Lebanese University

Faculty of Engineering Branch 3

Prepared by Fouad Atwi (5180)

Presented to Prof. Youssef Harkouss

ABSTRACT

This project aims to test the capabilities of several control techniques by applying them on a rotary inverted pendulum and studying its response.

Control Methods tested in this project are Fuzzy Logic controllers, PID controllers and Full State Feedback controller

The implementation is done using MATLAB and Simulink.

TABLE OF CONTENTS

TABLE OF FIGURES

INTRODUCTION

A control system manages, commands, directs, or regulates the behavior of other devices or systems using control loop.

Control systems are present in many systems around us and are used for various applications.

Control systems vary in complexity and each has its own properties

CHAPTER 1: CONTROLLERS

Fuzzy Logic Controller

Introduction

An *FLC*[1] is a controller based on fuzzy logic.

Fuzzy logic is a mathematical system. It analyses analog input values in terms of logical variables that take continuous values (usually between 0 and 1) rather than taking discrete values (either 0 or 1).

The term "fuzzy" refers to the fact that the logic involved can deal with concepts that cannot be expressed as the "true" or "false" but rather "partially true".

Fuzzy logic is used to control tasks that can be successfully performed by humans since it relies on human experience for the controller to be designed. In turn, *FLC* provides a controller that could be understood by a human operator.

Theory

Fuzzy controllers are very simple conceptually, they consist of an input stage, a processing stage, and an output stage.

The input stage maps the inputs to the appropriate *membership functions*[2] and truth values.

The processing stage invokes each appropriate rule and generates a result for each, then combines the results of the rules.

The output stage converts the combined result back into a specific control output value.

A block diagram for a generic *fuzzy logic feedback controller*:

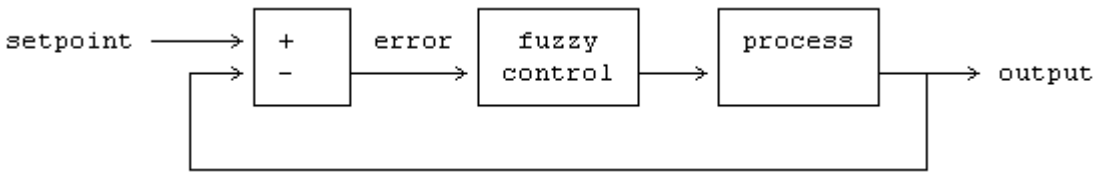

Figure 1: Fuzzy Logic Feedback block diagram

[1] Fuzzy Logic Controller
[2] Appendix

Example on inputs and outputs[3]:

PID Controller

Introduction

PID[4] controller is a widely used controller due to its simplicity and high effectiveness in most situations.

It has three main components that allow this controller to work and they are as its name suggests:

- Proportional Controller
- Integral Controller
- Derivative Controller

Theory

The Proportional Controller:

Produces an output proportional to the input signal. It does so by simply multiplying the input signal with a gain *Kp*.

The theory behind this controller is to provide an output in direct relation to input hence follow it somehow.

The Integral Controller:

Produces an output proportional to the integral of the input signal. This is achieved by multiplying the input signal by $\frac{1}{s}$ in the s-domain to get the integral of the signal. It is then multiplied by the integral gain *Ki*.

The theory behind this controller is to generate an output related to the integral of the input hence eliminating steady state error. This is done since even small errors provide big enough outputs when integrated long enough.

The Derivative Controller:

Produces an output proportional to the derivative of the input signal. This could be done in theory by multiplying the input signal by s in the s-domain then multiplying it by the derivative gain *Kd*.

---

[3]Appendix: The structure and the figures of the example are from "https://en.wikipedia.org/wiki/Fuzzy_control_system"

[4] Proportional-Integral-Derivative

The theory behind this controller is to provide outputs directly related to the input signal. Hence the controller is better at following the input by studying how rapidly it is changing and providing an output according to it.

However, *high frequency noise*[5] has a huge effect on the derivative controller. Therefore, the derivative controller is designed to incorporate a low pass filter to filter out higher frequency noise and disturbances.

This is done by multiplying by $\frac{N}{1+N\frac{1}{s}}$ instead of simply multiplying by s (in the s domain) where N[6] is the filter coefficient.

The PID controller uses the 3 previously mentioned controllers to give its output by summing them all together.

The PID controller is tuned by providing certain gains (Kp, Ki, Kd) that allow the system to react according to certain constraints.

Hence the *PID controller* has the following open loop transfer function:

$$Kp + Ki\frac{1}{s} + kd\frac{N}{1+N\frac{1}{s}}$$

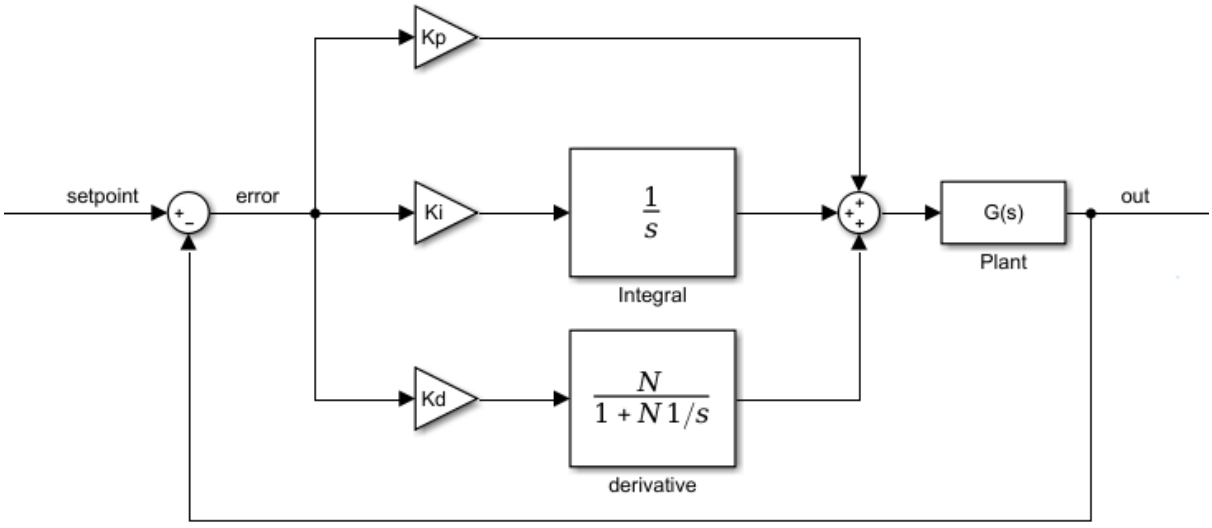

Figure 2 PID Controller Block diagram

A block diagram of a generic *PID feedback controller* is has the following form (**Figure 4**)

[5] Relatively high with respect to the input function
[6] N is the coefficient used to specify the critical frequency low pass filter incorporated in the derivative controller

Full State Feedback

Introduction

FSF controller provides control to a dynamic system described in *state-space*[7].

A state-space of a system has the form:

$$\begin{cases} \dot{X} = AX + Bu \\ Y = CX + Du \end{cases}$$

Where A defines the dynamics of the system, B defines the influence of the input on the system, C defines the output of the system contributed by its state and D defines the contribution of the input to the output of the system.

LQR[8] provides a solution such system given a cost described by a quadratic function. LQR is based on *optimal control*[9].

Theory

The controller behavior is altered by using a mathematical algorithm [10] that minimizes a cost function with weighting factors provided by a human. The cost function is often defined as a sum of deviation of some variables from their desired values. The algorithm generates setting that minimizes undesired deviations.

