**School of Mathematics, Computer Science and Engineering**

**Department of Electrical and Electronic Engineering**

# Modelling, Simulation and Control System Design of a Real Rotary Inverted Pendulum

by

Zhonghe Jiang

Project for the Degree of BEng (Honours) in Electrical and Electronic Engineering (Avionics and Control)

Supervisor: Dr Efstathios Milonidis

London
4 May 2018

# Abstract

The rotary inverted pendulum (RIP) is an underactuated mechanical system exhibiting open-loop unstable and extremely nonlinear dynamic behaviour. This thesis investigates the modelling and control of a real RIP system to swing up the pendulum and balance it in the inverted position. The model is developed by Lagrangian equations of motion. Estimation of unknown parameters in the model is accomplished by extended Kalman filter (EKF) and nonlinear least squares. A simplified energy-based method is used to swing up the pendulum by arm movements. Stabilization at the inverted position is achieved by a double-PD-loop controller which is also proved to be a state-feedback controller. Simulation is conducted in Simulink and control algorithms are implemented in a STM32 microcontroller. Methods discussed in this report work successfully in a real RIP system.

## Acknowledgements

I would like to express my great appreciation to my parents for their love and support during my study at City, University of London.

I am particularly grateful for the assistance provided by my supervisor Dr Efstathios Milonidis and for his valuable suggestions for the completion of this project.

# Table of Contents

# Chapter 1 Introduction

## 1.1 Project Overview

The rotary inverted pendulum, or Furuta pendulum, is a well-known system invented by Furuta et al in 1992. The inverted position of the system is unstable in an open-loop configuration. The system exhibits extremely nonlinear dynamic behaviour and is an underactuated mechanical system with two degrees of freedom and only one actuator. These features make it an excellent experimental platform for testing control algorithms. The system has many practical applications like aircraft stabilization in the turbulent air-flow, stabilization of a cabin in a ship, balance of a rocket during vertical take-off and keeping a walking bipedal robot in the upright position.

This project explores the modelling and control system design of a real rotary inverted pendulum system. Primarily, there are three control problems to be solved:

- Swing-up control which gradually swings the pendulum to the inverted position.
- Stabilizing control which balances the pendulum in the unstable upright position.
- Switching control which catches the pendulum around the inverted position by switching accurately between swing-up and stabilizing control.

To solve the above problems, the mathematical model of the RIP system is first derived to help understand the system dynamics. This is achieved by using Lagrangian approach and a simplified DC motor model. Since there are some unknown parameters in the model which cannot be measured directly, parameter estimation is done to identify those parameters. An extended Kalman filter is designed to estimate those parameters only related to the pendulum. The other parameters are determined using built-in nonlinear least squares algorithm in Simulink Design Optimization™. Two simulation models are created in the Simulink environment: one approach is to use the derived mathematical model to build an analog-computer block diagram and the other is to use Simscape™ to build the physical system.

Once all parameters of the system are known, the theoretical analysis of the control methods is conducted. The swing-up strategy used in this project is a simplified energy-based method which gradually adds energy into the pendulum by movements of the arm at the right time and in the right direction. The moving range of the pendulum is divided into the swing-up and stabilizing zone, and a simple switching control is implemented to switch between these two zones. Stabilization at the inverted position is achieved by a double-PD-loop controller which is developed by intuition and tuned by trial and error. One negative feedback loop controls the angle of the pendulum and another positive feedback loop controls the angle of the arm. These two control inputs are summed up to give the actual control input to the system. This structure is later proved to be a state feedback controller and a detailed theoretical control system design is then carried out and verified in the Simulink environment. The basic idea is to linearize the nonlinear model around the inverted equilibrium point and design a state feedback controller via linear–quadratic regulator (LQR). The stabilizing gain designed using LQR also indirectly proves the double-PD-loop structure.

## 1.2 Project Objectives

The main objectives of this project include:

- To analyse the rotary inverted pendulum apparatus in detail.
- To derive a mathematical model of the apparatus.
- To estimate the unknown parameters of the system as accurate as possible.
- To create simulation models of the system in Simulink
- To design a swing-up controller which drives the pendulum to the upright position.
- To design a stabilizing controller which maintains the pendulum in the upright position.
- To design a switching controller which can switch properly between swing-up and stabilizing control.
- To implement the control algorithms in a microcontroller.
- To test the entire apparatus and adjust the program to satisfy the requirements of the project.

The requirements of the project:

- The controller can swing up the pendulum and balance it in the upright position within 20 seconds.
- The controller can maintain the pendulum in the upright position for at least 10 seconds.
- The pendulum can still keep upright after being hit by a weight (about 5 g) which is suspended using a 15-cm thread and released at an angle of 30 degrees.

## 1.3 Report Structure

Chapter 2 contains the mathematical modelling of the RIP system, with the details of building Simulink model in Section 2.5. Chapter 3 describes the details of parameter identification including designing extended Kalman filter for parameter estimation. Chapter 4 explores the control system design of the real RIP system. The methods discussed in this chapter are also implemented in the microcontroller to successfully swing up the pendulum and balance it in the upright position. Stabilization control using double-PD-loop controller is covered in Section 4.3. Chapter 5 gives the theoretical analysis of the stabilizing problem using state feedback via LQR. Finally, Chapter 6 presents conclusions and recommendations for future improvements.

# Chapter 2 Mathematical Modelling

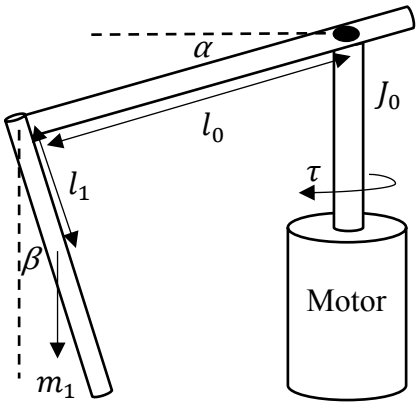


*Fig. 2.1. Schematic drawing of the RIP system*

The system consists of a horizontal link (the rotating arm) and a vertical link (the pendulum). The arm is driven by a motor with encoder and rotates about a vertical axis to swing up and balance the pendulum. The pendulum is attached to the end of the arm with a joint and rotates freely in the vertical plane. The mechanical structure of the RIP is simpler and easier to construct than the cart-pendulum system because the motor directly drives the arm and they can be connected using only revolute joints. However, the dynamic behaviour of the RIP is more complex than the cart-pendulum system.

The system can be represented as the following block diagram:

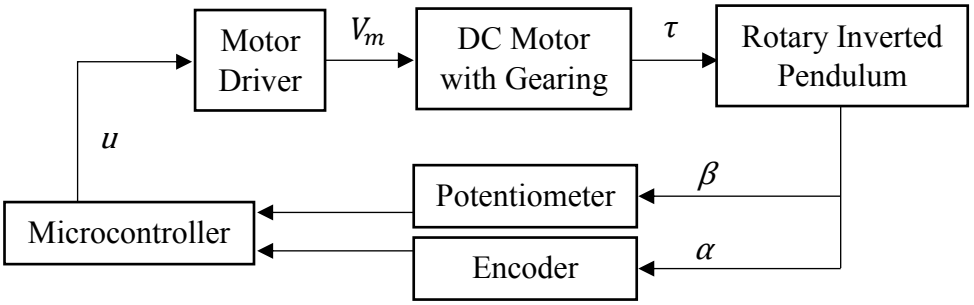


*Fig. 2.2. Block diagram of the RIP system*

## 2.1 Derivation of the Equations of Motion

Here gives a method of deriving the equations of motion using Lanrangian dynamics, the details of which can be found in [1].

The system has two degrees of freedom: the rotation of the arm and the rotation of the pendulum. A set of generalized coordinates is defined so that the position of every particle in the system is a function of these coordinates:

$$\boldsymbol{q} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \tag{2.1}$$

As shown in Fig. 2.1, $\alpha$ is the angle of the actuated arm from its initial position, $\beta$ is the angle of the unactuated pendulum from its downward position.

Using $\boldsymbol{r}$ to denote the position of the centre of mass of the pendulum, the kinematics of the mass $m_1$ are:

$$\boldsymbol{r} = \begin{bmatrix} l_0 cos\alpha - l_1 sin\alpha sin\beta \\ l_0 sin\alpha + l_1 cos\alpha sin\beta \\ -l_1 cos\beta \end{bmatrix}, \tag{2.2}$$

$$\dot{\boldsymbol{r}} = \begin{bmatrix} -l_0\dot{\alpha} sin\alpha - l_1\dot{\alpha} cos\alpha sin\beta - l_1\dot{\beta} sin\alpha cos\beta \\ l_0\dot{\alpha} cos\alpha - l_1\dot{\alpha} sin\alpha sin\beta + l_1\dot{\beta} cos\alpha cos\beta \\ l_1\dot{\beta} sin\beta \end{bmatrix}. \tag{2.3}$$

As shown in Fig. 2.1, $l_0$ is the length between the revolute joint and the mass-centre of the pendulum, while $l_1$ is the length between the revolute joint and the shaft of the motor.

The total kinetic energy of the system is written as:

$$\begin{aligned} T &= \frac{1}{2}J_0\dot{\alpha}^2 + \frac{1}{2}J_1'\dot{\beta}^2 + \frac{1}{2}m_1|\dot{\boldsymbol{r}}|^2 \\ &= \frac{1}{2}J_0\dot{\alpha}^2 + \frac{1}{2}J_1'\dot{\beta}^2 + \frac{1}{2}m_1 l_0^2\dot{\alpha}^2 + \frac{1}{2}m_1 l_1^2 \, sin^2\beta \, \dot{\alpha}^2 \\ &\quad + \frac{1}{2}m_1 l_1^2\dot{\beta}^2 + m_1 l_0 l_1 cos\beta\dot{\alpha}\dot{\beta}. \end{aligned} \tag{2.4}$$

Here, $J_0$ is the moment of inertia of the arm about the shaft of the motor, while $J_1'$ is the moment of inertia of the pendulum about its mass-centre.

The total potential energy of the system is given by:

$$V = -m_1 g l_1 cos\beta \,. \tag{2.5}$$

To consider the non-conservative generalized forces, take the virtual work along the virtual displacement $\delta \boldsymbol{q} = \begin{bmatrix} \delta\alpha \\ 0 \end{bmatrix}$:

$$\delta W = -(C_0\dot{\alpha})\delta\alpha - \left[K_{f0} sign(\dot{\alpha})\right]\delta\alpha + \tau\delta\alpha = F_0\delta\alpha \,. \tag{2.6}$$

Here, $C_0\dot{\alpha}$ models the viscous friction between the arm and the air, $K_{f0} sign(\dot{\alpha})$ represents the Coulomb friction at the shaft of the motor and $\tau$ is the gearbox output torque applied to the arm. $F_0$ is the generalized force corresponding to $\alpha$.

Then, take the virtual work along the virtual displacement $\delta \boldsymbol{q} = \begin{bmatrix} 0 \\ \delta\beta \end{bmatrix}$:

$$\delta W = -\left[C_1\dot{\beta}\right]\delta\beta - \left[K_{f1} sign\left(\dot{\beta}\right)\right]\delta\beta = F_1\delta\beta \,. \tag{2.7}$$

Here, $C_1\dot{\beta}$ models the viscous friction between the pendulum and the air, while $K_{f1} sign\left(\dot{\beta}\right)$ represents the Coulomb friction at the shaft of the potentiometer. $F_1$ is the generalized force corresponding to $\beta$ .

Therefore, the non-conservative generalized forces can be written as:

$$\boldsymbol{F} = \begin{bmatrix} F_0 \\ F_1 \end{bmatrix} = \begin{bmatrix} -C_0\dot{\alpha} - K_{f0} sign(\dot{\alpha}) + \tau \\ -C_1\dot{\beta} - K_{f1} sign\left(\dot{\beta}\right) \end{bmatrix} . \tag{2.8}$$

The Lagrangian dynamic equations are:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = F_i \tag{2.9}$$

where $L = T - V$ is the Lagrangian function.

Taking the partial derivatives $\frac{\partial L}{\partial \dot{q}_i}$, $\frac{\partial L}{\partial q_i}$, then $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right)$, and substitute them into the Lagrangian, reveals the nonlinear equations of motion of the system:

$$\begin{bmatrix} J_0 + m_1 l_0^2 + m_1 l_1^2 \sin^2\beta & m_1 l_0 l_1 \cos\beta \\ m_1 l_0 l_1 \cos\beta & J_1' + m_1 l_1^2 \end{bmatrix} \begin{bmatrix} \ddot{\alpha} \\ \ddot{\beta} \end{bmatrix}$$
$$+ \begin{bmatrix} C_0 + m_1 l_1^2 \dot{\beta} \sin\beta\cos\beta & m_1 l_1^2 \dot{\alpha} \sin\beta\cos\beta - m_1 l_0 l_1 \dot{\beta} \sin\beta \\ -m_1 l_1^2 \dot{\alpha} \sin\beta\cos\beta & C_1 \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix} \quad (2.10)$$
$$+ \begin{bmatrix} K_{f0} sign(\dot{\alpha}) \\ K_{f1} sign(\dot{\beta}) + m_1 g l_1 \sin\beta \end{bmatrix} = \begin{bmatrix} \tau \\ 0 \end{bmatrix}$$

## 2.2 DC Motor Model

Note that the actual input to the system is not a real torque $\tau$, but an integer $u$ (either positive or negative) with no units representing the pulse width of the PWM signal. In practice, the DC motor in this setup is only capable of producing limited amount of torque. This input constraint on $\tau$ is now transferred to the constraint on $u$, which depends on the hardware used to generate PWM signal. For Arduino UNO R3, a call to analogWrite() is on a scale of 0 - 255, representing 0 – 100% duty cycle. For STM32F103C8, the duty cycle is determined by the value of the TIMx_CCRx register, while the maximum range of $u$ is determined by the value of the TIMx_ARR register which is set to 7200 in this project. This input constraint should be considered in the simulation and final control system design.

To derive the relationship between $\tau$ and $u$, the following DC motor model needs to be considered (the coil inductance of the motor is neglected).

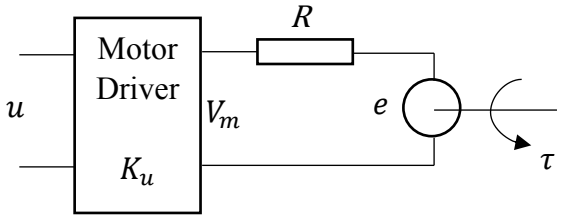


*Fig. 2.3. Schematic drawing of the DC motor model*

Assume the PWM amplifier has a constant gain $K_u$ :

$$V_m = K_u u \,. \quad (2.11)$$

According to Kirchhoff's voltage law:

$$V_m = iR + e\,. \tag{2.12}$$

Here, $R$ is the armature resistance, *i* is the armature current, *e* is the motor's back EMF.

The back EMF $e$ is proportional to the angular velocity of the shaft:

$$e = K_b\dot{\alpha} \tag{2.13}$$

where, $K_b$ is the back-EMF constant.

The torque $\tau_i$ at the input gear is proportional to the armature current:

$$\tau_i = K_t i = K_t\frac{V_m - e}{R} = \frac{K_t K_u}{R}u - \frac{K_t K_b}{R}\dot{\alpha} \tag{2.14}$$

where, $K_t$ is the motor torque constant.

The torque $\tau$ at the output gear is proportional to that at the input gear:

$$\tau = \eta\, K_g \tau_i = \frac{\eta K_g K_t K_u}{R}u - \frac{\eta K_g K_t K_b}{R}\dot{\alpha} \tag{2.15}$$

where, $K_g$ is the gear ratio, $\eta$ is the gearbox efficiency.

## 2.3 Coulomb Friction Model

The Coulomb friction model used in (2.6), (2.7) involves a *sign* function which is discontinuous if the velocity is zero. This discontinuity will cause problem when doing simulation in Simulink. Reference [2] shows that the Simulink software uses a technique known as *zero-crossing detection* to accurately locate a discontinuity without resorting to excessively small time steps. Usually this technique improves simulation run time, but it can cause some simulations to halt before the intended completion time. Actually, the simulation of this mathematical model (2.10) will be terminated caused by hitting too many consecutive zero crossings. If the zero crossing of the Sign block is disabled, the simulation will still be terminated because the variable step solver will continuously decrease the time steps in the vicinity of a discontinuity. Therefore, this problem can only be solved either by using the Adaptive zero-crossing detection algorithm or by approximating the *sign* function using a continuous sigmoid function like:

$$sign(x) \approx f(x) = 1 - \frac{2}{e^{kx} + 1} \tag{2.16}$$

The above function is a good approximation of the *sign* function if *k* is large enough.

The following diagram shows the graph of $f(x)$ when *k*=30 and *sign(x)*.

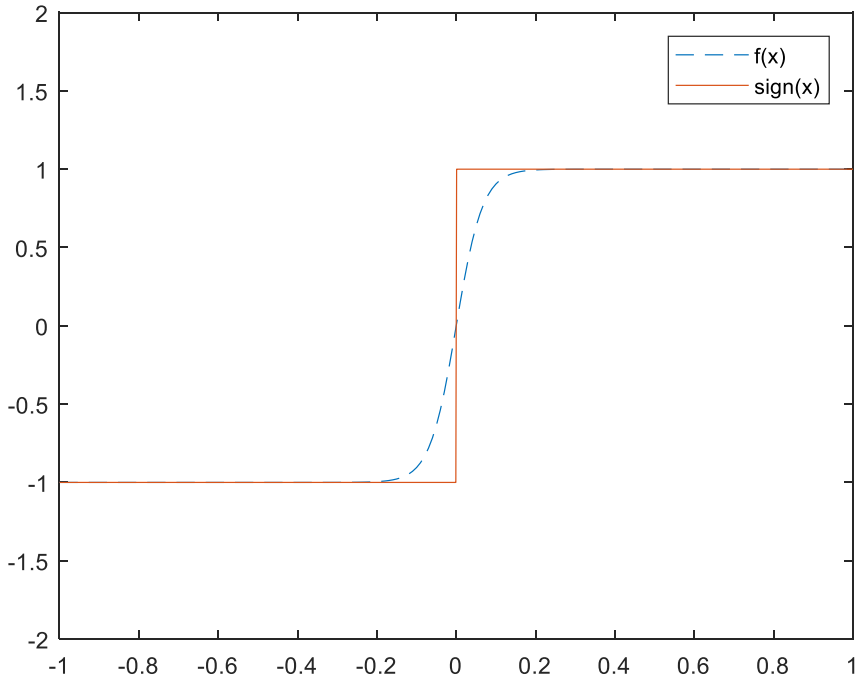


*Fig. 2.4. The graph of sigmoid and sign function*

## 2.4 Proof of Underactuated System

The overall mathematical model is obtained by combining (2.10), (2.15) and (2.16):

$$\begin{bmatrix} J_0 + m_1 l_0^2 + m_1 l_1^2 \sin^2\beta & m_1 l_0 l_1 \cos\beta \\ m_1 l_0 l_1 \cos\beta & J_1' + m_1 l_1^2 \end{bmatrix} \begin{bmatrix} \ddot{\alpha} \\ \ddot{\beta} \end{bmatrix}$$
$$+ \begin{bmatrix} C_0 + \dfrac{\eta K_g K_t K_b}{R} + m_1 l_1^2 \dot{\beta} \sin\beta\cos\beta & m_1 l_1^2 \dot{\alpha} \sin\beta\cos\beta - m_1 l_0 l_1 \dot{\beta} \sin\beta \\ -m_1 l_1^2 \dot{\alpha} \sin\beta\cos\beta & C_1 \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix}$$
$$+ \begin{bmatrix} K_{f0} f(\dot{\alpha}) \\ K_{f1} f(\dot{\beta}) + m_1 g l_1 \sin\beta \end{bmatrix} = \begin{bmatrix} \dfrac{\eta K_g K_t K_u}{R} \\ 0 \end{bmatrix} u \,. \tag{2.17}$$

In the area of robotics, the above mathematical model is in the form of standard "manipulator equations":

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{B}(\boldsymbol{q})\boldsymbol{u} \tag{2.18}$$

where, $\boldsymbol{H}(\boldsymbol{q})$ is called inertial matrix and is always symmetric and positive definite, hence invertible.

The RIP system is also "control affine", which means that the above manipulator equations can be written as:

$$\ddot{\boldsymbol{q}} = \boldsymbol{f_1}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{t}) + \boldsymbol{f_2}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{t})\boldsymbol{u} = -\boldsymbol{H}^{-1}(\boldsymbol{q})[\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q})] + \boldsymbol{H}^{-1}(\boldsymbol{q})\boldsymbol{B}(\boldsymbol{q})\boldsymbol{u} \tag{2.19}$$

Reference [3] shows that a control system described by (2.19) is a fully-actuated system if it is able to command an instantaneous acceleration in an arbitrary direction in $\boldsymbol{q}$. That is rank $[\boldsymbol{f_2}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{t})] = \dim(\boldsymbol{q})$, i.e. $\boldsymbol{f_2}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{t})$ is full row rank. This means that a fully-actuated system can follow arbitrary trajectories in state space.

Because $\boldsymbol{H}^{-1}(\boldsymbol{q})$ is always full rank, then rank $[\boldsymbol{f_2}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{t})] =$ rank $[\boldsymbol{H}^{-1}(\boldsymbol{q})\boldsymbol{B}(\boldsymbol{q})] =$ rank $[\boldsymbol{B}(\boldsymbol{q})]$. This means that the system is fully-actuated if and only if $\boldsymbol{B}(\boldsymbol{q})$ is full row rank. In this case, rank $[\boldsymbol{B}(\boldsymbol{q})] =$ rank $\begin{bmatrix} \frac{\eta K_g K_t K_u}{R} \\ 0 \end{bmatrix} = 1 <$ $\dim(\boldsymbol{q}) = 2$, which proves that the RIP system is indeed underactuated.

