Fuzzy Logic Examples using Matlab

Consider a very simple example:
We need to control the speed of a motor by changing the input voltage. When a set point is defined, if for some reason, the motor runs faster, we need to slow it down by reducing the input voltage. If the motor slows below the set point, the input voltage must be increased so that the motor speed reaches the set point.

Let the input status words be:

Too slow
Just right
Too fast

Let the output action words be:

Less voltage (Slow down)
No change
More voltage (Speed up)

Define the rule-base:

1. If the motor is running too slow, then more voltage.
2. If motor speed is about right, then no change.
3. If motor speed is to fast, then less voltage.

Define the membership functions for inputs and output variable as shown in figure below.

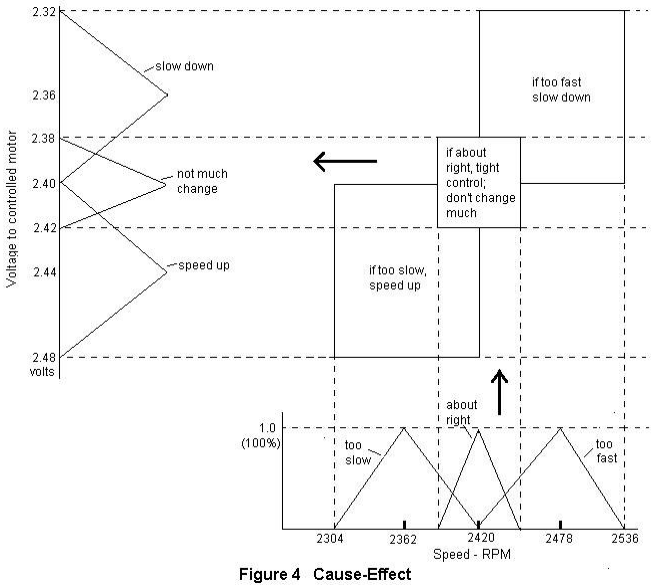

Figure 4 Cause-Effect

Figure 1. Membership Functions

Suppose, the speed increases from the set point of 2420 to 2437.4 rpm. This is depicted on the membership function as shown below.

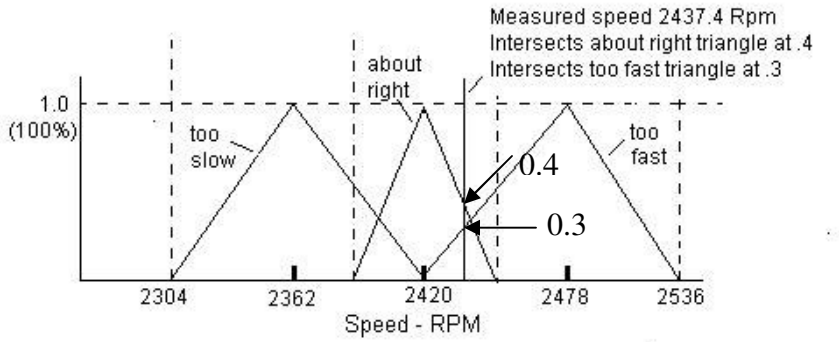

Figure 2. Speed above set point

The intersection points would be 0.4 and 0.3. From figure 1, we see that this speed would only intersect the rectangles consisting of rules 2 and 3. We now change the height of the triangles for input voltage.

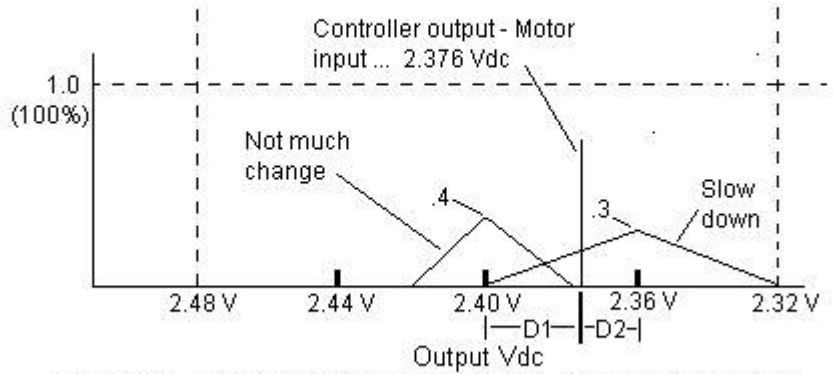

Figure 3. Motor voltage

Now, area of "Not much change" triangle is 0.008 and area of "Slow down" triangle is 0.012.

The output, as seen in Figure 3 (above), is determined by calculating the point at which a fulcrum would balance the two triangles.

