Kalman Filtering of IMU Sensor for Robot Balance Control

by

Gina Christine Angelosanto

Submitted to the Department of Mechanical Engineering
on May 9, 2008, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Mechanical Engineering

Abstract

This study explores the use of Kalman filtering of measurements from an inertial measurement unit (IMU) to provide information on the orientation of a robot for balance control. A test bed was created to characterize the random noise and errors inherent to orientation sensing in the MicroStrain 3DM-GX1 IMU for static cases as well as after experiencing an impact force. Balance simulations were performed to control the center of mass location of a robot modeled as an inverted pendulum. The controlled center of mass trajectories with state estimates generated from Kalman filtering were compared, where possible, to the CM trajectory based on unfiltered sensor measurements of the states. For the simple case of inverted pendulum control, it was determined that noise and error in the IMU are sufficiently small that Kalman filtering is not necessary when all states can be measured, but results in significant improvements in the RMS error of the actual and desired center of mass positions.

Thesis Supervisor: Wai K. Cheng
Title: Professor, Department of Mechanical Engineering

Thesis Supervisor: Andreas Hofmann
Title: Director of Machine Learning R&D, Vecna Technologies
Acknowledgments

I would like to thank my advisors, Andreas Hofmann from Vecna Technologies and Professor Wai Cheng from Massachusetts Institute of Technology, for their help and guidance in developing this thesis. I would also like to thank Josh Ornstein and Josh Young from Vecna Technologies for their help with electronics and programming.
List of Figures

1.1 Vecna BEAR robot [1]. .......................................................... 7

2.1 Block diagram representation of a system with Kalman filtering [5]. Noisy output measurements and uncorrupted control are inputs to the Kalman filter. ........ 11

3.1 Hardware test bed used to compare angle readings of the IMU and a magnet encoder. 13

3.2 Simplified schematic diagram of “inverted pendulum” model of robot. A massless link of length \( l \) extends from the zero moment point to the center of mass. The angle of the link with respect to the ground \( \theta \), horizontal locations of the center of mass and zero moment point \( x_{cm} \) and \( x_{zmp} \), and height of the center of mass \( z_{zmp} \) are shown. .................................................. 16

3.3 Force \( F \) acting on the center of mass from massless link is shown resolved into components in the \( x \) and \( z \) directions. ................................. 17

4.1 Graph of angle in degrees vs. time for both encoder (shown in red) and IMU (shown in blue). Angle was held at rest for approximately 5 seconds per angle in angle increments of roughly 10 degrees. ................................. 22

4.2 Graph of angle in degrees vs. time for both encoder (shown in red) and IMU (shown in blue). This graph shows the angle recordings from the two sensors over a full 360 degrees, and demonstrates that the difference between the IMU and encoder is minimized around both zero and +/- 180 degrees. ................................. 24

4.3 Graph of IMU error vs. angle in degrees for static angle increments. Red dashed line connects error averaged over each angle increment; black dots show error for each data point included in the angle increment, or the range of the error for each average angle. .................................................. 25
4.4 Cumulative average of encoder angle converging to overall average of encoder angle across a single 'step.' ........................................... 26

4.5 Encoder error vs. angle. Dots represent mean error over each angle increment; line is a sine curve fit to the data. ........................................... 26

4.6 Standard deviation of error for each static angle increment vs. mean encoder reading. Data for three trials is shown. ........................................... 27

4.7 Graph on left shows 1000 linear fits to standard deviation of error vs. angle generated by bootstrapping imposed over actual data. Histogram on right shows the frequency of line slopes. Note that the order of magnitude of the standard deviation is $10^{-2}$ degrees, while the order of magnitude of the slopes is $10^{-4}$. ........................................... 28

4.8 Graph of angle in degrees vs. time for both encoder (shown in red) and IMU (shown in blue). Pendulum link holding IMU was released from a 90 degree starting angle, struck an object at 0 degrees and came to rest for approximately 3 seconds, and was then released. ........................................... 29

4.9 Graph of angle in degrees vs. time for both encoder (shown in red) and IMU (shown in blue). Pendulum link holding IMU was released from a 90 degree starting angle, struck an object at 0 degrees and was immediately released. ........................................... 31

4.10 Center of mass position vs. time from balance simulation without Kalman filter. Actual CM position and sensor measurements are shown. ........................................... 32

4.11 Center of mass x-position vs. time with an initial displacement of 10 cm, generated from linearized pendulum model with all states measured. ........................................... 33

4.12 Center of mass x-position vs. time with an initial displacement of 10 cm, generated from nonlinearized pendulum model with all states measured. ........................................... 34

4.13 Center of mass x-position vs. time with an initial displacement of 10 cm, generated from nonlinearized pendulum model with impact force. ........................................... 36

4.14 Center of mass x-position vs. time with an initial displacement of 10 cm, generated from linearized pendulum model without velocity measurements. ........................................... 38

4.15 Center of mass x-position vs. time with an initial displacement of 10 cm, generated from nonlinearized pendulum model without velocity measurements. ........................................... 39
1 Introduction

Vecna Technologies, Inc. is currently developing a Battlefield Extraction Assist Robot (BEAR) to remove wounded soldiers from battlefields, handle hazardous materials, perform search and rescue missions, and assist in various other dangerous tasks which would otherwise pose a threat to human life. The design for the BEAR, pictured in Figure 1.1, features a mobility base with two tumbled joints. The BEAR may drive on the treads in a “kneeling” position as shown, or rise into a “standing” position and walk as a bipedal robot. Given the difficult nature of its tasks, the BEAR robot must be able to operate on rough terrain and under a variety of different environmental conditions, in addition to being able to carry up to 500 pounds of payload. The ability to operate in a rugged terrain with a payload necessitates
excellent balance control for the BEAR.

Knowing the location of the center of mass is essential for balance of the BEAR; bipedal control schemes frequently attempt to maintain the center of mass vertically upright and reduce angular momentum about the center of mass [2]. Consequently, a good sensor is needed to determine the orientation of the robot in order to calculate the center of mass position. Inertial measurement units (IMU) are potentially very useful sensors for military robotics. A single IMU can report measurements about three axes, reducing the number of independent sensors needed on the robot. More importantly, the IMU is a single self-contained box that can be mounted anywhere on the robot; it does not need to be installed in a joint or connected to the rotating joint through a gear train as would an encoder. The compactness, flexibility in sensor placement, and ability to sense orientation without wear associated with gear trains or other methods of coupling to a joint make the IMU an excellent candidate for use on a heavy-duty robot subjected to rough terrain and environments.

