Estimation of Fundamental Harmonics using Neural Network

Project report submitted in partial fulfillment
of the requirements for the degree of

Bachelor of Technology
in
Electronics and Communication Engineering

by

Barnabh Chandra Goswami – 19UEC161
Ayush Maherchandani – 19UEC160

Under the Guidance of
Dr. Vinay Kumar Tiwari

Department of Electronics and Communication Engineering
The LNM Institute of Information Technology, Jaipur

December 2022

The LNM Institute of Information Technology

Jaipur, India

CERTIFICATE

This is to certify that the project entitled "**Estimation of Fundamental Harmonics using Neural Network**" , submitted by **Barnabh Chandra Goswami (19UEC161)** and **Ayush Maherchandani (19UEC160)** in partial fulfillment of the requirement of degree in Bachelor of Technology (B. Tech), is a bonafide record of work carried out by them at the Department of Electronics and Communication Engineering, The LNM Institute of Information Technology, Jaipur, (Rajasthan) India, during the academic session 2022-2023 under my supervision and guidance and the same has not been submitted elsewhere for award of any other degree. In my/our opinion, this report is of standard required for the award of the degree of Bachelor of Technology (B. Tech).

Date: November 17, 2022 Advisor: Dr. Vinay Kumar Tiwari

Acknowledgements

Without our fellow batchmates in our project, we would not have been able to complete it. We would like to express our gratitude for their selfless cooperation with us. First and foremost, we would like to express our heartfelt gratitude to our mentor, the distinguished Dr. Vinay Kumar Tiwari, who came up with the idea of having such a technique for extracting harmonics using neural network on hardware under whose direction we were able to complete the designing of the project. We're also grateful to our college for providing us with a stress free and positive environment.

I would also like to acknowledge with much appreciation the crucial role of the staff in Electronics Laboratory, who gave me a permission to use the lab equipment and also the machine and to design the drawing and giving a permission to use all the necessary tools in the laboratory.

We owe a debt of gratitude to everyone participating in this project because it would not have been possible to complete the project on time without their inspiration and insightful suggestions.

Last, but not the least, my parents are also important inspiration for me. So, with due regards, I express my gratitude to them.

Abstract

In this BTP (B. Tech Project) we are using a random wave which contains the Harmonics and try to extract the odd Harmonics from the wave using the Neural Network, we will first work on the software implementation of this, then try to achieve this result on the hardware platform using a Virtex ML 605 FPGA board because the hardware prototype doesn't support the non-linear Activation Functions unlike the software prototype.

We will try to find a method so that we can successfully apply it on hardware prototype, we will train our Neural Network in MATLAB (or any other platforms according to our feasibility) such that it can identify the Harmonics in the wave, after that we will take the weights which we will obtained from the Neural Network and implement it on hardware prototype.

Our Neural Network consists of 20 Input Neurons in Input Layer, Hidden Layer and 8 Output Neurons in Output Layer. After getting weights from the trained Neural Network, we will provide the data to the manually made Neural Network in System Generator (a tool used for establishing a connection between PC and the FPGA's).

We will first work on the simple stationary wave having all the parameters constant and noise in it like Additive White Gaussian Noise (AWGN), Laplacian Noise, Simple Noise etc. After that we will slowly introduce the noise and varying parameters in them, also in the Harmonics part, we will first use the first, third, fifth and seventh Harmonics for the simplicity in logics. After that we will increase the number of odd Harmonics in the wave up to fifteenth Harmonics.

At last, the aim of this project is to make hardware design on Virtex ML 605 FPGA board such that when we provide an input wave to the hardware model it detects the fundamental Harmonics in it using Neural Network Algorithm.

Table of Contents

List of Figures

List of Tables

Tools and Software used

The tools and software which we have used in this project are as follows:

1. MATLAB (2015b)
2. Vivado (2016.3)
3. Vivado HLS (2016.3)
4. System Generator (2016.3)
5. Virtex ML 605 FPGA board

1 Chapter 1:

1.1 Introduction

Currently, neural networks (NN) are transforming industry and daily life and advancing artificial intelligence (AI). NN-enabled machines, including as the smartphones and computers we use every day, are now taught to learn, detect patterns, and make predictions in a human-like manner, as well as solve issues in every business sector, by simulating the way interconnected brain cell's function.

Artificial neural networks (ANN), a subset of machine learning's neural networks (NN), are computer models that are simply algorithms. In order to locate patterns and identify trends in data that is too complicated for the human brain or other computer systems to process, neural networks have the unique capacity to extract meaning from ambiguous or complex data. We now enjoy more convenience thanks to neural networks in a variety of ways, such as ridesharing apps, Gmail smart sorting, and Amazon suggestions. The most revolutionary feature of neural networks is their ability to learn on their own after being trained. In doing so, they mimic human brains, which are composed of neurons, the basic unit of information transmission in both neural networks and human brains.

The harmonics in waves that are present in any system where standing waves can develop have a variety of natural frequencies, and we will use this kind of aspect to locate them. The Harmonics of a system are the set of all feasible standing waves. The fundamental or first harmonic is the most basic of the harmonics. The second Harmonic, third Harmonic, etc. are the names given to succeeding standing waves. Particularly in music theory, the harmonics above the fundamental are sometimes referred to as overtones.

Whereas in terms of power systems, a sinusoidal wave whose frequency is an integer multiple of the fundamental frequency is referred to as a harmonic of a voltage or current waveform. Non-linear loads like rectifiers, discharge lamps, or saturated electric machinery emit harmonic frequencies. In addition to increasing equipment and conductor heating, misfiring in variable speed drives, and torque pulsations in motors and generators are also possible effects of them. They are a common source of power quality issues.

2 Chapter 2:

2.1 Theory

We will first discuss some concepts which we needed to understand before heading to our aim of the project. Like some basic principles working behind neural networks, stationary waves or standing waves, non-stationary waves, harmonics in waves, System Generator, Methods of extracting Harmonics from the wave etc.

2.2 Neural Networks

2.2.1 What are Neural Networks?

A neural network is an interconnected collection of discrete processing "nodes," or units, whose operation is somewhat analogous to that of an animal neuron. The interunit connection strengths, or weights, acquired through a process of adaptation to, or learning from, a set of training patterns, are where the network's processing power is kept.

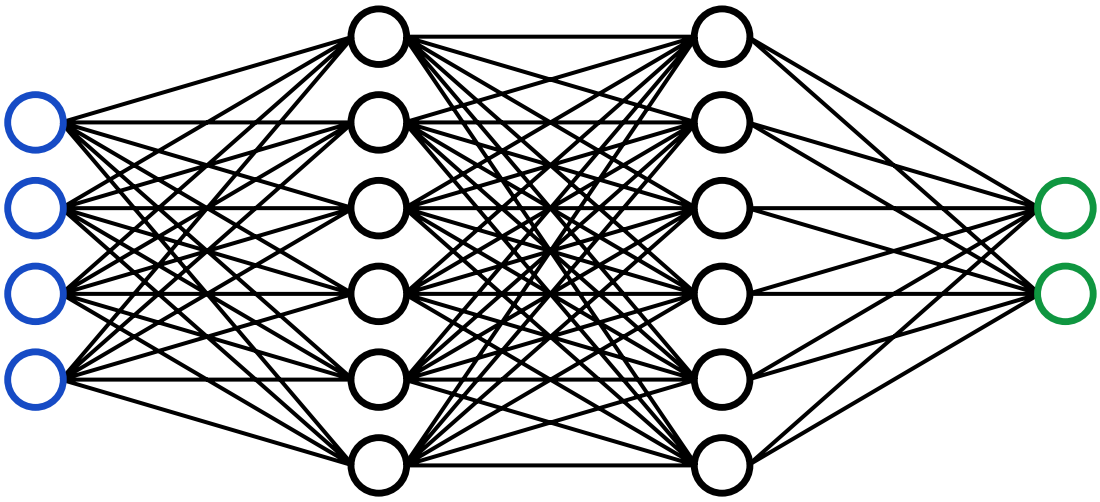

Figure 1: Basic structure of a Neural Network.

2.2.2 What are neural networks used for?

Neural networks have several use cases across many industries, such as the following:

1. Computer vision (Important)
2. Speech recognition (Important)
3. Natural language processing (Important)
4. Medical diagnosis by medical image classification
5. Targeted marketing by social network filtering and behavioral data analysis
6. Financial predictions by processing historical data of financial instruments
7. Electrical load and energy demand forecasting
8. Process and quality control
9. Chemical compound identification

2.2.3 How do Neural Networks work?

The human brain is the inspiration behind neural network architecture. Human brain cells, called neurons, form a complex, highly interconnected network and send electrical signals to each other to help humans process information. Similarly, an artificial neural network is made of artificial neurons that work together to solve a problem. Artificial neurons are software modules, called nodes, and artificial neural networks are software programs or algorithms that, at their core, use computing systems to solve mathematical calculations.

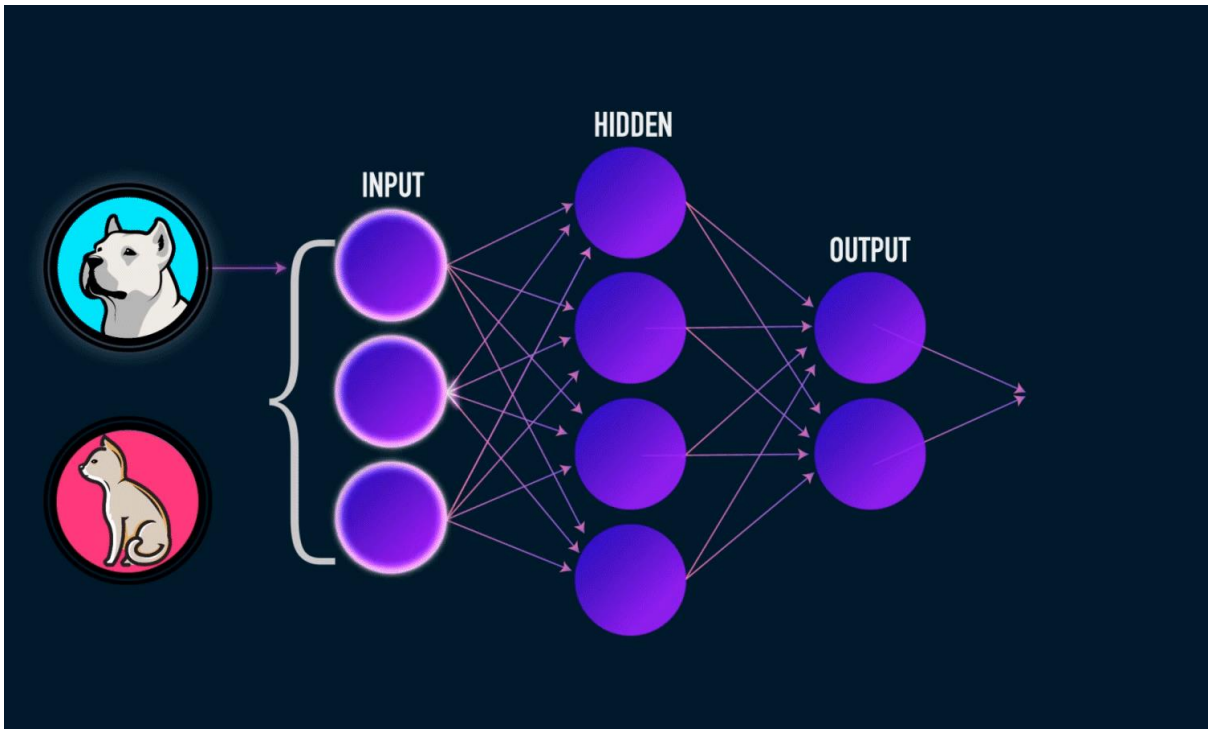

Figure 2: Working of Neural Network.

