**Lab # 9**

**Fuzzy Logic Toolbox Graphical User Interface Tools**

This lab shows how to build a fuzzy inference system (FIS) for the tipping example, described in The Basic Tipping Problem, using the Fuzzy Logic Toolbox™ UI tools. These tools support only type-1 fuzzy systems.

You use the following tools to build, edit, and view fuzzy inference systems:

∙ **Fuzzy Logic Designer** to handle the high-level issues for the system — How many input and output variables? What are their names?

Fuzzy Logic Toolbox software does not limit the number of inputs. However, the number of inputs may be limited by the available memory of your machine. If the number of inputs is too large, or the number of membership functions is too big, then it may also be difficult to analyze the FIS using the other tools.

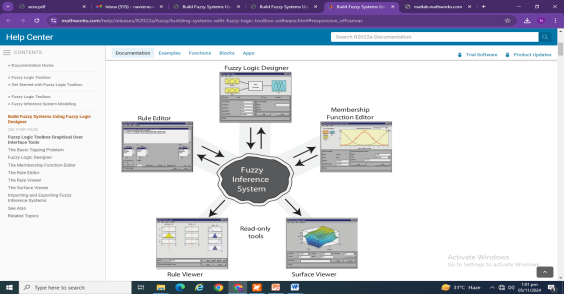
∙ **Membership Function Editor** to define the shapes of all the membership functions associated with each variable ∙ **Rule Editor** to edit the list of rules that defines the behavior of the system.

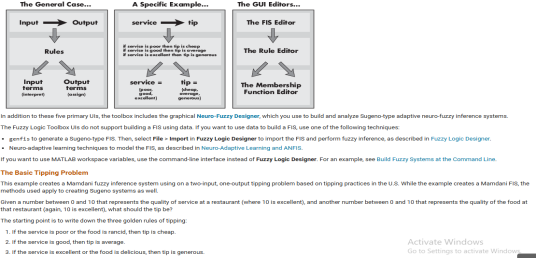
∙ **Rule Viewer** to view the fuzzy inference diagram. Use this viewer as a diagnostic to see, for example, which rules are active, or how individual membership function shapes influence the results

∙ **Surface Viewer** to view the dependency of one of the outputs on any one or two of the inputs; that is, it generates and plots an output surface map for the system.

These UIs are dynamically linked, in that changes you make to the FIS using one of them, affect what you see on any of the other open UIs. For example, if you change the names of the membership functions in the Membership Function Editor, the changes are reflected in the rules shown in the Rule Editor. You can use the UIs to read and write variables both to the MATLAB® workspace and to a file (the

read-only viewers can still exchange plots with the workspace and save them to a file). You can have any or all of them open for any given system or have multiple editors open for any number of fuzzy systems.

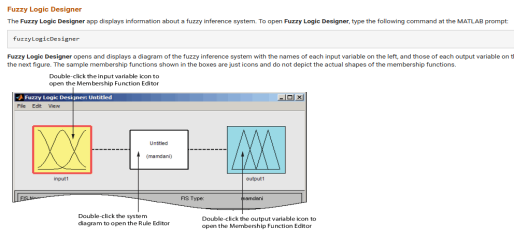


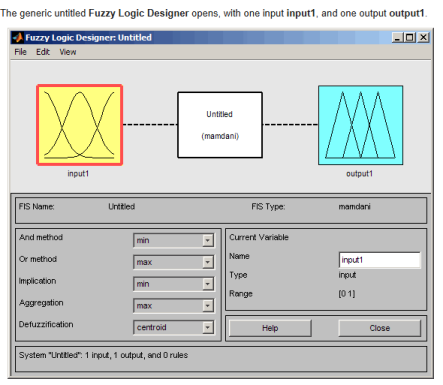
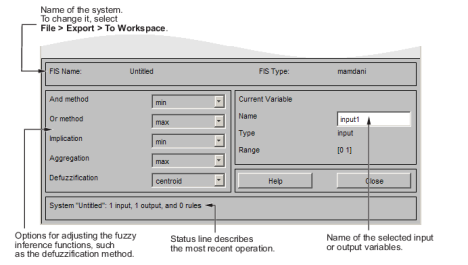


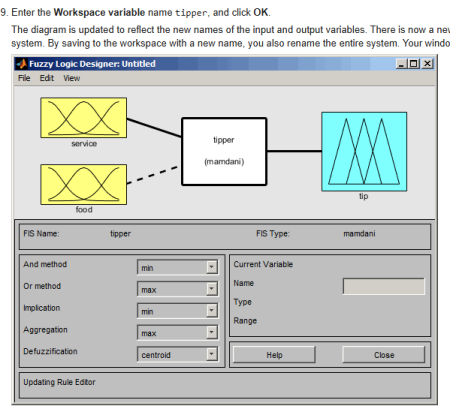
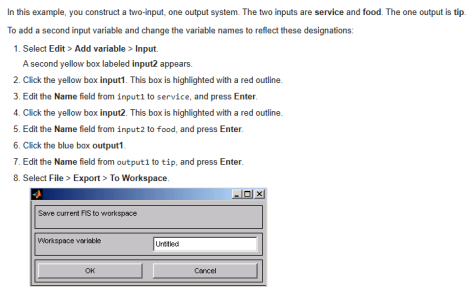
**Fuzzy Logic Designer**

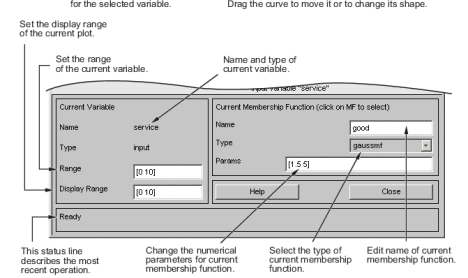
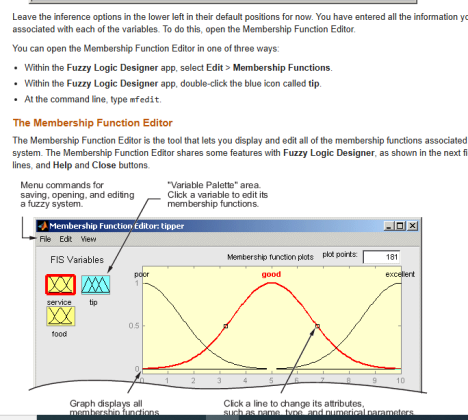
The **Fuzzy Logic Designer** app displays information about a fuzzy inference system. To open **Fuzzy Logic Designer**, type the following command at the MATLAB prompt:

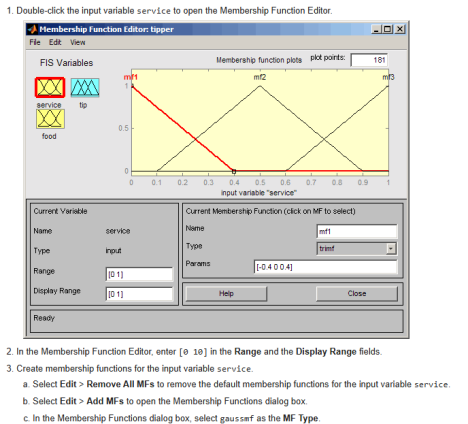
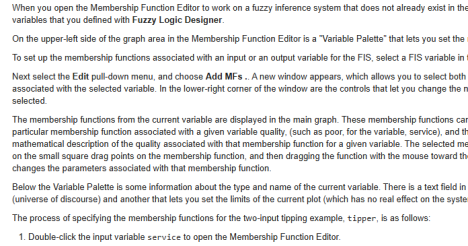
fuzzyLogicDesigner

