

Modeling, Simulation and Fabrication of a Balancing Robot

Ye Ding¹, Joshua Gafford¹, Mie Kunio²

¹Harvard University, ²Massachusetts Institute of Technology

1 Introduction

A balancing robot is a common demonstration of controls in a dynamic system. Due to the inherent instability of the equilibrium point, appropriate controllability and observability measures must be undertaken to stabilize the system about the desired equilibrium point. We will investigate this system by developing both an analytical and experimental model of the system and analyze the dynamic characteristics. In the process, we hope to answer questions regarding observability and controllability of the system. In addition, we will derive and implement a discrete-time Kalman filter to handle the inherent noise of the system and improve response.

The report will begin with a discussion of the hardware design of the balancing robot, including any rationale behind component selection. We will then give a derivation of the equations of motion using a Lagrangian approach, and investigate the effect of center-of-mass position on the closed-loop dynamics of the system. In addition, we will discuss controllability and observability of the system and derive a full-state feedback control based on the Linear Quadratic Regulator (LQR) method. Finally we will derive a two-state discrete Kalman filter for smoothing out sensor/process noise, and simulate the entire system in Simulink. We will close with some discussion on the actual response of the physical balancing robot, outfitted with full-state control and Kalman filter.

2 Hardware Design of the Balancing Robot

(Ye Ding)

The balancing robot we built is basically a simple inverted pendulum on two wheels. This is one of the most widely used examples in control classes. For this project, not only did we simulate the system in MATLAB/Simulink; we also built and tested the results of our analyses.

2.1 System Estimation

The free body diagram of the balancing robot is shown in Figure 1. The robot consists of two parts: the trunk and the wheel. The center of the mass will be higher than the height of the motor shaft because we want to investigate an inherently unstable system. A small scale robot was built with the following parameters in mind: the mass (m) is around 0.5kg, the height of the COM (l) is around 0.08m, and the maximum recovery angle (θ) would be 5° for the robot. We chose such a small recovery angle since the dynamics of the system will be linearized to calculate the control gains of the system.

With the estimation of the mass m , height of center of the mass l and maximum recovery angle θ , the following equations could be set up for the maximum torque t_{max} produced by the motor.

$$t_{max} = mg * l * \sin\theta_{max} = 5 * 9.8 * 0.08 * \sin(5^\circ) = 0.0342Nm \quad (1)$$

There are two motors on the robot, and as such, the maximum torque produced by the motor should be 0.0171Nm (half of the maximum torque required by the system).

There is no specific requirement on the speed, so a normal RC toy car speed is set as the target speed for the system, which is around 400rpm. Later on, the speed will be decided by the price and lead time of the gearbox.

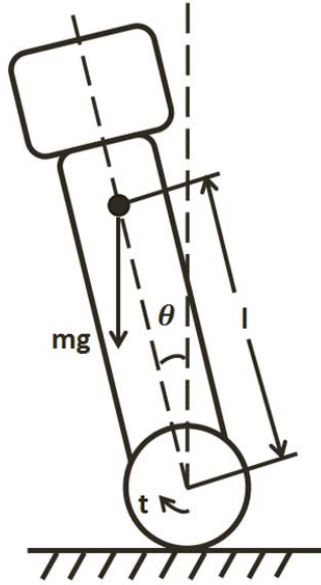


Figure 1: Free body diagram of the balancing robot

2.2 Components Selection

2.2.1 Motor

With the target torque 0.171Nm and target speed 400rpm, the power rate (P) of the motor is calculated as follows:

$$P = t * w = 0.0171 * \frac{400}{60} * 2 * \pi = 0.759w \quad (2)$$











This is the power rate we need without considering the efficiency of the power transmission chain, namely, the gearbox. Normally, a planetary gearbox or spur gearbox would have an efficiency of 80% for the 1-2 stages, which means the gear ratio is around 10:1.

A modified power rate of the motor is obtained by accounting for the efficiency of the gearbox into the calculation.

$$P_m = \frac{P}{80\%} = \frac{0.759}{0.8} = 0.948w \quad (3)$$

Several DC brushed Maxon motors in the stock program were considered based on the calculated motor power rate shown in Table 1.

Table 1: Motor Selections

Motor Name	Power Rate (w)	Output Speed (rpm)	Output Torque(mNm)	Encoder	Total Price	Motor	Gearbox
RE 13	1.5	422.	15.47	No	122.7		
RE 13	1.5	422	15.47	No	126.4		
A-Max 16	1.2	496	16.06878	Yes	128.7		
A-Max 19	1.5	450	17.4474	Yes	173.1		
RE-Max 13	1.2	360	15.589	No	98.3		

It can be seen that the combination of A-max 16 with an encoder has the best tradeoff in terms of price and performance. As such, this combination was selected for the robot system.

2.2.2 Motor Controller

A motor controller is needed, since the PCB board of the micro-processor cannot provide enough current for the DC motor. Based on the stall current of 0.72A from the A-Max 16 datasheet, a small two-channel motor controller board was selected as shown in Figure 2.

**Figure 2:** Qik dual serial motor controller

This qik dual serial motor controller allows any microcontroller or computer with a serial port to easily drive two small, brushed DC motors with full directional and speed control. It provides 8-bit PWM speed control via an advanced, two-way serial protocol that features automatic baud rate detection up to 38.4 kbps and optional CRC error checking. The continuous output current per channel is 1A, and the peak current reaches up to 3A.

2.2.3 Sensor

For full state feedback, we need to be able to measure both the rotation of the wheel and the leaning angle of the body. Thus we need an encoder on the motor and an inertial sensor which could measure the body orientation.

The encoder shown in Figure 3 comes with the motor and has a resolution of 512 counts per rotation. It's a two channel encoder, but only one is going to be used, because the speed of our microprocessor can't handle the update rate.



Figure 3: Maxon encoder MR, Type M

By comparing the price and performance among several sensors, we selected a 6 DOF IMU. The selected, IMU shown in Figure 4, is a compact motion sensor with a 3-axis gyroscope and a 3-axis accelerometer. The tri-axis gyro has a sensitivity up to 131 LSBs/dps and a full-scale range of ± 250 , ± 500 , ± 1000 , and ± 2000 dps. And the Tri-Axis accelerometer has a programmable full scale range of $\pm 2g$, $\pm 4g$, $\pm 8g$ and $\pm 16g$. Then, it provides us the possibilities of having a very accurate leaning angle reading for input to the Kalman filter which we will discuss later. The communication protocol of the IMU is I2C.



Figure 4: Triple axis accelerometer & gyro

2.2.4 Micro Processor

Arduino microprocessor, shown in Figure 5, is the most widely used micro-processor for educational projects due to its user-friendliness. It is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. We selected the Arduino Mega because I/Os are needed, not only for sensor input/motor output, but also for a LCD screen and buttons would be integrated in the system for the ease of tuning the control parameters.