Simple neural network architecture

A basic neural network has interconnected artificial neurons in three layers:

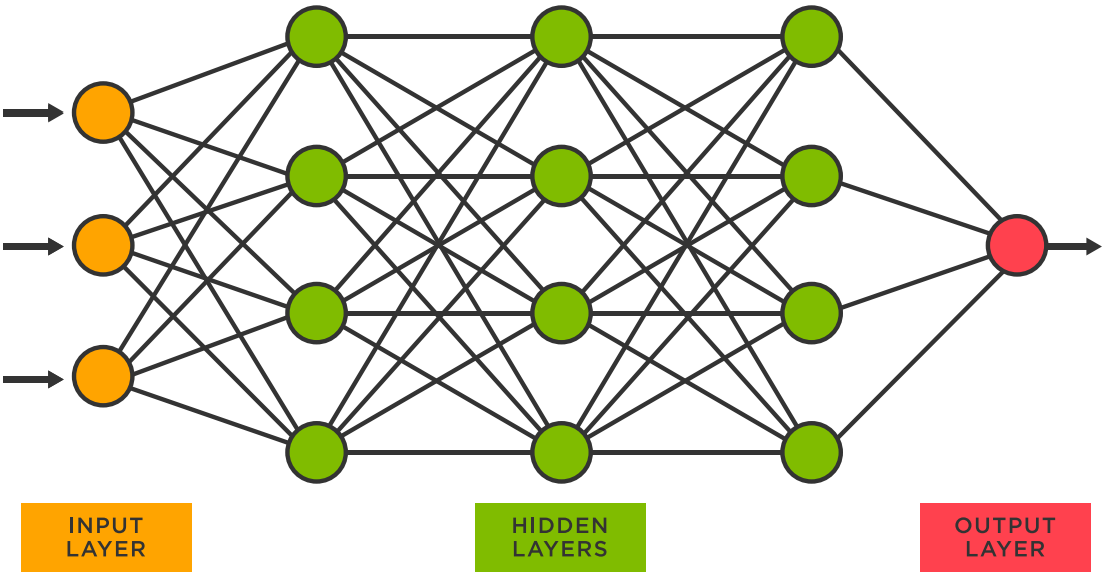

Figure 3: Visualising the Input, Hidden and Output Layer of the Neural Network.

- **Input Layer:** Information from the outside world enters the artificial neural network from the input layer. Input nodes process the data, analyze or categorize it, and pass it on to the next layer.
- **Hidden Layer:** Hidden layers take their input from the input layer or other hidden layers. Artificial neural networks can have a large number of hidden layers. Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.
- **Output Layer:** The output layer gives the final result of all the data processing by the artificial neural network. It can have single or multiple nodes. For instance, if we have a binary (yes/no) classification problem, the output layer will have one output node, which will give the result as 1 or 0. However, if we have a multi-class classification problem, the output layer might consist of more than one output node.

2.2.4 What are the types of Neural Networks?

Artificial neural networks can be categorized by how the data flows from the input node to the output node. Below are some examples:

- Feedforward neural networks
- Backpropagation algorithm

- Convolutional neural networks

In this project we have used the Backpropagation algorithm so let's have a slight look on it.

Backpropagation Algorithm

Artificial neural networks learn continuously by using corrective feedback loops to improve their predictive analytics. In simple terms, you can think of the data flowing from the input node to the output node through many different paths in the neural network. Only one path is the correct one that maps the input node to the correct output node. To find this path, the neural network uses a feedback loop, which works as follows:

1. Each node makes a guess about the next node in the path.
2. It checks if the guess was correct. Nodes assign higher weight values to paths that lead to more correct guesses and lower weight values to node paths that lead to incorrect guesses.
3. For the next data point, the nodes make a new prediction using the higher weight paths and then repeat Step 1.

For more details about BPNN (Back Propagation Neural Network) you may refer to the YouTube videos which are referenced in the Bibliography section.

2.2.5 Activation Functions (Transformation Functions or Squashing Functions)

An Activation Function decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations. The role of the Activation Function is to derive output from a set of input values fed to a node (or a layer).

The purpose of an activation function is to add non-linearity to the neural network. It brings down the data value between 0 and 1.

Types of Activation Functions:

- Binary Step
- Linear
- Sigmoid
- Tanh
- ReLU (Rectified Linear Unit)
- Leaky ReLU
- Parameterized ReLU
- Exponential Linear Unit
- Swish
- SoftMax

Depending upon the properties of the problem we might be able to make a better choice for easy and quicker convergence of the network.

- Sigmoid functions and their combinations generally work better in the case of classifiers.
- Sigmoid and Tanh functions are sometimes avoided due to the vanishing gradient problem.
- ReLU function is a general activation function and is used in most cases these days.
- If we encounter a case of dead neurons in our networks the leaky ReLU function is the best choice.
- Always keep in mind that ReLU function should only be used in the hidden layers.
- As a rule of thumb, you can begin with using ReLU function and then move over to other activation functions in case ReLU doesn't provide with optimum results.

2.2.6 Learning Rate (α)

Learning Rate is a hyper-parameter that controls the weights of our neural network with respect to the loss gradient. It defines how quickly the neural network updates the concepts it has learned.

A desirable learning rate is low enough that the network converges to something useful, but high enough that it can be trained in a reasonable amount of time.

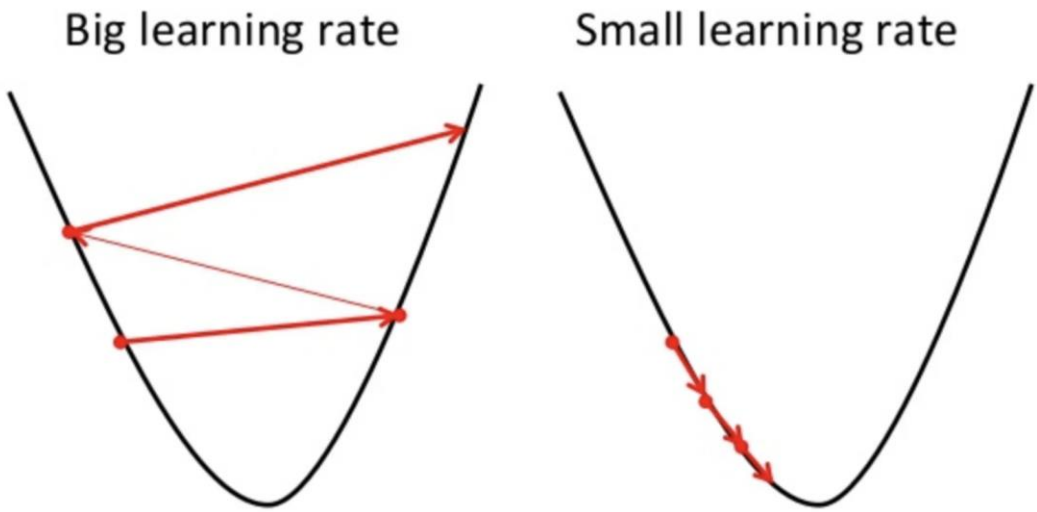

Figure 4: A small learning rate makes the model converge slowly to the global minimum loss.

Smaller learning rates require more training epochs (requires more time to train) due to the smaller changes made to the weights in each update, whereas larger learning rates result in

rapid changes and require fewer training epochs. However, larger learning rates often result in a sub-optimal final set of weights.

2.2.7 Gradient Descent

Gradient Descent is defined as one of the most commonly used iterative optimization algorithms of machine learning to train the machine learning and deep learning models. It helps in finding the local minimum of a function.

The best way to define the local minimum or local maximum of a function using gradient descent is as follows:

- If we move towards a negative gradient or away from the gradient of the function at the current point, it will give the local minimum of that function.
- Whenever we move towards a positive gradient or towards the gradient of the function at the current point, we will get the local maximum of that function.

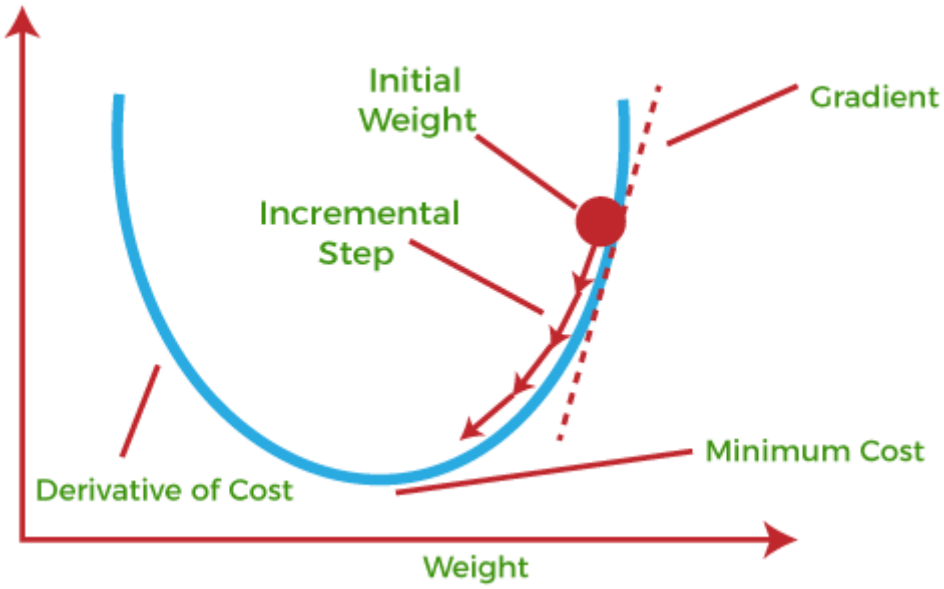

Figure 5: Understanding the Gradient Descent.

This entire procedure is known as Gradient Ascent, which is also known as steepest descent. The main objective of using a gradient descent algorithm is to minimize the cost function using iteration.

2.2.8 Vanishing Gradient Problem

The sigmoid function is one of the most popular activations functions used for developing deep neural networks. The use of sigmoid function restricted the training of deep neural networks because it caused the vanishing gradient problem. This caused the neural network to learn at a slower pace or in some cases no learning at all. This blog post aims to describe the vanishing gradient problem and explain how use of the sigmoid function resulted in it.

For the nodes with sigmoid activation functions, we know that the partial derivative of the sigmoid function reaches a maximum value of 0.25. When there are more layers in the network, the value of the product of derivative decreases until at some point the partial derivative of the loss function approaches a value close to zero, and the partial derivative vanishes. We call this the vanishing gradient problem.

When it comes to deep networks, the vanishing gradient could have a significant impact on performance. The weights of the network remain unchanged as the derivative vanishes. During back propagation, a neural network learns by updating its weights and biases to reduce the loss function. In a network with vanishing gradient, the weights cannot be updated, so the network cannot learn. The performance of the network will decrease as a result.

2.2.9 Exploding Gradient Problem

The exploding gradient is the inverse of the vanishing gradient and occurs when large error gradients accumulate, resulting in extremely large updates to neural network model weights during training. As a result, the model is unstable and incapable of learning from your training data.

2.3 Stationary Waves

A standing wave, also known as a stationary wave, is a wave that oscillates in time but whose peak amplitude profile does not move in space. The peak amplitude of the wave oscillations at any point in space is constant with respect to time, and the oscillations at different points throughout the wave are in phase. The locations at which the absolute value of the amplitude is minimum are called nodes, and the locations where the absolute value of the amplitude is maximum are called antinodes.

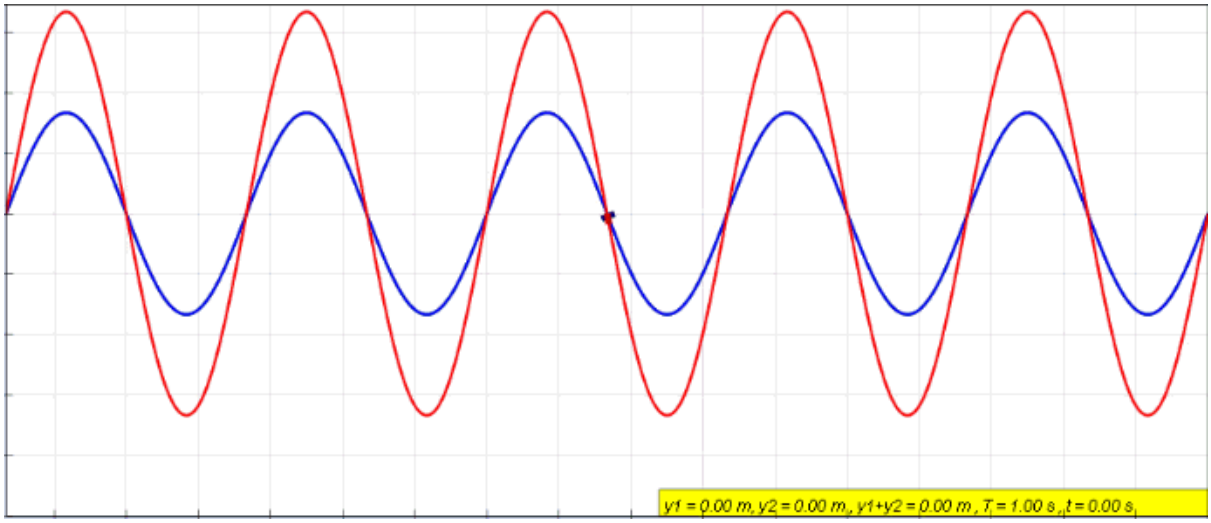

Figure 6: Animation of a standing wave (red) created by the superposition of a left traveling (blue) and right traveling (green) wave.