## 2.5 Linearization around the Inverted Equilibrium Point

Here gives a method of linearizing nonlinear model in "manipulator equations" form. Taylor series expansion around an equilibrium point:

$$\begin{aligned} \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) &\approx \boldsymbol{f}(\boldsymbol{x_e}, \boldsymbol{u_e}) + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x_e}, \boldsymbol{u}=\boldsymbol{u_e}} (\boldsymbol{x} - \boldsymbol{x_e}) + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\boldsymbol{x}=\boldsymbol{x_e}, \boldsymbol{u}=\boldsymbol{u_e}} (\boldsymbol{u} - \boldsymbol{u_e}) \\ &= \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x_e}, \boldsymbol{u}=\boldsymbol{u_e}} (\boldsymbol{x} - \boldsymbol{x_e}) + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\boldsymbol{x}=\boldsymbol{x_e}, \boldsymbol{u}=\boldsymbol{u_e}} (\boldsymbol{u} - \boldsymbol{u_e}) . \end{aligned} \tag{2.20}$$

where $(\boldsymbol{x_e}, \boldsymbol{u_e})$ is the equilibrium point and hence $\boldsymbol{f}(\boldsymbol{x_e}, \boldsymbol{u_e}) = \boldsymbol{0}$ .

Then equation (2.18) can be written in the state-space form by choosing $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix}$:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{H}^{-1}(\boldsymbol{q})[\boldsymbol{B}(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{G}(\boldsymbol{q})] \end{bmatrix} \tag{2.21}$$

The above nonlinear system can be linearized around the equilibrium $(\boldsymbol{x_e}, \boldsymbol{u_e})$ using formula (2.20):

$$\dot{\boldsymbol{x}} \approx \bar{\boldsymbol{A}}(\boldsymbol{x} - \boldsymbol{x_e}) + \bar{\boldsymbol{B}}(\boldsymbol{u} - \boldsymbol{u_e}) \tag{2.22}$$

where:

$$\bar{\boldsymbol{A}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}=\boldsymbol{x_e},\boldsymbol{u}=\boldsymbol{u_e}}$$

$$= \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \dfrac{\partial \boldsymbol{H}^{-1}}{\partial \boldsymbol{q}}[\boldsymbol{Bu} - \boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{G}] + \boldsymbol{H}^{-1}[\dfrac{\partial \boldsymbol{B}}{\partial \boldsymbol{q}}\boldsymbol{u} - \dfrac{\partial \boldsymbol{C}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} - \dfrac{\partial \boldsymbol{G}}{\partial \boldsymbol{q}}] & -\boldsymbol{H}^{-1}\dfrac{\partial \boldsymbol{C}}{\partial \dot{\boldsymbol{q}}}\boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{H}^{-1}\boldsymbol{C} \end{bmatrix}_{\boldsymbol{x}=\boldsymbol{x_e},\boldsymbol{u}=\boldsymbol{u_e}}$$

$$= \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{H}^{-1}\dfrac{\partial \boldsymbol{G}}{\partial \boldsymbol{q}} & -\boldsymbol{H}^{-1}\boldsymbol{C} \end{bmatrix}_{\boldsymbol{x}=\boldsymbol{x_e},\boldsymbol{u}=\boldsymbol{u_e}}$$

$$\bar{\boldsymbol{B}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\Big|_{\boldsymbol{x}=\boldsymbol{x_e},\boldsymbol{u}=\boldsymbol{u_e}} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{H}^{-1}\boldsymbol{B} \end{bmatrix}_{\boldsymbol{x}=\boldsymbol{x_e},\boldsymbol{u}=\boldsymbol{u_e}}$$

In the above derivation, some terms are zero because:

$$\boldsymbol{f}(\boldsymbol{x_e}, \boldsymbol{u_e}) = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{H}^{-1}(\boldsymbol{q})[\boldsymbol{B}(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{G}(\boldsymbol{q})] \end{bmatrix}_{\boldsymbol{x}=\boldsymbol{x_e},\boldsymbol{u}=\boldsymbol{u_e}} = \boldsymbol{0}$$

$$\Rightarrow [\boldsymbol{Bu} - \boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{G}]_{\boldsymbol{x}=\boldsymbol{x_e},\boldsymbol{u}=\boldsymbol{u_e}} = \dot{\boldsymbol{q}}|_{\boldsymbol{x}=\boldsymbol{x_e},\boldsymbol{u}=\boldsymbol{u_e}} = \boldsymbol{0}$$

Define $\delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x_e}$ and $\delta\boldsymbol{u} = \boldsymbol{u} - \boldsymbol{u_e}$ ,then

$$\delta\dot{\boldsymbol{x}} = \bar{\boldsymbol{A}}\delta\boldsymbol{x} + \bar{\boldsymbol{B}}\delta\boldsymbol{u}$$

Linearize the nonlinear model of the RIP system around the equilibrium point $\boldsymbol{x_e}^T = [0 \quad \pi \quad 0 \quad 0]$ and $u_e = 0$ using the above formula (2.22):

$$\boldsymbol{H}(\boldsymbol{q})|_{\boldsymbol{x}=\boldsymbol{x_e},\boldsymbol{u}=\boldsymbol{u_e}} = \begin{bmatrix} J_0 + m_1 l_0^2 + m_1 l_1^2 \sin^2\beta & m_1 l_0 l_1 cos\beta \\ m_1 l_0 l_1 cos\beta & J_1' + m_1 l_1^2 \end{bmatrix}_{\boldsymbol{x}=\boldsymbol{x_e},\boldsymbol{u}=\boldsymbol{u_e}}$$

$$= \begin{bmatrix} J_0 + m_1 l_0^2 & -m_1 l_0 l_1 \\ -m_1 l_0 l_1 & J_1' + m_1 l_1^2 \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

$$\Rightarrow \boldsymbol{H}^{-1}(\boldsymbol{q})|_{\boldsymbol{x}=\boldsymbol{x_e},\boldsymbol{u}=\boldsymbol{u_e}} = \frac{1}{ac - b^2}\begin{bmatrix} c & -b \\ -b & a \end{bmatrix}$$

$$\boldsymbol{G}(\boldsymbol{q}) = \begin{bmatrix} K_{f0} f(\dot{\alpha}) \\ K_{f1} f(\dot{\beta}) + m_1 g l_1 sin\beta \end{bmatrix}$$

$$\left.\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{q}}\right|_{\boldsymbol{x=x_e,u=u_e}} = \begin{bmatrix} 0 & 0 \\ 0 & m_1 g l_1 cos\beta \end{bmatrix}_{\boldsymbol{x=x_e,u=u_e}} = \begin{bmatrix} 0 & 0 \\ 0 & -m_1 g l_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & e \end{bmatrix}$$

$$\boldsymbol{C(q,\dot{q})}|_{\boldsymbol{x=x_e,u=u_e}}$$

$$= \begin{bmatrix} C_0 + \frac{\eta K_g K_t K_b}{R} + m_1 l_1^2 \dot{\beta} sin\beta cos\beta & m_1 l_1^2 \dot{\alpha}\, sin\beta cos\beta - m_1 l_0 l_1 \dot{\beta} sin\beta \\ -m_1 l_1^2 \dot{\alpha}\, sin\beta cos\beta & C_1 \end{bmatrix}_{\boldsymbol{x=x_e,u=u_e}}$$

$$= \begin{bmatrix} C_0 + \frac{\eta K_g K_t K_b}{R} & 0 \\ 0 & C_1 \end{bmatrix} = \begin{bmatrix} d & 0 \\ 0 & C_1 \end{bmatrix}$$

$$\boldsymbol{B(q)}|_{\boldsymbol{x=x_e,u=u_e}} = \begin{bmatrix} \frac{\eta K_g K_t K_u}{R} \\ 0 \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}$$

$$-\boldsymbol{H^{-1}(q)}\left.\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{q}}\right|_{\boldsymbol{x=x_e,u=u_e}} = \frac{1}{ac-b^2}\begin{bmatrix} c & -b \\ -b & a \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & e \end{bmatrix} = \frac{1}{ac-b^2}\begin{bmatrix} 0 & be \\ 0 & -ae \end{bmatrix}$$

$$-\boldsymbol{H^{-1}(q)C(q,\dot{q})}|_{\boldsymbol{x=x_e,u=u_e}} = -\frac{1}{ac-b^2}\begin{bmatrix} c & -b \\ -b & a \end{bmatrix}\begin{bmatrix} d & 0 \\ 0 & C_1 \end{bmatrix}$$

$$= \frac{1}{ac-b^2}\begin{bmatrix} -cd & bC_1 \\ bd & -aC_1 \end{bmatrix}$$

$$\boldsymbol{H^{-1}(q)B(q)}|_{\boldsymbol{x=x_e,u=u_e}} = \frac{1}{ac-b^2}\begin{bmatrix} c & -b \\ -b & a \end{bmatrix}\begin{bmatrix} f \\ 0 \end{bmatrix} = \frac{1}{ac-b^2}\begin{bmatrix} cf \\ -bf \end{bmatrix}$$

Therefore, the linearized model of the RIP system around the inverted equilibrium point $\boldsymbol{x_e}^T = [0 \quad \pi \quad 0 \quad 0]$ and $u_e = 0$ is as follows:

$$\delta \dot{\boldsymbol{x}} = \bar{\boldsymbol{A}}\delta \boldsymbol{x} + \bar{\boldsymbol{B}}\delta \boldsymbol{u} \tag{2.23}$$

where

$$\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{H^{-1}}\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{q}} & -\boldsymbol{H^{-1}C} \end{bmatrix}_{\boldsymbol{x=x_e,u=u_e}} = \frac{1}{ac-b^2}\begin{bmatrix} 0 & 0 & ac-b^2 & 0 \\ 0 & 0 & 0 & ac-b^2 \\ 0 & be & -cd & bC_1 \\ 0 & -ae & bd & -aC_1 \end{bmatrix}$$

$$\bar{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{H^{-1}B} \end{bmatrix}_{\boldsymbol{x=x_e,u=u_e}} = \frac{1}{ac-b^2}\begin{bmatrix} 0 \\ 0 \\ cf \\ -bf \end{bmatrix}$$

$a = J_0 + m_1 l_0^2$

$b = -m_1 l_0 l_1$

$c = J_1' + m_1 l_1^2$

$$d = C_0 + \frac{\eta K_g K_t K_b}{R}$$

$$e = -m_1 g l_1$$

$$f = \frac{\eta K_g K_t K_u}{R}$$

## 2.5 Simulation Model

The simulation model of the RIP system is constructed in two different ways in this project. One is called an analog-computer block diagram representation, which is based on the mathematical model (2.17). The other is constructing the physical system using Simscape™.

### 2.5.1 Analog-Computer Block Diagram

This block-diagram structure uses integrators as the central elements, where each state of the system is the output of an integrator. The major reason of using integrators is that the initial conditions of the states are automatically introduced into the simulation model. This structure is quite easy to build from state-space representation of the system:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \ddot{\alpha} \\ \ddot{\beta} \end{bmatrix} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{2.24}$$

From the second equation of (2.17), $\ddot{\beta}$ is obtained as:

$$\ddot{\beta} = -\frac{m_1 l_0 l_1 cos\beta}{J_1' + m_1 l_1^2}\ddot{\alpha} + \frac{m_1 l_1^2 sin\beta cos\beta}{J_1' + m_1 l_1^2}\dot{\alpha}^2 - \frac{C_1}{J_1' + m_1 l_1^2}\dot{\beta} - \frac{K_{f1} f(\dot{\beta})}{J_1' + m_1 l_1^2} - \frac{m_1 g l_1 sin\beta}{J_1' + m_1 l_1^2} \tag{2.25}$$

Substituting the above equation into the first equation of (2.17) and solve for $\ddot{\alpha}$ :

$$\ddot{\alpha} = f_1\dot{\alpha} + f_2\dot{\alpha}^2 + f_3 f(\dot{\alpha}) + f_4 + f_5 u \tag{2.26}$$

where:

$$f_1 = -\frac{C_0 + \frac{\eta K_g K_t K_b}{R} + 2m_1 l_1^2 \dot{\beta} sin\beta cos\beta}{J_0 + m_1 l_0^2 + m_1 l_1^2 sin^2\beta - \frac{(m_1 l_0 l_1 cos\beta)^2}{J_1' + m_1 l_1^2}}$$

$$f_2 = -\frac{\frac{{m_1}^2 l_0 {l_1}^3 sin\beta cos^2\beta}{J_1' + m_1 l_1^2}}{J_0 + m_1 l_0^2 + m_1 l_1^2 sin^2\beta - \frac{(m_1 l_0 l_1 cos\beta)^2}{J_1' + m_1 l_1^2}}$$

$$f_3 = -\frac{K_{f0}}{J_0 + m_1 l_0^2 + m_1 l_1^2 sin^2\beta - \frac{(m_1 l_0 l_1 cos\beta)^2}{J_1' + m_1 l_1^2}}$$

$$f_4 = \frac{\frac{C_1 m_1 l_0 l_1 \dot{\beta} cos\beta}{J_1' + m_1 l_1^2} + \frac{{m_1}^2 g l_0 {l_1}^2 sin\beta cos\beta}{J_1' + m_1 l_1^2} + m_1 l_0 l_1 \dot{\beta}^2 sin\beta}{J_0 + m_1 l_0^2 + m_1 l_1^2 sin^2\beta - \frac{(m_1 l_0 l_1 cos\beta)^2}{J_1' + m_1 l_1^2}}$$

$$f_5 = \frac{\frac{\eta K_g K_t K_u}{R}}{J_0 + m_1 l_0^2 + m_1 l_1^2 sin^2\beta - \frac{(m_1 l_0 l_1 cos\beta)^2}{J_1' + m_1 l_1^2}}$$

The first two equations in the state space representation (2.24) can be constructed using the following cascade integrators:

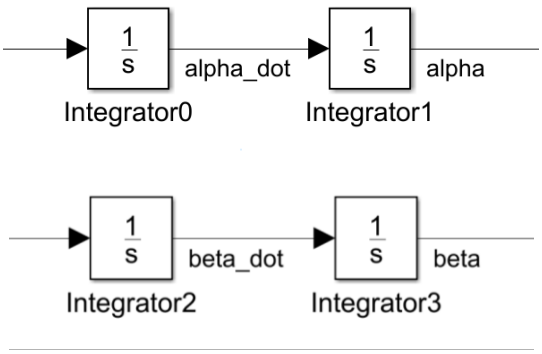


*Fig. 2.5. Cascade integrators*

Note that the inputs to the Integrator0 and Integrator2 are $\ddot{\alpha}$ and $\ddot{\beta}$ respectively. First, $\ddot{\alpha}$ is constructed using (2.26) by feeding back the corresponding functions of states, adding them up and then connected with $\ddot{\alpha}$. Then, $\ddot{\beta}$ is constructed in a similar way using (2.25).

### 2.5.2 Simscape Model

The arm is modelled as a brick using a Solid block and the pendulum is modelled as a cylinder. Both of them are assumed as point mass. One end of the pendulum is connected to the edge of the arm using a Revolute Joint as shown in Fig. 2.6.

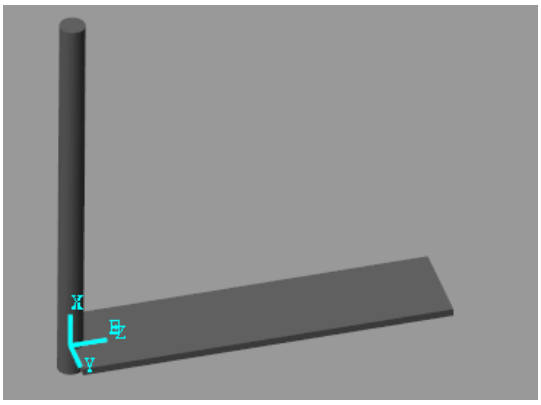


*Fig. 2.6. Revolute joint between the pendulum and the arm*

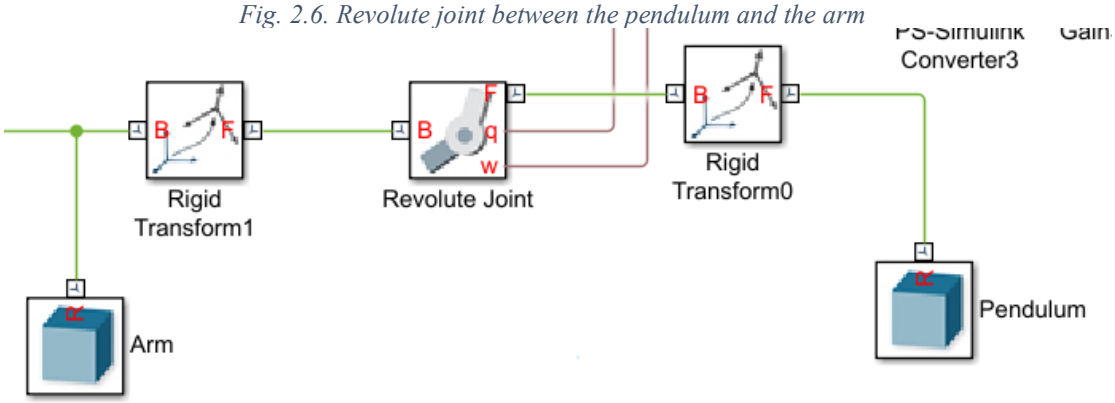


*Fig. 2.7. Block diagram of the Simscape model*

Note that there are two Rigid Transform blocks in the block diagram as shown in Fig. 2.7. The Rigid Transform0 is used to transform a reference frame on the pendulum associated with its geometry to a frame at the joint. The Rigid Transform1 does the same thing for the frame on the arm. The viscous friction $C_1$ is modelled within the Revolute Joint block, but the Coulomb friction $K_{f1}$ is neglected. The angle and angular velocity of the pendulum are sensed by the Revolute Joint block.

The other end of the arm is connected to the shaft of a DC motor using a revolute joint as shown in Fig 2.8.

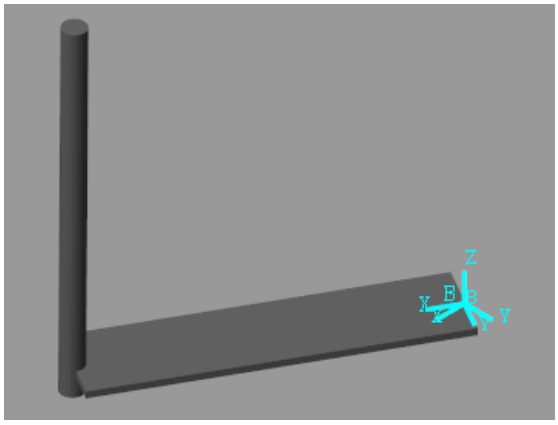

*Fig. 2.8. Revolute joint between the arm and the shaft of the DC motor*

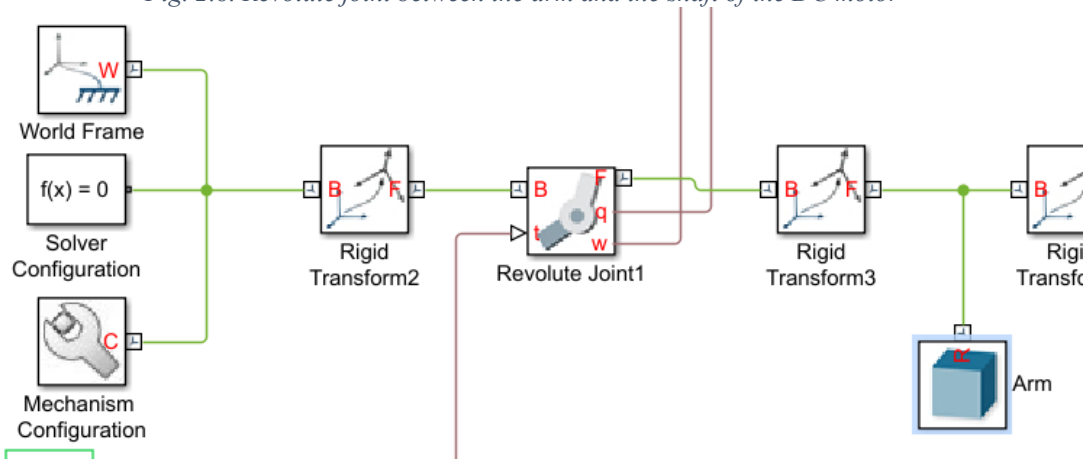


*Fig. 2.9. Block diagram of the Simscape model*

As shown in Fig. 2.9., the solver parameters are specified in the Solver Configuration block and this is the only one Solver Configuration block used in this physical network. The gravity is configured in the Mechanism Configuration block. The World Frame is a global reference frame and is at absolute rest, which is transformed by Rigid Transform2 to give the base frame of the Revolute Joint1. The Rigid Transform3 is used to transform a reference frame on the arm to a frame at the joint, which gives the follower frame of the Revolute Joint1. The viscous friction $C_0$ is modelled within the Revolute Joint1 block, but the Coulomb friction $K_{f0}$ is neglected. The angle and angular velocity of the arm are sensed by the Revolute Joint1 block. The input torque to the Revolute Joint1 block is given by the following DC motor model.

