Furuta Pendulum

Andrew Careaga Houck, Robert Kevin Katzschmann, Joao Luiz Almeida Souza Ramos

Department of Mechanical Engineering, Massachusetts Institute of Technology

2.151 Advanced System Dynamics & Control

Fall 2013

Table of Contents

ABSTRACT

In this project, we set out to design, build, and control a rotational inverted pendulum, also known as a Furuta pendulum. We succeeded in these goals, and achieved a system capable of stability and robustness to disturbance in the upright position, as well as swing-up and swing-down control. This report describes our physical system and dynamic model, as well as our controller design. Key results from our simulation and experimental results are presented, as well as conclusions and recommendations for future work. Connections to our 2.151 coursework, as well as the real-world hurdles we encountered during our project, are emphasized.

Note: OK to publish on Stellar

I. INTRODUCTION

The Furuta pendulum was invented at the Tokyo Institute of Technology by Katsuhisa Furuta in 1992, and has become a classic system for the application of linear and non-linear control theory. The Furuta pendulum is an under-actuated, 2-DOF system. It is composed of a pendulum attached to the end of a motor-driven link. The motor-driven link rotates in the horizontal plane, while the pendulum moves in a vertical plane perpendicular to the driven link.

We selected this system because of it offers an exciting opportunity to demonstrate the linear control methods we studied in 2.151. The mechanical system itself is not overly complicated, which allowed us to fabricate it ourselves.

The primary goal of our control system is to maintain the pendulum in an upright position by applying torque to the horizontal link, and for the system to be able to maintain this position in the presence of disturbances. We also implemented control methods for swing-up and swing-down control by using energy shaping methods.

II. PENDULUM DESIGN AND HARWARE

A. MECHANICAL DESIGN

The pendulum was designed with the goal of developing a functional but low-cost hardware. The round base was water jet cut from a scrap metal aluminum plate. The first link was 3D printed and motor and sensors are all hobby grade. Figure 2.1 shows the proposed CAD model for the experimental platform.

Figure 2.1: Proposed version of the Furuta pendulum.

The first link and the pendulum were designed so the system center of mass would be localized over the motor, to reduced the bending moment on the motor shaft. The 1024 pulses per revolution (PPR) encoder is connected to the pendulum shaft by a shaft coupler. Figure 2.2 show a view of the setup proposed.

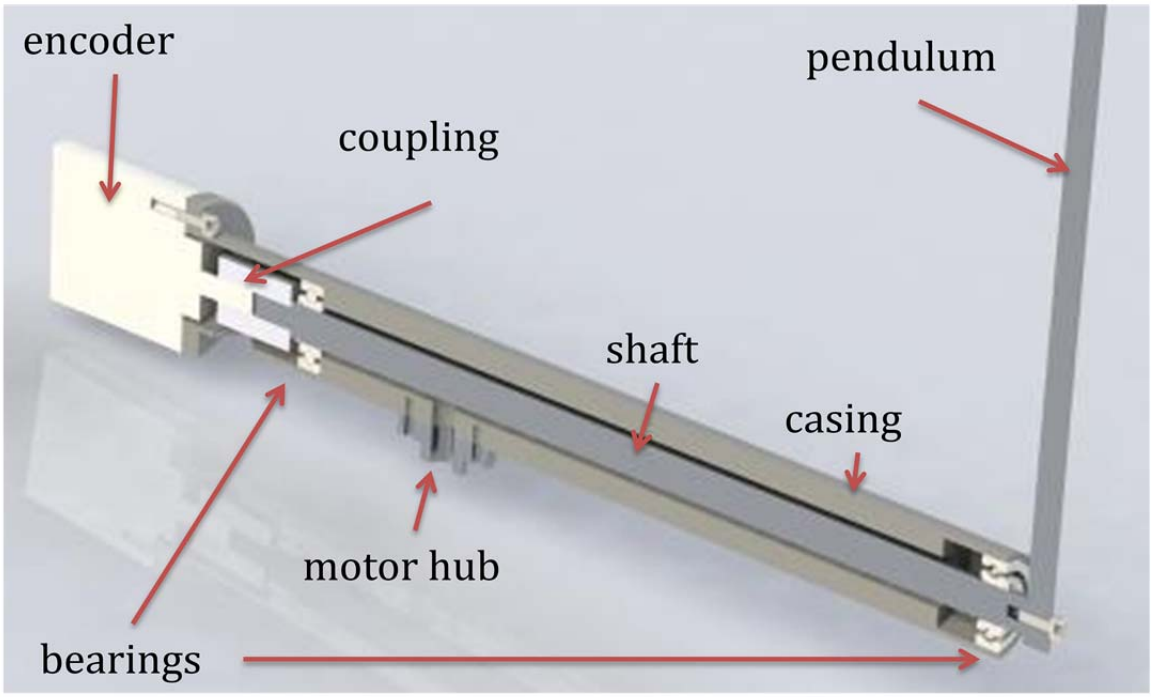


Figure 2.2: Proposed design for the pendulum.

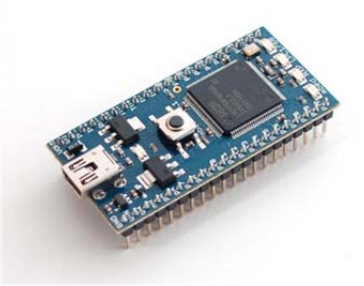
The motor is a brushed DC motor with 12V nominal voltage and is driven by a motor amplifier with build in current sensor. Figure 2.3 shows the motor and its driver. The output motor shafts account for a resolution of 300PPR one the encoder.

VNH5019 motor driver

Pololu 19:1 gearmotor w/encoder

Figure 2.3: Motor and driver used for the pendulum.

The system is controlled in real time using a ARM Cortex M3 Core based microcontroller programmed in C environament (figure 2.4) through serial communication to the host computer, This controller is able to run the control loop at 1kHz.



mbed NXP LPC1768 microcontroller

Figure 2.4: Real-time controller used.

Figure 2.5 shows the final mechanical setup for the pendulum and figure 2.6 shows the breadboard with the circuit implemented.

Figure 2.5: The Furuta Inverted Pendulum assembled.

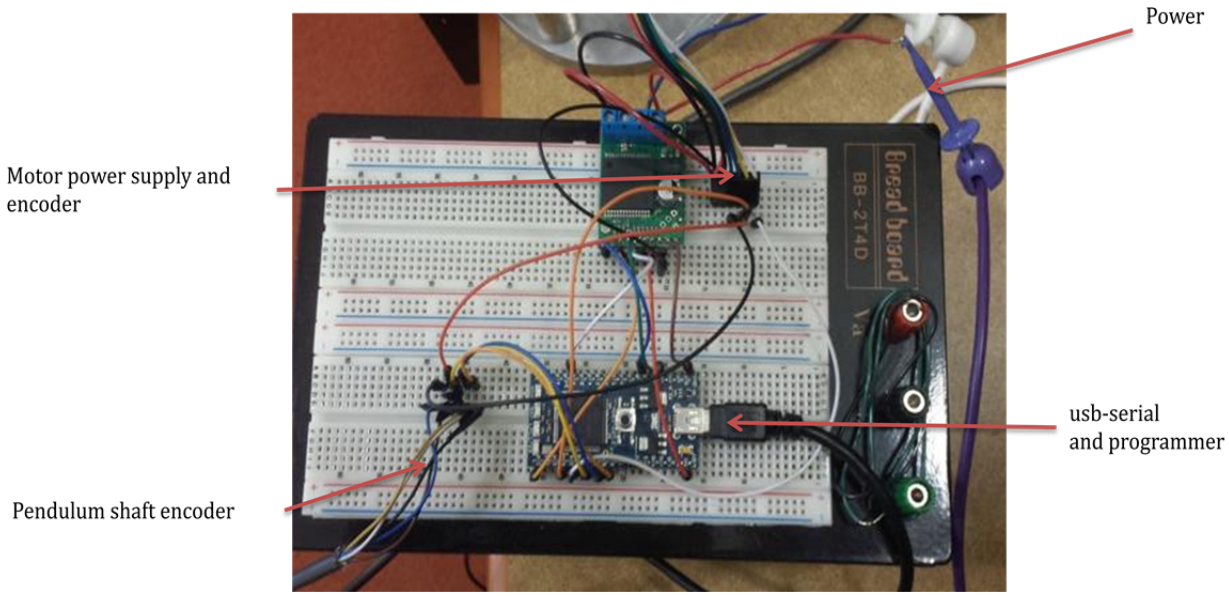


Figure 2.6: Breadboard and circuit used to control the motor.

III. SYSTEM MODEL

A. EQUATIONS OF MOTION

The fixed (XYZ) and body (xyz for the first link and x'y'z' for the pendulum) frames are shown in figure 3.1 and 3.2. Where θ_1 is positive in the Z//z direction and θ_2 is positive in the x direction.

Figure 3.1: Reference frames for the Furuta inverted pendulum.