However, use of an IMU may present some problems for balance control. Problems with using an IMU to measure the orientation of the robot can result from nonlinearities or systematic error in the sensor as well as from random sensor noise. Inaccurate or incomplete calculation of measurements that cannot be taken directly can also negatively affect balance control. For instance, in a multi degree of freedom robot, finding the center of mass and ground contact point locations will likely depend on measurements from several different IMUs located on various parts of the robot. Small inaccuracies in each of these measurements could add up to significant errors in the final calculation. The IMU may also be subjected to jarring and impact as the BEAR performs its tasks; the sensor accuracy may be affected by such impact forces. If the problems of sensor error, noise, and incomplete measurements are significant, they must be overcome before the IMU can be used to reliably balance the BEAR.

One possible way of reducing high-frequency sensor noise is to use a low-pass filter. However, discrete low-pass filters result in inherent time delays when applied to systems in real time, which could reduce the stability of the system. Low-pass filters also do not address the issues of inaccurate or unavailable measurements. The use
of a model-based recursive discrete Kalman filter with measurements from the IMU will be examined as a means of controlling the center of mass position of the BEAR. Although generally considered to be somewhat computationally expensive, the ability of the Kalman filter to incorporate both a model of the system and inaccurate measurements, along with the availability of fast and inexpensive computers make the Kalman filter an excellent choice to filter noise and estimate states [3].

The goal of this study is to obtain sufficiently accurate state estimations from noisy and incomplete measurements to successfully control the robot center of mass position using an IMU to measure orientation. The robot is mechanically modeled as a linearized inverted pendulum, one of the simplest plant models that still maintains the problem of balance control. Error and noise from a MicroStrain 3DM-GX1 IMU in one dimension are characterized by creating a test bed to compare the orientation reported by the IMU with a known value; the static error and error upon impact are both examined. Control of the inverted pendulum model is then simulated in Matlab using a Kalman filter to optimize estimation of the states based on both the linear model provided and the noisy and/or inaccurate measurements expected from the IMU based on testing.

2 Background

2.1 Kalman Filter

The method of least-squares has been used to fit a model to data in a wide range of applications from finance to engineering. The concept of using a least-squares analysis on noisy data to find optimal coefficients for a model of a system has been of particular interest in the field on control theory. One of the main problems with using least-squares filtering techniques to estimate states in controls is “batch processing,” or the requirement that all measured data be available before a fit to the data can be generated. Consequently, filtering by least-squares could not occur in real time [3].
One solution to the problem of real-time least-squares filtering proposed by Rudolph Kalman in 1960 was a recursive filter that allowed each state estimation to be made using the method of least-squares as soon as a measurement is taken. The Kalman filter estimates states based on a mathematical model represented by linear difference equations; essentially, these state estimations are made by computing a model-based estimate and adding to it a scaled "residual" representing the difference between the predicted and actual measurements. The scaling factor, which is calculated for each time step, is referred to as the Kalman gain and reflects how much "trust" is placed on the measurements as compared to the model. A small Kalman gain indicates low confidence in the measurements, while a high Kalman gain places priority on measurements when estimating the state.

The Kalman filter may be thought of as a two-step process consisting of a prediction step and a correction step [4]. In the prediction step, the anticipated state $x^*$ based on the linear mathematical model of the system is represented as

$$x_k^* = A \hat{x}_{k-1} + B u_{k-1}$$  \hspace{1cm} (1)

where $\hat{x}_{k-1}$ is the current state estimate and $u$ represents system inputs. The anticipated error covariance matrix $P^*$ can then be estimated as a function of the model matrix $A$, the current error covariance $P_{k-1}$, and the process noise covariance $Q$ reflecting random noise at the control input:

$$P_k^* = A P_{k-1} A^T + Q$$  \hspace{1cm} (2)

In the correction step, the Kalman gain is computed at time step $k$ as

$$K_k = P_k^* H^T [H P_k^* H^T + R]^{-1}$$  \hspace{1cm} (3)

where $H$ is a matrix relating the states to the measurements and $R$ is the measurement noise covariance reflecting noise added at the sensors. The estimate of the states may then be calculated as a function of the model prediction, measurement, and Kalman gain:
\[ \hat{x}_k = \hat{x}_k^* + K_k (z_k - H \hat{x}_k^*) \] (4)

The error covariance \( P \) is also corrected based on the Kalman gain as

\[ P_k = (I - K_k H) P_k^* \] (5)

This process of predicting and correcting the states repeats for all \( k \). This description of the Kalman filter assumes white Gaussian noise and a linear model of the system. The extended Kalman filter has been developed to incorporate nonlinear difference equations [4]. A more detailed description of the Kalman filter may be found in [3 - 4].

Figure 2.1 shows a block diagram of Kalman filter implementation in Matlab. Process noise is added to the control input \( u \); this corrupted control signal is then sent to the plant. Noise is likewise added to the “true” outputs of the plant, \( y \), to represent noisy measurements \( y_v \). Both the uncorrupted control signal and noisy measurements are input to the Kalman filter, which outputs the estimated states \( y_e \) based on the recursive least-squares filter discussed above. The Kalman filter combines both a plant model and measurements to make optimal state estimates in a least-squares sense.

![Block diagram representation of a system with Kalman filtering](image)

**Figure 2.1:** Block diagram representation of a system with Kalman filtering [5]. Noisy output measurements and uncorrupted control are inputs to the Kalman filter.
2.2 Sensor Technology

Inertial measurement units generally consist of three accelerometers and three gyroscopes oriented to sense accelerations and angular rates about three orthogonal axes. The MicroStrain 3DM-GX1 IMU also uses three orthogonally oriented magnetometers to more accurately sense orientation. The technology used in accelerometers normally consist of a mass on springs or a beam allowed to move relative to the IMU housing; distance sensors or strain gages inside the IMU are used to determine the acceleration as a function of the mass and strain in the springs or distance moved [6]. Common implementation of accelerometers may involve capacitive, piezoresistive or piezoelectric distance sensing [7]. Gyroscopes may be implemented with a vibrational element sensitive to the Coriolis acceleration as the sensor is rotated [6]. The MicroStrain 3DM-GX1 includes “gyro g-stabilization” implemented in software to counteract the effects of gravity on the gyros [8]. Information from the accelerometers and gyros within the IMU is processed internally to yield the orientation; in some models the angular rates can be reported as well. Further information on the internal processing of IMU sensor data may be found in [9].