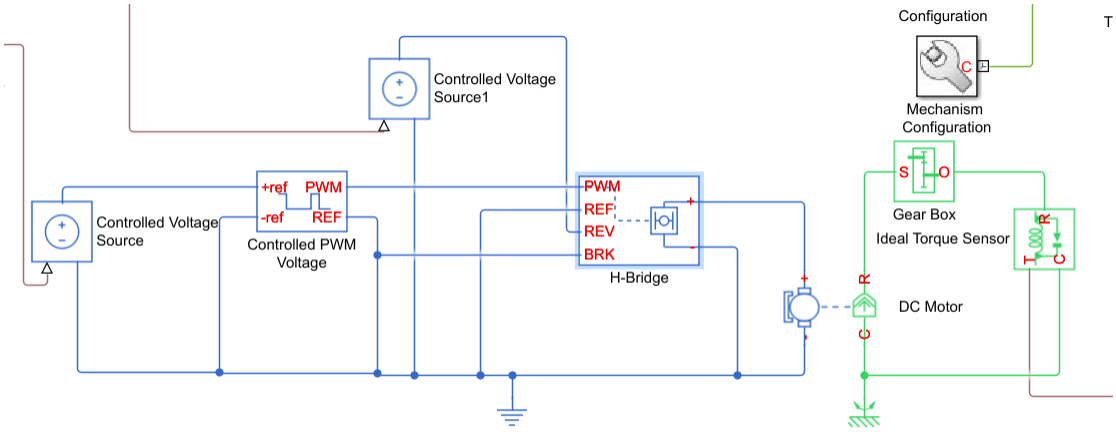


*Fig. 2.10. Block diagram of the Simscape model*

As shown in Fig. 2.10., the Ideal Torque Sensor measures the output torque of the Gear Box and outputs this mechanical signal to the Revolute Joint1. The gear ratio $K_g$ and gear box efficiency $\eta$ are defined in the Gear Box block. The DC Motor block is configured using equivalent circuit parameters $K_b, K_t, R$. Moreover, $J_0$, the moment of inertia of the arm about the shaft of the motor, is totally modelled as the shaft inertia in the DC Motor block.

The H-Bridge block models the motor driver and is driven by the Controlled PWM Voltage block in Averaged mode. In the Controlled PWM Voltage block, the reference voltage Vref across its +ref and -ref ports is given by a Controlled Voltage Source. The input of this voltage source is defined as the absolute value of the input signal $u$, which represents the pulse width (duty cycle) of the PWM signal. The direction of the output torque of the DC motor (i.e. the polarity of the input voltage to the DC motor) is controlled by the REV port voltage in the H-Bridge block. From the user's point of view, if $u > 0$, then the motor is in the forward mode and if $u < 0$, then the motor is in the reserve mode. Therefore, one solution is to connect this REV port to another Controlled Voltage Source1, the input of which is defined as $u + u_{max}$. For example, if $u$ is in the range of $[-5,5]$, the reverse threshold voltage is defined as 5 and $u > 0$ (i.e. $u + 5 = \text{REV voltage} > 5 = \text{threshold voltage}$), then the motor is in the forward mode.

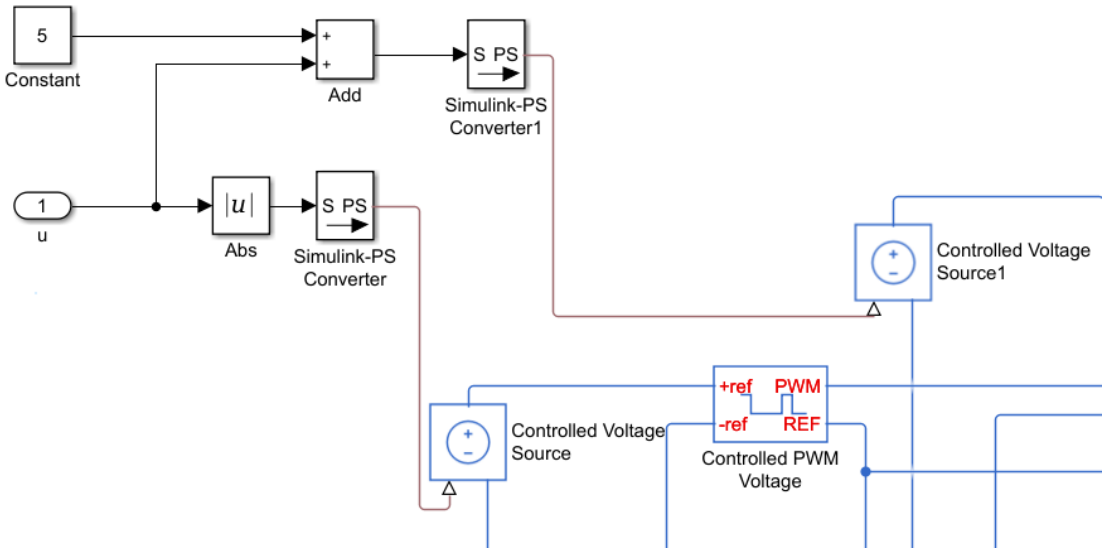


*Fig. 2.11. Block diagram of the Simscape model*

# Chapter 3 Parameter Identification

As shown in the mathematical model (2.17), all the parameters involved need to be determined. Some of them can be measured directly, while the others are not. The following table presents all the unknown parameters in the model:

| | |
|---|---|
| Measured | $m_1, l_0, l_1$ |
| Caculated | $J_1', K_b, K_t, R, \eta$ |
| Estimated | $J_0, C_0, C_1, K_{f0}, K_{f1}, K_u$ |

*Table 3.1 Table of unknown parameters to be identified*

The values of measured parameters can be found in Table 3.3.

## 3.1 Calculated Parameters

$J_1'$ is the moment of inertia of the pendulum about its mass-centre, which can be calculated using the formula for a uniform rod with negligible thickness:

$$J_1' = \frac{1}{12} m_1 l^2 = \frac{1}{12} \times 0.035 \times 0.16^2 = 7.467 \times 10^{-5}\ kgm^2 \qquad (3.1)$$

where, $l$ is the total length of the pendulum.

The following table shows the motor parameters provided by the manufacturer:

| | |
|---|---|
| Gear ratio | 20 |
| Rated voltage | 12 V |
| Rated current | 360 mA |
| Rated speed with gearbox | 549 RPM |
| Rated speed without gearbox | 11000 RPM |
| Rated torque with gearbox | 0.66 kgf.cm |
| Power | 4.32 W |
| Resolution of encoder | 260 count/revolution |

*Table 3.2 Table of motor parameters*

In SI units, the motor torque and back EMF constants are equal, i.e. $K_b = K_t$ . Here gives a way to calculate these constants from a motor nameplate:

$$K_b = K_t = \frac{E_N}{n_N} = \frac{U_N - I_N R}{n_N} \tag{3.2}$$

where, $U_N$ is the rated voltage, $I_N$ is the rated current, $n_N$ is the rated speed without gearbox, $E_N$ is the rated back EMF, $R$ is the armature resistance.

As shown in the above equation (3.2), if $E_N$ or $R$ is known, then $K_b$ and $K_t$ can be calculated. There are two ways to do this:

1. Estimate $E_N$ from experience:

$$E_N = (0.93\sim0.97)U_N\,. \tag{3.3}$$

   For a low-power DC motor, 0.97 is chosen.

2. Measure $R$ from experiments.

The first method is chosen and therefore:

$$K_b = \frac{E_N}{n_N} = \frac{0.97 \times 12}{11000 \times 2\pi \div 60} = 0.01\ \frac{V}{rad/s}\,, \tag{3.4}$$

$$K_t = K_b = 0.01\ \frac{N \cdot m}{A}\,. \tag{3.5}$$

Moreover,

$$R = \frac{U_N - E_N}{I_N} = \frac{(1 - 0.97) \times 12}{0.36} = 1\ \Omega\,. \tag{3.6}$$

The gearbox efficiency is calculated as follows:

$$\eta = \frac{T_N}{I_N \times K_t \times K_g} = \frac{0.66 \times 0.098}{0.36 \times 0.01 \times 20} = 0.889 \tag{3.7}$$

where, $T_N$ is the rated torque, $K_g$ is the gear ratio.

## 3.2 Estimation of $C_1, K_{f1}$ by Extended Kalman Filter (EKF)

Detailed information of the extended Kalman filter can be found in [4], here only gives the central ideas of EKF. The extended Kalman filter is a recursive state estimator of discrete-time nonlinear systems. Consider a discrete-time nonlinear system with nonadditive process and measurement noise:

$$\begin{aligned} \boldsymbol{x}[k+1] &= \boldsymbol{f}(\boldsymbol{x}[k], \boldsymbol{u}[k], \boldsymbol{w}[k]) \\ \boldsymbol{y}[k] &= \boldsymbol{h}(\boldsymbol{x}[k], \boldsymbol{u}[k], \boldsymbol{v}[k]) \end{aligned} \tag{3.8}$$

where:

$\boldsymbol{f}$ is the nonlinear state-transition function

$\boldsymbol{h}$ is the nonlinear measurement function

$\boldsymbol{x}$ is the state vector

$\boldsymbol{y}$ is the output vector

$\boldsymbol{u}$ is the input vector

$\boldsymbol{w}$ is the zero-mean Gaussian white process noise, i.e. $\boldsymbol{w}[k] \sim N(0, \boldsymbol{Q}[k])$

$\boldsymbol{v}$ is the zero-mean Gaussian white measurement noise, i.e. $\boldsymbol{v}[k] \sim N(0, \boldsymbol{R}[k])$

Also assume that $\boldsymbol{w}$ and $\boldsymbol{v}$ are uncorrelated, i.e. $E(\boldsymbol{w}[k]\boldsymbol{v}^T[l]) = 0$, for all $k$ and $l$. And the initial state is independent of $\boldsymbol{w}$ and $\boldsymbol{v}$, i.e. $E(\boldsymbol{w}[k]\boldsymbol{x}^T[0]) = 0$, $E(\boldsymbol{v}[k]\boldsymbol{x}^T[0]) = 0$, for all $k$.

The basic idea of how the EKF works can be summarised as follows:

1. Configure the EKF by setting the initial values of the state $\hat{\boldsymbol{x}}[0|0]$, its error covariance matrix $\boldsymbol{P}[0|0]$, process noise covariance $\boldsymbol{Q}$ and measurement noise covariance $\boldsymbol{R}$.
2. Predict the state and error covariance matrix at $k+1$ using information at $k$

$$\begin{aligned} \hat{\boldsymbol{x}}[k+1|k] &= \boldsymbol{f}(\hat{\boldsymbol{x}}[k|k], \boldsymbol{u}[k], \boldsymbol{0}) \\ \boldsymbol{P}[k+1|k] &= \boldsymbol{A}[k]\boldsymbol{P}[k|k]\boldsymbol{A}[k]^{\boldsymbol{T}} + \boldsymbol{G}[k]\boldsymbol{Q}[k]\boldsymbol{G}[k]^T \end{aligned} \tag{3.9}$$

   where:

   $\hat{\boldsymbol{x}}[k+1|k]$ is the state estimate at $k+1$ using measurements up to $k$

   $\boldsymbol{P}[k+1|k]$ is the state estimation error covariance matrix

   [ $\boldsymbol{A}$ $\boldsymbol{G}$ ] forms the Jacobian matrix of the state transition function, i.e.

$$\boldsymbol{A}[k] = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}[k|k]}, \qquad \boldsymbol{G}[k] = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{w}}\right|_{\hat{\boldsymbol{x}}[k|k]}$$

3. Correct the state and error covariance matrix at $k+1$ using measured output $\boldsymbol{y}[k+1]$

$$\begin{aligned} \boldsymbol{K}[k+1] &= \boldsymbol{P}[k+1|k]\boldsymbol{C}[k+1]^T(\boldsymbol{C}[k+1]\boldsymbol{P}[k+1|k]\boldsymbol{C}[k+1]^T + \\ &\quad \boldsymbol{S}[k+1]\boldsymbol{R}[k+1]\boldsymbol{S}[k+1]^T)^{-1} \\ \hat{\boldsymbol{x}}[k+1|k+1] &= \hat{\boldsymbol{x}}[k+1|k] + \boldsymbol{K}[k+1](\boldsymbol{y}[k+1] - \\ &\quad \boldsymbol{h}(\hat{\boldsymbol{x}}[k+1|k], \boldsymbol{u}[k+1], \boldsymbol{0}) \\ \boldsymbol{P}[k+1|k+1] &= \boldsymbol{P}[k+1|k] - \boldsymbol{K}[k+1]\boldsymbol{C}[k+1]\boldsymbol{P}[k+1|k] \end{aligned} \tag{3.10}$$

   where:

$\boldsymbol{K}$ is the Kalman gain

[ $\boldsymbol{C}$ $\boldsymbol{S}$ ] forms the Jacobian matrix of the measurement function, i.e.

$$\boldsymbol{C}[k+1] = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}[k+1|k]}, \qquad \boldsymbol{S}[k+1] = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{v}}\right|_{\hat{\boldsymbol{x}}[k+1|k]}$$

### 3.2.1 EKF for Parameter Estimation

The EKF can be used for joint estimation of state and system parameters. In joint estimation, the state and parameter vectors are combined. Consider a discrete-time nonlinear system described by equations (3.8) with some unknown parameters $\boldsymbol{\theta}$ :

$$\begin{aligned} \boldsymbol{x}[k+1] &= \boldsymbol{f}(\boldsymbol{x}[k], \boldsymbol{u}[k], \boldsymbol{\theta}, \boldsymbol{w}[k]) \\ \boldsymbol{y}[k] &= \boldsymbol{h}(\boldsymbol{x}[k], \boldsymbol{u}[k], \boldsymbol{\theta}, \boldsymbol{v}[k]) \end{aligned} \tag{3.11}$$

It is straightforward to implement the EKF by treating $\boldsymbol{\theta}$ as a random constant vector and combining the state and parameter vectors into a new state vector, such that:

$$\begin{aligned} \boldsymbol{x}[k+1] &= \boldsymbol{f}(\boldsymbol{x}[k], \boldsymbol{u}[k], \boldsymbol{\theta}[k], \boldsymbol{w}[k]) \\ \boldsymbol{\theta}[k+1] &= \boldsymbol{\theta}[k] + \boldsymbol{r}[k] \\ \boldsymbol{y}[k] &= \boldsymbol{h}(\boldsymbol{x}[k], \boldsymbol{u}[k], \boldsymbol{\theta}[k], \boldsymbol{v}[k]) \end{aligned} \tag{3.12}$$

where,

$\boldsymbol{r}$ is zero-mean Gaussian white noise and is uncorrelated with $\boldsymbol{v}$

To simplify the notation, (3.12) is rewritten as:

$$\begin{aligned} \boldsymbol{X}[k+1] &= \boldsymbol{F}(\boldsymbol{X}[k], \boldsymbol{u}[k], \boldsymbol{W}[k]) \\ \boldsymbol{y}[k] &= \boldsymbol{H}(\boldsymbol{X}[k], \boldsymbol{u}[k], \boldsymbol{v}[k]) \end{aligned} \tag{3.13}$$

where,

$\boldsymbol{X}[k] = \begin{bmatrix} \boldsymbol{x}[k] \\ \boldsymbol{\theta}[k] \end{bmatrix}$ is the new augmented state vector

$\boldsymbol{W}[k] = \begin{bmatrix} \boldsymbol{w}[k] \\ \boldsymbol{r}[k] \end{bmatrix}$ is the combined process noise including the parameter noise

$\boldsymbol{F}(\boldsymbol{X}[k], \boldsymbol{u}[k], \boldsymbol{W}[k]) = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}[k], \boldsymbol{u}[k], \boldsymbol{\theta}[k], \boldsymbol{w}[k]) \\ \boldsymbol{\theta}[k] + \boldsymbol{r}[k] \end{bmatrix}$ is the combined state transition function including the dynamics of the state and parameter vector

$\boldsymbol{H}(\boldsymbol{X}[k], \boldsymbol{u}[k], \boldsymbol{v}[k]) = \boldsymbol{h}(\boldsymbol{x}[k], \boldsymbol{u}[k], \boldsymbol{\theta}[k], \boldsymbol{v}[k])$ is the new measurement function

With the combined model of the dynamic system, the EKF can be used to estimate the system state and parameters simultaneously.

Here give general procedures for performing parameter estimation using EKF with additive process and measurement noise:

1. Discretize the mathematical model.
2. Derive the state-transition function with additive process noise.
$$\boldsymbol{X}[k+1] = \boldsymbol{F}(\boldsymbol{X}[k], \boldsymbol{u}[k]) + \boldsymbol{W}[k]$$
3. Derive the measurement function with additive measurement noise.
$$\boldsymbol{y}[k] = \boldsymbol{H}(\boldsymbol{X}[k], \boldsymbol{u}[k]) + \boldsymbol{v}[k]$$
4. Construct the filter by setting initial state estimate, initial state estimation error covariance, process noise covariance and measurement noise covariance.
5. Perform estimation using predict and correct.

### 3.2.2 Design of EKF for Estimating $C_1, K_{f1}$

These two parameters $C_1, K_{f1}$ are only related to the pendulum; hence a simple free vibration of the pendulum was recorded without the need of exciting the dynamics of the arm. In this case, the mathematical model is simplified as follows:

$$J_1\ddot{\beta} + C_1\dot{\beta} + m_1 g l_1 sin\beta + K_{f1} f(\dot{\beta}) = 0 \tag{3.14}$$

where, $J_1$ is the moment of inertia of the pendulum with respect to the shaft of the potentiometer. According to the parallel axis theorem, $J_1 = J_1' + m_1 l_1^2$ and $J_1', m_1, l_1$ can be found in Table 3.3.

The above equation (3.14) can be rewritten in state-space representation:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \ddot{x}_2 &= -\frac{C_1 x_2 + m_1 g l_1 sin x_1 + K_{f1} f(x_2)}{J_1' + m_1 l_1^2} \end{aligned} \tag{3.15}$$

where, $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \beta \\ \dot{\beta} \end{bmatrix}$.

The above mathematical model (3.15) is then discretized using Euler method:

$$\dot{x} \approx \frac{x[k+1] - x[k]}{T_S} \tag{3.16}$$

where, $T_S$ is the sample time.

Therefore, the discretized mathematical model is:

$$\begin{aligned} x_1[k+1] &= x_1[k] + T_S x_2[k] \\ x_2[k+1] &= x_2[k] - T_S \frac{C_1 x_2[k] + m_1 g l_1 sin x_1[k] + K_{f1} f(x_2[k])}{J_1' + m_1 l_1^2} \end{aligned} \tag{3.17}$$

Note that higher-order Runge-Kutta methods can be used if the accuracy needs to be improved. However, in this project, the sample time is 5ms which makes it enough to use simple first-order numerical approximation.

Since $C_1, K_{f1}$ are the unknown parameters, the augmented state vector is formed as:

$$\boldsymbol{X}[k] = \begin{bmatrix} \beta(k) \\ \dot{\beta}(k) \\ C_1(k) \\ K_{f1}(k) \end{bmatrix} = \begin{bmatrix} x_1[k] \\ x_2[k] \\ x_3[k] \\ x_4[k] \end{bmatrix} \tag{3.18}$$

Therefore, the state transition function can be obtained from (3.17):

$$\begin{aligned} \boldsymbol{X}[k+1] &= \begin{bmatrix} x_1[k+1] \\ x_2[k+1] \\ x_3[k+1] \\ x_4[k+1] \end{bmatrix} = \boldsymbol{F}(\boldsymbol{X}[k], \boldsymbol{u}[k]) + \boldsymbol{W}[k] \\ &= \begin{bmatrix} x_1[k] + T_S x_2[k] \\ x_2[k] - T_S \dfrac{x_3[k] x_2[k] + m_1 g l_1 sin x_1[k] + x_4[k] f(x_2[k])}{J_1' + m_1 l_1^2} \\ x_3[k] \\ x_4[k] \end{bmatrix} \\ &+ \boldsymbol{W}[k] \end{aligned} \tag{3.19}$$

The measurements are the angle and angular velocity of the pendulum:

$$\boldsymbol{y}[k] = \begin{bmatrix} y_1[k] \\ y_2[k] \end{bmatrix} = \boldsymbol{H}(\boldsymbol{X}[k], \boldsymbol{u}[k]) + \boldsymbol{v}[k] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1[k] \\ x_2[k] \\ x_3[k] \\ x_4[k] \end{bmatrix} + \boldsymbol{v}[k] \tag{3.20}$$

### 3.2.3 Off-Line Simulation

In most applications, parameter estimation method is used in real-time and an off-line simulation is always conducted before implementing the algorithm in the hardware. This is used to ensure that the algorithm works properly for this particular mathematical model. To achieve this, the Simulink model created in Section 2.5.1 is used. The unknown parameters in the model are defined as:

$$\begin{bmatrix} C_1 \\ K_{f1} \end{bmatrix} = \begin{bmatrix} 1 \times 10^{-4} \dfrac{kgm^2}{s} \\ 1 \times 10^{-4}\ kgm^2 \end{bmatrix} \tag{3.21}$$

The general procedures of performing off-line simulation is described by the following flow chart:

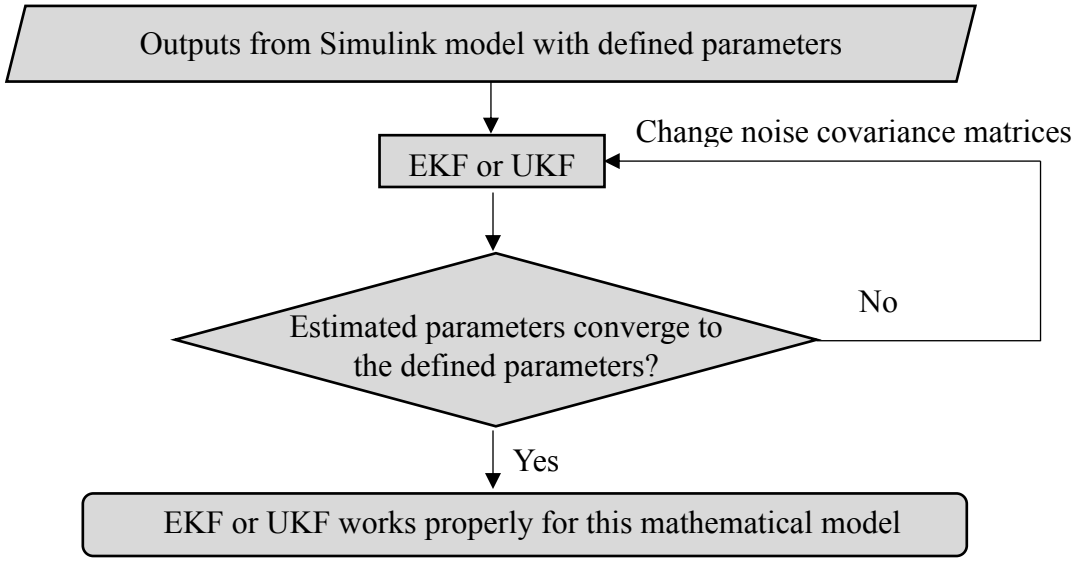


*Fig. 3.1. Flow chart of performing off-line simulation*

Note that the convergence of the filter has two meanings:

1. The estimated parameters vary around the mean values with accepted errors. Note that the estimated parameters may not be able to converge to an absolutely stable value if the dynamics of the system does not vanish. This should be considered when doing the experiments.
2. Estimated parameters converge to almost the same values regardless of the initial conditions of the system.

