University Of Balamand

Faculty of Engineering

Fuzzy Logic Control

Inverted Pendulum Modeling

Presented to: Dr. Issam Dagher

Presented by: Kifah DAHER

Date: 31-10-2012

Introduction:

This report introduces an inverted pendulum example and a typical procedure used to design and realize of a fuzzy controller. To simulate a fuzzy control system it is necessary to specify a mathematical model of the inverted pendulum. Using MATLAB a code representing the mathematical model for the pendulum is integrated, membership functions are realized. The mathematical model is represented by a differential equation of second order which require using the ode23 command in Matlab in order to solve.

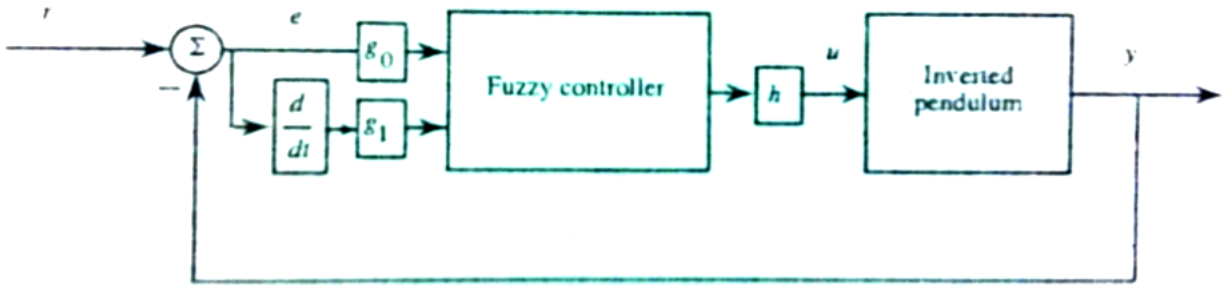
One model for the inverted pendulum is given by:

$$\ddot{y} = \frac{9.8\sin(y) + \cos(y)\left[\frac{-\bar{u} - 0.25\dot{y}^2\sin(y)}{1.5}\right]}{0.5\left[\frac{4}{3} - \frac{1}{3}cos^2(y)\right]}$$

$$\dot{\bar{u}} = -100\bar{u} + 100u$$

The first order filter on u to produce $\bar{u}$ represents an actuator. we let the initial condition be y(0) = 0.1 radians (=5.73 deg), $\dot{y}(0) = 0$ and the initial condition for the actuator state is zero. Gains are introduced on proportional and derivative terms; a gain h between the fuzzy controller and the inverted pendulum.

The fuzzy controller for the inverted pendulum with scaling gains g0, g1 and h is the following:



The gains let the output angle reacts much faster and the control input smoother. If we still find the response of the pendulum slow, the proportional gain should be increased which can speed up the system. As well by increasing the output gain h faster response is obtained as well a faster balancing is provided as we can conclude from the successive simulation results.

First of all the membership function of the system are implemented in Matlab, where the error (e) and the change of error (de/dt) are the inputs whereas u is the output of the fuzzy controller. Considering the gains are unity the membership function are the following:

The two input membership functions are of triangular type which involve the uses of "trimf" command.