3 Methods

3.1 Physical testbed

A testbed was created to study the measurement error in the IMU. When held static, the IMU angle was compared to the known angle of the test setup; dynamic orientation and orientation after impact were compared to angles registered by an encoder. The hardware, shown in Figure 3.1, consisted of a enclosed Hall-effect encoder held rigidly inside of an aluminum base. A pendulum link of length 6 inches was affixed to the shaft of the encoder and had a right-angle shelf on the bottom where the IMU sat. The IMU was fixed to the link base with heavy-duty foam tape and zip ties threaded through mounting holes in the IMU. An arc-shaped plate with
Figure 3.1: Hardware test bed used to compare angle readings of the IMU and a magnet encoder.
holes drilled every 10 degrees over a 180 degree interval was attached to the aluminum base such that a pin, when placed through a hole in the arc and a corresponding hole in the pendulum link, could hold the link at a fixed angle. The link could be held at any multiple of 10 degrees over a full circle. The base of the test bed could be clamped to a tabletop to prevent motion of the base itself as the link was moved.

The parts for the test bed were created from 1/4-inch aluminum sheet, 1-inch by 2.5-inch extruded rectangular aluminum stock, and 1-inch aluminum L-bracket. The pendulum link and the affixed shelf to hold the IMU were fabricated with an Omax waterjet rapid prototyping machine from the aluminum sheet. The solid model shown in part (a) of figure 3.1 was generated to create the toolpaths for the waterjet; all holes were deliberately made under-sized and reamed on a drill press to avoid inaccuracies do to the waterjet kerf lines. The rectangular components of the base and L-bracket were made with a manual mill with a digital position readout accurate to .0001 inches. The mill head was realigned immediately before fabrication. The shelf to hold the IMU was attached to the pendulum link with the L-bracket and the entire pendulum link assembly was bolted to a gear attached to the encoder shaft as shown in part (b) of Figure 3.1.

3.2 Noise Characterization

Data was recorded simultaneously from the encoder and the IMU for two cases: static angle measurements and impact applied to the pendulum link. The base of the testbed was clamped to a laboratory table for all measurements in order to prevent the base from moving while the pendulum was in motion. For each case, prior to measuring the angles the encoder reading was recorded when the pendulum link was allowed to come to rest hanging freely under its own weight. This position was then considered the true zero degree point and the value was consequently subtracted from each encoder reading. Data from the encoder, IMU, and the elapsed time were recorded on a laboratory computer and imported into Matlab for further analysis. Encoder values were converted into degrees from the raw data prior to analysis.

The static angle tests involved pinning the pendulum link at a set angle for
approximately 5-10 seconds, then moving the link down one angle increment on the test bed and repeating the process. Two tests were run recording the static angles over the entire range of 360 degrees in 10-degree increments, beginning at 90 degrees (horizontal). Five tests were run recording static angles over a range of 180 degrees, beginning at 90 degrees and moving in 10-degree increments to -90 degrees. Errors between the encoder, IMU and expected static angle based on the test setup were calculated over each angle increment, and the standard deviation of the IMU noise was determined over the increment as well.

Impact tests involved releasing the pendulum link from a fixed angle so that it fell freely under its own weight. Ten tests involved an impact force applied to the pendulum after its release, in the form of a pin inserted into a hole on the test bed which interfered with the falling pendulum link. For three of these impact tests, the pendulum was released from 90 degrees, struck a pin at 50 degrees, and remained at rest for approximately 5 seconds before the pin was removed and the pendulum continued its fall. Another four tests involved the same process with a pin located at 0 degrees. The final three impact tests involved the pendulum being released from 90 degrees and striking a pin at 0 degrees which was immediately removed, allowing the pendulum to continue its oscillation about the zero degree point. The time history of these motions was recorded and imported into Matlab for analysis.

### 3.3 Balance Simulation

A simple mathematical model of an inverted pendulum was used to simulate the problem of robot balance. The robot was modeled as a single point mass fixed to a rigid, massless link of constant length $l$, as shown in Figure 3.2. The point of contact with the ground, or zero moment point of the model was allowed to move in the x-direction but constrained to remain at a fixed vertical position $z = 0$. The force acting on the center of mass through the link is shown decomposed into its components in the x and z directions in Figure 3.3.

Three state variables were selected for the system: the position in x of the center of mass $x_{cm}$, the velocity in x of the center of mass, $x_{cm}$, and the position in x of the
Figure 3.2: Simplified schematic diagram of “inverted pendulum” model of robot. A massless link of length \( l \) extends from the zero moment point to the center of mass. The angle of the link with respect to the ground \( \theta \), horizontal locations of the center of mass and zero moment point \( x_{cm} \) and \( x_{zmp} \), and height of the center of mass \( z_{zmp} \) are shown.

zero moment point \( x_{zmp} \). Three linear equations relating these state variables and their derivatives are necessary to represent the system in state-determined form. The input to the system is the velocity of the zero moment point, which is implemented in real life as the speed of the treads driving the base of the robot. From Figure 3.3, the forces acting on the center of mass may be expressed as

\[
F_x = F \cos \theta = F \frac{x_{cm} - x_{zmp}}{l} \tag{6}
\]

\[
F_z = F \sin \theta = F \frac{z_{cm}}{l} \tag{7}
\]

Combining equations 6 and 7 yields the force acting on the center of mass in the x-direction as a function of the x positions of the center of mass and zero moment point, the force on the center of mass in the z-direction, and the height of the center of mass:
Figure 3.3: Force $F$ acting on the center of mass from a massless link is shown resolved into components in the $x$ and $z$ directions.

$$F_x = F_z \frac{d^2x_{cm}}{dt^2} = m \frac{d^2x_{cm}}{dt^2}$$

(8)

where $m$ is the mass of the robot. This continuous equation can be used to approximate a finite difference equation relating the derivative of the CM velocity to the position of the CM and ZMP:

$$\ddot{x}_{cm}[i + 1] = \ddot{x}_{cm}[i] + \frac{dt}{m z_{cm}} \frac{F_z}{x_{cm}[i]} - \frac{dt}{m z_{cm}} \frac{F_z}{x_{zmp}[i]}$$

(9)

Note that the difference equation assumes a first-order approximation of the acceleration of the CM as the difference between two subsequent velocities of the CM divided by some small time step, $dt$. Two other difference equations needed to describe the system states are

$$x_{cm}[i + 1] = x_{cm}[i] + \dot{x}_{cm}[i] dt$$

(10)

and
\[ x_{zmp}[i + 1] = x_{zmp}[i] + u[i]dt \]  