This phenomenon can occur because the medium is moving in the direction opposite to the movement of the wave, or it can arise in a stationary medium as a result of interference between two waves traveling in opposite directions. The most common cause of standing waves is the phenomenon of resonance, in which standing waves occur inside a resonator due to interference between waves reflected back and forth at the resonator's resonant frequency.

2.4 Non-Stationary Waves

A signal is said to be non-stationary if one of these fundamental assumptions is no longer valid. For example, a finite duration signal, and in particular a transient signal (for which the length is short compared to the observation duration), is non-stationary.

2.5 Harmonics in Wave

In an electric power system, a harmonic is a voltage or current at a multiple of the fundamental frequency of the system. Harmonics can best be described as the shape or characteristics of a voltage or current waveform relative to its fundamental frequency. When waveforms deviate from a sinewave shape, they contain harmonics.

Harmonic frequencies in the power grid are a frequent cause of power quality problems, resulting in increased heating in the equipment and conductors, misfiring in variable speed drives and torque pulsations in motors.

2.5.1 What are the consequences of harmonics?

A power system's ability to perform at optimal levels is compromised when harmonic distortion enters the system. Harmonics create inefficiencies in equipment operations due to the increased need for power consumption. The increase of overall current required creates higher installation and utility costs, overheating and decreased profitability.

2.5.2 Odd Harmonics

Odd harmonics are harmonics in which frequencies are odd numbers such as 150, 250, 350 Hz, etc. in the fundamental frequency of 50 Hz.

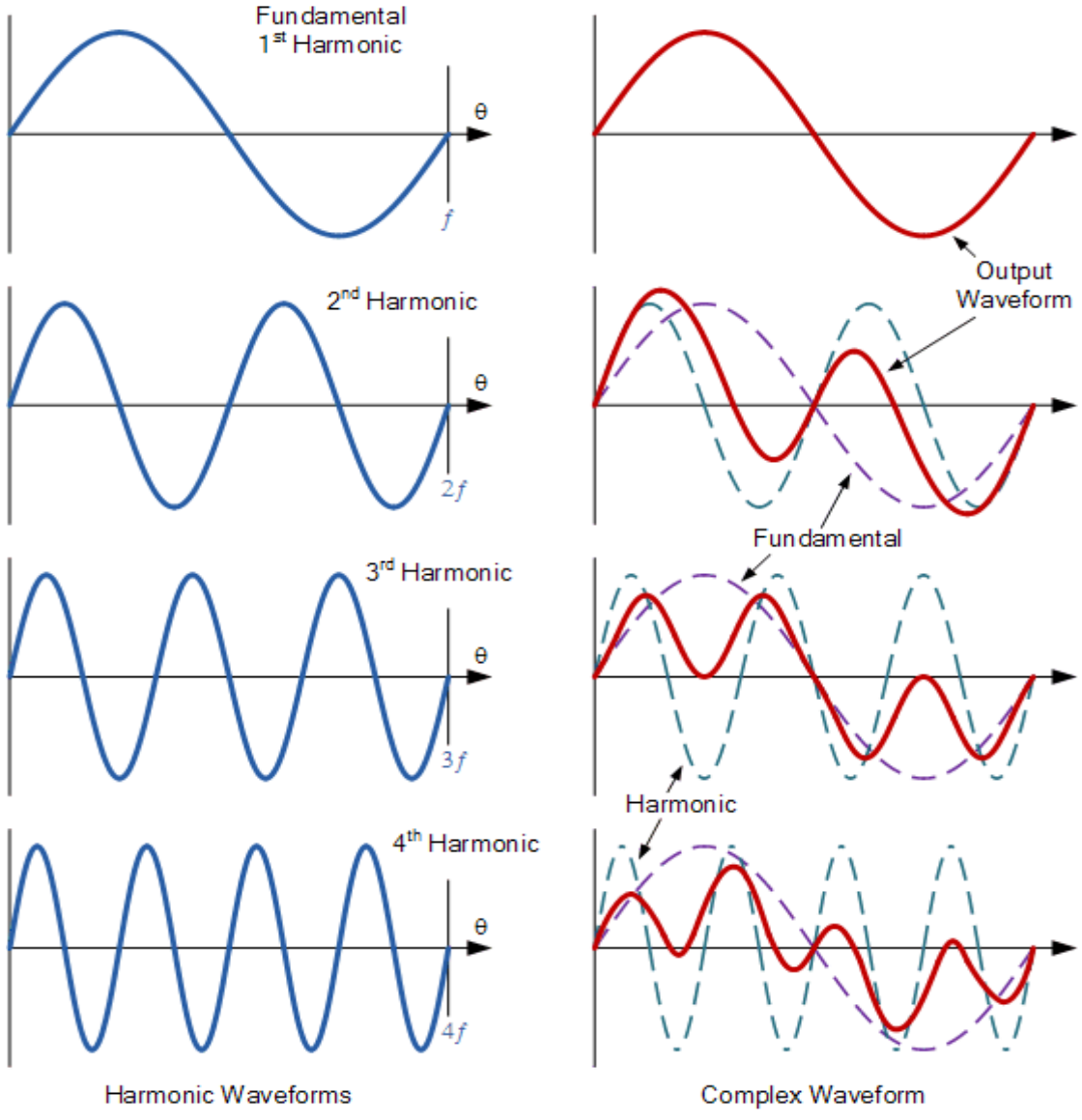

Figure 7: Complex waveforms produced due to combination of different Harmonics.

2.5.3 Methods of extracting Harmonics from the wave

In this project we mainly focused on the detection of Harmonics from the wave rather than extraction, so for detecting the Harmonics we used:

- Fourier Transform Method: Discussed in detail in section 3.2.
- Hit and Trial Method: Discussed in detail in section 3.2.

2.6 MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modelling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

2.7 System Generator

System Generator for DSP is a design tool in the Vivado® Design Suite that enables you to use the MathWorks® model-based Simulink® design environment for FPGA design. Previous experience with Xilinx® FPGA devices or RTL design methodologies is not required when using System Generator. Designs are captured in the Simulink™ modelling environment using a Xilinx-specific block set. Downstream FPGA steps including RTL synthesis and implementation (where the gate level design is placed and routed in the FPGA) are automatically performed to produce an FPGA programming bitstream.

3 Chapter 3:

3.1 Proposed work

As we have already discussed about the theory which tells us about the knowledge which we must require for avoiding the doubts and future problems regarding the concepts of our project.

Now in this section we will described that what we have done so far from the starting to end. How we started? How we make it simple at first then slowly added complexity in it? How did we tackle the problems we faced?

3.1.1 Timeline of our work

Here is the timeline of our progress and work from February 2022 to November 2022.

Progress	**Month**
Studied about Neural Networks.	March 2022
Description	
Started learning the course “Deep Learning” to get to know about Neural Networks and its working in detail, we use the book of Prof. Simon Haykin, watch the videos on YouTube, read through the websites, all of the links are referenced in Bibliography section. You can also see the basics in the section 2.2.	
Studied about Vivado Design Suite using System Generator.	April 2022
Description	
We have learnt how the System Generator application works, for this we downloaded the tutorial guide from the [11] (referenced in Bibliography Section) and then study the configuration of MATLAB to the Vivado Design Suite, locating and preparing the Tutorial Design Files, creating a Design in an FPGA, work with Data types, Multi-Rate Systems, Workspace Variables etc., modelling blocks with C Code, M Code and HDL. You can also get the basic info in section 2.7 and images related to it in section 4.1.1.	
Getting ideas from our Advisor’s research paper.	May 2022

Description	
Read the research paper of Dr. Vinay Kumar Tiwari from the [12] (referenced in Bibliography Section). Where the research paper is about "Hardware Implementation of Polyphase-Decomposition-Based Wavelet Filters for Power System Harmonics Estimation".	
Using more research papers for the getting the vast idea of our project.	May 2022
Description	
Read the research papers of Mohammed Bahoura, Chan-Wang Park, Hassan Ezzaidi and Hsiung Cheng Lin from the links [13], [14], [15] and [16] (referenced in Bibliography Section).	
Learnt about Stationary waves and non-Stationary waves.	June 2022
Description	
Understood the concepts of Stationary waves and non-Stationary waves from the websites [7] and [8] (referenced in Bibliography Section). You can see the basics in section 2.3 and 2.4.	
Studied about Harmonics in power systems and waves.	July 2022
Description	
Understood the concepts of Harmonics in different domains with the help of websites [9] and [10]. You can see all the basics in section 2.5.	
Learnt about the methods of extracting and detecting the harmonics in waves.	August 2022
Description	
First method is about the Fourier Transform method or rather we could say Fast Fourier Transform method, in which we tried to find out the FFT (Fast Fourier Transform) of the input wave which is the combination of many Harmonics, since from FFT we can detect the frequency components in the wave and can tell which multiple of the fundamental Harmonic is present in it therefore, we can detect the Harmonics in the wave. You can see the images related to it in the section 4.1.3 and 4.1.4. Second method is what we called Hit and Trial method in which we introduced the Harmonics in the input wave and then tried to train it according to which Harmonic component we want in that particular section of the input wave, since in this case our outputs	

and inputs are already known therefore, we only have to train the Neural Network just by giving these inputs and outputs. From this we can detect the Harmonics in the wave. You can see the images related to it in the section 4.1.6.	
Using MATLAB and producing Harmonics up to 7th and then increase up to 15th.	August 2022
Description	
We wrote the code in MATLAB in which we introduced the Harmonics in the input wave which consists of 1^{st}, 3^{rd}, 5^{th} up to 15^{th} multiple of fundamental Harmonics and simply plot its graph to get the idea about how the mixed Harmonic wave looks like. You can see the images related to it in the section 4.1.2.	
Tried to use the Neural Network in MATLAB by nntool inbuilt-toolbox and then by manual coding in MATLAB.	September 2022
Description	
We have learnt 2 types of ways to use the Neural Network in MATLAB, first we used the inbuilt tool box named as "nntool" and tried to train some random networks to see the results and accuracy then we make the neural network by manual coding in MATLAB, which was more user friendly because we can manipulate many parameters if we use the manual coding while in nntool box we have limited set of parameters to change. You can see the images related to it in the section 4.1.4.	
Tried to introduce the noise in the wave to increase more practicality.	September 2022
Description	
Now since we can handle basic operations on the input wave therefore to increase more practicality from ideality, we assigned noise to it, which can be AWGN, or random noise according to our choice. You can see the images related to it in the section.	
Applying FFT method to detect the Harmonics in the wave.	September 2022
Description	
In the MATLAB we used the FFT function to find the frequency components in our input wave and then tried to train the neural network using FFT to check the results but after completion of the training we saw that results were not accurate. On further observations we came to know that FFT method gives only up to 90% accuracy so it was logical that Neural	

Network design through this logic will always gave the accuracy below 90%. Also, we noted that FFT method can only give such results in stationary waves. It is not recommended for non-stationary waves. Therefore, we decided to discard this method. You can see the images related to it in the section 4.1.4.	
Tried to design manually the neural network in System Generator.	October 2022
Description	
After this we are going to design a hardware based neural network manually using system generator to finally implement it on the VIRTEX ML 605 board. In this model we are going to implement the hyperbolic function using a LookUp Table (LUT) and also updating the weights using the weight function $w(n+1) = w(n) + \Delta w(n)$ (where $w(n)$ is the weight function and $\Delta w(n)$ is the difference between 2 weight functions). You can see the images related to it in the section 4.1.5.	
Applying Hit and Trial method to detect the Harmonics in the wave and checking its accuracy.	October 2022
Description	
Created a wave of duration 200s in MATLAB with odd Harmonics then we sampled the wave (total 10,000 samples) after that we varied the amplitude of every harmonic at an interval of 0.02s by multiplying random integers ranging from 1 to 5 and then sampled the wave in such a way so that we can have 20 samples per division, each of this sample is our input. Then we took the amplitude of each Harmonic as output which are in total 8, then we trained our neural network using these 20 inputs and 8 outputs. At last, this makes our Neural Network having 20 Neurons in our input layer, 8 Neurons in output layer and we can adjust the number of neurons in hidden layer according to our feasibility. Here we take the simplest case of detecting the Harmonics from waves, where we didn't include any kind of noise, took the amplitude of each sample to be constant and used total 8 odd multiples of fundamental Harmonics i.e., 1^{st}, 3^{rd}, 5^{th}, 7^{th}, 9^{th}, 11^{th}, 13^{th} and 15^{th}. You can see the images related to it in the section 4.1.6.	
Trained the neural network and take the weights of each of the nodes from the neural network.	November 2022
Description	

We made the matrices of “Input Data” of [20 x 10000] and “Output Data” of [8 x 10000], where we trained the Neural Network and get the “Neural Network Output” matrix of [8 x 10000]. Here we took 10000 observations, for increasing accuracy we can increase number of epochs according to our feasibility. After training the Neural Network using Hit and Trial method for detection of Harmonics, we will record the weights used by each neuron, which will used in feeding the weights to the Neural Network made in the System Generator. You can see the images related to it in the section 4.1.6.	
Used the weights to train the manually created model in System Generator.	November 2022
Description	
The model which we created on the system generator (), we are going to manually set the weights of each of the nodes to replicate the neural network where we took 70% data as training set and 30% data for testing from the dataset which was created in the MATLAB () then we will implement that model on the VIRTEX ML 605 FPGA board to get a working model of the neural network which will tell us the amplitude of each of the harmonic. In this model we can take input directly from the power supply.	
Making the BTP report	November 2022
Description	
Finally making the report to record all the progress, simulations, results etc.	

