The Solution of DFS revolves around the pseudo-code and the actual programming code:

The pseudo-code of DFS is given as:

Graph Traversal Universal Code

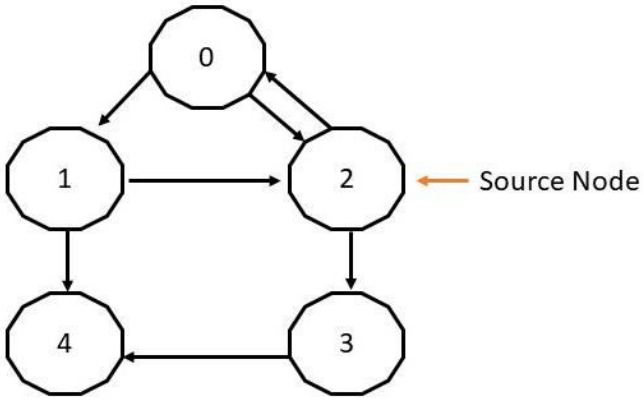
```
DFS(G, u)
      u.visited = true
      for each v ∈ G.Adj[u]
            if v.visited == false
                  DFS(G,v)

init() {
      For each u ∈ G
            u.visited = false
       For each u ∈ G
          DFS(G, u)
}
```

Lab 2

Topic: Simulate DFS

Problem Statement: You are required to simulate DFS technique with the help of following stimulator.

Download simulator from the Link **https://visualgo.net/en/dfsbfs**



DFS(u)

for each neighbor v of u

if v is unvisited, tree edge, DFS(v)

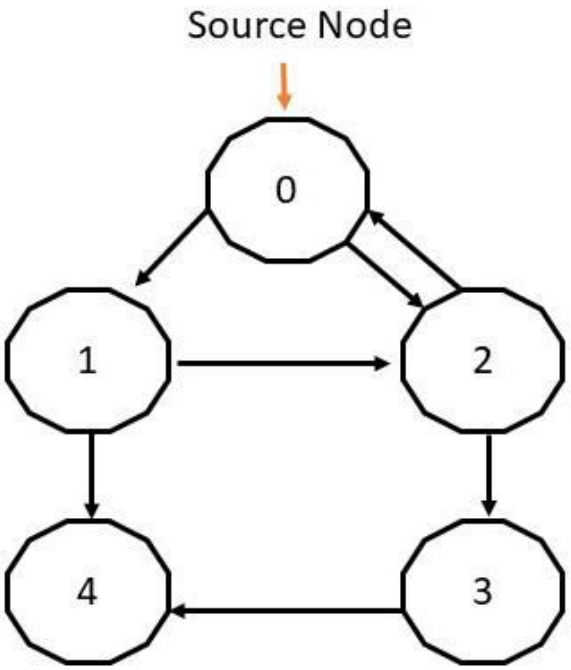
else if v is explored, bidirectional/back edge

else if v is visited, forward/cross edge

Lab 3

Topic: Breadth First Search (BFS)

Problem Statement: You are required to code breadth first strategy for the following directed graph in C/C++. Your program is capable to input source vertex.



```
// Program to print BFS traversal from a given
// source vertex. BFS(int s) traverses vertices
// reachable from s.
#include<bits/stdc++.h>
using namespace std;

// This class represents a directed graph using
// adjacency list representation
class Graph
{
    int V;    // No. of vertices

    // Pointer to an array containing adjacency
    // lists
    vector<list<int>> adj;
public:
    Graph(int V);  // Constructor

    // function to add an edge to graph
    void addEdge(int v, int w);

    // prints BFS traversal from a given source s
    void BFS(int s);
};

Graph::Graph(int V)
{
    this->V = V;
```

```
    adj.resize(V);
}

void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w); // Add w to v's list.
}

void Graph::BFS(int s)
{
    // Mark all the vertices as not visited
    vector<bool> visited;
    visited.resize(V,false);

    // Create a queue for BFS
    list<int> queue;

    // Mark the current node as visited and enqueue it
    visited[s] = true;
    queue.push_back(s);

    while(!queue.empty())
    {
        // Dequeue a vertex from queue and print it
        s = queue.front();
        cout << s << " ";
        queue.pop_front();

        // Get all adjacent vertices of the dequeued
```

```
        // vertex s. If a adjacent has not been visited,
        // then mark it visited and enqueue it
        for (auto adjecent: adj[s])
        {
            if (!visited[adjecent])
            {
                visited[adjecent] = true;
                queue.push_back(adjecent);
            }
        }
    }
}

// Driver program to test methods of graph class
int main()
{
    // Create a graph given in the above diagram
    Graph g(4);
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(3, 3);

    cout << "Following is Breadth First Traversal "
         << "(starting from vertex 2) \n";
    g.BFS(2);
```

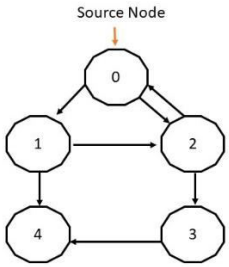
```
    return 0;
}
```

Lab 4

Topic: Breadth First Search through simulator

You are required to stimulate BFS technique with the help of following stimulator.

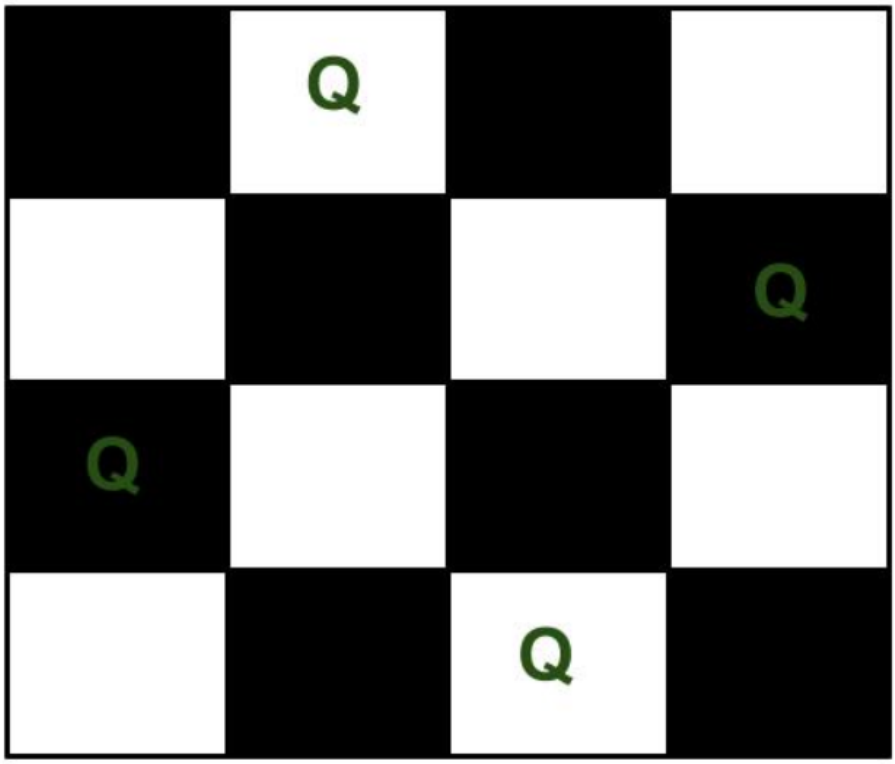
https://visualgo.net/en/dfsbfs



Lab 5

Topic: N-Queen Problem

Problem Statement: You are required to code 04 N Queen Problem using using C/C++



Lab 6

Topic: Simulate N Queen Problem

Problem statement: You are required to stimulate n-queen technique with the help of following stimulator. You are initially required to take 4x4 matric to solve n-queen problem

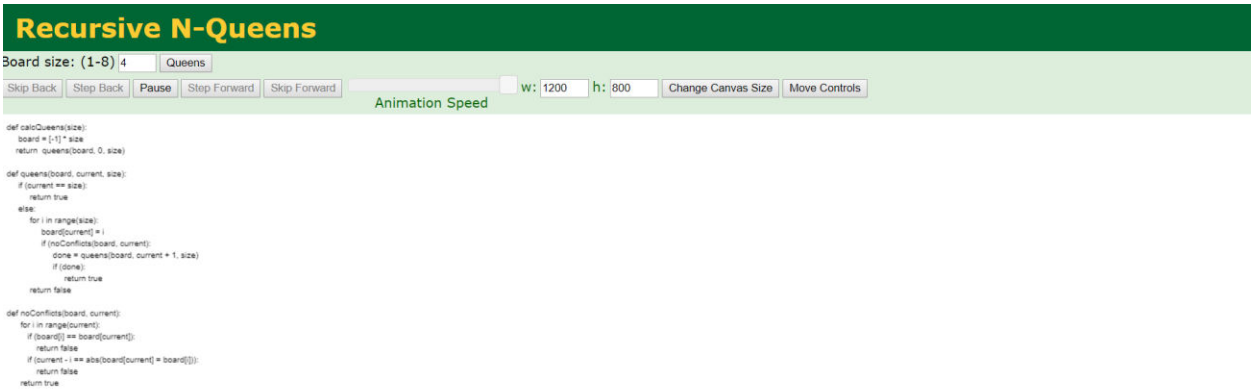
https://www.cs.usfca.edu/~galles/visualization/RecQueens.html

		Q	
Q			
			Q
	Q		

Solution:

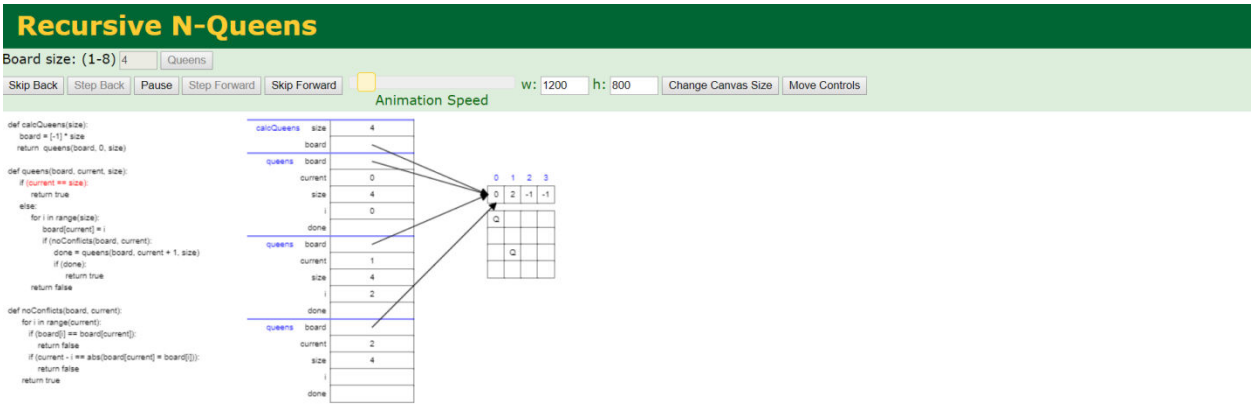
Step 1:

First you enter the queen Size(1-8)

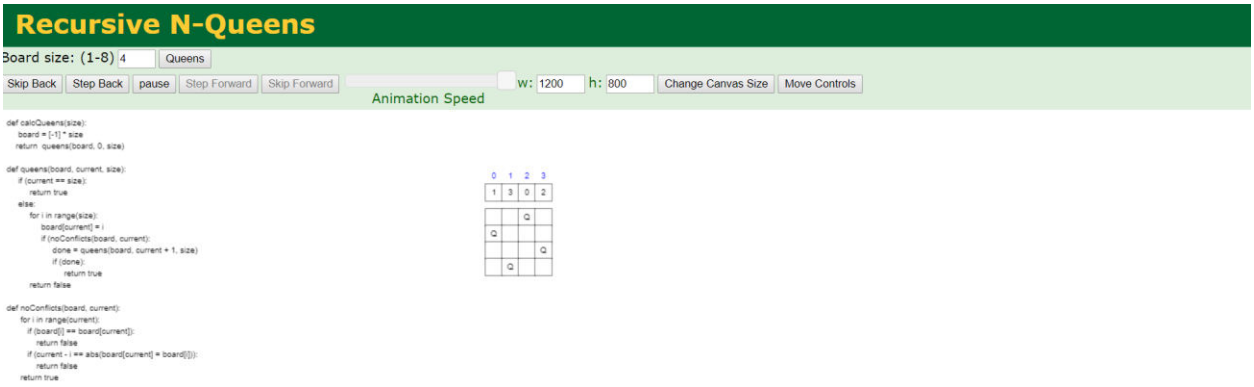


Step 2:

In Step 2, you will enter board size i.e, 4



Step 3: After all simulations the final result will be



Lab 7

Topic: CLIPS Installation Guide

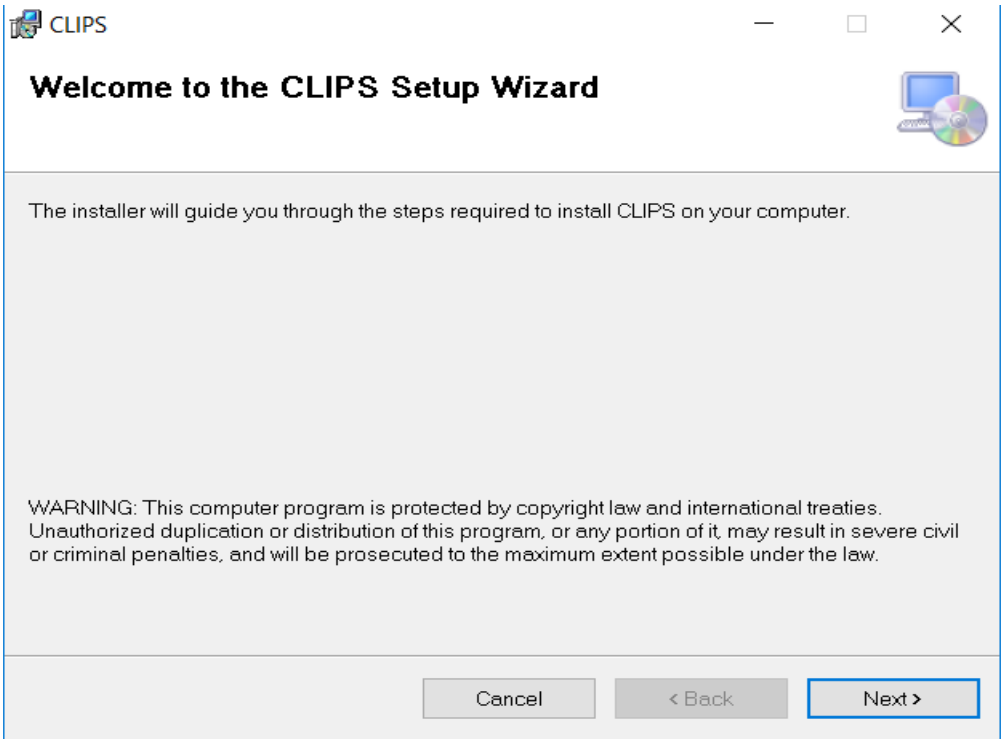
Problem Statement: You are required to install CLIPS from the following link.

