# AMME5520 Inverted Pendulum Tutorial

Jack Umenberger Ian R. Manchester

May 8, 2017

## Contents

## 1 Background

In this tutorial you will be designing a controller to stabilize a rotary pendulum in the upright equilibrium position.

### 1.1 Pendulum model

The rotary pendulum model is shown in 1. The rotary arm pivot is attached to the QUBE-Servo system and is actuated. The arm has a length of $L_r$, a moment of inertia of $J_r$, and its angle, $\theta$, increases positively when it rotates counter-clockwise (CCW). The servo (and thus the arm) should turn in the CCW direction when the control voltage is positive, i.e. $V_m > 0$.

The pendulum link is connected to the end of the rotary arm. It has a total length of $L_p$ and it center of mass is $\frac{L_p}{2}$. The moment of inertia about its center of mass is $J_p$. The inverted pendulum angle, $\alpha$, is zero when it is hanging downward and increases positively when rotated CCW.

In the previous tutorial, we derived the equations of motion for the rotary pendulum using the Euler-Lagrange equation:

$$\frac{\partial^2 L}{\partial t \partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q_i$$

The variables $q_i$ are called generalized coordinates. For this system let

$$q(t) = (\theta(t), \alpha(t)) \tag{1}$$

where, as shown in (1), $\theta(t)$ is the rotary arm angle and $\alpha(t)$ is the inverted pendulum angle. The Lagrangian of the system described is

$$L = T - V$$

where $T$ is the total kinetic energy of the system and $V$ is the total potential energy of the system. Thus the Lagrangian is the difference between a system's kinetic and potential energies. The generalized forces $Qi$ are used to describe the non-conservative forces (e.g., friction) applied to a system with respect to the generalized coordinates. In this case, the generalized force acting on the rotary arm is

$$Q_1 = \tau - D_r \dot{\theta}$$

and acting on the pendulum is

$$Q_2 = -D_p \dot{\alpha}.$$

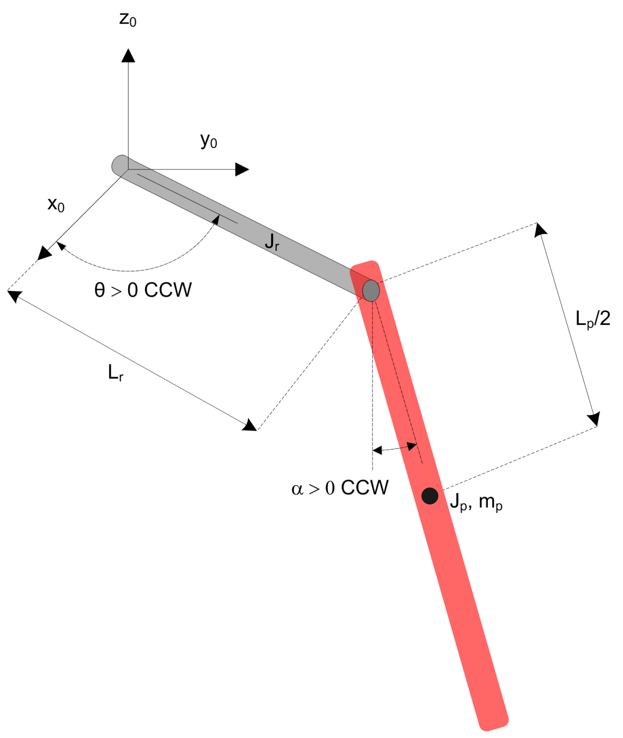

Figure 1: Rotary inverted pendulum model.

Our control variable is the input servo motor voltage, $V_m$. Opposing the applied torque is the viscous friction torque, or viscous damping, corresponding to the term $D_r$. Since the pendulum is not actuated, the only force acting on the link is the damping. The viscous damping coefficient of the pendulum is denoted by $D_p$.