By utilizing LQR, one can get a feedback gain K.

A Full State Feedback controller block diagram can be shown in **Figure 5**.

[7] Set of linear differential Equations
[8] Linear-Quadratic Regulator
[9] The theory of optimal control is concerned with operation a dynamic system at minimum cost
[10] Appendix

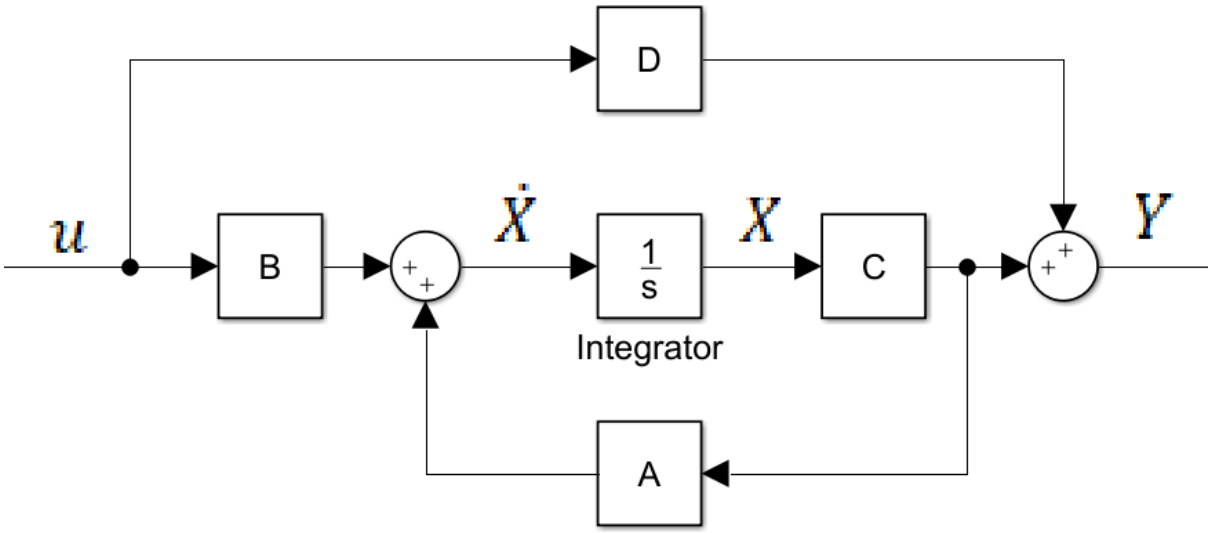

Figure 3 State-Space System Block Diagram

By feeding back the system state (X) multiplied by the gain K calculated by the LQR Algorithm gives the block diagram described in **Figure 6.**

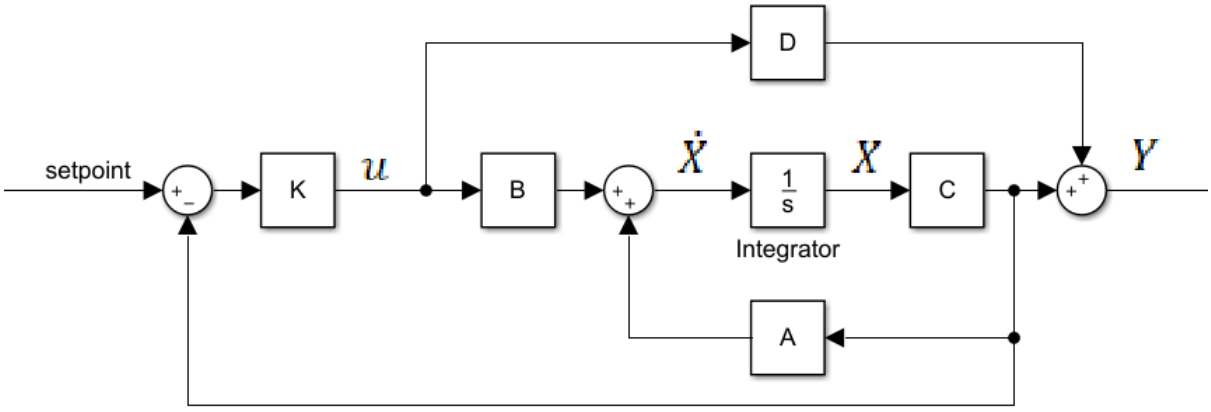

Figure 4 Full State Feedback with LQR

CHAPTER 2: MODELING ROTARY INVERTED PENDULUM

Introduction

A rotary inverted pendulum (as shown in **Figure 7**) is system consisting of a motor (coupled to a *shaft encoder*[11]) rotating an arm.

The rotating arm is connected to another shaft encoder having its shaft parallel to the arm.

The second encoder now acts as a pivot point of a pendulum connected to it.

The pendulum is either a homogeneous rod or a light rod connected to an end block.

Figure 5 Quanser Rotary Inverted Pendulum V1.0

[11] A shaft encoder is a sensor that can be used the angular position and speed of its rotating shaft

State Space Model of the System

Given the below simplified schematic of an inverse pendulum in **Figure 8**.

α denotes the angle the pendulum. Hence $\dot{\alpha}$ is its angular speed.

θ denotes the angle of the motor shaft. Hence $\dot{\theta}$ is its angular speed.

The system described as a state-space can be shown to be:

$$\begin{cases} \dot{X} = AX + Bu \\ Y = CX + Du \end{cases} \text{ Where:}$$

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{M_p^2 \cdot l_p^2 \cdot r \cdot g}{J_{eq} \cdot M_p \cdot l_p^2 + M_p \cdot r^2 \cdot J_p + J_{eq} \cdot J_p} & -\frac{(J_p . K_t . K_m + M_p . l_p^2 . K_t . K_m)}{R_m . (J_{eq} . J_p + J_{eq} . M_p . l_p^2 + M_p . r^2 . J_p)} & -B_{eq} \\ 0 & \frac{M_p . l_p . g (J_{eq} + M_p . r^2)}{J_{eq} \cdot M_p \cdot l_p^2 + M_p \cdot r^2 \cdot J_p + J_{eq} \cdot J_p} & -\frac{M_p . l_p . r . K_t . K_m}{R_m . (J_{eq} . J_p + J_{eq} . M_p . l_p^2 + M_p . r^2 . J_{p)}} & -B_p \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ -\frac{K_t . (J_p + M_p . l_p^2)}{R_m . (J_{eq} . J_p + J_{eq} . M_p . l_p^2 + M_p . r^2 . J_{p)}} \\ -\frac{M_p . l_p . r . K_t}{R_m . (J_{eq} . J_p + J_{eq} . M_p . l_p^2 + M_p . r^2 . J_{p)}} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

provided by the manufacturer of the *Quanser Rotary Inverted Pendulum V1.0*[12]

Where the parameters mentioned are shown in **Figure 8**

[12] Appendix

Parameter	Description	Unit	Value
g	Gravitational constant	N/kg	9.81
M_p	Mass pendulum assembly	kg	0.027
l_p	Center of mass of pendulum assembly	m	0.153
r	Length from motor shaft to pendulum pivot	m	0.0826
J_p	Pendulum moment of inertia relative to pivot	$kg.m^2$	0.000698
J_{eq}	Equivalent moment of inertia acting on motor shaft	$kg.m^2$	0.000368
B_p	Viscous Damping about the pendulum pivot	N.m.s/rad	1.4
B_{eq}	Equivalent viscous damping acting on motor shaft	N.m.s/rad	0.93
K_t	DC motor current-torque constant	N.m/A	0.0333
K_m	DC motor back-emf constant	V.s/rad	0.0333
R_m	Electric resistance of motor armature	Ω	8.7

Figure 6 Description and values of parameters

CHAPTER 3: CONTROLLING THE PENDULUM

A simulation of the pendulum was done over MATLAB and Simulink using the provided model to test the different controllers.