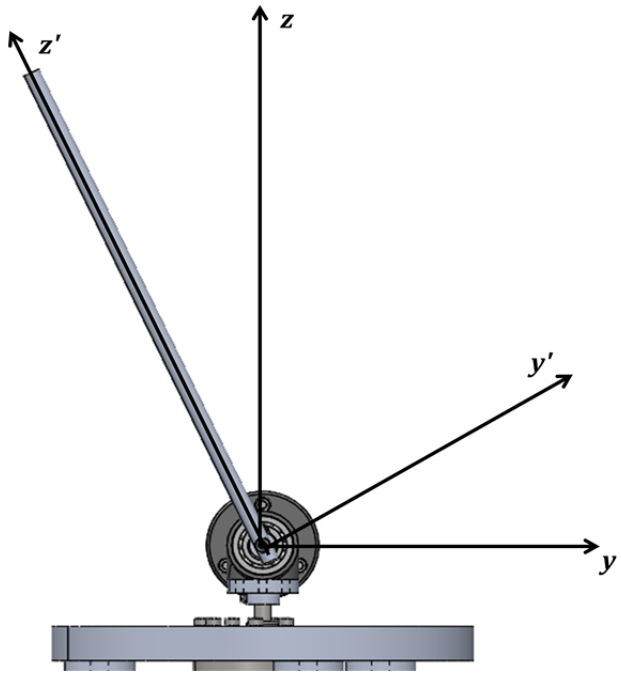


Figure 3.2: Pendulum fixed frame (x'y'z').

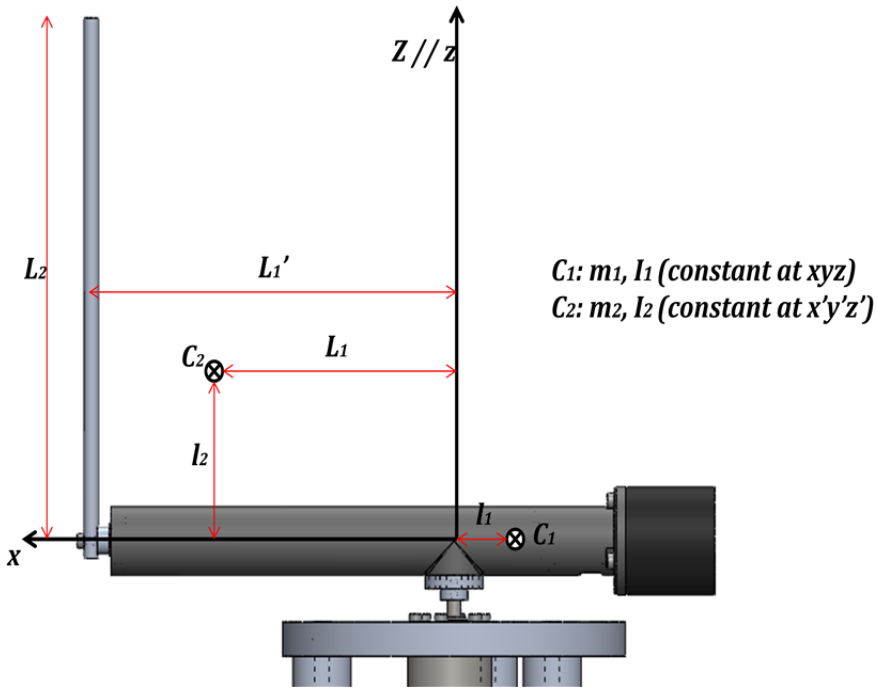


Figure 3.3: Center of mass position on the first link frame.

The following assumptions and simplifications where used for the model:

- Motor and pendulum have only viscous friction, static friction is not taken into account;
- Rigid connections between motor/link 1 and encoder/pendulum shaft
- Backlash effects are not being modeled;

- All materials are homogeneous;
- The encoder is a solid homogeneous object with known weight (for inertia estimation);
- When $\theta_2$is in a range between ±15 degrees the system dynamics can be considered to be linear.
- No base dynamics

The frame rotations were:

$$XYZ \xrightarrow{R(\theta_1)\ in\ Z} xyz \xrightarrow{R(\theta_2)\ in\ x} x'y'z'$$

From the previously defined frames and according to figure 3.3, we can write the position of the centers of mass for the first link (C_1) and for the pendulum (C_2) as:

$$p_{c1} = l_1 \begin{bmatrix} \cos(\theta_1) \\ \sin(\theta_1) \\ 0 \end{bmatrix}; \qquad p_{c2} = L_1 \begin{bmatrix} \cos(\theta_1) \\ \sin(\theta_1) \\ 0 \end{bmatrix} + l_2 \begin{bmatrix} \sin(\theta_2)\sin(\theta_1) \\ -\sin(\theta_2)\cos(\theta_1) \\ \cos(\theta_2) \end{bmatrix}$$

At xyz' frame, the angular velocity can be written as:

$$\omega_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}; \qquad \omega_2 = \begin{bmatrix} \dot{\theta}_2 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}$$

On the other hand, the linear velocity for C_1 and C_2 are:

$$v_{c1} = \begin{bmatrix} 0 \\ -l_1\dot{\theta}_1 \\ 0 \end{bmatrix}; \qquad v_{c2} = v_A + \omega_2 \times r_{A-C2} = \begin{bmatrix} 0 \\ L_1\dot{\theta}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\theta}_2 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \times \begin{bmatrix} 0 \\ -l_2\sin(\theta_2) \\ l_2\cos(\theta_2) \end{bmatrix}$$

$$v_{c2} = \begin{bmatrix} l_2\dot{\theta}_1\sin(\theta_2) \\ L_1\dot{\theta}_1 - l_2\dot{\theta}_2\cos(\theta_2) \\ -l_2\dot{\theta}_2\sin(\theta_2) \end{bmatrix}$$

Thus, we can write the kinetic energy for the system:

$$T = \frac{1}{2}m_1 {v_{c1}}^T v_{c1} + \frac{1}{2}\omega_1 I_1 \omega_1 + \frac{1}{2}m_2 {v_{c2}}^T v_{c2} + \frac{1}{2}\omega_2 {I_2}^r \omega_2$$

Where ${I_2}^r$ the inertia tensor at the xyz frame, which is calculated rotating the x'y'z' inertia tensor (constant) along the x axis:

$${I_2}^r = R(\theta_2)^T I_1 R(\theta_2)$$

$$I_2{}^r = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_2) & \sin(\theta_2) \\ 0 & -\sin(\theta_2) & \cos(\theta_2) \end{bmatrix} \begin{bmatrix} I_{2xx} & 0 & I_{2xz} \\ 0 & I_{2yy} & 0 \\ I_{2xz} & 0 & I_{2zz} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_2) & -\sin(\theta_2) \\ 0 & \sin(\theta_2) & \cos(\theta_2) \end{bmatrix}$$

$$I_2{}^r = \begin{bmatrix} I_{2xx} & I_{2xz}\sin(\theta_2) & I_{2xz}\cos(\theta_2) \\ I_{2xz}\sin(\theta_2) & I_{2yy}\cos^2(\theta_2) + I_{2zz}\sin^2(\theta_2) & \frac{(I_{2zz} - I_{2yy})\sin(2\theta_2)}{2} \\ I_{2xz}\cos(\theta_2) & \frac{(I_{2zz} - I_{2yy})\sin(2\theta_2)}{2} & I_{2zz}\cos^2(\theta_2) + I_{2yy}\sin^2(\theta_2) \end{bmatrix}$$

And the inertia matrix for the first link is simply:

$$I_1 = \begin{bmatrix} I_{1xx} & 0 & 0 \\ 0 & I_{1yy} & 0 \\ 0 & 0 & I_{1zz} \end{bmatrix}$$

In which only I_{1zz} has influence over the system dynamics.

The same way, the potential energy for the pendulum:

$$P = m_2 g l_2 \cos(\theta_2)$$

From these we can write the Lagrangian,

$$L = T - P$$

And the equations of motion for each angle:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) - \frac{\partial L}{\partial \theta_1} = u - b_1\dot{\theta}_1$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_2}\right) - \frac{\partial L}{\partial \theta_2} = -b_2\dot{\theta}_2$$

In which u is the motor input torque over the first link.

The equations of motion can be written in matrix form:

$$H\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + D\left(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2\right) + G(\theta_1, \theta_2) = \begin{bmatrix} u \\ 0 \end{bmatrix}$$

And the nonlinear representation:

$$\ddot{q} = \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} = H^{-1}\left[\begin{bmatrix} u \\ 0 \end{bmatrix} - D - G\right] = f(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2, u)$$

And can be written in the linearized state space form about the upright position

$X_0 = [0 \quad 0 \quad 0 \quad 0]^T$:

$$\dot{X} = AX + Bu$$

$$\dot{X} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \end{bmatrix} X + \begin{bmatrix} 0 \\ 0 \\ B_1 \\ B_2 \end{bmatrix} u$$

Where:

$$A_{11} = \left.\frac{\partial f_1}{\partial \theta_1}\right|_{X=X_0} ; \; A_{12} = \left.\frac{\partial f_1}{\partial \theta_2}\right|_{X=X_0} ; \; A_{13} = \left.\frac{\partial f_1}{\partial \dot{\theta}_1}\right|_{X=X_0} ; \; A_{14} = \left.\frac{\partial f_1}{\partial \dot{\theta}_2}\right|_{X=X_0}$$

$$A_{21} = \left.\frac{\partial f_2}{\partial \theta_1}\right|_{X=X_0} ; \; A_{22} = \left.\frac{\partial f_2}{\partial \theta_2}\right|_{X=X_0} ; \; A_{23} = \left.\frac{\partial f_2}{\partial \dot{\theta}_1}\right|_{X=X_0} ; \; A_{24} = \left.\frac{\partial f_2}{\partial \dot{\theta}_2}\right|_{X=X_0}$$

$$B_1 = \left.\frac{\partial f_1}{\partial u}\right|_{X=X_0} ; \; B_2 = \left.\frac{\partial f_2}{\partial u}\right|_{X=X_0}$$

The previous equation of motion where calculated using the Symbolic Toolbox in MATLAB (the respective code is shown in APPENDIX A). The inputs for the A-matrix are shown in APPENDIX B.

