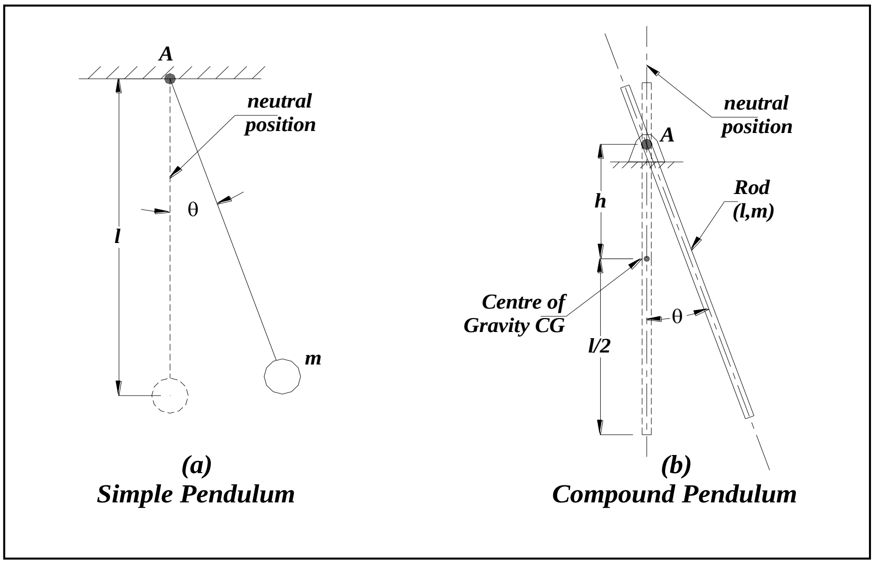
LABORATORY EXERCISE 1: FUZZY SYSTEMS

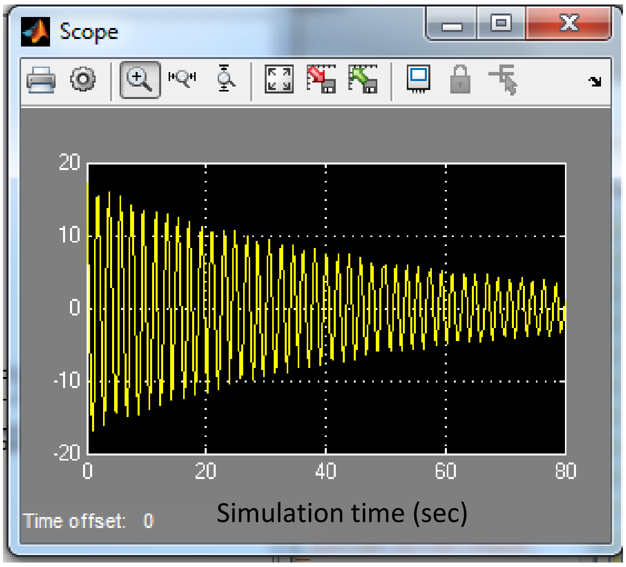
In this lab exercise you are going to develop a compound pendulum by means of Simulink and Matlab fuzzy toolbox. The compound pendulum is an interesting challenge since it can be used as the representation of a robot arm movement.

To this end, you need to understand the problem and develop part of the Mamdani controller as it will be explained latter.

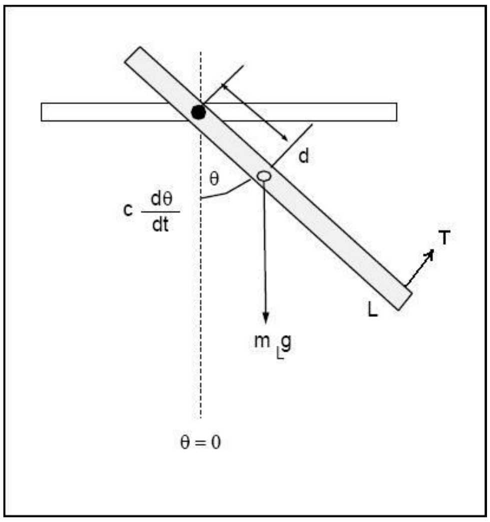
The compound pendulum is schematically shown in next figure, and it consists of a uniform slender bar of total mass *m* and length *l*, which may be suspended at various points *A* along the bar with the aid of a sliding pivot situated at any distance *h* from the centre of gravity (CG) of the pendulum. The centre of mass is at the middle of the rod. The pendulum has a motor with a propeller.



Such a system has a natural oscillatory response. This is corroborated by the open-loop response to a 2 volt step input as seen in the figure below.