Manipulating an inverted pendulum utilizes three different controllers that switch automatically. The swing up controller, catch controller and balance controller.

The job of swing up controller is to swing the pendulum from its resting down state to an up state.

The job of the catch controller is to catch the flung-up pendulum by reducing the angular velocity of the pendulum.

The job of the balance controller is to keep the pendulum in upright position once it is up.

The scope of this project focuses only on the balance controller where the pendulum is assumed to not deviate far from the upright position.

The initial conditions on the simulation are:

$\theta = 0$ $\alpha = 0.2$ $\dot{\theta} = 0$ $\dot{\alpha} = 0$[13]

Control using Fuzzy Logic

The FLC was made with one input and one output, the input being α.

The block diagram of the FLC is as shown in **Figure 9**

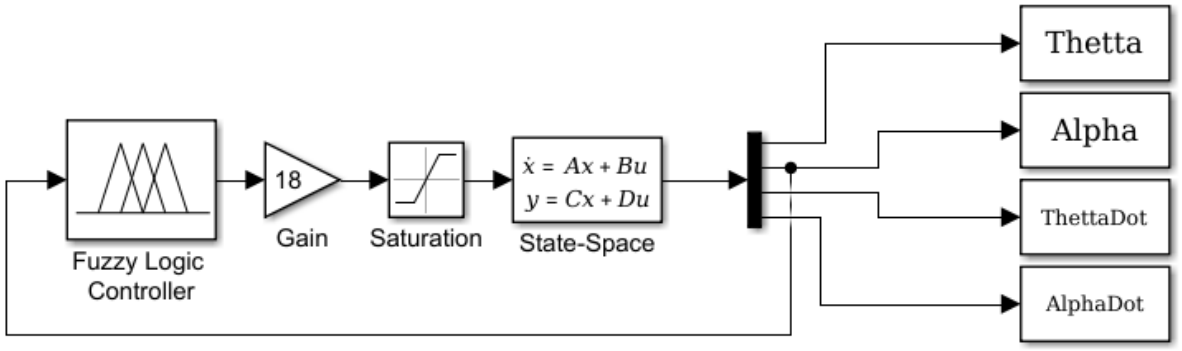

Figure 7 FLC block diagram

The membership functions of the input are two trapezoidal functions and a triangular one (**Figure 10)** and those of the output are 3 triangular functions (**Figure 11**)

[13] Pendulum is slightly tipped with the motor shaft on zero and all angular speeds are zero

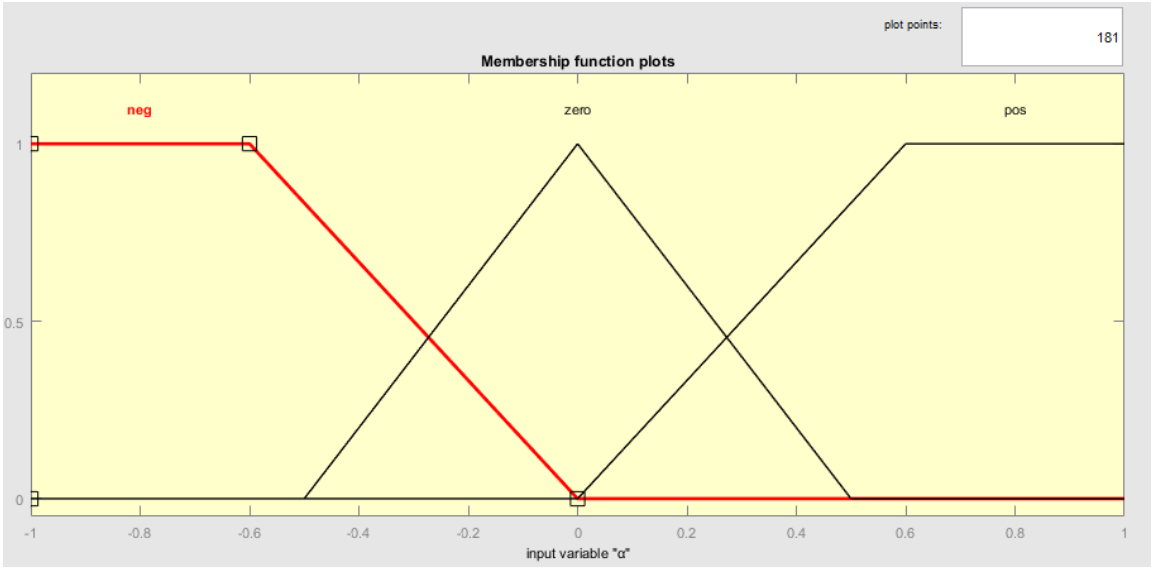

Figure 8 FLC input membership functions

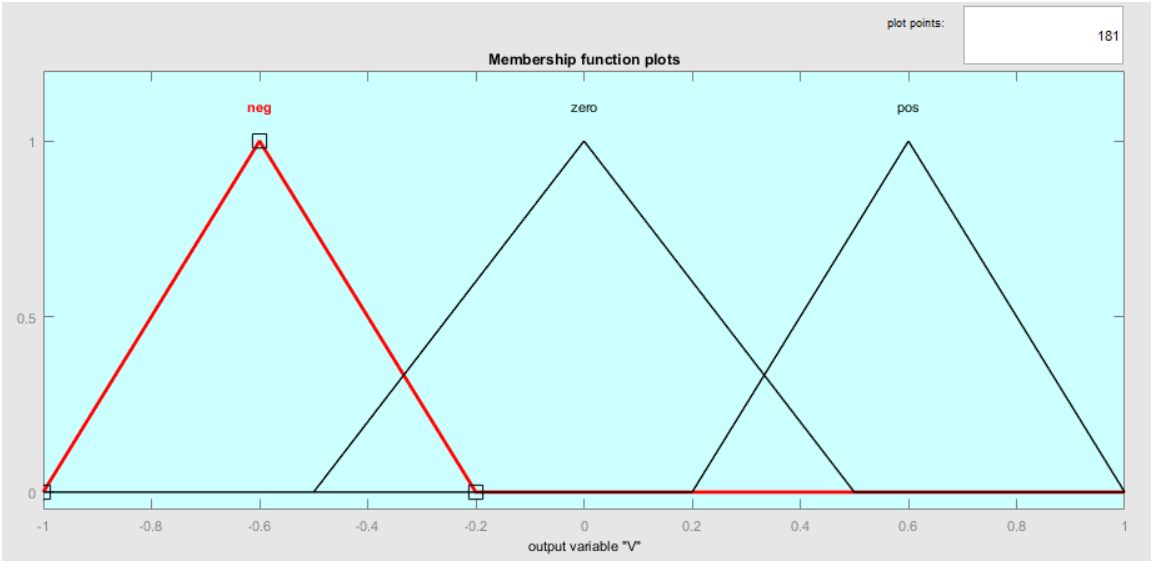

Figure 9 FLC output membership functions

The rules set for the FLC are as follows:

If input is pos, then output is pos

If input is zero, then output is zero

If input is neg, then output is neg

The output is defuzzied using centroid method

The result of simulating the controller after several attempts of tweaking are as shown in **Figures 12-15**