Figure 5: Arduino Mega 1280

2.2.5 LCD, LED, Button, Switch, Battery

As mentioned above, the LCD1602 and buttons are integrated into the system for the purpose showing and tuning the control parameters. A switch has been implemented to turn the system on or off. LED's comprise the robot's eyes and receive the same PWM signal as a motor to give a visual indication of the struggle to balance. These components are shown in Figure 6.



Figure 6: LCD, LED, button, switch, battery (left to right)

2.3 CAD Design

2.3.1 Version 1.0

Based on all the components selected, an initial prototype was fabricated. 3D printing was selected as the fabrication method for the robot shell, and as such, alignment/interior fixture features were designed in the shell. A CAD rendering and actual photograph of our robot (Mr. Struggles, Version 1.0) is shown in Figure 7. The battery, microprocessor, and LCD are all in the head, the motor controller is in the trunk, and the IMU is at the bottom between the two motors such that the signal from the accelerometer is the pure rotation of the robot and not contaminated by tilting effects had it been placed somewhere else.

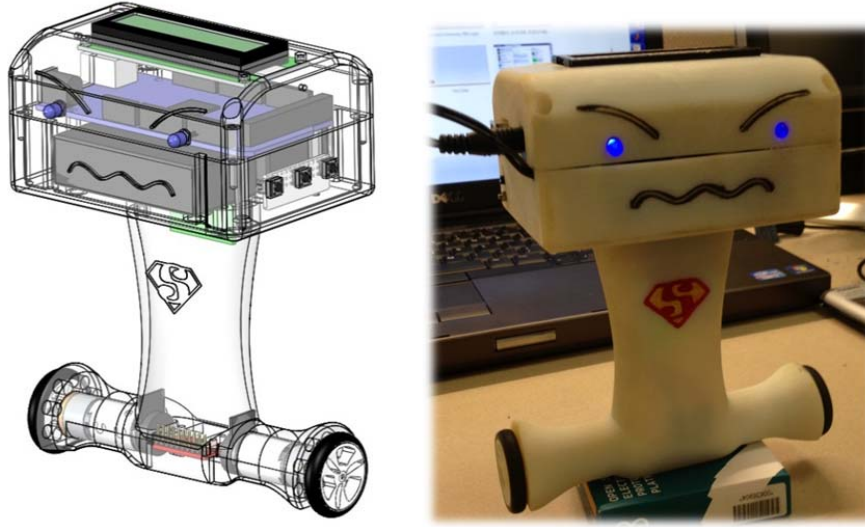


Figure 7: Mr. Struggles version 1.0

2.3.2 Version 1.1

After building the dynamic model (to be discussed in Section 4) and tuning the robot, we found that it was extremely difficult to balance the robot even if the COM is within the estimated range to satisfy the small-angle approximation. It is remarkably easy to go out of the stable region given external disturbance. Thus we decided to relocate some of the weight down from the head. The battery was moved to the back of the robot as shown in Figure 8 to lower the COM.



Figure 8: Mr. Struggles version 1.1

2.4 Robot Specs

A functional robot was designed and built. The following Table 1 gives all the specs of the robot.

Table 2: Mr. Stuggles Version 1.1 specs

Robot Weight	513g
Robot Inertia	13067.6gcm ²
Terminal Resistance	8.3Ω
Terminal Inductance	0.306mH
Torque Constant	5.57mNm/A
Speed Constant	1720rpm/V
Rotor Inertia	0.862gcm ²
Gearhead Ratio	9.1:1
Gearhead Efficiency	0.81
Wheel Dia	31.5mm
Mass center	63.6mm

The dynamic modeling of the robot is all based on the specs.

3 Dynamic Model of the Robotic System

(Mie Kunio)

3.1 Definition of Coordinates

This robotic system is similar to “the inverted pendulum on moving cart” problem [1]. Similarly to the problem of the pendulum, this robot system can be separated into two parts: “wheel part” and “body part” (Figure 9). The main difference between this system and the inverted pendulum is the inertia and the position of the center of mass (COM) of the body part, as we are striving for a more detailed model of the system rather than approximating it as a point-mass. For this system, we need to calculate the inertia and the position of the COM based on the mass distribution of the body part, which is determined by the hardware design.



Figure 9: CAD model of the robot and schematic image of the robot

To derive dynamic equations of this system, we defined the coordinates as shown in Figure 10. In this system, we assume that the robot moves horizontally without any slipping between the ground and the wheel.

x	horizontal position of the center of the wheel relative to a defined origin
x_c	horizontal position of the COM of the body part relative to a defined origin
z_c	vertical position of the COM of the body part from the ground
φ	clockwise rotational angle of the wheel from the horizontal axis at $t=0$
θ	clockwise rotational angle of the body part from upright position

Other parameters which are used in this figure are summarized below.

m	mass of the body part (513.3 g)
m_w	mass of the wheel part (7.2 g)
R	radius of the wheel (16.0 mm)
L	length between the COM and the center of the wheel
τ_0	applied torque
I	inertia of the body part
I_w	inertia of the wheel part (779.2 g mm ²)

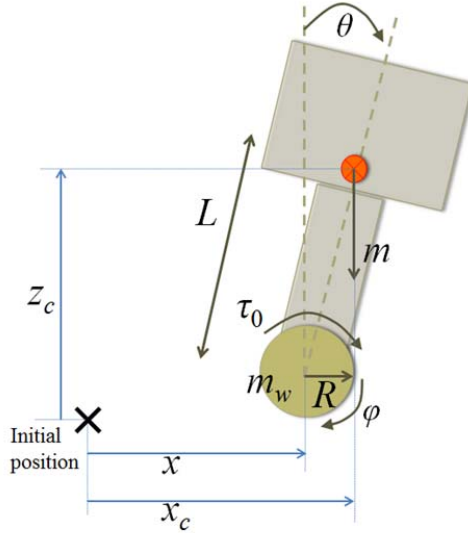


Figure 10: Schematic of balancing robot with relevant parameters

3.2 Derivation of the Dynamic Equations of Motion

We used the Lagrange's method to derive the dynamic equations for this system. We can write $x, x_c, z_c, \dot{x}, \dot{x}_c, \dot{z}_c$ as follows:

$$x = R\varphi \quad (4) \quad \dot{x} = R\dot{\varphi} \quad (5)$$

$$x_c = R\varphi + L\sin\theta \quad (6) \quad \dot{x}_c = R\dot{\varphi} + L\dot{\theta}\cos\theta \quad (7)$$

$$z_c = R + L\cos\theta \quad (8) \quad \dot{z}_c = -L\dot{\theta}\sin\theta \quad (9)$$