(11)

where \( u \) is the control input, \( \dot{x}_{zmp} \). Combining these equations into the form

\[ \dot{x}[i + 1] = A\dot{x}[i] + Bu[i] \]

yields the following expression used to generate a model of the plant:

\[
\begin{pmatrix}
    x_{cm}[i + 1] \\
    \dot{x}_{cm}[i + 1] \\
    x_{zmp}[i + 1]
\end{pmatrix} =
\begin{pmatrix}
    1 & dt & 0 \\
    F_{z_{cm}}/m & 1 & F_{z_{cm}}/m \\
    0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
    x_{cm}[i] \\
    \dot{x}_{cm}[i] \\
    x_{zmp}[i]
\end{pmatrix} +
\begin{pmatrix}
    0 \\
    0 \\
    u[i]
\end{pmatrix}
\]

(12)

For the purposes of simulation, the model is considered to be linear; that is, the variations in the angle \( \theta \) are assumed to be within +/- 10 degrees such that \( \sin \theta \approx \theta \). As a result, there is little variation in the CM height \( z_{zmp} \), and the acceleration in the z-direction is negligible such that the force in z is assumed to be only the constant weight of the center of mass. Consequently, all entries in the matrices A and B are constants for the linear model.

To form the control law used to determine the x-velocity of the zero moment point, a PD control law is first used to determine the desired acceleration of the center of mass ("desired" states are marked with a hat). PD control is used for a fast response with low overshot. The terms \( K_p \) and \( K_v \) represent the proportional and derivative gains, respectively.

\[ \ddot{x}_{cm} = K_p(\dot{x}_{cm} - x_{cm}) + K_v(\ddot{x}_{cm} - \dot{x}_{cm}) \]

(13)

The desired zero moment point position can be found by rearranging Eqn. 8,

\[ \hat{x}_{zmp} = -\frac{z_{zmp}m\ddot{x}_{cm}}{F_z} + x_{cm} \]

(14)

Next, the desired ZMP velocity (the control input) was computed using proportional control:

\[ \dot{x}_{zmp} = u = K(\dot{x}_{zmp} - x_{zmp}) \]

(15)
With the plant modeled and the feedback control laws generated, it was possible to create simulations balancing the pendulum in Matlab. However, simulations run using the linear plant model to provide the Kalman input with a basis for estimates and generate the “true” states fail to take into account the fact that in reality, the measured states will reflect the nonlinear dynamics and not the linearized form of the plant. To calculate the value of the control input \( u \) more accurately given that the true system is nonlinear and therefore increase the validity of the simulations, the measured values of the states should reflect those of a nonlinear falling pendulum rather than of the linearized inverted pendulum used to form the plant model. The nonlinear dynamics were modeled assuming that any \( x \)-acceleration of the zero moment point could be represented as a 'virtual' force acting on the center of mass in the opposite direction of the ZMP acceleration:

\[
F_{\text{virtual}} = m\ddot{x}_{\text{zmp}}
\]  

(16)

Taking a sum of torques about the ZMP using the notation of Figure 3.2 yields a differential equation for the angular acceleration in terms of \( \theta \) and the ZMP acceleration,

\[
J\ddot{\theta} = \tau_{\text{virtual}} - \tau_{\text{gravity}} = m\ddot{x}_{\text{zmp}}l\sin\theta - mgl\cos\theta
\]

(17)

where \( J \) is the moment of inertia of the point mass about \( x_{\text{zmp}} \) and \( g \) is the acceleration due to gravity. This equation can be checked by examining Lagrange equations for the inverted pendulum with \( x_{\text{zmp}} \) and its derivatives specified as a constraint:

\[
L = T^* - V = \frac{1}{2}m(\dot{x}_{\text{zmp}})^2 - m\dot{\theta}\dot{x}_{\text{zmp}}\sin\theta + \frac{1}{2}m(\dot{\theta})^2 - mgl\sin\theta
\]

(18)

In the absence of generalized forces, the single equation required to describe the constrained motion of the pendulum reduces to
\[
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\theta}} \right) - \frac{\partial L}{\partial \theta} = 0 = -\ddot{x}_{zmp} \sin \theta + l \ddot{\theta} + g \cos \theta
\]  

(19)

which simplifies to the expression of equation 17. The ZMP acceleration \( \ddot{x}_{zmp} \) was calculated by subtracting subsequent values of the control input \( u \) (the ZMP velocity) and dividing by the time step \( dt \). Eqn. 17 was solved in Matlab for \( \theta \) using Euler’s method. The initial values of \( \theta \) and \( \dot{\theta} \) were calculated each time the control input was changed (that is, every \( \Delta t \) seconds), as

\[
\theta_0 = \arccos \left( \frac{x_{cm0} - x_{zmp0}}{l} \right)
\]

(20)

\[
\dot{\theta}_0 = \frac{u_0 - \dot{x}_{cm0}}{l \sin \theta_0}
\]

(21)

The state \( x_{zmp} \) was calculated by integrating the vector \( u \); \( x_{cm} \) and \( x_{zmp} \) were determined as

\[
x_{cm} = x_{zmp} + l \cos \theta
\]

(22)

\[
\dot{x}_{cm} = \dot{x}_{zmp} - l \dot{\theta} \sin \theta
\]

(23)

A discrete Kalman filter was implemented using the ‘kalmf’ command to estimate the states based on the given plant model and state measurements. Simulations were run with all states measured, then repeated with the CM velocity as an unmeasured state. Three pieces of information were returned for each simulation: the “true” CM position state as determined by the dynamics, the measurement of the CM position including measurement noise, and the Kalman estimate of the CM position. All simulations were performed initially with true states generated from the linear plant, then repeated with true states generated from the nonlinear dynamics simulation. Control of the CM position was also simulated without filtering where possible to show how Kalman filtering can improve upon the stability of the system. The magnitude and form of measurement noise added to the system was based on the
characterization of the noise from physical testing described in the previous section. Separate simulations were performed using data gathered from static testing and impact testing to represent two separate cases of "normal" control of the CM position and control after the IMU has experienced an impact.