Thus,

$0.008 \times D1 = 0.012 \times D2$ (1)

$D1 + D2 = 0.04$ (2)

Solving (1) and (2) simultaneously we get,

D1=0.024

D2=0.016

Thus the voltage required would be 2.40-0.024=2.376 V

Let's solve this using Matlab.

Type "fuzzy" in the Matlab command prompt.

Draw the appropriate membership functions as shown below:

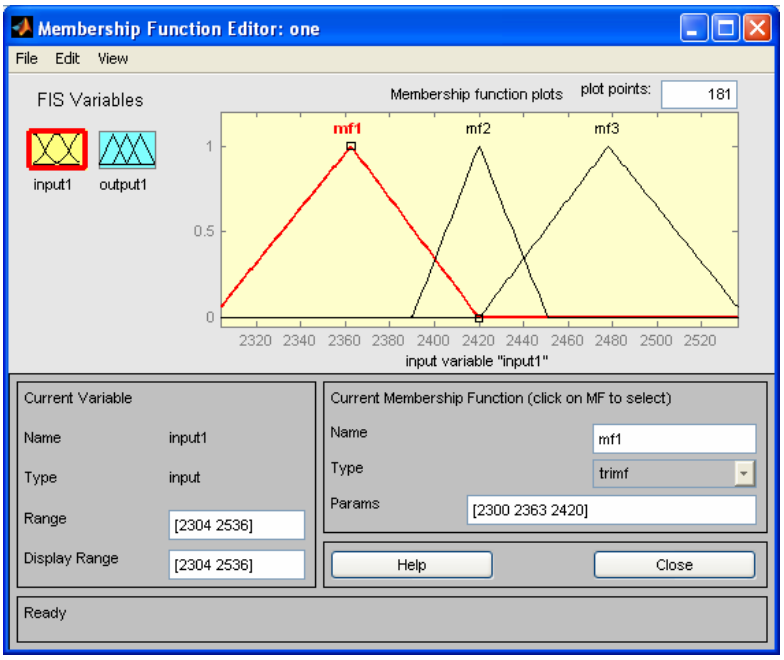

Figure 4. Input Membership Function

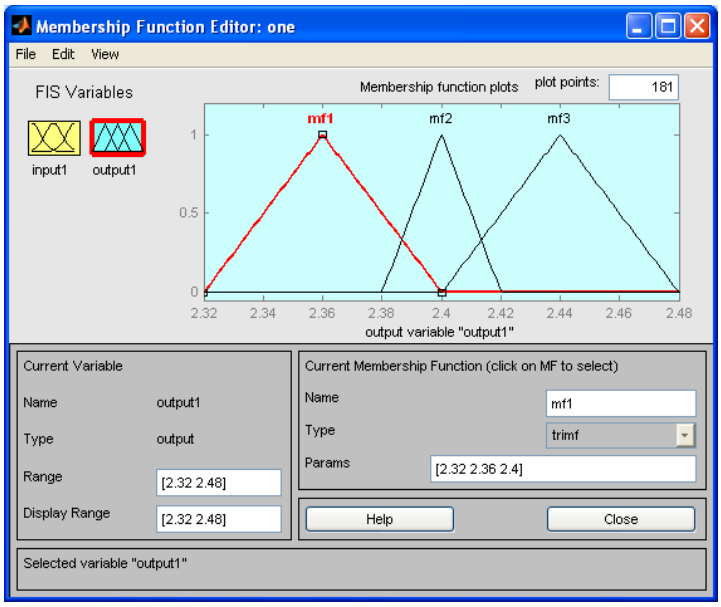

Figure 5. Output Membership Function

Now set the rules 1-3 as defined earlier.

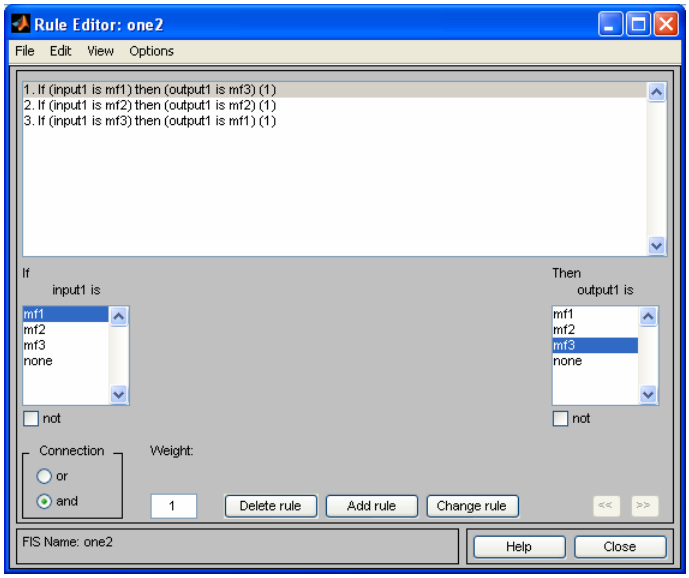

Figure 6. Rule Base

Save the file as "one.fis".
Now type in the following to get the result for the same example:

```
fis = readfis('one');
out=evalfis(2437.4,fis)
```

```
>>out =

    2.376  (Same as above)
```

Consider the next example with two inputs and one output.