You can see that even after 80 seconds, the pendulum is still oscillating more than +/- 2 degrees. This underscores the need for control methodologies to be implemented on the thrust tester to improve the system's transient response. Towards this, an encoder is mounted to the pivot point shaft to measure the angle subtended by the pendulum. Feeding this information back into the system allows for many different approaches to quickly stabilize the compound pendulum at a desired angle.



From the free body diagram above, it can be seen that evaluating Newton's second law about the pivot point of the pendulum yields the following equation:

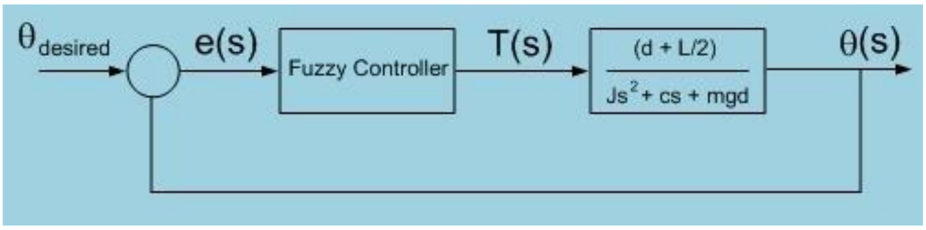
$$J\ddot{\theta} + c\,\dot{\theta} + mgd\,\sin\theta = \left(d + L/2\right)T$$

where • (one dot) stands for the first derivative and •• (two dots) stands for the second derivative. *L* is the bar length, *d* is the pivot to CG distance, *m* is the mass of pendulum, *J* is the moment of Inertia, *c* is the viscous damping and *g* is the Earth's Gravity.

The open-loop transfer function of the plant can be found by taking the Laplace Transform of the second-order differential equation (and linearizing the sin(theta) term => sin(theta)=theta):

$$\frac{\theta(s)}{T(s)} = \frac{d + L/2}{Js^2 + cs + mgd}$$

Therefore, the system's control block diagram is:

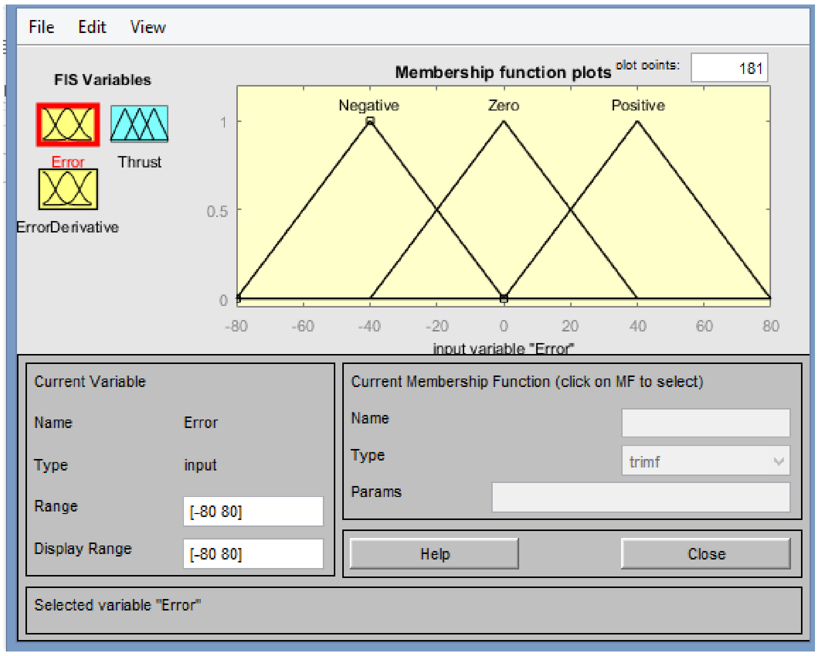


where **e** is the error.

In this exercise you must design a Mamdani FIS and implement the complete system by means of Simulink.

