Stabilizing an Inverted Pendulum Using PID Control

Ezra Santos

Introduction

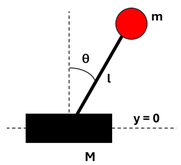
The stabilization of the inverted pendulum on a cart is a common task in introductory control theory. Due to its simplicity, most algorithms use the inverted pendulum as a benchmark for control testing.

PID (Proportional, Integral, Derivative) control is a basic, yet effective way to stabilize a system. The error of the system is recorded over time, and the input inserted into the system depends on three factors: the current error at time t, the integral of the error in the interval $[0, t]$, and the time derivative of the error at time t. Each factor has its coefficient, which is called the gain, and they are tuned for the system to reach stability.

In this project, we will be using matplotlib to display a simulation of the inverted pendulum. For modeling the motion of the pendulum, we will be using SciPy's IVP solver, with equations derived from Lagrangian mechanics. Finally, we will create and manually fine-tune a PID control function to stabilize the system.

Modeling and Visualization

We will assume that the rod connecting the bob and the cart is massless with length l, as well as the cart being restricted mainly on the x-axis. Let M be the mass of the cart and m be the mass of the bob. Using Lagrangian mechanics, we have the following equations of motion:



$$(M+m)\ddot{x} - ml\ddot{\theta}cos(\theta) + ml\dot{\theta}^2 sin(\theta) = F(t)$$

$$l\ddot{\theta} - \ddot{x}cos(\theta) - gsin(\theta) = 0$$

where θ is the angle from the vertical, and x is the position of the center of the cart.

To use SciPy's IVP solver, we need to feed a state vector $[\dot{x}, \dot{\theta}, \ddot{x}, \ddot{\theta}]$. Thus, solving for $\ddot{x}$ and $\ddot{\theta}$, we have

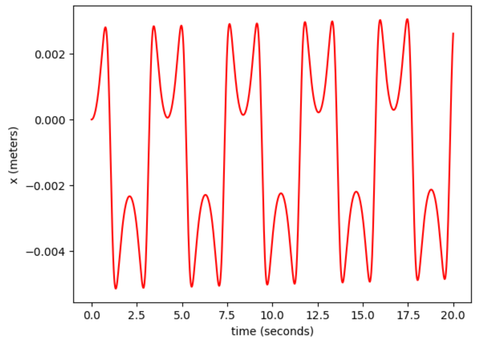
$$\ddot{\theta} = \frac{(M+m)gsin\theta + ml\dot{\theta}^2 sin\theta cos\theta - F(t)cos\theta}{l(M+m+mcos^2\theta)}$$

$$\ddot{x} = \frac{gsin\theta - l\ddot{\theta}}{cos\theta}$$

For animation, we use matplotlib's FuncAnimation library with a function that updates the positions of the bob, rod, and cart every frame [2].

As a simple example, suppose that $F(t) = 0, M = 50, m = 0.2, l = 1$. Additionally, assume that $x_0 = 0$ and $\theta_0 = 0.3$. This reduces to a simple pendulum. However, since M is not infinite, x slightly deviates from the origin.

Modeling and Visualization (cont'd)



Before trying to implement PID control, the following modifications were made to the "update frame" function to take into account a variable force $F(t)$.

1. The function allows the force to be changed during each frame.
2. The function will continuously solve the ODEs for each frame, updating the IVP solutions over time.
3. Instead of updating frames based on one IVP solution, the next frame is based on the updated IVP solution.

PID Implementation and Fine-Tuning

Let $\gamma(t) = \pi - \theta(t)$ be the error of the system at time t. Then the force exerted by the PID controller at time T is

$$F(T) = K_p\gamma(T) + K_i\int_0^T \gamma(t)dt + K_d\gamma'(T)$$

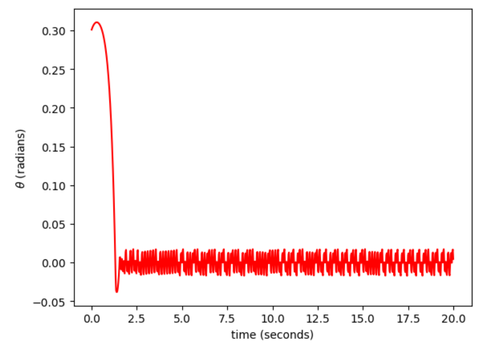
where K_p, K_i, K_d are the gains of the system. Stabilization is achieved when the gains achieve the correct value via manual fine-tuning [1].

Fine-Tuning Steps:

1. Start with no integral and derivative gains and a small proportional gain (here, we start with $[K_p, K_i, K_d] = [10, 0, 0]$).
2. Slowly increase K_p until the system exhibits somewhat steady oscillations (here, we increase K_p by five for each step).
3. Slowly increase K_d to achieve faster damping.
4. (Optional) Very slightly increase K_i for potential steady-state errors.

PID Implementation and Fine-Tuning (cont'd)

After tuning the gains to $[K_p, K_i, K_d] = [60, 0, 8]$, we get the following plot of theta over time:



Limitations and Future Improvements

While the goal of stabilizing the inverted pendulum is complete, it's worth emphasizing that this is only one example. After playing around with different values and a lot of trial and error, here is a list of limitations and improvements that can be addressed in the future:

- **Manual Tuning:** It's time-consuming to manually tune the gains, especially if the system itself is sensitive to initial conditions. In theory, it's possible to program a self-tuning mechanism that increases the gains until the system is stabilized. A method of interest is Ziegler-Nichols [1].
- **Optimization:** Can we punish the system for moving too far from the origin? Similarly, can we minimize the force exerted towards the cart? Can we take advantage of the fact that the pendulum can go below $y = 0$ by aggressively moving the cart to swing the pendulum upwards?
- **Methodology:** Other than PID control, there are other methods of solving the inverted pendulum, most of which are particularly interesting: Fuzzy Control, Neural Networks, Evolutionary Algorithms, etc.

References

[1] Lim et al. Stabilising an inverted pendulum with pid controller. *MATEC Web of Conferences*, 152, 2018. doi:https://doi.org/10.1051/matecconf/201815202009.

[2] Christian Hill. *Learning Scientific Programming with Python*. Cambridge University Press, 2nd edition, 2020.

esantos04@uchicago.edu