Table 1: Timeline of the BTP progress.

4 Chapter 4:

4.1 Simulations and Results

In this section, we will show our codes done in the MATLAB and graphs we plotted for observation and all the other images about our work.

4.1.1 Creating a basic model for FPGA in System Generator

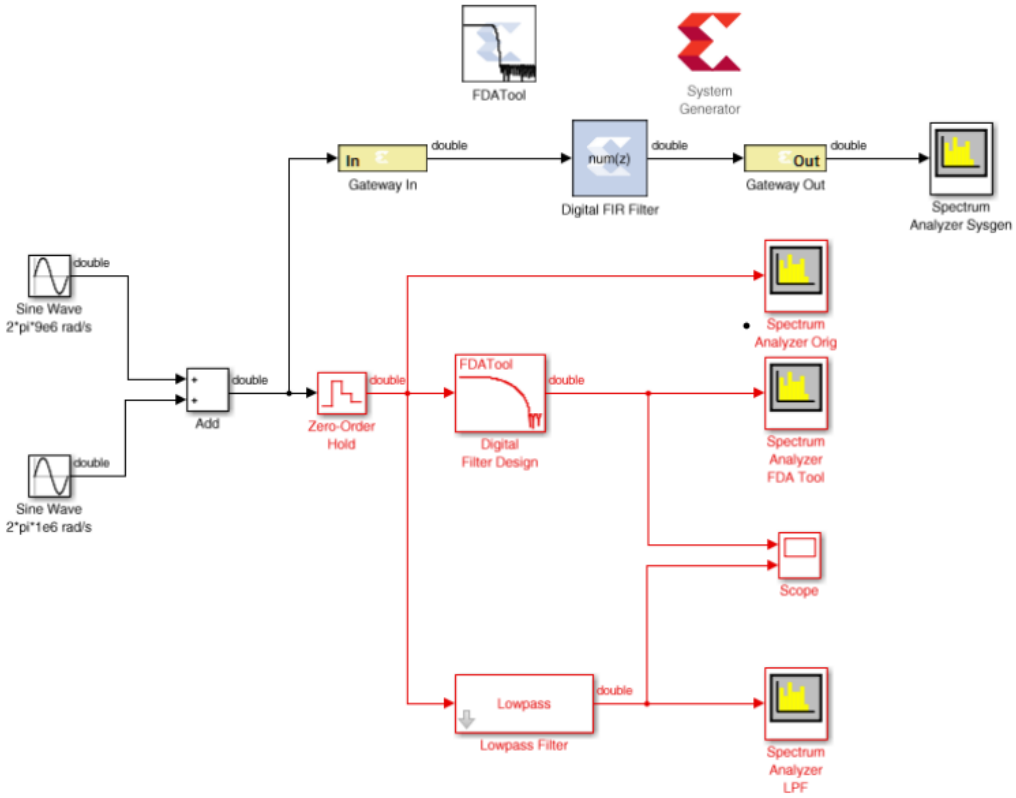

Figure 8: Initial Design for FPGA in System Generator.

In this design, it shows two sine wave sources being added together and passed separately through two low-pass filters. This design highlights that a low-pass filter may be implemented using the Simulink FDA Tool or Lowpass Filter blocks.

This is the basic operation of System Generator and how to synthesize a Simulink design into an FPGA.

4.1.2 Combining Harmonics up to 15th Multiple

Code

```
clc;
clear all;
close all;

fs = 10000; % Sampling Frequency
ts = 1/fs; % Sampled Time
t = 0:ts:0.1; % Making intervals of the wave or sampling the wave.
f = 50; % Fundamental Frequency = 50 Hz.
h1 = 1; % 1st Harmonic
h3 = 3; % 3rd Harmonic
h5 = 5; % 5th Harmonic
h7 = 7; % 7th Harmonic
h9 = 9; % 9th Harmonic
h11 = 11; % 11th Harmonic
h13 = 13; % 13th Harmonic
h15 = 15; % 15th Harmonic

v1 = 1*sin(2*pi*h1*f*t);
v3 = 1*sin(2*pi*h3*f*t);
v5 = 1*sin(2*pi*h5*f*t);
v7 = 1*sin(2*pi*h7*f*t);
v9 = 1*sin(2*pi*h9*f*t);
v11 = 1*sin(2*pi*h11*f*t);
v13 = 1*sin(2*pi*h13*f*t);
v15 = 1*sin(2*pi*h15*f*t);

v_sum = v1 + v3 + v5 + v7 + v9 + v11 + v13 + v15;

% Plotting the Graphs of Waves.

figure (1)
subplot (2, 2, 1)
plot (t, v1)
title('1^{st} Fundamental Harmonic')
xlabel('Time')
ylabel('v_1')

subplot (2, 2, 2)
plot (t, v3)
title('3^{rd} Fundamental Harmonic')
xlabel('Time')
ylabel('v_3')

subplot (2, 2, 3)
plot (t, v5)
title('5^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_5')

subplot (2, 2, 4)
plot (t, v7)
title('7^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_7')

suptitle('1^{st}, 3^{rd}, 5^{th} and 7^{th} Harmonics') % We can use sgtitle also but it comes
in the 2018th version of MATLAB.
```

```
figure (2)
subplot (2, 2, 1)
plot (t, v9)
title('9^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_9')

subplot (2, 2, 2)
plot (t, v11)
title('11^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_11')

subplot (2, 2, 3)
plot (t, v13)
title('13^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_13')

subplot (2, 2, 4)
plot (t, v15)
title('15^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_15')
ylabel('v_7')

suptitle('9^{th}, 11^{th}, 13^{th} and 15^{th} Harmonics')


figure (4)
plot (t, v_sum)
title('Overall sum of the wave containing 1st, 3rd, 5th, 7th, 9th, 11th, 13th and 15th
Harmonics')
xlabel('Time')
ylabel('v_{sum}')
```

Graphs

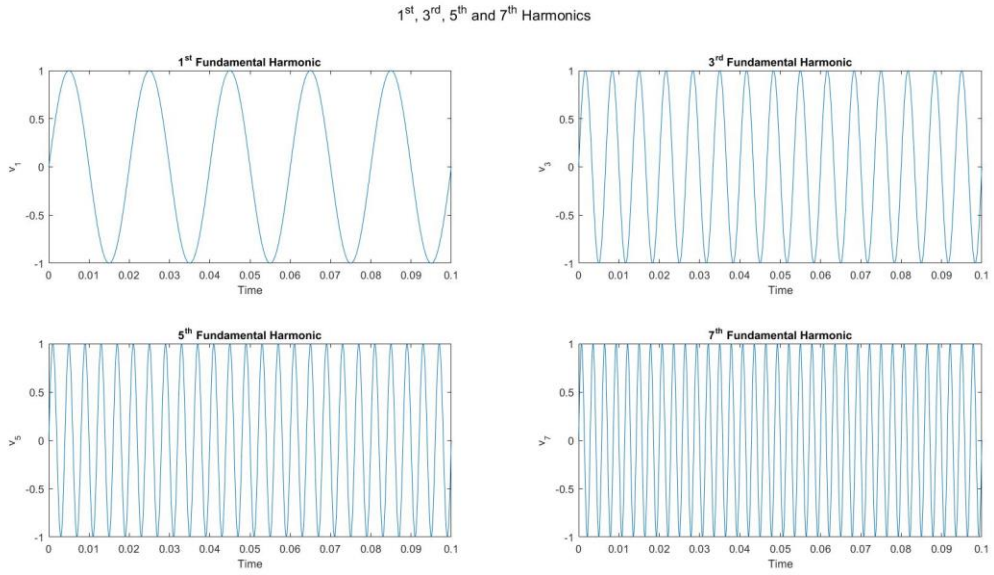

Figure 9: 1st, 3rd, 5th and 7th Fundamental Harmonics.

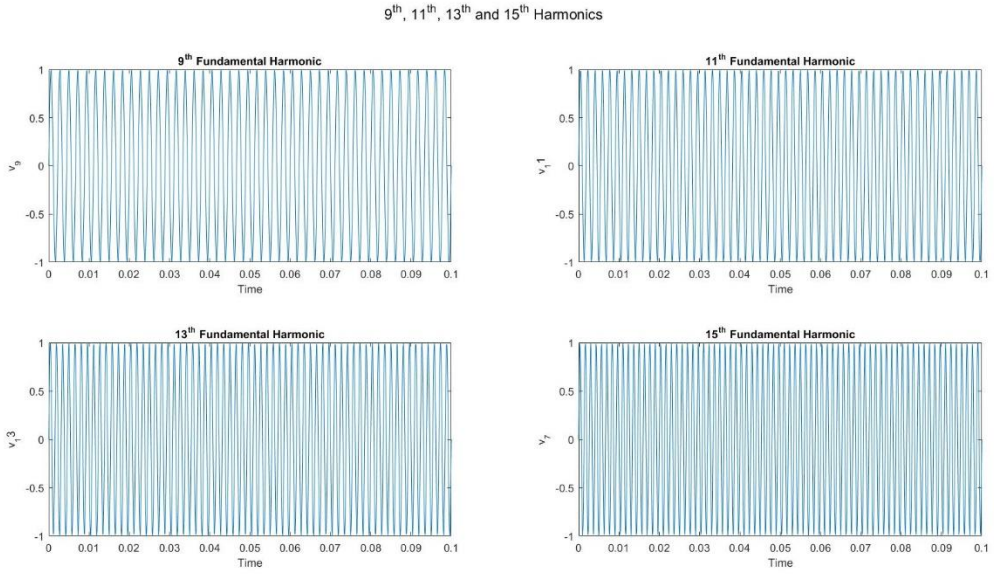

Figure 10: 1st, 3rd, 5th and 7th Fundamental Harmonics.

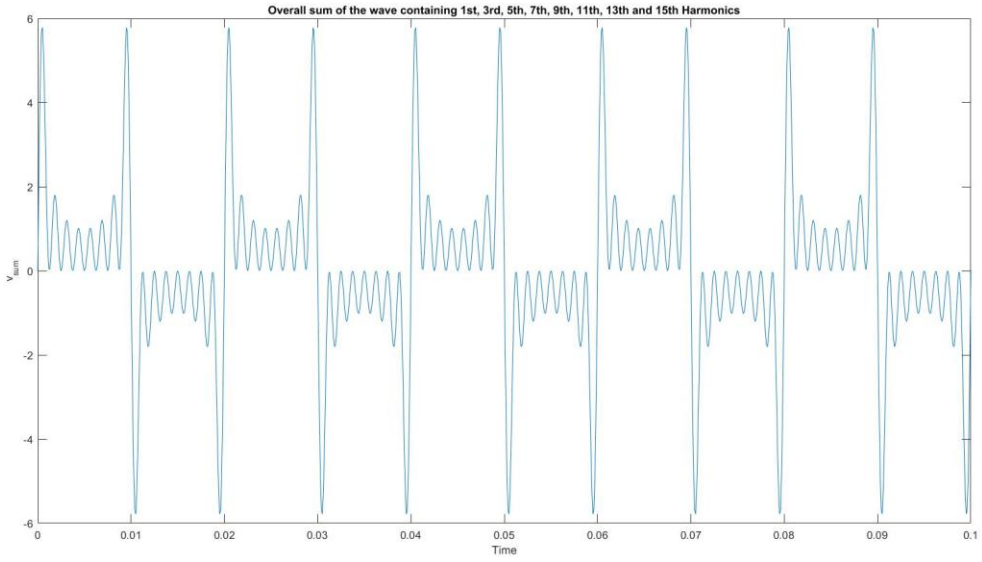

Figure 11: Overall final wave combining all the odd Harmonics up to 15th.