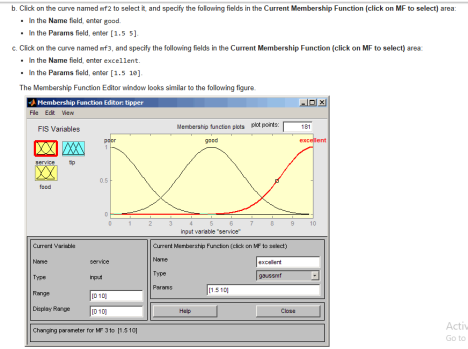
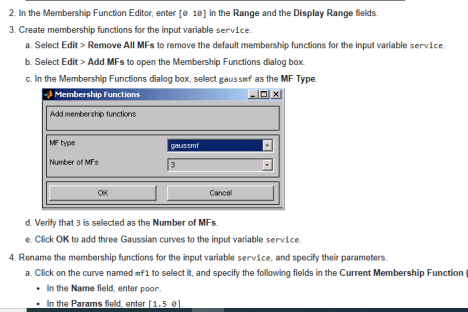


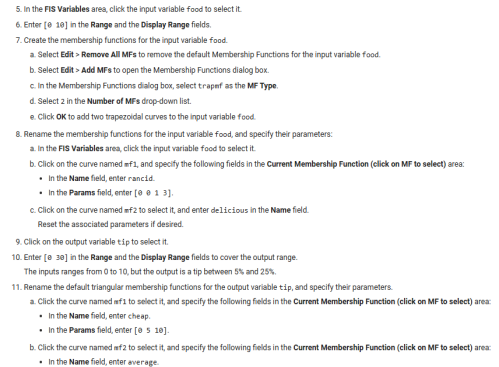
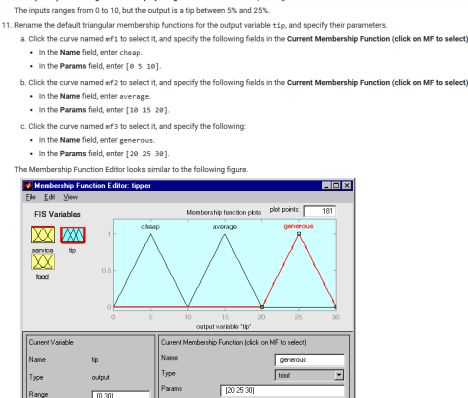
















Constructing rules using the graphical Rule Editor interface is fairly self evident. Based on the descriptions of the input and output variables defined with **Fuzzy Logic Designer**, the Rule Editor allows you to construct the rule statements automatically. You can:

∙ Create rules by selecting an item in each input and output variable box, selecting one **Connection** item, and clicking **Add Rule**. You can choose none as one of the variable qualities to exclude that variable from a given rule and choose not under any variable name to negate the associated quality.

∙ Delete a rule by selecting the rule and clicking **Delete Rule**.

∙ Edit a rule by changing the selection in the variable box and clicking **Change Rule**.

∙ Specify weight to a rule by typing in a desired number between 0 and 1 in **Weight**. If you do not specify the weight, it is assumed to be unity (1).

Similar to those in **Fuzzy Logic Designer** and the Membership Function Editor, the Rule Editor has the menu bar and the status line. The menu items allow you to open, close, save and edit a fuzzy system using the five basic UI tools. From the menu, you can also:

∙ Set the format for the display by selecting **Options** > **Format**.

∙ Set the language by selecting **Options** > **Language**.

You can access information about the Rule Editor by clicking **Help** and close the UI using **Close**.

To insert the first rule in the Rule Editor, select the following:

∙ poor under the variable **service**

∙ rancid under the variable **food**

∙ The **or** radio button, in the **Connection** block

∙ cheap, under the output variable, **tip**.

Then, click **Add rule**.

The resulting rule is

*1. If (service is poor) or (food is rancid) then (tip is cheap) (1)*

The numbers in the parentheses represent weights.

Follow a similar procedure to insert the second and third rules in the Rule Editor to get

1. *If (service is poor) or (food is rancid) then (tip is cheap) (1)*

2. *If (service is good) then (tip is average) (1)*

3. *If (service is excellent) or (food is delicious) then (tip is generous) (1)*

The **Format** pop-up menu from the **Options** menu indicates that you are looking at the verbose form of the rules. Try changing it to symbolic. You will see

*1. (service==poor) | (food==rancid) => (tip=cheap) (1)*

*2. (service==good) => (tip=average) (1)*

*3. (service==excellent) | (food==delicious) => (tip=generous) (1)*

There is not much difference in the display really, but it is slightly more language neutral, because it does not depend on terms like *if* and *then*. If you change the format to indexed, you see an extremely compressed version of the rules.

*1 1, 1 (1) : 2*

*2 0, 2 (1) : 1*

*3 2, 3 (1) : 2*

This is the version of the rules that the machine deals with.

∙ The first column in this structure corresponds to the input variables.

∙ The second column corresponds to the output variable.

∙ The third column displays the weight applied to each rule.

∙ The fourth column is shorthand that indicates whether this is an OR (2) rule or an AND (1) rule.

∙ The numbers in the first two columns refer to the index number of the membership function.

A literal interpretation of rule 1 is "If input 1 is MF1 (the first membership function associated with input 1) or if input 2 is MF1, then output 1 should be MF1 (the first membership function associated with output 1) with the weight 1."

The symbolic format does not consider the terms, *if*, *then*, and so on. The indexed format doesn't even bother with the names of your variables. Obviously the functionality of your system doesn't depend on how well you have named your variables and membership functions. The whole point of naming variables descriptively is, as always, making the system easier for you to interpret. Thus, unless you have some special purpose in mind, it is probably be easier for you to continue with the **verbose** format.

At this point, the fuzzy inference system has been completely defined, in that the variables, membership functions, and the rules necessary to calculate tips are in place. Now, look at the fuzzy inference diagram presented at the end of the previous section and verify that everything is behaving the way you think it should. You can use the Rule Viewer, the next of the UI tools we'll look at. From the **View** menu, select **Rules**.

**The Rule Viewer**

****

The Rule Viewer displays a roadmap of the whole fuzzy inference process. It is based on the fuzzy inference diagram described in the previous section. You see a single figure window with 10 plots nested in it. The three plots across the top of the figure represent the antecedent and consequent of the first rule. Each rule is a row of plots, and each column is a variable. The rule numbers are displayed on the left of each row. You can click on a rule number to view the rule in the status line.

∙ The first two columns of plots (the six yellow plots) show the membership functions referenced by the antecedent, or the if-part of each rule.

∙ The third column of plots (the three blue plots) shows the membership functions referenced by the consequent, or the then-part of each rule.

Notice that under **food**, there is a plot which is blank. This corresponds to the characterization of none for the variable **food** in the second rule.

∙ The fourth plot in the third column of plots represents the aggregate weighted decision for the given inference system. This decision will depend on the input values for the system. The defuzzified output is displayed as a bold vertical line on this plot. 

Upon opening the Surface Viewer, you see a three-dimensional curve that represents the mapping from food and service quality to tip amount. Because this curve represents a two-input one-output case, you can see the entire mapping in one plot. When we move beyond three dimensions overall, we start to encounter trouble displaying the results.

Accordingly, the Surface Viewer is equipped with drop-down menus **X (input)**, **Y (input)** and **Z (output)** that let you select any two inputs and any one output for plotting. Below these menus are two input fields **X grids** and **Y grids** that let you specify how many x-axis and y-axis grid lines you want to include. This capability allows you to keep the calculation time reasonable for complex problems.

By default, the surface plot updates automatically when you change the input or output variable selections or the number of grid points. To disable automatic plot updates, in the **Options** menu, clear the **Always evaluate** option. When this option is disabled, to update the plot, click **Evaluate**.

If you want to create a smoother plot, use the **Plot points** field to specify the number of points on which the membership functions are evaluated in the input or output range. This field defaults to the minimum number of plot plots, 101. If you specify fewer plot points, the field value automatically resets to 101. When you specify the number of plot points, the surface plot automatically updates.

By clicking on the plot axes and dragging the mouse, you can manipulate the surface so that you can view it from different angles.

The **Ref. Input** field is used in situations when there are more inputs required by the system than the surface is mapping. You can edit this field to explicitly set inputs not specified in the surface plot.