The noise covariance matrices are obtained by trial and error and have the following values:

$$\boldsymbol{Q} = diag(1 \times 10^{-10}, 1 \times 10^{-10}, 1 \times 10^{-10}, 1 \times 10^{-10})$$

$$\boldsymbol{R} = diag(1 \times 10^{-3}, 1 \times 10^{-3})$$

The results of the simulations when $\begin{bmatrix} \beta(0) \\ \dot{\beta}(k) \end{bmatrix} = \begin{bmatrix} 1.5\, rad \\ 0\, rad/s \end{bmatrix}$ and $\begin{bmatrix} \beta(0) \\ \dot{\beta}(k) \end{bmatrix} = \begin{bmatrix} 0.6\, rad \\ 0\, rad/s \end{bmatrix}$ are shown in the following Fig. 3.2 and Fig. 3.3 respectively

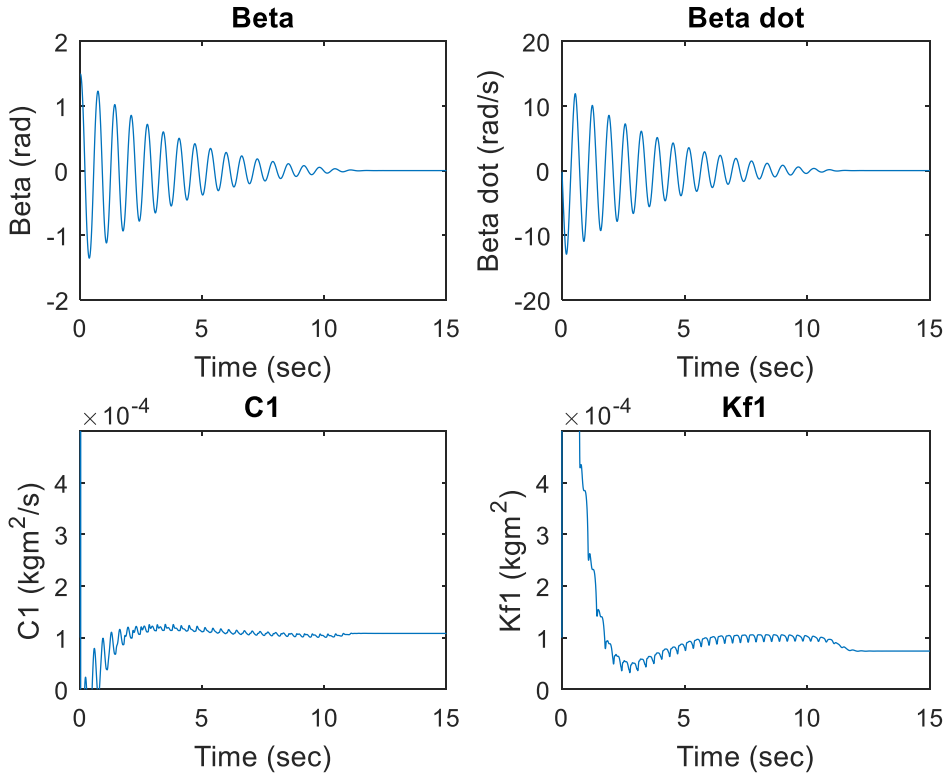


*Fig. 3.2. Results of off-line simulation when $\beta(0) = 1.5\, rad$*

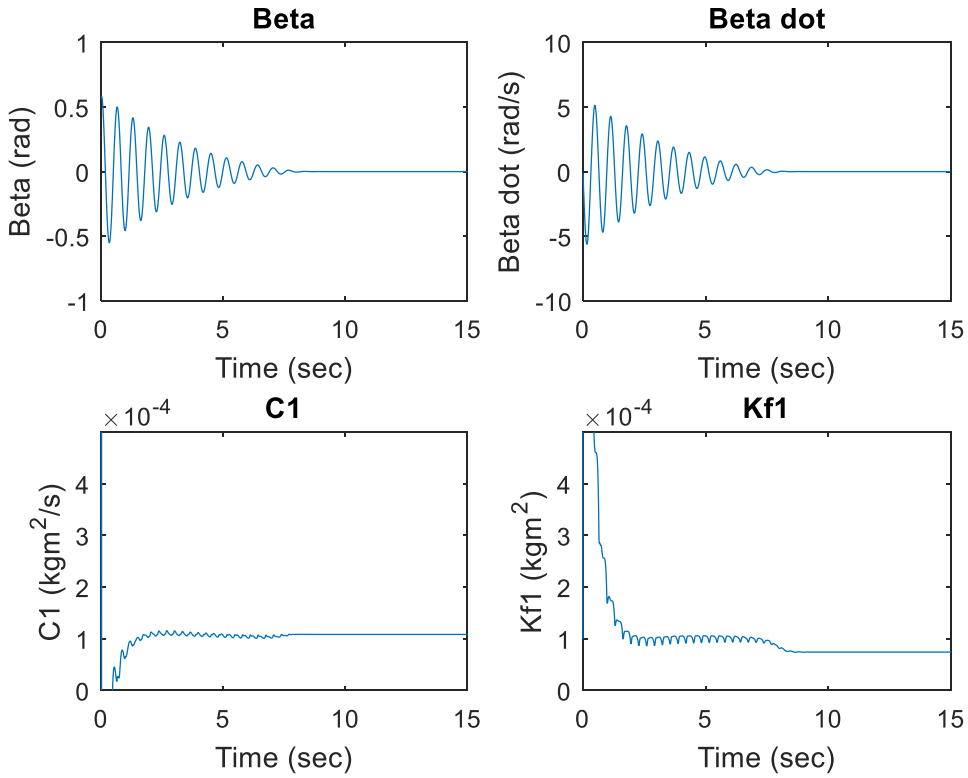


*Fig. 3.3. Results of off-line simulation when $\beta(0) = 0.6\,rad$*

As shown in the above figures, both $C_1$ and $K_{f1}$ almost converge to the defined values (3.21) regardless of the initial conditions of the system. Therefore, the Kalman filter works properly and some experiments can be conducted. The details of the MATLAB code which implements the EKF for off-line simulation can be found in the Appendix A.

### 3.2.4 Off-Line Experiments

Since all the data was recorded before performing the estimation in this project, the actual parameter estimation was still conducted off line by loading the data into the workspace of MATLAB.

The data recorded for $\dot{\beta}$ is calculated by using the average velocity described by (3.16). Since $\beta$ is obtained by a potentiometer and is noisy, this signal will be filtered through a low-pass filter before being used to calculate $\dot{\beta}$. Not using this filter shows that the Kalman filter will not converge.

The detailed MATLAB code of designing a digital low-pass filter can be found in the Appendix B. The filter used in this project is a first-order recursive filter:

$$y[k] = ay[k-1] + (1-a)x[k]\,. \tag{3.22}$$

where, $x[k]$ is the input signal to be filtered, $y[k]$ is the output filtered signal and $a\epsilon(0,1)$.

The Z-transform of this filter is:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1-a}{1-az^{-1}} \tag{3.23}$$

The magnitude response is:

$$\begin{aligned} \left|H\left(e^{j\omega}\right)\right| &= \left|\frac{1-a}{1-ae^{-j\omega}}\right| = \left|\frac{1-a}{1-acos(\omega)+asin(\omega)\,j}\right| \\ &= \frac{1-a}{\sqrt{1-2\,acos(\omega)+a^2}} \end{aligned} \tag{3.24}$$

The normalized cut-off frequency can be obtained by:

$$\left|H\left(e^{j\omega_c}\right)\right| = \frac{1}{\sqrt{2}} \tag{3.25}$$

Hence the relationship between $a$ and $\omega_c$ can be found by combining (3.24) and (3.25):

$$a = 2 - cos(\omega_c) - \sqrt{cos^2(\omega_c) - 4\,cos(\omega_c) + 3} \tag{3.26}$$

The frequency response of the noisy signal $\beta$ is plotted by computing the discrete Fourier transform (DFT) of it using a fast Fourier transform (FFT) algorithm:

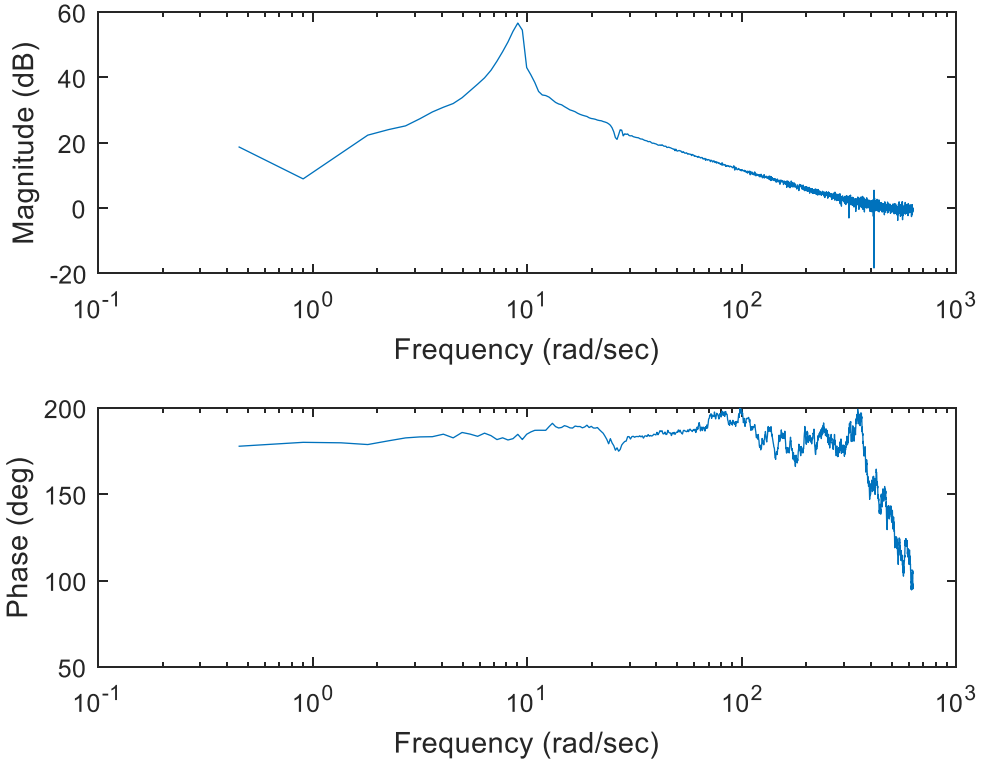


*Fig. 3.4. Frequency response of the input signal β*

According to the above spectrum Fig. 3.4., the cut-off frequency is chosen to be $45\ rad/s$, this corresponds to a normalized frequency with the sample time to be $0.005\ s$:

$$\omega_c = 45 \times T_S = 45 \times 0.005 = 0.225$$

Hence, $a$ can be found by substituting $\omega_c$ into (3.26) and solving for $a$:

$$a \approx 0.8$$

The frequency response of the low-pass filter when $a = 0.8$ is:

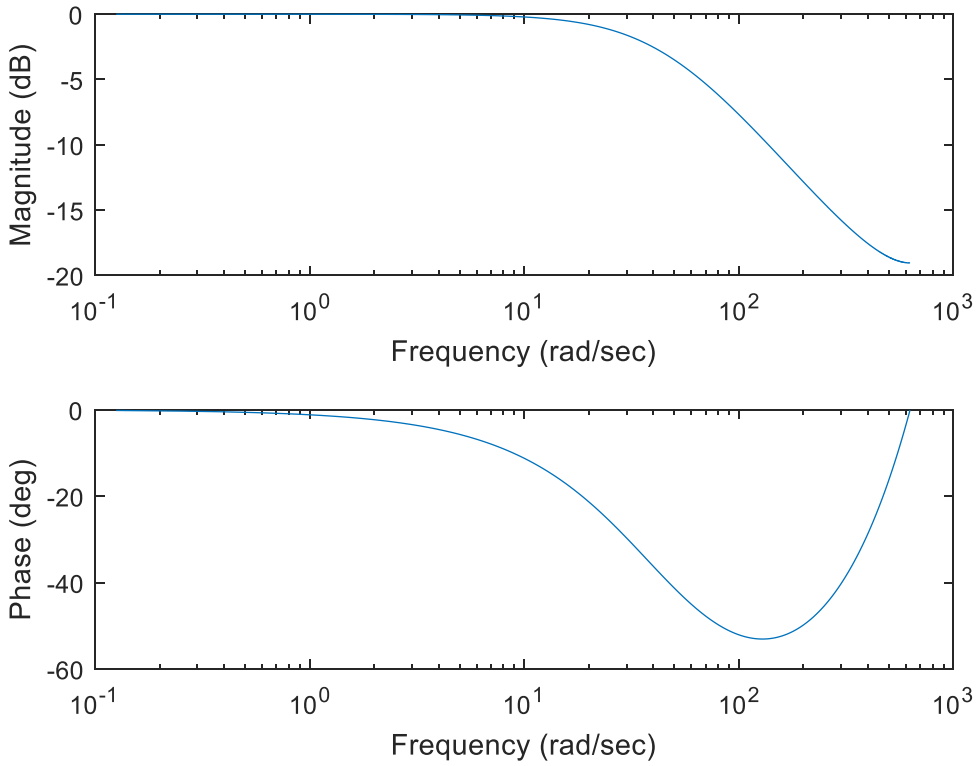


*Fig. 3.5. Frequency response of the designed digital low-pass filter*

Since the phase contribution of this low-pass filter is negative, it will cause a phase delay in the output signal.

To avoid the phase delay, the zero-phase shift filtering is used, which is achieved by filtering the input data in both the forward and reserve direction. In other words, after the data is filtered in the forward direction, the filtered data will be reversed and put back through the filter again. Therefore, this squares the magnitude response of the filter but zeros the phase response. The result is shown in the following Fig. 3.6.:

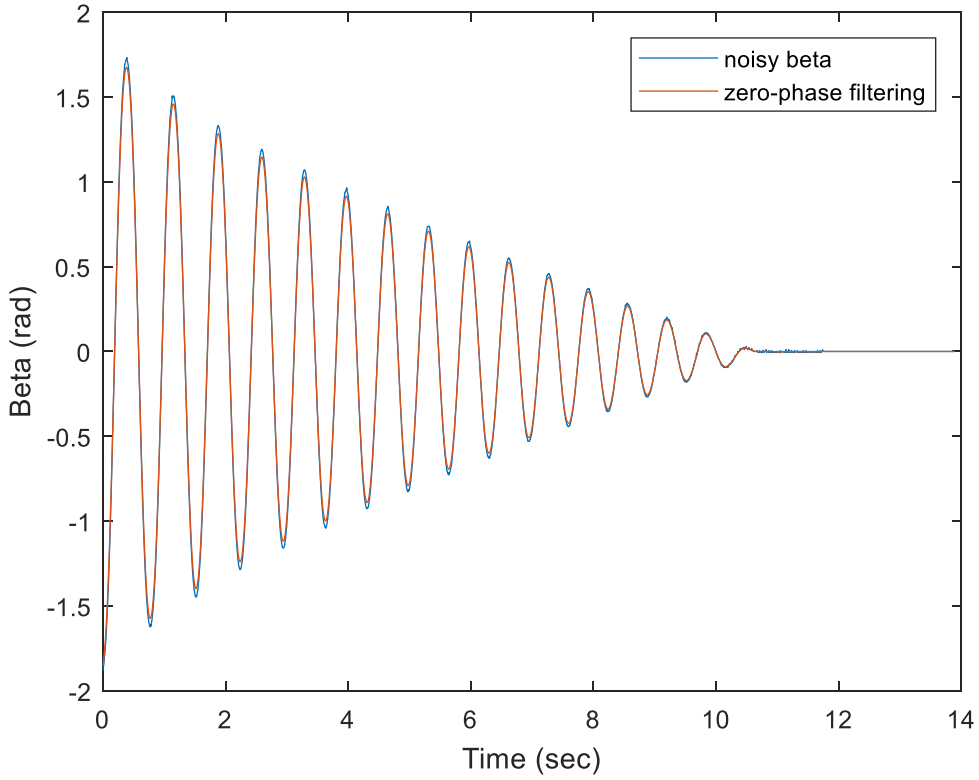


*Fig. 3.6. Noisy input and filtered input*

Note that it is impossible to perform a zero-phase filtering in real-time because a zero-phase filter requires filter coefficients that are symmetric around zero, which means that the filter is non-causal, or that current output depends on future inputs. This is physically-impossible. Therefore, the conventional filtering will still be used in the hardware algorithm design.

The overall procedures of estimating parameters using a Kalman filter are described by the following flow chart:

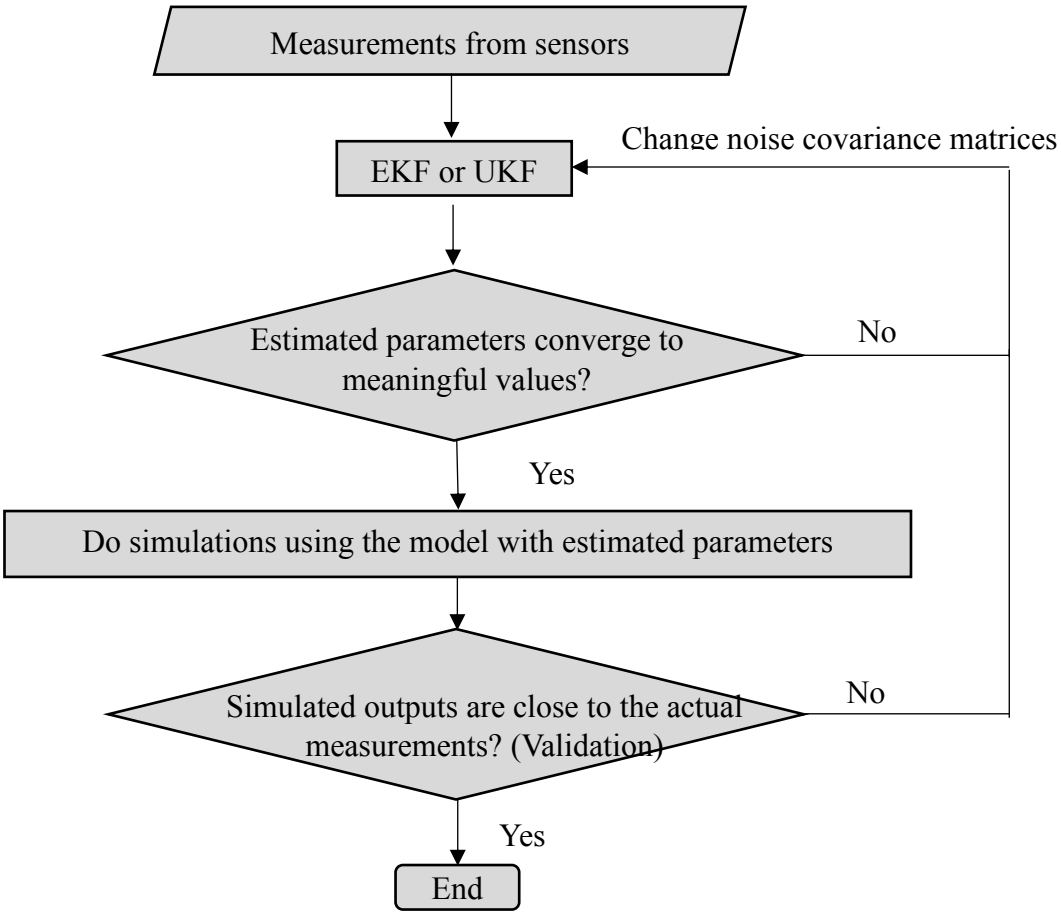


*Fig. 3.7. Flow chart of performing off-line experiments*

Tuning a Kalman filter is an iterative process, which is mainly achieved by trial and error. Note that the estimated parameters may not be able to converge to an absolutely stable value if the dynamics of the system does not vanish. Therefore, the results of estimation should be the mean values of the last, for example, 100 samples. Therefore, one should keep changing the process and measurement noise covariance matrix until the estimated parameters vary around these mean values with accepted errors. If so, the unknown parameters in the model will be set to these mean values and some simulations will be done to check if the simulated outputs are close to the actual measurements. This step is called validation in modelling.