The linear dynamic equations for the DC brushed motor are broadly known and can be written as:

$$e = R_m i + L_m \frac{di}{dt} + k_t \dot{\theta}_1$$

$$k_t i - b_1 \dot{\theta}_1 = J_m \ddot{\theta}_1$$

Which introduces another state variable for the current i and defines the motor input voltage e. These equations can be added to the previous matrix expression before linearization, thus:

$$H = \begin{bmatrix} H_{11} & H_{12} & 0 \\ H_{21} & H_{22} & 0 \\ 0 & 0 & L_m \end{bmatrix}; \qquad D = \begin{bmatrix} C_1 - k_t i \\ C_2 \\ k_t \dot{\theta}_1 + R_m i \end{bmatrix}$$

The final system is 5th order in which both angles and angular velocities and motor current. The input signal is the voltage applied to the motor terminals.

$$X = [\theta_1 \quad \theta_2 \quad \dot{\theta}_1 \quad \dot{\theta}_2 \quad i]^T$$

$$u = e$$

Considering the system has encoders for angle feedback and a current sensor, the C-matrix and D-matrix are:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}; \qquad D[0]$$

B. SYSTEM IDENTIFICATION

The inertia and length parameters for the pendulum where extracted from the CAD model developed in SolidWorks. Figure 3.4 shows the assemblies for the first link and for the pendulum, the inertia tensors where calculated at each center of mass and aligned to the xyz and x'y'z' frames.

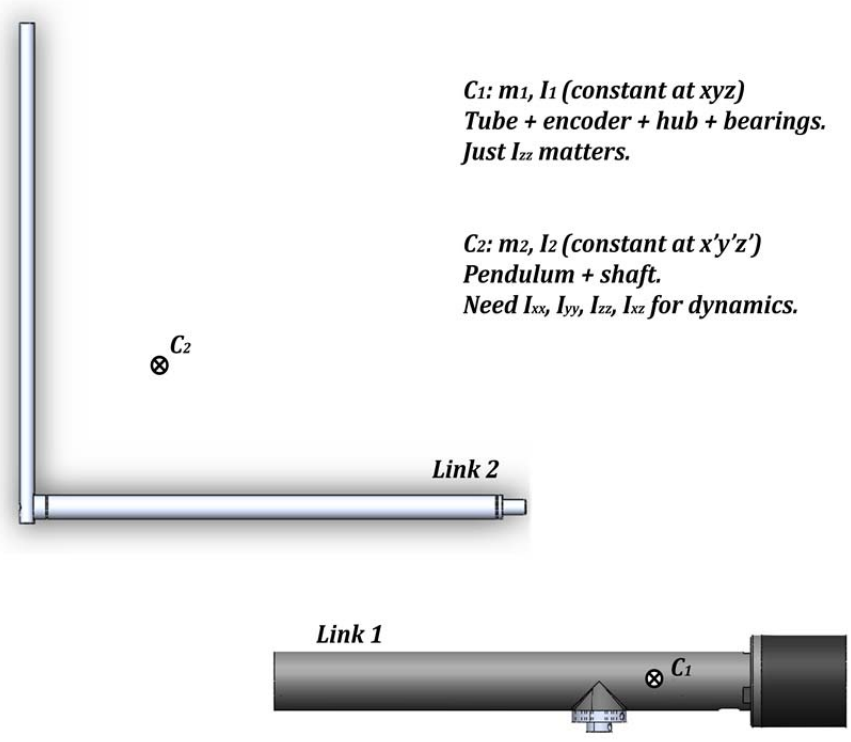


Figure 3.4: First and second link assemblies for inertia estimation.

For angle about ±15 degrees around the down position, the dynamic equation for the pendulum can be approximated as linear and can be written as

$$I_{2x}\ddot{\theta}_2 + b_2\dot{\theta}_2 + m_2 g l_2 \theta_2 = 0;$$

Or

$$\ddot{\theta}_2 + 2\zeta\omega_n\dot{\theta}_2 + {\omega_n}^2\theta_2 = 0;$$

The damped oscillating frequency ω_d is given as

$$\omega_n = \omega_d\sqrt{1-\zeta^2};$$

Figure 3.5 shows the experimental decay of the free pendulum in time. It's noticeable that the amplitude decay is very close to linear, thus the exponential decay estimated in this work will just be an approximation. Then, we can write:

$$\delta = ln\left|\frac{x_1}{x_2}\right|; \qquad \zeta = \frac{1}{\sqrt{1+\left(\frac{2\pi}{\delta}\right)^2}}; \qquad \frac{b_2}{I_{2x}} = 2\zeta\omega_n;$$

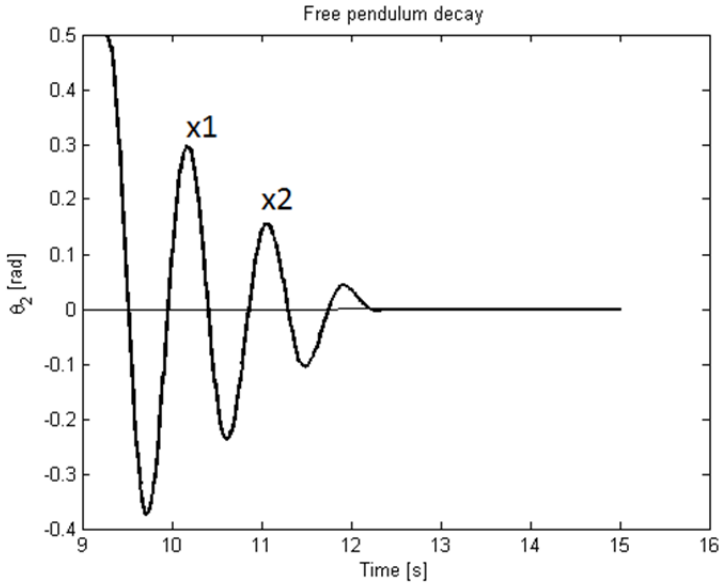


Figure 3.5 Pendulum damped oscillation

The friction in the bearings for the pendulum was estimated using a second order model for the pendulum and dropping it form a known initial angle, allowing it to swing until all energy was totally dissipated.

Similarly, the motor dynamic parameters were estimated applying different voltages and analyzing the no load steady state response. The response can be visualized in figures 3.6, 3.7 and 3.8.

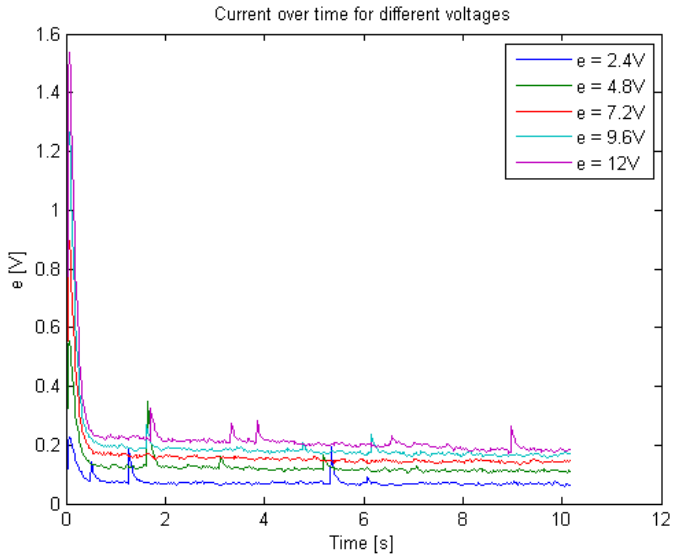


Figure 3.6 Time evolution of the armature current.

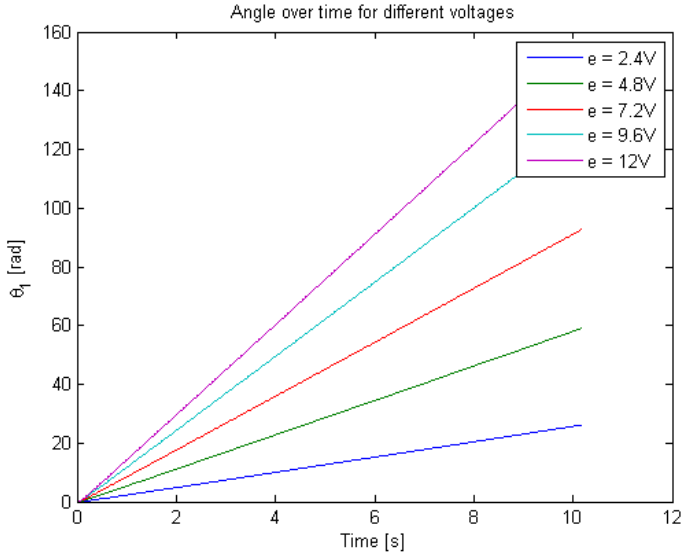


Figure 3.7: Motor position over time.