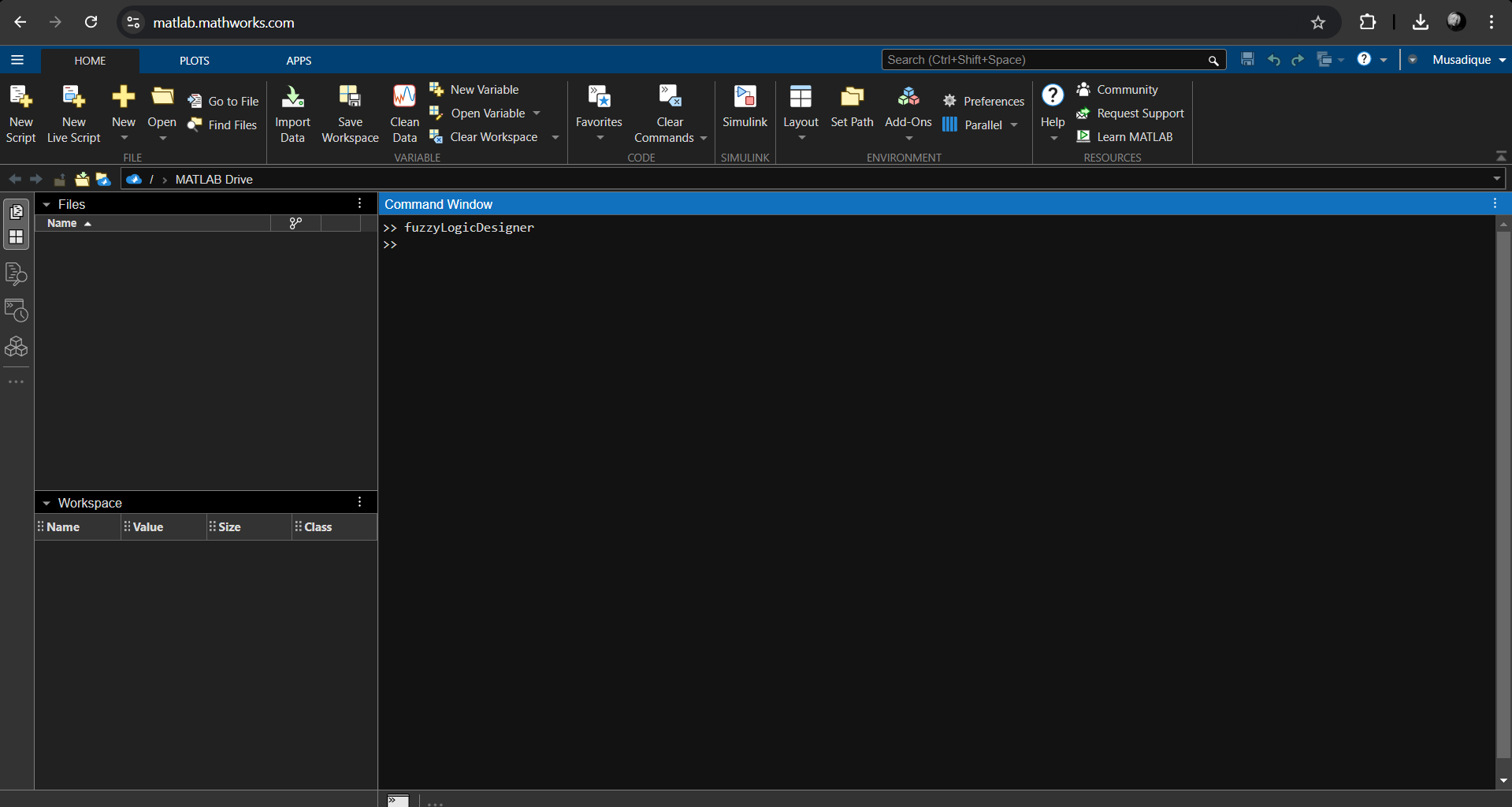
Suppose you have a four-input one-output system and would like to see the output surface. The Surface Viewer can generate a three dimensional output surface where any two of the inputs vary, but two of the inputs must be held constant because computer monitors cannot display a five-dimensional shape. In such a case, the input is a four-dimensional vector with NaNs holding the place of the varyinginputs while numerical values indicates those values that remain fixed.

The menu items allow you to open, close, save and edit a fuzzy system using the five basic UI tools. You can access information about the Surface Viewer by clicking **Help** and close the UI using **Close**.

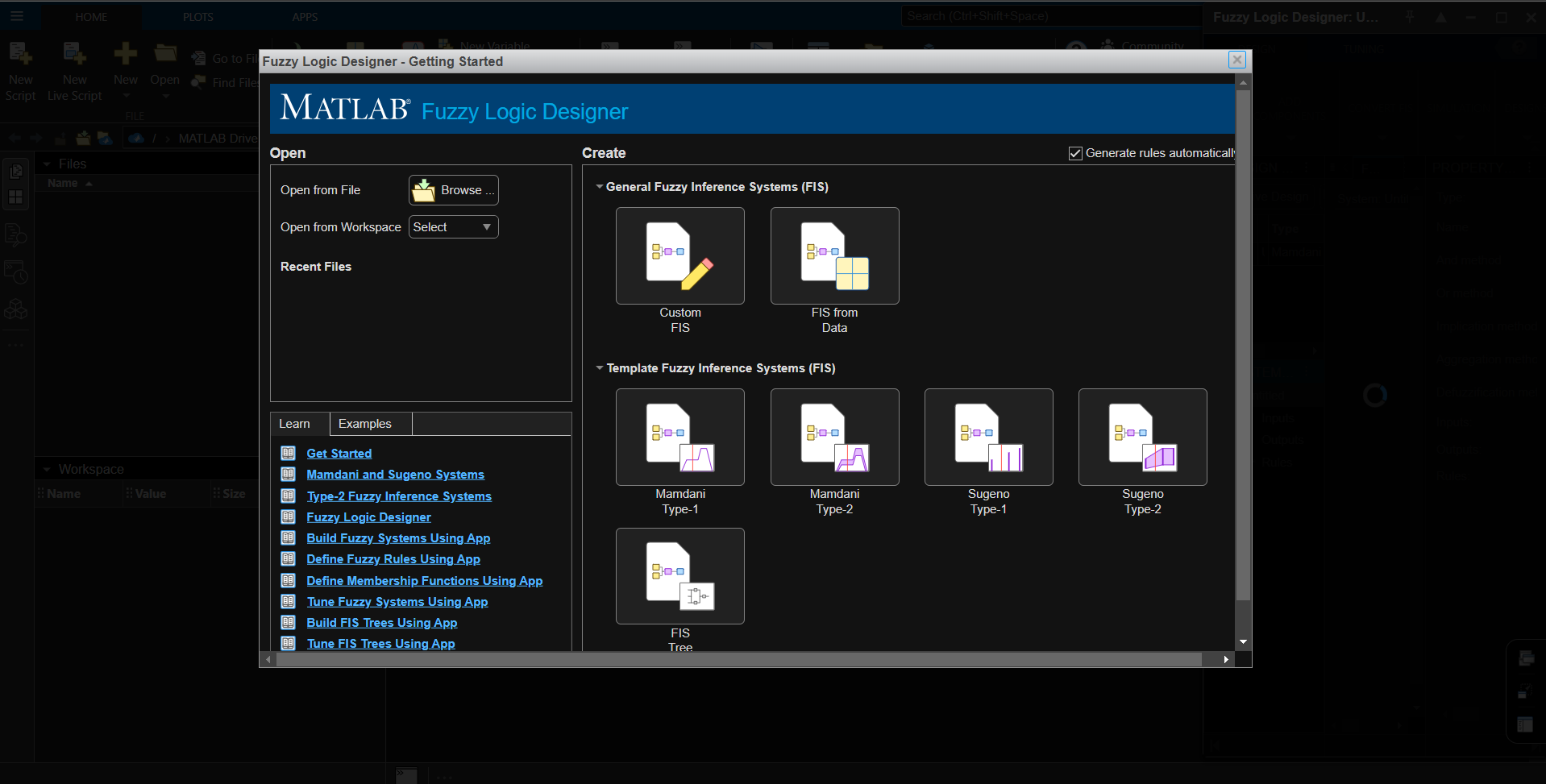
Tasks:

Create FIS through MATLAB and attach the screen shot of surface viewer after creating and evaluating the rules.