Continuing, we can write the potential energy E_p (zero is defined as the potential energy at the upright position) and the kinetic energy E_k as follows:

$$E_p = mg(R + L \cos \theta) - mg(R + L) = mgL(\cos \theta - 1) \quad (10)$$

$$E_k = \frac{1}{2} m_w \dot{x}^2 + \frac{1}{2} I_w \dot{\phi}^2 + \frac{1}{2} m \dot{x}_c^2 + \frac{1}{2} m \dot{z}_c^2 + \frac{1}{2} I \dot{\theta}^2 \quad (11)$$

$$= \frac{1}{2} (I_w + m_w R^2 + m R^2) \dot{\phi}^2 + m R L \cos \theta \dot{\phi} \dot{\theta} + \frac{1}{2} (I + m L^2) \dot{\theta}^2 \quad (12)$$

3.2.1 Derivation of the Lagrangian

Lagrangian \mathcal{L} can be written as the difference between the kinetic energy E_k and the potential energy E_p .

$$\mathcal{L} = E_k - E_p \quad (13)$$

$$= \frac{1}{2} (I_w + m_w R^2 + m R^2) \dot{\phi}^2 + m R L \cos \theta \dot{\phi} \dot{\theta} + \frac{1}{2} (I + m L^2) \dot{\theta}^2 - mgL(\cos \theta - 1) \quad (14)$$

Therefore, the dynamic equations for ϕ -coordinate and θ -coordinate can be derived as follows:

(i) ϕ -coordinate

$$\frac{\partial \mathcal{L}}{\partial \dot{\phi}} = (I_w + m_w R^2 + m R^2) \dot{\phi} + m R L \dot{\theta} \cos \theta \quad (15)$$

$$\frac{\partial \mathcal{L}}{\partial \phi} = 0 \quad (16)$$

$$\therefore \frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{\phi}} \right) - \frac{\partial \mathcal{L}}{\partial \phi} = (I_w + m_w R^2 + m R^2) \ddot{\phi} + m R L \cos \theta \ddot{\theta} - m R L \sin \theta \dot{\theta}^2 = \mu \quad (17)$$

(ii) θ -coordinate

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}} = m R L \cos \theta \dot{\phi} + (I + m L^2) \dot{\theta} \quad (18)$$

$$\frac{\partial \mathcal{L}}{\partial \theta} = -m R L \sin \theta \dot{\phi} + mgL \sin \theta \quad (19)$$

$$\therefore \frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}} \right) - \frac{\partial \mathcal{L}}{\partial \theta} = (I + m L^2) \ddot{\theta} + m R L \cos \theta \ddot{\phi} - m R L \sin \theta = \chi \quad (20)$$

where μ and χ are generalized forces (torques) for each coordinate.

Then, we can rewrite these non-linear dynamic equations in a second-order matrix style as follows:

$$\begin{bmatrix} I_w + (m_w + m)R^2 & mRL\cos\theta \\ mRL\cos\theta & I + mL^2 \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & -mRL\sin\theta \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ -mgL\sin\theta \end{bmatrix} = \begin{bmatrix} \mu \\ \chi \end{bmatrix} \quad (21)$$

For the next step, we need to express the generalized torques by our known parameters. The generalized torques are the differences between the torque actually applied to the system and the dissipated torque by friction. Defining the dissipated torque as τ_{hinge} (the torque dissipated due to friction in the axle), and τ_{floor} , dissipated between the ground and the wheel, we can write μ and χ by using the applied torque τ_0 , the rolling damping ratio β_γ , and the friction damping ratio β_m :

$$\mu = \tau_0 - \tau_{hinge} - \tau_{floor} = \tau_0 - \beta_m(\dot{\phi} - \dot{\theta}) - \beta_\gamma \dot{\phi} \quad (22)$$

$$\chi = -\tau_0 + \tau_{hinge} = -\tau_0 + \beta_m(\dot{\phi} - \dot{\theta}) \quad (23)$$

These relations can be expressed in matrix-form as follows:

$$\begin{bmatrix} \mu \\ \chi \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tau_0 - \begin{bmatrix} \beta_\gamma + \beta_m & -\beta_m \\ -\beta_m & \beta_m \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \end{bmatrix} \quad (24)$$

3.2.2 Linearization

In order to solve these equations of motion, we need to linearize the system. The control objective for this system is to stand vertically, so we can linearize the system within a small range of motion from the upright position (i.e., $\theta = 0$, $\dot{\phi} = 0$, $\dot{\theta} = 0$). Within this range, we can assume that $\cos\theta \cong 1$, $\sin\theta \cong \theta$. Therefore, we can rewrite the dynamic equations for this system as follows:

$$\begin{bmatrix} I_w + (m_w + m)R^2 & mRL \\ mRL & I + mL^2 \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} \beta_\gamma + \beta_m & -\beta_m \\ -\beta_m & \beta_m \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ -mgL \end{bmatrix} \theta = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tau_0 \quad (25)$$

To write the matrix in a compact style, we rewrite this equation as follows:

$$E \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \end{bmatrix} + F \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \end{bmatrix} + G\theta = H\tau_0 \quad (26)$$

3.3 Expression of the Position of COM and the Inertia for the Body

Moment arm L and inertia I are determined by the mass distribution of the body part as described in 3.1. We need to separate the body part into two parts (head and shaft) as shown in Figure 11. and approximate these section as homogeneous masses to approximate the total inertia of the system. All parameters needed to explain L and I are summarized below.

m_1	mass of the head
m_2	mass of the shaft
L_1	height of the head (40 mm)
L_2	height of the shaft (60 mm)

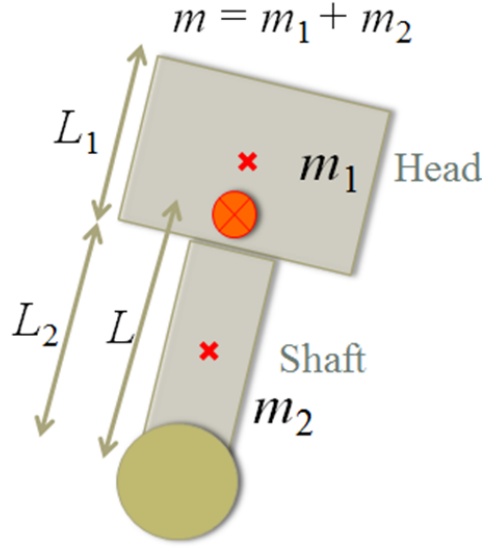


Figure 11: Parameters considered in the determination of the position of the COM and the inertia

We assume that the head can be considered as a point mass and that the shaft can be considered as a homogenous cylinder. Then, the position of the COM for each component is located at the center of each component. Therefore, the position of the COM for the body part (combination of the head and the shaft) can be expressed as the point which divides the distance of the COMs for the head and the shaft internally in the ratio of $m_1 : m_2$ from the COM for the head. As a result, L can be written as follows:

$$L = \frac{L_2}{2} + \frac{L_1 + L_2}{2} \frac{m_1}{m_1 + m_2} \quad (27)$$

Since the inertia for the body part, I , can be calculated as the sum of the inertias of the head and the shaft, I can be expressed as follows:

$$I = m_1 \left(\frac{L_1}{2} + L_2 \right)^2 + \frac{1}{12} m_2 L_2^2 \quad (28)$$

