Development of an Inverted Pendulum System for Investigating Fluidic Thrust Vectoring Control

Alexander D. Colli

University of New South Wales at the Australian Defence Force Academy

Small satellites have seen an increase in recent years, and their use has been proposed in active space debris removal, which demands motors capable of high delta-Vs and precise attitude control. However, existing methods for attitude control impose relatively high mass and volume penalties when used for small-scale applications. Fluidic thrust vectoring is being investigated as a possible solution to this problem. The inverted pendulum problem was chosen to represent a highly simplified case of fluidic thrust vectoring in order to investigate how such a thrust vectoring method may be controlled, with the secondary aim of developing an educational tool for teaching control at UNSW Canberra introduced during the project. The system was simulated in MATLAB® Simulink™ to determine suitable control and hardware parameters for both PID and Fuzzy Logic Control, and two iterations of a physical system capable of achieving the calculated outputs were constructed. Both control methods were then demonstrated on each iteration with varying success.

Nomenclature

Abbreviations:

- EEPROM = Electrically Erasable Programmable Read-Only Memory
- FTV = Fluidic Thrust Vectoring
- PID = Proportional-Integral-Derivative
- RCS = Reaction Control System
- SRM = Solid Rocket Motor

Variables:

- \( g \) = Acceleration due to gravity [m.s\(^{-2}\)]
- \( l \) = Pendulum length [m]
- \( I \) = Moment of inertia [kg.m\(^2\)]
- \( m \) = Pendulum arm mass [kg]
- \( N \) = Pendulum horizontal force [N]
- \( P \) = Pendulum normal force [N]
- \( v \) = Cart velocity [m.s\(^{-1}\)]
- \( \ddot{x} \) = Cart acceleration [m.s\(^{-2}\)]
- \( \theta \) = Angular displacement [rad]
- \( \ddot{\theta} \) = Angular acceleration [rad.s\(^{-2}\)]

I. Introduction

A. General Introduction

In recent years, small satellites have become increasingly attractive to space agencies and companies around the world concerned with space debris removal and asteroid mining missions, largely due to their relatively low manufacturing and payload delivery costs, as well as developments in miniaturisation technologies. However, such missions currently require solid rocket motor (SRM) propulsion systems to provide large delta-Vs, as well as precise attitude control capabilities in order to conduct the desired rendezvous successfully. The most

\(^1\) School of Engineering & Information Technology. ZEIT4500

Final Project Report 2017, UNSW Canberra at ADFA
common attitude control system in small spacecraft built for these missions uses reaction control system (RCS) thrusters to provide control torques by consuming the main propellant. Thruster Orientation Mechanisms capable of manipulating the thrust vector of the spacecraft’s main engines have been developed and implemented to reduce propellant consumption, but at high mass and reliability costs. Initial results from research conducted at the University of New South Wales, Canberra, indicate that fluidic thrust vectoring (FTV) may provide a greater range of thrust angles than commonly used RCS thrusters[1], motivating the investigation of how such a system would be controlled.

An inverted pendulum, also referred to as a “pendulum and cart” system, as shown at Fig. 1, is favoured for representing the dynamics of a rocket with thrust misalignment under the assumption that all other disturbance torques are neglected[2-5]. This makes it useful for investigating and demonstrating FTV attitude control for satellites using SRMs. Further increasing the appeal of the system to this problem, the inverted pendulum is a well-studied control problem, particularly in the field of control system design and implementation, due to its inherently non-linear and unstable nature. Many undergraduate level textbooks such as Ogata[2] use the inverted pendulum as an example for mathematical modelling physical systems.

![Typical inverted pendulum system.](image)

**Figure 1. Typical inverted pendulum system.**

### B. Aims

The primary aim of this project was to develop a pendulum and cart system to demonstrate the attitude control of small satellites using FTV. This included simulation of the system to stabilise the pendulum arm in MATLAB®, implementation of a PID and Fuzzy controller in the Arduino Environment and comparison of the final system with the simulated results.

A secondary aim of the project was to develop a package consisting of software code and apparatus for use as an educational tool in the teaching of Vibrations and Control within the Mechanical Engineering program at UNSW Canberra. The requirements introduced to the project were the need for the system to be compact in size and easily transportable to allow students to take the apparatus home if required.