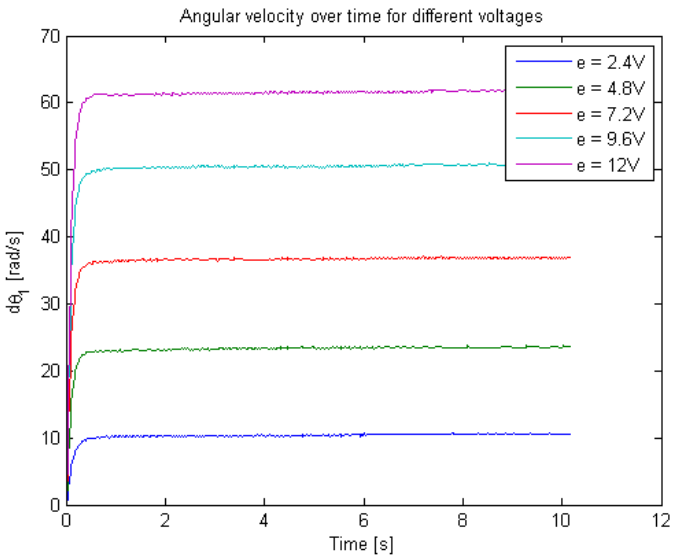


Figure 3.8: Time evolution of the angular velocity of the motor shaft

For each voltage we can write that, in steady state:

$$\frac{di}{dt} = 0; \qquad \ddot{\theta}_1 = 0; \qquad \dot{\theta}_1 = \dot{\theta}_{1_{ss}} = constant; \qquad i = i_{ss} = constant.$$

$$k_t = \frac{e - R_m i_{ss}}{\dot{\theta}_{1_{ss}}}$$

$$b_1 = \frac{k_t i_{ss}}{\dot{\theta}_{1_{ss}}}$$

The final values for the motor constants were calculating using the average for each experiment:

$$k_t = \frac{1}{5}\sum_{k=1}^{5} k_{t_i}; \qquad b_1 = \frac{1}{5}\sum_{k=1}^{5} b_{1_i}$$

Values for the motor impedance and resistance were measured according to figure 3.9.

Figure 3.9: Measurement of armature internal resistance and winding impedance.

Finally, all the parameters for the system are shown in table 2.1.

Table 2.1: Estimated system parameters:

Parameter	Value	Parameter	Value
$\boldsymbol{g}$	$9.8\ m.s^{-2}$	I_{1z}	$0.0013\ kg\ m^2$
$\boldsymbol{b_1}$	$6.0\ 10^{-4} Nm/s$	I_{2x}	$5.34\ 10^{-4} kg\ m^2$
$\boldsymbol{b_2}$	$5.52\ 10^{-4} Nm/s$	I_{2y}	$8.41\ 10^{-4} kg\ m^2$
$\boldsymbol{m_1}$	$0.1862 kg$	I_{2z}	$3.10\ 10^{-4} kg\ m^2$
$\boldsymbol{m_2}$	$0.065 kg$	I_{2xz}	$-2.40\ 10^{-4} kg\ m^2$
$\boldsymbol{L_1}$	$0.0955\ m$	k_t	$0.182\ Nm\ A^{-1}$
$\boldsymbol{l_1}$	$0.03478\ m$	R_m	$3.6\ \Omega$
$\boldsymbol{L_2}$	$0.2983\ m$	L_m	$1.845\ mH$
$\boldsymbol{l_2}$	$0.05847\ m$	J_m	$0.001\ kg\ m^2$

Thus, the numeric A-matrix is given by:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0.0152 & -0.0003 & -0.0002 & 0.0928 \\ 0 & 0.0613 & -0.0002 & -0.0009 & 0.0740 \\ 0 & 0 & -0.0986 & 0 & -1.9512 \end{bmatrix} .10^3; \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 542.0 \end{bmatrix}$$

Using the obsv() commands from MATLAB we can check if the system is observable, which means all variables can be estimated or measured by the model:

```
rank(obsv(A,C)) = 5;
```

Numerically, the rank of the observability matrix is apparently full, but the condition number is considerably high:

```
cond(obsv(A,C)) = 1.4409e+13;
```

The system is at the borderline of being observable. Matlab calculates that the rank of the controllability matrix is not full:

```
rank(ctrb(A,B)) = 4;
```

Once again, having a closer looking at the condition number reveals an even higher condition number than for the observability matrix.

```
cond(obsv(A,C)) = 1.6016e+15;
```

Matlab interprets the system to be not fully controllable, we assume for now that the high condition numbers for both the observability and the controllability are caused by the coarsely estimated values for the inductance of the motor circuit and the inertia of the motor shaft. Since the bandwidth of the electrical system part is not overall determining, we conclude that we will still be able to control and observe the system.

The eigenvalues and eigenvectors for the A-matrix are

$$\lambda_1 = 0;\ \lambda_2 = -1946.5;\ \lambda_3 = 7.1;\ \lambda_4 = -9.3;\ \lambda_5 = -3.7;$$

$$v_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix};\ v_2 = \begin{bmatrix} 0.0000 \\ 0.0000 \\ -0.0476 \\ -0.0380 \\ 0.9981 \end{bmatrix};\ v_3 = \begin{bmatrix} -0.0218 \\ -0.1377 \\ -0.1545 \\ -0.9781 \\ 0.0078 \end{bmatrix};\ v_4 = \begin{bmatrix} -0.0429 \\ -0.0982 \\ 0.3981 \\ 0.9108 \\ -0.0202 \end{bmatrix};\ v_5 = \begin{bmatrix} 0.2470 \\ -0.0729 \\ -0.9256 \\ 0.2733 \\ 0.0469 \end{bmatrix};$$

The first eigenvalue ($\lambda_1 = 0$) corresponds to the rigid body mode when the first link spins and the pendulum angle is fixed. The second eigenvalue ($\lambda_2 = -1946.5$), the fastest and stable one is the electrical part of the system, with very fast dynamics. The third eigenvalue ($\lambda_3 = 7.1$), corresponds to the unstable equilibrium for the pendulum in the upright position, when $\theta_2 = 0$. The last two eigenvalues ($\lambda_4 = -9.3;\ \lambda_5 = -3.7$) correspond to the slower and stable dynamics from the mechanical part. The system has only real valued eigenvalues due to the friction; there's no oscillation within the linear range considered.

IV. CONTROLLER DESIGN

For the upright control of the pendulum, a full-state feedback controller was designed under the assumption that we can measure all system states directly. The tuning of the feedback gains was done using a linear quadratic regulator design approach (LQR). The swing-up from the stable equilibrium and the swing-down from the upright position were achieved by a non-linear controller approach called energy shaping. These two controller implementations will be further elaborated in the following sections.

A. DIFFERENTIATION AND NOISE FILTERING

Both rotating axis have a quadrature encoder connected to it, which allows us to directly measure the first two system states. We did not implement a physical device like a tachometer to directly measure the velocity, we therefore looked into either using an observer or design a differentiator. An observer would be based upon an uncertain system model, which is why we opted for combining a discrete differentiator with a first order low-pass filter to derive the angular velocities from the noisy quadrature encoder outputs. We manually tuned the roll-off frequencies of each velocity filter to achieve a balance between an acceptable velocity phase offset and a desirable noise rejection.

The measured motor current is quite noisy due to the nature of commutator brushes. We also applied a first order low- pass filter to the current signal before feeding it back to the controller algorithm.

B. LINEAR CONTROLLER FOR UPRIGHT POSE

We calculated the initial gains K of the full state feedback controller using a conservative, "expensive control" ratio between the system Q and the input R weighting matrix in order to avoid saturating the output of our small motor. This first gains estimate was sufficient enough to allow the controller to balance the pendulum upright during the very first try run.

Gradually adjusting the ratio between Q and R towards "cheap control" improved the robustness of the pendulum to disturbance. We settled at a ratio that allowed the motor to rarely saturate while in action. We were also able to observe a better and more stable control behavior by decreasing the maximal allowable angular position deviations while keeping the angular velocities at a similar order of magnitude.

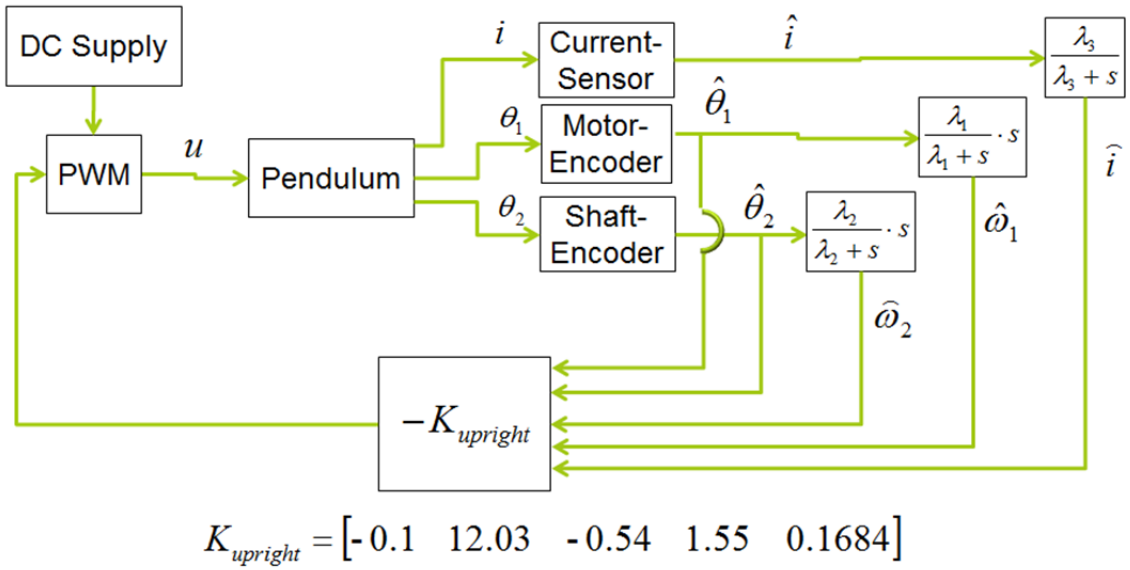
The control architecture of the linear controller is depicted in Figure 4.1. A pulse-width modulated signal is generated by an "mbed NXP LPC1768" microcontroller an applied to a motor controller, which in turn applies the amplified voltage signal e onto the motor of axis 1 of the Furuta pendulum. The quadrature-encoded pulses of the motor encoder and the shaft encoder are counted by the microcontroller and are translated into angular measurements. At the same time, a current sensor measures the motor current and provides its readings to the microcontroller. The angles are differentiated and filtered to obtain the velocities, and the current itself is also low-pass filtered. The unfiltered angles, the derived velocities and the filtered current, encompassing all states of the system, are then multiplied by the LQR determined gain $K_{upright}$ and are feed back to the PWM.