3.4 State-Space Representation of the Robotic System

From 3.2, we can derive the state-space representation of the robot system. In this model, we use $\varphi, \theta, \dot{\varphi}, \dot{\theta}$ as state variables. The input of this system is the applied torque τ_0 , and the outputs of this system are the distance which this robot moves horizontally (x) and the rotational angle of the body part (θ). In this system, we assume that there is no slipping between the ground and the wheel. Therefore, the state-space representation of this robot system can be written as follows:

$$\frac{d}{dt} \begin{bmatrix} \varphi \\ \theta \\ \dot{\varphi} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -E^{-1}G & & \\ 0 & & -E^{-1}F & \end{bmatrix} \begin{bmatrix} \varphi \\ \theta \\ \dot{\varphi} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -E^{-1}H \end{bmatrix} \tau_0 \quad (29)$$

$$\mathbf{y} = \begin{bmatrix} R & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \varphi \\ \theta \\ \dot{\varphi} \\ \dot{\theta} \end{bmatrix} \quad (30)$$

To express this representation more easily, we can rewrite it as follows:

$$\frac{d}{dt} \mathbf{x} = A\mathbf{x} + B\tau_0 \quad (31)$$

$$\mathbf{y} = C\mathbf{x} \quad (32)$$

4 Position of the COM and Controllability, Observability

(Mie Kunio)

The controllability and observability of the system can be checked by calculating the controllability matrix, $Cont$, and the observability matrix, O , for the system.

$$Cont = [B \quad AB \quad A^2B \quad A^3B] \quad (33)$$

$$O = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \end{bmatrix} \quad (34)$$

We calculated the controllability matrix $Cont$ and the observability matrix O for this robot system in MATLAB using the matrices A , B , and C which are described in 2.4. We found that this robot system is always completely controllable from input τ_0 and completely observable from output \mathbf{y} , no matter how much we change the position of the COM (i.e., change the masses of the head (m_1) and the shaft (m_2)) with the constraint that $m = m_1 + m_2 = 513.3$ g.

5 Designing the Full-State Feedback Controller

(Mie Kunio)

We need to know when this robot becomes difficult to be controlled in order to decide the mass distribution of the body part. For this purpose, we design the full-state feedback controller and compare the gains of the controller and the free response of the closed-loop system.

For designing the full-state feedback controller, we use the LQR method. Since we want to consider all state variables $\varphi, \theta, \dot{\varphi}, \dot{\theta}$ and the input τ_0 equally, we define the weighing matrices Q and R as follows:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (35)$$

$$R = 1 \quad (36)$$

By MATLAB, we design the full-state feedback controller. Table 3 shows the gains of the full-state feedback controller.

Table 3: Gains of the Full-State Feedback Controller

m_1 [g]	m_2 [g]	L [m]	K
513.3	0.0	0.0800	[1.0000 57.4808 -0.1385 7.7517]
413.3	100.0	0.0703	[1.0000 51.3278 -0.1741 6.6274]
313.3	200.0	0.0605	[1.0000 44.8904 -0.2099 5.4986]
213.3	300.0	0.0508	[1.0000 38.1008 -0.2435 4.3623]
113.3	400.0	0.0410	[1.0000 30.8321 -0.2671 3.2140]
0.0	500.0	0.0300	[1.0000 21.5065 -0.2366 1.8944]

As shown in Table 3, when the position of the COM goes down, the gains of the full-state feedback controller become smaller. This means that the robot becomes easier to control. This result is more obvious when we compare the free response of the closed-loop system (Figure 12). The initial condition is set as the robot inclines at 10 degrees from the upright position ($\mathbf{x}(0)=[0 \ 0.17 \ 0 \ 0]^T$).

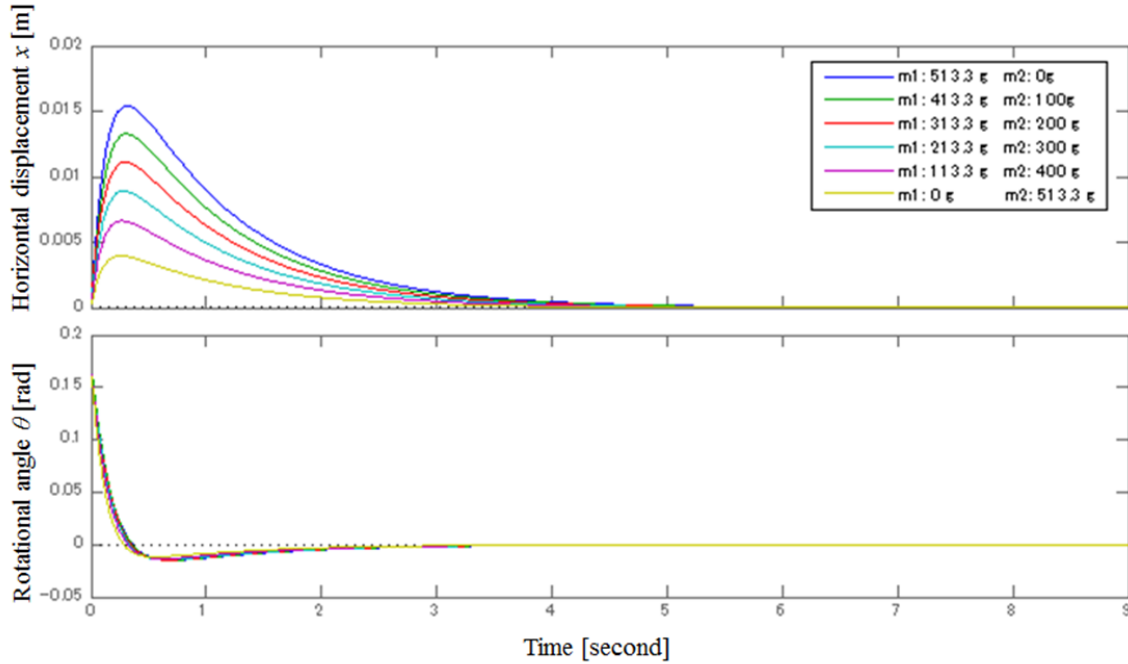


Figure 12: Comparison of the Free-Response of the closed-loop system

From Figure 12, we can see that when the position of the COM goes down, the maximum horizontal displacement decreases. Moreover, the initial moving speed of the robot (the slope of the $x(t)$ at $t=0$) decreases as the position of the COM goes down. These findings also indicate that the robot becomes easier to be controlled when the position of the COM goes down because it needs less energy to stand vertically if the horizontal displacement and the initial speed are small. In fact, this robot can not stand vertically when almost all mass is located at the head part. The reason is supposed that the requirement to control the robot exceeds the maximum performance of the robot (i.e., the maximum initial speed which the robot can perform). Therefore, we need to consider the maximum performance of the robot when we design the full-state feedback controller, even though the system is always completely controllable theoretically.