Download link:

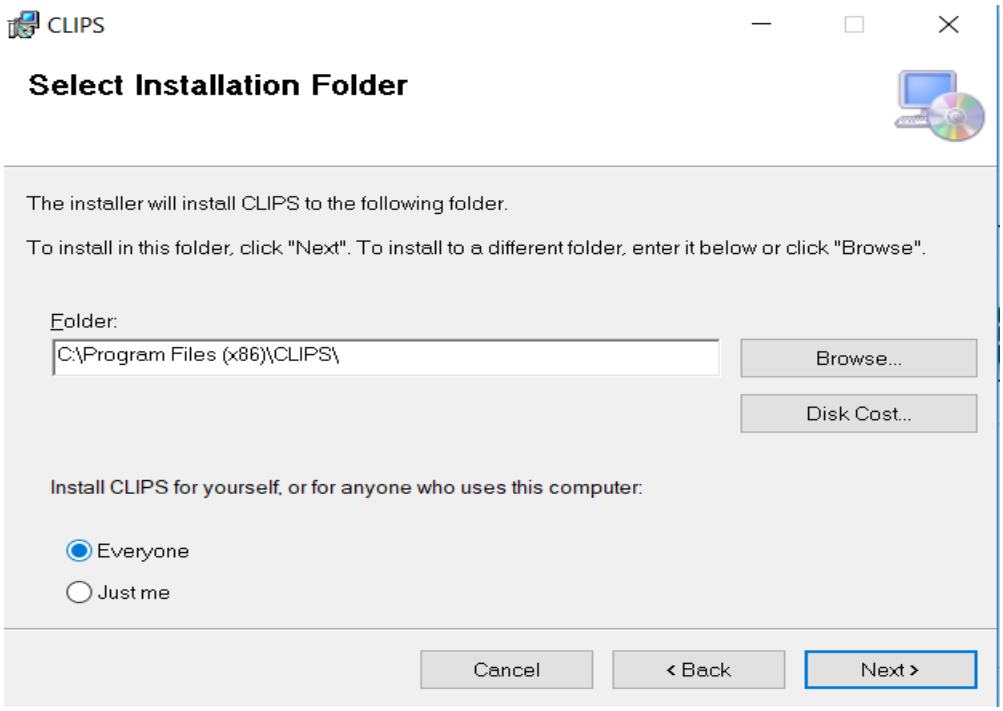
clipsrules.sourceforge.net/

Solution:

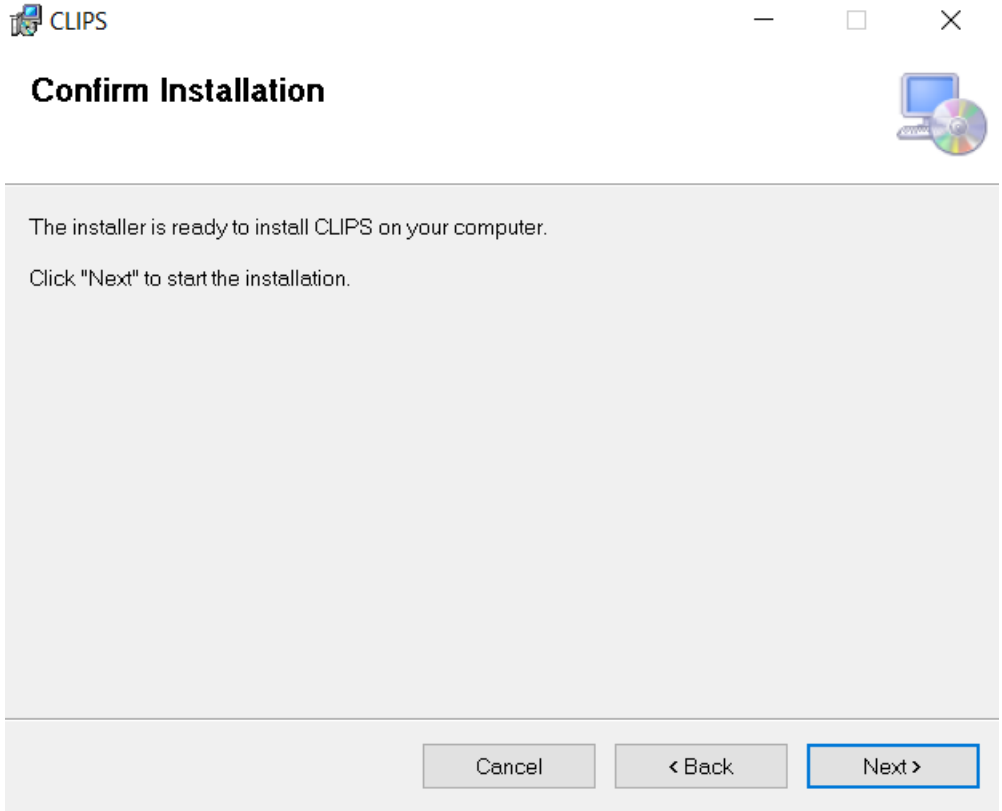
Step:1



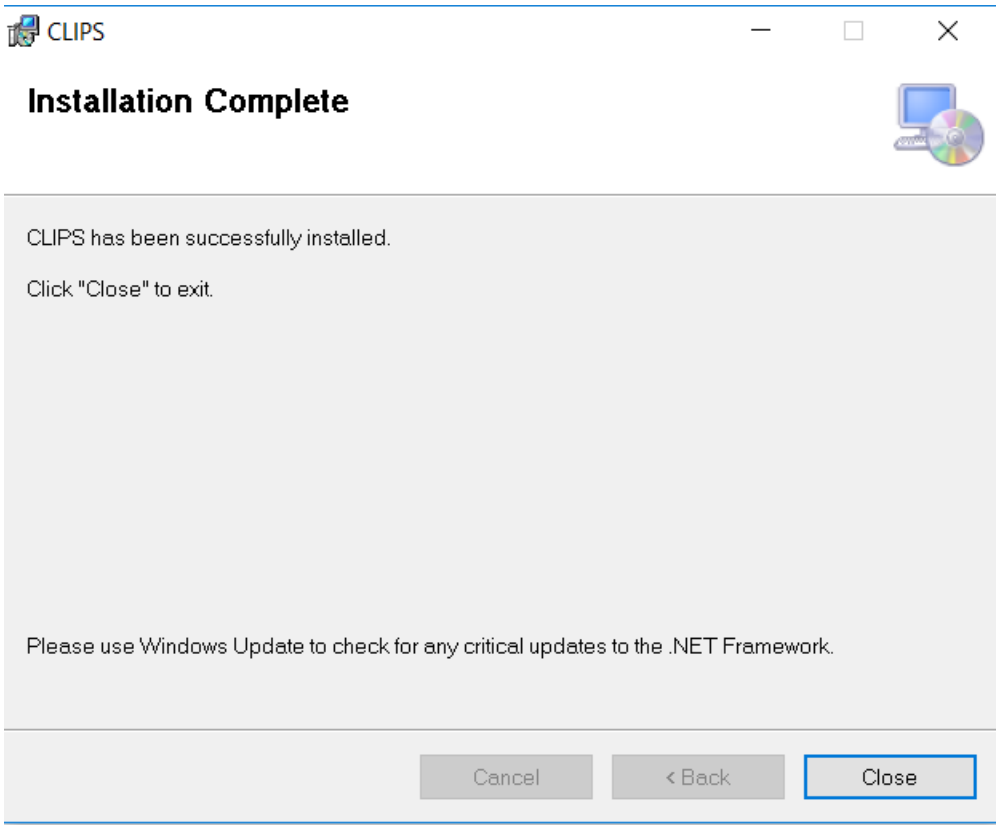
Step 2:


CLIPS
Select Installation Folder
The installer will install CLIPS to the following folder.
To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".
Folder:
C:\Program Files (x86)\CLIPS\
Browse...
Disk Cost...
Install CLIPS for yourself, or for anyone who uses this computer:
Everyone
Just me
Cancel
< Back
Next >

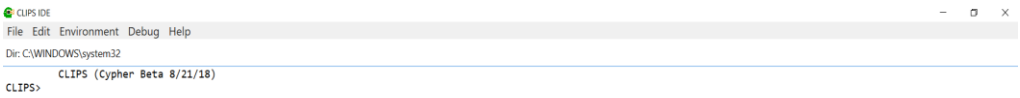
Step 3:


CLIPS
Confirm Installation
The installer is ready to install CLIPS on your computer.
Click "Next" to start the installation.
Cancel
< Back
Next >

Step 4


CLIPS
Installation Complete
CLIPS has been successfully installed.
Click "Close" to exit.
Please use Windows Update to check for any critical updates to the .NET Framework.
Cancel
< Back
Close

Step 6:


CLIPS IDE
File Edit Environment Debug Help
Dir: C:\WINDOWS\system32
CLIPS (Cypher Beta 8/21/18)
CLIPS>

Lab 8

Topic: Introduction to CLIPS

Problem Statement: Procedure of using CLIPS:

Solution:

To start CLIPS, double-click on the CLIPSWin.exe file. You'll get a window with nothing in it but the command prompt CLIPS>. For the time being, this is where you type in your commands and programs. To exit CLIPS, type **(exit)** or shut down the program like any other windows application. Note that CLIPS commands are always encased in brackets thus: **(assert (foo))**. Here is a list of some important commands:

(exit)	Shuts down CLIPS
(clear)	Removes all rules and facts from memory. Equivalent to shutting down and restarting CLIPS.
(reset)	Removes facts information from memory (but not rules) and resets the agenda.
(run)	Starts executing a CLIPS program.

The above commands can also be executed from the CLIPS menu bar.

Data Types

- Integer
- Float
- Symbol
- String
- External-address
- Fact-address
- Instance-name
- Instance-address

Numeric information can be represented using floats and integers. Symbolic information can be represented using symbols and strings. A number consists

only of digits (0-9), a decimal point (.), a sign (+ or -), and, optionally, an (e) for exponential notation with its corresponding sign.

A number is either stored as a float or an integer. Any number consisting of an optional sign followed by only digits is stored as an integer (represented internally by CLIPS as a C long integer). All other numbers are stored as floats (represented internally by CLIPS as a C double-precision float).

- Some examples of integers are
- 237 15 +12 -32
- Some examples of floats are
- 237e3 15.09 +12.0 -32.3e-7
- A symbol in CLIPS is any sequence of characters that starts with any printable ASCII character and is followed by zero or more printable ASCII characters. When a delimiter is found, the symbol is ended. The following characters act as delimiters: any non-printable ASCII character (including spaces, tabs, carriage returns, and line feeds), a double quote, opening and closing parentheses “(” and “)”, an ampersand “&”, a vertical bar “|”, a less than “<”, and a tilde “~”.
- A semicolon “;” starts a CLIPS comment and also acts as a delimiter.
- Delimiters may not be included in symbols with the exception of the “<“ character which may be the first character in a symbol. In addition, a symbol may not begin with either the “?” character or the “$?” sequence of characters (although a symbol may contain these characters). These characters are reserved for variables
- CLIPS is case sensitive (i.e. uppercase letters will match only uppercase letters). Note that numbers are a special case of symbols
- Some simple examples of symbols are
- foo Hello B76-HI bad_value
- 127A 456-93-039 @+=-% 2each

  String:

- A string is a set of characters that starts with a double quote (") and is followed by zero or more printable characters.

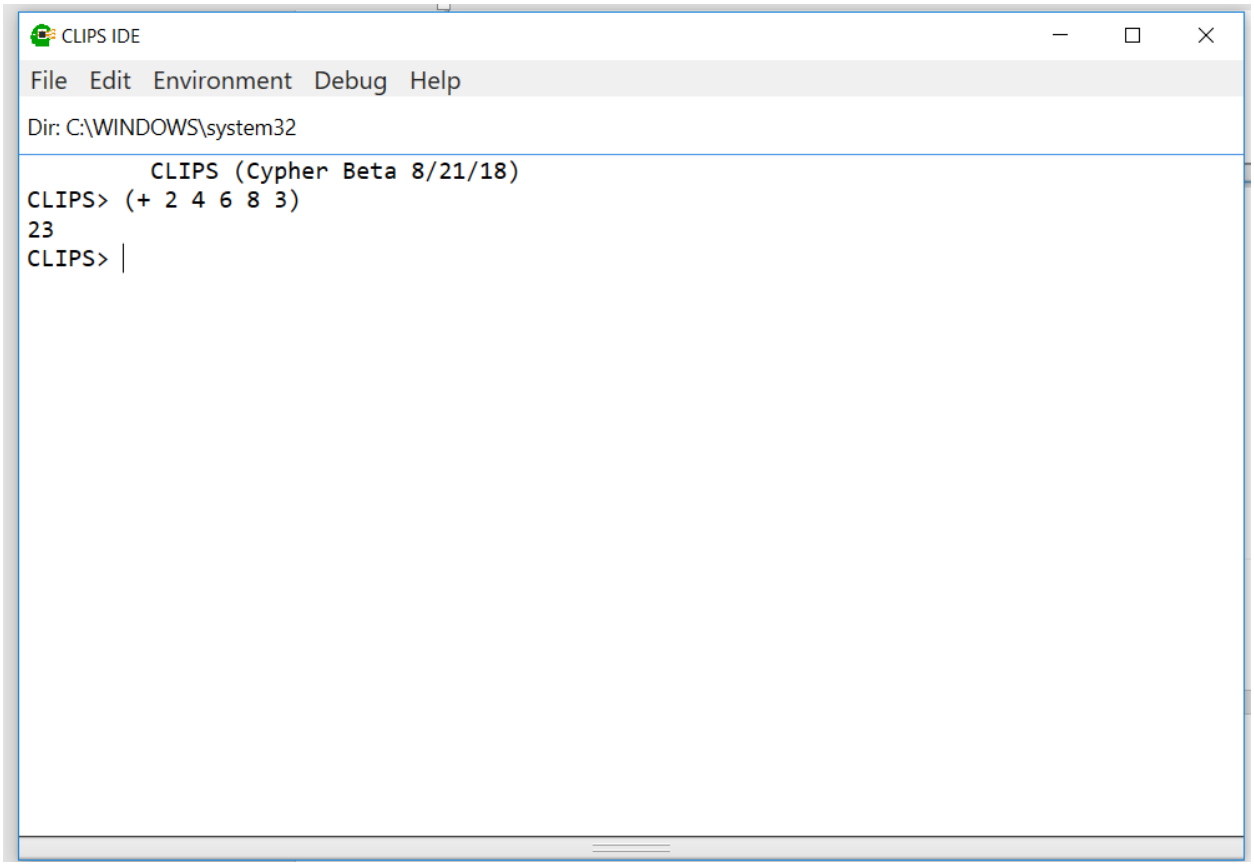
- A string ends with double quotes. Double quotes may be embedded within a string by placing a backslash (\) in front of the character.
- A backslash may be embedded by placing two consecutive backslash characters in the string.
- Some examples are
    - "foo" "a and b" "1 number" "a\"quote"

Example 1:

Write a code that enter two integer values and add them?

Example 2:

Write a code that adds five values?



Lab 9

Topic: Function in clips

Problem statement: use given below function using CLIPS?

Write code for the following:

- To add 3,4,5
- To multiply 5,6.0,2
- (+3 (*8 9)4
- (* 8 (+ 3 (* 2 3 4) 9) (* 3 4))
- (+ 2 3(- 10 5))
- (+ 2 3(- 5 10))
- (+ 2 3(- 5 15))

Lab 10

Topic: Insert Facts in CLIPS
Problem Statement: Using CLIPS you required to put data in the fact-list, you can use the **assert** command. For this purpose. You can insert any fact in fact list.

Solution:

CLIPS> (assert (duck))

<Fact-1>

CLIPS> (facts)

f-0 (initial-fact)

f-1 (duck)

For a total of 2 facts.

CLIPS>

- What happens if you try to put a second duck into the fact-list?
- Let's try it and see. Assert a new (duck), then issue a (facts) command as follows

FACTS COMMANDS:

- The keyboard command to see facts is with the facts command. Enter (facts) in response to the CLIPS prompt and CLIPS will respond with a list of facts in the fact-list.
- Be sure to put parentheses around the command or CLIPS will not accept it. The result of the (facts) command in this example should be

CLIPS> (facts)

f-0 (initial-fact)

f-1 (duck)

For a total of 2 facts.

CLIPS

Lab 11

Topic: Clearing Up The Facts:

Problem Statement: Using clips clear command actually does more than just remove facts. Besides removing all the facts, (clear) also removes all the rules.

Solution:

CLIPS> (facts)

f-0 (initial-fact)

f-1 (duck)

f-2 (quack)

For a total of 3 facts.

CLIPS> (clear)

CLIPS>

- CLIPS> (clear)
- CLIPS> (assert (a) (b) (c))
- <Fact-3>
- CLIPS> (facts)
- f-0 (initial-fact)
- f-1 (a)
- f-2 (b)
- f-3 (c)
- For a total of 4 facts.
- Removing facts from the fact-list is called *retraction and is done with the retract* command.
- To retract a fact, you must specify the fact-index.

CLIPS> (retract 3)

CLIPS>

- You can also retract multiple facts at once, as shown by the following.

CLIPS> (retract 1 3)

CLIPS> (facts)

f-0 (initial-fact)

f-2 (animal-sound quack)

For a total of 2 facts.

