Working from the Command Line

The tipping system is one of many examples of fuzzy inference systems provided with the Fuzzy Logic Toolbox. The FIS is always cast as a MATLAB structure. To load this system (rather than bothering with creating it from scratch), type

```matlab
a = readfis('tipper.fis')
```

MATLAB will respond with

```matlab
a =
```
```
name: 'tipper'
type: 'mamdani'
andMethod: 'min'
orMethod: 'max'
defuzzMethod: 'centroid'
impMethod: 'min'
aggMethod: 'max'
input: [1x2 struct]
output: [1x1 struct]
rule: [1x3 struct]
```

The labels on the left of this listing represent the various components of the MATLAB structure associated with `tipper.fis`. You can access the various components of this structure by typing the component name after typing `a`. At the MATLAB command line, type

```matlab
a.type
```

for example. MATLAB will respond with

```matlab
ans =
mamdani
```

The function

```matlab
getfis(a)
```

returns almost the same structure information that typing `a`, alone does.
getfis(a) returns

Name = tipper
Type = mamdani
NumInputs = 2
InLabels =
  service
  food
NumOutputs = 1
OutLabels =
  tip
NumRules = 3
AndMethod = min
OrMethod = max
ImpMethod = min
AggMethod = max
DefuzzMethod = centroid

Notice that some of these fields are not part of the structure, a. Thus, you cannot get information by typing a.Inlabels, but you can get it by typing

getfis(a,'Inlabels')

Similarly, you can obtain structure information using getfis in this manner.

getfis(a,'input',1)
getfis(a,'output',1)
getfis(a,'input',1,'mf',1)

The structure.field syntax also generates this information. For more information on the syntax for MATLAB structures and cell arrays, see the MATLAB documentation.

For example, type

a.input

or

a.input(1).mf(1)
The function `getfis` is loosely modeled on the Handle Graphics® function `get`. There is also a function called `setfis` that acts as the reciprocal to `getfis`. It allows you to change any property of an FIS. For example, if you wanted to change the name of this system, you could type

```matlab
a = setfis(a,'name','gratuity');
```

However, since `a` is already a MATLAB structure, you can set this information more simply by typing

```matlab
a.name = 'gratuity';
```

Now the FIS structure `a` has been changed to reflect the new name. If you want a little more insight into this FIS structure, try

```matlab
showfis(a)
```

This returns a printout listing all the information about `a`. This function is intended more for debugging than anything else, but it shows all the information recorded in the FIS structure.

Since the variable, `a`, designates the fuzzy tipping system, you can display any of the GUIs for the tipping system directly from the command line. Any of the following will display the tipping system with the associated GUI:

- `fuzzy(a)` displays the FIS Editor.
- `mfedit(a)` displays the Membership Function Editor.
- `ruleedit(a)` displays the Rule Editor.
- `ruleview(a)` displays the Rule Viewer.
- `surfview(a)` displays the Surface Viewer.

If, in addition, `a` is a Sugeno-type FIS, then `anfisedit(a)` will display the ANFIS Editor GUI.

Once any of these GUIs has been opened, you can access any of the other GUIs using the pull-down menu rather than the command line.
System Display Functions

There are three functions designed to give you a high-level view of your fuzzy inference system from the command line: `plotfis`, `plotmf`, and `gensurf`. The first of these displays the whole system as a block diagram much as it would appear on the FIS Editor.

```
plotfis(a)
```

![Block diagram of fuzzy system](image)

After closing any open MATLAB figures or GUI windows, the function `plotmf` plots all the membership functions associated with a given variable as follows.

```
plotmf(a, 'input', 1)
```

returns
plotmf(a,'output',1)
These plots will appear in the Membership Function Editor GUI, or in an open MATLAB figure, if \texttt{plotmf} is called while either of these is open.

Finally, the function \texttt{gensurf} will plot any one or two inputs versus any one output of a given system. The result is either a two-dimensional curve, or a three-dimensional surface. Note that when there are three or more inputs, \texttt{gensurf} must be generated with all but two inputs fixed, as is described in \texttt{gensurf}.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{gensurf.png}
\end{figure}

\textbf{Building a System from Scratch}

It is possible to use the Fuzzy Logic Toolbox without bothering with the GUI tools at all. For instance, to build the tipping system entirely from the command line, you would use the commands \texttt{newfis}, \texttt{addvar}, \texttt{addmf}, and \texttt{addrule}.