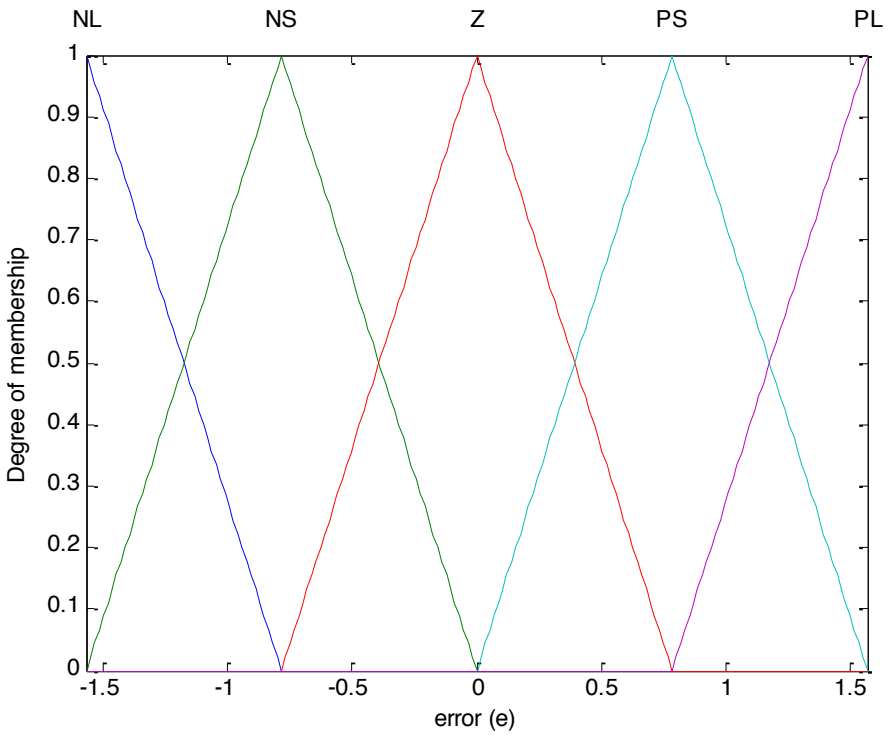


Figure 1: Membership function of input1: error (e)

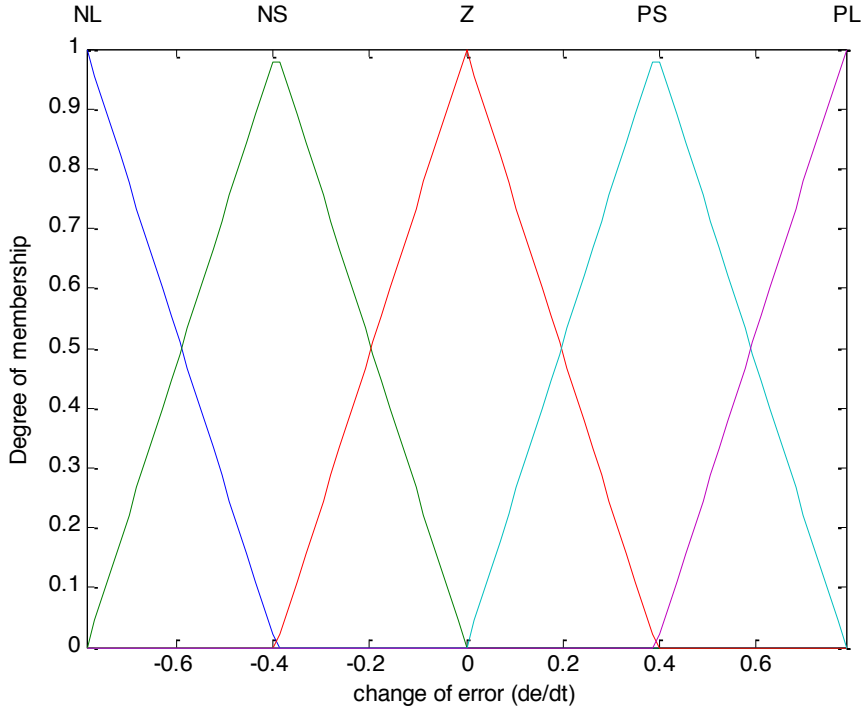


Figure 2: Membership Function of input 2: change of error (de/dt)