You can just use (**retract *)** to retract all the facts, where the "*" indicates *all* **.**

Lab 12

Topic: Installation guide MATLAB

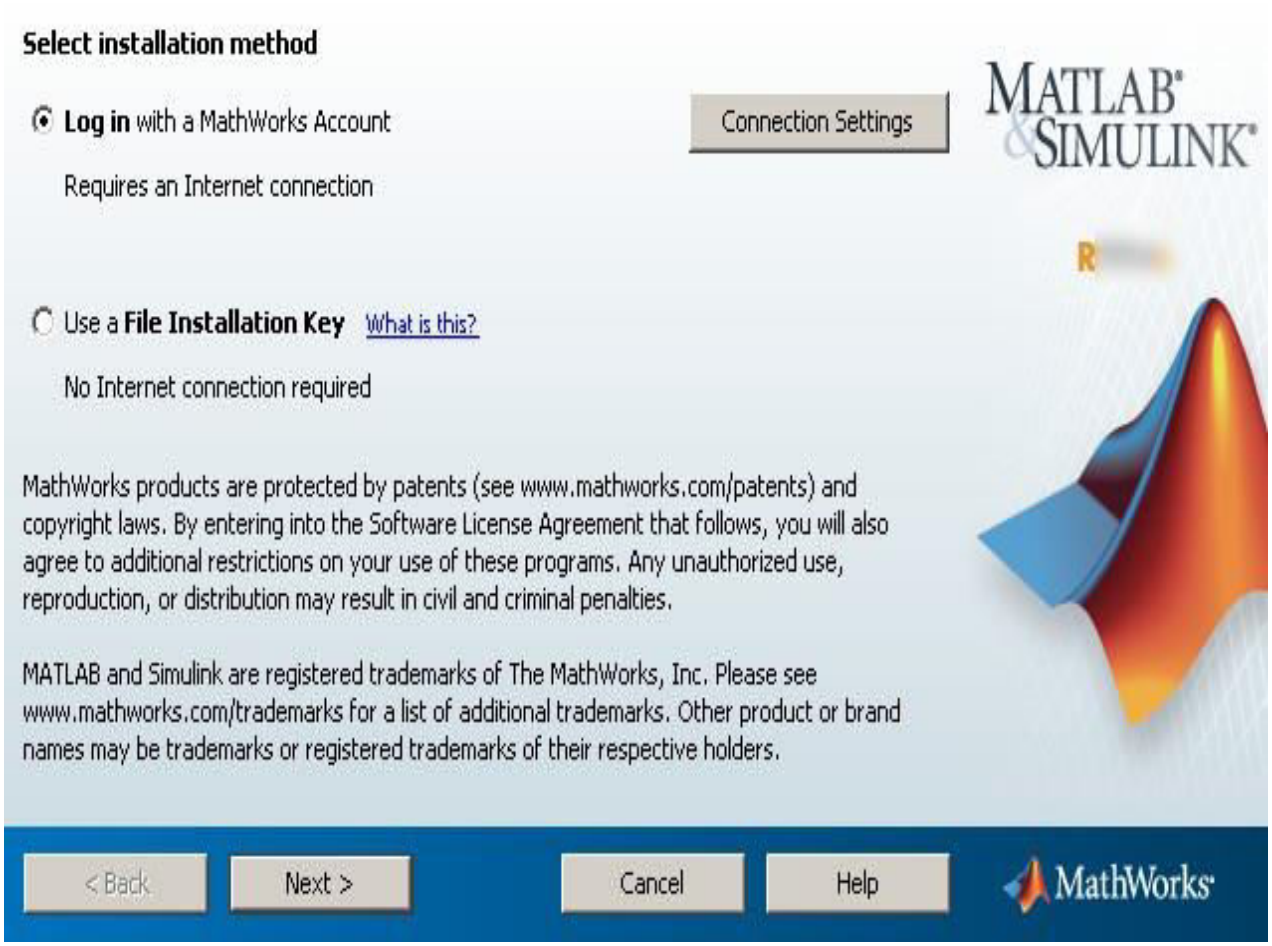
Problem Statement: You are required to install a MATLAB 2017 through DVD. Installation guide is available on following link

https://www.kth.se/polopoly_fs/1.492736!/matlab_withma.pdf

Solution

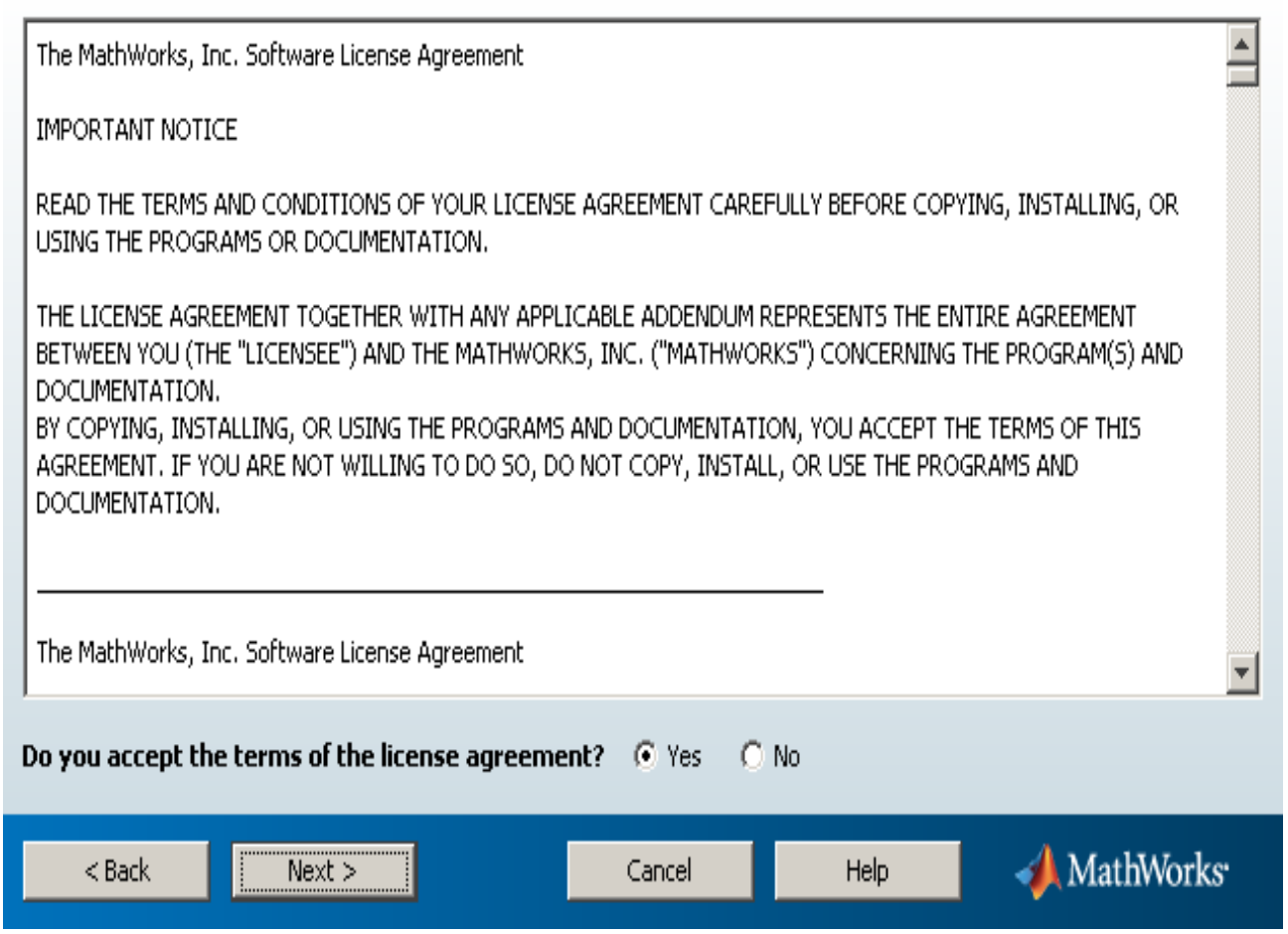
Step 1

Select *"Log in with a MathWorks Account"* and press **Next**.



Read the license agreement, mark "*yes*" to accept the terms, and press **Next**.

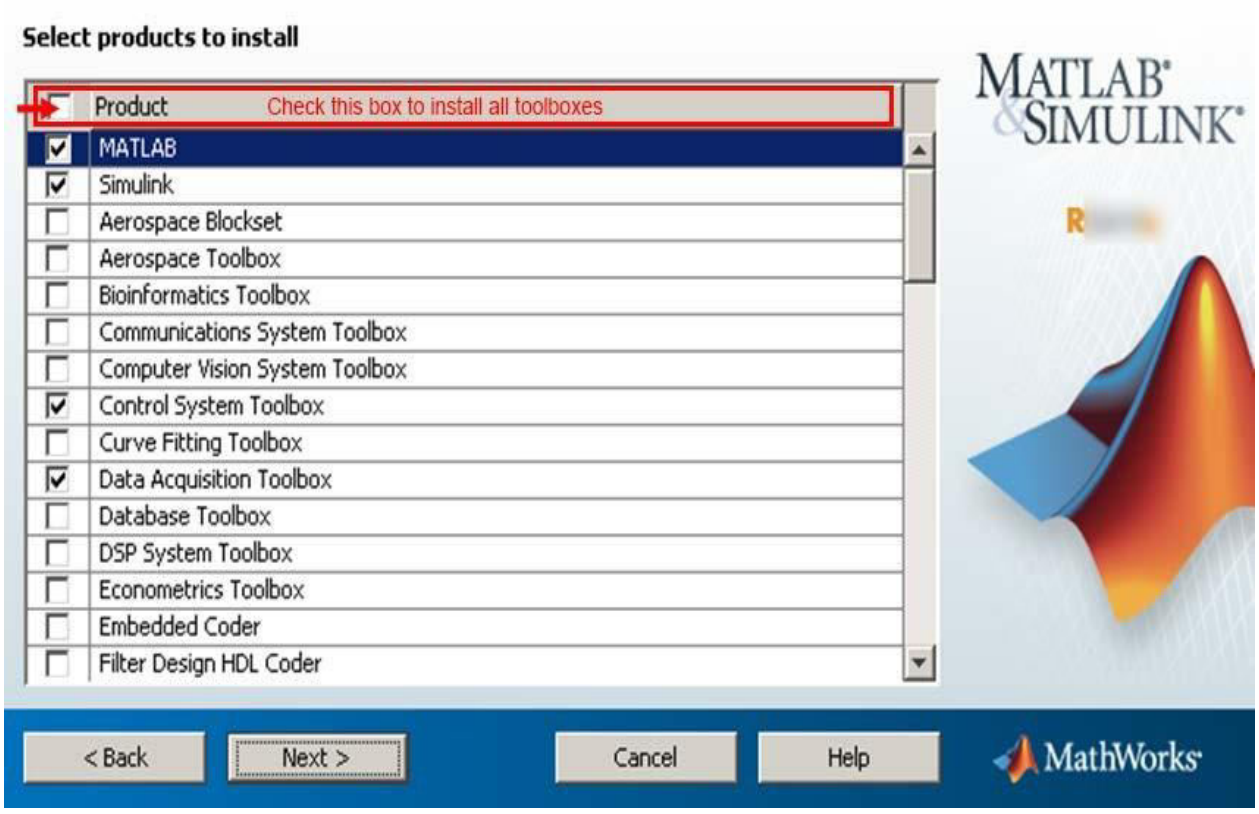
Step 3



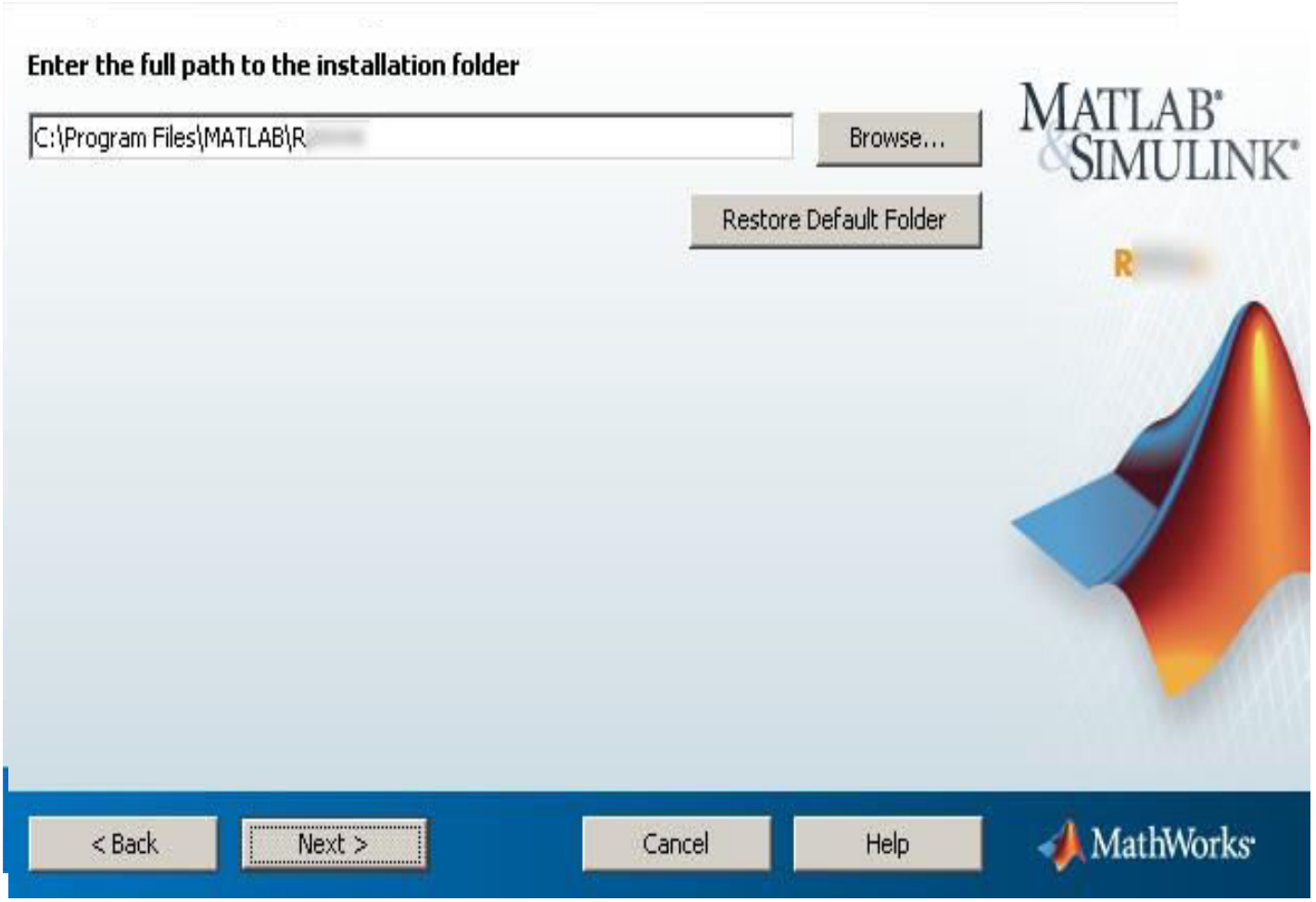
Select *"Log in to your MathWorks Account"*, provide your MathWorks account login information, and press **Next**.