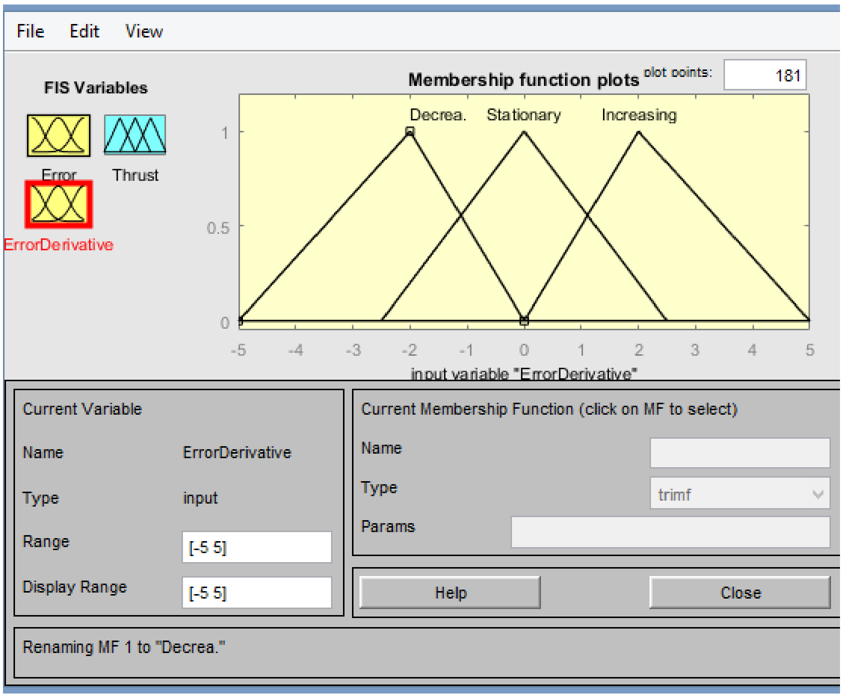
You will start by developing the Mamdani FIS who will act as controller. To do so, you can use the information that follows:

First you can define the fuzzy sets to represent the error measure as described in the next figure:



For example, the input for our thrust tester system is the subtended angle (measured by an encoder) and you are trying to stabilize this at 20 degrees. If the angle was measured at 40 degrees, then the error would be -20 degrees. This would make the membership values of negative error (NE) and zero error (ZE) both equal to 0.5. The positive error (PE) variable would be assigned a value of 0.

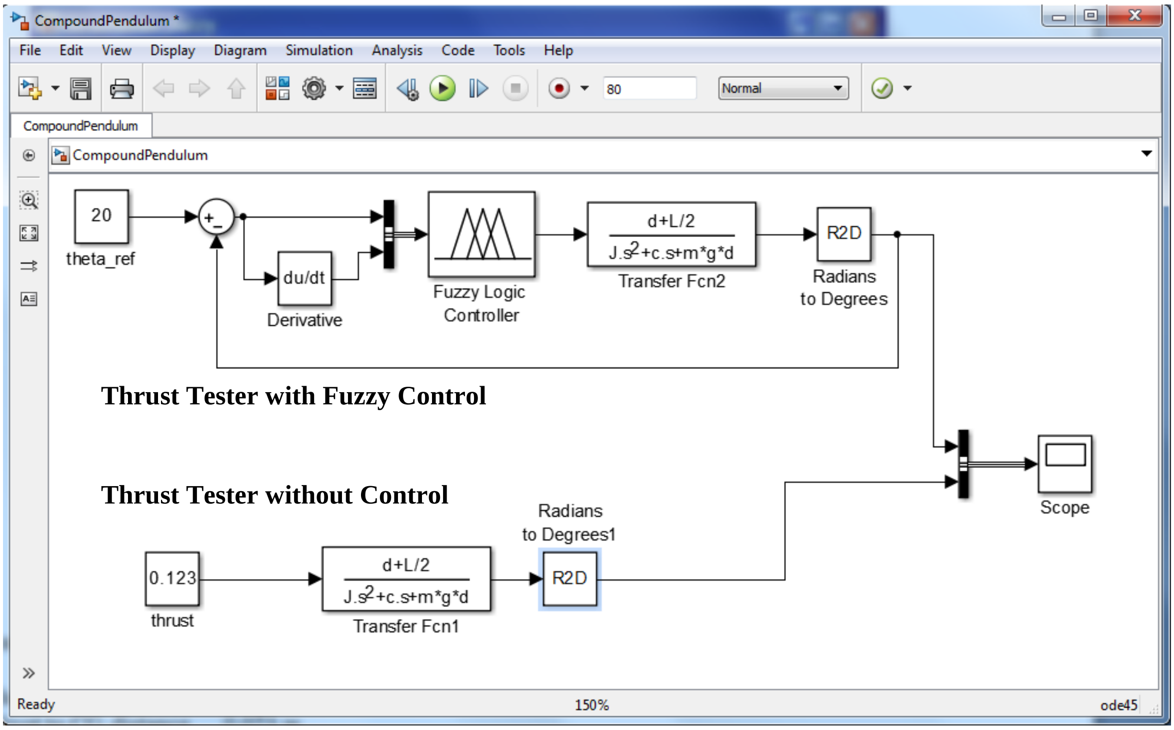
To further increase the effectiveness of the controller, the derivative of the error should also be evaluated. Therefore, the system will be able to determine if the pendulum is moving further from or towards the desired angle at any instant (i.e. if the error is increasing, decreasing or remaining constant). Since the sampling time is on the order of milliseconds, the change in error will never be greater than a few degrees. As such, the bounds on the membership function is +/- 5 and is shown below.