The details of the MATLAB code which implements the EKF for off-line experiments can be found in the Appendix C.

The noise covariance matrices have the following values which are tuned by trial and error:

$$\boldsymbol{Q} = diag(1 \times 10^{-14}, 1 \times 10^{-14}, 1 \times 10^{-14}, 1 \times 10^{-14})$$

$$\boldsymbol{R} = diag(\,2\,, 2\,)$$

The results of the experiments when $\begin{bmatrix}\beta(0)\\ \dot{\beta}(k)\end{bmatrix} = \begin{bmatrix}-1.87\ rad\\ 0\ rad/s\end{bmatrix}$ and $\begin{bmatrix}\beta(0)\\ \dot{\beta}(k)\end{bmatrix} = \begin{bmatrix}1.09\ rad\\ 0\ rad/s\end{bmatrix}$ are shown in the following Fig. 3.8. and Fig. 3.9. respectively

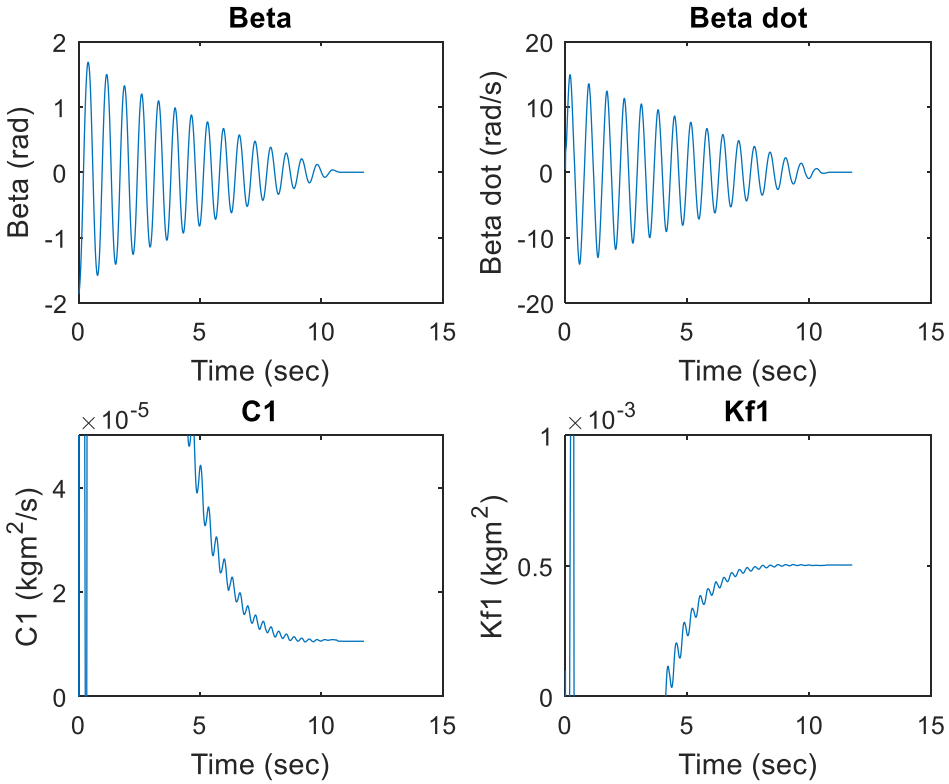


*Fig. 3.8. Results of off-line experiments when $\beta(0) = -1.87\ rad$*

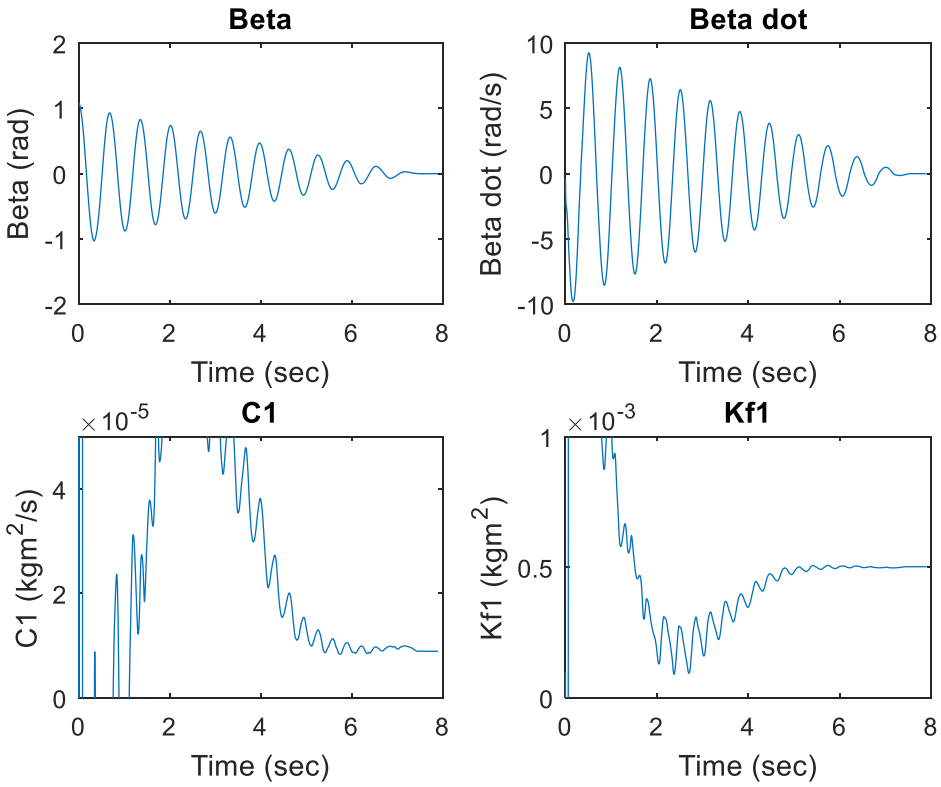


*Fig. 3.8. Results of off-line experiments when $\beta(0) = -1.09\ rad$*

As shown in the above Fig. 3.7. and 3.8., both $C_1$ and $K_{f1}$ almost converge to the same values regardless of the initial conditions of the system:

$$\begin{bmatrix} C_1 \\ K_{f1} \end{bmatrix} = \begin{bmatrix} 1.059 \times 10^{-5}\ kgm^2/s \\ 5.036 \times 10^{-4}\ kgm^2 \end{bmatrix}$$

The validation result is shown below using another two experiments:

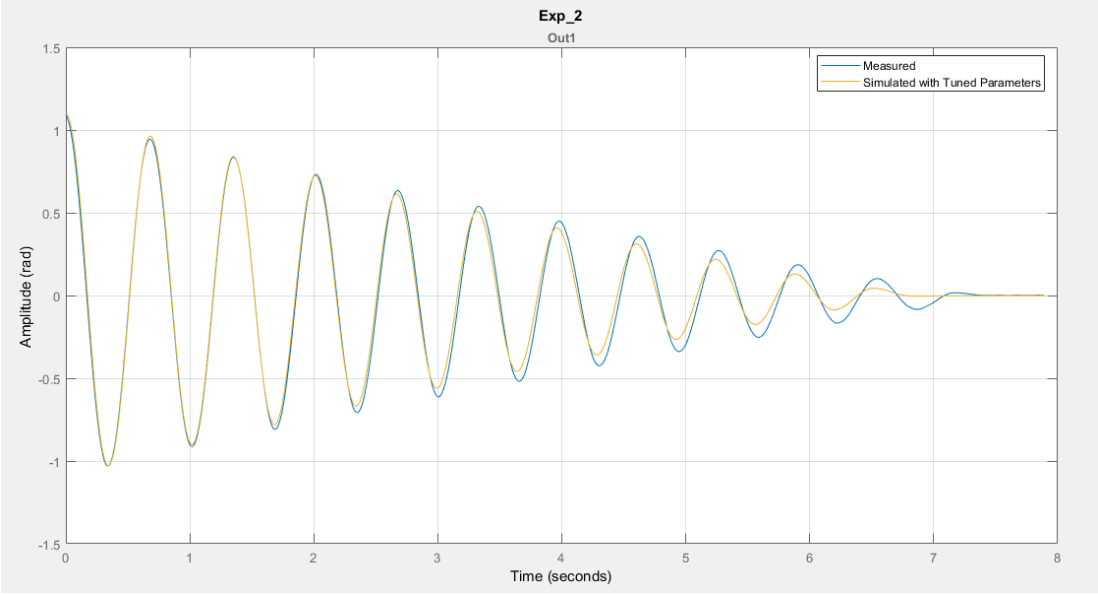


*Fig. 3.9. Validation result when $\beta(0) = 1.1\ rad$*

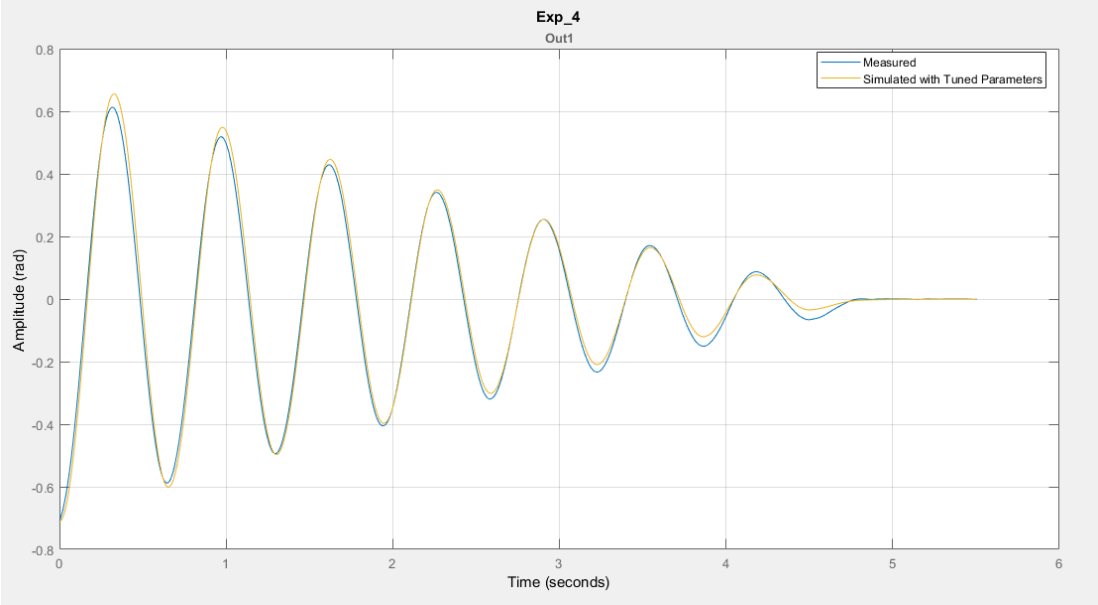


*Fig. 3.10. Validation result when $\beta(0) = -0.7\ rad$*

As shown in the above Fig. 3.9. and 3.10., there are still visible errors between the simulated results and measurements. This may be explained by the effect of low-pass filter which decreases the magnitude of measurements. Therefore, the Kalman filter uses these filtered measurements with smaller magnitude to give a model fitting in which the damping coefficients seem to be too big.

### 3.2.5 Discussions

Although tuning a Kalman filter is mainly done by trial and error, here are some general ideas concluded.

The process noise covariance should be much smaller than the measurement noise covariance, otherwise the filter will not converge. This is reasonable because smaller process noise means that one has more confidence in the mathematical model. If the model built is of much process noise, there is no way the filter will converge because it relies on the model to predict.

The convergence criterion in parameter estimation should not be based on the residuals, namely the errors between estimated measurements and actual measurements. This is simply because one can always obtain residuals with zero mean

and small magnitude by decreasing the measurement noise covariance. In most cases, smaller residuals do not mean that the estimated parameters are close to the true values, hence the validation still needs to be done.

In this project, no matter how the filter is tuned, the estimated parameters will become stable eventually because the dynamics of the system vanish and there is no update in the filter due to the property of a free damped response. However, this should not be used as a criterion for convergence because, in most cases, the estimated parameter will not converge to an absolutely stable value if the dynamics of the system still exists. Moreover, if the dynamics of the system is too short, then the filter will not have enough time to converge to the right values. Therefore, one should modify the experiments or try to record more data if the filter does not have any sign of convergence.

If more measurements are available or less parameters to be estimated, it is easier for the filter to converge. Noise should be eliminated in the measurements as much as possible, otherwise the filter will not converge.

## 3.3 Estimation of $J_0, C_0, K_{f0}, K_u$ by Nonlinear Least Squares

The least squares method is an optimization approach which gives a solution for an overdetermined system, namely it selects the parameters that minimizes the sum of the squared residuals:

$$S = \sum_{i=1}^{n} [y_{data} - y_{model}]^2 \tag{3.27}$$

where,

$y_{data}$ is the observed measurement at the i-th data point.

$y_{model}$ is the predicted measurement obtained from the mathematical model, which contains the unknown parameters to be estimated.

The details of the specific least-squares algorithms used for model fitting like Trust-Region-Reflective and Levenberg-Marquardt are beyond the scope of this project.

Therefore, in this project, Simulink Design Optimization™ is used to estimate parameters from measured data without dealing with the details of how to implement these algorithms.

### 3.3.1 Experiments

To estimate these unknown parameters $J_0, C_0, K_{f0}, K_u$, pulse responses of the angle of the arm and the pendulum are recorded to excite all the dynamics of the system. The responses are then filtered using the built-in data processing function in Simulink Design Optimization™. The algorithm used here is Trust-Region-Reflective and the stop criterion is when the current norm of the cost function is less than $1 \times 10^{-3}$. The values of the estimated parameters using four different experiments are as follows:

$$\begin{bmatrix} J_0 \\ C_0 \\ K_{f0} \\ K_u \end{bmatrix} = \begin{bmatrix} 3.822 \times 10^{-3}\ kgm^2 \\ 1.155 \times 10^{-3} kgm^2/s \\ 0.0665\ kgm^2 \\ 0.0225\ V \end{bmatrix}$$

The above results are validated using another two experiments:

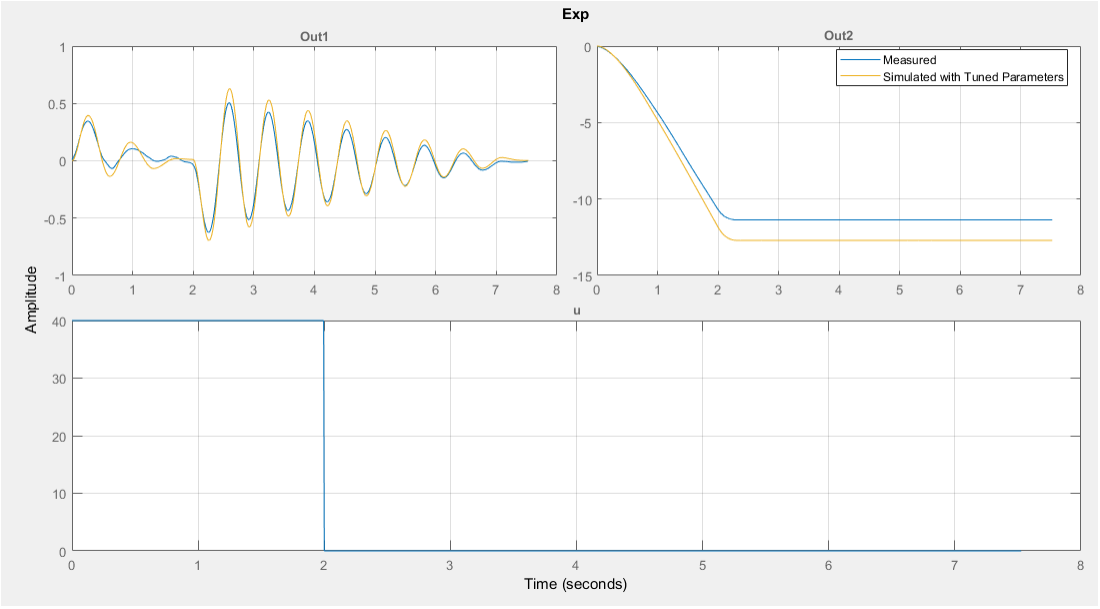


*Fig. 3.11. Validation result when* $u(0) = 40$

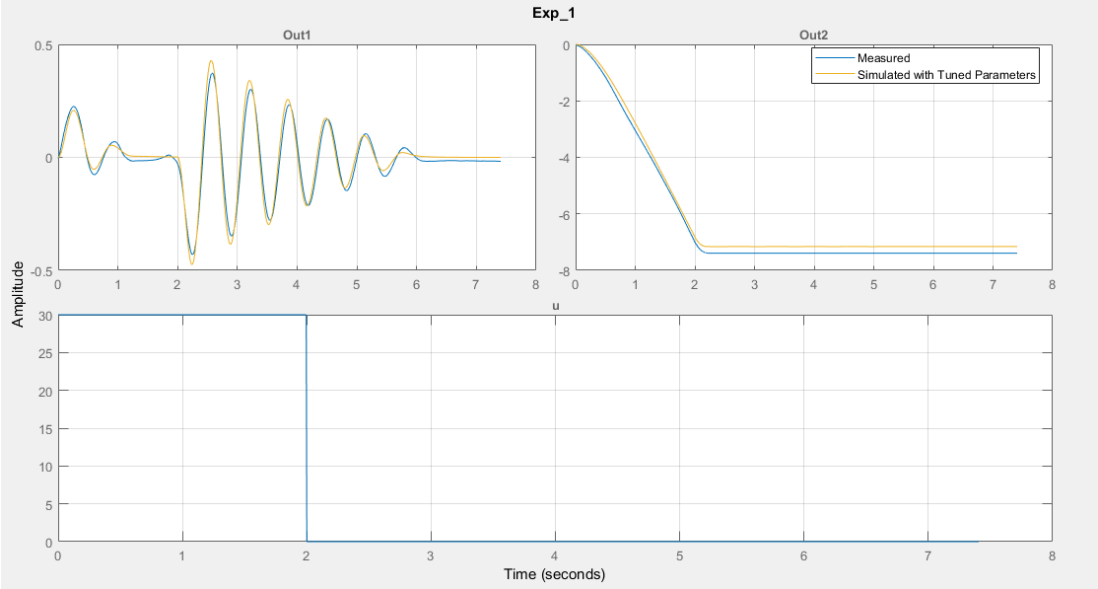


*Fig. 3.12. Validation result when $u(0) = 30$*

### 3.3.2 Discussions

Although the validation results above seem to be good, the responses with the estimated parameters are in great error if the input becomes larger. The major reason for that is the gain of the amplifier $K_u$ is not a constant. Moreover, the performance of using least squares method for parameter estimation highly depends on the initial guess of the system parameters. Even slightly changing the initial values can result in the algorithm converging to different values.

## 3.4 Table of Parameters

| | | |
|---|---|---|
| $m_1 = 0.035\ kg$ | $l_0 = 0.145\ m$ | $l_1 = 0.07\ m$ |
| $J_1' = 7.467 \times 10^{-5}\ kgm^2$ | $K_t = 0.01\ \frac{N \cdot m}{A}$ | $K_b = 0.01\ \frac{V}{rad/s}$ |
| $R = 1\ \Omega$ | $\eta = 0.889$ | $J_0 = 3.822 \times 10^{-3}\ kgm^2$ |
| $C_0 = 1.155 \times 10^{-3} kgm^2/s$ | $K_u = 0.0225\ V$ | $K_{f0} = 0.0665\ kgm^2$ |
| $K_{f1} = 5.036 \times 10^{-4}\ kgm^2$ | $C_1 = 1.059 \times 10^{-5}\ kgm^2/s$ | |

*Table 3.3: Table of parameters*

## 3.5 Possible Reasons for Inaccurate Modelling

Although the dynamics of the amplifier can be neglected in this project, the gain of the amplifier in not a constant, but depends on the input value. In fact, the amplifier has higher gain if the input value, namely the duty cycle of the PWM signal, becomes larger.

The parameters of the DC motor are estimated by simply assuming the rated back EMF is 0.97 times the rated voltage. This formula is obtained from experience and is not accurate.

The static frictions at the revolute joints are not modelled, this will cause problems when designing stabilizing controller in simulation environment, which will be discussed in Section 5.2.

# Chapter 4 Control System Design

The main purpose of this project is to design a hybrid controller consisting of the following three parts:

- Swing-up control which gradually swings the pendulum to the inverted position.
- Stabilizing control which balances the pendulum in the unstable upright position.
- Switching control which catches the pendulum around the inverted position by switching accurately between swing-up and stabilizing control.

## 4.1 Swing-up Control

The swing-up control is a completely nonlinear problem with many difficulties and different control algorithms have been proposed for this problem during the past decades. A cascade PD controller with iterative impulsive control is presented in [5], which is used to minimize the swing-up time. Energy based control is implemented in [6], which is based on a dimension 2 model. Moreover, some modern control theories in [7]-[10] like sliding mode control, fuzzy control, adaptive control and intelligent control have been investigated.

However, all these strategies require the angle of the pendulum to be fed back in whole 360°. This is not the case in this project since the potentiometer used to sense the angle of the pendulum has a 20° blind zone. Therefore, the control strategies presented in most papers cannot be used in this project. For this reason, a simplified energy-based method is presented here.

Note that the 20° blind zone always exists and there is no way to let the controller recognize this zone because the data recorded in this zone is random. Therefore, the only thing can be done with this zone is to minimize its effect, not get rid of it. One simple way is to give no control input when the pendulum is within this blind zone, which means that all control algorithms should not be activated in this zone. Therefore,

the range of the angle of the pendulum, where the swing-up control is activated, is defined as the following swing-up zone.