Step 4


Log in to your MathWorks Account
Email address:
Password:
****
Forgot your password?
Create a MathWorks Account (requires an Activation Key)
MATLAB
SIMULINK
< Back
Next >
Cancel
Help
MathWorks

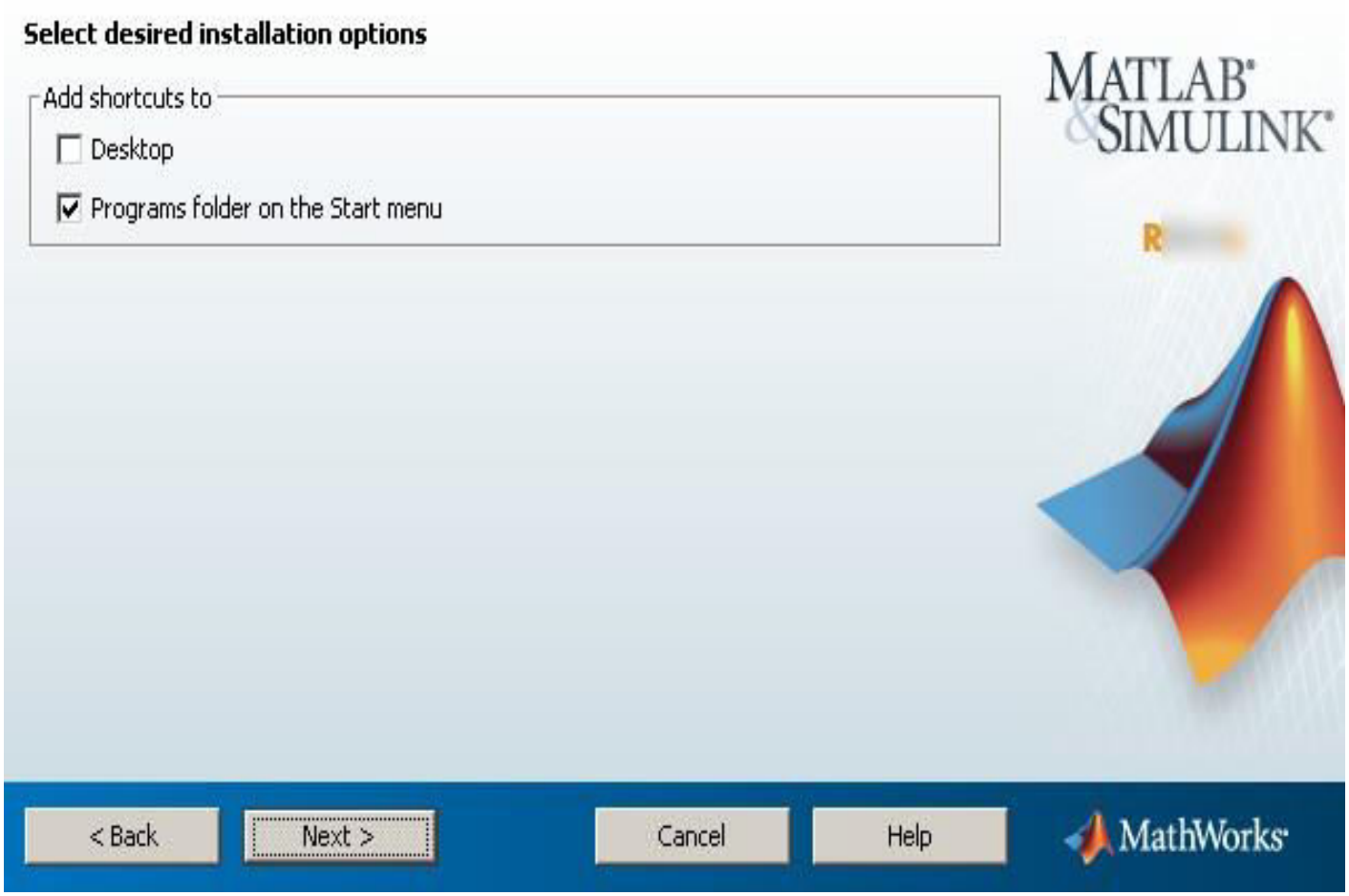

Select products to install
Product
Check this box to install all toolboxes
MATLAB
Simulink
Aerospace Blockset
Aerospace Toolbox
Bioinformatics Toolbox
Communications System Toolbox
Computer Vision System Toolbox
Control System Toolbox
Curve Fitting Toolbox
Data Acquisition Toolbox
Database Toolbox
DSP System Toolbox
Econometrics Toolbox
Embedded Coder
Filter Design HDL Coder
MATLAB
SIMULINK
< Back
Next >
Cancel
Help
MathWorks

Step 5

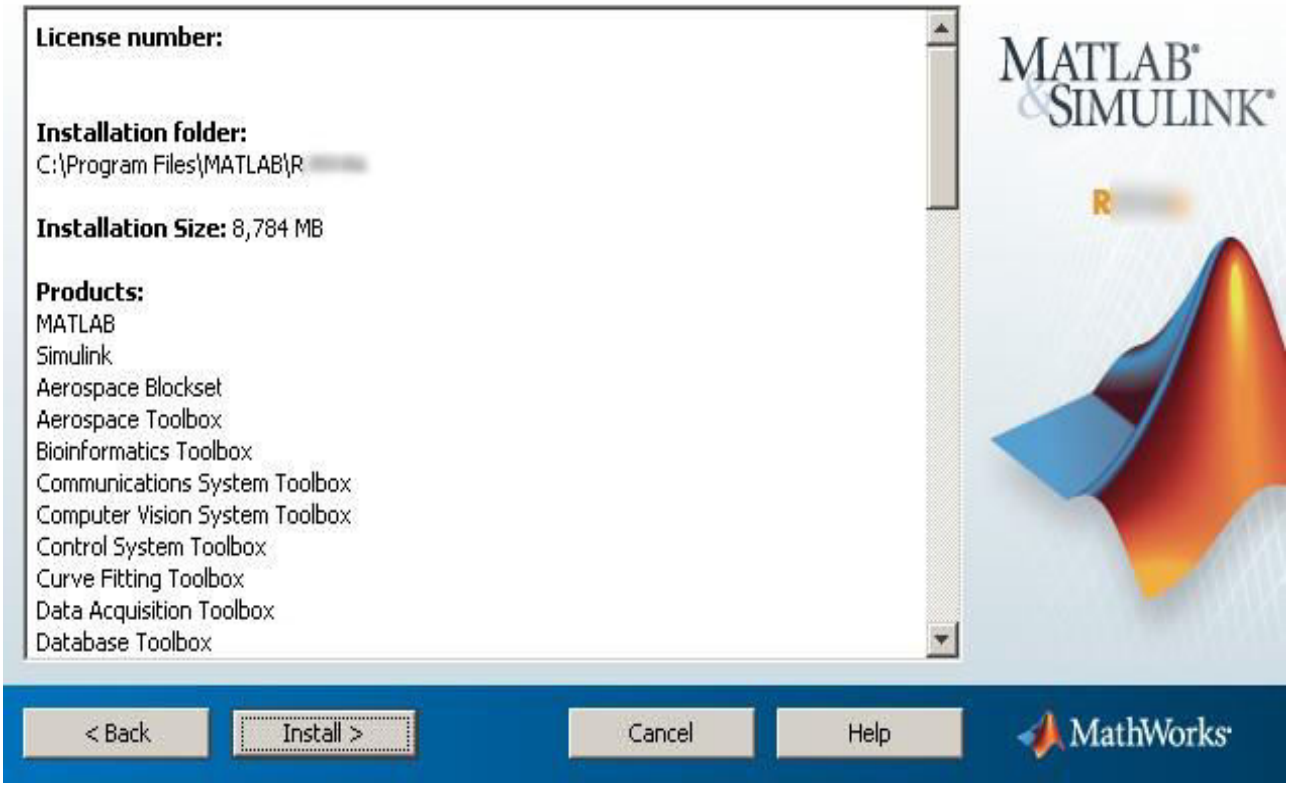


Make your selection and press **Next** to continue.

Step 6

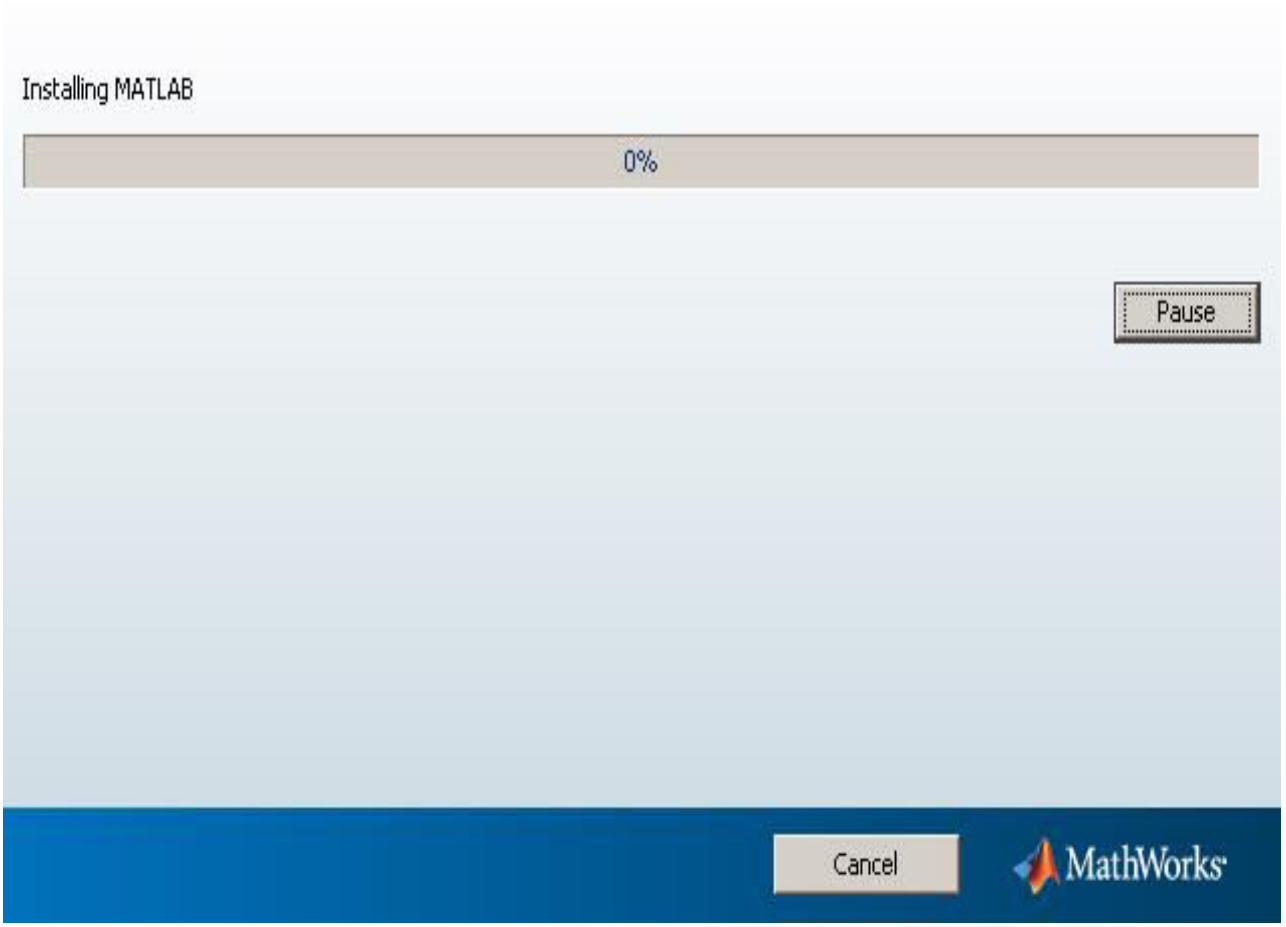


A last chance to review all the choices before proceeding. If ready, press **Install** to commit.



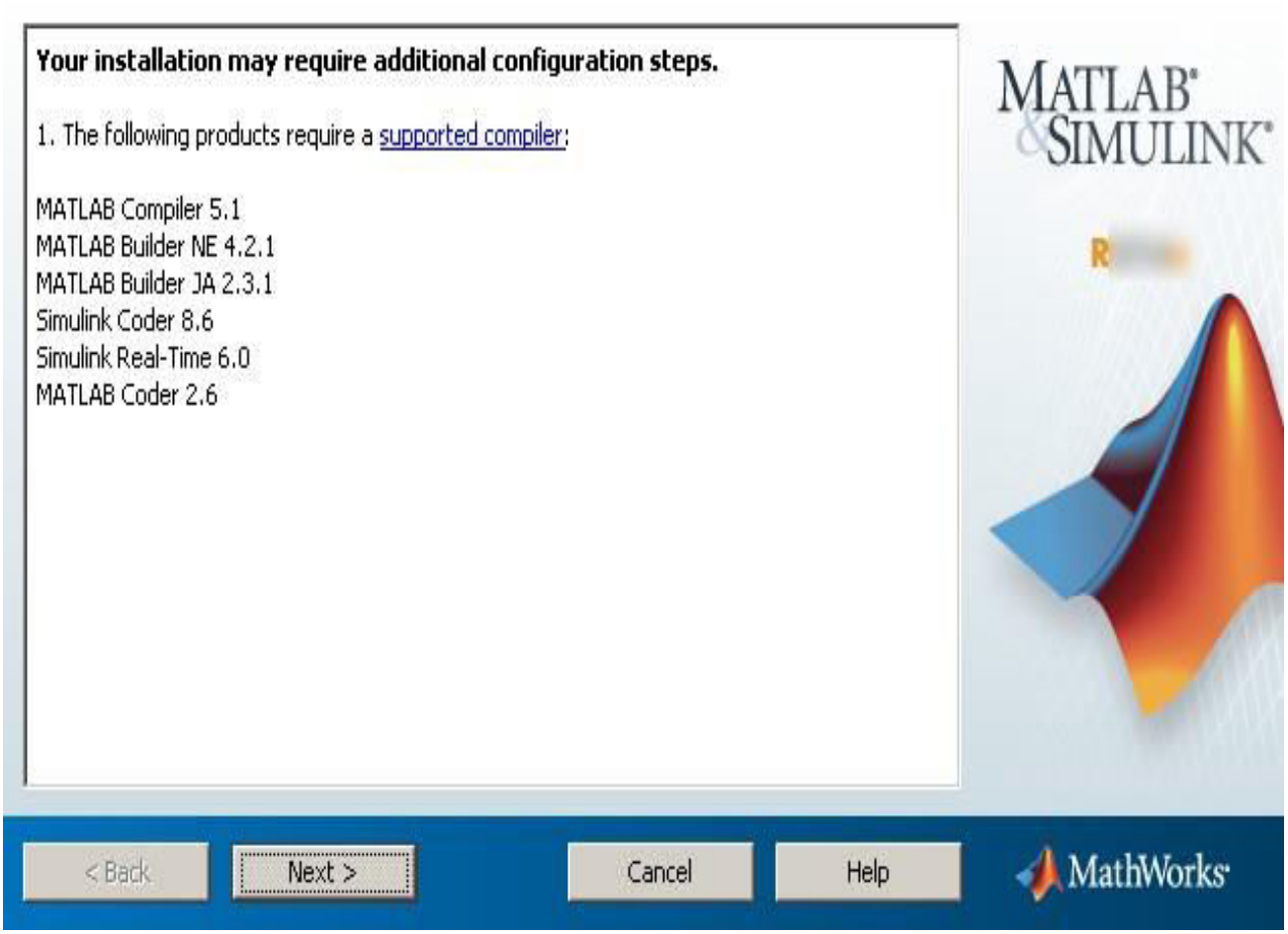
Step 7

After the installation, the installer will notify about any requirements if needed. Press **Next** to