Similar membership functions can be defined for the output variable, thrust, using bounds between -25 and 25. You should define three fuzzy sets that represent the output and characterize them through 3 membership functions.

Next, you should define the rule base to be used by the fuzzy controller. Your rule base will have 9 rules (maximum), since the system has two input variables represented by 3 membership functions each.

Once you have the Mamdani FIS available, you can create the whole model using Simulink. The derivative and the transfer function blocks are in the "Simulink / Continuous" module. Theta_ref and Thrust are constant blocks defined in the "Simulink / Commonly Used Blocks". You can find R2D (Radians to Degrees) block in "Simulink Extras / Transformations" module. Follow next figure.



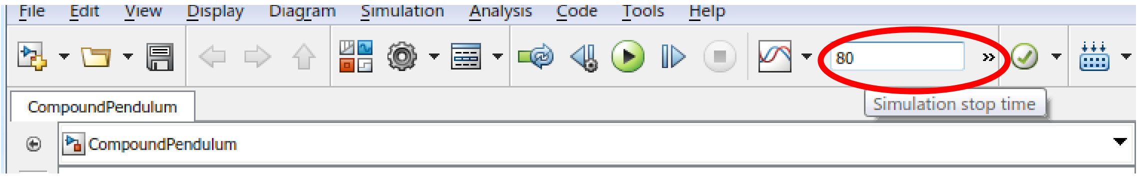
You can set, initially, the parameters to the following values:

L	Bar length	0.495 *m*
d	Pivot to CG distance	0.023 *m*
m	Mass of pendulum	0.43 *kg*
J	Moment of Inertia	0.0090 *kgm2*
c	Viscous damping	0.00035 *Nms / rad*
g	Earth's Gravity 9.80665 m/s^2	

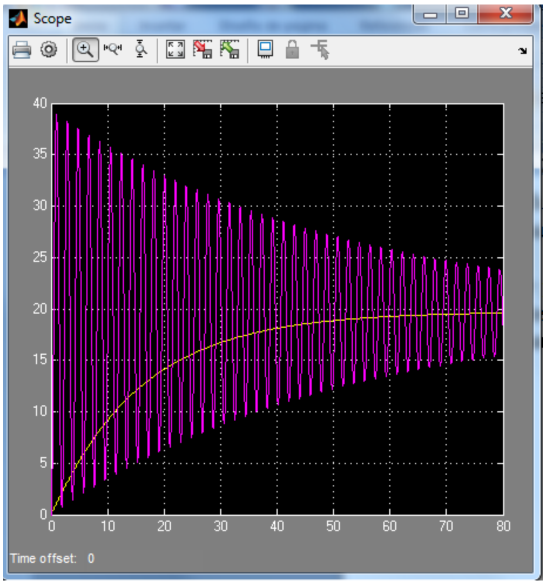
To do so, you just have to define them in the workspace.

In the *Simulation / Model Configuration Parameters / Solver / Solver Details* set the *Zero-crossing control* to *disable all* and in the *Simulation / Model Configuration Parameters / Diagnostics* set the *Automatic solver parameter selection* to *none*. Remember that you have to choose *Boolean logic off* in SIMULINK, otherwise the fuzzy block doesn't work. You should do the following: from the Simulink screen go to *Simulation / Model Configuration parameters / Math and Data Types* / … / *Advanced Parameters* / find: *Implement logic signals as Boolean data* and then deactivate it.

Select a simulation stop time of 80 and run it.



The plot shows the 2 volt step response, that correspond to a torque value of 0.123, of the open loop system (pink) and the response under fuzzy control (yellow). In both cases, the system stabilizes at around 20 degrees. However, the fuzzy controller has a much quicker settling time with no overshoot.



Write a small document (two pages maximum) that includes:

1) Description of the membership functions that you have designed for the output variable and the rule base defined. Argue why you have chosen these membership and rules.
2) Plots of the results that you get for the following values with a simulation stop time of 80:
   2.1) theta_ref = 20 and thrust = 0.123
   2.2) theta_ref = -10 and thrust = -0.062
   Write your own comments about the results that you get.
3) What happens if you increase the number of membership functions of the output?