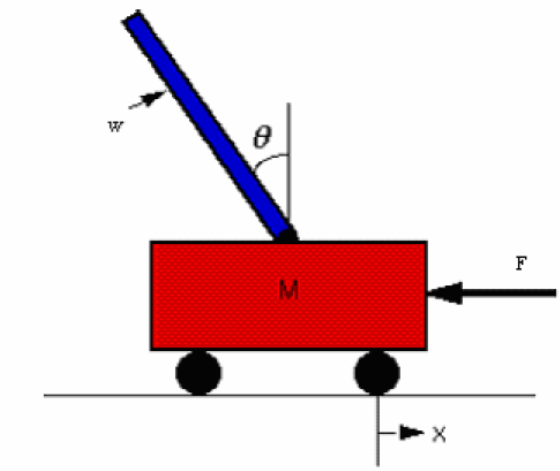

Figure 7. Inverted Pendulum

Let the inputs be angle and angular velocity and the controller output be the force on the mass.

Now, define the rule base as:

ANGULAR VELOCITY	ANGLE				
	Neg. High	**Neg. Low**	**Zero**	**Pos. Low**	**Pos. High**
Neg. High	Neg. High	Neg. High	Neg. High	Neg. Low	Zero
Neg. Low	Neg. High	Neg. High	Neg. Low	Zero	Pos. Low
Zero	Neg. High	Neg. Low	Zero	Pos. Low	Pos. High
Pos. Low	Neg. Low	Zero	Pos. Low	Pos. High	Pos. High
Pos. High	Zero	Pos. Low	Pos. High	Pos. High	Pos. High

Table 1

Consider a scenario:

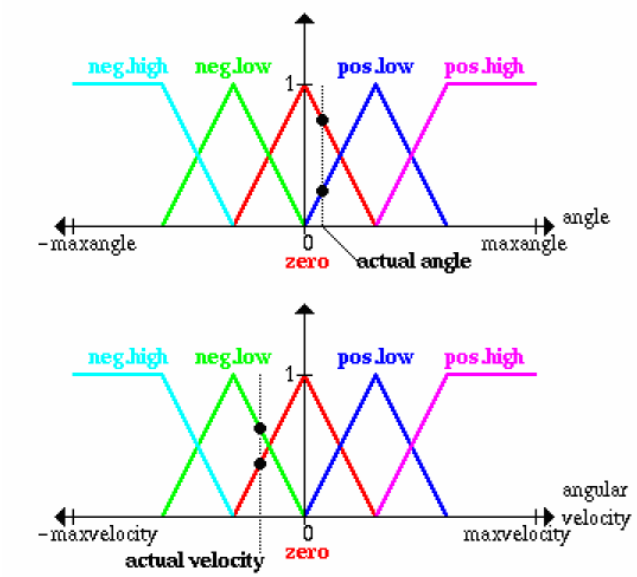

Figure 8. Memberships for angle and velocity

Let the measured angle and velocity be as shown in the figure above.
We see that this will fire 4 rules:

1. If angle is zero and velocity is zero then force is zero
2. If angle is zero and velocity is NL then force is NL
3. If angle is PL and velocity is zero then force is PL
4. If angle is PL anfd velocity is NL then force is zero

Now, as explained in the previous class notes,

$$f(x) = \frac{\sum_{i=1}^{N} z^i \prod_{j=1}^{n} \mu_{ij}(x_j)}{\sum_{i=1}^{N} \prod_{j=1}^{n} \mu_{ij}(x_j)}$$

From the above formula,

$$\text{force} = \frac{\text{“Zero”} \times (\mu_{33}) + \text{“Neg. Low”} \times (\mu_{23}) + \text{“Pos. Low”} \times (\mu_{34}) + \text{“Zero”} \times (\mu_{24})}{(\mu_{33} + \mu_{23} + \mu_{34} + \mu_{24})}$$

where $\mu_{33} = 0.75 \times 0.4 = 0.3$

$\mu_{23} = 0.75 \times 0.6 = 0.45$

$\mu_{34} = 0.4 \times 0.25 = 0.1$

$\mu_{24} = 0.6 \times 0.25 = 0.15$

= -0.35 N

We will solve this using Matlab. Define the inputs and output similar to example 1.