Applying the Euler-Lagrange equations to the rotary pendulum yields the following nonlinear dynamics:

$$\begin{aligned}\left(m_pL_r^2 + \frac{1}{4}m_pL_p^2 - \frac{1}{4}m_pL_p^2\cos(\alpha)^2 + J_r\right)\ddot{\theta} + \left(\frac{1}{2}m_pL_pL_r\cos(\alpha)\right)\ddot{\alpha} \\ + \left(\frac{1}{2}m_pL_p^2\sin(\alpha)\cos(\alpha)\right)\dot{\theta}\dot{\alpha} - \left(\frac{1}{2}m_pL_pL_r\sin(\alpha)\right)\dot{\alpha}^2 &= \tau - D_r\dot{\theta} \quad (2)\\ \frac{1}{2}m_pL_pL_r\cos(\alpha)\ddot{\theta} + \left(J_p + \frac{1}{4}m_pL_p^2\right)\ddot{\alpha} - \frac{1}{4}m_pL_p^2\cos(\alpha)\sin(\alpha)\dot{\theta}^2 \\ + \frac{1}{2}m_pL_pg\sin(\alpha) &= -D_p\dot{\alpha} \quad (3)\end{aligned}$$

with an applied torque at the base of the rotary arm generated by the servo motor as described by the equation:

$$\tau = \frac{k_m(V_m - k_m\dot{\theta})}{R_m}. \quad (4)$$

The Matlab script `lagrangianDerivation.m` contains (partially complete) symbolic math code to automatically derive these equations. If you have the time/inclination, it may be worthwhile deriving these equations for yourself, **but for the purpose of this tutorial, you may simply use the nonlinear equations provided above.**

## 1.2 Linearized dynamics

For the purpose of controller design, we wish to obtain a linearized model of the above nonlinear system dynamics; i.e., we seek a model of the form

$$\dot{x} = Ax + Bu \quad (5a)$$

$$y = Cx + Du \quad (5b)$$

where

$$x = (\theta, \alpha, \dot{\theta}, \dot{\alpha})$$

and

$$y = (\theta, \alpha).$$

There are a couple of ways you might go about doing this: you may wish to make use of the representation

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = Ru$$

to convert the dynamics to the form $\dot{x} = f(x, u)$, and then compute the Jacobians (partial derivatives) of $f$. Alternatively, you might find it simpler to linearize equations (2) and (3) directly; c.f. §3 for some hints.

## 2 Exercises

You should complete the following exercises in **groups of two**.

**Note:** Please open all Matlab/Simulink files in Matlab r2013b.

### 2.1 Modeling

a. Given the nonlinear dynamics in (2) and (3), derive a linearized state-space model of the form (5). Modify the code in the Matlab script `rotaryPendulum.m` so that the linear system matrices $A, B, C, D$ are automatically generated for **any** linearization point $x_0$.

   *Note: it is sufficient for your script to compute the linearization about just two points: the upright equilibirum and downward equilibrium.*

b. Compute the linearized system about the downward equilibrium point $x_0 = (0, 0, 0, 0)$, and run the script `rotaryPendulum.m` to populate the Matlab workspace with $A, B, C, D$. Use Matlab to check the eigenvalues of $A$; is the linear model stable? Why/why not?

### 2.2 Model validation

For this section, we will be comparing our mathematical model to actual (physical) rotary pendulum. Again, you may work in groups of two; once you've formed a group, you can come and collect some hardware.

a. Within the group, compare your linear models for the rotary pendulum about the downward equilibrium position to ensure consistency.

b. Make sure $A, B, C, D$ are in the Matlab workspace.

c. Open the Simulink model `q_ss_model_step`, which applies a 1Hz square wave to both the linearized model and the actual rotary pendulum. Build, connect to and then run the Simulink model, and examine the output from the scopes. Does the model represent the actual pendulum faithfully? Are there any parameters you can tweak to improve the fit (e.g. damping coefficients)?