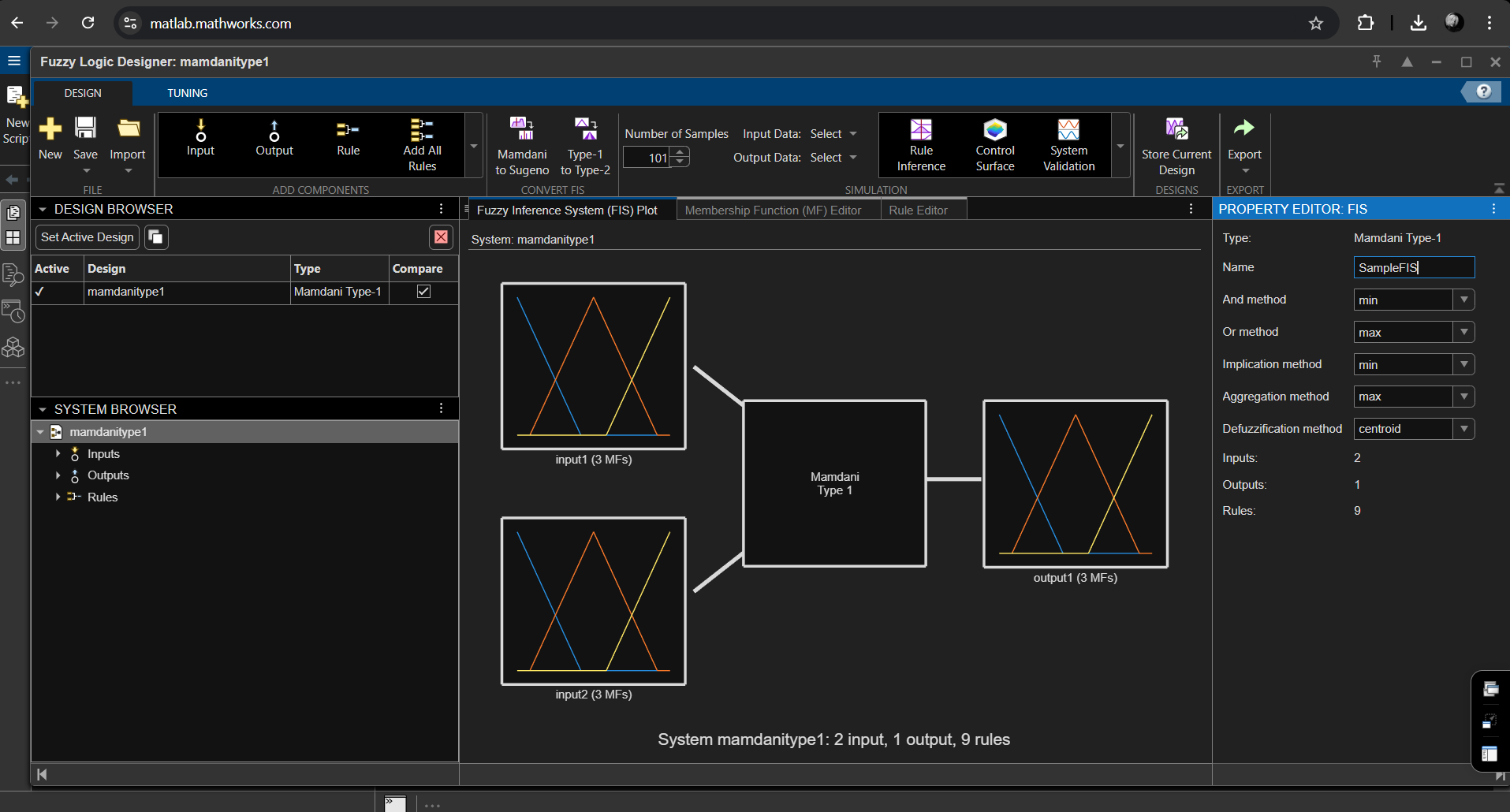
**Screenshots of FIS Implementation Through MATLAB**

