the spaceship, while the gray line indicates the original orbit.

3.4. **Changing orbit using PMP.**

To achieve a larger change in orbit without linearizing the dynamics, we can use the PMP method. When applying the same orbit change as in Figure 6 but using PMP, the transition is much smoother, as shown in Figure 7.

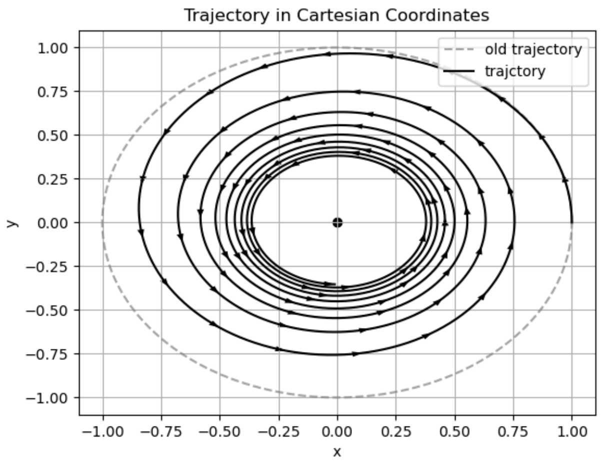

FIGURE 7. The black line shows the trajectory of the spaceship, while the gray line indicates the original orbit.

However, as discussed earlier, PMP provides an open-loop solution, making it sensitive to perturbations. If we introduce the same perturbations as in Figure 2 during the transition, the resulting path, shown in Figure 8, is less smooth and deviates from the optimal trajectory.

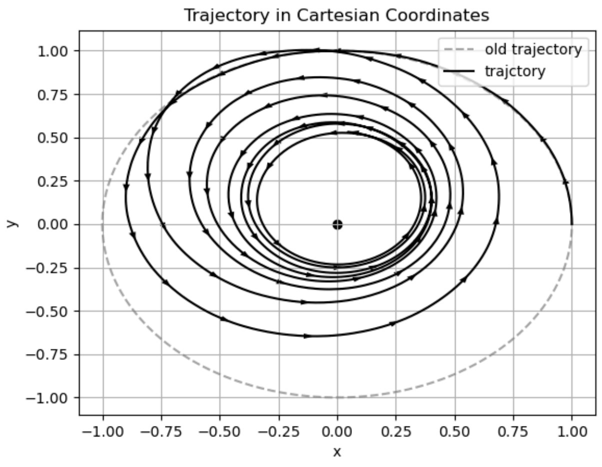

FIGURE 8. The black line shows the perturbed trajectory of the spaceship, while the gray line indicates the original orbit.

3.5. **Combination of HJB and PMP.**

To achieve a smooth transition even when the system is perturbed, it is possible to combine PMP and HJB by solving for a time-varying controller. The dynamics are still linearized, but the linearization matrix A becomes time-dependent, $A(t)$, so that the linearization is performed around the point where PMP predicts the spaceship should be at time t. The entire HJB equation is solved in a similar way to PMP, resulting in a controller $K(t)$ that does not need to be recomputed once the trajectory begins. The resulting path is shown in Figure 9, which is as close as can be expected when perturbations occur during the transition.

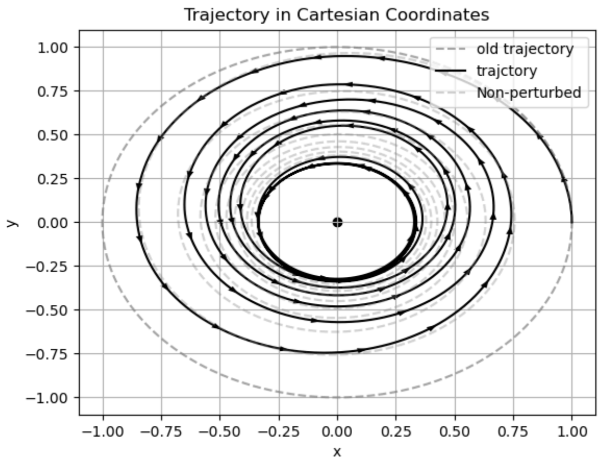

FIGURE 9. The black line shows the perturbed path of the spaceship, while the gray line shows the unperturbed PMP path.

To understand the trajectory when following the controller without perturbations, see Figure 10. This path is nearly identical to the one taken in Figure 7.

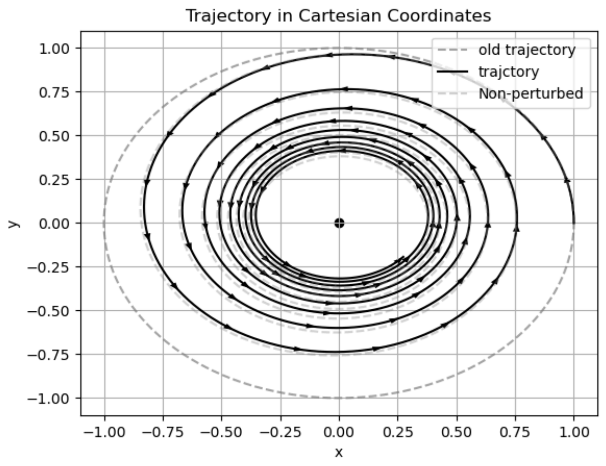

Figure 10. The black line shows the unperturbed time-varying HJB path of the spaceship, while the gray line shows the unperturbed PMP path.

4. Discussion

This project has been very helpful for solidify my knowledge about HJB and PMP in practice, especially how to solve these equations numerically, which is necessary for most interesting problems. My main motivation was to test my understanding of these methods and to implement them for a fun problem, rather than to present any type of cutting-edge research. The project represents my prefered way of working, deriving the problem formulation analytically and then solving it numerically.

5. End Notes

The main code for this project can be found in the file `MoonOrbit.py`, and the notebook used to generate the figures is included as `MoonOrbitNotebook`. I aimed to make the derivations in the methods section as clear as possible, though some steps may be less detailed than others.