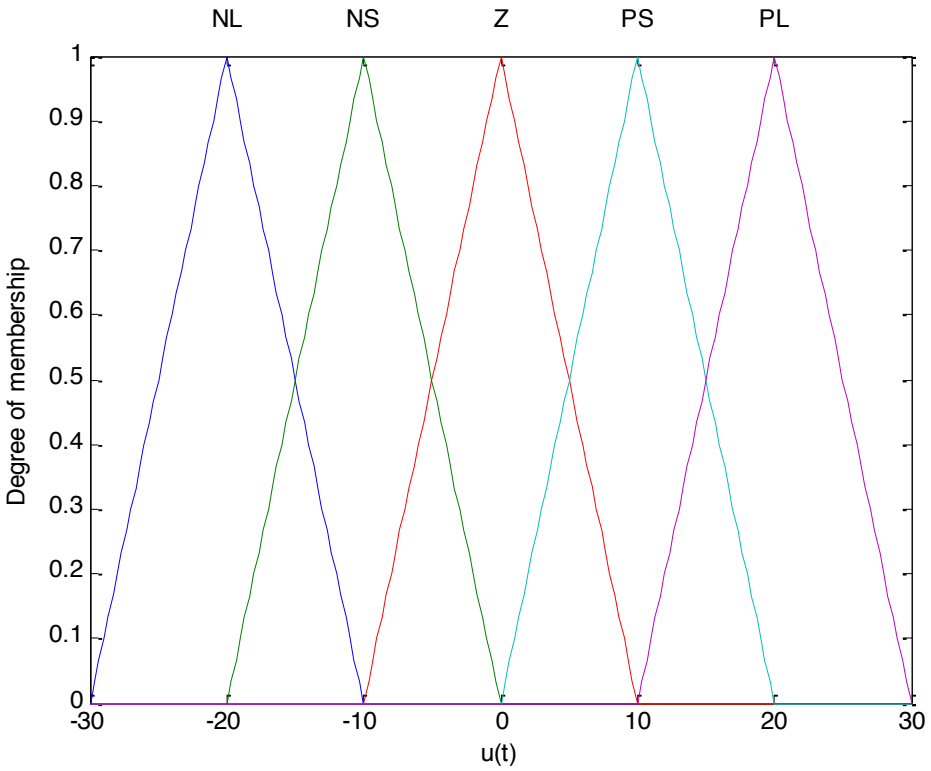


Figure 3: Membership Function of the output

The membership functions are summarized such that:

- NL: Negative Large
- NS: Negative Small
- Z: Zero
- PS: Positive Small
- PL: Positive Large

They describe the angular position of the inverted pendulum as well the response of the output which is built based on the following rules:

de/dt / e	NL	NS	Z	PS	PL
NL	PL	PL	PL	PS	Z
NS	PL	PL	PS	Z	NS
Z	PL	PS	Z	NS	NL
PS	PS	Z	NS	NL	NL
PL	Z	NS	NL	NL	NL

The Matlab Code concerning the Membership Function construction is the following:

```
a=newfis('Inverted Pendalum');
a=addvar(a,'input','error (e)',[-pi/2 pi/2]);
a=addmf(a,'input',1,'NL','trimf', [-inf -pi/2 -pi/4]);
a=addmf(a,'input',1,'NS','trimf', [-pi/2 -pi/4 0]);
a=addmf(a,'input',1,'Z','trimf', [-pi/4 0 pi/4]);
a=addmf(a,'input',1,'PS','trimf', [0 pi/4 pi/2]);
a=addmf(a,'input',1,'PL','trimf', [pi/4 pi/2 +inf]);
a=addvar(a,'input','change of error (de/dt)',[-pi/2 pi/2]);
a=addmf(a,'input',2,'NL','trimf', [-inf -pi/4 -pi/8]);
a=addmf(a,'input',2,'NS','trimf', [-pi/4 -pi/8 0]);
a=addmf(a,'input',2,'Z','trimf', [-pi/8 0 pi/8]);
a=addmf(a,'input',2,'PS','trimf', [0 pi/8 pi/4]);
a=addmf(a,'input',2,'PL','trimf', [pi/8 pi/4 +inf]);
a=addvar(a,'output','u(t)', [-30 30]);
a=addmf(a,'output',1,'NL','trimf', [-30 -20 -10]);
a=addmf(a,'output',1,'NS','trimf', [-20 -10 0]);
a=addmf(a,'output',1,'Z','trimf', [-10 0 10]);
a=addmf(a,'output',1,'PS','trimf', [0 10 20]);
a=addmf(a,'output',1,'PL','trimf', [10 20 30]);
rulelist=[1 1 5 1 1;
          1 2 5 1 1;
          1 3 5 1 1;
          1 4 4 1 1;
          1 5 3 1 1;
          2 1 5 1 1;
          2 2 5 1 1;
          2 3 4 1 1;
          2 4 3 1 1;
          2 5 2 1 1;
          3 1 5 1 1;
          3 2 4 1 1;
          3 3 3 1 1;
          3 4 2 1 1;
          3 5 1 1 1;
          4 1 4 1 1;
          4 2 3 1 1;
```

```
            4 3 2 1 1;
            4 4 1 1 1;
            4 5 1 1 1;
            5 1 3 1 1;
            5 2 2 1 1;
            5 3 1 1 1;
            5 4 1 1 1;
            5 5 1 1 1;];
a=addrule(a,rulelist);
plotmf(a,'input',1);
axis([-pi/2 pi/2 0 1])
figure
plotmf(a,'input',2);
axis([-pi/4 pi/4 0 1])
figure
plotmf(a,'output',1);
axis([-30 30 0 1])
gensurf(a);
```