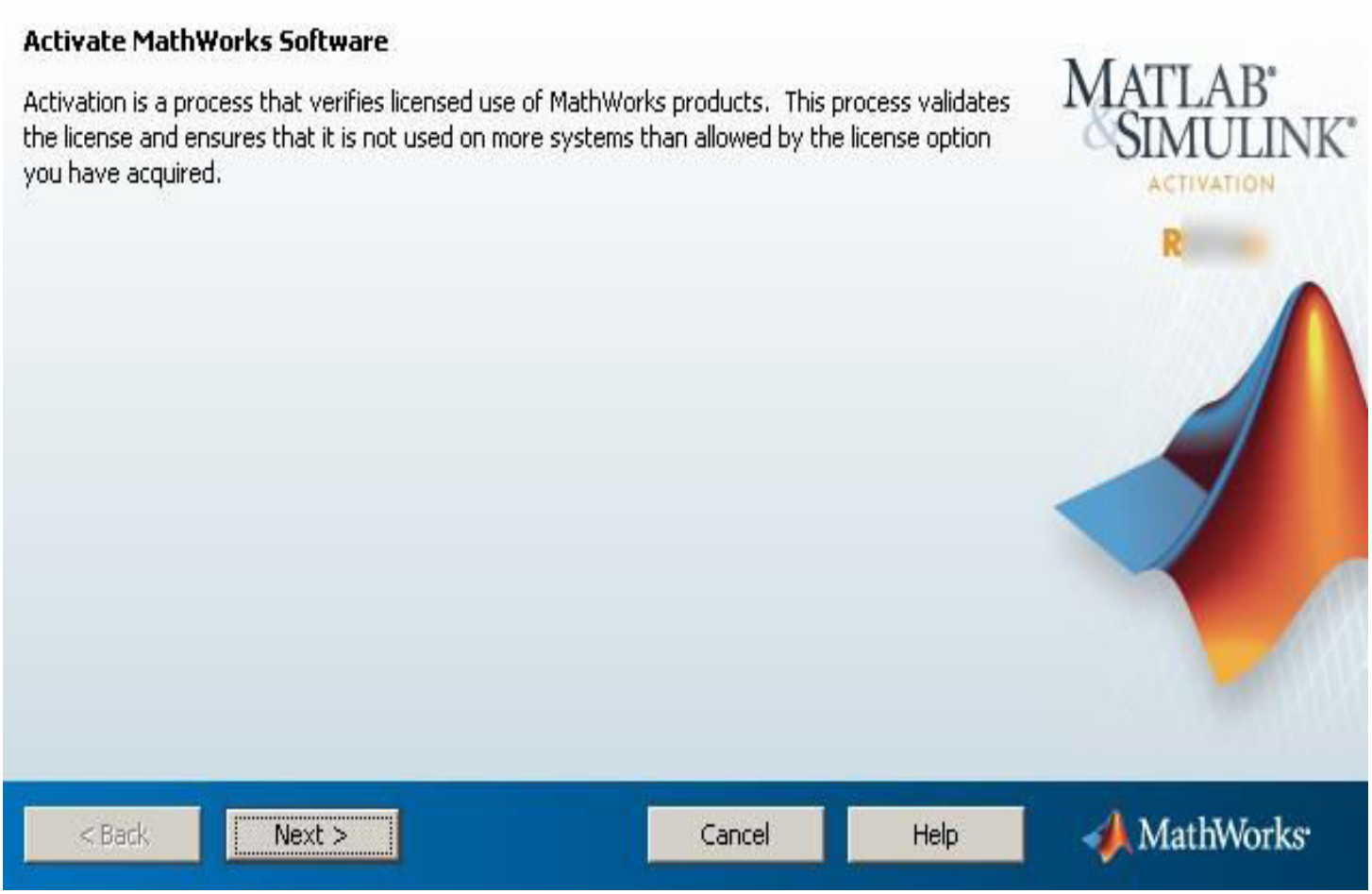
continue.

Step 8

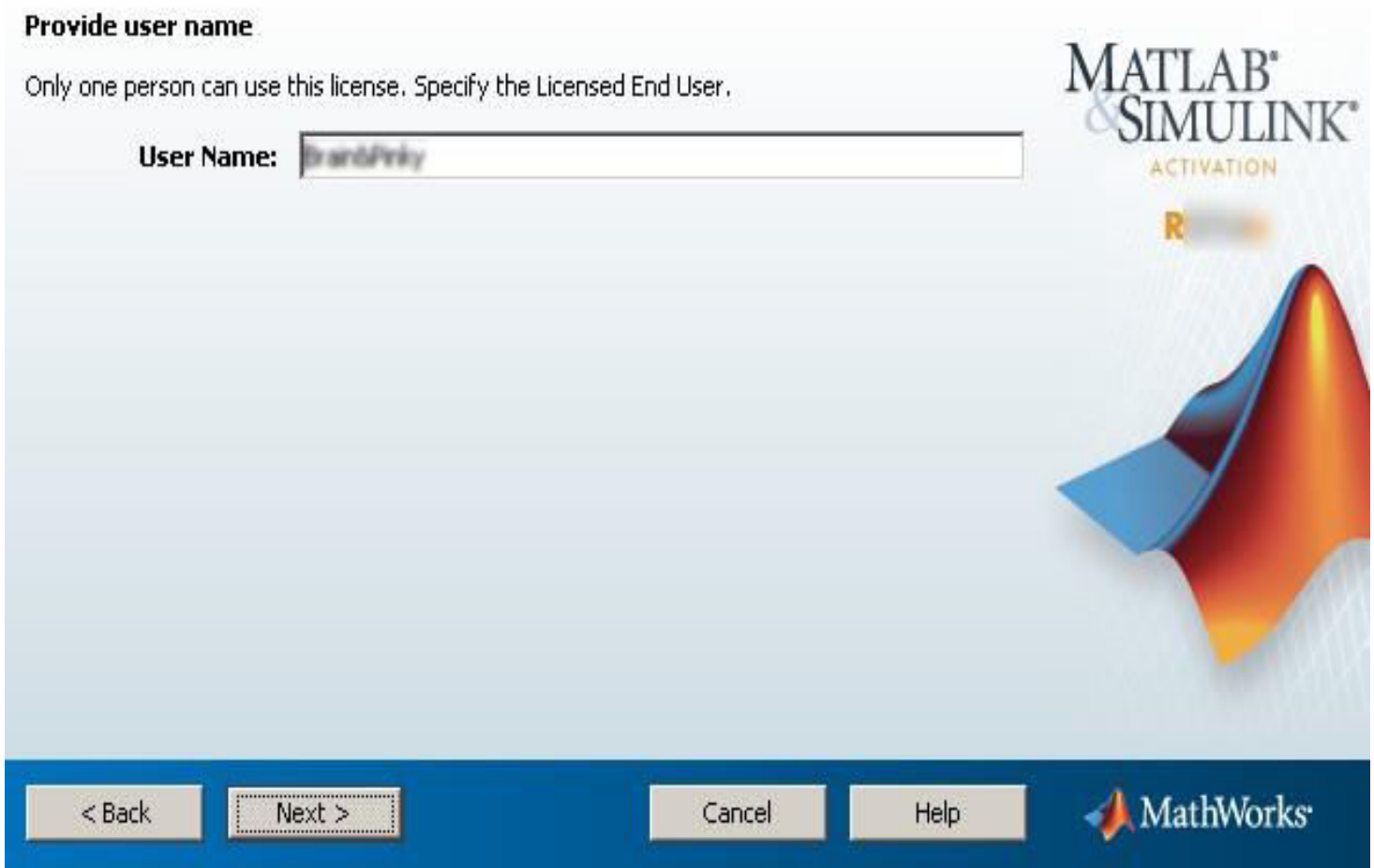


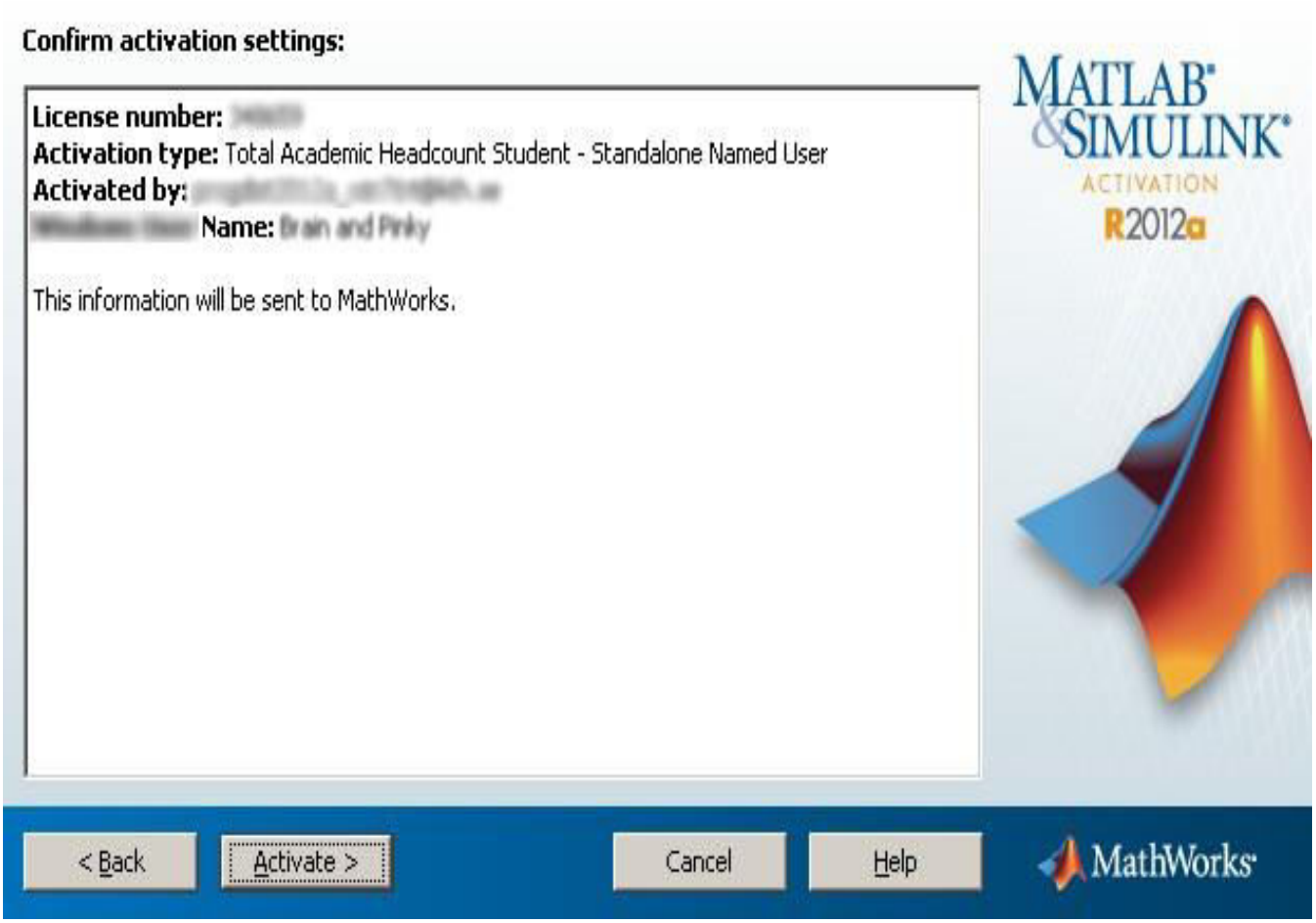
Make sure the *"Activate MATLAB"* box is checked and press **Next**.

Step 9 Press **Next** to start the activation process.



Provide your user name on the computer (**the <u>local</u> user on the computer for whom Matlab is being installed**) and press **Next** to proceed.




Confirm activation settings:
License number:
Activation type: Total Academic Headcount Student - Standalone Named User
Activated by:
Name: Brain and Pinky
This information will be sent to MathWorks.
MATLAB
&SIMULINK
ACTIVATION
R2012a
< Back
Activate >
Cancel
Help
MathWorks

Press **Finish** to end the MATLAB installation.

Mechanism to Conduct Lab:

Students and teacher will communicate through google meet/ zoom platform. Student will run the code OR show the procedure for a given problem and share the screen with instructor.

Lab 13

Topic: Tutorial of MATLAB Interface

Problem Statement: How to use MATLAB for fuzzy inference system

Solution:

Following is the default interface that appears on starting the *MATLAB R2017a* software. This is divided into three windows i.e. command window, workspace window and current folder window are shown in Figure1.

- **Command window** allows commands to be executed are written and results from the executed commands are also displayed.
- **Workspace window** displays the variables to be used.the variables used for analysis are displayed.
- **Current folder window** displays the files displayed and the researcher can access same.

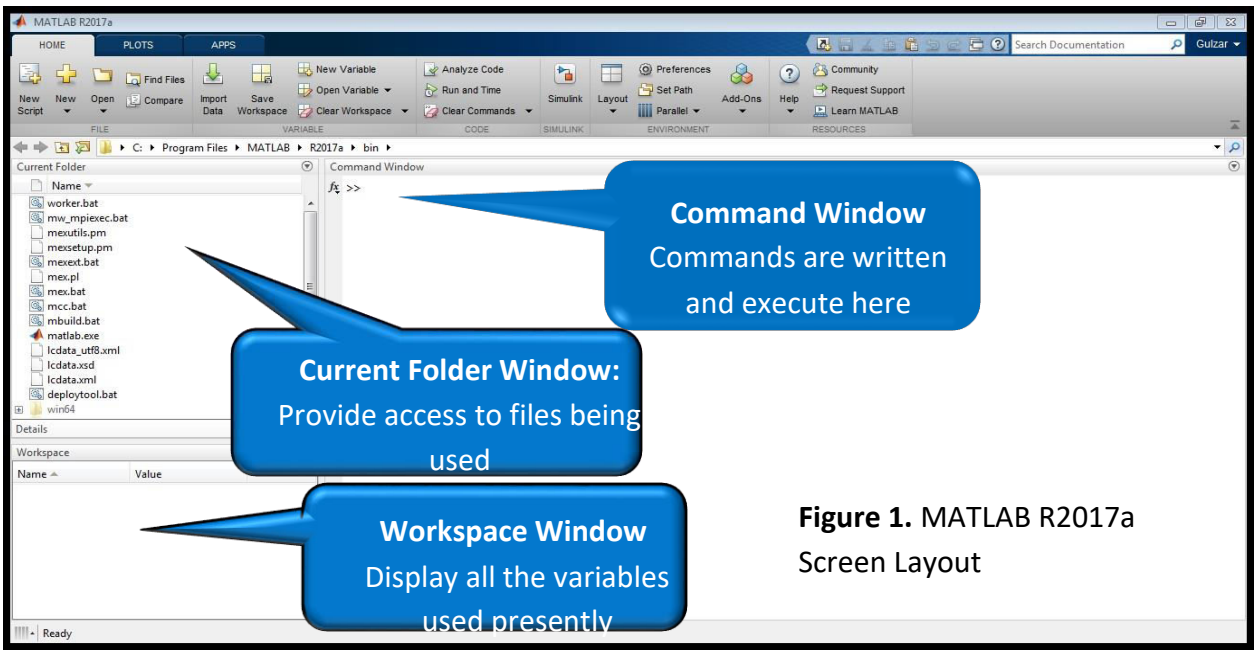


Figure 1. MATLAB R2017a Screen Layout

Understanding the Quick Access Toolbar

Further, the *MATLAB* interface consists of a toolbar on top of the window called "Quick Access Toolbar" consisting of three tabs namely **"HOME", "PLOTS" and "APPS".** These are called Global Tabs. The figures below show the **"HOME", "PLOTS" and "APPS"** tab respectively.