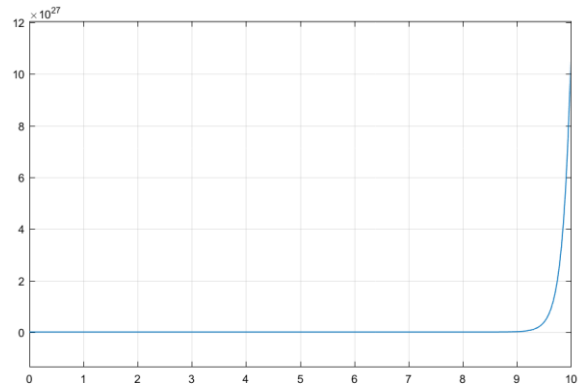

Figure 12 θ result of FLC

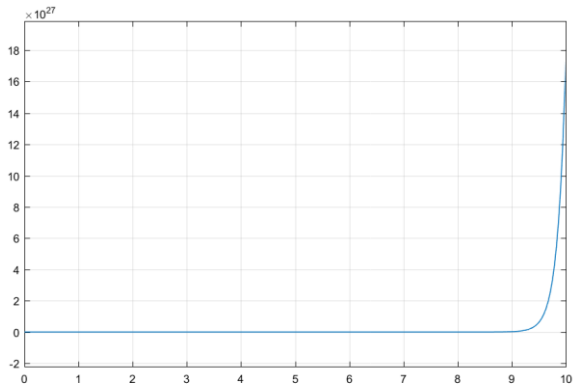

Figure 13 α result of FLC

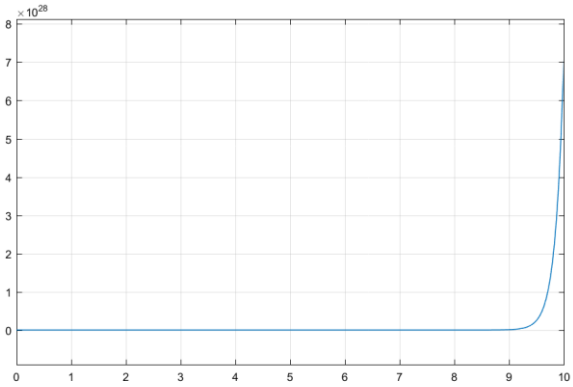

Figure 10 $\dot{\theta}$ result of FLC

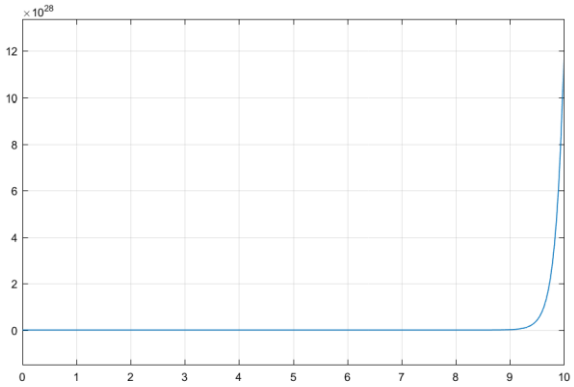

Figure 11 $\dot{\alpha}$ result of FLC

Control using PID

A PID controller is also single input single output controller.

The setup was done as shown in the block diagram of **Figure 16**

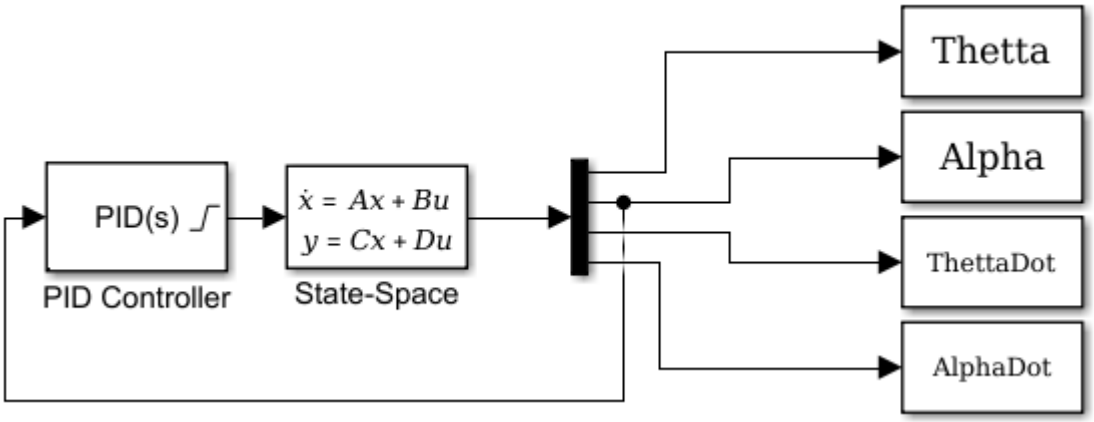

Figure 14 Block diagram of PID

The block diagram shows the PID controller using α as an input and tries to stabilize it at zero.

The results of simulating the above system for 10s after the PID controller was tuned are shown in the **Figures 17-20**

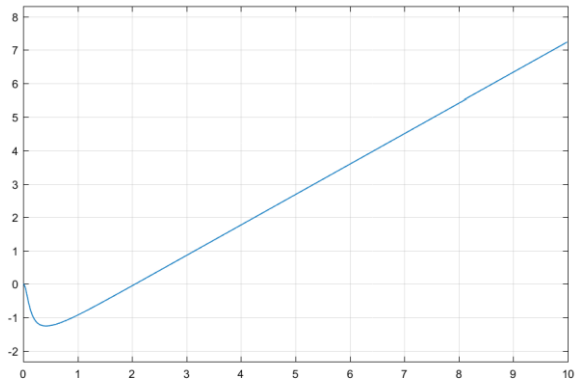

Figure 18 θ result of PID

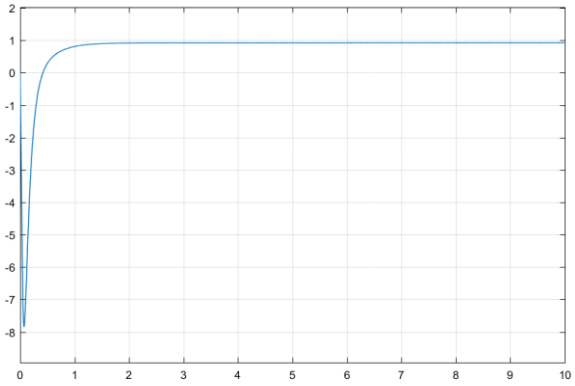

Figure 16 α result of PID

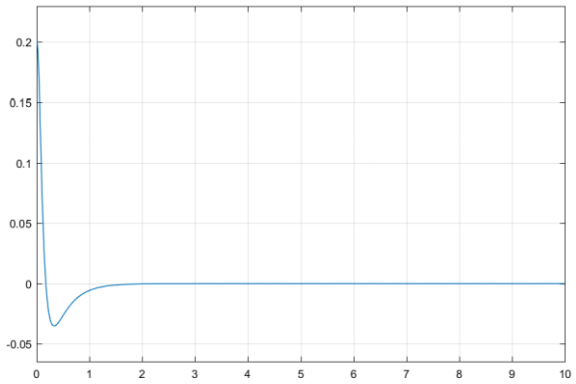

Figure 17 $\dot{\theta}$ result of PID

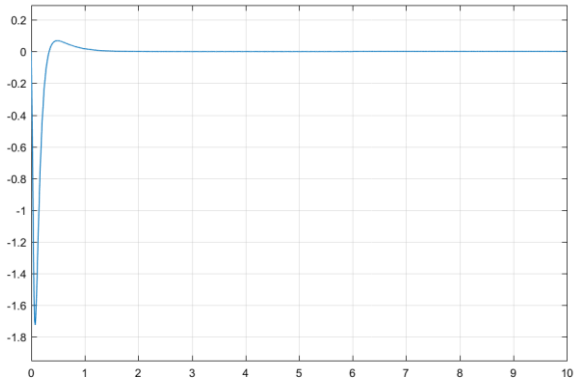

Figure 15 $\dot{\alpha}$ result of PID

Though the PID controller was able to stabilize α at zero, θ̇ is nonzero and hence θ is constantly changing even after steadying α.