4.1.3 Finding FFT of the wave without noise (AWGN and random) and with noise

FFT with no noise Code

```
% In FFT always try to take number of samples in the power of 2.
% Greater the samples greater will be the accuracy.

clc;
clear all;
close all;

fs = 10000; % Sampling Frequency
ts = 1/fs; % Sampled Time
t = 0:ts:0.1-ts; % Making intervals of the wave or sampling the wave.
L = length(t); % Number of Samples.
f = 50; % Fundamental Frequency = 50 Hz.
h1 = 1; % 1st Harmonic
h3 = 3; % 3rd Harmonic
h5 = 5; % 5th Harmonic
h7 = 7; % 7th Harmonic
h9 = 9; % 9th Harmonic
h11 = 11; % 11th Harmonic
h13 = 13; % 13th Harmonic
h15 = 15; % 15th Harmonic

v1 = 1*sin(2*pi*h1*f*t);
v3 = 1*sin(2*pi*h3*f*t);
v5 = 1*sin(2*pi*h5*f*t);
v7 = 1*sin(2*pi*h7*f*t);
v9 = 1*sin(2*pi*h9*f*t);
v11 = 1*sin(2*pi*h11*f*t);
```

```
v13 = 1*sin(2*pi*h13*f*t);
v15 = 1*sin(2*pi*h15*f*t);

v_sum = v1 + v3 + v5 + v7 + v9 + v11 + v13 + v15;

v_fft = fft(v_sum); % Taking Fast Fourier Transform of v_sum.
v_abs_with_random_amplitude = abs(v_fft); % Taking real values so that we can plot.
v_abs = 2*abs(v_fft)/L; % Correcting the Amplitude part so that graph can show correct amplitude
of the harmonics also.
x_axis = (0:L-1)*fs/L;

% Plotting the Graphs of Waves.

figure (1)
subplot (2, 2, 1)
plot (t, v1)
title('1^{st} Fundamental Harmonic')
xlabel('Time')
ylabel('v_1')

subplot (2, 2, 2)
plot (t, v3)
title('3^{rd} Fundamental Harmonic')
xlabel('Time')
ylabel('v_3')

subplot (2, 2, 3)
plot (t, v5)
title('5^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_5')

subplot (2, 2, 4)
plot (t, v7)
title('7^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_7')

suptitle('1^{st}, 3^{rd}, 5^{th} and 7^{th} Harmonics') % We can use sgtitle also but it comes
in the 2018th version of MATLAB.


figure (2)
subplot (2, 2, 1)
plot (t, v9)
title('9^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_9')

subplot (2, 2, 2)
plot (t, v11)
title('11^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_11')

subplot (2, 2, 3)
plot (t, v13)
title('13^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_13')

subplot (2, 2, 4)
plot (t, v15)
title('15^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_15')
ylabel('v_7')
```

```
suptitle('9^{th}, 11^{th}, 13^{th} and 15^{th} Harmonics')


figure (3)
plot (t, v_sum)
title('Overall sum of the wave containing 1st, 3rd, 5th, 7th, 9th, 11th, 13th and 15th
Harmonics')
xlabel('Time')
ylabel('v_{sum}')

figure (4)
plot (x_axis, v_abs)
title('Fourier Transform of v_{sum}')
ylabel('v^{fft}_{sum}')
```

Graph of FFT with no noise.

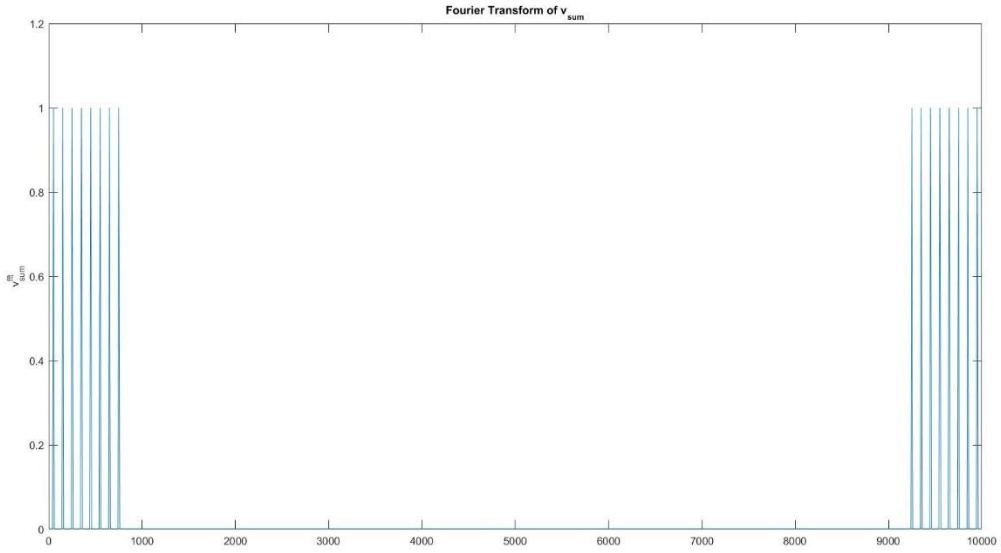

Figure 12: FFT of the wave containing 1st, 3rd, 5th, 7th, 9th, 11th, 13th and 15th.

FFT with AWGN noise Code

```
% In FFT always try to take number of samples in the power of 2.
% Greater the samples greater will be the accuracy.

clc;
clear all;
close all;

fs = 10000; % Sampling Frequency
ts = 1/fs; % Sampled Time
t = 0:ts:0.1-ts; % Making intervals of the wave or sampling the wave.
L = length(t); % Number of Samples.
f = 50; % Fundamental Frequency = 50 Hz.
```

```
h1 = 1; % 1st Harmonic
h3 = 3; % 3rd Harmonic
h5 = 5; % 5th Harmonic
h7 = 7; % 7th Harmonic
h9 = 9; % 9th Harmonic
h11 = 11; % 11th Harmonic
h13 = 13; % 13th Harmonic
h15 = 15; % 15th Harmonic

v1 = 1*sin(2*pi*h1*f*t);
v1_n1 = awgn(v1, 10); % Adding AWGN in signal v1 with SNR of 10 or of suitable choice.

v3 = 1*sin(2*pi*h3*f*t);
v3_n3 = awgn(v3, 10); % Adding AWGN in signal v3 with SNR of 10 or of suitable choice.

v5 = 1*sin(2*pi*h5*f*t);
v5_n5 = awgn(v5, 10); % Adding AWGN in signal v5 with SNR of 10 or of suitable choice.

v7 = 1*sin(2*pi*h7*f*t);
v7_n7 = awgn(v7, 10); % Adding AWGN in signal v7 with SNR of 10 or of suitable choice.

v9 = 1*sin(2*pi*h9*f*t);
v9_n9 = awgn(v9, 10); % Adding AWGN in signal v9 with SNR of 10 or of suitable choice.

v11 = 1*sin(2*pi*h11*f*t);
v11_n11 = awgn(v11, 10); % Adding AWGN in signal v11 with SNR of 10 or of suitable choice.

v13 = 1*sin(2*pi*h13*f*t);
v13_n13 = awgn(v13, 10); % Adding AWGN in signal v13 with SNR of 10 or of suitable choice.

v15 = 1*sin(2*pi*h15*f*t);
v15_n15 = awgn(v15, 10); % Adding AWGN in signal v15 with SNR of 10 or of suitable choice.

v_sum_n = v1_n1 + v3_n3 + v5_n5 + v7_n7 + v9_n9 + v11_n11 + v13_n13 + v15_n15;

v_fft = fft(v_sum_n); % Taking Fast Fourier Transform of v_sum.
v_abs_with_random_amplitude = abs(v_fft); % Taking real values so that we can plot.
v_abs = 2*abs(v_fft)/L; % Correcting the Amplitude part so that graph can show correct amplitude
of the harmonics also.
x_axis = (0:L-1)*fs/L;


% Plotting the Graphs of Waves.

figure (1)
subplot (2, 2, 1)
plot (t, v1_n1)
title('1^{st} Fundamental Harmonic')
xlabel('Time')
ylabel('v_1')

subplot (2, 2, 2)
plot (t, v3_n3)
title('3^{rd} Fundamental Harmonic')
xlabel('Time')
ylabel('v_3')

subplot (2, 2, 3)
plot (t, v5_n5)
title('5^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_5')

subplot (2, 2, 4)
plot (t, v7_n7)
title('7^{th} Fundamental Harmonic')
xlabel('Time')
```

```
ylabel('v_7')

suptitle('1^{st}, 3^{rd}, 5^{th} and 7^{th} Harmonics') % We can use sgtitle also but it comes
in the 2018th version of MATLAB.


figure (2)
subplot (2, 2, 1)
plot (t, v9_n9)
title('9^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_9')

subplot (2, 2, 2)
plot (t, v11_n11)
title('11^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_11')

subplot (2, 2, 3)
plot (t, v13_n13)
title('13^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_13')

subplot (2, 2, 4)
plot (t, v15_n15)
title('15^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_15')
ylabel('v_7')

suptitle('9^{th}, 11^{th}, 13^{th} and 15^{th} Harmonics')


figure (3)
plot (t, v_sum_n)
title('Overall sum of the wave containing 1st, 3rd, 5th, 7th, 9th, 11th, 13th and 15th
Harmonics')
xlabel('Time')
ylabel('v_{sum}')

figure (4)
plot (x_axis, v_abs)
title('Fourier Transform of v_{sum}')
ylabel('v^{fft}_{sum}')
```

Graph of FFT with AWGN noise

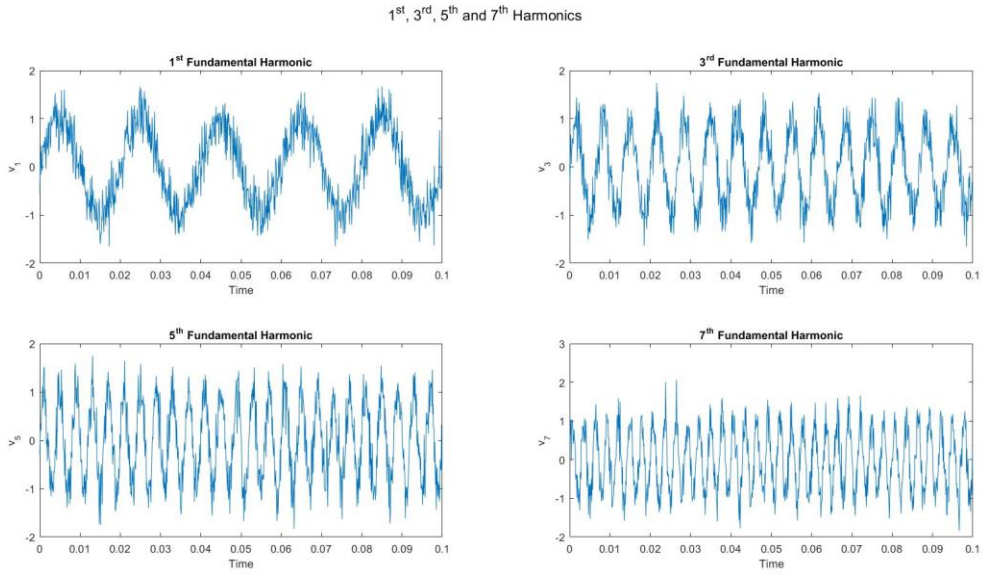

Figure 13: 1st, 3rd, 5th and 7th Harmonics with AWGN noise.