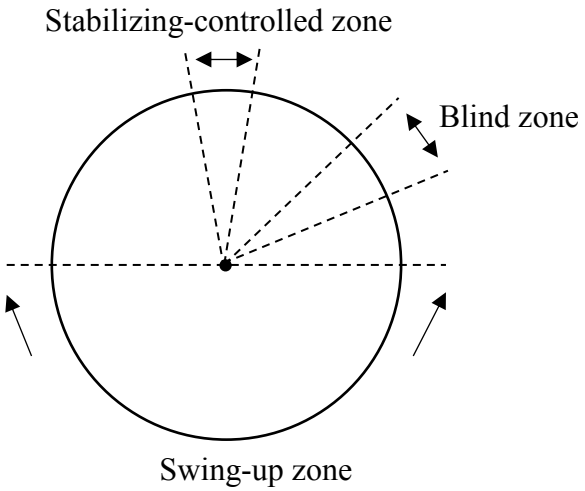


*Fig. 4.1 Different zones of the angle of the pendulum*

The basic idea of this method is to gradually add energy into the pendulum by movements of the arm at the right time and in the right direction. The moment at which the energy is inserted into the system is when the pendulum is within the swing-up zone and moving towards the downward position. At that moment, the arm should move in the opposite direction as the pendulum is heading to, otherwise the arm will stop. The following pseudocode illustrates this process:

*If (the pendulum is within the swing-up zone and moving towards the downward position)*
*{*
*Move the arm in the opposite direction as the pendulum is heading to;*
*}*
*Else*
*{*
*Stop the arm;*
*}*

This method works well if the torque of the motor is great enough. The control input of the motor to move the arm is a constant obtained by trial and error, but there is a

trade-off in choosing this value. A large control input adds more energy into the pendulum at each time and thus will drive the pendulum to the inverted position within less time but at larger angular velocity. In this case, the stabilizing controller may not be able to catch the pendulum since its moving too fast in the upright position. By contrast, a small value will lead to more swings but smaller angular velocity of the pendulum in the upright position.

This method is simple and easy to implement in a microcontroller, but it has no control of the velocity of the pendulum since this value is not fed back. Therefore, a slightly different initial condition can result in a different velocity when the pendulum is moving across the inverted position. Moreover, a sudden constant torque may be applied to the arm when the pendulum is within the blind zone since the data acquired within this zone may incorrectly indicate the pendulum is within the swing-up zone. Therefore, this swing-up strategy is able to swing up the pendulum, but the stabilizing controller may not be able to catch the pendulum every time.

## 4.2 Switching Control

The switching controller can switch properly between the swing-up and the stabilizing control when any disturbance is applied to the pendulum. Note that it is almost impossible to switch when the pendulum is exactly in the inverted position. Therefore, the extent to which the stabilizing control can still take effect is found by experiments. In case the pendulum is disturbed from the upright position, the swing-up controller should be activated again. As shown in Fig. 4.1., when the pendulum is within the stabilizing-controlled zone (±10° from the inverted position), the stabilizing controller will take over from the swing-up controller.

## 4.3 Stabilizing Control

The stabilization of RIP in the upright position is another challenging problem and many control algorithms have been proposed to guarantee the stability and robustness in the closed loop system. The PID controller presented in [11] gives the simplest solution to this problem, but tuning the controller is not an easy task. Another

simplified solution discussed in [12] is to linearize the model around the upright position and apply the linear quadratic regulator (LQR). Modern control theories like fuzzy logic regulator, sliding mode controller and adaptive control in [13]-[16] have been proposed in recent years, but they are too complex to be implemented in a microcontroller.

The stabilizing-control algorithm used in this project is a double-PD-loop controller. One loop feeds back the angle of the pendulum while the other feeds back the angle of the arm. Initially, this structure is developed by intuition as shown in the following sections, but it is actually a full state feedback controller and has a complete theoretical background supporting it, which will be discussed in Chapter 5.

### 4.3.1 Single-P-Loop Controller

In feedback control theory, if an output needs to be stabilized, one simple solution is to calculate the difference between the desired reference and the measured output and minimize the error over time by a negative feedback. Therefore, in order to balance the pendulum in the upright position, the following proportional control is tried.

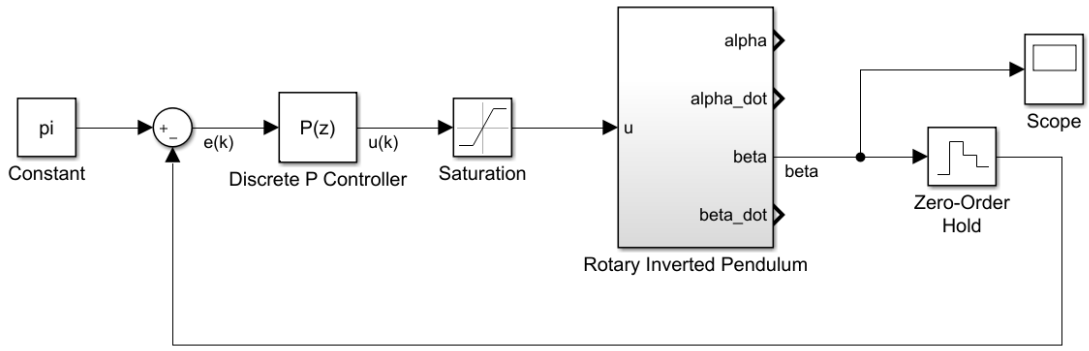


*Fig. 4.2. Block diagram of a single-P-loop controller*

$$P(z) = K_{P_{beta}} \Rightarrow u(k) = K_{P_{beta}} e(k) \tag{4.1}$$

As shown in Fig. 4.2., a single proportional controller which feeds back the angle of the pendulum is used. The reference is the angle of the pendulum where the pendulum is in the upright position (i.e. $\beta = \pi$ ). According to experience, this P controller should be able to balance the pendulum for some time like a few seconds. After

adjusting the value of the proportional gain by trial and error, it can be found that if the gain is larger, the pendulum can be stabilized for a longer time. This is reasonable because a larger control input always corresponds to a better dynamic behaviour. The actual sign of the proportional gain implemented in the microcontroller is determined by experiments. The following response obtained in Simulink illustrates the behaviour of this controller.

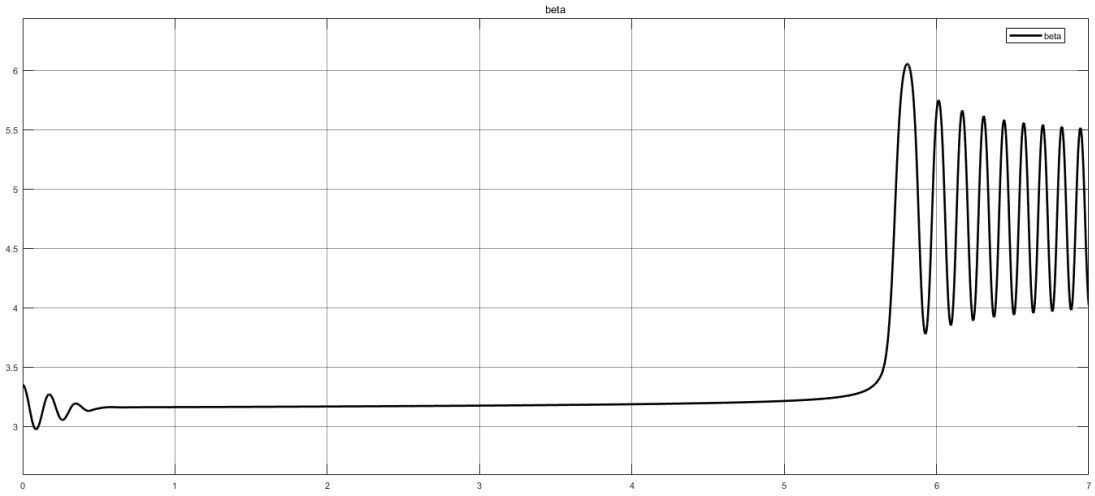


*Fig. 4.3. Response of the single-P-loop controller*

As shown in the above Fig. 4.3., the P controller can stabilize the pendulum for about 5s. This time can still be increased by increasing the gain of the controller. However, in any physical system, the control input has a limitation. In this project, the input represents the pulse width of the PWM signal and has a maximum value which corresponds to 100% duty cycle or maximum voltage applied to the motor or maximum torque provided by the motor. In other words, a control input without bound will saturate the system eventually. Moreover, a larger gain will increase the overshoot of the response as shown in the above Fig. 4.3. Therefore, the criterion of tuning the value of the proportional gain $K_{P\_beta}$ is not the larger, the better. In contrast, this value should be chosen in such a way that it will result in a lightly damped system (i.e. with some amount of overshoot). In other words, $K_{P\_beta}$ should be continuously increased until the pendulum exists low-frequency vibration.

### 4.3.2 Single-PD-Loop Controller

To decrease the overshoot of the response, the P controller is modified into a PD (proportional and derivative) controller. This additional derivative term is used to reduce the effect of the proportional one, which means that it will add damping into the system and hence reduce the overshoot.

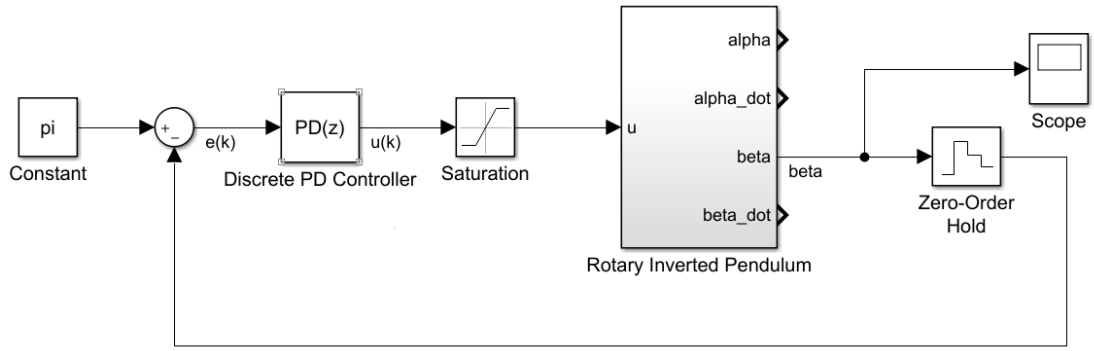


*Fig. 4.4. Block diagram of the single-PD-loop controller*

$$PD(z) = K_{P\_beta} + K_{D\_beta}\frac{1}{T_S}\frac{z-1}{z}$$
$$\Rightarrow u(k) = K_{P\_beta}e(k) + K_{D\_beta}\frac{e(k)-e(k-1)}{T_S} \quad (4.2)$$

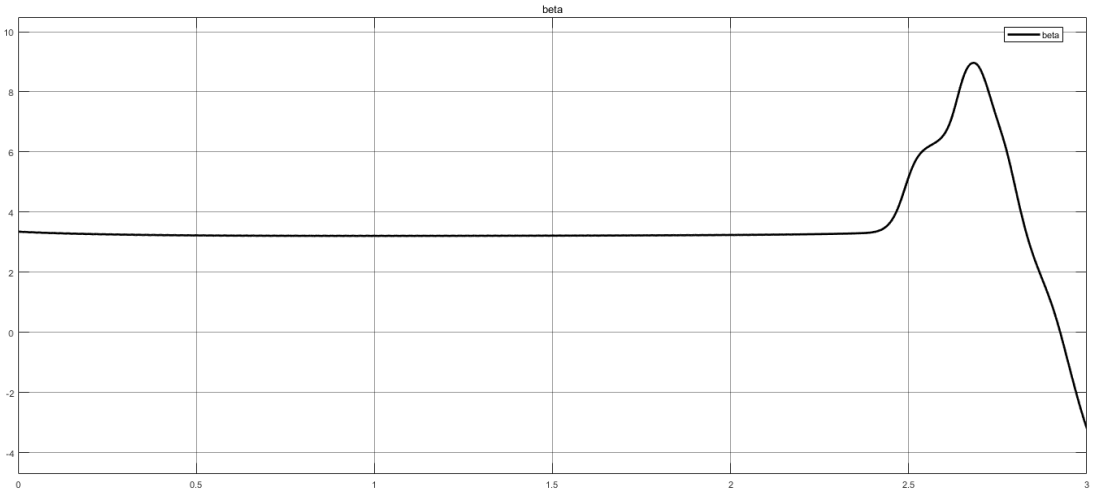


*Fig. 4.5. Response of the single-PD-loop controller*

Compared with Fig. 4.3., the response has no overshoot, but the pendulum loses balance more quickly at about 2.5s as shown in Fig. 4.5. This result is expected because the derivative term counteracts the effect of the proportional one and hence reduce the

overshoot and balancing effect. The sign of this derivative term should be the same as the proportional one, while the value of this D term (i.e. $K_{D\_beta}$) is determined by trial and error. Note that this D term is only used to reduce the overshoot, but a larger value of $K_{D\_beta}$ will weaken the effect of stabilization. Therefore, $K_{D\_beta}$ should be continuously increased until the low-frequency vibration of the pendulum is just eliminated.

### 4.3.3 Single-PD-Loop Controller with Another P loop

In fact, no matter how the values of $K_{P\_beta}$ and $K_{D\_beta}$ are chosen, the above single PD loop structure is not able to stabilize the system around the inverted position. In other words, although the pendulum is balanced for a few seconds, eventually the arm will continuously move in one direction and the pendulum will fall down. This behaviour can be illustrated by the following response.

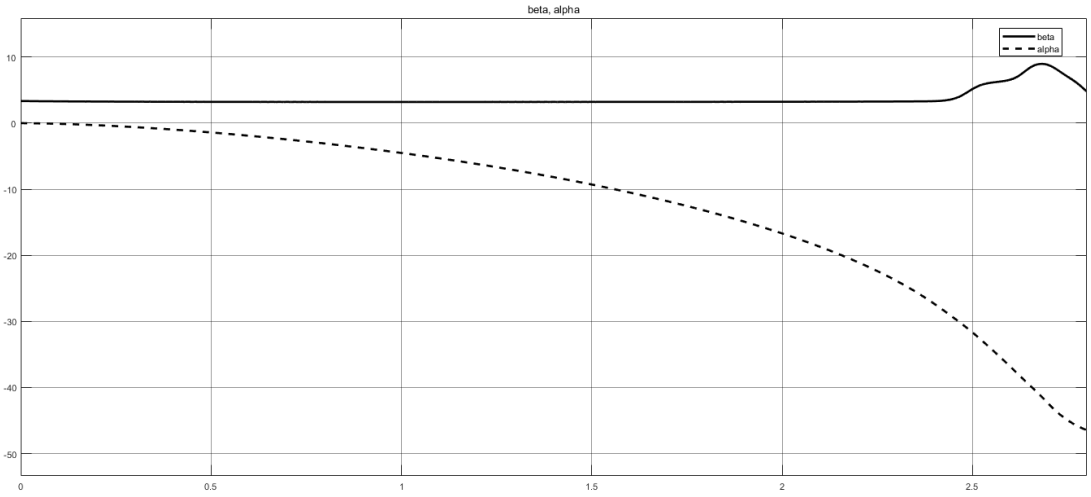


*Fig. 4.6. Response of the single-PD-loop controller*

The solid line shows the angle of the pendulum and the dashed line shows the angle of the arm. As shown in the above Fig. 4.6., at the moment when the pendulum loses balance, the moving direction of the arm is in such a way that it will prevent the pendulum from falling down. This is exactly the effect of the negative feedback loop of the angle of the pendulum, which means that this structure does its job properly. However, the pendulum still falls down which means that this amount of stabilizing effect is not enough to restore the pendulum back to the inverted position. In other

words, the arm is not moving fast enough to balance the pendulum. An intuitive solution is to accelerate the movement of the arm by adding another positive feedback loop of the angle of the arm. The structure is shown as the following block diagram.

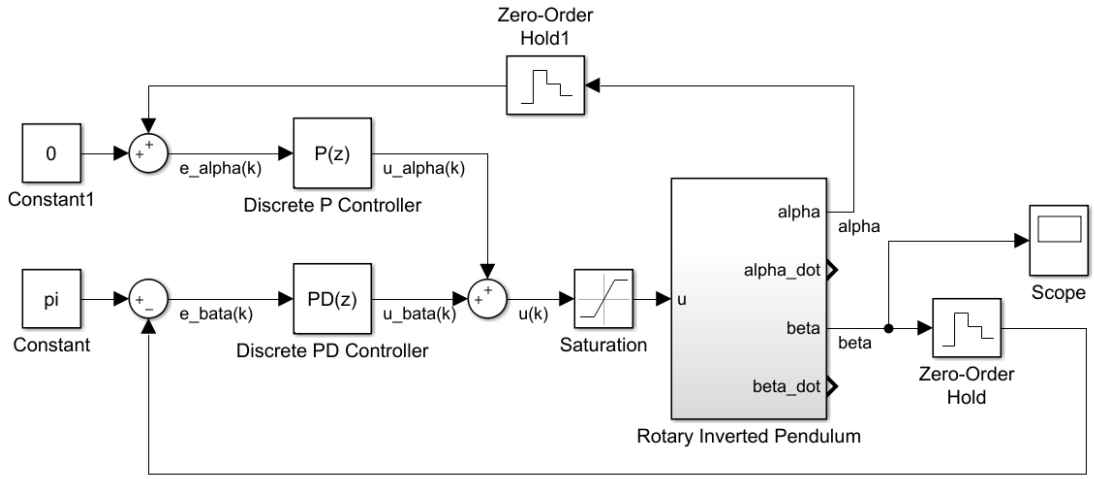


*Fig. 4.7. Block diagram of the single-PD-loop controller with another P loop*

$$u_\alpha(k) = K_{P\_alpha} e_\alpha(k)$$

$$u_\beta(k) = K_{P\_beta} e_\beta(k) + K_{D\_beta} \frac{e_\beta(k) - e_\beta(k-1)}{T_S} \tag{4.3}$$

$$u(k) = u_\alpha(k) + u_\beta(k)$$

The actual control input to the RIP system is the sum of the control outputs of the two loops. For now, only a single proportional positive feedback loop of the arm is used. This positive feedback loop will move the arm in the direction which is determined by the position of the arm. The dynamic response of the system using this structure can be described by the following process.

Initially, if the pendulum is released at, for example, +10° from the upright position ($\beta = 10°$), the arm will then tend to move in the negative direction because $u_\beta(k) < 0$ , $u_\alpha(k) = 0 \Rightarrow u(k) < 0$ , which will tend to restore the pendulum back to the inverted position. After this initial moment, the control output of the arm loop has the same sign as that of the pendulum loop, i.e. $u_\alpha(k) < 0$ because $\alpha < 0$ . In other words, before the pendulum is back to the inverted position, the arm loop will help the pendulum loop balance the pendulum more quickly. This is desirable because it is exactly the purpose of this addition arm loop.

However, at the moment when the pendulum is back at the inverted position, the control output of the pendulum loop is zero ($u_\beta(k) = 0$), but the control of the arm loop is nonzero ($u_\alpha(k) < 0$), which makes the arm continuously move in the negative direction. After that, the pendulum will deviate from the inverted position for a negative angle ($\beta < 0$) and accordingly, the control of the negative feedback pendulum loop should have a positive sign ($u_\beta(k) > 0$). Note that the arm is still in the negative direction ($\alpha < 0$), which means that the control of the arm loop still has the negative sign ($u_\alpha(k) < 0$). In this case, the two control outputs have opposite signs, which means that the arm loop counteracts the balancing effect of the pendulum loop. This is undesirable because it is against the purpose of this arm loop.

Therefore, after the pendulum is deviated from the inverted position, the angle of the pendulum will continuously increase ($|\alpha|\uparrow$), and the arm will continuously move in the negative direction until $u_\alpha(k)$ and $u_\beta(k)$ are in the same magnitude but opposite sign, which gives no control input to the system ($u(k) = u_\alpha(k) + u_\beta(k) = 0$). After that, $|u_\alpha(k)| < |u_\beta(k)| \Rightarrow u(k) > 0$, which means that the arm will move in the positive direction towards its zero-initial position. This behaviour is surprising because the angle of the arm has a trend of stabilization, even though the control loop of the arm is a positive feedback which generally destabilizes a system.

When the arm is moving towards its zero position, the two control loops still have opposite control outputs and the pendulum will never be restored back to the inverted position, because the arm loop counteracts the balancing effect provided by the pendulum loop.

The above whole process can be illustrated by the following response.

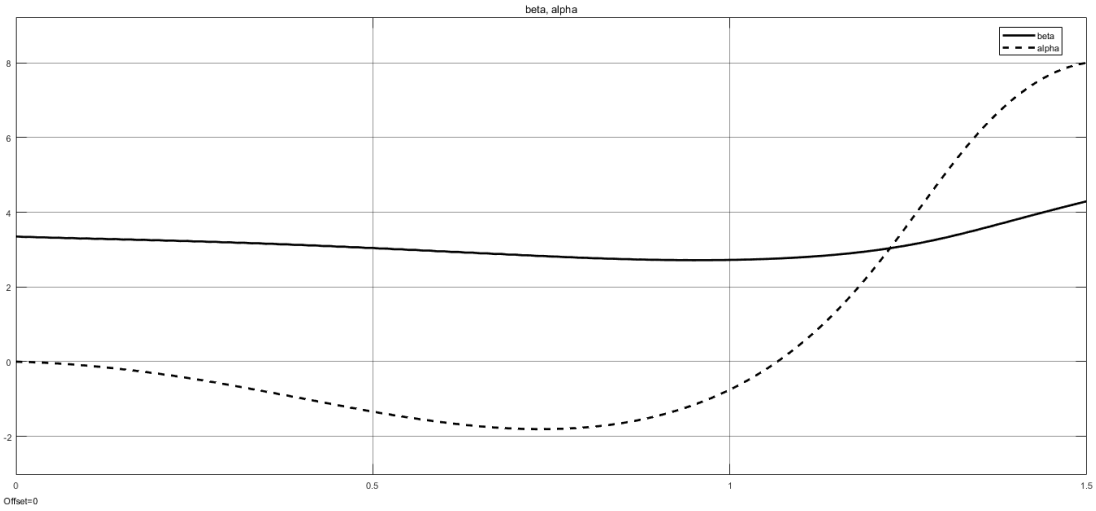


*Fig. 4.8. Response of the Single-PD-loop controller with another P loop*

As shown in the above Fig. 4.8., the arm is not continuously moving in one direction compared with Fig. 4.6., but has a trend of stabilization, which proves the discussion above. The sign of $K_{P\_alpha}$ is determined by experiments, while the value of it is obtained by trial and error. In fact, a larger value of $K_{P\_alpha}$ will help the pendulum back to its inverted position more quickly and hence the arm will start to move back to its initial position at a smaller angle. However, this structure is incomplete because it is still not able to stabilize the whole system, and therefore, the value of $K_{P\_alpha}$ still needs to be modified when the structure is complete.