Though we can't fix θ at zero easily without sacrificing the stability of our existing controller, we can however stop it from moving by eliminating the steady state error of θ̇ to zero.

To fix this problem, an I[14] controller is added that uses θ̇ as input and generates an output that is summed with the existing PID controller.

[14] A PID controller with Kp and Kd equal zero hence only the integrator part is functioning with Ki set to 5

Thus, the block diagram becomes as show in **Figure 21**

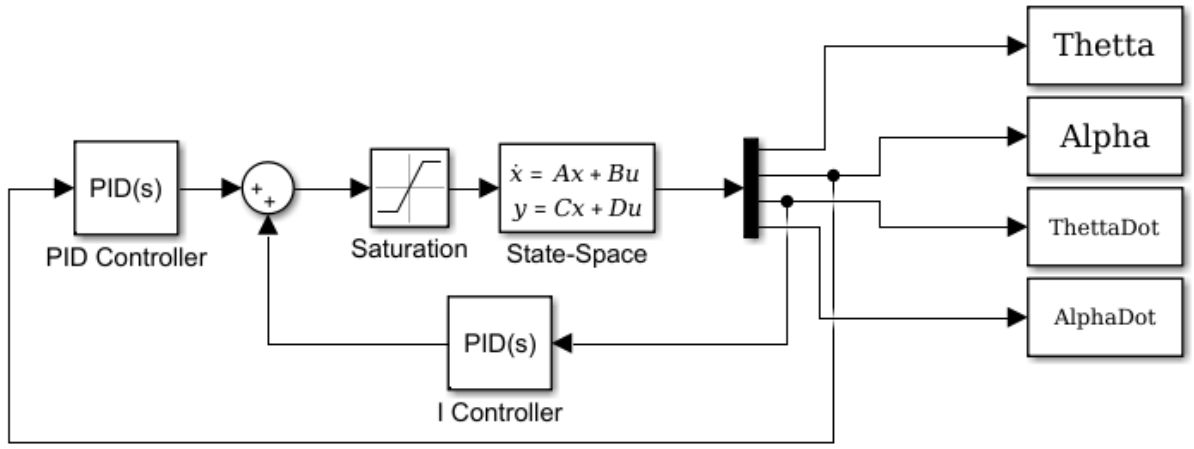

Figure 19 Block diagram of Improved PID

The result of simulating the above system gives the results in **Figures 22-25**

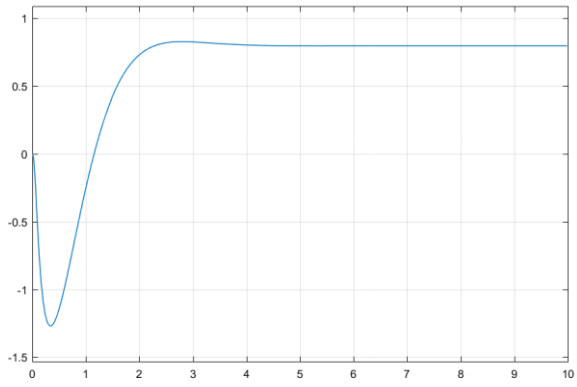

Figure 21 θ result of improved PID

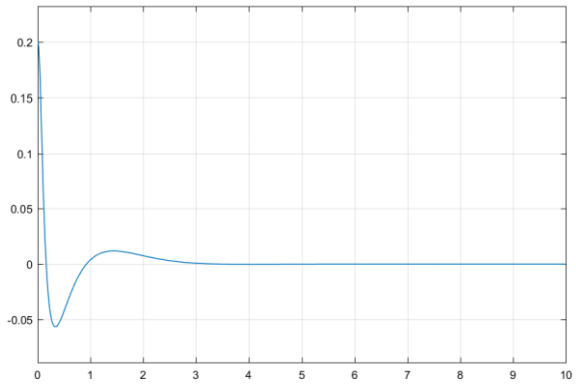

Figure 22 α result of improved PID

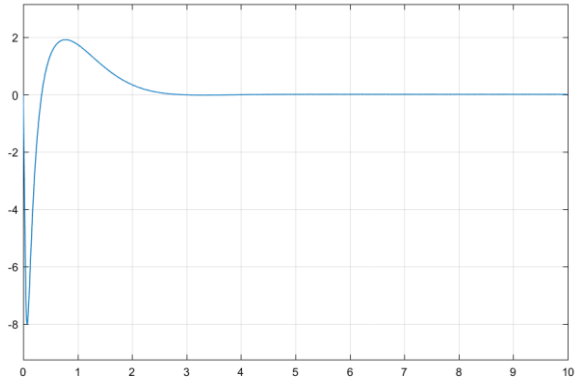

Figure 20 $\dot{\theta}$ result of improved PID

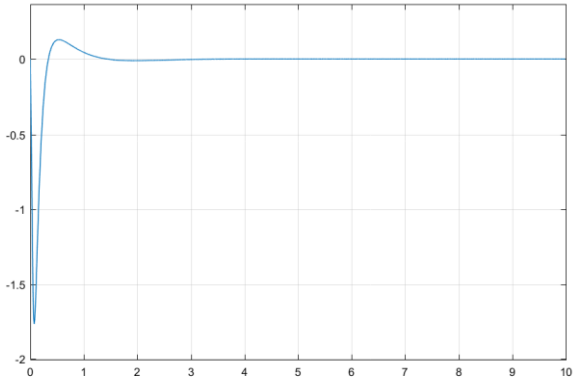

Figure 23 $\dot{\alpha}$ result of improved PID

Control using Full State Feedback

In our FSF controller, we used LQR to provide solution to the optimal control algorithm by calculating K from the matrix Q and the scalar R.

The system was tuned by manipulating the values of Q and R by trial and error using Q = I[15] and R=1 as base values[16].

The results used were: $Q = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ $R = 1$

The block diagram of the controller is shown in **Figure 26.**

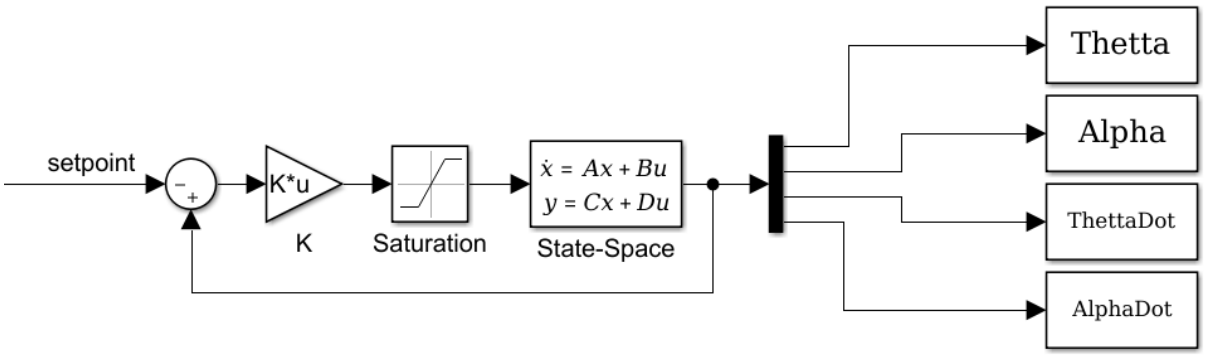

Figure 24 Block diagram of FSF using LQR

The block diagram is the same as shown in Figure 6, where the only difference it the presence of a saturation block to prevent the controller from bypassing the rated voltage of the motor.