### 2.3 Controller design

a. Using `rotaryPendulum.m`, recompute the linearization about the vertical equilibrium position $x_0 = (0, \pi, 0, 0)$. Use Matlab to check the eigenvalues of $A$; is the linear model stable? Why/why not?

b. Modify the script `rotaryPendulum.m` to design a LQR to stabilize the vertical equilibrium position. First, choose $Q = \text{diag}(1, 1, 1, 1)$ and $R = 1$, and compute the gain $K$. Next try $Q = \text{diag}(5, 1, 1, 1)$ and $R = 1$ and recompute $K$. How does the value of $Q$ affect the controller gain?

c. Make sure the gain $K$ resulting from $Q = \text{diag}(5, 1, 1, 1)$ and $R = 1$ is in the workspace. Open the Simulink model `q_qube_bal_lqr`, which applies this LQR controller to the actual rotary pendulum. Build, connect to and run the model. Manually rotate the pendulum into the upright position until the controller engages.

d. Once the pendulum is balanced, set the Gain to 30 to make the arm angle go between ±30 deg.

e. Experiment with different controller gains, by adjusting the diagonal elements of the $Q$ matrix to reduce how much the pendulum angle deflects (i.e., overshoots) when the arm angle changes.

f. Adjust $Q$ so that the rotary arm tracks the square wave reference signal with a settling time of 1 second. Hint: consider using a 'sliding mode' cost for the rotary arm.

**Be sure to save your work; future assignments (may) make use of these scripts.**

## 3 Hints

This section recaps the linearization method that was presented during the tutorial.

### 3.1 Linearizing a nonlinear mapping

The first step involves the linearization of the equation of motion in (2) and (3). Let's proceed by way of example. Begin by rewriting (2) in the form $f_1(z) = \tau$, where $z = (\ddot{\theta}, \ddot{\alpha}, \dot{\theta}, \dot{\alpha}, \theta, \alpha)$. Explicitly, we have

$$\left(m_p L_r^2 + \frac{1}{4} m_p L_p^2 - \frac{1}{4} m_p L_p^2 \cos(\alpha)^2 + J_r\right)\ddot{\theta} + \left(\frac{1}{2} m_p L_p L_r \cos(\alpha)\right)\ddot{\alpha} \\ + \left(\frac{1}{2} m_p L_p^2 \sin(\alpha)\cos(\alpha)\right)\dot{\theta}\dot{\alpha} - \left(\frac{1}{2} m_p L_p L_r \sin(\alpha)\right)\dot{\alpha}^2 + D_r\dot{\theta} = \tau$$

To linearize, we find the first order (multivariate) Taylor series of $f_1(z)$, about some expansion point $z_0$, i.e.,

$$f_1(z) \approx f_1(z_0) + \nabla_z f_1(z_0)(z - z_0)$$

where $\nabla_z f_1(z_0)$ is the Jacobian or vector of partial derivatives, w.r.t $z$, evaluated at $z = z_0$

$$\nabla_z f_1(z_0) = \frac{\partial f_1(z_0)}{\partial z} = \left[\frac{\partial f_1(z_0)}{\partial \ddot{\theta}}, \frac{\partial f_1(z_0)}{\partial \ddot{\alpha}}, \cdots, \frac{\partial f_1(z_0)}{\partial \alpha}\right].$$

The partial derivatives are computed in the usual way, e.g.,

$$\begin{aligned} \frac{\partial f_1(\ddot{\theta})}{\partial \ddot{\theta}} &= m_p L_r^2 + \frac{1}{4} m_p L_p^2 - \frac{1}{4} m_p L_p^2 \cos(\alpha)^2 + J_r \\ \frac{\partial f_1(\ddot{\alpha})}{\partial \ddot{\alpha}} &= \frac{1}{2} m_p L_p L_r \cos(\alpha) \\ &\vdots \end{aligned}$$

Ideally, your code would compute the linearization about an arbitrary point $z_0$ and you're encouraged to give this a go. However, as we observed during the tutorial, if we linearize about an equilibrium point it is often quite straightforward to infer which partial derivatives go to zero, simply by inspection.