The values of constants used in the simulation were a link length $l$ of 1 m, CM height $z_{cm}$ of 1 m, z-direction force $F_z$ of 1000 N, and mass $m$ of 100 kg. The value of $dt$ used for the simulation of dynamics was .01 sec; a value of $\Delta t$ during which the same control input $u$ was applied to the system was chosen to be .1 sec for a mechanical system. Allowing 10 time steps for dynamics to change between recalculations of the control input is an attempt to simulate the fact that discrete computation of the input $u$ may always occur slowly compared to the continuous changes of states due to the system dynamics. Control parameters used a value of 8 for $K_p$, 5 for $K_v$, and 10 for $K_1$; these were determined by simulating balance control without noise or filtering and selecting gains to provide a reasonably fast dynamic response with low overshot. Initial values were set for the system with a .1 m offset of the center of mass in the positive x-direction, zero CM velocity, and zero as the starting x-position of the ZMP.

4 Results

4.1 Noise characterization

Figure 4.1 shows examples of the angles recorded from the encoder and the IMU while the pendulum link was held stationary. In both graphs shown, the angle is held in increments of roughly 10 degrees for a period of time between 5-10 seconds before being moved to another angle. Figure 4.1 (a) shows an overview of the sensor readings over the range of angles from -90 to 90 degrees, which spans the entire range of angles that the inverted pendulum can achieve, while Figure 4.1 (b) focuses more closely on the range of angles from 0 to 90 degrees. In both cases, it is clear from the graphs that there is a wide systematic error in the encoder reading at the +/-
Figure 4.1: Graph of angle in degrees vs. time for both encoder (shown in red) and IMU (shown in blue). Angle was held at rest for approximately 5 seconds per angle in angle increments of roughly 10 degrees.
90 degree extremes which tapers down to a much smaller error near zero degrees. Fortunately, both sensors will be used in real life to measure small fluctuations about zero degrees, where the systematic error is smallest (note that the angle $\theta$ shown in Figure 3.2 is different from the angle recorded on the physical setup by 90 degrees).

One incongruity to clarify is the fact that on the robot, the IMU sensor will be mounted such that the center of mass is in an unstable position (in an "inverted pendulum" configuration) whereas testing of the IMU was performed such that the center of mass was allowed to fall into a stable equilibrium (a regular pendulum configuration). This choice of testing configuration was made so that the IMU could be made to swing back and forth periodically about a desired point or be allowed to fall and impact an object at a set angle under gravity. If the configuration of the testing were reversed such that the pendulum could be made to oscillate around 180 degrees or impact an object as it approached the 180 degree point, some form of external actuation would be needed to make the pendulum exhibit this behavior, complicating the hardware test bed unnecessarily. Figure 4.2 shows that when when the angle at which the pendulum is statically held approaches $\pm$ 180 degrees, the error between the encoder and the IMU again approaches zero. From these results it follows that the IMU and encoder error behave the same near 0 and 180 degrees; consequently, information taken near zero degrees should be useful in describing sensor behavior on an inverted pendulum.

To average the IMU noise, each region of constant angle (the "flat" ranges of the graphs in Figures 4.1 and 4.2) registered by both the encoder and the IMU must be recognized in Matlab as an isolated set of data referred to from here on as a "step". Steps were defined by first creating a new vector containing every third data point recorded from the IMU; each element in this vector was then subtracted from the previous element. When the difference between subsequent resampled angles was sufficiently large, a "jump" in the angles was reported and the time was used as a reference point where the previous step ended and a new step began. This method returned the times corresponding to each individual step in the recorded data. To compare the IMU and encoder angle readings to the anticipated or "true" angle
Figure 4.2: Graph of angle in degrees vs. time for both encoder (shown in red) and IMU (shown in blue). This graph shows the angle recordings from the two sensors over a full 360 degrees, and demonstrates that the difference between the IMU and encoder is minimized around both zero and +/- 180 degrees.

reading based on the physical testbed, a new vector of the true values was created with steps in exact 10-degree increments; the times of each step in this true value vector corresponded to the step times determined by analysis of the IMU and encoder data. Sensor error was then determined simply as the difference between each element in the vector of true angles and the actual angle readings recorded by the sensors. The mean IMU error was determined over each step and is shown as a function of the angle in Figure 4.3 for a single trial. The mean IMU error vs. angle is plotted as a dotted line; the range of errors observed in each step are represented as black dots on the same graph. It is reassuring to note that all observed errors for the IMU are within the sensor specifications of +/- 2 degrees. The error observed for all trials ranged between 0.1359 and -1.712 degrees, a span of 1.848 degrees. The IMU error is likely to be incorrectly biased toward negative values due to the test setup; the pin used to secure the pendulum link in place, though sized to fit the holes in both the link and the base, was likely not a prefect fit. Any looseness when the link was pinned
would be seen as a slight negative error due to gravity acting on the link holding the IMU.

![Graph of IMU error vs. angle in degrees for static angle increments](image)

Figure 4.3: Graph of IMU error vs. angle in degrees for static angle increments. Red dashed line connects error averaged over each angle increment; black dots show error for each data point included in the angle increment, or the range of the error for each average angle.

Another point of concern with reporting the mean orientation of the IMU over each step is the possibility that the data was not recorded for a sufficiently long time for the mean of the IMU readings to reflect the steady-state angle reading. To examine this possibility, the cumulative mean of the IMU values was compared to the overall average IMU angle for every step. If the cumulative average converged to the overall average, the sampling time was considered to be adequate. Figure 4.4 shows an example of the cumulative mean converging to the overall mean for one step, demonstrating that the angle was held constant for a sufficiently long period of time.

It is clear from the data presented in Figures 4.1 and 4.2 that the hall effect rotary encoder exhibits only quasi-linear behavior. The error between the encoder and the
Figure 4.4: Cumulative average of encoder angle converging to overall average of encoder angle across a single 'step.'

Figure 4.5: Encoder error vs. angle. Dots represent mean error over each angle increment; line is a sine curve fit to the data.
angle of the test bed setup, shown in Figure 4.5, is minimized at zero and +/- 180 degrees and reaches a maximum at +/- 90 degrees. Here, black x-marks show the averaged encoder error over each step while the red line shows a sine curve fit to the data. The fit curve is of the form

\[ y = 7.07 \sin (0.035\theta + 1.43) - 6.23 \]  \hspace{1cm} (24)

and was found using Matlab's curve fitting toolbox ("cftool"). The static error in the encoder will be approximated from this equation in future analysis.

After determining the mean of the IMU error over each step, the standard deviation per step was determined. Standard deviation of error vs. angle is plotted for three tests run between -90 and 90 degrees in Figure 4.6. From the graph, it appears that the standard deviation of the IMU error has no correlation with the angle of the IMU, as anticipated for random sensor noise.