### II. Method

#### A. System Dynamics

An understanding of the system dynamics was necessary to determine performance requirements of hardware to be selected, by providing an insight into required motor speeds and accelerations. The system dynamics, shown in Eq. 1 and Eq. 2, were adapted from a derivation of an inverted pendulum system with a force as the input [6], using the previous nomenclature from Fig. 1. It should be noted that only the angular position of the pendulum was to be controlled, and hence the cart equations were not required.

\[
P \sin(\phi) + N \cos(\phi) - mg \sin(\phi) = ml \ddot{\phi} + mx \cos(\phi) \quad (1)
\]

\[
-P \sin(\phi) - N \cos(\phi) = I \ddot{\phi} \quad (2)
\]

Combining Eq. 1 and Eq. 2 returns the nonlinear mathematical model of the system:

\[
(I + ml^2) \ddot{\phi} + mg \sin(\phi) = -mlx \cos(\phi) \quad (3)
\]
In order to simplify the model for ease of implementation, Eq. 3 was linearised for the operating range. Given that the system was to be controlled about an unstable equilibrium point at $\pi$ radians from the stable equilibrium, linearisation was conducted about $\theta = \pi$. By taking $\theta = \pi + \phi$ in the operating range ($\phi$ being a small angle from the y-axis), $\cos \theta = -1$, $\sin \theta = -\theta$ and $\theta \approx 0$. Note also $\dot{\theta} = \dot{\theta}$ and $\ddot{\theta} = \ddot{\theta}$. The resulting linearised system model is described by Eq. 4.

\[(l + ml^2) \ddot{\theta} - mgl \dot{\theta} = ml \dot{x} \quad (4)\]

The linearised model was then adapted from a model which takes force as an input to a model with takes cart velocity as an input, given the test apparatus developed was better suited to providing a motor speed as an input, rather than a force. This was achieved by representing acceleration in terms of velocity instead of position as shown in Eq. 5. By translating this into the Laplace domain and rearranging, the transfer function of the linearised system, for use in system simulation and development of a PID controller, was then obtained as shown in Eq. 6.

\[
\frac{\theta(s)}{V(s)} = \frac{cs}{b_2s^2 - b_1} \quad (6)
\]

Where:

$c = ml$

$b_1 = mgl$

$b_2 = (l + ml^2)$

B. MATLAB® Simulation/Controller Design

1. Open Loop Response

The transfer function derived in Section II was used to construct a linear system model in MATLAB® Simulink™. The open loop response was found to be unstable as expected, however featured angular position continually increasing beyond the known equilibrium point of $\theta=180^\circ$. In order to examine a more realistic nonlinear system model, a second Simulink™ model was constructed as shown in Fig. 2. This model exhibited more realistic behaviour than the linear model, finding the expected stable point at $180^\circ$.

![Figure 2. Non-linear Simulink™ system model.](image)

2. PID Control

A PID controller was projected onto both the linear and non-linear systems and tuned to achieve a response time under 0.1s using MATLAB®’s PID Tuner, an application allowing automatic tuning of PID controllers for single-input-single-output plants. The resulting closed loop impulse response for both the linear and non-linear model, with the PID parameters from Table 1 applied, is shown in Fig. 3.
Table 1: Tuned PID values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_P$</td>
<td>1300</td>
</tr>
<tr>
<td>$K_I$</td>
<td>6730</td>
</tr>
<tr>
<td>$K_D$</td>
<td>65</td>
</tr>
</tbody>
</table>

Figure 3. Closed-loop impulse response of the linear and non-linear models using tuned PID values.

3. Fuzzy Logic Control

A fuzzy logic controller was developed using the Fuzzy Logic Toolbox within MATLAB®, due to its easy-to-use interface and to allow visualization of each fuzzy set. The fuzzy controller featured two input variables, pendulum angle and angular velocity, and the singular output of cart velocity. Each of these fuzzy sets were given five membership functions: high negative (HN), medium negative (MN), small (S), medium positive (MP) and high positive (HP). The input and output fuzzy sets are shown within the Fuzzy Logic Toolbox interface in Fig. 4.