6 Kalman Filtering

(Joshua Gafford)

6.1 Overview and Statistical Motivation

In selecting an appropriate observer, we need to know some important details concerning the dynamics of the system, as well as the inherent noise in our sensors. Our measurement unit consists of accelerometer and gyro triads to measure motion in 6DOF, and both of these sensors are prone to noise. As such, it was decided early on that a Luenberger observer would be inadequate. As such, we decided to implement a Kalman filter to smooth the noise. A Kalman Filter is an algorithm which recursively computes estimates of observed states over time. Kalman filters are widely applicable due to their ability to handle noise inherent to both the process and the system of measurement and optimized for noise signatures with zero-mean Gaussian distributions. In order to further motivate the inclusion of a Kalman filter in our observer, we collected both static data and incremental angle data for both the gyro and the accelerometer on board the robot. Data was captured in both static (white noise) and dynamic (error noise) environments in order to compute measurement noise covariances of the system (Figure 13).

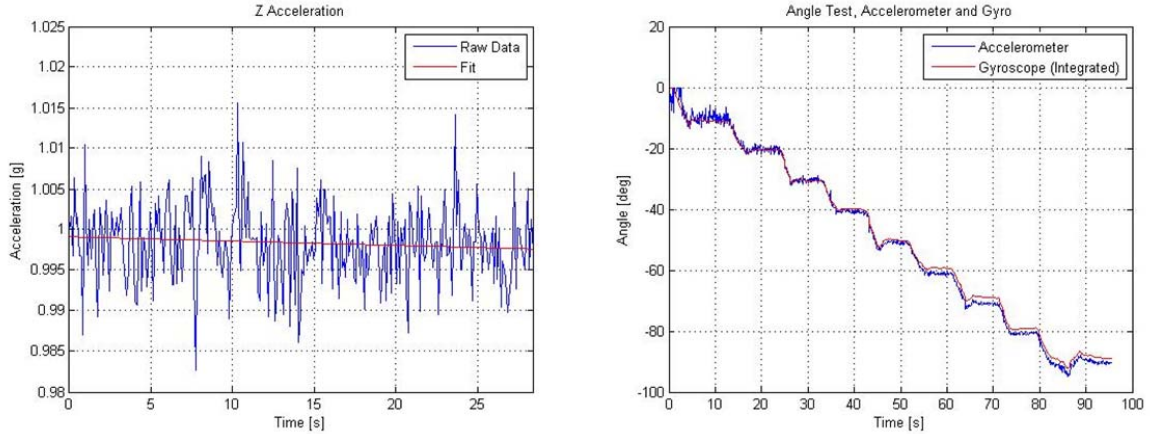
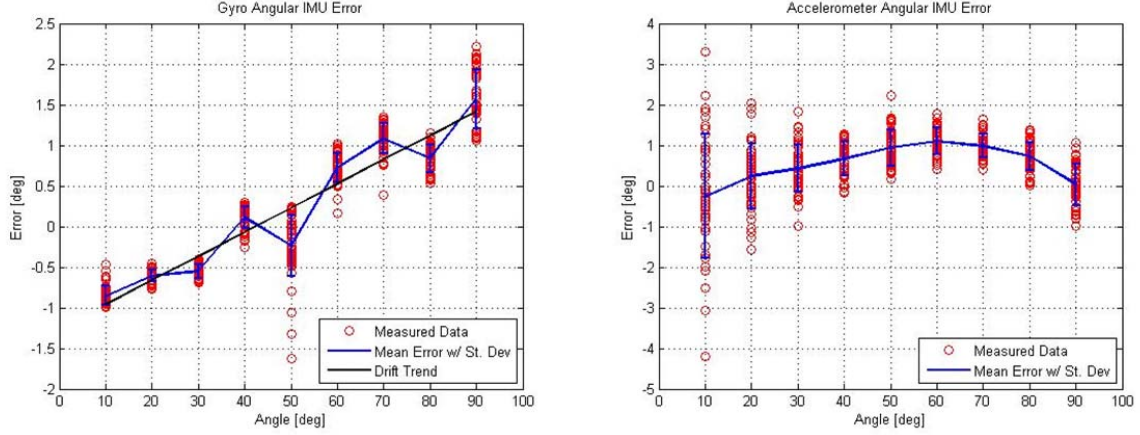


Figure 13: (left) Static Noise Characterization, (right) Known angle error characterization.

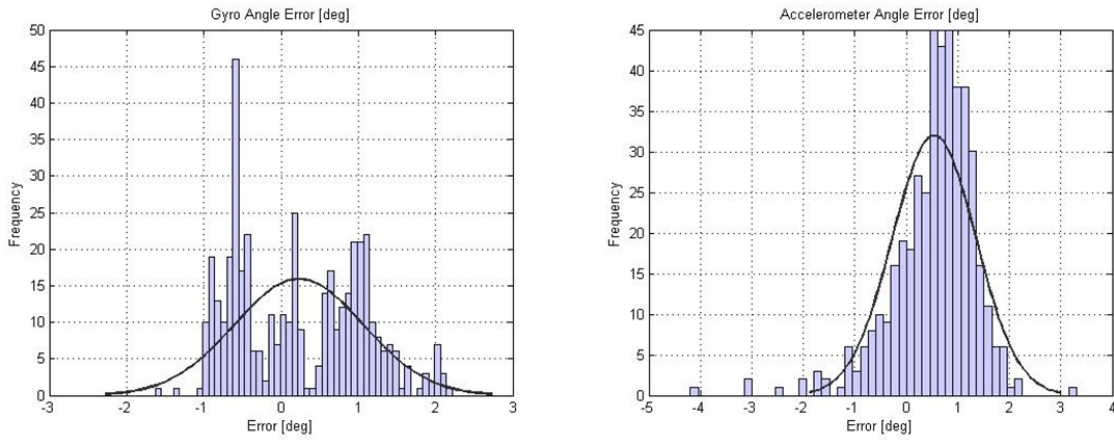
Given the results of the angular error test, it was quite evident that any white noise error was completely trumped by the overall estimate error of the sensors. Figure 14 shows computed error between the measured and actual angles recorded during the incremental angle tests. Our suspicions concerning potential sources of error have been confirmed, which are summarized in Table 4.