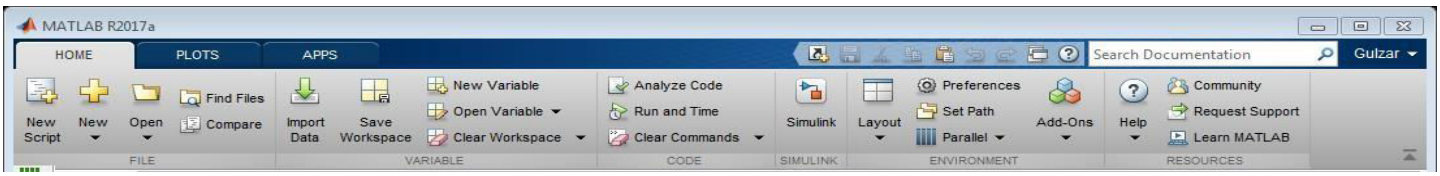
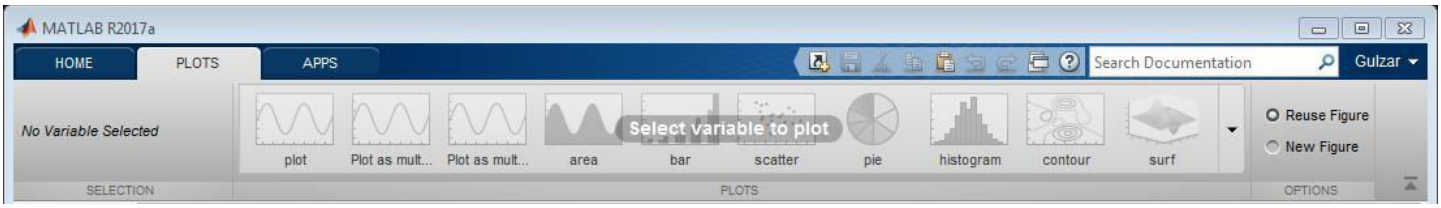


Figure2a. Quick Access Toolbar for HOME tab in MATLAB



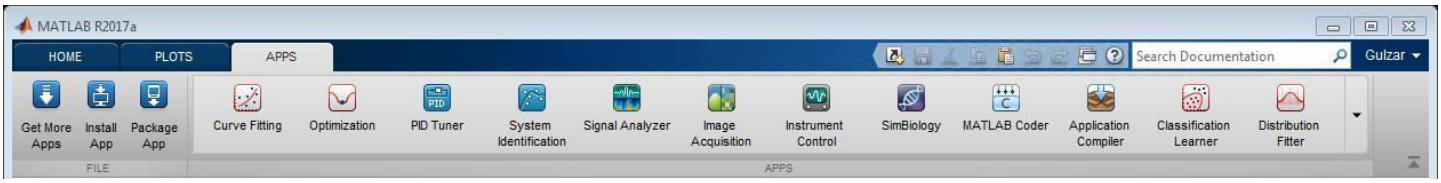


Figure2b. Using Quick Access Toolbar for PLOT tab in *MATLAB*

Figure2c. Quick Access Toolbar for APPS tab in *MATLAB* interface

Fuzzy Inference Systems

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

1. **Fuzzy Logic Designer**
2. **Membership Function Editor**
3. **Rule Editor**
4. **Rule Viewer**
5. **Surface Viewer**

1. **Fuzzy Logic Designer:**

Fuzzy Logic Designer is used 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.

2. **Membership Function Editor:**

Membership Function Editor is used to define the shapes of all the membership functions associated with each variable

3. **Rule Editor:**

Rule Editor is used to edit the list of rules that defines the behavior of the system.

4. **Rule Viewer:**

Rule Viewer is used 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

5. **Surface Viewer:**

Surface Viewer is used 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 User Interfaces (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 FIS systems.

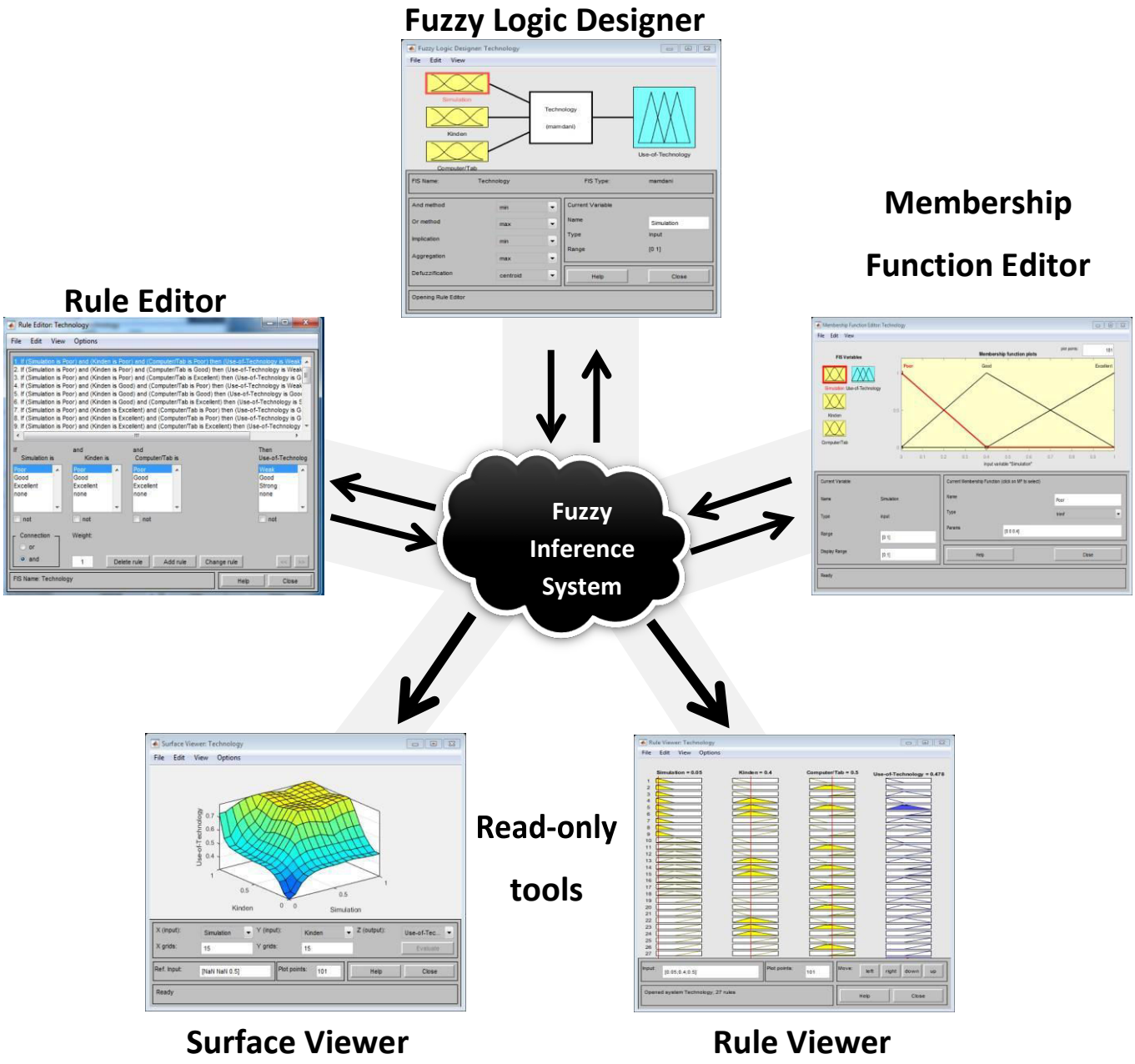


Figure 3. components of a fuzzy inference system

The above figure.3 shows how the main components of a FIS and the three editors fit together. The two viewers examine the behavior of the entire system.

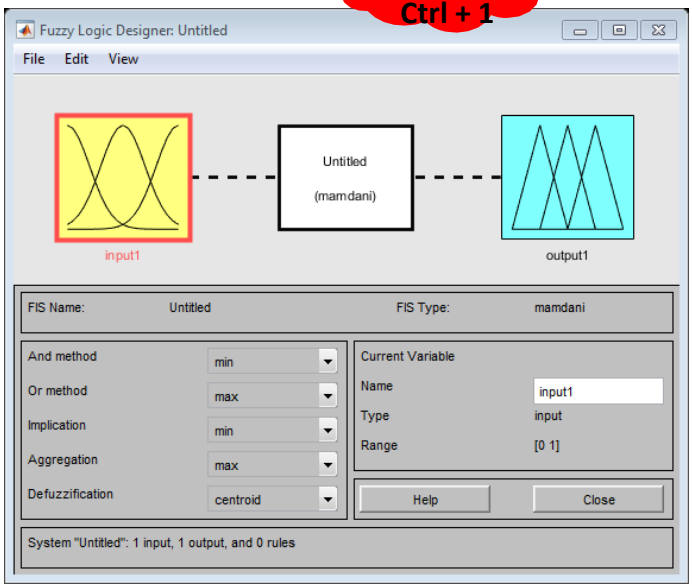
1. The Fuzzy Logic Designer

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

>>fuzzyLogicDesigner or Simply type: >> fuzzy

Short Key

Ctrl + 1



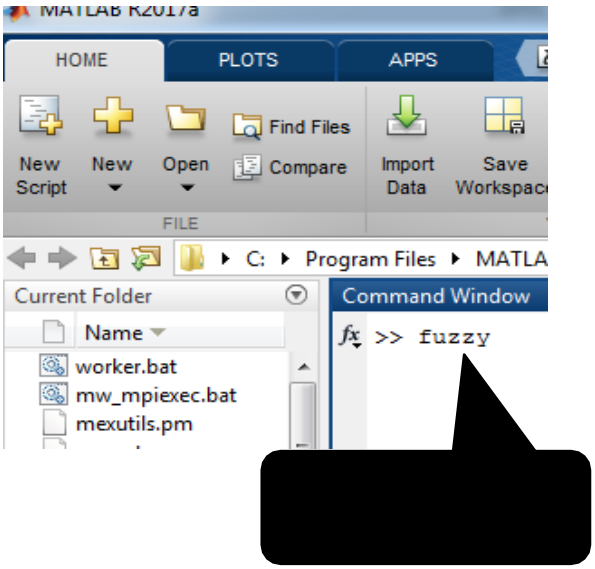


Figure 4. Fuzzy logic design of a fuzzy inference system

The **Fuzzy Logic Designer** opens and displays a diagram of the fuzzy inference system with the names of each input variable on the left, and those of each output variable on the right, as shown in the above figure. The
sample membership functions shown in the boxes are just icons and do not depict the actual shapes of the membership functions.

Alternative Method

1: Click at: **APPS**

2: Click at Arrow Tip

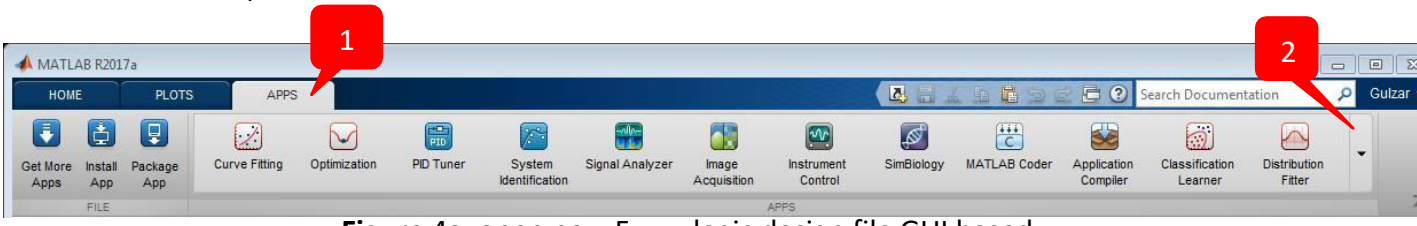


Figure 4a. open new Fuzzy logic design file GUI based

3: Click at: **Fuzzy Logic Designer** icon as shown at in below figure 4b.

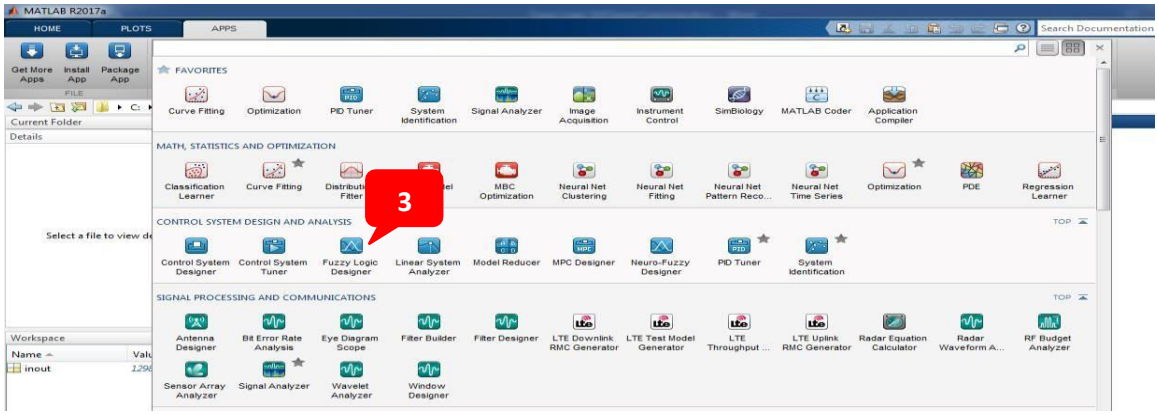


Figure 4b. open new Fuzzy logic design file GUI

Lab 13

Topic: Tutorial of MATLAB Interface

Problem Statement: How to use MATLAB for fuzzy inference system

Solution:

Following is the default interface that appears on starting the *MATLAB R2017a* software. This is divided into three windows i.e. command window, workspace window and current folder window are shown in Figure1.

- **Command window** allows commands to be executed are written and results from the executed commands are also displayed.
- **Workspace window** displays the variables to be used.the variables used for analysis are displayed.
- **Current folder window** displays the files displayed and the researcher can access same.

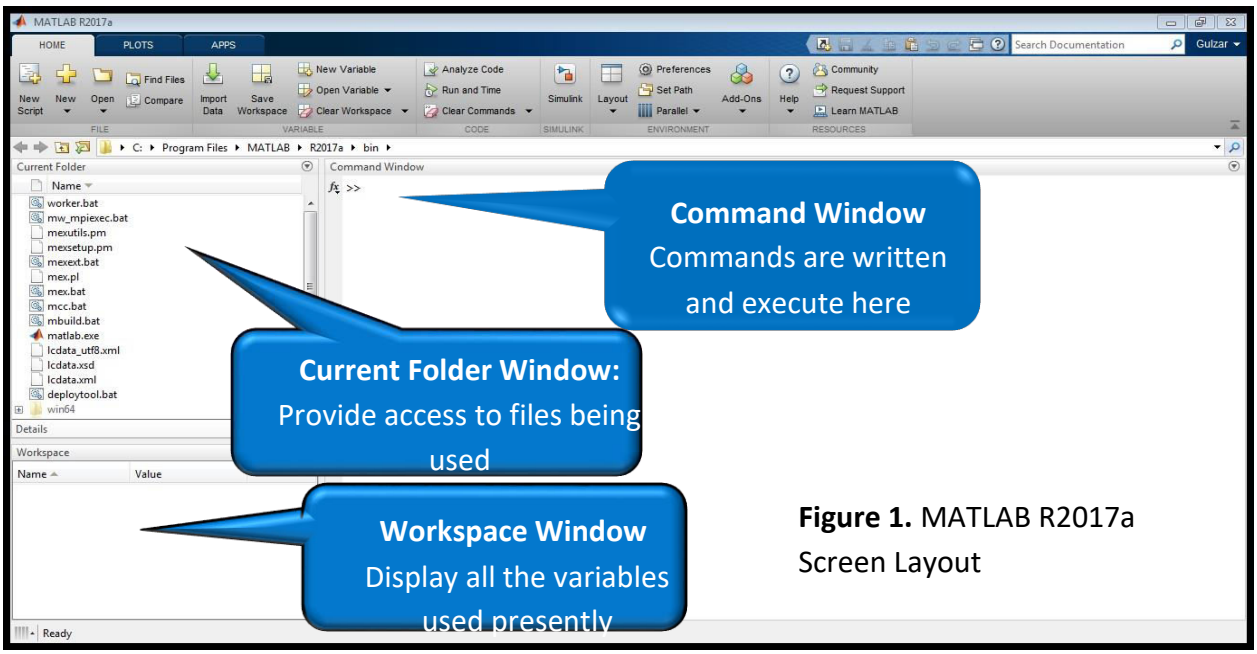


Figure 1. MATLAB R2017a Screen Layout

Understanding the Quick Access Toolbar

Further, the *MATLAB* interface consists of a toolbar on top of the window called "Quick Access Toolbar" consisting of three tabs namely **"HOME", "PLOTS" and "APPS".** These are called Global Tabs. The figures below show the **"HOME", "PLOTS" and "APPS"** tab respectively.

