APPENDIX A

MATLAB PROGRAM FOR SIMULATION

Programming of Switching Table for Conventional DTC

%switching table for DTC
clc, clear
a(:,:,1) = [1 0 2; 5 7 6];
a(:,:,2) = [5 7 3; 4 0 2];
a(:,:,3) = [4 0 1; 6 7 3];
a(:,:,4) = [6 7 5; 2 0 1];
a(:,:,5) = [2 0 4; 3 7 5];
a(:,:,6) = [3 7 6; 1 0 4];

Programming of If-Then Rules of the Rule Base used for the design of the Fuzzy Logic Controller

All the 49 If-Then Rules of the Rule Base used for the design of the Fuzzy Logic Controller are as follows:

IF (Error IS NL) AND (ChangeInError IS NL) THEN (ChangeOfControl IS NL)
IF (Error IS NM) AND (ChangeInError IS NL) THEN (ChangeOfControl IS NL)
IF (Error IS NS) AND (ChangeInError IS NL) THEN (ChangeOfControl IS NLM)
IF (Error IS ZE) AND (ChangeInError IS NL) THEN (ChangeOfControl IS NM)
IF (Error IS PS) AND (ChangeInError IS NL) THEN (ChangeOfControl IS NMS)
IF (Error IS PM) AND (ChangeInError IS NL) THEN (ChangeOfControl IS NS)
IF (Error IS PL) AND (ChangeInError IS NL) THEN (ChangeOfControl IS ZE)
IF (Error IS NL) AND (ChangeInError IS NM) THEN (ChangeOfControl IS NL)
IF (Error IS NM) AND (ChangeInError IS NM) THEN (ChangeOfControl IS NLM)
IF (Error IS NS) AND (ChangeInError IS NM) THEN (ChangeOfControl IS NM)
IF (Error IS ZE) AND (ChangeInError IS NM) THEN (ChangeOfControl IS NMS)
IF (Error IS PS) AND (ChangeInError IS NM) THEN (ChangeOfControl IS NS)
IF (Error IS PM) AND (ChangeInError IS NM) THEN (ChangeOfControl IS ZE)
IF (Error IS PL) AND (ChangeInError IS NM) THEN (ChangeOfControl IS PS)
IF (Error IS NL) AND (ChangeInError IS NS) THEN (ChangeOfControl IS NLM)
IF (Error IS NM) AND (ChangeInError IS NS) THEN (ChangeOfControl IS NM)
IF (Error IS NS) AND (ChangeInError IS NS) THEN (ChangeOfControl IS NMS)
IF (Error IS ZE) AND (ChangeInError IS NS) THEN (ChangeOfControl IS NS)
IF (Error IS PS) AND (ChangeInError IS NS) THEN (ChangeOfControl IS ZE)
IF (Error IS PM) AND (ChangeInError IS NS) THEN (ChangeOfControl IS PS)
IF (Error IS PL) AND (ChangeInError IS NS) THEN (ChangeOfControl IS PMS)
IF (Error IS NL) AND (ChangeInError IS ZE) THEN (ChangeOfControl IS NM)
IF (Error IS NM) AND (ChangeInError IS ZE) THEN (ChangeOfControl IS NMS)
IF (Error IS NS) AND (ChangeInError IS ZE) THEN (ChangeOfControl IS NS)
IF (Error IS ZE) AND (ChangeInError IS ZE) THEN (ChangeOfControl IS ZE)
IF (Error IS PS) AND (ChangeInError IS ZE) THEN (ChangeOfControl IS PS)
IF (Error IS PM) AND (ChangeInError IS ZE) THEN (ChangeOfControl IS PMS)
IF (Error IS PL) AND (ChangeInError IS ZE) THEN (ChangeOfControl IS PM)
IF (Error IS NL) AND (ChangeInError IS PS) THEN (ChangeOfControl IS NMS)
IF (Error IS NM) AND (ChangeInError IS PS) THEN (ChangeOfControl IS NS)
IF (Error IS NS) AND (ChangeInError IS PS) THEN (ChangeOfControl IS ZE)
IF (Error IS ZE) AND (ChangeInError IS PS) THEN (ChangeOfControl IS PS)
IF (Error IS PS) AND (ChangeInError IS PS) THEN (ChangeOfControl IS PMS)
IF (Error IS PM) AND (ChangeInError IS PS) THEN (ChangeOfControl IS PM)
IF (Error IS PL) AND (ChangeInError IS PS) THEN (ChangeOfControl IS PLM)
IF (Error IS NL) AND (ChangeInError IS PM) THEN (ChangeOfControl IS NS)
IF (Error IS NM) AND (ChangeInError IS PM) THEN (ChangeOfControl IS ZE)
IF (Error IS NS) AND (ChangeInError IS PM) THEN (ChangeOfControl IS PS)
IF (Error IS ZE) AND (ChangeInError IS PM) THEN (ChangeOfControl IS PMS)
IF (Error IS PS) AND (ChangeInError IS PM) THEN (ChangeOfControl IS PM)
IF (Error IS PM) AND (ChangeInError IS PM) THEN (ChangeOfControl IS PLM)
IF (Error IS PL) AND (ChangeInError IS PM) THEN (ChangeOfControl IS PL)
IF (Error IS NL) AND (ChangeInError IS PL) THEN (ChangeOfControl IS ZE)
IF (Error IS NM) AND (ChangeInError IS PL) THEN (ChangeOfControl IS PS)
IF (Error IS NS) AND (ChangeInError IS PL) THEN (ChangeOfControl IS PMS)
IF (Error IS ZE) AND (ChangeInError IS PL) THEN (ChangeOfControl IS PM)
IF (Error IS PS) AND (ChangeInError IS PL) THEN (ChangeOfControl IS PLM)
IF (Error IS PM) AND (ChangeInError IS PL) THEN (ChangeOfControl IS PL)
IF (Error IS PL) AND (ChangeInError IS PL) THEN (ChangeOfControl IS PL)