For instance, suppose we wish to linearize about the downward equilibrium position, $z_0 = (0, 0, 0, 0, 0, 0)$, and consider the partial derivative w.r.t $\alpha$,

$$\frac{\partial f_1(z_0)}{\partial \alpha} = \underbrace{\frac{1}{2} m_p L_p^2 \sin(\alpha)\cos(\alpha)\ddot{\theta}}_{=0 \text{ as } \ddot{\theta}=0} - \underbrace{\frac{1}{2} m_p L_p L_r \sin(\alpha)\ddot{\alpha}}_{=0 \text{ as } \ddot{\alpha}=0} + \underbrace{\frac{1}{2} m_p L_p^2 \cos(2\alpha)\dot{\theta}\dot{\alpha}}_{=0 \text{ as } \dot{\theta}=\dot{\alpha}=0} - \underbrace{\frac{1}{2} m_p L_p L_r \cos(\alpha)\dot{\alpha}^2}_{=0 \text{ as } \dot{\alpha}=0} = 0.$$

As the velocity $(\dot{\theta}, \dot{\alpha})$ and acceleration $(\ddot{\theta}, \ddot{\alpha})$ are zero at any equilibrium point, any terms of the partial derivative that multiply with velocity or acceleration can be set to zero immediately.

Let's continue with the example of linearizing about the downward equilibrium. Applying the above procedure to both (2) and (3) should lead to expressions

$$\frac{\partial f_1(z_0)}{\partial \ddot{\theta}}\ddot{\theta} + \frac{\partial f_1(z_0)}{\partial \ddot{\alpha}}\ddot{\alpha} + \cdots + \frac{\partial f_1(z_0)}{\partial \alpha}\alpha = \tau$$

and

$$\frac{\partial f_2(z_0)}{\partial \ddot{\theta}}\ddot{\theta} + \frac{\partial f_2(z_0)}{\partial \ddot{\alpha}}\ddot{\alpha} + \cdots + \frac{\partial f_2(z_0)}{\partial \alpha}\alpha = 0.$$

To convert this to the standard linear form (5), we can first write these two equations in a 'matrix form'

$$\underbrace{\begin{bmatrix} \frac{\partial f_1(z_0)}{\partial \ddot{\theta}} & \frac{\partial f_1(z_0)}{\partial \ddot{\alpha}} \\ \frac{\partial f_2(z_0)}{\partial \ddot{\theta}} & \frac{\partial f_2(z_0)}{\partial \ddot{\alpha}} \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} \ddot{\theta} \\ \ddot{\alpha} \end{bmatrix}}_{\ddot{q}} + \underbrace{\begin{bmatrix} \frac{\partial f_1(z_0)}{\partial \dot{\theta}} & \frac{\partial f_1(z_0)}{\partial \dot{\alpha}} \\ \frac{\partial f_2(z_0)}{\partial \dot{\theta}} & \frac{\partial f_2(z_0)}{\partial \dot{\alpha}} \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} \dot{\theta} \\ \dot{\alpha} \end{bmatrix}}_{\dot{q}} + \underbrace{\begin{bmatrix} \frac{\partial f_1(z_0)}{\partial \theta} & \frac{\partial f_1(z_0)}{\partial \alpha} \\ \frac{\partial f_2(z_0)}{\partial \theta} & \frac{\partial f_2(z_0)}{\partial \alpha} \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} \theta \\ \alpha \end{bmatrix}}_{q} = \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{R} \tau.$$

By solving for $\ddot{q}$ we have

$$\ddot{q} = -M^{-1}V\dot{q} - M^{-1}Gq + M^{-1}R\tau.$$

Finally, we are ready to represent these dynamics in 'standard form'. Choosing our state variable to be $x = (\theta, \alpha, \dot{\theta}, \dot{\alpha}) = (q, \dot{q})$ and our control input to be $u = \tau$ we have

$$\underbrace{\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} 0 & I \\ -M^{-1}G & -M^{-1}V \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} q \\ \dot{q} \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 0 \\ M^{-1}R \end{bmatrix}}_{B} \tau.$$