In conclusion, this additional P loop of the arm helps balance the pendulum before the pendulum reaches its inverted position, and most importantly, this arm loop tends to stabilize the position of the arm, even though it is a positive feedback loop.

### 4.3.4 Double-PD-Loop Controller

Although the additional P loop has some stabilizing effects, the whole system is still unstable. The major reason is that after the pendulum is restored to the inverted position for the first time, the single P loop of the arm counteracts the stabilizing effect of the pendulum loop. This is undesirable and hence the structure needs to be modified. One solution is to change the P loop into a PD loop. The following block diagram illustrates this structure.

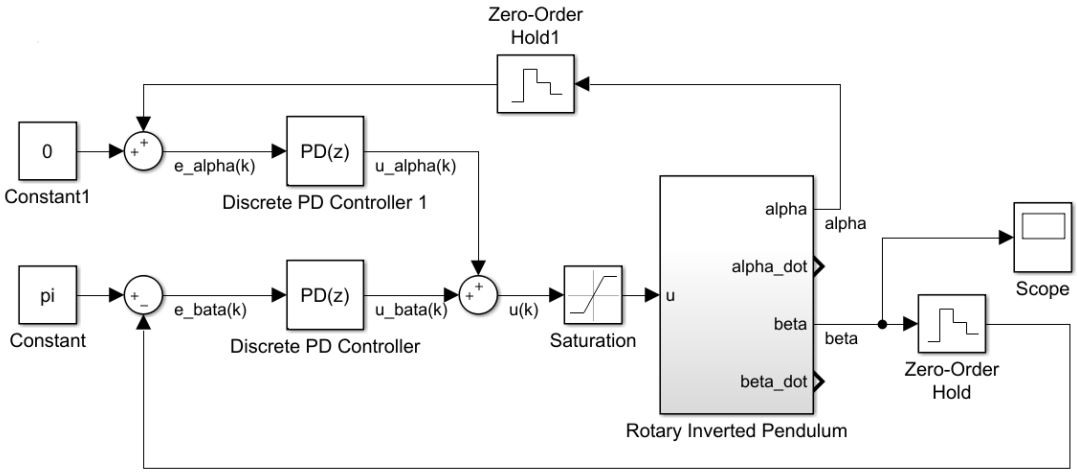


*Fig. 4.9. Block diagram of the double-PD-loop controller*

$$u_\alpha(k) = K_{P\_alpha}e_\alpha(k) + K_{D\_alpha}\frac{e_\alpha(k) - e_\alpha(k-1)}{T_S}$$

$$u_\beta(k) = K_{P\_beta}e_\beta(k) + K_{D\_beta}\frac{e_\beta(k) - e_\beta(k-1)}{T_S} \tag{4.4}$$

$$u(k) = u_\alpha(k) + u_\beta(k)$$

The effect of this additional derivative term in this positive feedback loop is not the same as that in the negative one. This D term will accelerate the movement of the arm in the same direction as the arm is heading to. Therefore, initially, this D term has the same sign as the P term does, which will help restore the pendulum back to its upright position more quickly together with the P term. After the pendulum is restored for the first time, the P term of the arm loop counteracts the stabilizing effect of the pendulum loop, but the D term of the arm loop enhances the stabilizing effect because it has the same sign as the value of $u_\beta(k)$. Therefore, after the pendulum is restored for the first time, it will still deviate from its inverted position for a negative angle, but this time, the magnitude of this angle will be decreased because the additional stabilizing effect provided by the D term of the arm loop. If the four parameters in these two PD controllers are chosen correctly, this structure should be able to restore the pendulum back to the inverted position for another time and at the same time the arm will stop. The sign of $K_{D\_alpha}$ is the same as that of $K_{P\_alpha}$ . In fact, it is not difficult to tune $K_{D\_alpha}$ without modifying the other three parameters to realize an oscillatory system as shown in the following Fig. 4.10.

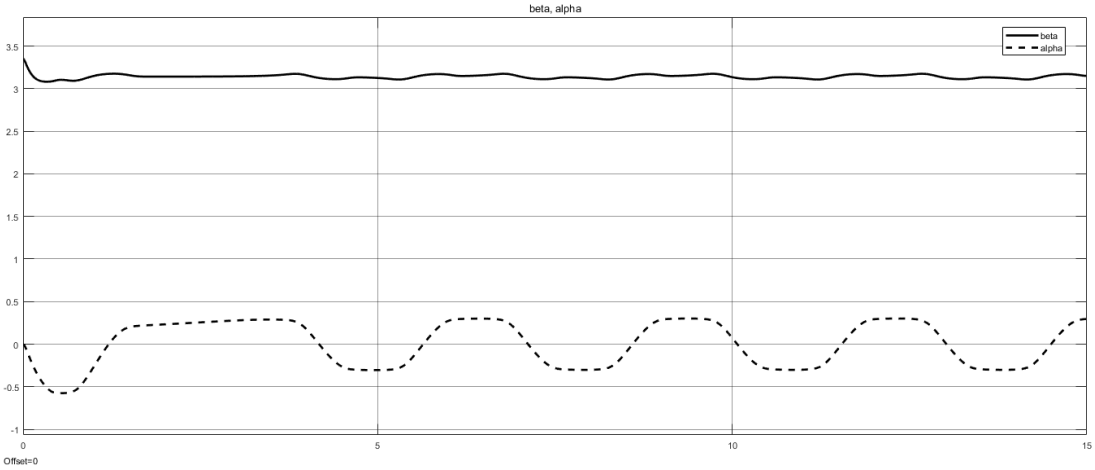


*Fig. 4.10. Response of the double-PD-loop controller*

However, in order to get a better stabilizing effect, the four parameters should be chosen carefully so that the arm can balance the pendulum in a single swing.

As discussed earlier, the key parameters to stabilize the system should be $K_{P\_alpha}$ and $K_{D\_alpha}$ because these two parameters determine the additional stabilizing effect provided by the arm loop. In fact, there is no obvious criterion of tuning these two parameters, hence they are obtained mainly by trial and error.

It is interesting to note that eventually the pendulum is stabilized at the inverted position without moving (i.e. $u_\alpha(k) = 0$ ), but the final stable position of the arm is not necessarily its zero position, in other words, the arm can be stabilized at different positions if the initial conditions of the system change. This implies that $u_\beta(k) \neq 0$ and hence $u(k) \neq 0$, which means that there is still a constant torque applied at the shaft of the motor, but the arm is not moving. This can be explained by the static friction applied at the motor shaft, which counteracts the effect of the constant input torque.

Note that the above simulation results are only used for illustration. The actual simulation shows that the arm and the pendulum will never stop at absolutely stable positions. The reason is that the simulation is only a simplified model of the actual physical system and the details will be discussed in Section 5.2.

### 4.3.5 Double-PD-loop Controller with Low-Pass Filter

As mentioned in Section 3.2.4, the data obtained from either the potentiometer or the encoder is of high-frequency noise. These data from noisy sensors cannot be used as an input signal to the PD controller directly because the derivative term will amplify the high-frequency noise and the actual control input will be totally different from the expected value. Therefore, two digital low-pass filters are used before the two discrete PD controllers to eliminate the high-frequency components as shown in the following block diagram.

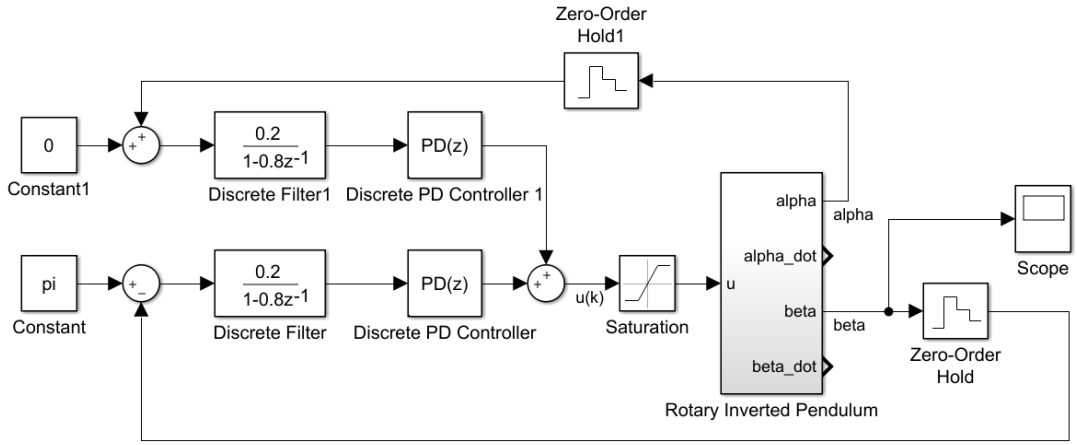


*Fig. 4.10. Block diagram of the double-PD-loop controller with low-pass filter*

The digital filter used here is the same as that used in the parameter estimation:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{0.2}{1 - 0.8z^{-1}}$$
$$\Rightarrow y[k] = 0.8y[k-1] + 0.2x[k] \tag{4.5}$$

However, it is still conventional filtering, not zero-phase shift filtering, which means that the input data is only filtered in the forward direction. Not using this filter has shown that the stabilizing effect was not good.

# Chapter 5 State Feedback via LQR

In this chapter, the theoretical study of the stabilization problem of the RIP system described by the mathematical model (2.17) is presented. Since all the analysis in this chapter is only verified in MATLAB (i.e. there is neither model uncertainty nor sensor noise), and the main purpose of this chapter is to give the theoretical support for the double-PD-loop controller developed by intuition, there is no need to consider the observation problem and hence all states are assumed to be measurable here.

## 5.1 Full State Feedback

Consider a linear time-invariant (LTI) system described by the following equations:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \end{aligned} \tag{5.1}$$

where $\boldsymbol{A} \in \boldsymbol{R}^{n\times n}$, $\boldsymbol{B} \in \boldsymbol{R}^{n\times m}$, $\boldsymbol{C} \in \boldsymbol{R}^{p\times n}$, $\boldsymbol{D} \in \boldsymbol{R}^{p\times m}$.

The static state feedback gives the control law:

$$\boldsymbol{u} = \boldsymbol{K}\boldsymbol{x} + \boldsymbol{r} \tag{5.2}$$

where $\boldsymbol{K} \in \boldsymbol{R}^{n\times m}$ is the feedback gain matrix and $\boldsymbol{r} \in \boldsymbol{R}^{m}$ is the external reference input.

The problem considered in this project is stabilization which is to design a state feedback controller so that the equilibrium $\boldsymbol{x} = \boldsymbol{0}$ of the compensated closed-loop system is asymptotically stable (in the sense of Lyapunov) when $\boldsymbol{r} = \boldsymbol{0}$.

The simplest way to stabilize a nonlinear system around an equilibrium point is to design a linear controller for the linearized model around that equilibrium point. This is feasible because if the linearized system around an equilibrium is asymptotically stable, then that equilibrium of the nonlinear system is locally asymptotically stable. Therefore, if possible, the designed linear controller should be able to stabilize the nonlinear system when the initial states are in the vicinity of the unstable equilibrium point.

The overall linearized system with identified parameters can be found by combining equations (2.23) and Table 3.3:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{5.3}$$

where:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 8.573 & -3.302 & -0.005013 \\ 0 & 110 & -4.766 & -0.06432 \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ 0.03539 \\ 0.05107 \end{bmatrix}$$

Note that the gain of the amplifier $K_u$ is increased by $\frac{255}{7200}$ because the model here is assumed to use STM32 as a controller, not Arduino.

The eigenvalues of the state matrix $\boldsymbol{A}$ are found by MATLAB:

$$\lambda_1 = 0, \qquad \lambda_2 = 10.3131, \qquad \lambda_3 = -10.7796, \qquad \lambda_4 = -2.9004\,.$$

The linearized system is unstable because one of the eigenvalues of A has positive real part which proves that the equilibrium $\boldsymbol{x_e}$ of the nonlinear system is unstable (in the sense of Lyapunov).

In most cases, the state feedback is used to solve the state regulation problem which means that the input signal should be able to drive the state vector of the system to zero within a given speed of convergence. But before that, a state feedback controller should first make the closed-loop system asymptotically stable. Since the uncontrollable modes of the linear system is not modified by state feedback, the LTI system is stabilizable only if the uncontrollable modes of the continuous system have negative real parts. The controllability matrix of the linearized model (5.3) is formed as follows:

$$\boldsymbol{C_m} = [\boldsymbol{B} \quad \boldsymbol{AB} \quad \boldsymbol{A^2B} \quad \cdots \quad \boldsymbol{A^{n-1}B}] = [\boldsymbol{B} \quad \boldsymbol{AB} \quad \boldsymbol{A^2B} \quad \boldsymbol{A^3B}] \tag{5.4}$$

The rank of it is calculated by MATLAB:

$$\text{rank}\,(\boldsymbol{C_m}) = 4 = n \tag{5.5}$$

From the above calculations, the controllability matrix of this linear system is full rank which means that the linearized system is completely state controllable. This implies

that all closed-loop eigenvalues of the linearized system can be assigned arbitrarily, and hence the linearized system can be returned to the zero state within finite time by using state feedback.

It is interesting to note that the RIP system is underactuated but controllable at the inverted position. In fact, an underactuated system cannot follow arbitrary trajectories but can arrive at arbitrary points in state space [3].

## 5.2 Linear Quadratic Regulator (LQR)

The controllability test above proves that there exists a trajectory to the upright equilibrium point, but the number of solutions is infinite. Here gives a method of designing the feedback gain matrix $\boldsymbol{K}$ by considering the stabilizing problem as an optimal control problem, i.e. the so-called Linear Quadratic Regulator (LQR) problem. Detailed information on this topic can be found in extensive literature on optimal control. In the following discussion, only brief central ideas of this topic are given.

The LQR problem is to find $\boldsymbol{u}(t)$ such that the following quadratic cost function is minimized:

$$J = \int_0^\infty (\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u})\, dt \tag{5.6}$$

for any initial state $\boldsymbol{x}(0)$, where $\boldsymbol{Q} = \boldsymbol{Q}^T$ , $\boldsymbol{R} = \boldsymbol{R}^T$ and $\boldsymbol{Q} > \boldsymbol{0}$ , $\boldsymbol{R} > \boldsymbol{0}$ (i.e. symmetric and positive-definite)

If we further let $\boldsymbol{Q}$ and $\boldsymbol{R}$ to be diagonal, then the term $\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x}$ and $\boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u}$ can be written as follows:

$$\begin{aligned} \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} &= \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix} \begin{bmatrix} q_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & q_n \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 q_1 & \dots & x_n q_n \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \\ &= {x_1}^2 q_1 + \cdots + {x_n}^2 q_n = \sum_{i=1}^{n} {x_i}^2 q_i \end{aligned} \tag{5.7}$$

$$\boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u} = [u_1 \quad \dots \quad u_n]\begin{bmatrix} r_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & r_n \end{bmatrix}\begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} = [u_1 r_1 \quad \dots \quad u_n r_n]\begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}$$

$$= {u_1}^2 r_1 + \cdots + {u_n}^2 r_n = \sum_{i=1}^{n} {u_i}^2 r_i$$

It can be seen from the above equations that the term $\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}$ is the sum of squared state $x_i$ with weighting coefficient $q_i$. To minimize the improper integral of this term, $\boldsymbol{x}(t)$ should approach zero as $t$ goes to infinity. As with the term $\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}$ , the term $\boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}$ minimizes $\boldsymbol{u}(t)$ so that it remains small. Obviously, the relative 'size' of the two weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ determine trade-offs between the speed of response and size of control input. It seems that this optimized $\boldsymbol{u}(t)$ should at least make the equilibrium $\boldsymbol{x} = \boldsymbol{0}, \boldsymbol{u} = \boldsymbol{0}$ of the LTI system (5.1) asymptotically stable and result in a finite performance index. The detailed work of analysing the stability of the optimal closed-loop system can be found in [17].

Assume the LTI system (5.1) is completely state-controllable, then the optimal control input is given by the following state feedback law which is independent of the initial states $\boldsymbol{x}(0)$:

$$\boldsymbol{u}^*(t) = \boldsymbol{F}^*(t)\boldsymbol{x}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}^*\boldsymbol{x}(t) \tag{5.8}$$

where $\boldsymbol{P}^*$ is the symmetric positive definite solution of the algebraic Riccati equation:

$$\boldsymbol{A}^T\boldsymbol{P}^* + \boldsymbol{P}^*\boldsymbol{A} - \boldsymbol{P}^*\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}^* + \boldsymbol{Q} = \boldsymbol{0} \tag{5.9}$$

The minimum cost is given by:

$$J_{min} = \boldsymbol{x}(0)^T\boldsymbol{P}^*\boldsymbol{x}(0) \tag{5.10}$$

The optimal closed-loop system:

$$\dot{\boldsymbol{x}} = [\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}^*]\boldsymbol{x}(t) \tag{5.11}$$

is asymptotically stable.

It is interesting to point out the limiting behaviour of the closed-loop eigenvalues as a function of the weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$. Here only a simple case is considered, where the system is assumed to be LTI and single-input. The weighting matrices are supposed to be diagonal with same entries in the main diagonal, i.e. $\boldsymbol{Q} = q\boldsymbol{I}$ and $\boldsymbol{R} = r$.

It has been proved in [18] that the closed-loop eigenvalues are the stable roots of

$$1 + \frac{q}{r}\boldsymbol{H}^T(-s)\boldsymbol{H}(s) = 0 \tag{5.12}$$

where $\boldsymbol{H}(s) = (s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}$

If $\frac{q}{r} \to \infty$ which means that the weighting of response is much larger than that of control effort in the LQR cost function, this case can be regard as a so-called "cheap control" where any arbitrary control input can be used by the optimal control law. In this case, the compensated closed-loop eigenvalues will be moved to the stable zeros of $\boldsymbol{H}^T(-s)\boldsymbol{H}(s)$; that is the stable zeros of $\boldsymbol{H}(s)$ and the mirror images of unstable zeros of $\boldsymbol{H}(s)$. The control law provides the fastest possible response and the feedback gain matrix $\boldsymbol{K}$ becomes unbounded.

If $\frac{r}{q} \to \infty$ which means that the weighting of control effort is much larger than that of the response in the LQR cost function, this case can be regard as a so-called "expensive control" where minimum control energy can be used by the optimal control law. In this case, the control law will not move any of the uncompensated open-loop eigenvalues except those that have positive real parts; that is the unstable open-loop eigenvalues in the RHP will be moved to the mirror images in the LHP. In other words, the compensated closed-loop eigenvalues will be moved to the stable poles of $\boldsymbol{H}^T(-s)\boldsymbol{H}(s)$. The control law in this situation uses minimum control effort to stabilize the system and does not speed up the response of the system.

One method of tuning the weights $\boldsymbol{Q}$ and $\boldsymbol{R}$ is proposed by Bryson's rule. A simplified version of it is to choose the diagonal matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ as follows:

$$\begin{aligned} q_i &= \frac{1}{(x_i)^2{}_{max}} \\ r_i &= \frac{1}{(u_i)^2{}_{max}} \end{aligned} \tag{5.13}$$

where $(x_i)_{max}$ and $(u_i)_{max}$ denotes the maximum acceptable value of each state or input respectively.

As discussed previously, the maximum control input is determined by the number

representing the maximum duty cycle of the PWM signal. In this project, the saturation is set to be ±6900, i.e. 95.8% duty cycle. Therefore, $(u)_{max} = 6900$ is used as a first iteration.