Program for designing the Fuzzy Logic Controller

The program for designing the Fuzzy Logic Controller using the FIS editor in MATLAB/SIMULINK® is as follows:

[System]
Name='rules'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=49
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='Error'
Range=[-1 1]
NumMFs=7
MF1='NL':trapmf,[-1 -1 -0.8 -0.5]
MF2='NM':trimf,[-0.8 -0.5 -0.2]
MF3='NS':trimf,[-0.5 -0.2 0]
MF4='ZE':trimf,[-0.2 0 0.2]
MF5='PS':trimf,[0 0.2 0.5]
MF6='PM':trimf,[0.2 0.5 0.8]
MF7='PL':trapmf,[0.5 0.8 1 1]

[Input2]
Name='ChangeInError'
Range=[-1 1]
NumMFs=7
MF1='NL':trapmf,[-1 -1 -0.8 -0.5]
MF2='NM':'trimf',[-0.8 -0.5 -0.2]
MF3='NS':'trimf',[-0.5 -0.2 0]
MF4='ZE':'trimf',[-0.2 0 0.2]
MF5='PS':'trimf',[0 0.2 0.5]
MF6='PM':'trimf',[0.2 0.5 0.8]
MF7='PL':'trapmf',[0.5 0.8 1 1]

[Output1]
Name='ChangeOfControl'
Range=[-1 1]
NumMFs=11
MF1='NL':'trimf',[-1 -1 -0.8]
MF2='NLM':'trimf',[-1 -0.8 -0.6]
MF3='NM':'trimf',[-0.8 -0.6 -0.4]
MF4='NMS':'trimf',[-0.6 -0.4 -0.2]
MF5='NS':'trimf',[-0.4 -0.2 0]
MF6='ZE':'trimf',[-0.2 0 0.2]
MF7='PS':'trimf',[0 0.2 0.4]
MF8='PSM':'trimf',[0.2 0.4 0.6]
MF9='PM':'trimf',[0.4 0.6 0.8]
MF10='PML':'trimf',[0.6 0.8 1]
MF11='PL':'trimf',[0.8 1 1]