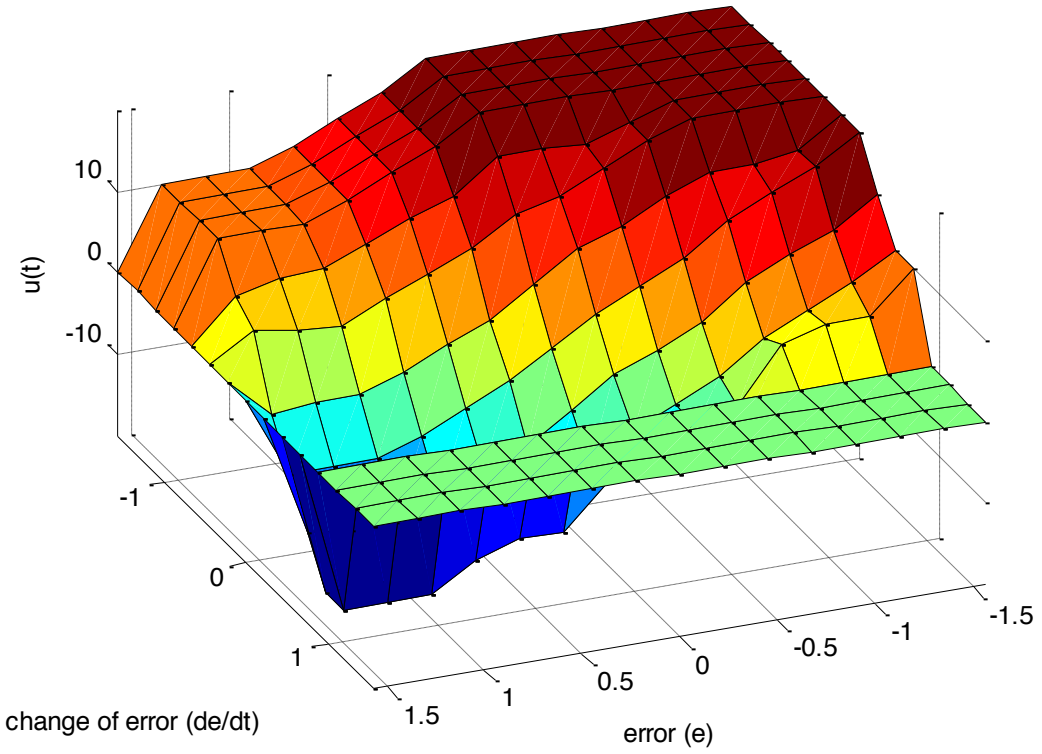


Figure 4: This is the control surface of the fuzzy controller for all gains equal to unity