C. SWING UP CONTROLLER

The non-linear swing up from the stable equilibrium is achieved by applying the concept of energy shaping (see notes of class 6.832 on under-actuated robotics by Russ Tedrake). The controller architecture is shown in Figure 4.2. The controller only requires the angular position and velocity of the pendulum axis 2. The maximal potential energy is achieved at the upright, instable equilibrium. The difference between the currently given energy and the maximal energy is multiplied by the axis 2 velocity and a manually designed gain value K_{swing}. The obtained control value

is used as the PWM signal value. The energy of the pendulum for any given angle is a long sum of inertial energy terms and the potential energy of the current height. Computational limitations of the used microcontroller lead to the effect that a swing up was not possible. No matter how we designed the gain value K_{swing} . Reducing the energy calculation to the only significantly large inertial energy term and the potential energy, as shown in Figure 4.2, enabled the controller to swing up the pendulum to the upright position range.

The Furuta pendulum demonstration is started with the pendulum itself at the stable equilibrium and in no motion. The filter initialization of axis 2 is intentionally offset by 0.2 percent in order to give the pendulum an automatic initial impulse. It then swings up to the upright position and switches to the linear controller the moment the pendulum passes through the instable equilibrium. After 20 seconds, the controller switches back to the energy shaping based controller, but applies this time a negative gain in order to swing down.



$$K_{upright} = \begin{bmatrix} -0.1 & 12.03 & -0.54 & 1.55 & 0.1684 \end{bmatrix}$$

Figure 4.1: Linear Controller within ±15º offset of the instable upright equilibrium

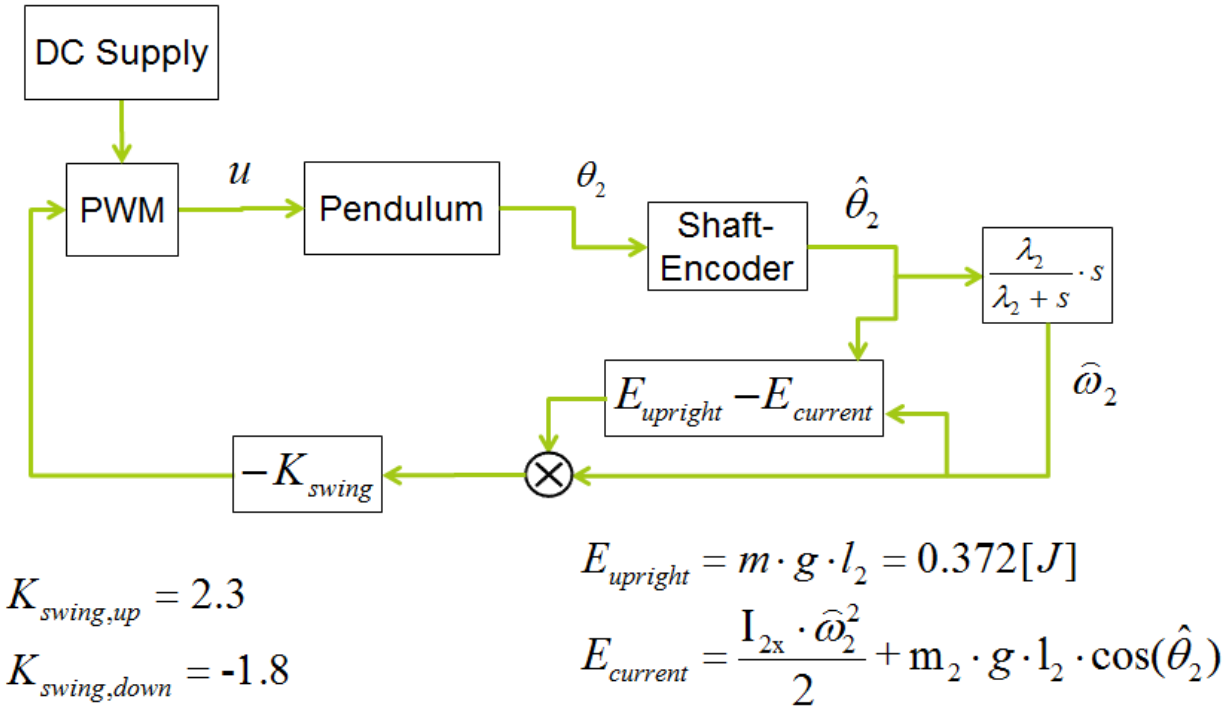


Figure 4.2: Swing Up Controller using Energy Shaping

V. RESULTS FROM SIMULATIONS AND EXPERIMENTS

SIMULATION

The system dynamics were simulated using the ode45 function on MATLAB, this function is able to solve nonlinear ODEs numerically. The controller was first designed and tuned using the simulation and the obtained gains were applied to the real system. The nonlinear equations of motion were always used during the simulation, including when the pendulum state lies within the considered to be linear region. Even though not used in practice, the simulation includes a state observer with poles eight times faster than the system poles.

The voltage applied to the motor is shown in figure 5.1. Notice that the motor never saturates (nominal voltage 12V), even during the swing-up control.

The MATLAB code for the simulation can be found in appendix C.

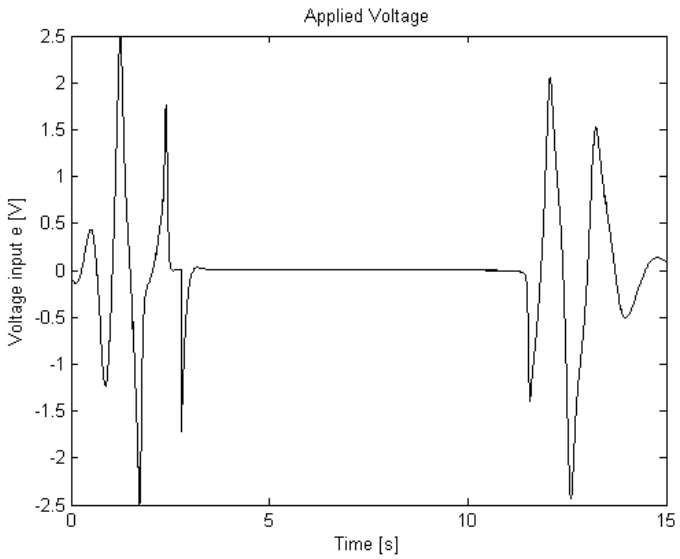


Figure 5.1: Input voltage for pendulum stabilization.

The angular positions and velocities are shown in figure 5.2 and 5.3 along with the estimated velocities for the observer. We can see that the observer is not able to satisfactory estimate the velocities on the nonlinear regime once it's a linear observer.

The pendulum starts from equilibrium at the down position where $\theta_2 = 0$, swings-up, stabilizes and swing back down to the initial state.

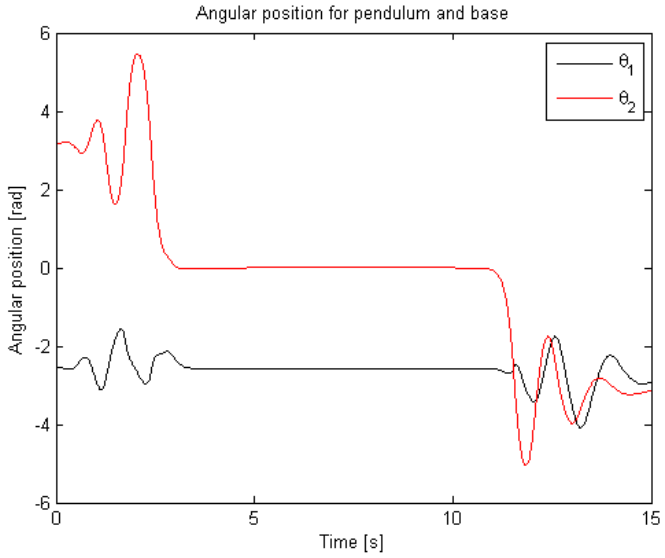


Figure 5.2: Pendulum position during swing-up, stabilization and swing-down control.