Table 4: Sensor characteristics and sensitivities

Sensor	Noise Immunity	Appreciable Drift	Response Time
Accelerometer	Poor	No	Slow
Gyroscope	Good	Yes	Fast

**Figure 14:** (left) Gyro error, mean and standard deviation, (right) Accelerometer error, mean and standard deviation

Error data for each sensor was fitted to Gaussian distributions, shown in Figure 15. Let $X \sim N(\mu = 0.540, \sigma^2 = 0.342)$ be a random variable given by accelerometer error, and $Y \sim N(\mu = 0.230, \sigma^2 = 0.035)$ be a random variable given by gyroscope error, as given by the normal fit (all units given in degrees as the base unit). Although it hasn't been presented here, angular error noise dwarfs random white noise in both the gyro and the accelerometer, so variances computed from error Gaussian fits were used in the filter model. An interesting observation is the non-zero mean in both cases; as the Kalman filter assume zero-mean Gaussian, this nonideality must be overlooked in the derivation of filter covariances. Thus we define $Q_\alpha = E(Y, Y) = Var(Y) = 0.035$ and $R = E(X, X) = Var(X) = 0.540$ (assuming $Cov(X, Y) = 0$). Note that the gyro has a zero-rate offset of about 2.87 deg/s so this must be corrected for in software. We will use these values for process and measurement noise covariances. We still need another covariance for bias error; since this is difficult to ascertain in practice, we will estimate Q_{bias} in simulation.

**Figure 15:** Angular error histograms (and Gaussian fits) for (left) gyroscope, and (right) accelerometer

6.2 Discrete-Time Kalman Filtering Algorithm

For the purposes of computational practicality, a finite-differencing-based two-state discrete-time Kalman filter based on that given in [2]. The implementation was kept simple since all processing would be done on-board an Arduino microprocessor with element-wise computational capabilities, so we wish to avoid complicated matrix operations. Let us define the following:

$\hat{\mathbf{x}}_k$	State Vector
$\dot{\theta}_k$	Input/Control Vector
\mathbf{z}_k	Observation Vector
$\hat{\mathbf{y}}_k$	Measurement Innovation
\mathbf{S}_k	Innovation Covariance
\mathbf{P}_k	Filter Covariance
\mathbf{K}_k	Kalman gains
\mathbf{B}	Input/Control Matrix
\mathbf{H}	Observation Matrix
\mathbf{Q}_k	Process Noise Covariance Matrix
\mathbf{R}	Measurement Noise Covariance Matrix

The measured state vector $\hat{\mathbf{x}}_k$ contains states $[\theta_\alpha \ \dot{\theta}_{bias}]$ used to represent the angle and bias errors, respectively. The filtering algorithm is given below. The multi-step approach to solving the algebraic Ricatti equation makes the algorithm simpler to implement in C for processing on-board the Arduino microprocessor.

$$\text{State Estimation} \quad \hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}\dot{\theta}_k \quad (37)$$

$$\text{Forecast Error Covariance} \quad \mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^T + \mathbf{Q}_k \quad (38)$$

$$\text{Innovation} \quad \hat{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1} \quad (39)$$

$$\text{Innovation Covariance} \quad \mathbf{S}_k = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R}_k \quad (40)$$

$$\text{Compute Kalman Gains} \quad \mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T\mathbf{S}_k^{-1} \quad (41)$$

$$\text{Update } a \text{ posteriori state} \quad \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\hat{\mathbf{y}}_k \quad (42)$$

$$\text{Update Covariance} \quad \mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k|k-1} \quad (43)$$

Gyro data are treated as external inputs to the filter rather than as measurements, so gyro measurement noise and bias enter the filter as process noise rather than as measurement noise [3]. Therefore, our measurement and process noise covariance matrices \mathbf{R}_k and \mathbf{Q}_k are defined as follows (based on noise covariances discussed in Section 6.1):

$$\mathbf{R}_k = 0.342 \quad (44)$$

$$\mathbf{Q}_k = \begin{bmatrix} Q_\alpha & 0 \\ 0 & Q_{bias} \end{bmatrix} = \begin{bmatrix} 0.035 & 0 \\ 0 & 0.0007 \end{bmatrix} \quad (45)$$

Where Q_{bias} was determined iteratively to minimize the root-mean-squared deviation (RMSD) between Kalman filtered angle and actual angle (since it is difficult to obtain this covariance empirically).

6.3 Simulink Implementation and Simulated Results

The dynamic model and control scheme derived in Section 3 was implemented in Simulink, along with the Kalman filter derived in the previous section. Noise with the same statistical properties as those found in our sensors was injected into the state observed by the sensors and passed to the simulated Kalman filter. The Simulink block diagram is shown in Figure 16, along with a screenshot of the dynamic simulation. Although the control gains were derived via linear approximation, nonlinear dynamics were implemented in Simulink for higher fidelity. Discrete time-steps were used for any integrations done via the measurement system, and continuous time-steps were used in the solution of the dynamic ODE's. As the motors weren't modeled explicitly, torque output from the full-state controller is passed through a first-order transfer function to slow down the dynamics. Note that the full-state feedback controller doesn't operate on an error term; the vector x_des in the model is an artifact from a previous PID implementation and is simply an empty set.

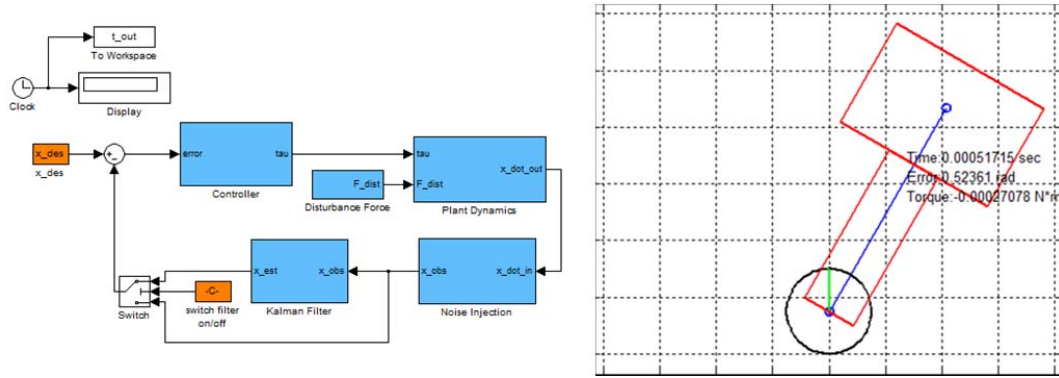


Figure 16: (left) Simulink implementation of dynamic model and Kalman filter, (right) dynamic simulation screenshot