****

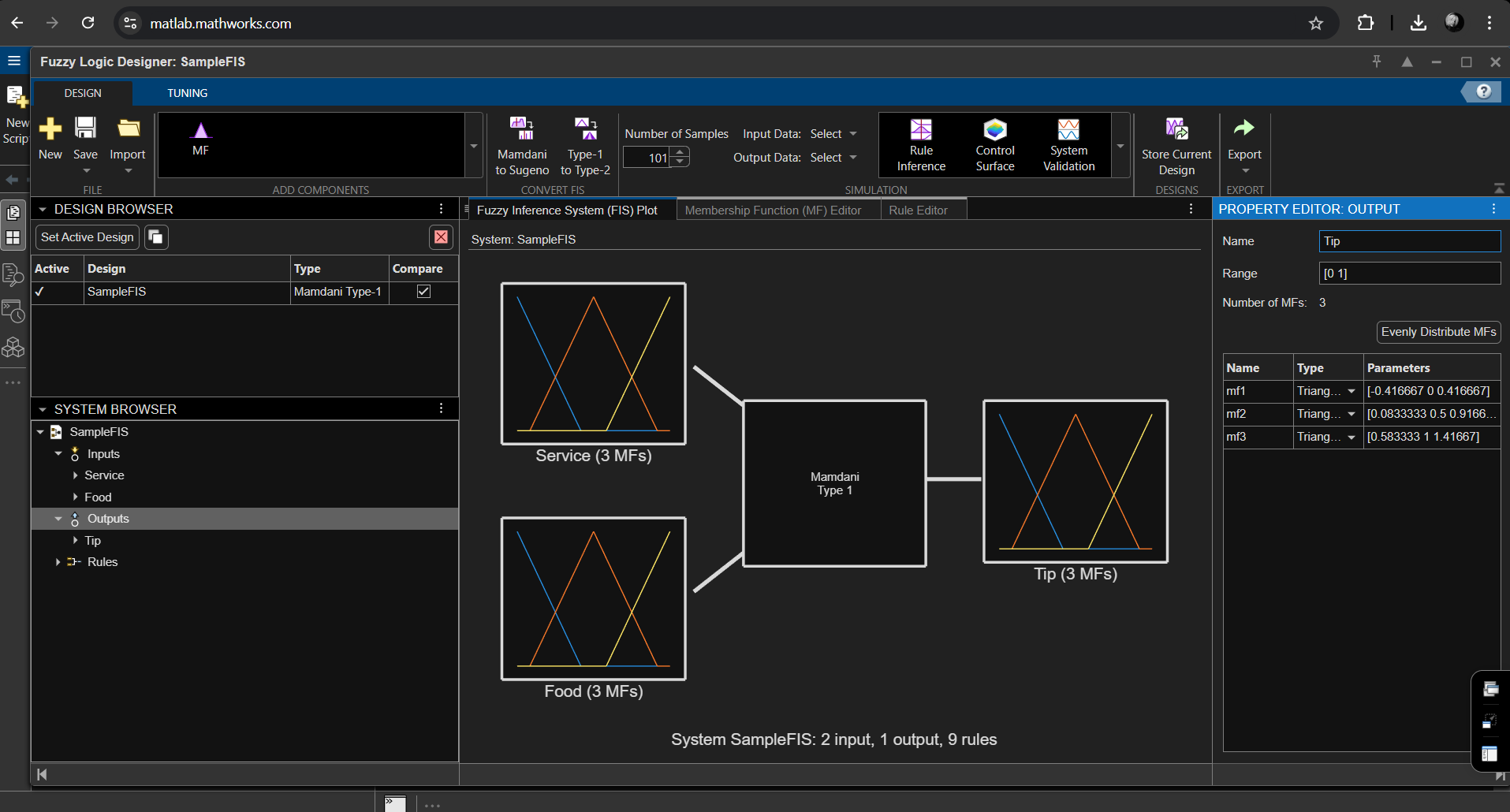
**Select the Mamdani Type-1**

****

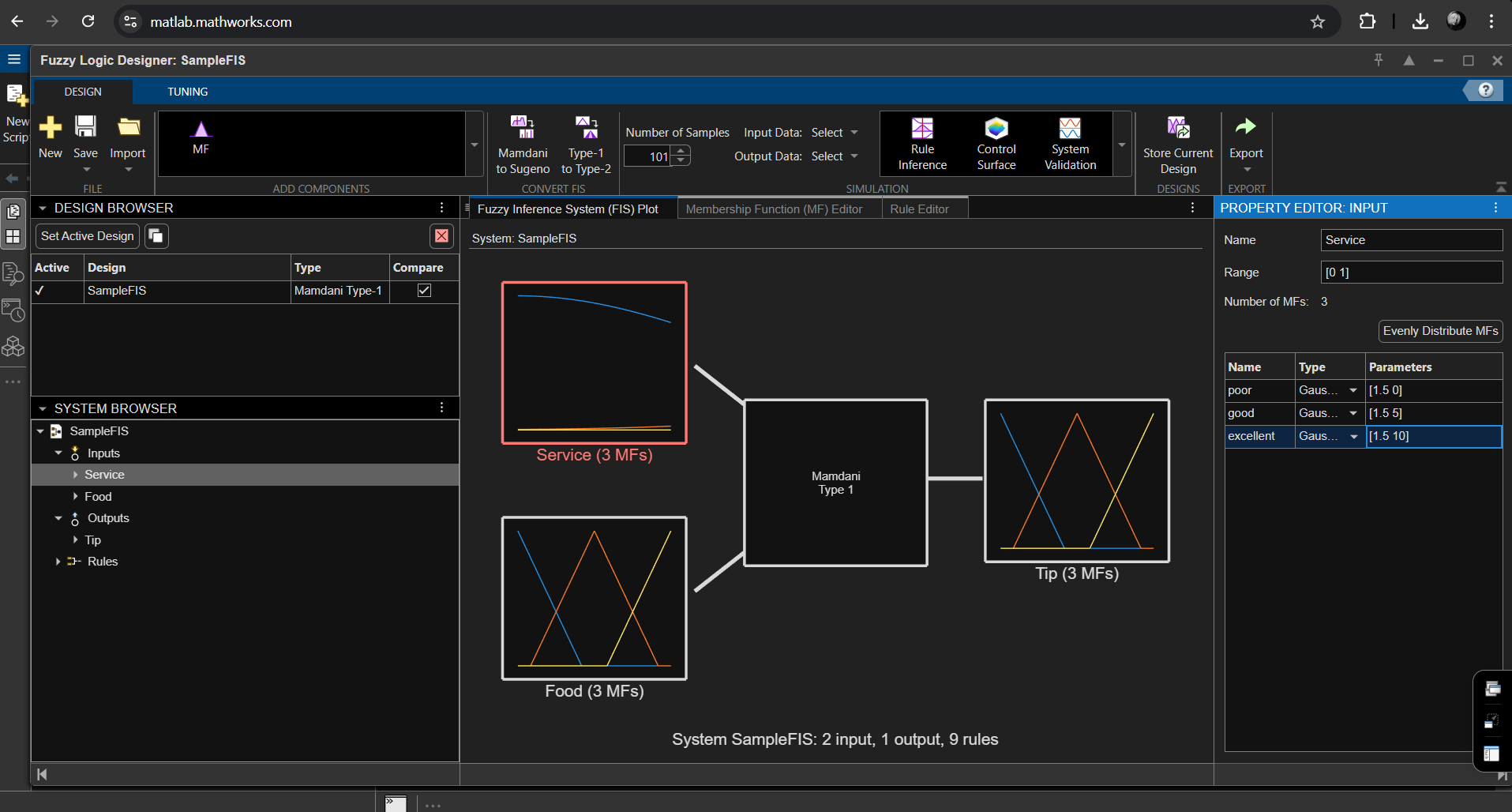
**The Mamdani Interfaces popped up**

****

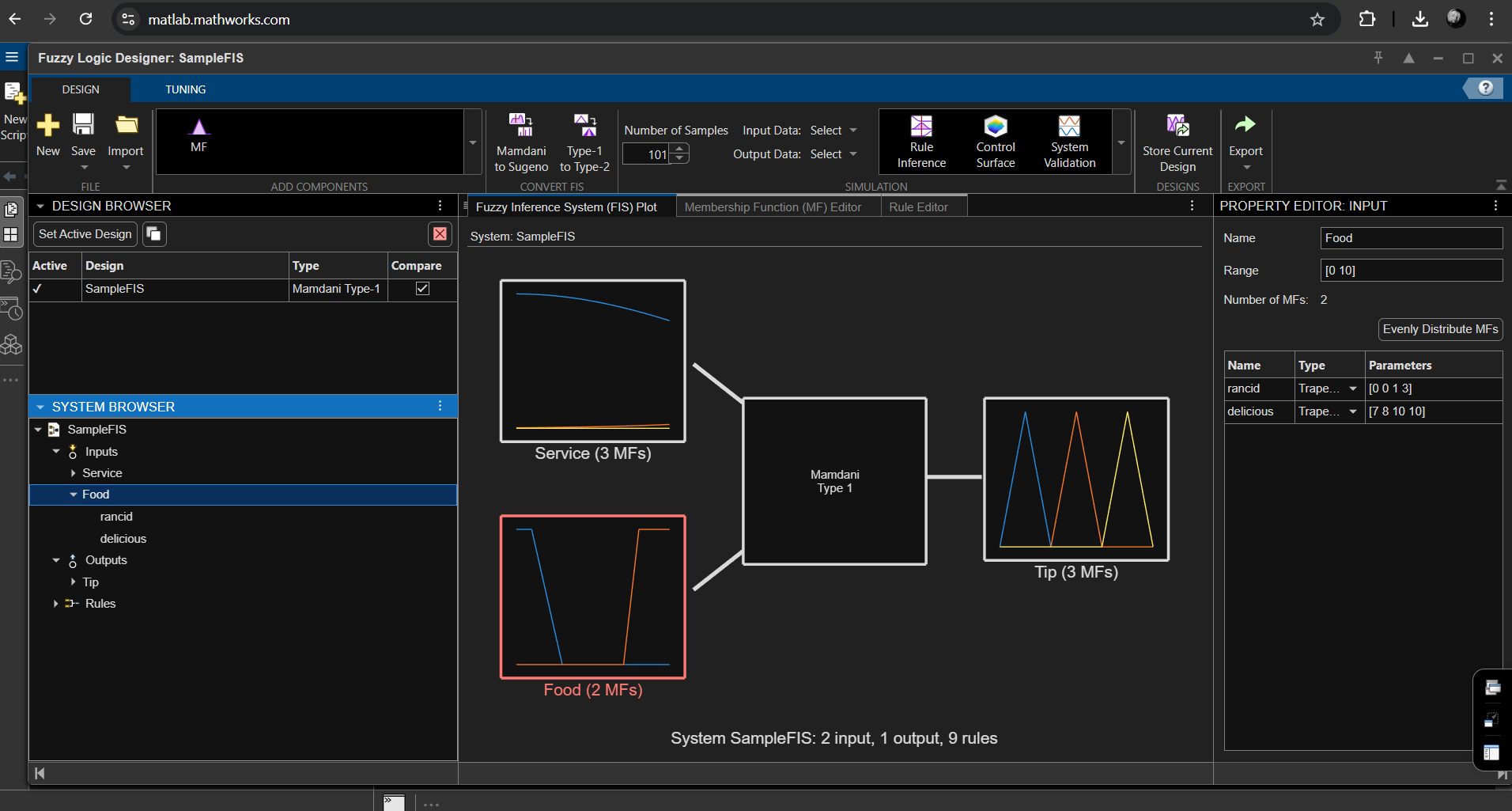
**Change the name of inputs into Service and Food and output to Tip**