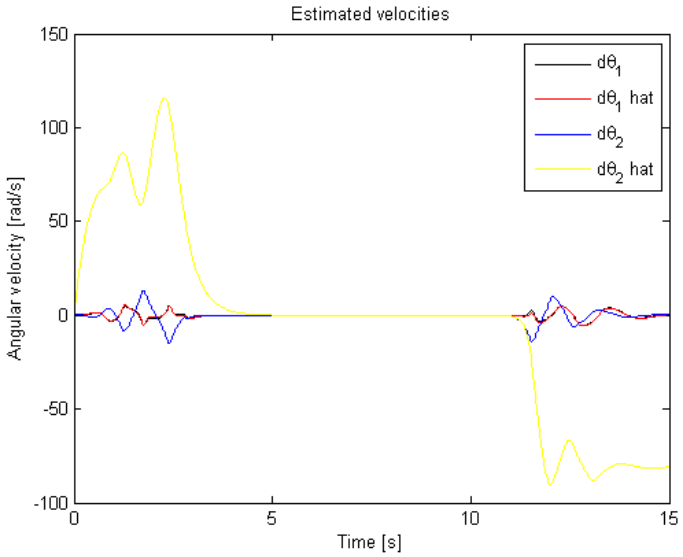


Figure 5.3: Pendulum real and estimated angular velocities during swing-up, stabilization and swing-down control.

EXPERIMENT

The following data was gathered during swing-up and stabilization, and demonstrates the good performance of our controller. The trial begins with the pendulum swinging up to its upright position. This employed a simple energy addition control, and swung the pendulum upright in approximately 4 seconds. After the pendulum crossed the upright position, the stabilization controller was activated. The system exhibited some persistent high-frequency oscillation in the upright position. This can be mostly attributed to backlash in our motor.

We disturbed the system by hitting the pendulum several times during the trial. The controller's response to these disturbances can be seen best in Figure 5.5, where they appear as impulses to the pendulum velocity. The system recovers to equilibrium quickly with negligible overshoot. We iterated our LQR design multiple times to achieve this performance. At first, an "expensive control" strategy was used, since we doubted the strength of our small motor. We gradually shifted our optimization parameters towards "cheap control" to improve performance while avoiding actuator saturation.

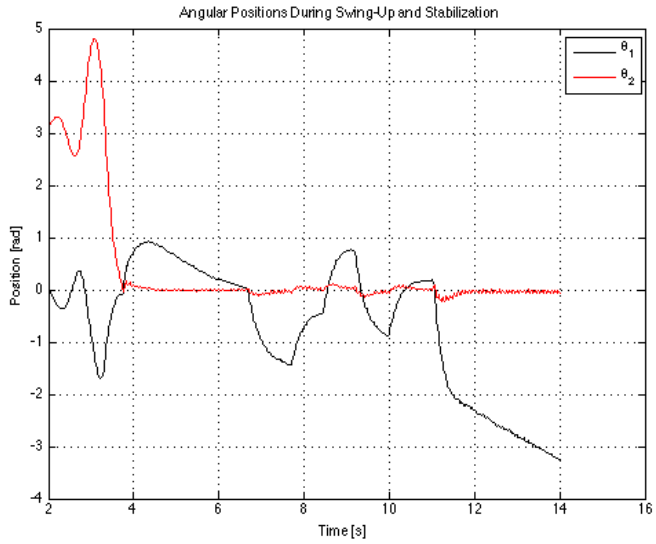


Figure 5.4: Angular Position During Swing-Up and Stabilization

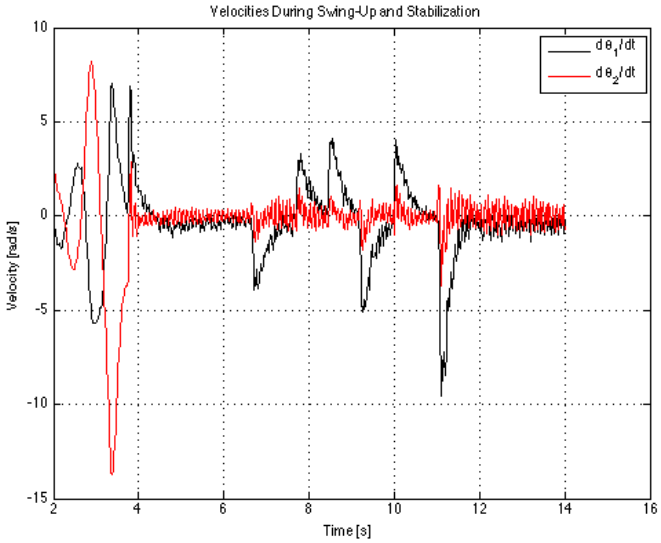


Figure 5.5: Angular Velocities During Swing-Up and Stabilization

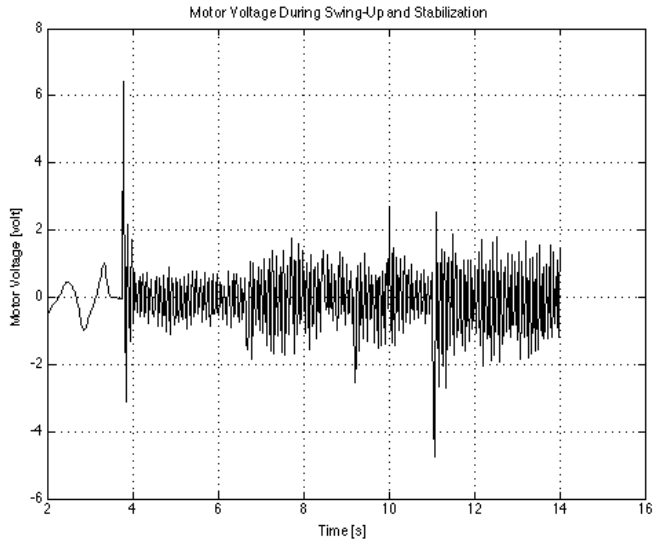


Figure 5.6: Motor Voltage During Swing-Up and Stabilization

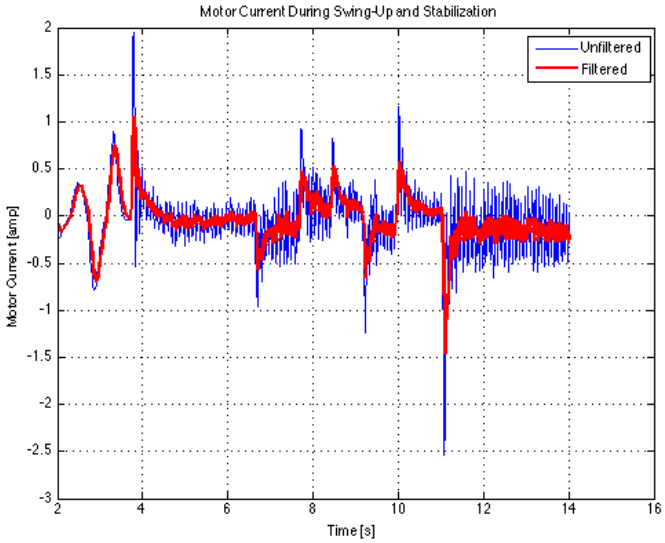


Figure 5.7: Motor Current During Swing-Up and Stabilization

Figure 5.7 shows the motor current during operation. The blue data shows raw current measurements, while the red shows the data after being conditioned with a first-order low pass filter. The break frequency of this filter was tuned manually. We experimented with different values until we found a value that tracked current spikes adequately, while filtering most of the unwanted high-frequency noise. This process was also used to set the break frequencies of the differentiators used to estimate $d\theta_1/dt$ and $d\theta_2/dt$.

VI. CONCLUSIONS AND RECOMMENDATIONS

In this project, we successfully designed, built, and controlled a Furuta pendulum. Our full-state controller demonstrates good performance, and maintains the pendulum in the upright position. The system demonstrates some oscillatory behavior, mostly as the result of backlash in our motor's gearbox. Careful system modeling and parameter estimation, as well as a simple and robust physical system, were essential for this success.

Although the application of linear methods we studied in 2.151 were central to the project, we learned most by working through the nonlinear behaviors that come with a real physical system. Highly nonlinear, un-modeled behaviors such as static friction and backlash weighed heavily into the behavior of the controlled system, and minimizing them required adjustments based more on intuition than theory. Additional problems with seemingly straightforward procedures, such as measuring motor resistance or reading encoder pulses, provided unexpected and time-consuming hurdles. These problems gave us even more appreciation for the difficulty of implementing a control system on hardware (as well as the satisfaction when it actually works).

The Furuta pendulum is a classic system for the application of various control methodologies, and offers many opportunities for future work. One step could be the application of nonlinear control methods, functional outside of the small linear range used in this project. Another interesting development would be the addition of adaptive control, in which the mass and geometry of the pendulum could be changed during operation, and responded to appropriately by the controller.