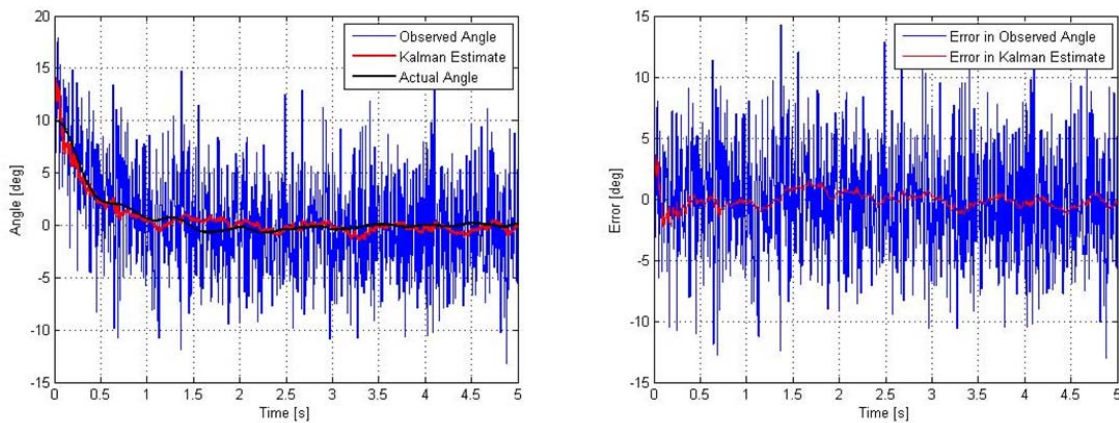


Figure 17: (left) Kalman estimate vs. observed and actual angle, (right) Error in observed angle and Kalman estimate

7 Actual Performance

(Joshua Gafford)

We outfitted Mr. Struggles with a Kalman filtering algorithm using the measurement and process noise covariances computed from accelerometer and gyro data. Results of a test run are given below. It is quite astounding how much of a difference the Kalman filter makes; due to the IMU's proximity to the DC motors, and given the accelerometer's vulnerability to noise, the robot would simply be inoperable without the filter. The drift of the gyro is clear in these results as well. It is also interesting to note that the window of equilibrium is very narrow for Mr. Struggles, on the order of ± 2 degrees. This was not captured adequately in simulation, possibly due to inadequate modeling of the motor dynamics and loading conditions.

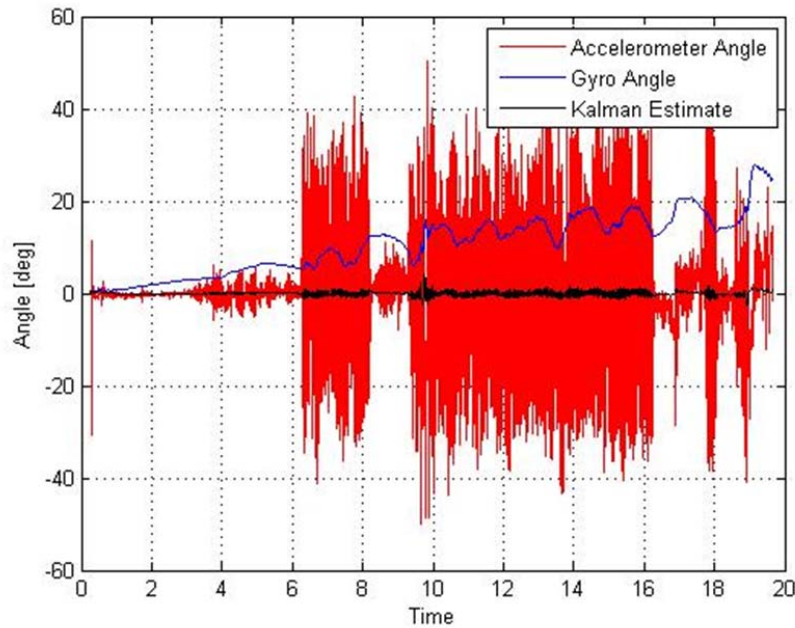


Figure 18: Data taken from Mr. Struggles during a balancing act

The influence of motor vibration on accelerometer noise was updated in the model and simulated results, including covariance convergence, are given below. The error covariance converges rather quickly, at least with respect to the dynamics of the system, even in light of the considerable noise in the measurement and process.

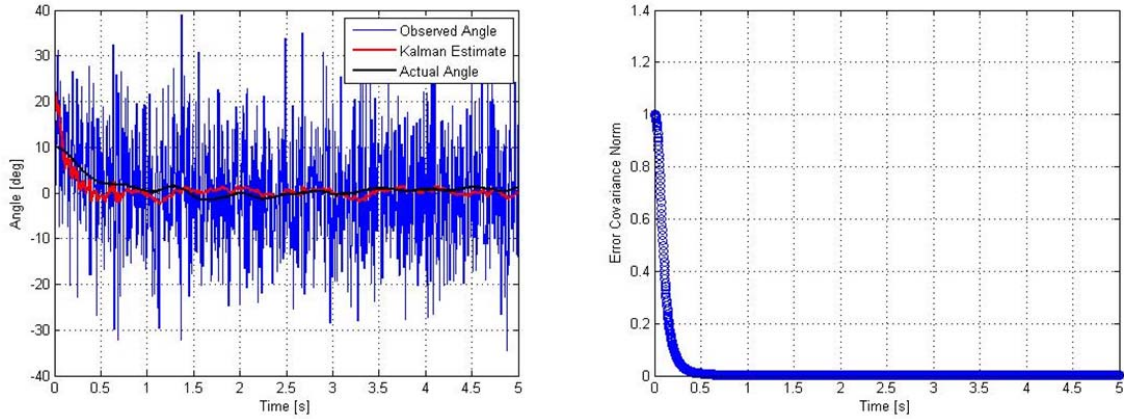


Figure 19: (left) Kalman filter results with updated accelerometer covariance based off of empirical results, (right) Covariance convergence

8 Conclusions

We have successfully modeled and demonstrated a balancing robot with full-state feedback and Kalman filtering. We generated results (both analytical and experimental) that demonstrate the merits of the Kalman filter, as the system would be high unstable without one due to the extreme incidence of parasitic noise within the system.

9 References

- [1] B. Friedland, "Control system design: an introduction to state-space methods", Dover Publications, Inc., 2005, pp.30-32
- [2] Terejanu, Gabriel. "Discrete Kalman Filter Tutorial." Department of Computer Science and Engineering, University at Buffalo.
- [3] Sabatini, "Kalman-Filter-Based Orientation Determination Using Inertial/Magnetic Sensors: Observability Analysis and Performance Evaluation" *Sensors*, 2011