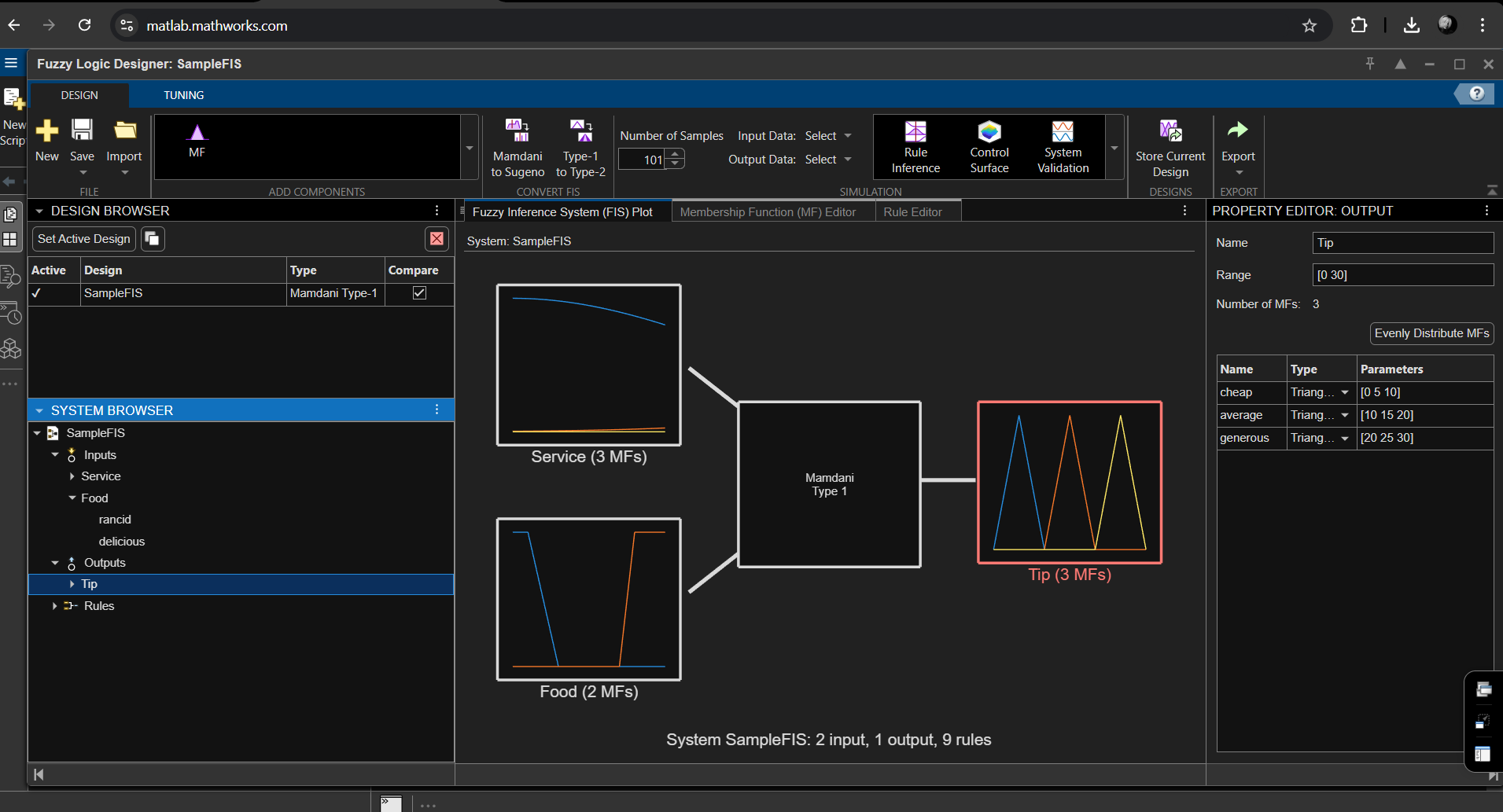
****

**Change the types into Gaussian and Change their parameters into [1.5 0] [1.5 5] [1.5 10]**

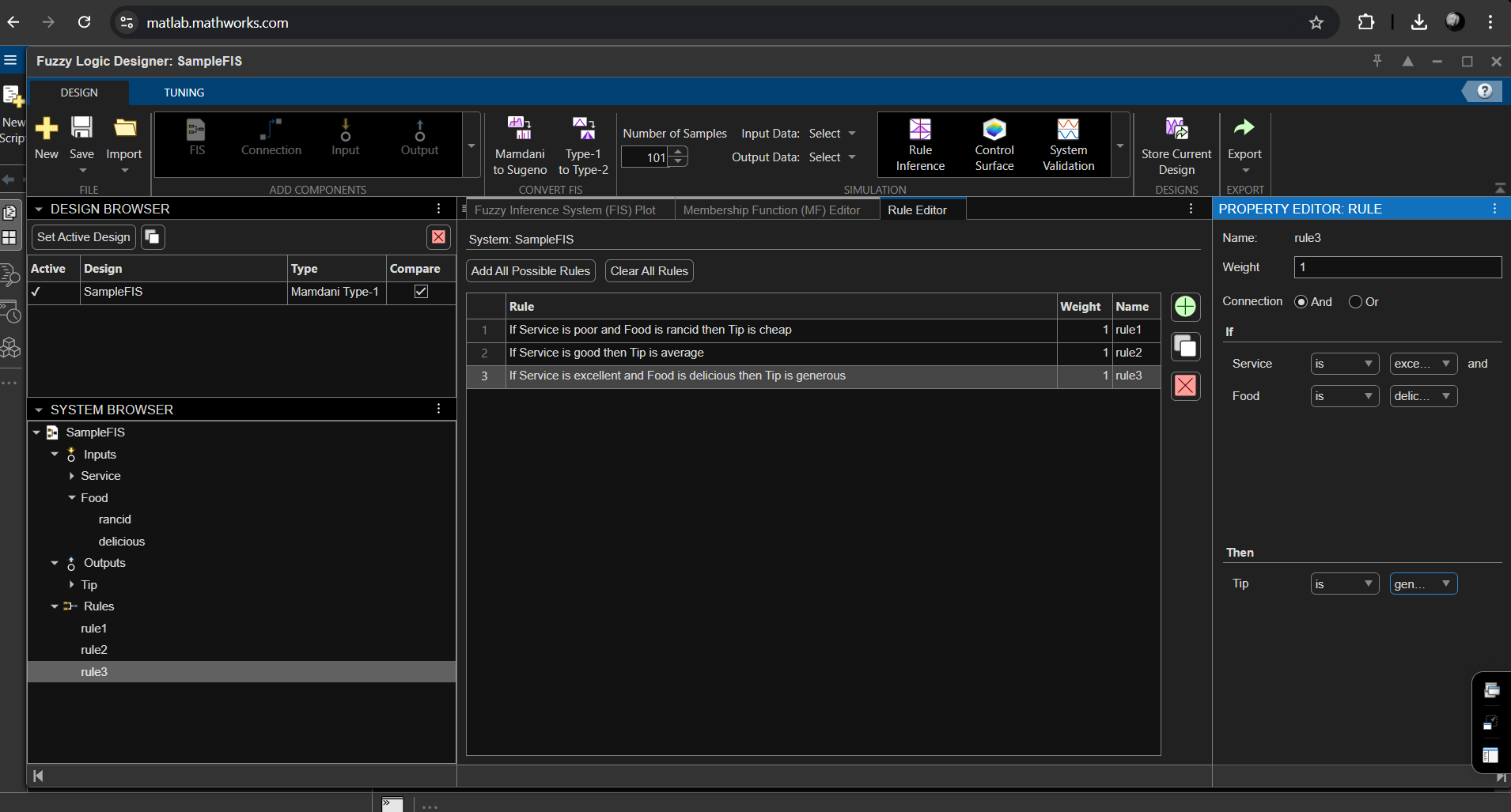