By setting the setpoint to zero, the simulation is run and the results are shown in **Figures 27-30**

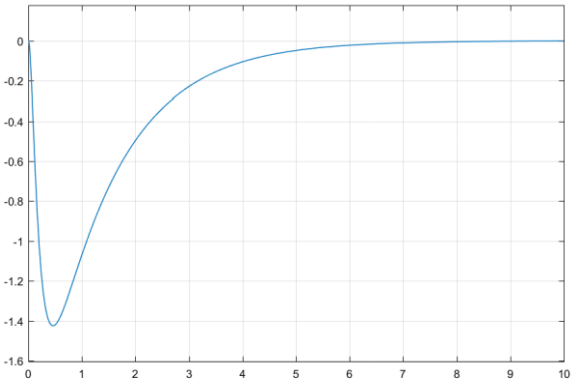

Figure 25 *θ result* of FSF

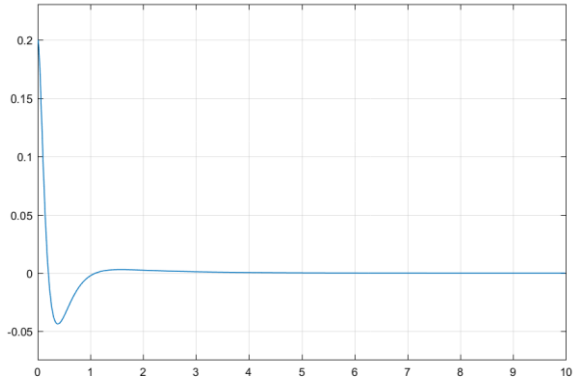

Figure 26 *α* result of FSF

[15] 4x4 Identity Matrix
[16] The MATLAB function lqr was used to obtain K from A, B, Q and R

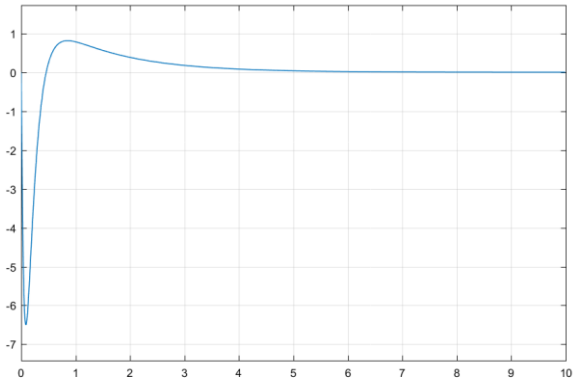

Figure 27 $\dot{\theta}$ result of FSF

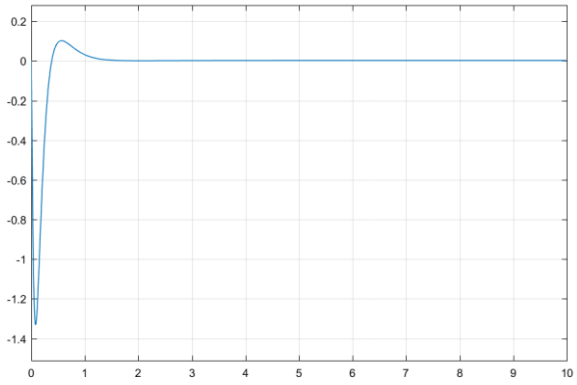

Figure 28 $\dot{\alpha}$ result of FSF

Results

The FLC did not perform well even after several attempts of modifying its behavior. In all the results that came out, the system showed no reaction at all until the pendulum falls and the system crumbles into instability.

The PID controller was able to precisely balance the pendulum and keep it upright. However, without introducing the second PID controller the system continues spinning even after successfully balancing the pendulum. Another PID controller has been added to eliminate spinning but we still weren't able to control θ.

The FSF controller performed well. It was able to keep the pendulum upright and the arm at position zero without compromising rise times and overshoot.

CONCLUSION

Summary

The full state feedback controller performed the best giving excellent reaction time and accuracy. Following it the PID controller which did not perform as well but was able to keep the pendulum upright. Finally, the fuzzy logic controller did the worse being not able to stabilize the pendulum at all even after several attempt proving the almost impossibility to perform precise task using FLC.

Fuzzy logic controller is suited best for mechanizing operations that could successfully be done by a human operator.

PID proved its ability to handle (although a bit limited) complex control systems given its simple structure

FSF demonstrated its ability to precisely and flawlessly control complex systems and thus it is recommended to utilize such controller for even more complex systems.

Perspective

Possible advancements on the topic include:

- Using a more advanced PID based controller to fully control the pendulum
- Implementing a swing up and a catch controller
    - Might include using different controllers
    - Dealing with the nonlinear aspect of the system
    - Combine different controllers and a allow the system to switch controllers to be able to perform a swift lift up and balance
- Implementation on hardware

APPENDIX:

Membership Functions

The input variables in a fuzzy control system are in general mapped by sets of membership functions known as "fuzzy sets". The process of converting a crisp input value to a fuzzy value is called "fuzzification"

Membership functions allow us to graphically represent a fuzzy set.

Simple functions are used to build membership functions. Because when defining fuzzy concepts, using more complex functions does not add more precision.

Examples of membership functions: Triangular, Trapezoidal, Gaussian…

Example on Inputs and Output of Fuzzy Logic Controller:

Figure 2 shows the membership functions of two input variables (Temperature and Pressure) and one output (Throttle setting) for a steam turbine.

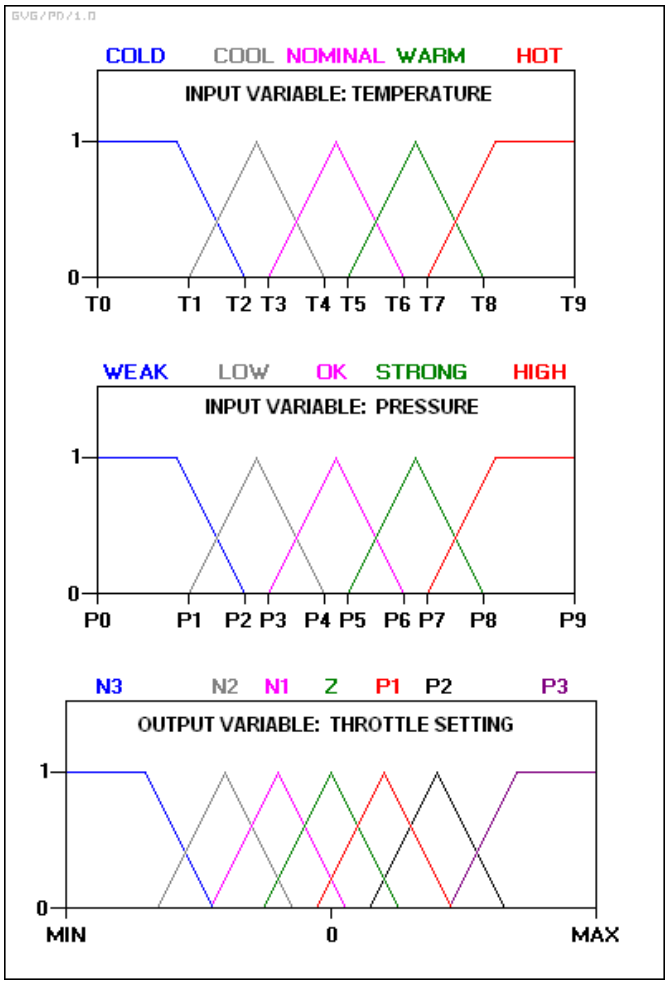

Figure 29: Fuzzy controller design for a steam turbine

Symbolic Variables:

```
N3:   Large negative.
N2:   Medium negative.
N1:   Small negative.
Z:    Zero.
P1:   Small positive.
P2:   Medium positive.
P3:   Large positive.
```

