Construction and Control of an Inverted Pendulum with Inertial Correctional Disk

ME102B Semester Project Final Report
December 8, 2011

Kevin Jenkins, Eric Lee, Tevis Miller, Sean Mitchell, Mitch Muller

Introduction

For our semester 102B project, we assembled and programmed an inverted pendulum that uses a motorized inertial disk to balance itself in the vertical position. The inertial disk strategy differs from the more classical ‘pendulum on a cart’ inverted pendulum since it applies a correctional torque at the top (free end) of the pendulum instead of at the base. The rest of this report documents the basic calculations, strategies, and implementation of our design.

Photograph

Physics

In order to keep the pendulum upright, the inertial disk must exert a torque on the arm that can counteract the torque due to gravity. This torque originates from the angular acceleration of the disk, and can be quantified using the disk’s change in angular momentum:
When calculating the amount of torque necessary, it is important to also consider the inertial torque of the entire pendulum as it falls, as well as torque due to friction. Ultimately, the governing equation of the physical system can be stated as:

\[ + = + \]

**Materials and Design**

**Bill of Materials:**
- Custom 18” Acrylic disk
- Steel weights
- 24 Volt Motor
- Arduino *Duemilanove* Board
- Potentiometer for position sensing
- ¾” aluminum square tube
- High current motor driver

Many features of the final pendulum design resulted from the careful consideration of tradeoffs in control effort, actuator size, and robustness of the system. The following is a list of variables we considered and their effect on the system. The catch angle is the widest initial angle (measured from vertical) from which the system can recover. We took this to be the angle at which maximum motor torque equaled torque due to gravity.
• **Pendulum arm length** – longer arm reduces effect of noise but reduces catch angle by increasing gravitational moment.

• **Motor** – We needed a light motor capable of high torque at decent speed without costing too much.

• **Wheel M.O.I** – higher M.O.I means motor can apply high torque for longer without harmful spin-up, but radius must be shorter than pendulum arm.

• **Wheel mass** – weights added on the wheel perimeter increase moment of inertia but make the system heavier, reducing catch angle.

Due to limited capital, our motor was the limiting factor we needed to design around. Based on our motor’s torque rating, its mass, and the catch angle we wanted (10°), we applied equations to determine appropriate Pendulum arm length as well as wheel mass and radius. We chose an acrylic wheel because we wanted a light, cheap material.

**Control Scheme**

The pendulum uses a PID controller based on a single angular displacement reading calculated from a potentiometer that is attached to the base of the arm. We measured \( \theta \), the angle between the pendulum arm and vertical, using a potentiometer. \( V_{\text{ref}} \) is the voltage reading from the potentiometer when the pendulum is vertical, which changes based on several factors as described below. This leads to:

\[
\theta = V_{\text{ref}} - \frac{\text{Position Reading}}{\text{Error}}
\]

Based on the error term and how it changes over time, our Arduino controller sends a direction bit and a PWM signal of varying duty cycle to our H-bridge. The H-bridge drives the motor in the direction and intensity indicated. The process can be seen in more detail in our control diagram. For simplicity on the diagram, “Position Reading” should be taken as “Error” as defined above.

An initial concern arose with our choice of PID control since we are only employing one sensor (position), and in order to implement PID control we would also have to calculate the integral and derivative of the sensor signal. While numerical integral calculation does not pose problems, derivative calculations of a discrete signal can lead to undesirable sensor noise amplification. To avoid this, we considered using an observer control strategy to simulate the higher order states of the system, but determination of the system’s exact physical parameters proved difficult and inaccurate. For this reason, we chose to use a potentiometer as the angular position sensor since it could provide a continuous analog signal, and then chose to implement PID control and test the effects of sensor noise on the system. Since the pendulum balanced dependably after some tuning of the PID gains, we chose to use PID as the final control method.

**Spin-Up Control**

When the wheel starts to spin quickly in one direction, the motor loses torque and the
system can go unstable. Spin-up usually occurs because the zero reference in our software does not match true vertical, and so the pendulum is trying to hang at some non-zero angle, which it cannot do indefinitely.

The way to stop spin-up is to adjust the reference so that software zero matches true zero. The free potentiometer allows for manual coarse adjustment, but we also wanted the controller to fine-tune this measurement automatically, by detecting spin-up and adjusting reference accordingly. We used the following equations in our software to accomplish this:

\[ V_{\text{ref}} = V_{\text{adj}} + V_{\text{spin}} = 0 \]

\( V_{\text{ref}} \) is the reference voltage defined in our control methodology. \( V_{\text{adj}} \) is the voltage reading from the manual-adjustment potentiometer. \( V_{\text{out}} \) is the output voltage we give to the motor. Within a linear region, the wheel spin-up is proportional to the integral of this voltage, and \( V_{\text{spin}} \) represents an adjustment to our zero reference used to combat spin-up. This strategy proved extremely effective in reducing spin-up, and was much easier to implement than directly measuring angular wheel velocity. The schematic below shows the entire control architecture.

**Applications**

Inverted pendulum systems are common in the engineering world and their stabilization is important in a variety of applications. Any inherently unstable system that needs to be balanced could potentially use a variation of our inertial disk to stabilize it. One example is a rocket during lift off. As it travels upward, it is critical that it remains vertical so that it will stay on its trajectory. The vertical equilibrium is unstable, and so there needs to be an automated actuator to keep the error close to zero. Another application where a system like ours can be used is in the stabilization of people walking. An inertial disk could apply a corrective torque to help people who have trouble walking maintain their balance. In the future, self regulating systems, from segways to jetpacks, will become increasingly common, and there is no doubt that automated control of these unstable systems, in particular inverted pendulums, will increase in prevalence.

**Conclusion**

The pendulum functions dependably, and can successfully reject minor disturbances. The final mechanism has a power switch that also serves the added purpose of initiating the balancing script on the Arduino. This is advantageous since the numerical integral variables
need to be cleared for each new trial. The pendulum is also equipped with position limit switches, which automatically cut the power to the pendulum when it falls all the way to either side. For more details about the pendulum’s final performance, please refer to the video provided. It can also be found online at the following URL:

http://www.youtube.com/watch?v=pWb8HbDCPwU