[Rules]
1 1, 1 (1) : 1
2 1, 1 (1) : 1
3 1, 2 (1) : 1
Simulink (Simulation and Link) is an extension of MATLAB by Mathworks Inc. It works with MATLAB to offer modeling, simulating, and analyzing of dynamical systems under a graphical user interface (GUI) environment. The construction of a model is simplified with click-and-drag mouse operations. Simulink includes a comprehensive block library of toolboxes for both linear and nonlinear analyses. Models are hierarchical, which allow using both top-down and bottom-up approaches. As Simulink is an integral part of MATLAB, it is easy to switch back and forth during the analysis process and thus, the user may take full advantage of features offered in both environments.

It is suitable to use version v.5 and above.

**Getting Started**

To start a Simulink session, first bring up the MATLAB program first.

From MATLAB command window, enter:

```
>> simulink
```

Alternately, click on the Simulink icon located on the toolbar as shown below.

![Simulink Icon](image)

This will open Simulink's library browser window like one shown below, which presents the block set for model construction.
To see the content of the blockset, click on the "+" sign at the beginning of each toolbox.

To start a model click on the “New File Icon” as shown in the above screen shot. Alternately, keystrokes “CTRL+N” may also be used.
A new window will appear on the screen. A model will be constructed in this window. Also in this window the constructed model is simulated. Screenshot of a typical working (model) window is shown below.

To become familiarized with the structure and the environment of Simulink, it is advisable to explore the toolboxes and scan their contents. These toolboxes may be unknown, but perhaps this will provide the information about the organization of these toolboxes according to the category. For instance, a Control System Toolbox consist of the Linear Time Invariant (LTI) system library and the MATLAB functions can be found under Function and Tables of the Simulink main toolbox.

A simple model is taken here to describe some basic features of Simulink. Following steps are used to construct a simple model.

**STEP 1 - CREATING BLOCKS:**

From Block Set Categories section of the SIMULINK Library Browser window, click on the "+" sign next to the Simulink group to expand the tree and select (click on) Sources.
A set of blocks will appear in the Block Set group. Click on the Sine Wave block and drag it to the workspace window (also known as model window).
Now, the source of the model has been established.

**NOTE:** It is advisable that to save the model at some point early on, so that if the PC crashes, then time will not lost in reconstructing the model.

Next save this model under the filename “simexample1” by clicking on the floppy diskette icon or by selecting the **Save from the File menu** or by using keystrokes “CTRL+S”. All Simulink model file will have an extension ".mdl". Simulink recognizes file with .mdl extension as a simulation model (similar to how MATLAB recognizes files with the extension .m as an MFile).

Further build the model by adding more components (or blocks) to the model window. Continue to add a **Scope** from **Sinks** library, an **Integrator** block from **Continuous** library, and a **Mux** block from **Signal Routing** library.

**NOTE:** By entering the name of the block in the SEARCH WINDOW (at **Find** prompt) a desired block can be find.

To move the blocks around, simply click on it and drag it to a desired location.

Once all the blocks are dragged over to the work space, it will consist of the following components:
A block can be removed (deleted) by simply clicking on it once to turn on the "select mode" (with four corner boxes) and use the DEL key or keys combination CTRL-X.

**STEP 2 - MAKING CONNECTIONS:**

To establish connections between the blocks, move the cursor to the output port represented by ">" sign on the block. Once placed at a port, the cursor will turn into a cross "+" enable it to make the connection between blocks.

To make a connection: left-click while holding down the control key (on your keyboard) and drag from source port to a destination port.

The connected model is shown below.

A sine signal is generated by the Sine Wave block (a source) and is displayed by the scope. The integrated sine signal is sent to scope for display along with the original signal from the source via the **Mux**, whose function is to multiplex signals in form of scalar, vector, or matrix into a bus.
STEP 3 - RUNNING SIMULATION:

Run the simulation of the simple system above by clicking on the play button ( ), alternatively, you may use key sequence CTRL+T, or choose Start submenu under Simulation menu).

Double click on the Scope block to display the scope.

To view/edit the parameters, simply double click on the block of interest.