Appendix A: MATLAB Reference Examples

A.1. Controllability, Observability, LQR design

(Mie Kunio)

```
%2.151 final project
%COM and controllability, observability, LQR design
%Date 2012/12/13
%Revised 2012/12/16

%Parameters
m=513.3*10^(-3); %body part mass [kg]
m2=0*10^(-3); %shaft part mass [kg], CAN CHANGABLE!!!!
m1=m-m2; %head part mass [kg], change by mass of shaft
mw=7.2*10^(-3); %wheel(*2) mass [kg]
L1=40*10^(-3); %height of head [m]
L2=60*10^(-3); %height of shaft [m]
I=m1*(L1/2+L2)^2+m2*L2*L2/12; %inertia of body part [kg*m^2]
Iw=389.6*10^(-9)*2; %inertia of wheel [kg*m^2]
Br=0.01; %rolling damping ratio [N*m/(rad/s)]
Bm=0.01; %bearing damping ratio [N*m/(rad/s)]
L=L2/2+(L1+L2)*m1/(2*m) %position of COM [m]
R=16*10^(-3); %radius of wheel [m]
g=9.8; %gravity [m/s^2]

%Weighting matrices
E=[Iw+(mw+m)*R*R m*R*L; m*R*L m*L*L+I]; %for d^2/dt^2 (phi and theta)
F=[Br+Bm -Bm; -Bm Bm]; %for d/dt (phi and theta)
G=[0; -m*g*L]; %for theta
H=[1; -1]; %for input torque

%state-space representation of the system
%state variable: phi, theta, d(phi)/dt, d(theta)/dt
A=[0 0 1 0; 0 0 0 1; [0; 0] -inv(E)*G -inv(E)*F] %system matrix
B=[0; 0; inv(E)*H] %input matrix
C=[R 0 0 0; 0 1 0 0] %output matrix
D=0
sys1=ss(A,B,C,D)
Gl=tf(sys1) %transfer function of sys1
Glzp=zpk(sys1) %Gain/pole/zero representation of sys1

%controllability and observability check for sys1
Cont=[B A*B A*A*B A*A*A*B];
rank(Cont)
Obs=[C; C*A; C*A*A; C*A*A*A];
rank(Obs)

%LQR controller design
xweight=eye(4); %weighting matrix Q
uweight=1; %weighting matrix R
K=-lqr(A,B,xweight,uweight) %gain matrix
sys1_lqr=ss(A+B*K,B,C,D); %close-loop system
initial(sys1_lqr, [0; 0.17; 0; 0]) %free response
```

A.2. Simulink Model Initialization

(Joshua Gafford)

```
%% Mr. Struggles Simulation Initialization File
```

```

%Description
% This script loads all initialization parameters required to run
% the Simulink model, modelSim.mdl. The model is run, a dynamic
% simulation is shown and the results are plotted.
clear all;clc;
global R L

% Struggles Mass Properties
m=.45;           % Mr. Struggles mass [kg]
mw=.007;         % Mr. Struggles wheel mass [kg]
R=.015;          % Wheel Radius [m]
L=.083;          % CG Height [m]
bh=0.001;        % Ground Damping
bf=0.001;        % Bearing Damping
g=9.81;          % Gravitational Constant [kg/m^2]
% Controller Gains
Kp=500;          % Proportional Gain
Kv=250;          % Derivative Gain
Ki=1;            % Integral Gain
% Disturbance Force
L_f=.1;          % Disturbance Force Location
F=0;             % Disturbance Force
th_f=pi/4;       % Disturbance Force angle

% Motor Parameters
t_sat=0.2;       % Motor Torque Limit [Nm]

% Noise Parameters
angle_var=0.2071^2; % Variance in Acceleration Angle Error
rate_var=1E-5;     % Variance in Gyro Rate Error
gyro_drift=(0.00016*pi/180); % Gyro Drift Rate [rad/s^2]
error_var=(0.18*pi/180)^2; % Gyro Angular Error[rad/s]
noise_var=(.1*pi/180)^2; % Variance in Gyro Rate Error

% Kalman Filter Covariance Parameters
Q_theta=error_var; % Angle Process Noise
Q_bias=.003;       % Rate Process Noise
R_theta=angle_var; % Angle Measurement Noise
kalman_switch=0;

% Initialization
init=[10*pi/180;0;0]; % Initial Conditions
theta_init=init(1);   % Kalman Filter Initialization
x_des=[0;0;0];        % Desired steady-state

% Miscellaneous
enc_res=1/(512*(2*pi)); % Encoder Resolution [counts/rev]
clock=0.005;           % Clock Period [s]
t_final=5;             % Simulation Stop Time

% Load model, set block properties
load_system('modelSim');
% Set Sample Rate for Discrete Blocks
set_param('modelSim/Process Noise/Random Number', 'SampleTime',
num2str('clock'));
set_param('modelSim/Noise Injection/Noise1', 'SampleTime', num2str('clock'));

```

```

set_param('modelSim/Noise Injection/Noise2', 'SampleTime', num2str('clock'));
set_param('modelSim/Noise Injection/Error1', 'SampleTime', num2str('clock'));
set_param('modelSim/Noise Injection/Discrete-Time Integrator', 'SampleTime',
num2str('clock'));
set_param('modelSim/Noise Injection/Quantizer', 'SampleTime',
num2str('clock'));
% Simulation Output Parameters
simOut = sim('modelSim', 'StopTime',
num2str(t_final), 'SimulationMode', 'rapid', 'AbsTol', '1e-5', ...
'SaveState', 'on', 'StateSaveName', 'xoutNew', ...
'SaveOutput', 'on', 'OutputSaveName', 'youtNew');

% Extract Results
theta=simOut.get('theta');theta=squeeze(theta);
gamma=simOut.get('gamma');gamma=squeeze(gamma);
z=simOut.get('z');z=squeeze(z);
tau=simOut.get('tau');tau=squeeze(tau);
z_obs=simOut.get('z_obs');z_obs=squeeze(z_obs);
z_filt=simOut.get('z_filt');z_filt=squeeze(z_filt);
P_out=simOut.get('P_out');P_out=squeeze(P_out);

t=simOut.get('t_out'); % Continuous Time Vector
t_d=0:clock:t_final; % Discrete Time Vector

% Plot Data
PlotSim(t,z(1,:)',z(2,:)',tau); % Dynamic Simulation

% Plot Actual, Observed, and Kalman Response
figure(1);
plot(t,z_obs(:,1).*180/pi,'b');hold on;
plot(t,z_filt(:,1).*180/pi,'r','LineWidth',2);hold on;
plot(t,z(1,:).*180/pi,'k','LineWidth',2);grid on;hold off;
legend('Observed Angle','Kalman Estimate','Actual Angle');
xlabel('Time [s]');ylabel('Angle [deg]');

% Plot Observed and Kalman Error
figure(2);
plot(t,(z_obs(:,1)-z(1,:)).*180/pi);hold on;grid on;
plot(t,(z_filt(:,1)-z(1,:)).*180/pi,'r');hold off;
legend('Error in Observed Angle','Error in Kalman Estimate');
xlabel('Time [s]');ylabel('Error [deg]');

% Display Root-Mean-Squared Deviation
rms=norm(z(1,:)-z_filt(:,1))/sqrt(length(z(1,:)));
disp(strcat('RMSD: ',num2str(rms),' rad'));

% Plot covariance norm
figure(3);
for j=1:length(t)
    plot(t(j),norm(P_out(:,:,j)),'o');hold on;
    grid on;
end
hold off;
xlabel('Time [s]');
ylabel('Error Covariance Norm');

```