After some design iterations, the values of $\boldsymbol{Q}$ and $\boldsymbol{R}$ are chosen as follows:

$$\boldsymbol{Q} = \begin{bmatrix} \frac{1}{0.1^2} & 0 & 0 & 0 \\ 0 & \frac{1}{0.1^2} & 0 & 0 \\ 0 & 0 & \frac{1}{1.5^2} & 0 \\ 0 & 0 & 0 & \frac{1}{1.2^2} \end{bmatrix} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 0.4444 & 0 \\ 0 & 0 & 0 & 0.6944 \end{bmatrix} \quad (5.14)$$

$$\boldsymbol{R} = \frac{1}{6900^2} = 2.1 \times 10^{-8}$$

This gives the state-feedback matrix:

$$\boldsymbol{K} = [-0.6900 \quad 2.2287 \quad -0.2180 \quad 0.2212] \times 10^5 \quad (5.15)$$

This state-feedback controller gives the following response when the initial state is $\boldsymbol{x}^T(0) = [0 \quad 3.35 \quad 0 \quad 0]$:

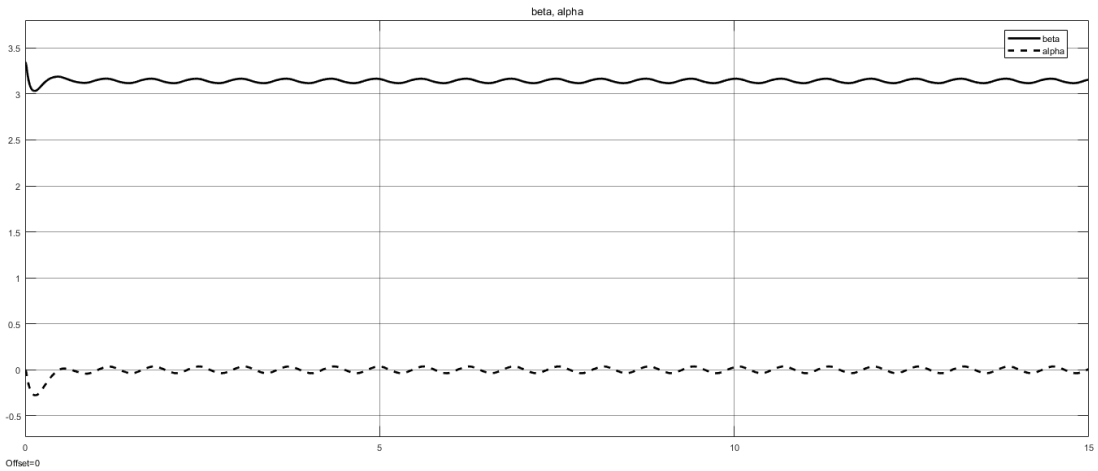


*Fig. 5.1. System response with a state-feedback controller when* $\boldsymbol{x}^T(0) = [0 \quad 3.35 \quad 0 \quad 0]$

As shown in the above Fig. 5.1., the controller stabilizes the nonlinear system, but both the arm and the pendulum are not at absolutely stable positions. In fact, whatever the values of $\boldsymbol{Q}$ and $\boldsymbol{R}$, the nonlinear model built in this project cannot be stabilized perfectly. The major reason for that is the contact forces involved in the real physical system are very complex and difficult to model so that these forces are simplified as

simple Coulomb and viscous frictions in this project. In practice, the pendulum can stay at, for example, ±2° from the inverted position without applying any force to it because of the contact forces between the pendulum and the revolute joint. In addition, the arm can stay at rest even if there is a small torque applied to it, because of the static friction at the motor shaft. These contact forces are the major difficulties in simulating multi-degrees-of-freedom manipulators and robotics locomotion. Building such an accurate simulating environment requires strong capabilities in modelling and numerical analysis, which is beyond the scope of this project.

## 5.3 Relationship between Double PD and State Feedback

As mentioned at the beginning of this chapter, the major purpose of this chapter is to give theoretical support for the double-PD-loop controller. The double-PD-loop controller discussed in the previous chapter, which is also the structure implemented in the microcontroller, can be reorganized into the state-feedback form. The PD loop of the pendulum gives the feedback of both the angle and angular velocity (two states) of the pendulum, while the PD loop of the arm provides feedback of the two states of the arm. This can be proved by rewritten equations (4.4) as follows:

$$\begin{aligned} u_\alpha(k) &= K_{P\_alpha} e_\alpha(k) + K_{D\_alpha} \frac{e_\alpha(k) - e_\alpha(k-1)}{T_S} \\ &= K_{P\_alpha} \alpha(k) + K_{D\_alpha} \dot{\alpha}(k) \\ u_\beta(k) &= K_{P\_beta} e_\beta(k) + K_{D\_beta} \frac{e_\beta(k) - e_\beta(k-1)}{T_S} \\ &= K_{P\_beta} \beta(k) + K_{D\_beta} \dot{\beta}(k) \end{aligned} \tag{5.16}$$

where

$\alpha(k) = e_\alpha(k)$ is the angle of the pendulum from its inverted position.

$\dot{\alpha}(k) = \frac{e_\alpha(k) - e_\alpha(k-1)}{T_S}$ is the Euler approximation of the angular velocity of the pendulum.

$\beta(k) = e_\beta(k)$ is the angle of the arm from its zero-initial position.

$\dot{\beta}(k) = \frac{e_\beta(k) - e_\beta(k-1)}{T_S}$ is the Euler approximation of the angular velocity of the arm.

Therefore, the control input can be rewritten as follows:

$$\begin{aligned} u(k) &= u_{\alpha}(k) + u_{\beta}(k) \\ &= K_{P\_alpha}\alpha(k) + K_{D\_alpha}\dot{\alpha}(k) + K_{P\_beta}\beta(k) \\ &+ K_{D\_beta}\dot{\beta}(k) \\ &= K_{P\_alpha}\alpha(k) + K_{P\_beta}\beta(k) + K_{D\_alpha}\dot{\alpha}(k) \\ &+ K_{D\_beta}\dot{\beta}(k) = \boldsymbol{K}\boldsymbol{x}(k) \end{aligned} \tag{5.17}$$

which is exactly the control law of a full state feedback controller.

It is interesting to mention that when designing the stabilizing gain $\boldsymbol{K}$ via LQR in this project, the signs of each parameters in the gain matrix are unique, no matter what the values of $\boldsymbol{Q}$ and $\boldsymbol{R}$ are chosen:

$$\boldsymbol{K} = \left[K_{P_{alpha}} > 0 \quad K_{P_{beta}} < 0 \quad K_{D_{alpha}} > 0 \quad K_{D_{beta}} < 0\right] \tag{5.18}$$

Note that the optimal gain matrix designed via LQR will always make the linearized system asymptotically stable. Therefore, to make the nonlinear equilibrium locally asymptotically stable, the two parameters related to $\alpha$ are always positive, while the other two parameters related to $\beta$ are always negative. This indirectly proves that the PD loop of the arm is a positive feedback, while the PD loop of the pendulum is a negative feedback.

# Chapter 6 Conclusions and Recommendations

## 6.1 Conclusions

This project investigates the modelling and control system design of a real rotary inverted pendulum. A hybrid controller is built including a swing-up, a switching and a stabilizing part, which take over from each other at the right moment. This hybrid controller is tested on the apparatus and can successfully swing up the pendulum and balance it in the upright position. The model built in this project can describe the overall dynamic behaviour of the system in simulation environment but is not accurate enough to be used to design actual control system. Instead, a double-PD-loop controller is designed by intuition to stabilize the pendulum, which is tuned by experience, as well as trial and error. This controller is proved to be a state feedback controller and the detailed theoretical background of this structure is analysed.

This report can serve as background information on rotary inverted pendulum and a guide for future students working on similar projects. The outcome of this thesis can also serve as a foundation for further research on related topics including double inverted pendulum, Segway self-balancing scooter or bipedal walking robot.

## 6.2 Recommendations

Although the methods discussed in this project work properly on a real rotary inverted pendulum system, serval improvements can be done. Many of the suggestions here can be found in specific sections of this report.

As suggested in Section 3.5, the model built in this project is not accurate. One reason is that the mathematical model itself is only an approximation. In fact, the input-output behaviour of the PWM amplifier is nonlinear and cannot be modelled as a single constant gain. Other unmodeled effects like static friction and armature induction of the DC motor are recommended to be considered. The other reason is that the parameters in the model are not accurate. Some DC motor parameters like the armature

resistance can be determined more accurately by experiments without estimation. The moment of inertia of the pendulum is approximated using the formula for a uniform rod with negligible thickness, which is inaccurate and should be estimated if possible. An EKF or UKF is recommended to be designed for estimating those parameters related to the total setup. These unmodeled effects or inaccurate parameters can be overcome by tuning the parameters of the controller when designing control systems. However, if there is an accurate model available, an optimal controller could be found, and the performance of the system would be much better.

As discussed in Section 4.1, the swing-up controller has no control of the velocity of the pendulum, which makes the stabilizing controller not able to catch the pendulum every time. The reason for using this swing-up controller is due to the blind zone of the potentiometer. If such a blind zone does not exist or the potentiometer is replaced by an encoder, a more sophisticated control algorithm can be designed by solving a completely nonlinear problem.

In the current apparatus, the angular velocities of the pendulum and the arm are generated by the derivative term of the discrete PD controllers, i.e. estimated using first-order Euler approximation. This will introduce significant noise into the system which is eliminated by the designed low-pass filter. However, this can be improved by building a reduced-order observer for estimating these two values around the inverted equilibrium point.

# References

[1]. Dare A. Wells, *Schaum's outline of theory & problems of lagrangian dynamics*. New York, NY: McGraw Hill Professional, 1967.

[2]. [online]. Available: https://uk.mathworks.com/help/simulink/ug/zero-crossing-detection.html [Accessed 20 Mar. 2018].

[3]. Russ Tedrake. Underactuated Robotics: Learning, Planning, and Control for Efficient and Agile Machines [Online]. Available: https://homes.cs.washington.edu/~todorov/courses/amath579/Tedrake_notes.pdf [Accessed 20 Mar. 2018].

[4]. C. K. Chui and G. Chen, *Kalman Filtering: with Real-Time Applications*, 4th ed. Springer-Verlag Berlin Heidelberg, 2009.

[5]. Wang Z, Chen YQ and Fang N, "Minimum-time swing-up of a rotary inverted pendulum by iterative impulsive control," Proceeding of the 2004. American Control Conference. Boston, MA, 2004: 1335-1340.

[6]. Astrom KJ and Furuta K, "Swinging up a pendulum by energy control," Automatica, 2000;36:287-295.

[7]. M. Yamakita, T. Hoshino, K. Furuta, "Control practice using pendulum", American Control Conference 1999. Proceedings of the 1999, vol. 1, pp. 490-494 vol.1, 1999, ISSN 0743-1619.

[8]. Torres-Pomales W and Gonzale OR, "Nonlinear control of swing-up inverted pendulum. Proceedings of the IEEE International Conference on control application," Dearborn, MI, 1996: 259-264.

[9]. Dotoli M, Maione B, Naso D and Turchiano B, "Fuzzy sliding model control for inverted pendulum swing up with restricted travel," Proceedings of the 10th IEEE International on Fuzzy Systems. Melbourne, Vic, 2001: 753-756.

[10]. Acosta JA, Gordillo F and Aracil J, "A new swing-up law for the Furuta pendulum," International Journal of Control, 2003; 76(8): 836-844.

[11]. Md. Akhtaruzzaman, and A. A. Shafie, "Modeling and Control of a Rotary Inverted Pendulum Using Various Methods," Comparative Assessment and Result Analysis. Proceedings of the 2010 IEEE International Conference on Mechatronics and Automation, Xi'an, China; 2010: 1342-1347.

[12]. M.B.Arnolds, "Identification and control of the Rotary Inverted Pendulum," 2003, DCT report 2003.100.

[13]. Ryu C, Choi BJ and Choi BY, "Design of a fuzzy logic controller for a rotary-type inverted pendulum system," International Journal of Fuzzy Logic and Intelligent Systems, 2002; 2(2): 109-114.

[14]. Yu, C.H., Wang, F.C. and Lu, Y.J, "Robust control of a Furuta pendulum," SICE Annual Conference, Aug. 2010, pp. 2559–2563 (2010)

[15]. Y.-F. Chen and A.-C. Huang, "Adaptive control of rotary inverted pendulum system with time varying uncertainties," Nonlinear Dynamics, vol. 76, no. 1, pp. 95–102, 2014.

[16]. P. Faradja, G. Qi, and M. Tatchum, "Sliding mode control of a Rotary Inverted Pendulum using higher order differential observer," in Proceedings of the 14th International Conference on Control, Automation and Systems (ICCAS '14), pp. 1123–1127, October 2014.

[17]. Anderson, Brian D. O., and John B. Moore, *Optimal Control: Linear Quadratic Methods*, Englewood Cliffs, N.J.: Prentice Hall, 1990.

[18]. P. J. Antsaklis and A. N. Michel, *Linear Systems*. Cambridge, MA: Birkhäuser Basel, 2006.

# Appendix A: MATLAB Code for Off-Line Simulation via EKF

**File:** ModelParameters.m

**Description:** This file contains the model parameters. For off-line simulation purpose, some parameters should be modified to predefined values.

```
M1=0.035;
L1=0.07;
L0=0.145;
G=9.81;
C1=1.4053e-05;
J1=M1*(0.16*0.16)/12+M1*(L1*L1);
Kf1=0.00044156;
Kt=0.97*12/(11000/60*2*pi);
Kb=Kt;
Ku=0.025;
Kg=20;
Ra=(1-0.97)*12/0.36;
n=0.66*0.098/(0.36*Kt*Kg);
J0=0.0038227;
C0=0.011546;
Kf0=0.0665;
```

**File:** StateTransitionFcn.m

**Description:** This file contains the state transition function (3.19).

```
function x_Next = StateTransitionFcn( x )
    M1=0.035;
    L1=0.07;
    L0=0.145;
    G=9.81;
    J1=M1*(0.16*0.16)/12+M1*(L1*L1);
    Ts=0.005;

    x_Next=zeros(4,1);
    x_Next(2)=(1-(x(3)*Ts/J1))*x(2)-Ts*x(4)*sigmoid(x(2))/(J1)-
M1*G*L1*Ts*sin(x(1))/(J1);
    x_Next(1)=x(1)+Ts*x_Next(2);
    x_Next(3)=x(3);
    x_Next(4)=x(4);
end

function y=sigmoid(x)
    k=20;
    y=1-(2/(exp(2*k*x)+1));
end
```

**File:** MeasurementFcn.m

**Description:** This file contains the measurement function (3.20).

```
function y = MeasurementFcn( x )
    y(1)=x(1);
    y(2)=x(2);
end
```

**File:** ExtendedKalmanFilterSimulation.m

**Description:** This file contains the extended Kalman filter used for off-line simulation. y is generated using To Workspace block in Simulink and is a structure with time.

```
close all;

N=numel(y.time);
t=y.time;
x_estimated=zeros(4,N);

%Generate beta_dot using Euler approximation
y_dot=zeros(1,N);
for k=2:N
    y_dot(k)=(y.signals.values(k)-y.signals.values(k-1))/0.005;
end

measurement=zeros(2,N);
for k=1:N
    measurement(1,k)=y.signals.values(k);
    measurement(2,k)=y_dot(k);
end

%Configure the EKF
ekf =
extendedKalmanFilter(@StateTransitionFcn,@MeasurementFcn,[y.signals.
values(1);0;1e-4;1e-4]);
ekf.ProcessNoise = diag([1e-10,1e-10,1e-10,1e-10]);
ekf.MeasurementNoise = diag([1e-3,1e-3]);
ekf.StateCovariance = 1e-3;

%Perform estimation by predict and correct
for k=1:N
    [PredictedState,PredictedStateCovariance] = predict(ekf);
    [CorrectedState,CorrectedStateCovariance] =
correct(ekf,measurement(:,k));
    for i=1:4
        x_estimated(i,k)=ekf.State(i);
    end
end

%Calculate errors
error_C=(ekf.State(3)-1e-4)*100/1e-4
error_Kf=(ekf.State(4)-1e-4)*100/1e-4
```

```
%Generate figures with estimated parameters
figure('Name','Off-line Simulation')
subplot(2,2,1);
plot(t,x_estimated(1,:))
title('Beta')
xlabel('Time (sec)')
ylabel('Beta (rad)')

subplot(2,2,2);
plot(t,x_estimated(2,:))
title('Beta dot')
xlabel('Time (sec)')
ylabel('Beta dot (rad/s)')

subplot(2,2,3);
plot(t,x_estimated(3,:))
ylim([0 5e-4])
title('C1')
xlabel('Time (sec)')
ylabel('C1 (kgm^2/s)')

subplot(2,2,4);
plot(t,x_estimated(4,:))
ylim([0 5e-4])
title('Kf1')
xlabel('Time (sec)')
ylabel('Kf1 (kgm^2)')
```

# Appendix B: MATLAB Code for Designing a Digital Low-Pass Filter

**File:** DigitalLowPassFilterDesign.m

**Description:** This file contains necessary procedures for designing a digital low-pass filter. y is the output data of beta recorded from a microcontroller and saved in the MATLAB workspace.

```
close all;

N=length(y);
Ts=0.005;
wc=45; %designed cut-off frequency
wc_normalized=wc*Ts; %Normalized frequency in rad/sample (2pi
correspond to 1)
a=2-cos(wc_normalized)-
sqrt((cos(wc_normalized))*(cos(wc_normalized))-
4*cos(wc_normalized)+3);

%Generate the frequency response of the input signal
FFT=fft(y);
w1=linspace(0,2*pi/Ts,N+1);
gain=20*log10(abs(FFT));
phase=unwrap((angle(FFT))*180/pi);
figure('Name','Frequency Response of Input Data');
subplot(2,1,1);
semilogx(w1(1:floor(N/2)),gain(1:floor(N/2)))
xlabel('Frequency (rad/sec)')
ylabel('Magnitude (dB)')
subplot(2,1,2);
semilogx(w1(1:floor(N/2)),phase(1:floor(N/2)))
xlabel('Frequency (rad/sec)')
ylabel('Phase (deg)')

%Generate the frequency response of the designed low-pass filter
[h,w2] = freqz(1-a,[1 -a],5000);
figure('Name','Frequency Response of First-Order Low Pass Filter');
subplot(2,1,1);
semilogx(w2/Ts,20*log10(abs(h)))
xlabel('Frequency (rad/sec)')
ylabel('Magnitude (dB)')
subplot(2,1,2);
semilogx(w2/Ts,(angle(h)*180/pi))
xlabel('Frequency (rad/sec)')
ylabel('Phase (deg)')

%Compare the magnitude response of the input signal and designed
low-pass filter
figure('Name','Frequency Response of Input Data and Filter');
subplot(2,1,1);
semilogx(w1(1:floor(N/2)),gain(1:floor(N/2)))
```

```
xlabel('Frequency (rad/sec)')
ylabel('Magnitude (dB)')
title('Input Data')
subplot(2,1,2);
semilogx(w2/Ts,20*log10(abs(h)))
xlabel('Frequency (rad/sec)')
ylabel('Magnitude (dB)')
title('Low-Pass Filter')
text(wc,-3,'cut-off')

%Perform the conventional filtering
y_filtered(1)=y(1);
for i=2:N
    y_filtered(i)=a*y_filtered(i-1)+(1-a)*y(i);
end

%Compare the input noisy signal and output filtered signal
t=linspace(0,Ts*(N-1),N);
figure('Name','First-Order Low Pass Filter')
plot(t,y)
hold on
plot(t,y_filtered)
legend('noisy beta','conventional filtering')
xlabel('Time (sec)')
ylabel('Beta (rad)')

%Perform the zero-phase filtering
y_filtfilt=filtfilt(1-a,[1 -a],y);
figure('Name','Zero Phase Delay Filter')
plot(t,y)
hold on
plot(t,y_filtfilt)
legend('noisy beta','zero-phase filtering')
xlabel('Time (sec)')
ylabel('Beta (rad)')
```

# Appendix C: MATLAB Code for Off-Line Experiments via EKF

**File:** ExtendedKalmanFilterExperiment.m

**Description:** This file contains the extended Kalman filter used for off-line experiments. Note that this file should be run together with the ModelParameters.m, StateTransitionFcn.m and MeasurementFcn.m in Appendix A. y is the output data of beta recorded from a microcontroller and saved in the MATLAB workspace.

```
close all;

Ts=0.005;

N=numel(y);
t=0:Ts:(N-1)*Ts;
x_estimated=zeros(4,N);

%Generate beta_dot using Euler approximation
y_dot=zeros(1,N);
for k=2:N
    y_dot(k)=(y(k)-y(k-1))/Ts;
end

measurement=zeros(2,N);
for k=1:N
    measurement(1,k)=y(k);
    measurement(2,k)=y_dot(k);
end

%Configure the EKF
ekf =
extendedKalmanFilter(@StateTransitionFcn,@MeasurementFcn,[y(1);0;1e-
4;1e-4]);
ekf.ProcessNoise = 1e-14;
ekf.MeasurementNoise = 2e0;
ekf.StateCovariance = 1e-2;

%Perform estimation by predict and correct
for k=1:N
    [PredictedState,PredictedStateCovariance] = predict(ekf);
    [CorrectedState,CorrectedStateCovariance] =
correct(ekf,measurement(:,k));
    e(k) = (y(k) - ekf.State(1))*100/y(k);
    j=k-floor((9*N/10));
    if(j>0) %Record the last 10% estimated parameters in x3 and x4
        x3(j) = ekf.State(3);
        x4(j) = ekf.State(4);
    end
    for i=1:4
        x_estimated(i,k)=ekf.State(i);
```

```
    end
end

%Generate the values of the estimated parameters and calculate the
%presition of each estimation
C1=mean(x3)
Kf1=mean(x4)
precision_C=(max(x3)-C1)*100/C1
precision_Kf1=(max(x4)-Kf1)*100/Kf1

%Generate figures with estimated parameters
figure('Name','Experiment1')
subplot(2,2,1);
plot(t,x_estimated(1,:))
title('Beta')
xlabel('Time (sec)')
ylabel('Beta (rad)')

subplot(2,2,2);
plot(t,x_estimated(2,:))
title('Beta dot')
xlabel('Time (sec)')
ylabel('Beta dot (rad/s)')

subplot(2,2,3);
plot(t,x_estimated(3,:))
ylim([0 5e-5])
title('C1')
xlabel('Time (sec)')
ylabel('C1 (kgm^2/s)')

subplot(2,2,4);
plot(t,x_estimated(4,:))
ylim([0 1e-3])
title('Kf1')
xlabel('Time (sec)')
ylabel('Kf1 (kgm^2)')
```