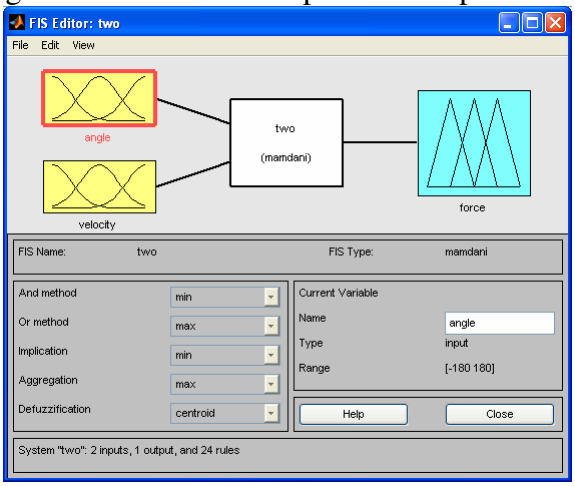

Figure 9. Inputs and one output

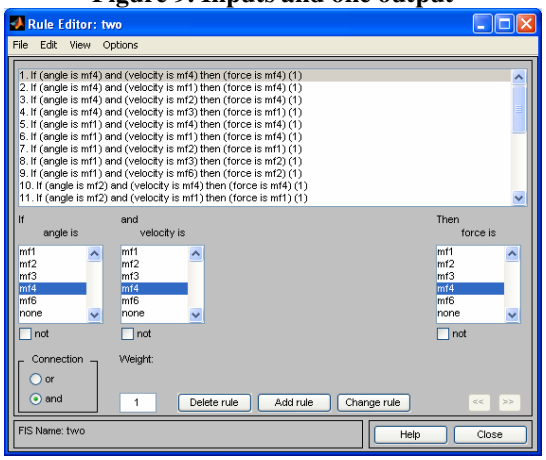

Figure 10. Rule base

Using the same command lines as in example 1:

```
fis = readfis('two');
out=evalfis([65 -0.1],fis)
```

```
>>out =

    0.3570 (same as above)
```

We can actually visualize the output surface of the fuzzy system using the command "surfview(fis)"

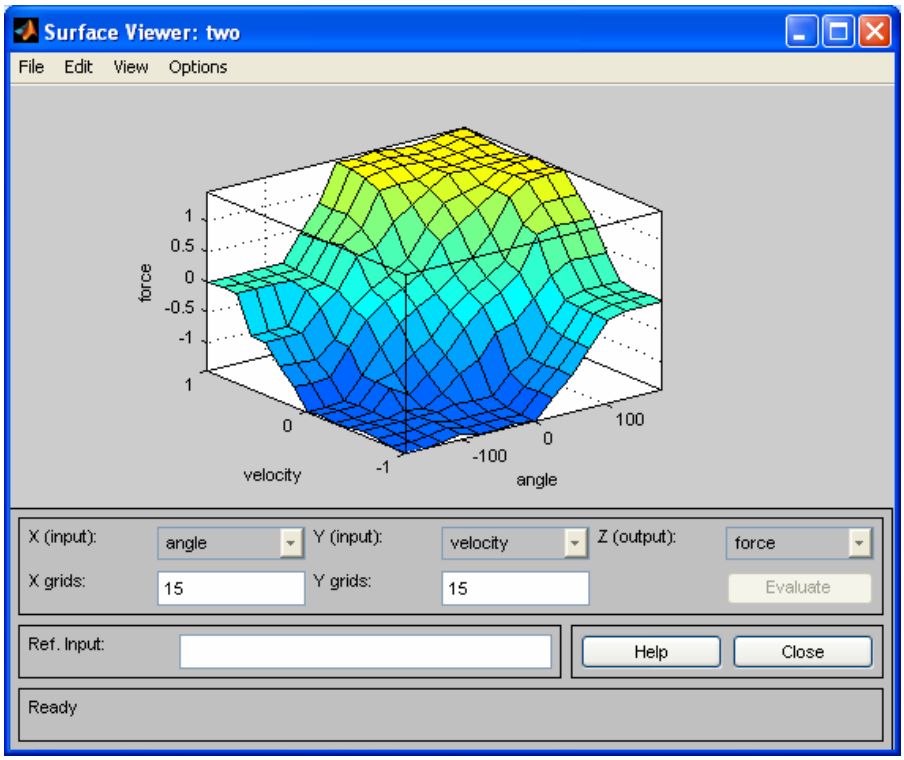

Figure 11. Output surface of the fuzzy system