Probably the trickiest part of this process is learning the shorthand that the fuzzy inference systems use for building rules. This is accomplished using the command line function, \texttt{addrule}.
Each variable, input, or output, has an index number, and each membership function has an index number. The rules are built from statements like this.

If input1 is MF1 or input2 is MF3, then output1 is MF2 (weight = 0.5)

This rule is turned into a structure according to the following logic. If there are \( m \) inputs to a system and \( n \) outputs, then the first \( m \) vector entries of the rule structure correspond to inputs 1 through \( m \). The entry in column 1 is the index number for the membership function associated with input 1. The entry in column 2 is the index number for the membership function associated with input 2, and so on. The next \( n \) columns work the same way for the outputs. Column \( m + n + 1 \) is the weight associated with that rule (typically 1) and column \( m + n + 2 \) specifies the connective used (where AND = 1 and OR = 2).

The structure associated with the rule shown above is

\[
1 \quad 3 \quad 2 \quad 0.5 \quad 2
\]

Here is one way you can build the entire tipping system from the command line, using the MATLAB structure syntax.

```matlab
a=newfis('tipper');
a.input(1).name='service';
a.input(1).range=[0 10];
a.input(1).mf(1).name='poor';
a.input(1).mf(1).type='gaussmf';
a.input(1).mf(1).params=[1.5 0];
a.input(1).mf(2).name='good';
a.input(1).mf(2).type='gaussmf';
a.input(1).mf(2).params=[1.5 5];
a.input(1).mf(3).name='excellent';
a.input(1).mf(3).type='gaussmf';
a.input(1).mf(3).params=[1.5 10];
a.input(2).name='food';
a.input(2).range=[0 10];
a.input(2).mf(1).name='rancid';
a.input(2).mf(1).type='trapmf';
a.input(2).mf(1).params=[-2 0 1 3];
a.input(2).mf(2).name='delicious';
a.input(2).mf(2).type='trapmf';
a.input(2).mf(2).params=[7 9 10 12];
```
a.output(1).name = 'tip';
a.output(1).range = [0 30];
a.output(1).mf(1).name = 'cheap';
a.output(1).mf(1).type = 'trimf';
a.output(1).mf(1).params = [0 5 10];
a.output(1).mf(2).name = 'average';
a.output(1).mf(2).type = 'trimf';
a.output(1).mf(2).params = [10 15 20];
a.output(1).mf(3).name = 'generous';
a.output(1).mf(3).type = 'trimf';
a.output(1).mf(3).params = [20 25 30];
a.rule(1).antecedent = [1 1];
a.rule(1).consequent = [1];
a.rule(1).weight = 1;
a.rule(1).connection = 2;
a.rule(2).antecedent = [2 0];
a.rule(2).consequent = [2];
a.rule(2).weight = 1;
a.rule(2).connection = 1;
a.rule(3).antecedent = [3 2];
a.rule(3).consequent = [3];
a.rule(3).weight = 1;
a.rule(3).connection = 2

Alternatively, here is how you can build the entire tipping system from the command line using Fuzzy Logic Toolbox commands. These commands are in the mktipper.m demo file.

a = newfis('tipper');
a = addvar(a, 'input', 'service', [0 10]);
a = addmf(a, 'input', 1, 'poor', 'gaussmf', [1.5 0]);
a = addmf(a, 'input', 1, 'good', 'gaussmf', [1.5 5]);
a = addmf(a, 'input', 1, 'excellent', 'gaussmf', [1.5 10]);
a = addvar(a, 'input', 'food', [0 10]);
a = addmf(a, 'input', 2, 'rancid', 'trapmf', [-2 0 1 3]);
a = addmf(a, 'input', 2, 'delicious', 'trapmf', [7 9 10 12]);
a = addvar(a, 'output', 'tip', [0 30]);
a = addmf(a, 'output', 1, 'cheap', 'trimf', [0 5 10]);
a = addmf(a, 'output', 1, 'average', 'trimf', [10 15 20]);
a = addmf(a, 'output', 1, 'generous', 'trimf', [20 25 30]);
ruleList=[ ... 
1 1 1 1 2 
2 0 2 1 1 
3 2 3 1 2 ]; 

a=addrule(a,ruleList);