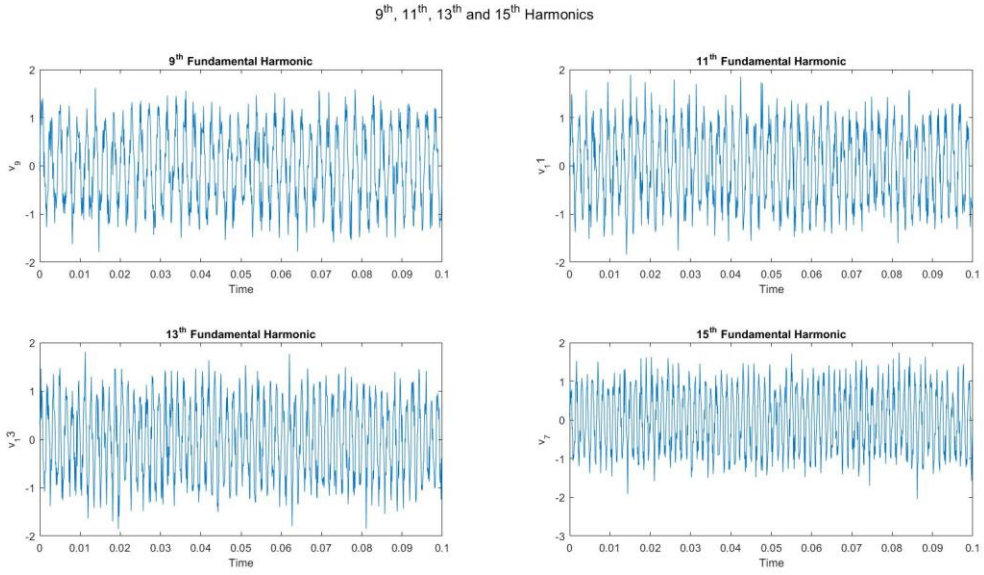

Figure 14: 9th, 11th, 13th and 15th Harmonics with AWGN noise.

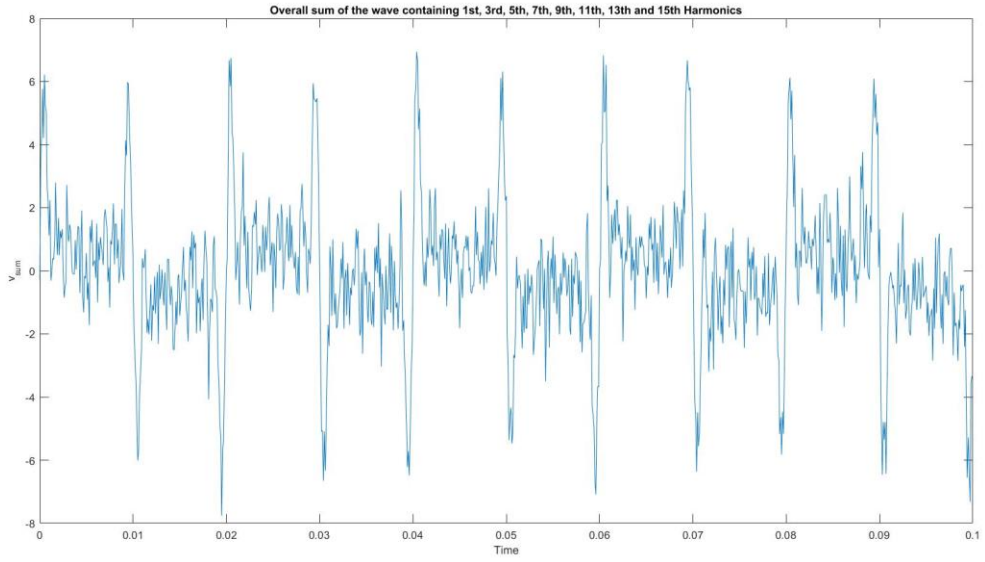

Figure 15: Overall final wave combining all the odd Harmonics up to 15th with AWGN noise.

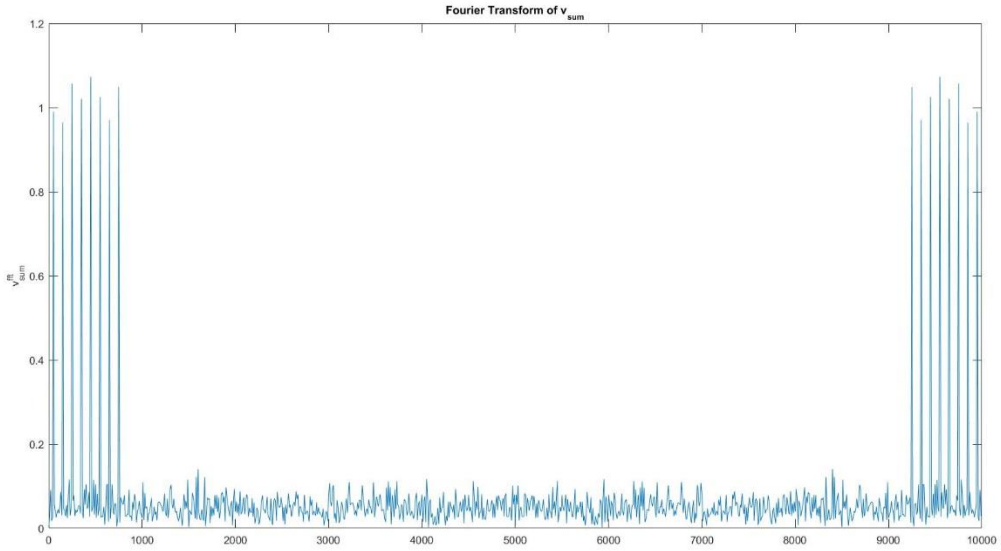

Figure 16: FFT of the wave containing 1st, 3rd, 5th, 7th, 9th, 11th, 13th and 15th with AWGN noise.

FFT with random noise Code

```
% In FFT always try to take number of samples in the power of 2.
% Greater the samples greater will be the accuracy.

clc;
clear all;
close all;

fs = 10000; % Sampling Frequency
ts = 1/fs; % Sampled Time
```

```
t = 0:ts:0.1-ts; % Making intervals of the wave or sampling the wave.
L = length(t); % Number of Samples.
f = 50; % Fundamental Frequency = 50 Hz.
h1 = 1; % 1st Harmonic
h3 = 3; % 3rd Harmonic
h5 = 5; % 5th Harmonic
h7 = 7; % 7th Harmonic
h9 = 9; % 9th Harmonic
h11 = 11; % 11th Harmonic
h13 = 13; % 13th Harmonic
h15 = 15; % 15th Harmonic

v1 = 1*sin(2*pi*h1*f*t);
n1 = rand(1, length(v1)); % Generating a random number of sequence equal to the v1 signal's
length.
v1_n1 = v1 + n1; % Adding random noise to the signal v1.

v3 = 1*sin(2*pi*h3*f*t);
n3 = rand(1, length(v3)); % Generating a random number of sequence equal to the v3 signal's
length.
v3_n3 = v3 + n3; % Adding random noise to the signal v3.

v5 = 1*sin(2*pi*h5*f*t);
n5 = rand(1, length(v5)); % Generating a random number of sequence equal to the v5 signal's
length.
v5_n5 = v5 + n5; % Adding random noise to the signal v5.

v7 = 1*sin(2*pi*h7*f*t);
n7 = rand(1, length(v7)); % Generating a random number of sequence equal to the v7 signal's
length.
v7_n7 = v7 + n7; % Adding random noise to the signal v7.

v9 = 1*sin(2*pi*h9*f*t);
n9 = rand(1, length(v9)); % Generating a random number of sequence equal to the v9 signal's
length.
v9_n9 = v9 + n9; % Adding random noise to the signal v9.

v11 = 1*sin(2*pi*h11*f*t);
n11 = rand(1, length(v11)); % Generating a random number of sequence equal to the v11 signal's
length.
v11_n11 = v11 + n11; % Adding random noise to the signal v11.

v13 = 1*sin(2*pi*h13*f*t);
n13 = rand(1, length(v13)); % Generating a random number of sequence equal to the v13 signal's
length.
v13_n13 = v13 + n13; % Adding random noise to the signal v13.

v15 = 1*sin(2*pi*h15*f*t);
n15 = rand(1, length(v15)); % Generating a random number of sequence equal to the v15 signal's
length.
v15_n15 = v15 + n15; % Adding random noise to the signal v15.

v_sum_n = v1_n1 + v3_n3 + v5_n5 + v7_n7 + v9_n9 + v11_n11 + v13_n13 + v15_n15;

v_fft = fft(v_sum_n); % Taking Fast Fourier Transform of v_sum.
v_abs_with_random_amplitude = abs(v_fft); % Taking real values so that we can plot.
v_abs = 2*abs(v_fft)/L; % Correcting the Amplitude part so that graph can show correct amplitude
of the harmonics also.
x_axis = (0:L-1)*fs/L;


% Plotting the Graphs of Waves.

figure (1)
subplot (2, 2, 1)
plot (t, v1_n1)
title('1^{st} Fundamental Harmonic')
```

```
xlabel('Time')
ylabel('v_1')

subplot (2, 2, 2)
plot (t, v3_n3)
title('3^{rd} Fundamental Harmonic')
xlabel('Time')
ylabel('v_3')

subplot (2, 2, 3)
plot (t, v5_n5)
title('5^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_5')

subplot (2, 2, 4)
plot (t, v7_n7)
title('7^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_7')

suptitle('1^{st}, 3^{rd}, 5^{th} and 7^{th} Harmonics') % We can use sgtitle also but it comes
in the 2018th version of MATLAB.


figure (2)
subplot (2, 2, 1)
plot (t, v9_n9)
title('9^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_9')

subplot (2, 2, 2)
plot (t, v11_n11)
title('11^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_11')

subplot (2, 2, 3)
plot (t, v13_n13)
title('13^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_13')

subplot (2, 2, 4)
plot (t, v15_n15)
title('15^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_15')
ylabel('v_7')

suptitle('9^{th}, 11^{th}, 13^{th} and 15^{th} Harmonics')


figure (3)
plot (t, v_sum_n)
title('Overall sum of the wave containing 1st, 3rd, 5th, 7th, 9th, 11th, 13th and 15th
Harmonics')
xlabel('Time')
ylabel('v_{sum}')

figure (4)
plot (x_axis, v_abs)
title('Fourier Transform of v_{sum}')
ylabel('v^{fft}_{sum}')
```

Graph of FFT with random noise

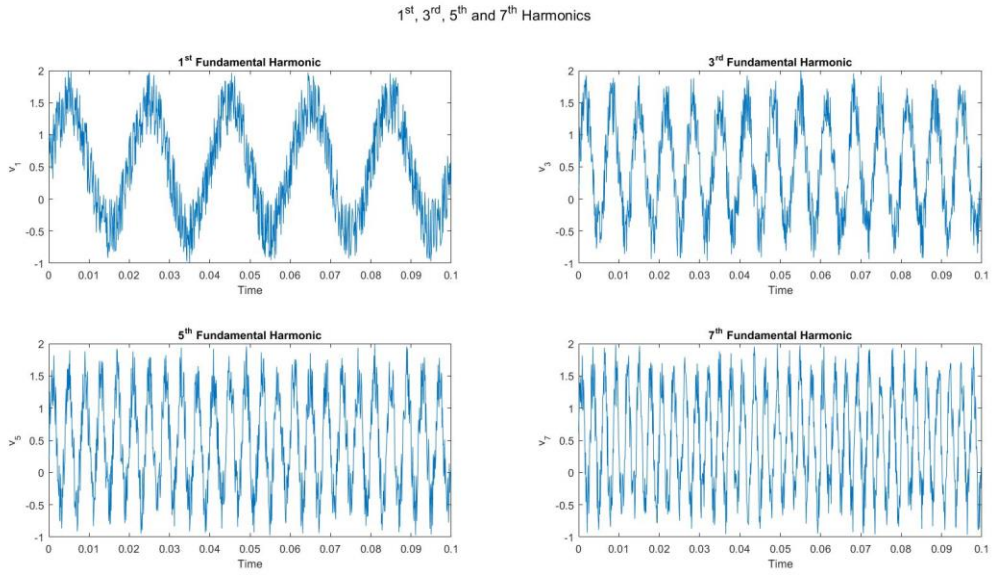

Figure 17: 1st, 3rd, 5th and 7th Harmonics with random noise.

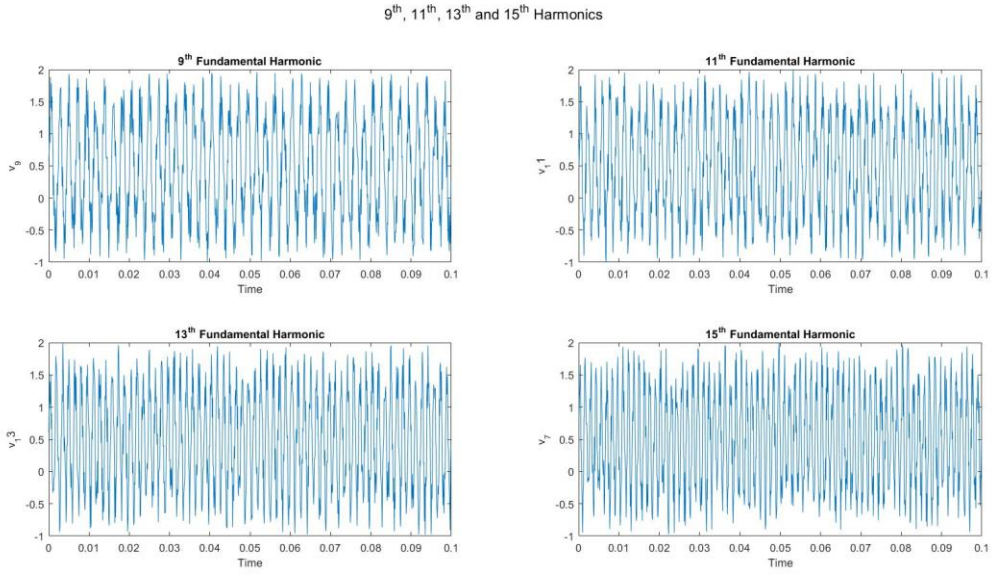

Figure 18: 9^{th}, 11^{th}, 13^{th} and 15^{th} Harmonics with random noise.