LIST OF PROJECT TASKS

Electronics

- Controller Hardware (Joao)
- Sensor and Actuator - Model Identification (Andrew)
- Real-Time Code on Microcontroller (Robert, Joao)

Mechanics

- CAD Design of System (Joao, Robert, Andrew)
- Purchase of Parts (Joao)
- Printing Parts/ Manufacture Parts (Andrew, Robert)
- Assembly of System (Joao, Andrew, Robert)
- Parameter Identification/Estimation (Friction, Motor Model) (Andrew)

Controls Simulation in MATLAB

- Modeling a physical system (Joao)
- Simplification/Linearization into state-determined form (Andrew)
- Full-state feedback control and LQ design (Robert)
- Observer design (Joao)
- Differentiator (Andrew)
- Robustness to modeling uncertainties (Robert)

REFERENCES

Bernard Friedland, Control system design: an introduction to state-space methods. Dover Books on Engineering, 2005

Eric R. Westervelt et al, Feedback Control of Dynamic Bipedal Robot Locomotion, CRC Press, 2007

Neville Hogan, 2.151 Advanced System Dynamics and Control course notes

Russ Tedrake, Underactuated Robotics: Learning, Planning, and Control for Efficient and Agile Machines (Course Notes for MIT 6.832), 2009

Wikipedia, Furuta Pendulum, last accessed 12/5/13, [http://en.wikipedia.org/wiki/Furuta_pendulum]

APPENDICES

APPENDIX A: MATLAB CODE FOR EQUATIONS OF MOTION

```
clc;
close all;

%Define global variables
global ssA ssB

%define symbolic variables
syms theta1 dtheta1 theta2 dtheta2 ddtheta1 ddtheta2 tau1 real
syms vc1 vc2 w1 w2 real
syms g m1 m2 I1x I1y I1z I2x I2y I2z Ixz2 l1 L1 l2 L2 b1 b2 e real
syms T P real

%Define Position of the center of mass
pC1 = l1*[cos(theta1) sin(theta1) 0]';
pA = L1*[cos(theta1) sin(theta1) 0]';
pC2 = pA + l2*[sin(theta2)*sin(theta1) -sin(theta2)*cos(theta1)
cos(theta2)]';
pT = pA + L2*[sin(theta2)*sin(theta1) -sin(theta2)*cos(theta1)
cos(theta2)]';

%Angular velocities in 1st body-fixed frame
w1 = [0 0 dtheta1]';
w2 = [dtheta2 0 dtheta1]';

%Angular velocities in 1st body-fixed frame
vc1 = [0 dtheta1*l1 0]';
vA = [0 dtheta1*L1 0]';
%vc2 = vA + w2 x rAc2
vc2 = [l2*dtheta1*sin(theta2) dtheta1*L1-l2*dtheta2*cos(theta2) -
l2*dtheta2*sin(theta2)]';
```

```
%Inertia Matrices
I1 = diag([I1x I1y I1z]);
I2 = [[I2x  0   Ixz2];
      [0    I2y 0 ];
      [Ixz2 0   I2z];];

%Rotating the I2 Inertia matrix:
R = [1 0            0;
     0 cos(theta2) -sin(theta2);
     0 sin(theta2)  cos(theta2);];
I2r = simplify(R'*I2*R);

%Kinetic Energy
Tp = m2*vc2'*vc2/2 + w2'*I2r*w2/2;
T = m1*vc1'*vc1/2 + w1'*I1*w1/2 + m2*vc2'*vc2/2 + w2'*I2r*w2/2;
T = simplify(T);

%Potential Energy
P = m2*g*l2*cos(theta2);

%Total pendulum Energy
Ep = simplify(Tp + P);

%Total Energy dT/dt is for the swing up control, needed later
TE = simplify(T + P);
dTE = diff(TE,dtheta1)*ddtheta1 + diff(TE,theta1)*dtheta1...
    + diff(TE,dtheta2)*ddtheta2 + diff(TE,theta2)*dtheta2;

%The Lagrangian
Lg = simplify(T - P);

%Partial derivative in dtheta1 and dtheta2
partial_dtheta1 = simplify(diff(Lg,dtheta1));
partial_dtheta2 = simplify(diff(Lg,dtheta2));

%Time derivative
time_der1 = diff(partial_dtheta1,dtheta1)*ddtheta1 +
diff(partial_dtheta1,theta1)*dtheta1...
    + diff(partial_dtheta1,dtheta2)*ddtheta2 +
diff(partial_dtheta1,theta2)*dtheta2;
time_der1 = simplify(time_der1);

time_der2 = diff(partial_dtheta2,dtheta1)*ddtheta1 +
diff(partial_dtheta2,theta1)*dtheta1...
    + diff(partial_dtheta2,dtheta2)*ddtheta2 +
diff(partial_dtheta2,theta2)*dtheta2;
time_der2 = simplify(time_der2);

%Partial derivative in theta1 and theta2
partial_theta1 = simplify(diff(Lg,theta1));
partial_theta2 = simplify(diff(Lg,theta2));

%Equations of motion
```

```
Eq1 = simplify(time_der1 - partial_theta1);
Eq2 = simplify(time_der2 - partial_theta2);

%Matrix Form: H*ddq + C + G = Tau
%Matrix terms for Eq1
H11 = diff(Eq1,ddtheta1);
H12 = diff(Eq1,ddtheta2);
G1 = diff(Eq1,g)*g;
C1 = Eq1 - H11*ddtheta1 - H12*ddtheta2 - G1;

%Matrix terms for Eq2
H21 = diff(Eq2,ddtheta1);
H22 = diff(Eq2,ddtheta2);
G2 = diff(Eq2,g)*g;
C2 = simplify(Eq2 - H21*ddtheta1 - H22*ddtheta2 - G2);

syms kt Rm Lm Jm im dim
% D matrix combines mechanical inertia with electrical inductance
D_mtx = [H11 H12 0;
         H21 H22 0;
         0   0   Lm];
C_vec = [C1-kt*im; C2; (kt*dtheta1+Rm*im);];
G_vec = [G1; G2; 0;];
B_mtx = diag([b1 b2 0]);

% ------------------------------------------------------------------------
---
% Create MATLAB functions for each EOM entry
param_list = {'g','p(1)'; 'L1','p(2)'; 'l1','p(3)'; 'm1','p(4)';
'I1x','p(5)';'I1y','p(6)';'I1z','p(7)';...
     'L2','p(8)';'l2','p(9)'; 'm2','p(10)';
'I2x','p(11)';'I2y','p(12)';'I2z','p(13)';'b1','p(14)';'b2','p(15)';'Ix
z2','p(16)';...
     'kt','p(17)';'Rm','p(18)';'Lm','p(19)';'Jm','p(20)';};

list_q = {'theta1','q(1)'; 'theta2','q(2)';'im','q(3)'};
list_dq = {'dtheta1','q(1)'; 'dtheta2','q(2)'; 'dim','q(3)'};

write_fcn('D_matrix.m',{'q','p'},[list_q; param_list],{D_mtx,'D'});
write_fcn('C_vector.m',{'q','dq','p'},[list_q; list_dq;
param_list],{C_vec,'C'});
write_fcn('G_vector.m',{'q','p'},[list_q; param_list],{G_vec,'G'});

write_fcn('KinEnergy.m',{'q','dq','p'},[list_q; list_dq;
param_list],{T,'KE'});
write_fcn('PotEnergy.m',{'q','p'},[list_q; param_list],{P,'PE'});
write_fcn('PendEnergy.m',{'q','dq','p'},[list_q; list_dq;
param_list],{Ep,'Pend_E'});

write_fcn('CG1.m',{'q','p'},[list_q; param_list],{pC1,'pC1'});
write_fcn('CG2.m',{'q','p'},[list_q; param_list],{pC2,'pC2'});
write_fcn('pAxis.m',{'q','p'},[list_q; param_list],{pA,'pA'});
write_fcn('pTip.m',{'q','p'},[list_q; param_list],{pT,'pT'});
```

```
% ------------------------------------------------------------------------
---
%Linearization about initial position
% q = [theta1 theta2 dtheta1 dtheta2];
f = simplify(D_mtx\(-C_vec - G_vec + [0 0 e]' - B_mtx*[dtheta1;
dtheta2; im;]));
subs_vec = [theta1 theta2 dtheta1 dtheta2 im];
lin_point = [0 0 0 0 0];
A11 = simplify(subs(diff(f(1),theta1),subs_vec,lin_point));
A12 = simplify(subs(diff(f(1),theta2),subs_vec,lin_point));
A13 = simplify(subs(diff(f(1),dtheta1),subs_vec,lin_point));
A14 = simplify(subs(diff(f(1),dtheta2),subs_vec,lin_point));
A15 = simplify(subs(diff(f(1),im),subs_vec,lin_point));

A21 = simplify(subs(diff(f(2),theta1),subs_vec,lin_point));
A22 = simplify(subs(diff(f(2),theta2),subs_vec,lin_point));
A23 = simplify(subs(diff(f(2),dtheta1),subs_vec,lin_point));
A24 = simplify(subs(diff(f(2),dtheta2),subs_vec,lin_point));
A25 = simplify(subs(diff(f(2),im),subs_vec,lin_point));

A31 = simplify(subs(diff(f(3),theta1),subs_vec,lin_point));
A32 = simplify(subs(diff(f(3),theta2),subs_vec,lin_point));
A33 = simplify(subs(diff(f(3),dtheta1),subs_vec,lin_point));
A34 = simplify(subs(diff(f(3),dtheta2),subs_vec,lin_point));
A35 = simplify(subs(diff(f(3),im),subs_vec,lin_point));

B1 = subs(diff(f(1),e),subs_vec,lin_point);
B2 = subs(diff(f(2),e),subs_vec,lin_point);
B3 = subs(diff(f(3),e),subs_vec,lin_point);

%State Space including motor: x = [theta1 theta dtheta1 dtheta2 i]
ssA = [[0 0 1 0 0];
       [0 0 0 1 0];
       [A11 A12 A13 A14 A15];
       [A21 A22 A23 A24 A25];
       [A31 A32 A33 A34 A35];];
ssB = [0; 0; B1; B2; B3;];
```