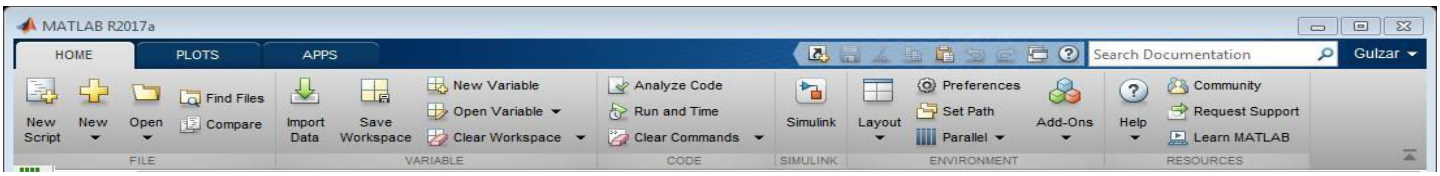
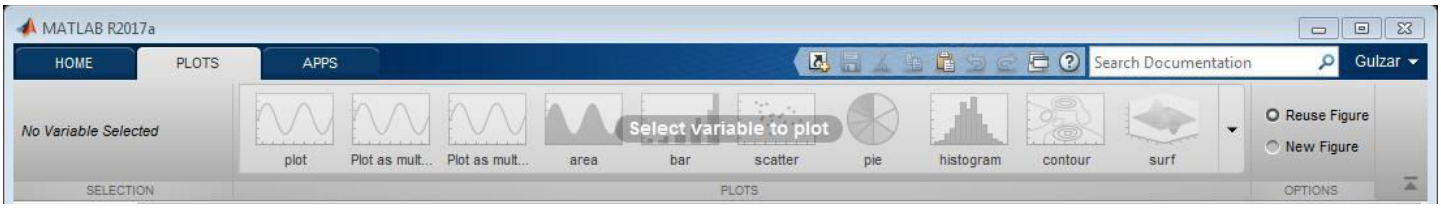


Figure2a. Quick Access Toolbar for HOME tab in MATLAB



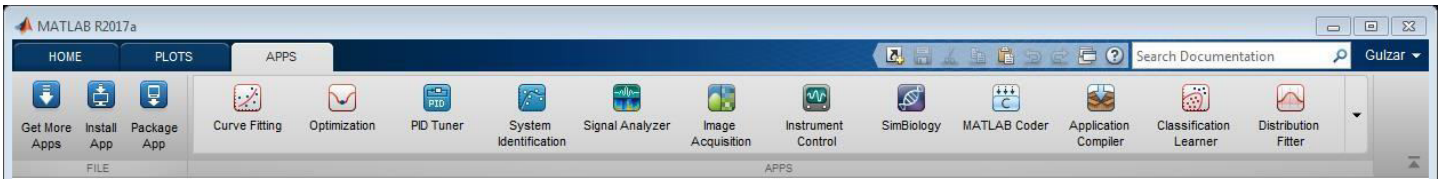


Figure2b. Using Quick Access Toolbar for PLOT tab in *MATLAB*

Figure2c. Quick Access Toolbar for APPS tab in *MATLAB* interface

Fuzzy Inference Systems

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

1. **Fuzzy Logic Designer**
2. **Membership Function Editor**
3. **Rule Editor**
4. **Rule Viewer**
5. **Surface Viewer**

1. **Fuzzy Logic Designer:**

Fuzzy Logic Designer is used 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.

2. **Membership Function Editor:**

Membership Function Editor is used to define the shapes of all the membership functions associated with each variable

3. **Rule Editor:**

Rule Editor is used to edit the list of rules that defines the behavior of the system.

4. **Rule Viewer:**

Rule Viewer is used 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

5. **Surface Viewer:**

Surface Viewer is used 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 User Interfaces (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 FIS systems.

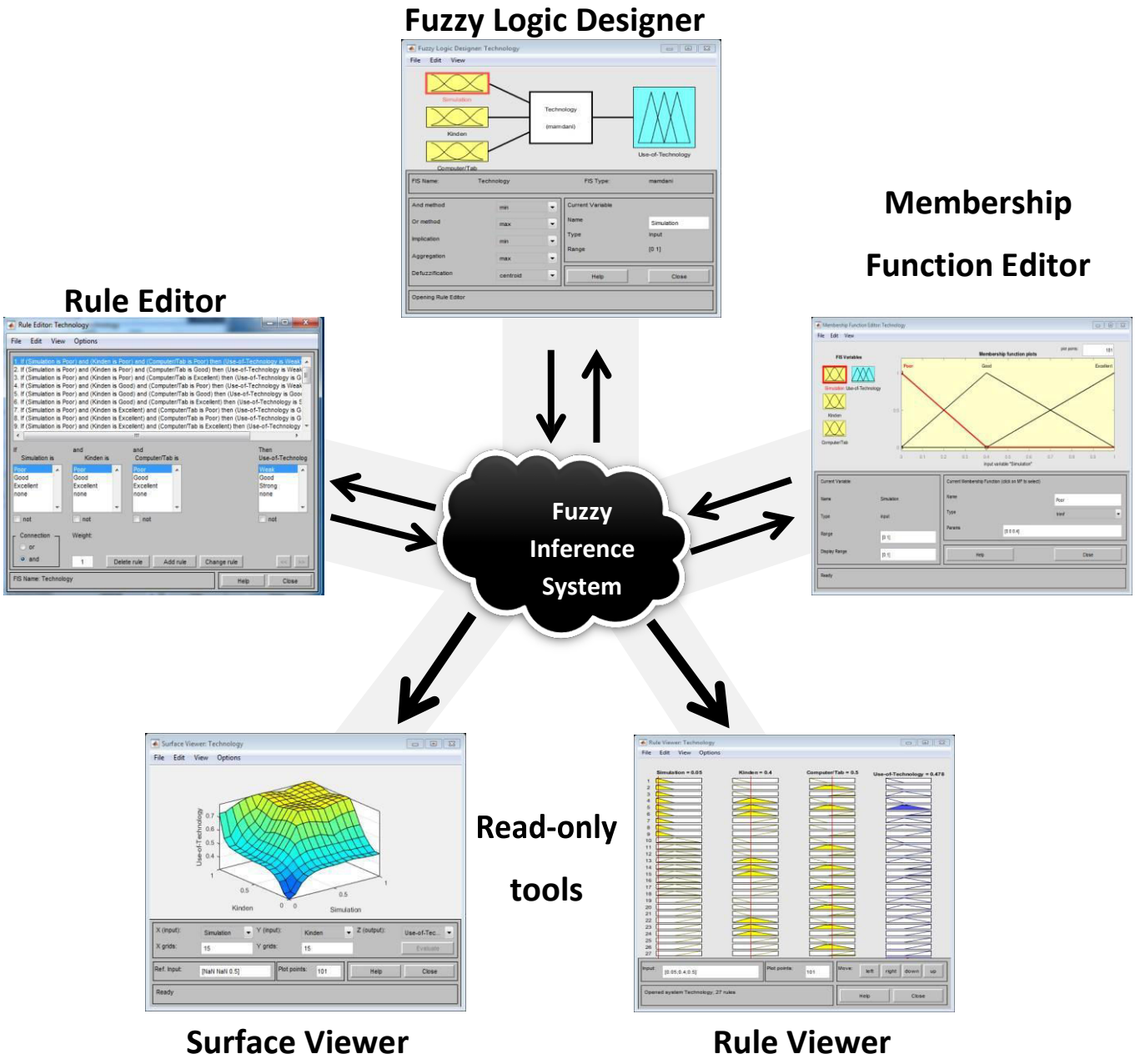


Figure 3. components of a fuzzy inference system

The above figure.3 shows how the main components of a FIS and the three editors fit together. The two viewers examine the behavior of the entire system.

1. The Fuzzy Logic Designer

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

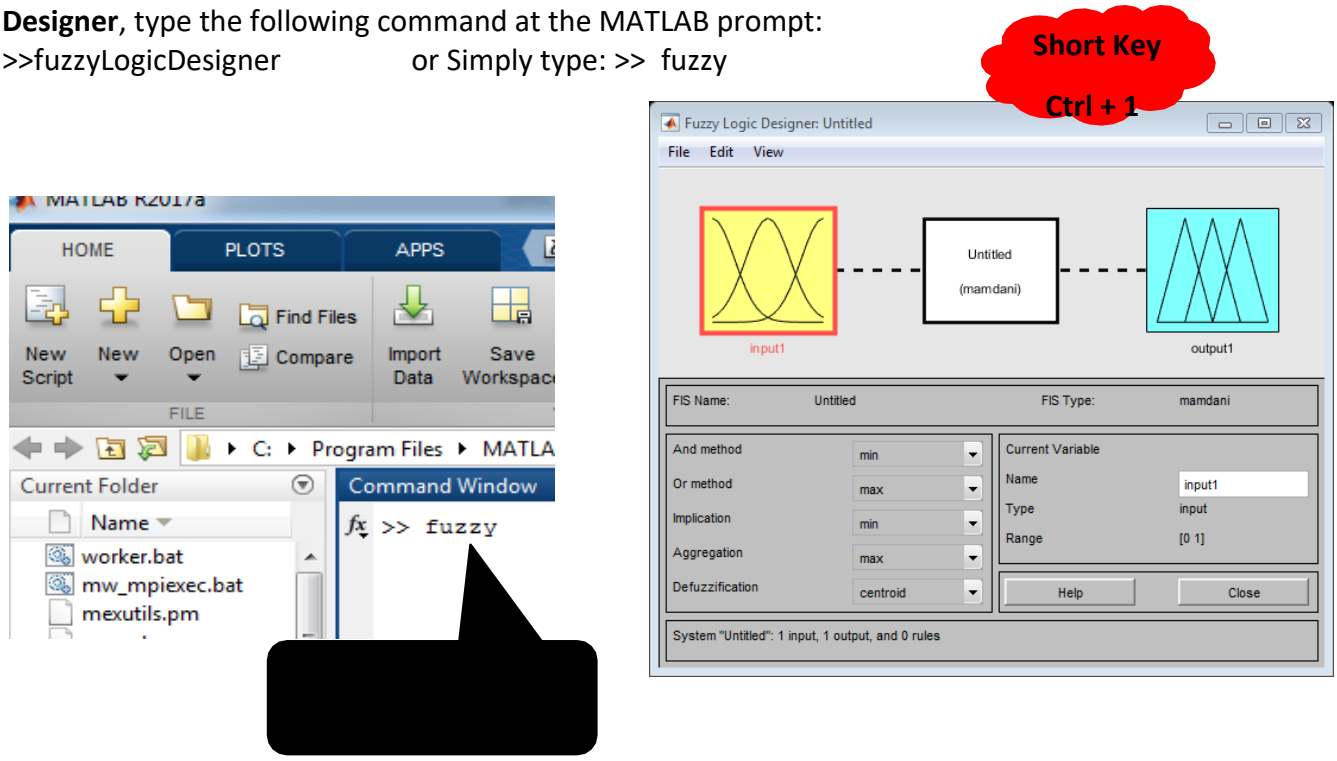


Figure 4. Fuzzy logic design of a fuzzy inference system

The **Fuzzy Logic Designer** opens and displays a diagram of the fuzzy inference system with the names of each input variable on the left, and those of each output variable on the right, as shown in the above figure. The sample membership functions shown in the boxes are just icons and do not depict the actual shapes of the membership functions.

Alternative Method

1: Click at: **APPS**

2: Click at Arrow Tip

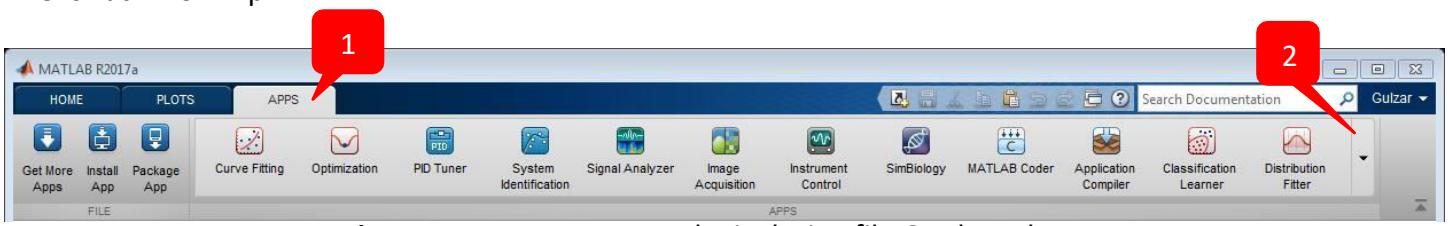


Figure 4a. open new Fuzzy logic design file GUI based

3: Click at: **Fuzzy Logic Designer** icon as shown at in below figure 4b.

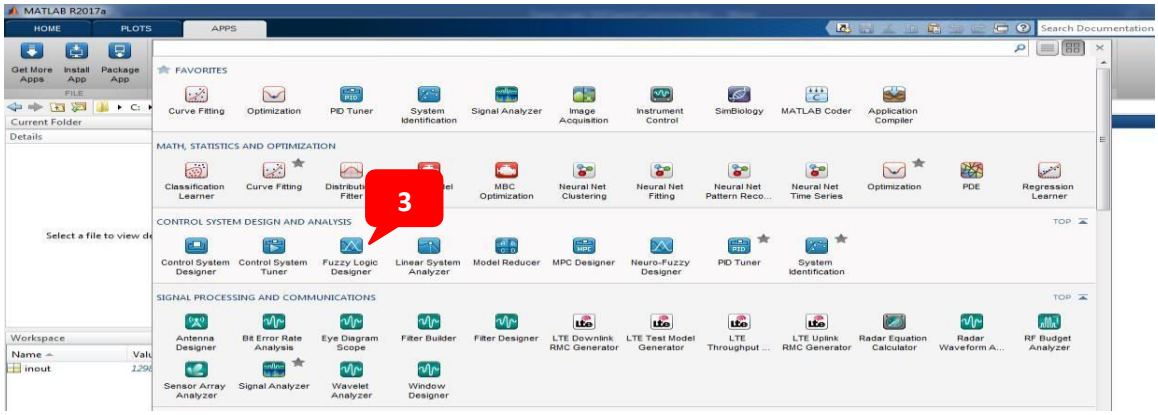


Figure 4b. open new Fuzzy logic design file GUI

Lab 14

Topic: Development of Fuzzy inference system

Problem Statement: You are required to make a fuzzy inference system with following requirements

- Defining input, output variables FIS system.
- Defining ranges of input and output variables.