****

**Apply same steps for Food and Tip too**

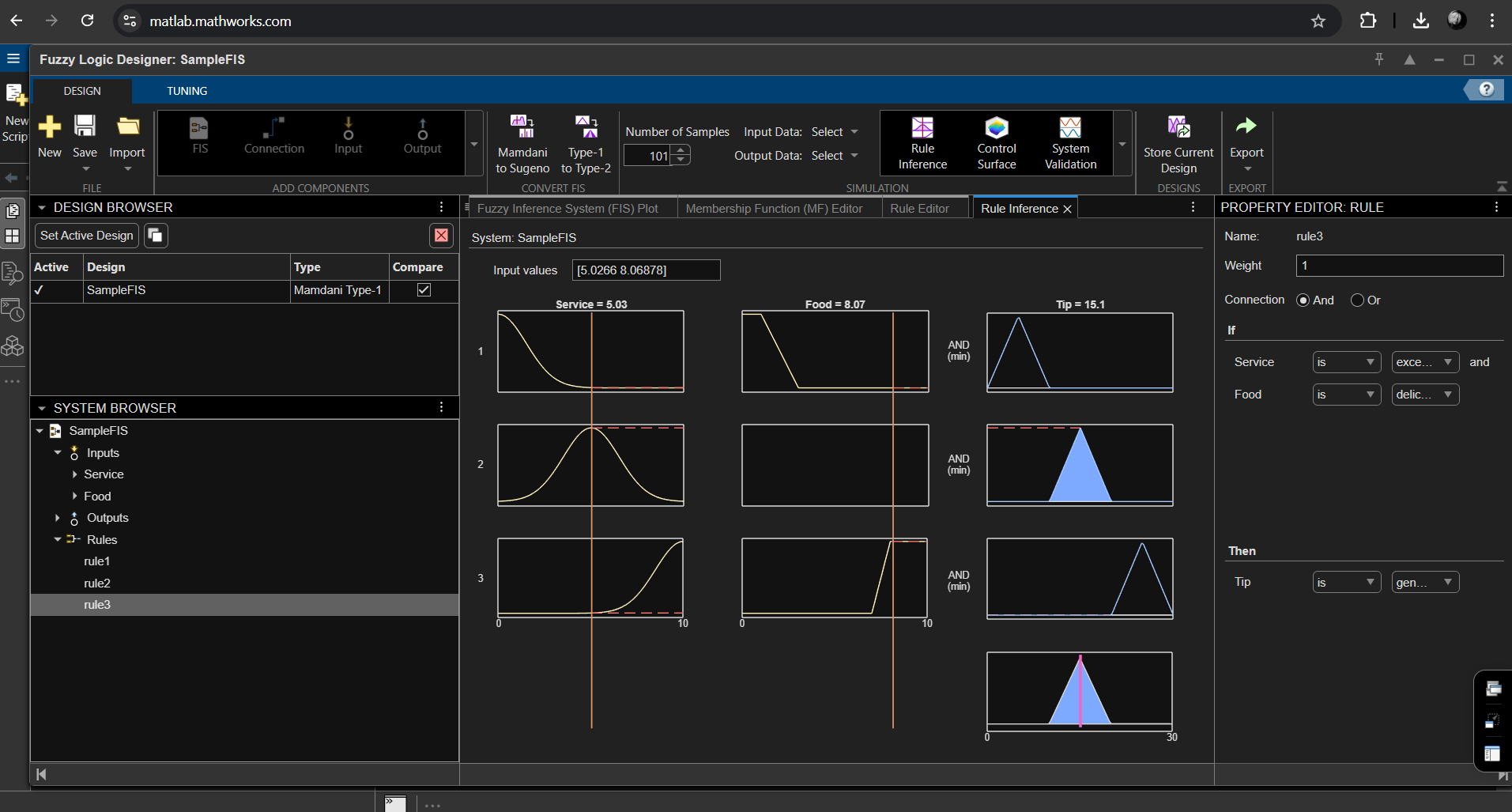
****

****

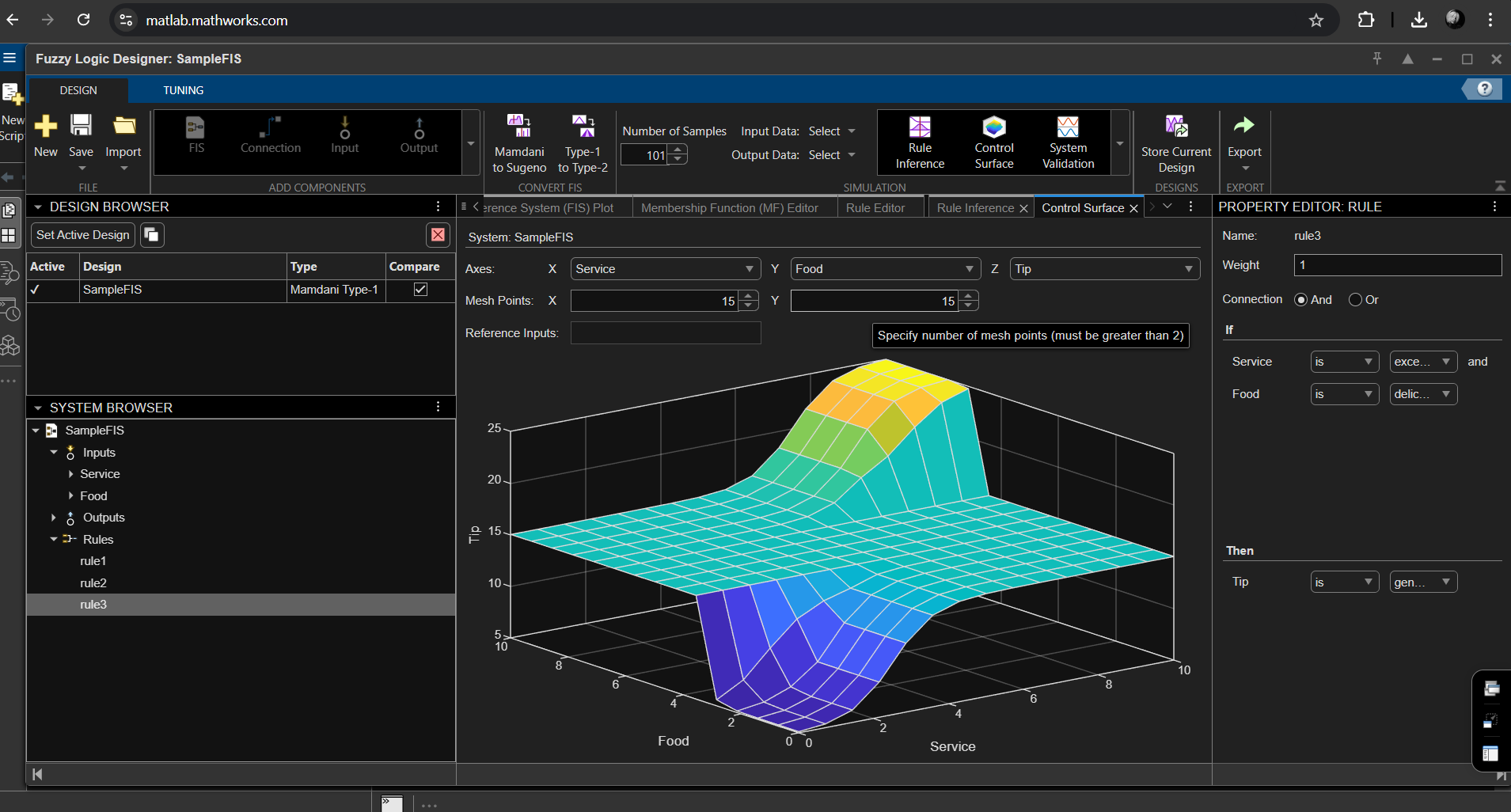
**Open the Rules Editor Open and Clear all the rules so we can create new rules**