A Matlab code including a function is written, then using Matlab interface ode23 which can solve for differential equation since the mathematical model is in form of the differential equation; the main is saved "inverted.m" in order to be called later on by the ode23 command. The main code is the following

```
function dx = inverted(t,x)
a=newfis('Inverted Pendalum');
a=addvar(a,'input','error (e)',[-pi/2 pi/2]);
a=addmf(a,'input',1,'NL','trimf', [-inf -pi/2 -pi/4]);
a=addmf(a,'input',1,'NS','trimf', [-pi/2 -pi/4 0]);
a=addmf(a,'input',1,'Z','trimf', [-pi/4 0 pi/4]);
a=addmf(a,'input',1,'PS','trimf', [0 pi/4 pi/2]);
a=addmf(a,'input',1,'PL','trimf', [pi/4 pi/2 +inf]);
a=addvar(a,'input','change of error (de/dt)',[-pi/2 pi/2]);
a=addmf(a,'input',2,'NL','trimf', [-inf -pi/4 -pi/8]);
a=addmf(a,'input',2,'NS','trimf', [-pi/4 -pi/8 0]);
a=addmf(a,'input',2,'Z','trimf', [-pi/8 0 pi/8]);
a=addmf(a,'input',2,'PS','trimf', [0 pi/8 pi/4]);
a=addmf(a,'input',2,'PL','trimf', [pi/8 pi/4 +inf]);
a=addvar(a,'output','u(t)', [-30 30]);
a=addmf(a,'output',1,'NL','trimf', [-30 -20 -10]);
a=addmf(a,'output',1,'NS','trimf', [-20 -10 0]);
a=addmf(a,'output',1,'Z','trimf', [-10 0 10]);
a=addmf(a,'output',1,'PS','trimf', [0 10 20]);
a=addmf(a,'output',1,'PL','trimf', [10 20 30]);
rulelist=[1 1 5 1 1;
          1 2 5 1 1;
          1 3 5 1 1;
          1 4 4 1 1;
          1 5 3 1 1;
          2 1 5 1 1;
          2 2 5 1 1;
          2 3 4 1 1;
          2 4 3 1 1;
          2 5 2 1 1;
          3 1 5 1 1;
          3 2 4 1 1;
          3 3 3 1 1;
          3 4 2 1 1;
          3 5 1 1 1;
          4 1 4 1 1;
          4 2 3 1 1;
          4 3 2 1 1;
          4 4 1 1 1;
          4 5 1 1 1;
          5 1 3 1 1;
          5 2 2 1 1;
          5 3 1 1 1;
          5 4 1 1 1;
          5 5 1 1 1;];
a=addrule(a,rulelist);
g0=1;
```

```
g1=1;
h=1;
inp=[-x(1)*g0 -x(2)*g1];
U=evalfis(inp,a)*h;
dx(1)=x(2);
dx(2)=((9.8*sin(x(1))+cos(x(1))*((-x(3)-
0.25*(x(2))^2*sin(x(1)))/(1.5)))/(0.5*(4/3-(1/3)*(cos(x(1))^2))));
dx(3)= -100*x(3)+100*U;
dx=dx';
end
```

****

For precision a time span of [0 1000] is recommended but in order to compare results with the one found in paper 3 seconds time base was selected;

Solving the system using ode23 the following code must be written in the command window to get the results:

```
[t0,y]=ode23('inverted',[0 1000],[10*pi/180 0 0 ])
subplot(2,1,1),plot(t0,y(:,1))
ylabel ('Angular position (rad)')
subplot(2,1,2),plot(t0,y(:,3))
ylabel ('Input force (N)')
```

For unity gain the first design is the following:

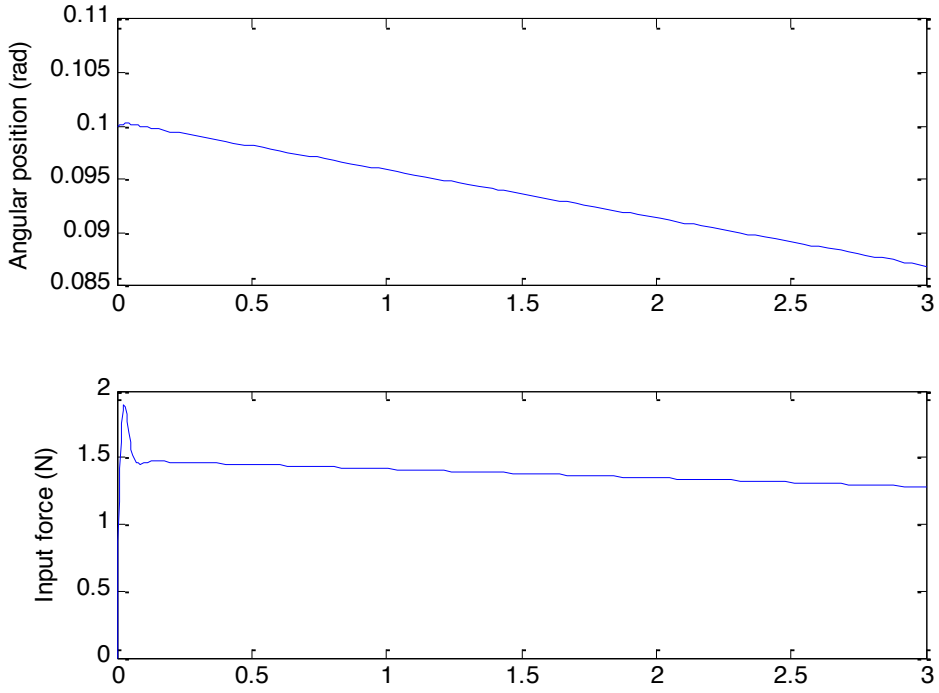


Figure 5: Fuzzy controller balancing an inverted pendulum, first design

The results are the same one as in paper simulation result. As we can see the aim is to balance the pendulum and to approach the angular position to zero.