Figure 4. System input and output fuzzy sets created within the MATLAB® interface.
The derivation of the fuzzy control rules was heuristic in nature and primarily based off the same criteria detailed in [7], with amendments made to the rules due to the absence of a requirement to control the position of the cart. The control rules established are best summarised using Table 2.

<table>
<thead>
<tr>
<th>Angle</th>
<th>HN</th>
<th>MN</th>
<th>S</th>
<th>MP</th>
<th>HP</th>
</tr>
</thead>
<tbody>
<tr>
<td>HN</td>
<td>HN</td>
<td>HN</td>
<td>HN</td>
<td>MN</td>
<td>S</td>
</tr>
<tr>
<td>MN</td>
<td>HN</td>
<td>HN</td>
<td>MN</td>
<td>S</td>
<td>MP</td>
</tr>
<tr>
<td>S</td>
<td>HN</td>
<td>MN</td>
<td>S</td>
<td>MP</td>
<td>HP</td>
</tr>
<tr>
<td>MP</td>
<td>MN</td>
<td>S</td>
<td>MP</td>
<td>HP</td>
<td>HP</td>
</tr>
<tr>
<td>HP</td>
<td>S</td>
<td>MP</td>
<td>HP</td>
<td>HP</td>
<td>HP</td>
</tr>
</tbody>
</table>

Table 2: Fuzzy Logic Control output rules.

### III. System Overview

#### A. System Architecture

The pendulum and cart system was developed using the system architecture shown in Fig. 5.

#### B. Hardware

1. **Stepper Motor**

   The main actuator of the system is a stepper motor connected to the wheels of the cart via a sprocket and plastic chain. A stepper motor was selected for actuation as it allows control of both velocity and acceleration. The current version of the cart is driven by a bipolar stepper motor, Polulu SY57STH76-2804A. This motor draws 2.8A current per phase and is capable of providing a holding torque of 19kg.cm and a maximum acceleration of 3m.s\(^{-2}\) to the cart.

2. **Motor Driver**

   Because of the relatively high current requirement of the motor, a dedicated stepper driver was purchased for use. The driver chosen was the Leadshine DM856, as more readily available driver boards such as the Sparkfun EasyDriver are current limited at around 1A/phase. The DM856 is configurable by a set of DIP switches that select the peak current each phase and the precision of the motor down to 100 increments per step. When using the driver, two digital pins are required to drive the motor: a ‘pulse’ pin to control the stepping motion, connected to PUL+; and a ‘direction’ pin to control the direction of rotation, connected to DIR+. The PUL- and DIR- pins should be connected to ground. The enable (ENA) pins were not used.
3. Sensors

The pendulum was initially designed to use the MPU6050 Inertial Measurement Unit, a 6 axis accelerometer and gyroscope, to determine both the angular position and velocity of the pendulum. This single unit was selected as it contains an onboard digital motion processor which allows system state variables to be obtained faster than separate accelerometer and gyroscope sensors, serving as an ‘all-in-one’ sensor. However, despite the MPU6050 being successfully configured to simultaneously process angular position and velocity data, it was discovered that the sensor exhibited processing lag when code for other hardware, such as the stepper motor, was introduced and executed in the same script. Therefore, a potentiometer was used in place of the MPU6050 for the second iteration of the pendulum cart system design, in order to provide angle position feedback. No replacement was able to be implemented for obtaining angular velocity data due to time constraints.

4. Controller

The pendulum cart system is controlled by an Arduino Uno R3. The Uno features an ATmega328P microcontroller unit with a clock speed of 16MHz, as well as 14 digital input/output pins, 6 analog inputs, 32kB flash memory, 2kB SRAM and 1kB EEPROM. For this project, pins 8 and 9 were used to drive the stepper motor, as DIR+ and PUL+ respectively. Pins A4 and A5 are used for I²C communications with both the MPU6050 and 256kbit EEPROM as the data and serial lines respectively, with pin 2 connected to the external interrupt pin of the MPU6050. With the introduction of the potentiometer for the second design iteration, pin A2 was also used for obtaining potentiometer readings.