![Figure 4.6: Standard deviation of error for each static angle increment vs. mean encoder reading. Data for three trials is shown.](image)

In order to demonstrate more rigorously the lack of any significant correlation between the standard deviation of IMU error and the angle, a bootstrapping
analysis was performed on the data presented in Figure 4.6. One thousand random permutations of the standard deviations and angles were generated, with some (random) data points repeated and others missing in each permutation. One regression line was fit to each new permuted set of data; the 1000 linear regression fits generated from this process are plotted in Figure 4.7 along with the original data. The histogram on the right of Figure 4.7 shows the frequency of slopes of the regression lines fit to the permuted data. The data shows a normal distribution centered on a slope of zero, suggesting once again a lack of significant correlation between the standard deviation of IMU error and the angle.

![Figure 4.7](image)

**Figure 4.7**: Graph on left shows 1000 linear fits to standard deviation of error vs. angle generated by bootstrapping imposed over actual data. Histogram on right shows the frequency of line slopes. Note that the order of magnitude of the standard deviation is $10^{-2}$ degrees, while the order of magnitude of the slopes is $10^{-4}$.

Once it was determined that the standard deviation of the IMU error (representing the random noise in the IMU) has no clear correlation with the angle, the value of the standard deviation of noise to be considered for balance simulation was determined by averaging the set of all error standard deviations calculated over each step in every trial. The mean value of the standard deviation of the IMU error was then found to be a constant 0.0315 degrees. This noise in the IMU readings corresponds to variations of approximately half a millimeter in the center of mass location for an
inverted pendulum model. The static tests of the IMU reveal that the sensor is far more precise than accurate; the average standard deviation of the error is roughly two orders of magnitude smaller than the error itself. This investigation leads to two conclusions regarding the sensor noise: that it is independent of the angle of the sensor and therefore regarded as constant, and it is negligibly small compared to the overall sensor error.

Impact testing

Figure 4.8 shows the recordings of encoder and IMU angles vs. time for an impact test in which the pendulum was released from 90 degrees, struck a pin located at 0 degrees, remained at rest for approximately 3 seconds, and was released. From the graph, it is clear that while the encoder shows a constant angle while the pendulum is at rest (between 10 and 14 seconds), the IMU data depicts a slight negative slope in the angle vs. time. The values of the slope of angle vs. time reported by the IMU show a small deviation from zero, indicating a slight decrease in angle over time. The values are reported for both encoder (shown in red) and IMU (shown in blue) data.
while at rest for the four tests with an impact at zero degrees are reported in Table 1. The average value of the slope of the IMU reading was -0.8875 degrees/second. Table 1 also reports the actual value of the angle of the pendulum link while at rest. Although the pin that the link struck was located at the zero degree point, the width of the pendulum link itself resulted in the non-zero angle of the pendulum center line. The true angle of the pendulum link was calculated by applying Equation 24 to the value of the encoder reading to find the encoder error; maximum IMU error was then determined as the greatest difference between the IMU and calculated true angle.

Table 1: Slope of IMU reading with pendulum link at rest immediately after impact.

<table>
<thead>
<tr>
<th>Mean encoder reading (degrees)</th>
<th>Pendulum angle (degrees)</th>
<th>Max. IMU error (degrees)</th>
<th>IMU Slope (degrees/second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.88</td>
<td>6.05</td>
<td>12.46</td>
<td>-0.7884</td>
</tr>
<tr>
<td>6.71</td>
<td>5.88</td>
<td>10.99</td>
<td>-0.8058</td>
</tr>
<tr>
<td>6.65</td>
<td>5.82</td>
<td>11.77</td>
<td>-1.0810</td>
</tr>
<tr>
<td>7.03</td>
<td>6.20</td>
<td>9.66</td>
<td>-0.8766</td>
</tr>
</tbody>
</table>

Figure 4.9 shows the results for a similar test in which the pendulum was released from 90 degrees and struck an object at zero degrees which was immediately removed. Here, the encoder once again demonstrates the (presumably) correct dynamic behavior, showing small oscillations about zero degrees. The IMU initially records oscillations about roughly 20 degrees; five seconds after the impact it reports oscillation around roughly 15 degrees. Repeated tests demonstrated similar results. This admittedly crudely determined slope of -1 degree/second agrees well with the resting slope of the IMU angle with time as calculated above and suggests that the IMU angle error due to an impact is indifferent to static or dynamic behavior.
Figure 4.9: Graph of angle in degrees vs. time for both encoder (shown in red) and IMU (shown in blue). Pendulum link holding IMU was released from a 90 degree starting angle, struck an object at 0 degrees and was immediately released.

4.2 Balance Simulation

All states measured

The results of the noise characterization indicated that the IMU noise was insignificant compared to the overall error in the angle reading. The error in the IMU had a total range of 1.848 degrees; this error was modeled as measurement noise in the balance simulation with standard deviation of $\sigma = 1.848/6 = .308$ degrees. Calculation of the standard deviation assumes a normal distribution with a total range roughly equal to $6\sigma$. The center of mass position vs. time without the use of a Kalman filter with true states generated from a linear dynamics simulation is shown in Figure 4.10. Use of a Kalman filter to estimate the states results in the graphs shown in Figure 4.11, where again the pendulum dynamics are assumed to be linear. Both of these simulations assume that the CM velocity can be measured and has error on the order of that measured in the IMU for position. Figure 4.11 (a) shows the actual and measured center of mass position vs. time while (b) compares the Kalman estimate...
of the center of mass vs. time with the actual center of mass position. Figure 4.12 shows the actual CM position, measurements of CM position, and Kalman estimate of the CM position when a nonlinear dynamic simulation generates the actual states.

A comparison between figures 4.10 and 4.11 reveals that Kalman filtering can significantly improve control of the inverted pendulum. The RMS error of the center of mass position was calculated from a time of 5 seconds to 100 seconds for the simulations with and without Kalman filtering. The five-second delay in evaluating the RMS difference allowed the system time to settle from the initial displacement to a steady-state behavior. Averages of the RMS values for 10 trials of each simulation are shown in Table 2. Unsurprisingly, when linear dynamics are used in the simulation to generate the ‘true’ states making the linear Kalman filter model perfectly accurate, the RMS error is small compared to a simulation run with states generated from nonlinear dynamics. Use of Kalman filtering with the IMU reduces the RMS error by a factor of 15 with a linear dynamic simulation and by a factor of 4.33 with a nonlinear dynamic simulation. Although the Kalman filter is not necessary to control
(a) Measurement with noise corresponding to IMU shown in black; actual center of mass position shown in red.