For a fast response modification have been done by changing the gain values g0=1, g1=0.1 & h=1 :

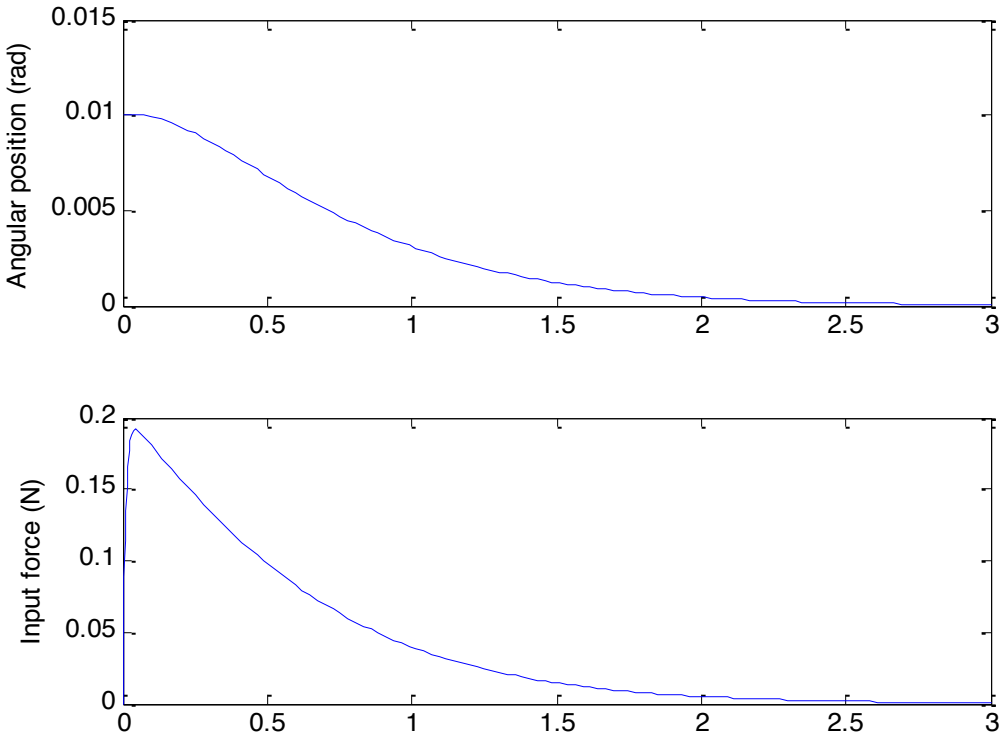


Figure 6: Fuzzy controller balancing an inverted pendulum with g0=1, g1=0.1 and h=1

Results are similar to the one in paper.

Further increasing in h will generally result in faster balancing at the expense of a large control input, At that point the simulation would not reflect the reality since if the controller were actually implemented, the input plant would saturate and the proper balancing behavior may not be achieved.

New results have been investigated by changing the gain values g0=2, g1=0.1 & h=1 :