5. Power and Communication

Power is supplied to the system via three rechargeable 11.1V 5500mAh Lithium Polymer batteries, with two connected in series to supply 22.2V to the stepper motor and one powering the Arduino Uno and peripheries through a 5V regulator. This theoretically allows 1.5 hours of system operation before recharge is required. All other materials used in the construction of the pendulum arm and cart were sourced from UNSW Canberra workshops. Communication between the cart and computer is intended to occur via serial connection, however a 256kbit I²C EEPROM was incorporated for storage of system state variables, allowing later retrieval upon connection to a computer should a suitably long serial cable be unavailable. With each state variable occupying four bytes, this allows 8192 readings to be stored on the EEPROM. A photograph of the final configuration of hardware is shown in Fig. 6.

![Figure 6. Constructed pendulum cart system during operation.](image-url)
C. Software

The operation of the developed system occurs via the execution of a program by the Arduino Uno. This program is written in C and can be uploaded to the controller using the Arduino Integrated Development Environment (IDE). When power is provided to the system, the main sketch first clears the external EEPROM, initialises connection with the MPU6050 if it is to be used, and defines the system control rules. Maximum speeds and accelerations to be used by the stepper are also set to prevent slippage of the motor.

The program then executes a continuous loop for the time period, in milliseconds, specified by the user. Each loop, two main tasks are performed. First, current values of system state variables are obtained from the MPU6050/potentiometer. Second, the control algorithm is evaluated, and commands are issued to drive the stepper motor at particular speed. The majority of functions in the main loop are non-blocking, however steps are called every few lines to prevent stuttering during blocking operations.

At an interval of milliseconds that may be specified by the user, the measured state variables are either sent to the serial console or written to the EEPROM, depending upon the configuration selected. This functionality may be disabled if data collection is not required.

In order to simplify communication between sensor and controller, as well as implementation of control rules and driving of the stepper motor, the built-in Arduino Wire and PID libraries were used in conjunction with the independently developed AccelStepper, MPU6050 and Fuzzy libraries [8-10]. This allowed significant time to be saved which would otherwise have been spent on software development.

D. Results

With the potentiometer fitted to measure angular position, the tuned PID parameters determined from the MATLAB® simulation were found to be unsuitable for control of the physical system. The Ziegler-Nichols method was used to determine new PID parameters of $K_p = 150$, $K_i = 250$ and $K_d = 2$. Additionally, application of fuzzy control was restricted to single input with the use of the potentiometer, as a suitable method for determining angular velocity could not be implemented due to time constraints. Pendulum data for both control methods was collected from the system at a rate of 200Hz and is represented in Fig. 7 and Fig. 8.

![Figure 7. Data collected from the system operating PID control.](image-url)
By applying the PID parameters to the final system the pendulum angle was successfully controlled. The single input fuzzy controller, however, was found to be insufficient for stabilisation of the pendulum, exhibiting oscillatory behaviour of increasing magnitude about the set point until the pendulum fell.

IV. Conclusions and Recommendations

The developed pendulum cart system provides a useful platform on which to investigate control methods for application in fluidic thrust vectoring. When combined with the simulation files generated, the usefulness of this system is extended further into the educational space for students, as the effect of varying control parameters and rules can be visualised using MATLAB® software. The design of the device allows system variables to be stored during operation without the requirement for serial connection, allowing it to be operated with large cart displacements if required.

Whilst PID control was successfully demonstrated, further work is required to achieve stabilisation of the pendulum using fuzzy control. It is recommended that future work focus on successful incorporation of the MPU6050 sensor with the system, or alternatively individual sensors to obtain system state values. This would allow fuzzy control, among other control methods not explored in this project such as state-space representation, to be implemented, increasing the value of the system as a basis for testing control methods.

The program that runs on the Arduino Uno could also be improved, as both writing to the EEPROM and reading from the MPU6050 feature non-blocking elements. This prevents the motor from stepping whilst these sections of code are being executed, resulting in stuttering of the motor. An improved version would use parallel processing or instruction scheduling to allow uninterrupted operation of the motor whilst code dealing with sensors and other peripheries is executed.

Acknowledgements

I would like to thank Dr. Sean Tuttle for his assistance and supervision throughout what has been a challenging but rewarding project, and extend this thanks to Simon Barraclough for his assistance also. Additionally, I would like to thank my friends and family for their support and encouragement over the year.

References