APPENDIX B: A AND B MATRIX DETAILED EQUATIONS

Thus the values for the A-matrix and B-matrix are:

A11 = 0
A12 = -(g*l2*m2*(Ixz2 - L1*l2*m2))/(I2x*I1z - Ixz2^2 + I2x*I2z + I2x*L1^2*m2 + I2x*l1^2*m1 + I1z*l2^2*m2 + I2z*l2^2*m2 + 2*Ixz2*L1*l2*m2 + l1^2*l2^2*m1*m2)
A13 = -(b1*(m2*l2^2 + I2x))/(I2x*I1z - Ixz2^2 + I2x*I2z + I2x*L1^2*m2 + I2x*l1^2*m1 + I1z*l2^2*m2 + I2z*l2^2*m2 + 2*Ixz2*L1*l2*m2 + l1^2*l2^2*m1*m2)

A14 = (b2*(Ixz2 - L1*l2*m2))/(I2x*I1z - Ixz2^2 + I2x*I2z + I2x*L1^2*m2 + I2x*l1^2*m1 + I1z*l2^2*m2 + I2z*l2^2*m2 + 2*Ixz2*L1*l2*m2 + l1^2*l2^2*m1*m2)
A15 = (kt*(m2*l2^2 + I2x))/(I2x*I1z - Ixz2^2 + I2x*I2z + I2x*L1^2*m2 + I2x*l1^2*m1 + I1z*l2^2*m2 + I2z*l2^2*m2 + 2*Ixz2*L1*l2*m2 + l1^2*l2^2*m1*m2)
A21 = 0
A22 = (g*l2*m2*(m2*L1^2 + m1*l1^2 + I1z + I2z))/(I2x*I1z - Ixz2^2 + I2x*I2z + I2x*L1^2*m2 + I2x*l1^2*m1 + I1z*l2^2*m2 + I2z*l2^2*m2 + 2*Ixz2*L1*l2*m2 + l1^2*l2^2*m1*m2)
A23 = (b1*(Ixz2 - L1*l2*m2))/(I2x*I1z - Ixz2^2 + I2x*I2z + I2x*L1^2*m2 + I2x*l1^2*m1 + I1z*l2^2*m2 + I2z*l2^2*m2 + 2*Ixz2*L1*l2*m2 + l1^2*l2^2*m1*m2)
A24 = -(b2*(m2*L1^2 + m1*l1^2 + I1z + I2z))/(I2x*I1z - Ixz2^2 + I2x*I2z + I2x*L1^2*m2 + I2x*l1^2*m1 + I1z*l2^2*m2 + I2z*l2^2*m2 + 2*Ixz2*L1*l2*m2 + l1^2*l2^2*m1*m2)
A25 =-(kt*(Ixz2 - L1*l2*m2))/(I2x*I1z - Ixz2^2 + I2x*I2z + I2x*L1^2*m2 + I2x*l1^2*m1 + I1z*l2^2*m2 + I2z*l2^2*m2 + 2*Ixz2*L1*l2*m2 + l1^2*l2^2*m1*m2)
A31 = 0
A32 = 0
A33 = -kt/Lm
A34 = 0
A35 = -Rm/Lm
B1 = 0
B2 = 0
B3 = 1/Lm

APPENDIX C: MATLAB CODE FOR SIMULATION

```
clc;
close all;
clear AXIS TIP KIN POT

%List of parameters
global p K L TAU ssA ssB ssC ssD t0 theta10 theta20 m20 g0 l20
g0 = 9.8;
b10 = 6e-4;
b20 = 5.52e-04;
m10 = 0.1862;
m20 = 0.0391 + 0.0259;
L10 = (51+44.55)*0.001;
l10 = -34.78e-3;
L20 = 0.2983;
l20 = 58.47e-3;
I1x0 = 0;
I1y0 = 0;
I1z0 = 3.13e-4 + 0.001;
I2x0 = 5.34e-4;
```

```
I2y0 = 8.41e-4;
I2z0 = 3.1e-4;
Ixz20 = -2.4e-4;
kt0 = 0.182;
Rm0 = 3.6;
Lm0 = 1.845e-3;
Jm0 = 0.001;

p = [g0 L10 l10 m10 I1x0 I1y0 I1z0 L20 l20 m20 I2x0 I2y0 I2z0 b10 b20
Ixz20...
    kt0 Rm0 Lm0 Jm0];

%Linear State Spate model: just measure angles
ssA = eval(subs(ssA,[g L1 l1 m1 I1x I1y I1z L2 l2 m2 I2x I2y I2z b1 b2
Ixz2 kt Rm Lm Jm],p));
ssB = eval(subs(ssB,[g L1 l1 m1 I1x I1y I1z L2 l2 m2 I2x I2y I2z b1 b2
Ixz2 kt Rm Lm Jm],p));
ssC = [[1 0 0 0 0];
       [0 1 0 0 0];
       [0 0 0 0 1]];
ssD = [0; 0; 0];

%Controller design:
%Pole placement
Poles = [-5 -20 -30 -40 -50];
K = place(ssA, ssB, Poles);

%Linear Quadratic Regulator
Q = diag([.01 0.1 0.05 0.05 1]);
R = 1;
K = lqr(ssA,ssB,Q,R);

%Observer design: 8x faster than close-loop poles
Pob = 8*eig(ssA-ssB*K);
L = place(ssA', ssC', Pob);
L = L';

t0 = 0;
TAU = [];
%Relative coordinates initial conditions q(0) = [theta10 theta20
dtheta10 dtheta20];
%Four last variables are estimated
q0 = [0 pi .2 .2 0]';
q0 = [q0 q0];
theta10 = q0(1);
theta20 = q0(2);
[t, Qp] = ode45('Pendulum', [0 15], q0);

%Camera position rotation
Qp(:,1) = Qp(:,1) - pi/2 - pi/4 - 0.2;

for i=1:length(Qp)

    AXIS(i,:) = pAxis(Qp(i,1:2),p);
    TIP(i,:) = pTip(Qp(i,1:2),p);
```

```
    KIN(i) = KinEnergy(Qp(i,1:2),Qp(i,3:4),p);
    POT(i) = PotEnergy(Qp(i,1:2),p);

end

f = figure;
set(f, 'doublebuffer', 'on');
 for k = 1:1:length(Qp)

    %Desenhando
    plot3([0 0 -.5*AXIS(k,1) AXIS(k,1) TIP(k,1)], [0 0 -.5*AXIS(k,2)
AXIS(k,2) TIP(k,2)], [-0.05 0 AXIS(k,3) AXIS(k,3) TIP(k,3)], 'k',
'LineWidth', 2)
    hold on
    axis equal
    axis ([-1.1*L10 1.1*L10 -1.1*L10 1.1*L10 -1.1*L10 1.1*L10]);
    grid on

    hold off
    drawnow;

 end

figure()
plot(TAU(1,:),TAU(2,:),'k')
title('Applied Voltage')
ylabel('Voltage input e [V]')
xlabel('Time [s]')

figure()
plot(t,Qp(:,1),'k')
hold on
plot(t,Qp(:,2),'r')
title('Angular position for pendulum and base')
ylabel('Angular position [rad]')
xlabel('Time [s]')
legend('\theta_1','\theta_2')

figure()
plot(t,Qp(:,3),'k')
hold on
plot(t,Qp(:,8),'r')
plot(t,Qp(:,4),'b')
plot(t,Qp(:,9),'y')
title('Estimated velocities')
legend('Real','Estimation')
ylabel('Angular velocity [rad/s]')
xlabel('Time [s]')
legend('d\theta_1','d\theta_1 hat','d\theta_2','d\theta_2 hat')

function dq = Pendulum(t, q)
global p K L TAU ssA ssB ssC t0 theta10 theta20 m20 g0 l20
%Evaluating the dynamic equations matrices
%D(q)ddq + C(q,dq) + G(q) = tau
```

```
D = D_matrix([q(1:2); q(5)],p);
C = C_vector([q(1:2); q(5)],q(3:4),p);
G = G_vector([q(1:2); q(5)],p);

%Estimation based controller
% xhat = q(6:10);
% e = -K*xhat;

%Joint torques
%Full state Feedback
e = -K*q(1:5);

%Swing-up
Energy = PendEnergy(q(1:2),q(3:4),p);
Ed = m20*g0*l20;
k = 5;
e = -k*q(4)*(Ed - Energy);

%Swing-up + LQR
if(cos(q(2)) > 0.95)
    e = -K*q(1:5);
else
    e = -k*q(4)*(Ed - Energy);
end

%Swing-down
if (t > 10)
    e = 1.5*k*q(4)*(Ed - Energy);
end

%Saturation
Saturation = 12;
if (abs(e) > Saturation)
    e = sign(e)*Saturation;
end

TAU = [TAU [t e]'];
tau = [0 0 e]';

%Model Estimation
dxhat = ssA*q(6:10) + ssB*e + L*(ssC*q(1:5) - ssC*q(6:10));

%Derivative of the state vector
dq = D\(tau - G - C);

%New vector of variables q = [x xhat]
dq = [q(3); q(4); dq; dxhat;];
end
```