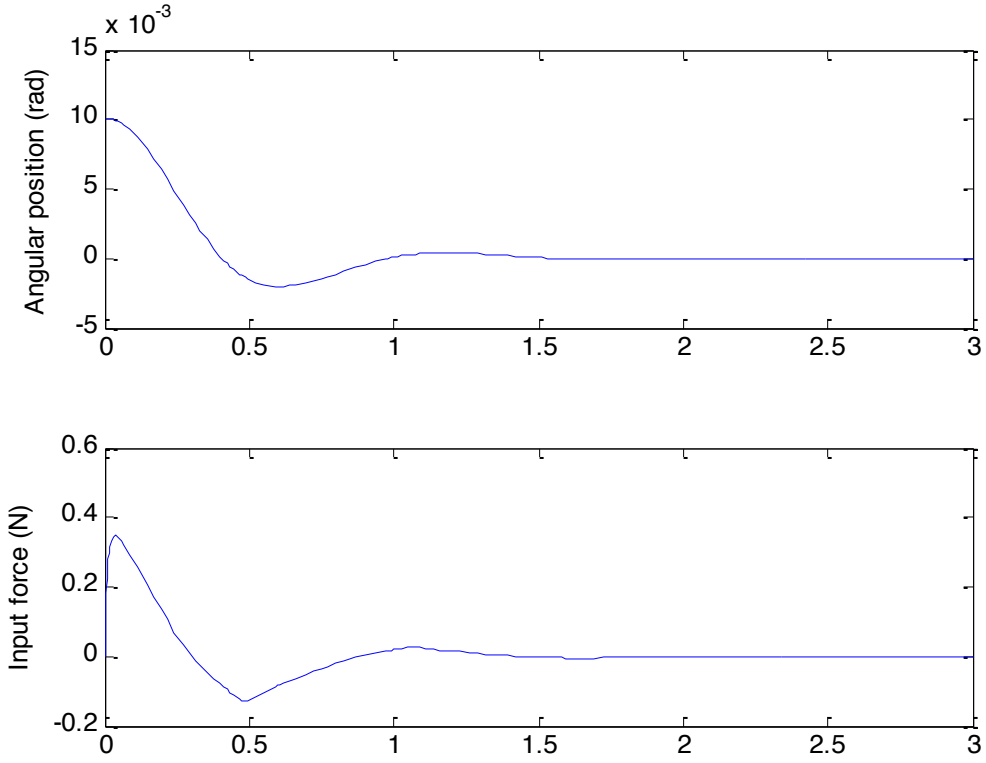


Figure 7: Fuzzy controller balancing an inverted pendulum with g0=2, g1=0.1 and h=1

We see that the change in the scaling gains at the input and output of the fuzzy controller can have significant impact on the performance of the resulting fuzzy control system. These parameters are usually used for tuning.

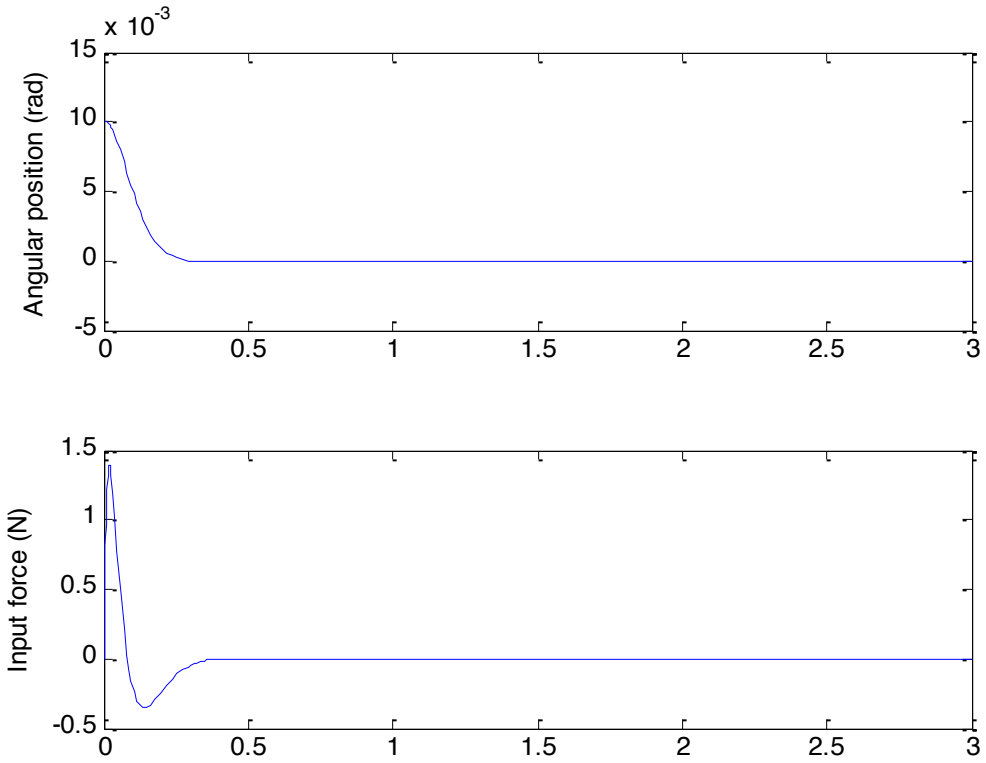


Figure 8: Fuzzy controller balancing an inverted pendulum with g0=2, g1=0.1 and h=5