**FIS Evaluation**

To evaluate the output of a fuzzy system for a given input, use the function `evalfis`. For example, the following script evaluates `tipper` at the input, [1 2].

```matlab
a = readfis('tipper');
evalfis([1 2], a)
ans =
  5.5586
```

This function can also be used for multiple collections of inputs, since different input vectors are represented in different parts of the input structure.

```matlab
evalfis([3 5; 2 7], a)
ans =
  12.2184 
  7.7885
```

**The FIS Structure**

The FIS structure is the MATLAB object that contains all the fuzzy inference system information. This structure is stored inside each GUI tool. Access functions such as `getfis` and `setfis` make it easy to examine this structure.

All the information for a given fuzzy inference system is contained in the FIS structure, including variable names, membership function definitions, and so on. This structure can itself be thought of as a hierarchy of structures, as shown in the following diagram.
You can generate a listing of information on the FIS using the `showfis` command, as shown below.

```
showfis(a)
1. Name          tipper
2. Type          mamdani
3. Inputs/Outputs [ 2 1 ]
4. NumInputMFs   [ 3 2 ]
5. NumOutputMFs  3
6. NumRules      3
7. AndMethod     min
8. OrMethod      max
9. ImpMethod     min
10. AggMethod    max
11. DefuzzMethod centroid
12. InLabels     service
13.              food
14. OutLabels    tip
15. InRange      [ 0 10 ]
16.              [ 0 10 ]
17. OutRange     [ 0 30 ]
```
18. InMFLabels  poor
19. good
20. excellent
21. rancid
22. delicious
23. OutMFLabels  cheap
24. average
25. generous
26. InMFTypes  gaussmf
27. gaussmf
28. gaussmf
29. trapmf
30. trapmf
31. OutMFTypes  trimf
32. trimf
33. trimf
34. InMFParams  [ 1.5 0 0 0 ]
35. [ 1.5 5 0 0 ]
36. [ 1.5 10 0 0 ]
37. [ 0 0 1 3 ]
38. [ 7 9 10 10 ]
39. OutMFParams  [ 0 5 10 0 ]
40. [ 10 15 20 0 ]
41. [ 20 25 30 0 ]
42. Rule Antecedent  [ 1 1 ]
43. [ 2 0 ]
44. [ 3 2 ]
42. Rule Consequent  1
43. 2
44. 3
42. Rule Weight  1
43. 1
44. 1
42. Rule Connection  2
43. 1
44. 2

The list of command-line functions associated with FIS construction includes getfis, setfis, showfis, addvar, addmf, addrule, rmvar, and rmmf.
**Saving FIS Files on Disk**

A specialized text file format is used for saving fuzzy inference systems to a disk. The functions `readfis` and `writefis` are used for reading and writing these files.

If you prefer, you can modify the FIS by editing its `.fis` text file rather than using any of the GUls. You should be aware, however, that changing one entry may oblige you to change another. For example, if you delete a membership function using this method, you also need to make certain that any rules requiring this membership function are also deleted.

The rules appear in indexed format in a `.fis` text file. Here is the file `tipper.fis`.

```
[System]
Name='tipper'
Type='mamdani'
NumInputs=2
NumOutputs=1
NumRules=3
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='service'
Range=[0 10]
NumMFs=3
MF1='poor':'gaussmf',[1.5 0]
MF2='good':'gaussmf',[1.5 5]
MF3='excellent':'gaussmf',[1.5 10]

[Input2]
Name='food'
Range=[0 10]
NumMFs=2
MF1='rancid':'trapmf',[0 0 1 3]
MF2='delicious':'trapmf',[7 9 10 10]
```
[Output1]
Name='tip'
Range=[0 30]
NumMFs=3
MF1='cheap':'trimf',[0 5 10]
MF2='average':'trimf',[10 15 20]
MF3='generous':'trimf',[20 25 30]

[Rules]
1 1, 1 (1) : 2
2 0, 2 (1) : 1
3 2, 3 (1) : 2