Rules:

```
rule 1:  IF temperature    IS
cool AND pressure IS weak, THEN
throttle is P3.

rule 2:  IF temperature IS cool
AND pressure IS low, THEN
throttle is P2.

rule 3:  IF temperature IS cool
AND pressure IS ok, THEN
throttle is Z.

rule 4:  IF temperature IS cool
AND pressure IS strong, THEN
throttle is N2.
```

Note that the AND and OR between inputs can have different predefined meaning like min, max, average… Here max-min inferencing[17] is used

Assume the temperature is in the "cool" state, and the pressure is in the "low" and "ok" states. The pressure values ensure that only rules 2 and 3 fire as shown in **Figure 3**

The multiple rules that fire could then be defuzzied using several methods (the centroid[18] method is very popular and is widely used and will be used in this example)

The defuzzied result is hence the output of the controller.

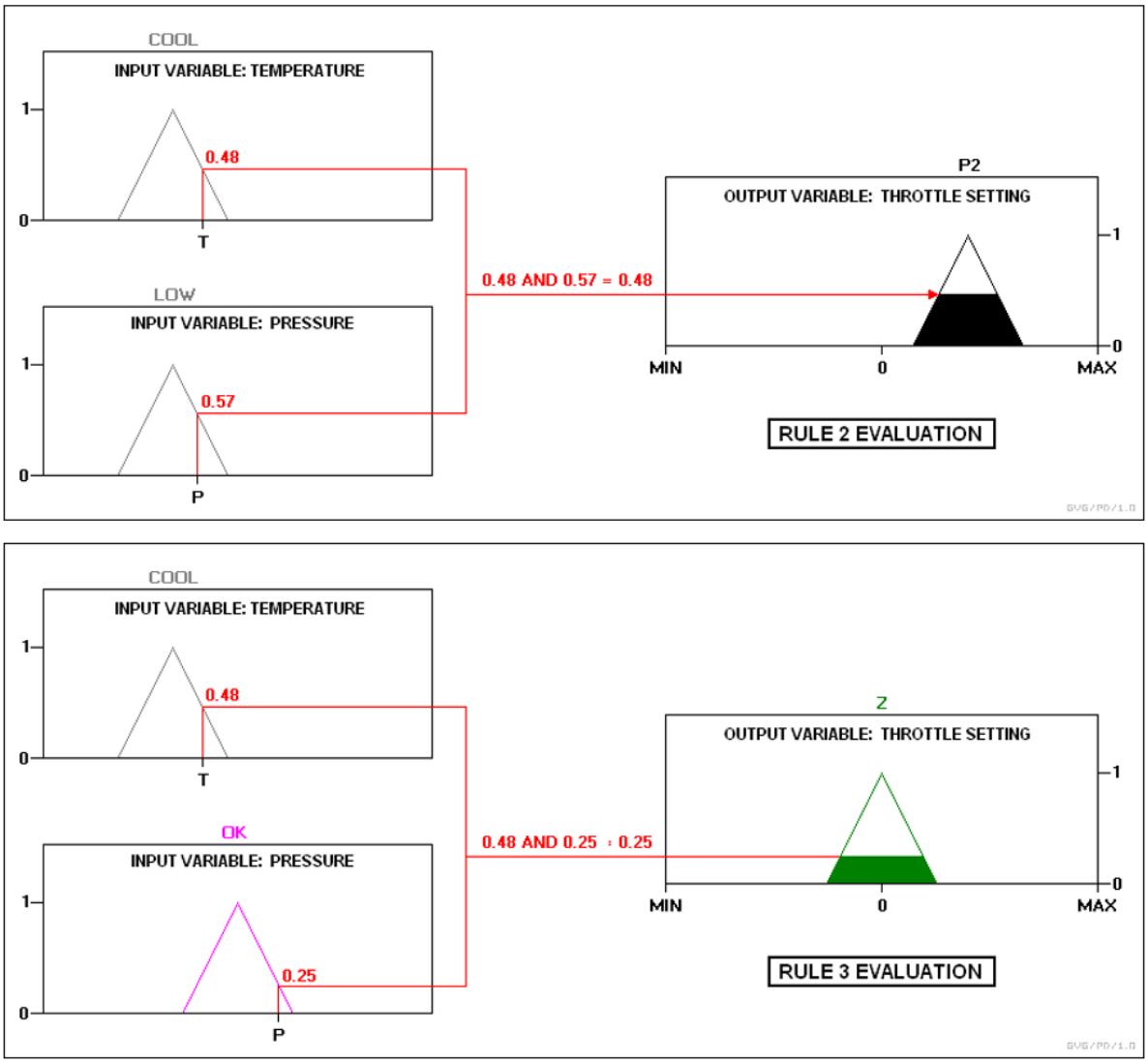

Figure 30: Rule 2 and Rule 3 Evaluation

[17] Minimum as logical AND, the Maximum as logical OR
[18] The centroid method gives an output which is the center of mass of the multiple obtained values

LQR

If the reference is assumed to be zero $\bar{r} = 0$, the input to a full-state feedback system is $\bar{u} = -K\bar{x}$ and the state equation becomes

$$\dot{\bar{x}} = A\bar{x} + B\bar{u} = (A - BK)\bar{x} = A_f\bar{x}$$

where A_f=*A*-*BK*. The goal of the LQR control strategy is to have the state variables to approach zero $\bar{x} \to 0$ from some initial condition $\bar{x}(0)$ in minimum amount of time and energy of the control $\bar{u}$, i.e., we want to minimize a performance index defined as:

$$J = \int_0^\infty [\bar{x}^T Q\bar{x} + \bar{u}^T R\bar{u}]\, dt = \int_0^\infty \bar{x}^T (Q + K^T RK)\bar{x}\, dt$$

where *Q* (n by n) and *R* (r by r) are, respectively, the state weighting matrix and control weighting matrix, both of which are symmetric (Q^T=*Q*, R^T=*R*) and positive definite ($\bar{z}^T Q\bar{z} > 0$, $\bar{z}^T R\bar{z} > 0$, where $\bar{z}$ is an arbitrary vector).