(b) Kalman filter estimate of center of mass position shown in black; actual position shown in red.

Figure 4.11: Center of mass x-position vs. time with an initial displacement of 10 cm, generated from linearized pendulum model with all states measured.
(a) Measurement with noise corresponding to IMU shown in black; actual center of mass position shown in red.

(b) Kalman filter estimate of center of mass position shown in black; actual position shown in red.

Figure 4.12: Center of mass x-position vs. time with an initial displacement of 10 cm, generated from nonlinearized pendulum model with all states measured.
balance with the sensor error in the IMU when all three states are measured, it can significantly improve upon the error between the desired and actual center of mass position.

Table 2: RMS error of actual CM position compared to desired CM position. Results given are averages from 10 trials. Simulation length was 100 seconds; RMS error was calculated after an initial 5-second period during which the system reacted to initial conditions.

<table>
<thead>
<tr>
<th></th>
<th>Linear dynamics (meters)</th>
<th>Nonlinear dynamics (meters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Kalman filtering</td>
<td>5.66 \cdot 10^{-3}</td>
<td>5.71 \cdot 10^{-3}</td>
</tr>
<tr>
<td>With Kalman filtering</td>
<td>3.75 \cdot 10^{-4}</td>
<td>1.32 \cdot 10^{-3}</td>
</tr>
</tbody>
</table>

The results of the noise characterization indicate that an impact force applied to the pendulum link can cause significant systematic error in the IMU angle reading. Based on testing, a maximum error of 10 degrees is assumed when the IMU experiences impact. In the simulation, the +10 degree error was translated into a center of mass position error of +0.1736 m. All measurements were assumed to have noise on the same order as the previous simulations; however, the CM position measurement had an additional constant error of .1736 m to represent an impact. Adjusting the measurement of the variance noise input to the Kalman filter for the center of mass position to reflect noise on the order of 10 degrees and running the simulation with nonlinear dynamics results in the graphs shown in Figure 4.13. To reiterate, a large noise variance was reported to the Kalman filter in order to decrease confidence in the measurements which are now known to be inaccurate after an impact. The CM measurements themselves were not then generated as random Gaussian noise on the order of 10 degrees, but rather as noise associated with static error plus an additional constant, positive position offset. It should be noted that this simulation assumes a constant error offset, while the results of the noise and error characterization demonstrate that the error decreases with time at a rate on the order of one degree per second. The simulation could be improved in future iterations by modeling the error as decreasing with the observed slope rather than a constant in
(a) Measurement with noise corresponding to IMU shown in black; actual center of mass position shown in red.

(b) Kalman filter estimate of center of mass position shown in black; actual position shown in red.

Figure 4.13: Center of mass x-position vs. time with an initial displacement of 10 cm, generated from nonlinearized pendulum model with impact force.
time.

From Figure 4.13, it can be seen that by lowering confidence in the measured values appropriately (thus having the Kalman filter place more emphasis on the model) balance may be achieved about a center of mass position of zero in spite of large differences in the measured and actual positions. This highlights some important issues for future work when considering the effect of impact forces on the IMU measurements: that either the noise parameters input to the Kalman filter must be corrected to reflect lower confidence in the measurements for a period of time after an impact, or the error must be understood well enough to be modeled effectively as a function of time and force. Additionally, if confidence in the measurements is lowered, the model used with the Kalman filter must be sufficiently accurate to compensate for poor measurements. These challenges with modeling impact force effects on the IMU could be the basis for further study and exploration.

No velocity

In previous simulations, it is assumed that all three states (center of mass position and velocity, zero moment point position) may be measured or calculated simply from existing measurements. The zero moment point position is determined from sensors associated with the tracks of the BEAR mobility base, the specifics of which are beyond the scope of this study. For the case of a linearized inverted pendulum, the center of mass position and velocity in the x-direction can be measured directly based on the approximations $x \approx l \sin \theta \approx l \theta$ and $\dot{x} = l \dot{\theta} \cos \theta \approx l \ddot{\theta}$. Even in the nonlinear case, the pendulum kinematics are simple enough that the linear position and velocity of the center of mass may be calculated with relative ease from the orientation and angular velocity measured by the IMU, as demonstrated in the Methods section 3.3. However, it is possible to imagine that calculating certain states, such as the center of mass velocity, could be significantly more difficult in the case of a more complicated multi-degree of freedom robot. When calculations to determine the states depend on several different measurements from IMUs located on various pieces of the robot, especially in the case of the velocity calculations where measurements
Figure 4.14: Center of mass x-position vs. time with an initial displacement of 10 cm, generated from linearized pendulum model without velocity measurements.

(a) Measurement with noise corresponding to IMU shown in black; actual center of mass position shown in red.

(b) Kalman filter estimate of center of mass position shown in black; actual position shown in red.
(a) Measurement with noise corresponding to IMU shown in black; actual center of mass position shown in red.

(b) Kalman filter estimate of center of mass position shown in black; actual position shown in red.

Figure 4.15: Center of mass x-position vs. time with an initial displacement of 10 cm, generated from nonlinearized pendulum model without velocity measurements.
are multiplied, errors in the measurements may propagate and yield results that are far different from the true states. Additionally, some IMUs only report orientation and do not offer information about angular velocity. In these cases, the ability of the Kalman filter to generate estimates for all states when not all states are measurable is important. Figures 4.14 and 4.15 show the actual CM position and the Kalman estimate of CM position with states generated from linear and nonlinear dynamics, respectively, without a CM velocity measurement. Measurement noise for the CM and ZMP positions are generated with $\sigma = .308$ degrees as is consistent with the error from the static IMU testing. A simulation that did not use Kalman filtering to estimate the unmeasurable velocity but rather differentiated the noisy measurements proved to be unstable for the same parameters.

Figures 4.14 and 4.15 demonstrate that the Kalman estimate of the CM position loses accuracy when the velocity is not measured and calculated only based on the linear model. The RMS error in the actual CM position compared to the desired position of zero was found to be $7.10 \times 10^{-3}$ m for both the linear and nonlinear dynamics generation of the true states. The fact that Kalman filtering can be used to estimate unmeasurable states with enough accuracy to control a system illustrates the usefulness of Kalman filtering techniques with complicated robots. A similar analysis of the capabilities of Kalman filtering with unmeasured states could be carried out on more complicated models of the BEAR for a more rigorous future study.