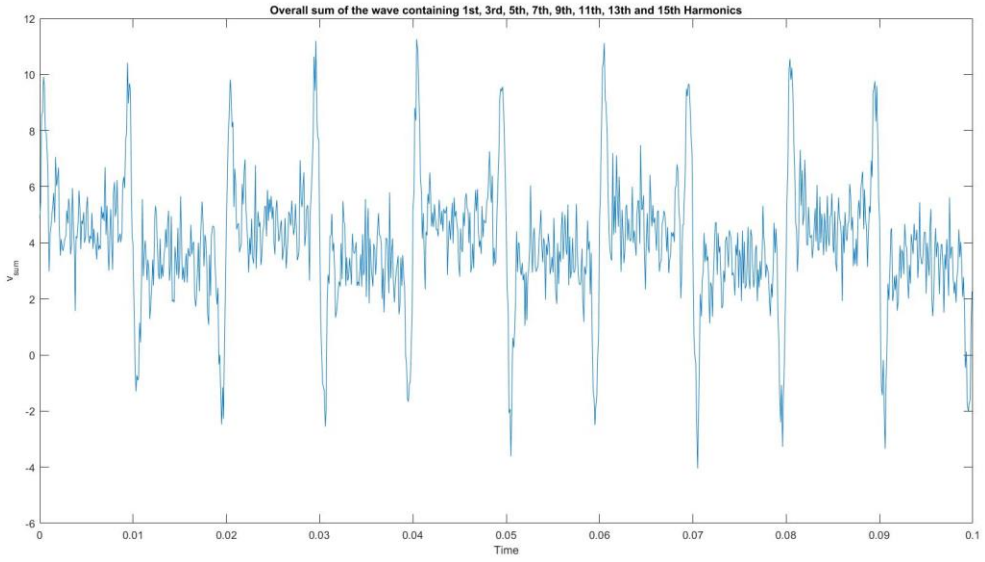

Figure 19: Overall final wave combining all the odd Harmonics up to 15th with random noise.

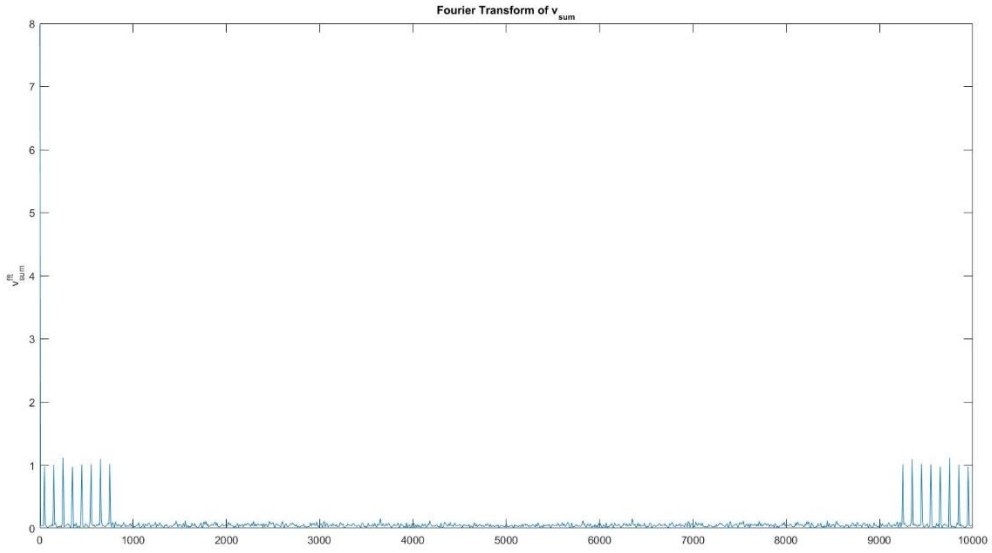

Figure 20: FFT of the wave containing 1^{st}, 3^{rd}, 5^{th}, 7^{th}, 9^{th}, 11^{th}, 13^{th} and 15^{th} with random noise.

4.1.4 Training the Neural Network with 2 ways for FFT method

Code using nntool box

```
clc;
clear all;
close all;

fs = 10000; % Sampling Frequency
ts = 1/fs; % Sampled Time
t = 0:ts:0.1-ts; % Making intervals of the wave or sampling the wave.
L = length(t); % Number of Samples.
f = 50; % Fundamental Frequency = 50 Hz.
h1 = 1; % 1st Harmonic
h3 = 3; % 3rd Harmonic
h5 = 5; % 5th Harmonic
h7 = 7; % 7th Harmonic

v1 = 1*sin(2*pi*h1*f*t);
v3 = 2*sin(2*pi*h3*f*t);
v5 = 3*sin(2*pi*h5*f*t);
v7 = 4*sin(2*pi*h7*f*t);

v_sum = v1 + v3 + v5 + v7;

v_fft = fft(v_sum); % Taking Fast Fourier Transform of v_sum.
v_abs_with_random_amplitude = abs(v_fft); % Taking real values so that we can plot.
v_abs = 2*abs(v_fft)/L; % Correcting the Amplitude part so that graph can show correct amplitude
of the harmonics also.
x_axis = (0:L-1)*fs/L;

input = v_sum; % Defining input neuron to the Neural Network
output = v_abs; % Defining output neuron to the Neural Network
```

Code manually to train Neural Network

```
clc;
clear all;
close all;

fs = 10000; % Sampling Frequency
ts = 1/fs; % Sampled Time
t = 0:ts:0.1-ts; % Making intervals of the wave or sampling the wave.
L = length(t); % Number of Samples.
f = 50; % Fundamental Frequency = 50 Hz.
h1 = 1; % 1st Harmonic
h3 = 3; % 3rd Harmonic
h5 = 5; % 5th Harmonic
h7 = 7; % 7th Harmonic

v1 = 1*sin(2*pi*h1*f*t);
v3 = 2*sin(2*pi*h3*f*t);
v5 = 3*sin(2*pi*h5*f*t);
v7 = 4*sin(2*pi*h7*f*t);

v_sum = v1 + v3 + v5 + v7;

v_fft = fft(v_sum); % Taking Fast Fourier Transform of v_sum.
v_abs_with_random_amplitude = abs(v_fft); % Taking real values so that we can plot.
v_abs = 2*abs(v_fft)/L; % Correcting the Amplitude part so that graph can show correct amplitude
of the harmonics also.
x_axis = (0:L-1)*fs/L;

input = v_sum; % Defining input neuron to the Neural Network
output = v_abs; % Defining output neuron to the Neural Network

net = newff(minmax(input), [20, 1], {'logsig', 'purelin', 'trainlm'});
```

```
%{
Create a feed-forward backpropagation network

Syntax
net = newff

net = newff(PR,[S1 S2...SNl],{TF1 TF2...TFNl},BTF,BLF,PF)

Description
net = newff creates a new network with a dialog box.

newff(PR,[S1 S2...SNl],{TF1 TF2...TFNl},BTF,BLF,PF) takes,

PR -- R x 2 matrix of min and max values for R input elements

Si -- Size of ith layer, for Nl layers

TFi -- Transfer function of ith layer, default = 'tansig'

BTF -- Backpropagation network training function, default = 'traingdx'

BLF -- Backpropagation weight/bias learning function, default = 'learngdm'

PF -- Performance function, default = 'mse'

and returns an N layer feed-forward backprop network.

The transfer functions TFi can be any differentiable transfer function such as tansig, logsig,
or purelin.

The training function BTF can be any of the backprop training functions such as trainlm,
trainbfg, trainrp, traingd, etc.

Caution: trainlm is the default training function because it is very fast, but it requires a
lot of memory to run. If you get an "out-of-memory" error when training try doing one of these:
Slow trainlm training, but reduce memory requirements by setting net.trainParam.mem_reduc to 2
or more. (See help trainlm.)
Use trainbfg, which is slower but more memory-efficient than trainlm.
Use trainrp, which is slower but more memory-efficient than trainbfg.
The learning function BLF can be either of the backpropagation learning functions such as
learngd or learngdm.

The performance function can be any of the differentiable performance functions such as mse or
msereg.

Examples
Here is a problem consisting of inputs P and targets T that we would like to solve with a
network.

P = [0 1 2 3 4 5 6 7 8 9 10];
T = [0 1 2 3 4 3 2 1 2 3 4];
Here a two-layer feed-forward network is created. The network's input ranges from [0 to 10].
The first layer has five tansig neurons, the second layer has one purelin neuron. The trainlm
network training function is to be used.

net = newff([0 10],[5 1],{'tansig' 'purelin'});
Here the network is simulated and its output plotted against the targets.

Y = sim(net,P);
plot(P,T,P,Y,'o')
Here the network is trained for 50 epochs. Again the network's output is plotted.

net.trainParam.epochs = 50;
net = train(net,P,T);
Y = sim(net,P);
plot(P,T,P,Y,'o')
```

```
%}

% In FFT always try to take number of samples in the power of 2.
% Greater the samples greater will be the accuracy.

net.trainparam.epochs = 1000; % Setting number of epochs.
net.trainparam.goal = 1e-30; % Setting our neural network performance that is the loss ratio
minimisation, how much loss ratio can be reduced.
net.trainparam.lr = 0.01; % Setting learning rate.

net = train(net, input, output); % Training neural network by train(1, 2, 3), where 1 is Neural
Network -> net, 2 is input parameter, 3 is output parameter or target parameter.

figure (1)
hold on % as long as hold on all plots will be plotted in the same graph until you type hold
off.
plot(output)
plot(net(input))

output_neural = net(input);

error = (output_neural - output).^2; % calculating the loss ratio and it's graph.

figure (2)
plot(error)
```

Graphs

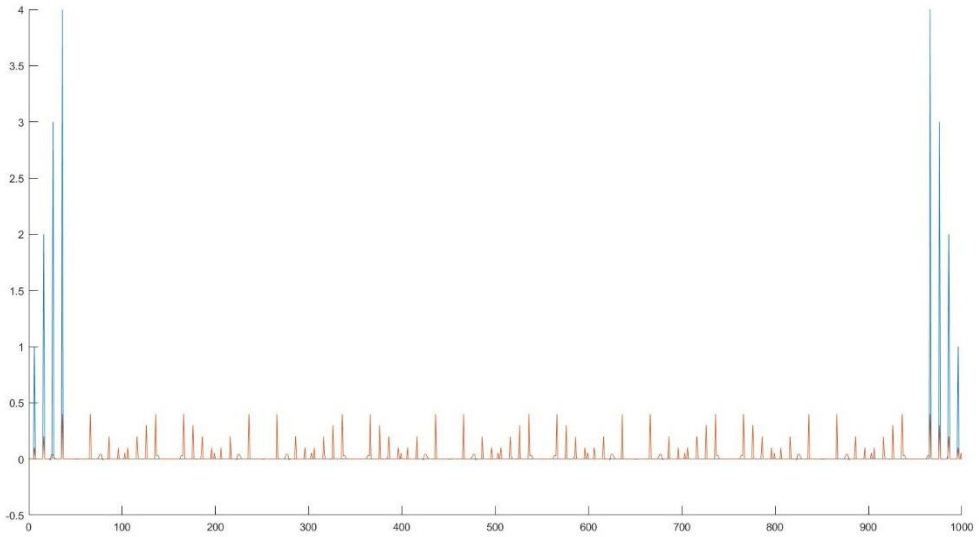

Figure 21: FFT vs Output of Neural Network (having very less accuracy)