Now assume we can find a symmetric matrix S=S^T so that:

$$\bar{x}^T(Q+K^T RK)\bar{x} = -\frac{d}{dt}(\bar{x}^T S\bar{x}) = -\dot{\bar{x}}^T S\bar{x}+\bar{x}^T S\bar{x} = (A_f\bar{x})^T Sx+\bar{x}^T S(A_f\bar{x}) = -\bar{x}^T(A_f^T S+SA_f)\bar{x}$$

i.e.,

$$Q + K^T RK = -A_f^T S - SA_f, \quad \text{or} \quad Q + K^T RK + A_f^T S + SA_f = 0$$

Replacing A_f by *A*-*BK*, we get:

$$Q + K^T RK + (A^T - B^T K^T)S + S(A - BK) = 0 \qquad (*)$$

Assume $\hat{K}$ to be the optimal *K*, then $\hat{K} + \delta K$ will be suboptimal (δK is a small deviation). Substituting both $\hat{K}$ and $\hat{K} + \delta K$ into the equation (*) and subtracting one from the other, we get:

$$(\delta K)^T(R\hat{K} - B^T S) + (\hat{K}^T R - SB)(\delta K) = 0$$

Note that a second order term $(\delta K)^T R(\delta K)$ in the expression was neglected as δK is assumed small. For the above equation to hold for any δK, we must have:

$$R\hat{K} = B^T S, \quad \text{i.e.,} \quad \hat{K} = R^{-1}B^T S$$

Substituting this solution *K* back into equation (*), we get

$A^TS+SA-SBR^{-1}B^TS+Q=0$

This is called the matrix Ricatti equation, which can be solved for matrix *S*. Now the performance index becomes:

$$J = \int_0^\infty -\frac{d}{dt}(\bar{x}^T S\bar{x})dt = -\bar{x}^T S\bar{x}|_0^\infty = \bar{x}^T(0)S\bar{x}(0)$$

Deriving State-Space from Equations of Motion

The pendulum link is connected to the end of the rotary arm. It has a total length of Lp and it center of mass is Lp/2. The moment of inertia about its center of mass is Jp. The inverted pendulum angle, α, is zero when it is perfectly upright in the vertical position and increases positively when rotated CCW. Instead of using classical mechanics, the Lagrange method is used to find the equations of motion of the system. The Euler- Lagrange equation is given as:

$$\frac{\partial^2 L}{\partial t \partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = Q_1$$
$$\frac{\partial^2 L}{\partial t \partial \dot{\alpha}} - \frac{\partial L}{\partial \alpha} = Q_2 \qquad (1)$$

The Lagrangian of the system is described as:

$$L=T-V \qquad (2)$$

Where, T is the total kinetic energy of the system and V is the total potential energy of the system.

The generalized forces Qi are used to describe the nonconservative forces (e.g., friction) applied to a system with respect to the generalized coordinates. In this case, the generalized force acting on the rotary arm and pendulum is:

$$Q_1 = \tau - B_r \dot{\theta}$$
$$Q_2 = -B_p \dot{\alpha} \qquad (3)$$

acting on the link is the damping. The viscous damping coefficient of the pendulum is denoted by Bp. The nonlinear equations of motion for the rotary inverted pendulum are:

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

$$-\frac{1}{2}m_p L_p L_r\cos(\alpha)\ddot{\theta}+(J_p+\frac{1}{4}m_p L_p^{\,2})\ddot{\alpha}-\frac{1}{4}m_p L_p^{\,2}\cos(\alpha)\sin(\alpha)\dot{\theta}^{\,2}$$
$$-\frac{1}{2}m_p L_p g\sin(\alpha)=-B_p\dot{\alpha} \quad (5)$$

The torque applied at the base of the rotary arm (i.e., at the load gear) is generated by the servo motor as described by the equation:

$$\tau=\frac{\eta_m\eta_g K_t K_g\left(V_m-K_g K_m\theta\right)}{R_m} \quad (6)$$

The nonlinear equations can be linearized using the below method:

$$f_{lin}=f(z_0)+(\frac{\partial f(z)}{\partial z_1})|_{z=z_0}(z_1-a)+(\frac{\partial f(z)}{\partial z_2})|_{z=z_0}(z_2-b) \quad (7)$$

With all initial state variables as zero, the linearized equation can be obtained as:

$$\left(m_p L_r^{\,2}+J_r\right)\ddot{\theta}-\tfrac{1}{2}m_p L_p L_r\ddot{\alpha}=\tau-B_r\dot{\theta} \quad (8)$$

$$-\frac{1}{2}m_p L_p L_r\ddot{\theta}+\left(J_p+\frac{1}{4}m_p L_p^{\,2}\right)\ddot{\alpha}-\left(\frac{1}{2}m_p L_p g\alpha\right)=-B_p\dot{\alpha} \quad (9)$$

MATLAB CODE:

The following MATLAB code was used to simulate the behavior of the pendulum in conjunction with Simulink:

```
clc;
close all;
clear;

g= 9.81;          %gravity constant
Mp = 0.027;       %mass of pendulum assembly
lp = 0.153;       %center of mass of pendulum assembly
r=  0.0826;       %length from motor shaft to pendulum pivot
Jp= 0.000698;     % pendulum moment of inertia relative to pivot
Jeq= 0.000368;    %equivalent moment of inertia acting on the DC motor shaft
```

```
Bp= 1.4;           %Viscous damping about the pendulum pivot
Beq= 0.93;          %equivalent viscous damping acting on the DC motor shaft
Kt= 0.0333;      %DC motor current-torque constant
Km= 0.0333;      %DC motor back-emf constant
Rm= 8.7;         %Electric resistance of the DC motor armature

IC=[0 0.2 0 0];

A= [[0 , 0 , 1 , 0];
    [0 , 0 , 0 , 1];
    [0 , Mp^2*lp^2*r*g/(Jeq*Jp+Jeq*Mp*lp^2+Mp*r^2*Jp) , -
(Jp*Kt*Km+Mp*lp^2*Kt*Km)/Rm/(Jeq*Jp+Jeq*Mp*lp^2+Mp*r^2*Jp) , -Beq];
    [0 , Mp*lp*g*(Jeq+Mp*r^2)/(Jeq*Jp+Jeq*Mp*lp^2+Mp*r^2*Jp) , -
Mp*lp*r*(Kt*Km)/Rm/(Jeq*Jp+Jeq*Mp*lp^2+Mp*r^2*Jp) , -Bp] ];

B=  [0 ; 0 ; Kt*(Jp+Mp*lp^2)/Rm/(Jeq*Jp+Jeq*Mp*lp^2+Mp*r^2*Jp) ;
Mp*lp*Kt*r/Rm/(Jeq*Jp+Jeq*Mp*lp^2+Mp*r^2*Jp)];

C= [[1 0 0 0];
    [0 1 0 0];
    [0 0 1 0];
    [0 0 0 1]];

D= [0 ; 0 ; 0 ; 0];

Q= [[3 0 0 0]
    [0 2 0 0]
    [0 0 5 0]
    [0 0 0 0]];
R=1;

[K,S,e]=lqr(A,B,Q,R);

% sim('Rot_Pen_LQR.slx',10);
% sim('Rot_Pen_PID.slx',10);
%sim('Rot_Pen_FLC.slx'x,10);
```

The following code was used to plot a graph to visualize the behavior of the pendulum:

```
close all;
axis([-0.1 0.1 -0.1 0.1 0 0.4]);



for t= 1:1000
    alpha=Alpha(t);
    thetta=Thetta(t);
    rod=line([0 , r*cos(thetta)],[0 , r*sin(thetta)], [0 , 0] ,
'linewidth',1);
    stand=line([r*cos(thetta) , r*cos(thetta)] ,
[r*sin(thetta),r*sin(thetta)],[0 , 0.13]);
    pivot=line([r*cos(thetta), (r+0.01)*cos(thetta)], [r*sin(thetta),
(r+0.01)*sin(thetta)] , [0.13 0.13]);
```

```
    pendulum=line([(r+0.01)*cos(thetta),
(r+0.01)*cos(thetta)+lp*sin(alpha)*sin(thetta)],[(r+0.01)*sin(thetta),(r+0.
01)*sin(thetta)+lp*sin(alpha)*cos(thetta)], [0.13 ,
0.13+lp*cos(alpha)],'linewidth',3);


    pause(0.007);
    delete(rod);
    delete(stand);
    delete(pivot);
    delete(pendulum);
end
```