5 Conclusion

Testing of the IMU revealed that the sensor is significantly more precise than accurate. The overall error in static orientation sensing spanned a range of approximately 2 degrees; the standard deviation of noise in the IMU during static testing proved to be roughly two orders of magnitude smaller. Overall, the sensor noise was deemed insignificant in comparison to systematic errors. Impact testing revealed maximum errors in the IMU on the order of 10 degrees after the pendulum link holding the IMU was struck. The IMU error then decreased with a slope of
approximately -1 degree per second regardless of whether the IMU was oscillating or held static. The information found regarding the static and impact error in the IMU was used in a balance simulation to evaluate the necessity of Kalman filtering with the IMU.

When all states of the inverted pendulum model were measurable, the Kalman filter was not necessary to control the center of mass position sufficiently to maintain balance. However, the use of Kalman filtering was found to decrease the root mean squared error of the CM position by more than a factor of four compared to a simulation using unfiltered measurements. When not all states are measurable, the Kalman filter can increase the system stability compared to other methods of obtaining the unmeasurable states, such as differentiating or integrating sensor data from the measurable states. Kalman filtering techniques can also improve balance of the inverted pendulum model in response to impact forces when noise parameters are adjusted to lower confidence in the faulty measurements. The ability to sense impact, change the noise parameters to the Kalman filter, or model the error accurately upon impact remain subjects for further study. Overall, based on this study Kalman filtering is not strictly necessary when the MicroStrain 3DM-GX1 IMU is used, but can provide significant benefits without the inherent time delays associated with low-pass filtering.

6 References


[5] Photograph from Matlab Kalman filtering help file


7 Appendix A: Matlab code

7.1 Kalman filter simulation

%Constants

dt = 0.01;
control_dt = 0.1;
m = 100;
kz = 1000;

Q = 1e-9; % process variance
R = (1.848/6*pi/180)^2; % measurement variance based on testing

%Making the Plant

A = [1, dt, 0; (dt*kz/m), 1, (-kz/m*dt); 0, 0, 1];
B = [0; 0; dt];
C = eye(3);

Plant = ss(A, B, C, 0, -1, 'inputname', {'u'}, 'outputname', {'x_cm', 'x_cm_dot', 'x_zmp'});

%Making the Kalman Filter

B2 = [0, 0; 0, 0; dt, dt];
Plant2 = ss(A, B2, C, 0, -1, 'inputname', {'u', 'w'}, 'outputname', {'x_cm', 'x_cm_dot', 'x_zmp'});

Q2 = Q;
R2 = eye(3) * R;

[kalmf,L,P,M] = kalman(Plant2, Q2, R2);

%Control Constants

kv = 5;
kp = 8;
k1 = 10;

%Simulation initial conditions

x_dcm = 0;
x = [.1, 0, 0];
x_est = x;
y = x;
y_est = x_est;
yv = y + (rand(1,3) - 0.5) * sqrt(R);

u_kalm = [0, x];
xs = size(x);

%Initialize accumulation arrays

x_acc = [x];
y_acc = [y];
yv_acc = [yv];
x_est_acc = [x_est];
y_est_acc = [y_est, y_est];
u_acc = [0];
u_noise_acc = [0];
t = 0: dt: control_dt;
u = zeros(size(t));

for control_inc = 0:200

    x_last = x(xs(1),:);
y_last = yv(xs(1),:);
x_est_last = x_est(xs(1),:);

    u_last = u(end);

    u = ones(size(u)) * ComputeU(x_est_last, kp, kv, x_dcm, m, kz, k1);
    u_noise = u + (rand(1,length(u)) - 0.5) * sqrt(Q);

    [y,x] = pendsim(u, u_last, x_last, dt);

    %since x grows, update the size
    xs = size(x);

    y_acc = [y_acc; y(2:end, :)];
x_acc = [x_acc; x(2:end, :)];
% Outputs with measurement noise
yv = y + (randn(num_rows,3)) * sqrt(R);
yv_acc = [yv_acc; yv(2:xs(1), :)];

u_noise_t = u_noise';
u_t = u';
u_noise_acc = [u_noise_acc; u_noise_t(2:xs(1), :)];
u_acc = [u_acc; u_t(2:xs(1), :)];

% Control input and measured output over the last control interval
u_kalm = [u_t, yv];
[y_est, t_est, x_est] = lsim(kalf, u_kalm, t, x_est_last);

% Kalman estimates of states and outputs
y_est_acc = [y_est_acc; y_est(2:xs(1), :)];
x_est_acc = [x_est_acc; x_est(2:xs(1), :)];

end

7.2 Pendulum dynamic simulation

function [x,y,theta] = pendsim(u, u_last, x_last, dt)

% Constants
l = 1;
g = 9.806;
m = 100;
cm_pos = 1;
cm_vel = 2;
zmp_pos = 3;
J = m*l^2;

% Capture the instantaneous change in velocity from u_last to u in a single vector.
x_zmp_d = u;
x_zmp_d(1) = u_last;

% Differentiate x_zmp to find the ZMP acceleration
x_zmp_dd = [diff(x_zmp_d/dt) 0];
%Force based on acceleration
F_x = x_zmp_dd*m;

%Initialize vectors and set initial values
theta = zeros(length(u),1);
theta_d = zeros(length(u),1);
theta(1) = acos((x-last(cm_pos)-x_last(zmppos))/l);
theta_d(1) = (x_last(cm_vel)-u_last)/(-l*sin(theta(1)));

%Run simulation
for i = 1:length(u)-1
    T_grav = -m*g*l*cos(theta(i));
    T_force = F_x(i)*l*sin(theta(i));
    theta_dd = 1/J*(T_grav + T_force);
    theta_d(i+1) = theta_d(i) + theta_dd*dt;
    theta(i+1) = theta(i) + theta_d(i+1)*dt;
end

%x_zmp by integrating command u
x_zmp = cumtrapz(u')*dt + x_last(zmp_pos);

%Find x_cm from x_zmp and angle
x_cm = x_zmp + l*cos(theta);

%Find cm velocity from other known values
x_cm_d = [u_last; u(2:end)'] - l*theta_d.*sin(theta);

%Return states and outputs
x = [x_cm, x_cm_d, x_zmp];
y = x;

7.3 Control command computation

function [u] = ComputeU(x_est, kp, kv, x_des_cm, mass, Kz, K1)
    x_des_cm_dd = kp * (x_des_cm - x_est(1)) - kv * x_est(2);
    x_des_zmp = -(mass/Kz) * x_des_cm_dd + x_est(1);
    u = K1 * (x_des_zmp - x_est(3));