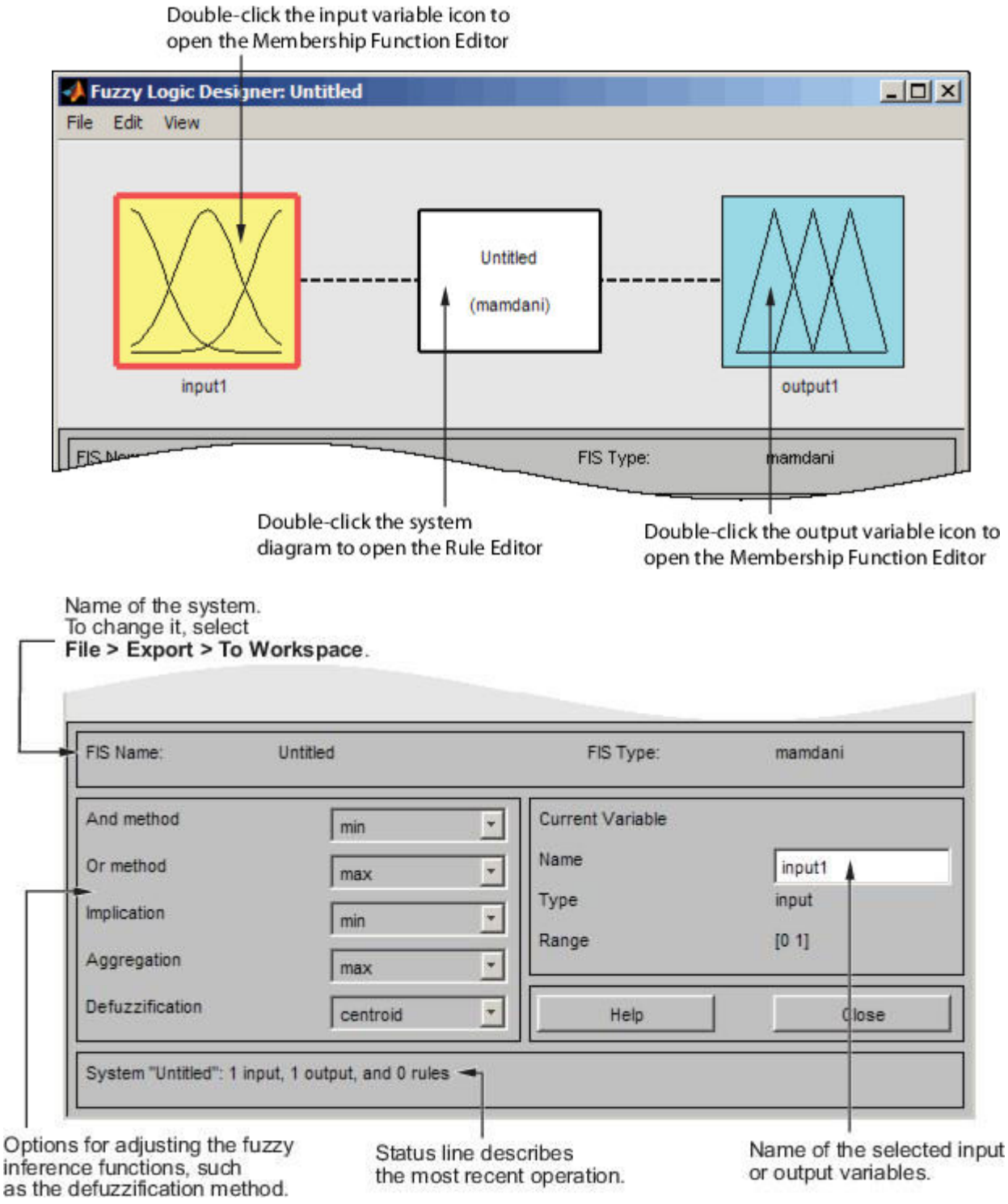
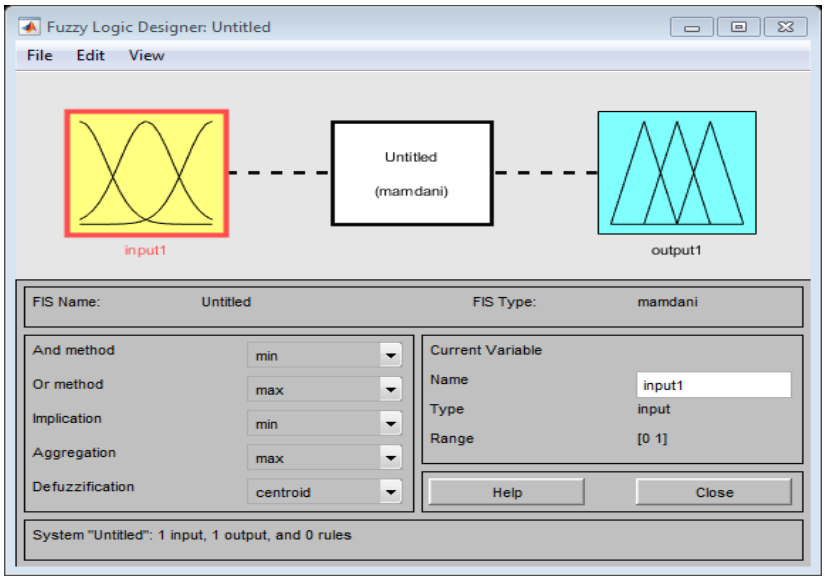


Figure 5. Overview of a Fuzzy logic design Interface

In this example, you use the default Mamdani-type inference. In the **Fuzzy Logic Designer**:

- The drop-down lists let you modify the fuzzy inference functions.
- The **Current Variable** area displays the name of either an input or output variable, its type, and default range.

The generic untitled **Fuzzy Logic Designer** opens, with one input **input1**, and one output **output1**.



Add Input Variables

To add a second and third input variables and change the variable names to reflect these designations:

1. Select **Edit** → **Add variable** → **Input**. Second and Third yellow boxes labeled **input2 and input3**
2. Click the yellow box **input1**. This box is highlighted with a red outline.
3. Edit the **Name** field from input1 to **Simulation**, and press **Enter**.
4. Click the yellow box **input2**. This box is highlighted with a red outline.
5. Edit the **Name** field from input2 to **Kindle**, and press **Enter**.
6. Click the yellow box **input3**. This box is highlighted with a red outline.
7. Edit the **Name** field from input3 to **Computer/Tab**, and press **Enter**.
8. Click the blue box **output1**.
9. Edit the **Name** field from output1 to **Use-of-Technology**, and press **Enter**.

10. Select **File** > **Export** > **To Workspace**.

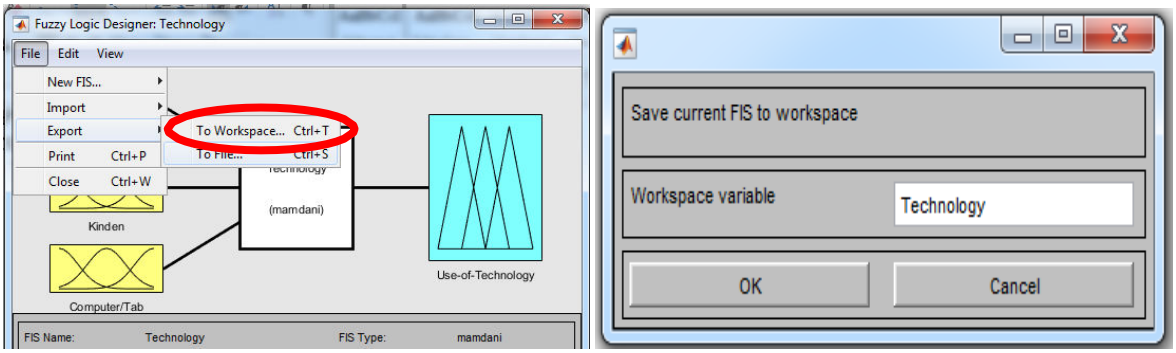


Figure 7a. Export existing FIS file to workstation

11. Enter the **Workspace variable** name **Technology** and click **OK**.
12. Or save in your desired location in computer by:
13. Select **File** > **Export** > **To File**

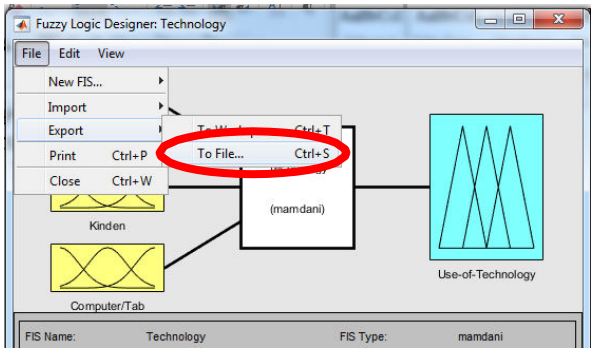


Figure 7b. Export existing FIS file to file (Desired folder)

The diagram is updated to reflect the new names of the input and output variables. There is now a new variable in the workspace called **Technology** that contains all the information about this system. By saving to the workspace with a new name, you also rename the entire system. Your window looks something like the Figure6.
Leave the inference options in the lower left in their default positions for now. You have entered all the information you need for this particular UI. Next, define the membership functions associated with each of the variables. To do this, open the Membership Function Editor.

Lab 15

Topic: Development of Fuzzy inference system

Problem Statement: You are required to make a fuzzy inference system with following requirements

- Defining input, output variables FIS system.
- Defining ranges of input and output variables.

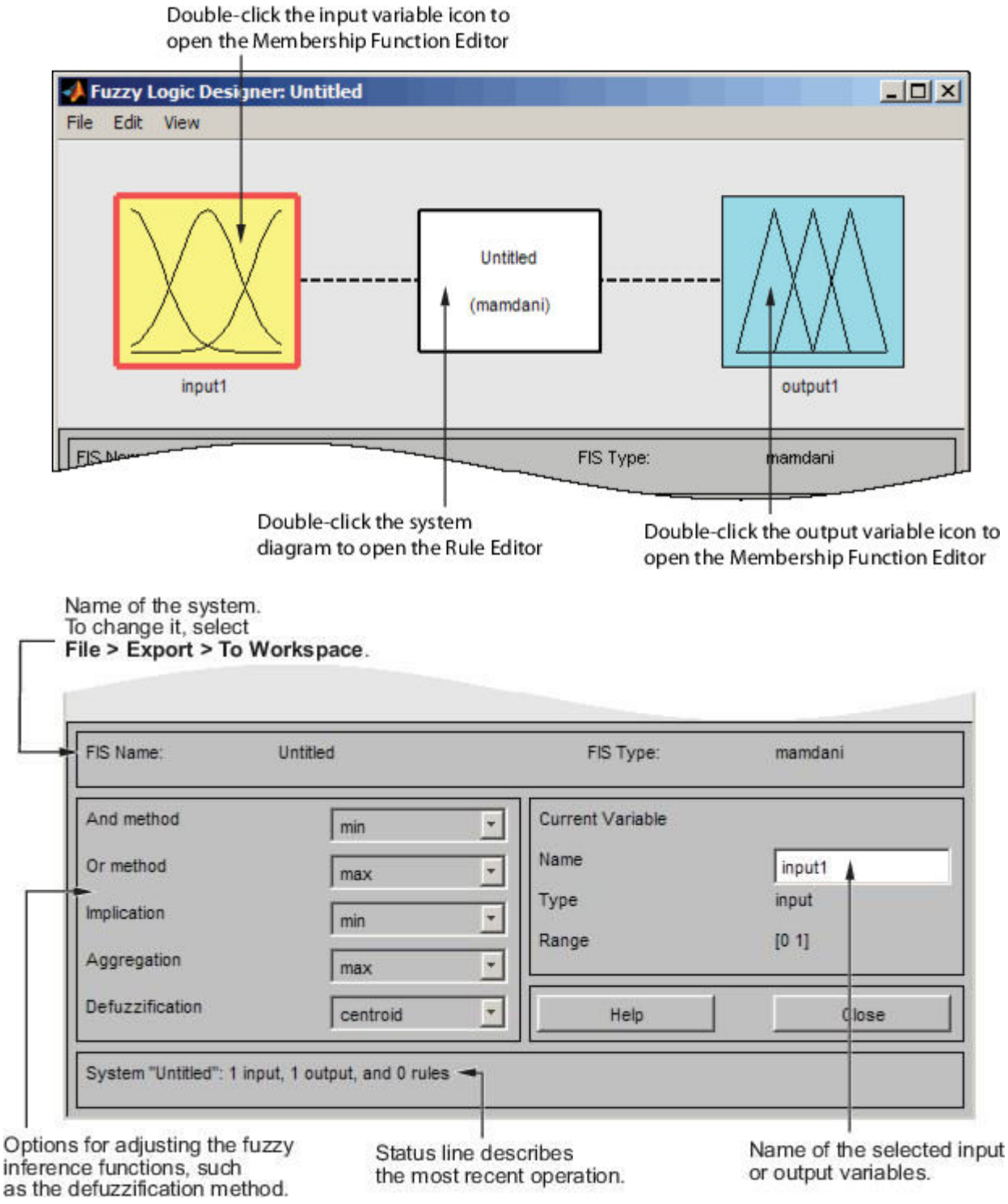
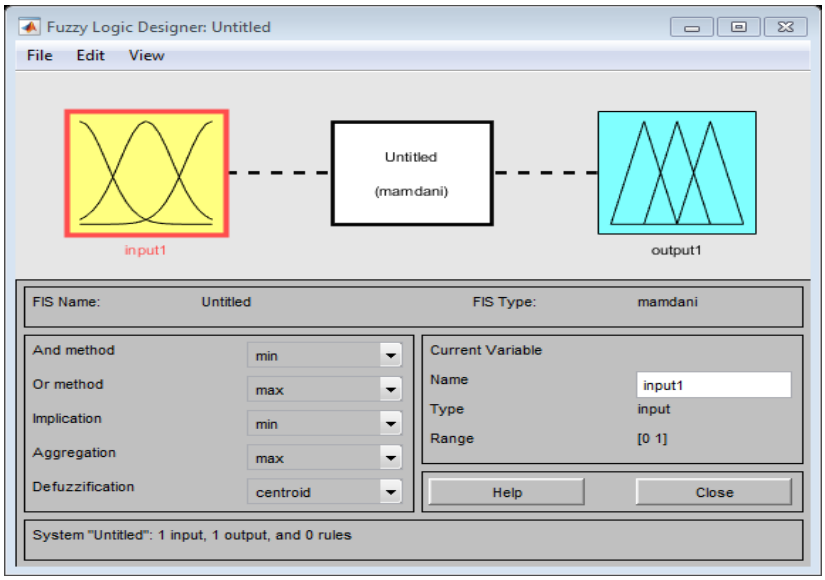


Figure 5. Overview of a Fuzzy logic design Interface

In this example, you use the default Mamdani-type inference. In the **Fuzzy Logic Designer**:

- The drop-down lists let you modify the fuzzy inference functions.
- The **Current Variable** area displays the name of either an input or output variable, its type, and default range.

The generic untitled **Fuzzy Logic Designer** opens, with one input **input1**, and one output **output1**.



Add Input Variables

To add a second and third input variables and change the variable names to reflect these designations:

1. Select **Edit** → **Add variable** → **Input**. Second and Third yellow boxes labeled **input2 and input3**
2. Click the yellow box **input1**. This box is highlighted with a red outline.
3. Edit the **Name** field from input1 to **Simulation**, and press **Enter**.
4. Click the yellow box **input2**. This box is highlighted with a red outline.
5. Edit the **Name** field from input2 to **Kindle**, and press **Enter**.
6. Click the yellow box **input3**. This box is highlighted with a red outline.
7. Edit the **Name** field from input3 to **Computer/Tab**, and press **Enter**.
8. Click the blue box **output1**.
9. Edit the **Name** field from output1 to **Use-of-Technology**, and press **Enter**.

10. Select **File** > **Export** > **To Workspace**.

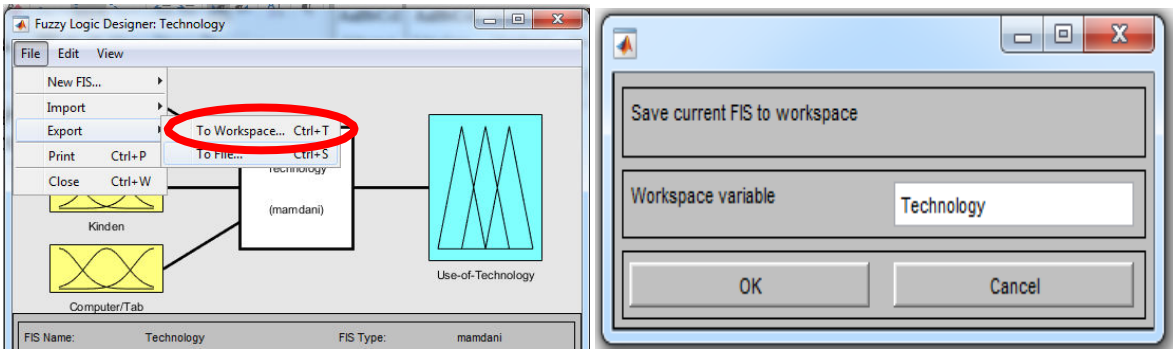


Figure 7a. Export existing FIS file to workstation

11. Enter the **Workspace variable** name **Technology** and click **OK**.
12. Or save in your desired location in computer by:
13. Select **File** > **Export** > **To File**

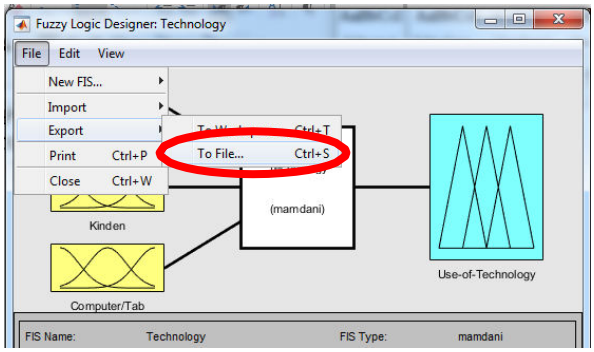


Figure 7b. Export existing FIS file to file (Desired folder)

The diagram is updated to reflect the new names of the input and output variables. There is now a new variable in the workspace called **Technology** that contains all the information about this system. By saving to the workspace with a new name, you also rename the entire system. Your window looks something like the Figure6.
Leave the inference options in the lower left in their default positions for now. You have entered all the information you need for this particular UI. Next, define the membership functions associated with each of the variables. To do this, open the Membership Function Editor.