****

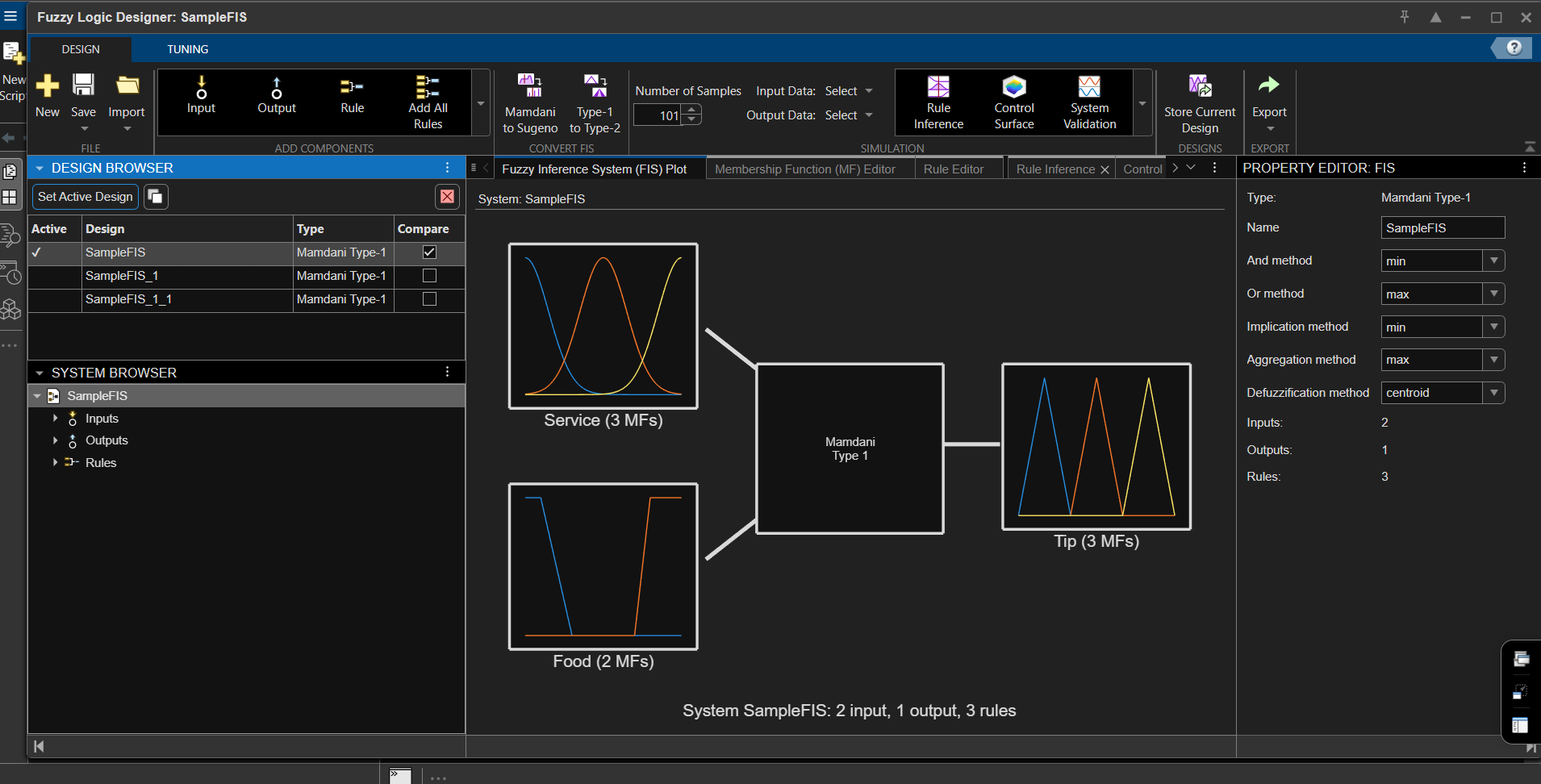
**This is the Rule inference system where we check our fuzzy system works or not**

****

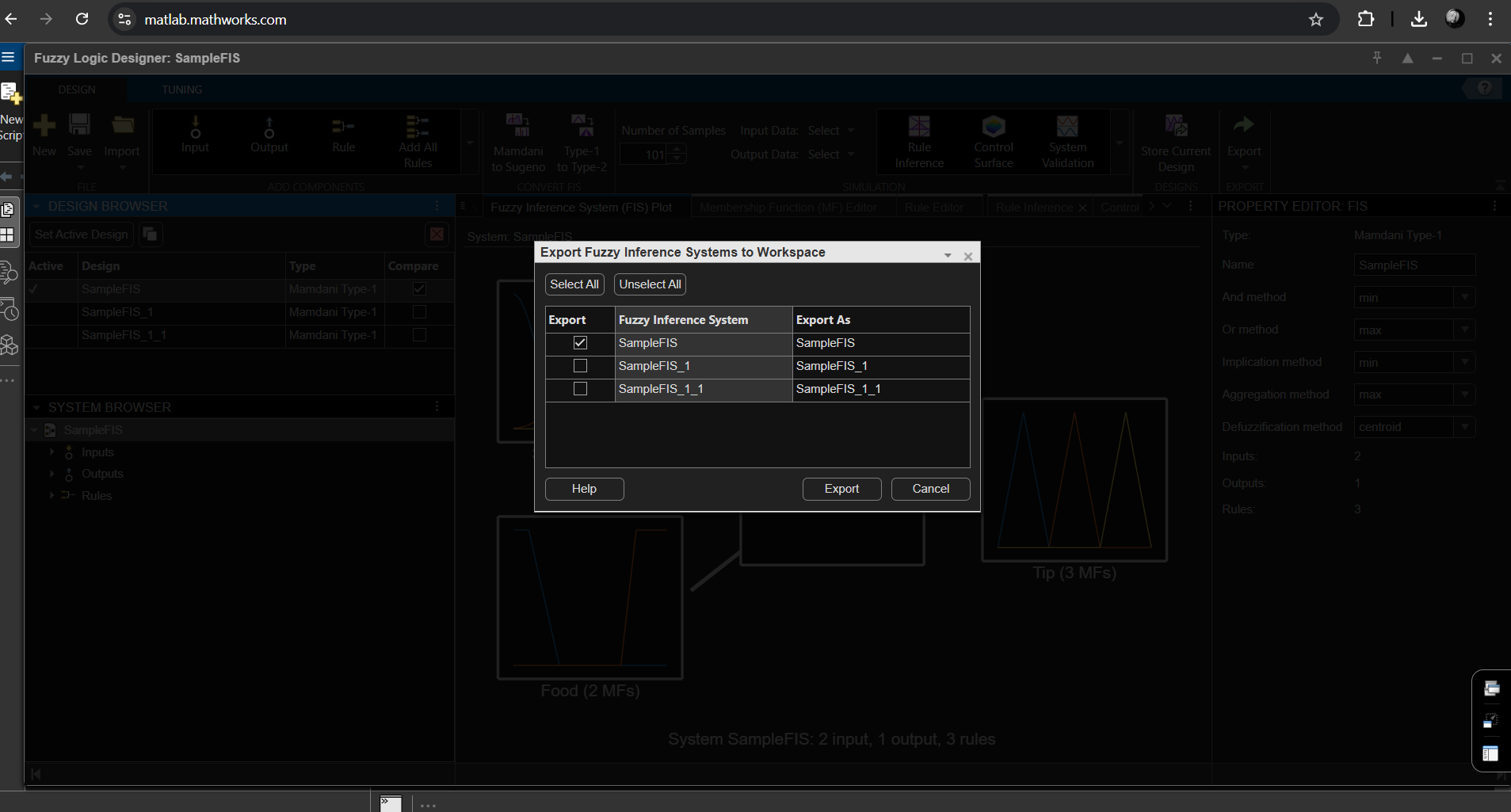
**In the control Surface it gives you an overview of your fuzzy design to check the calculations are accurate**

****

**We can create the multiple designs for comparing**

****

**We can export our fuzzy system work and use it again for the future**

****