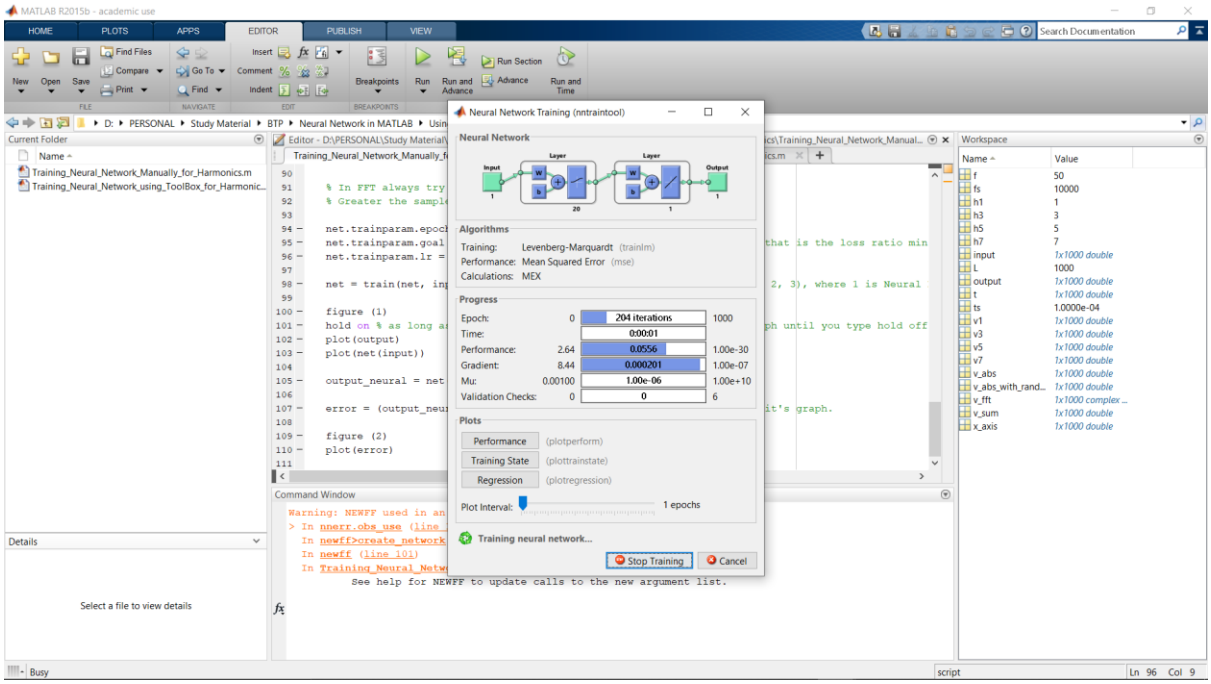

Figure 22: Training of the Neural Network.

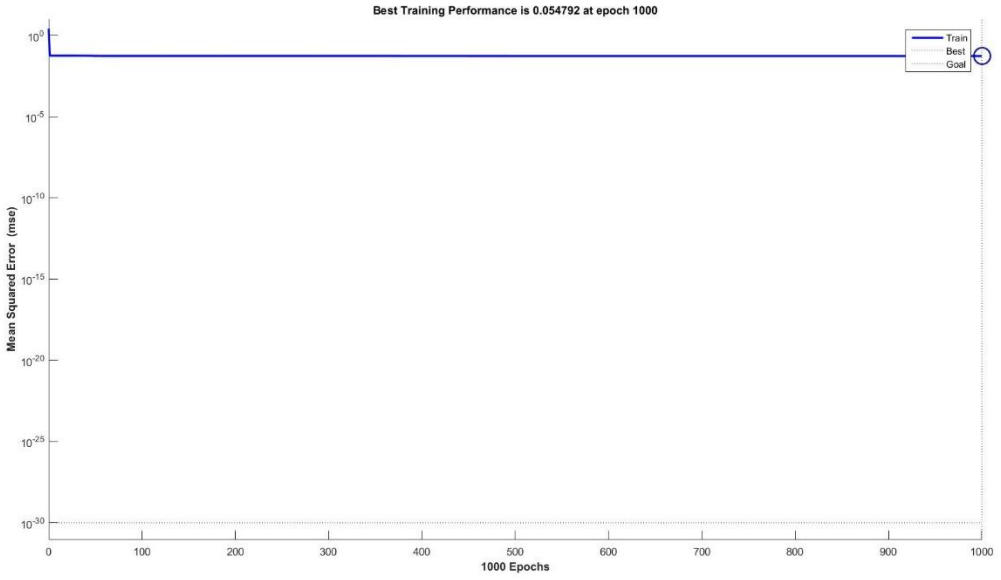

Figure 23: Performance of the Neural Network.

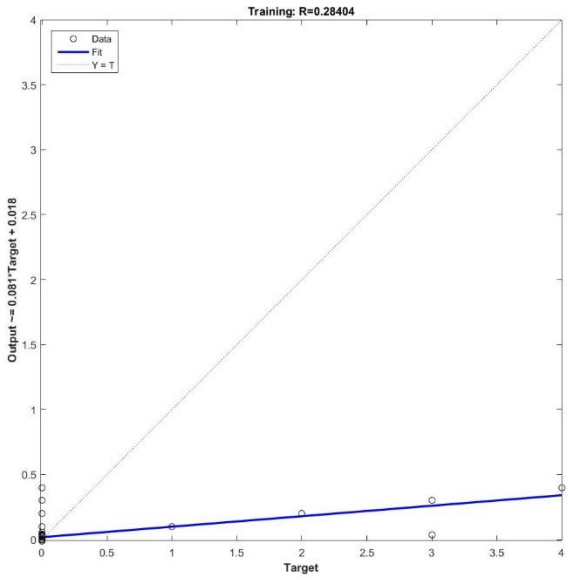

Figure 24: Regression of the Neural Network.

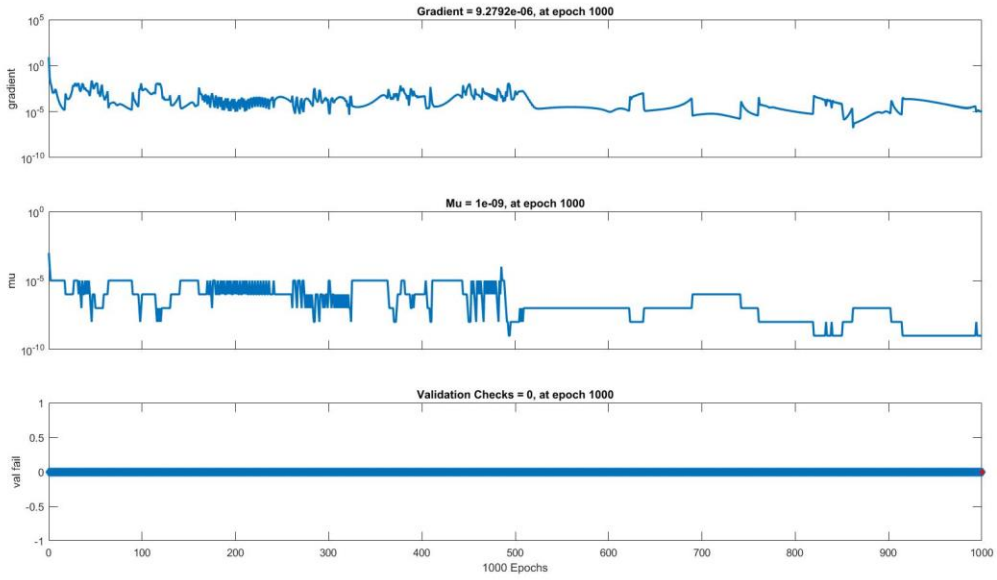

Figure 25: Training State of the Neural Network.

4.1.5 Simulink diagrams of the Neural Network design from scratch in System Generator

Main Structure

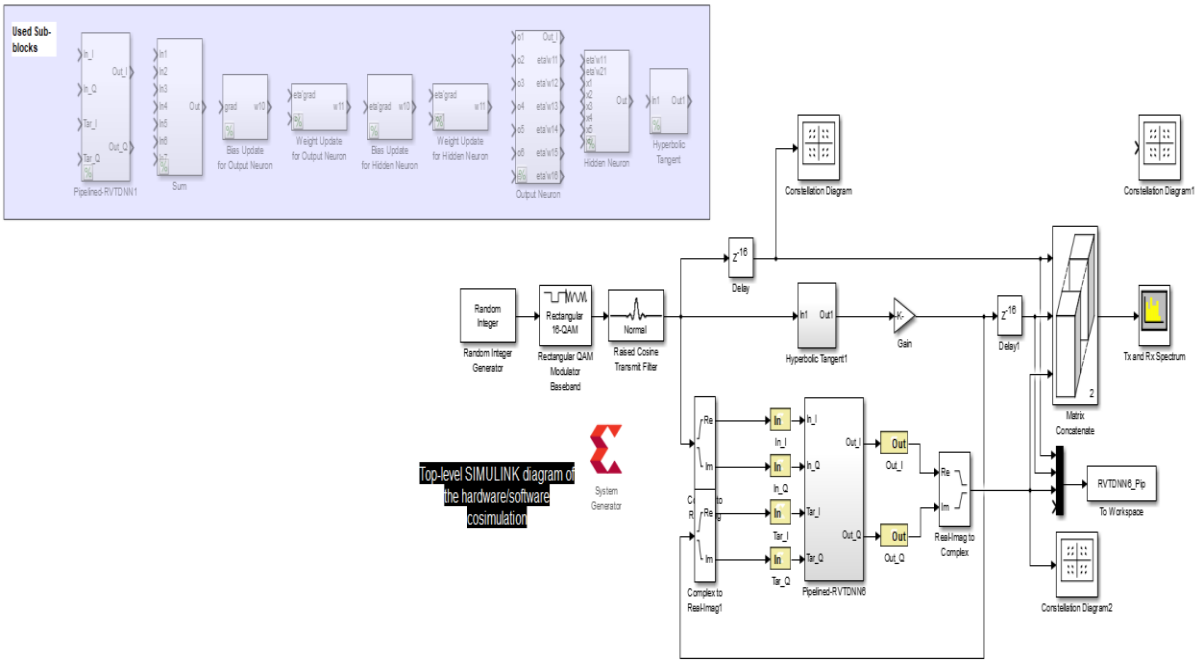

Figure 26: Neural Network model design in System Generator.

Sub-Structures

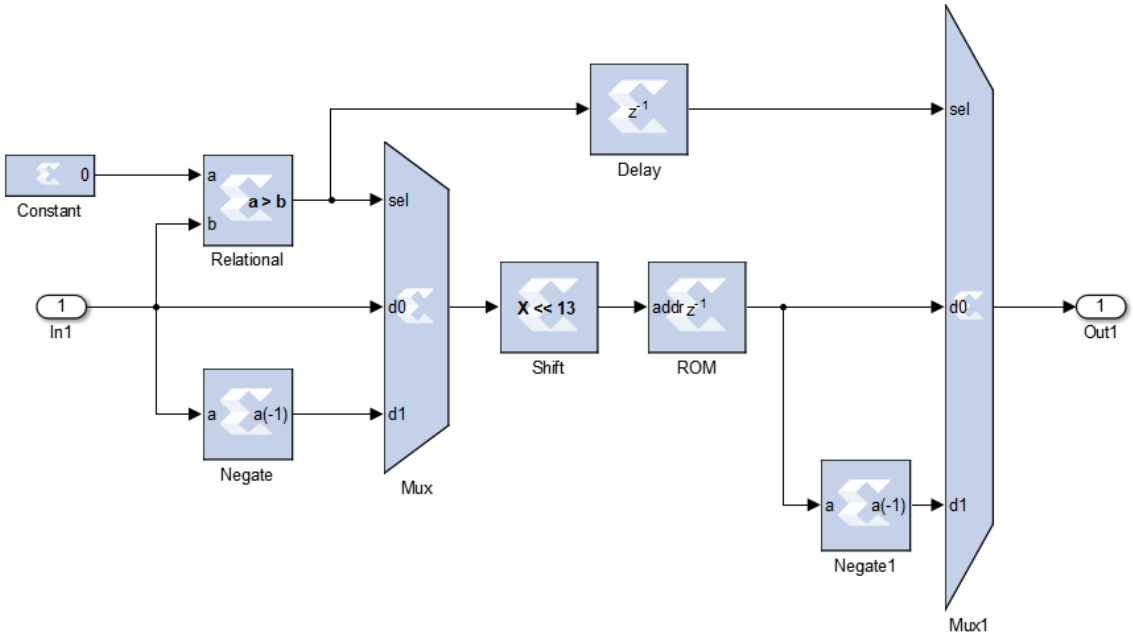

Figure 27: Sub-Structure of Hyperbolic Tangent block.

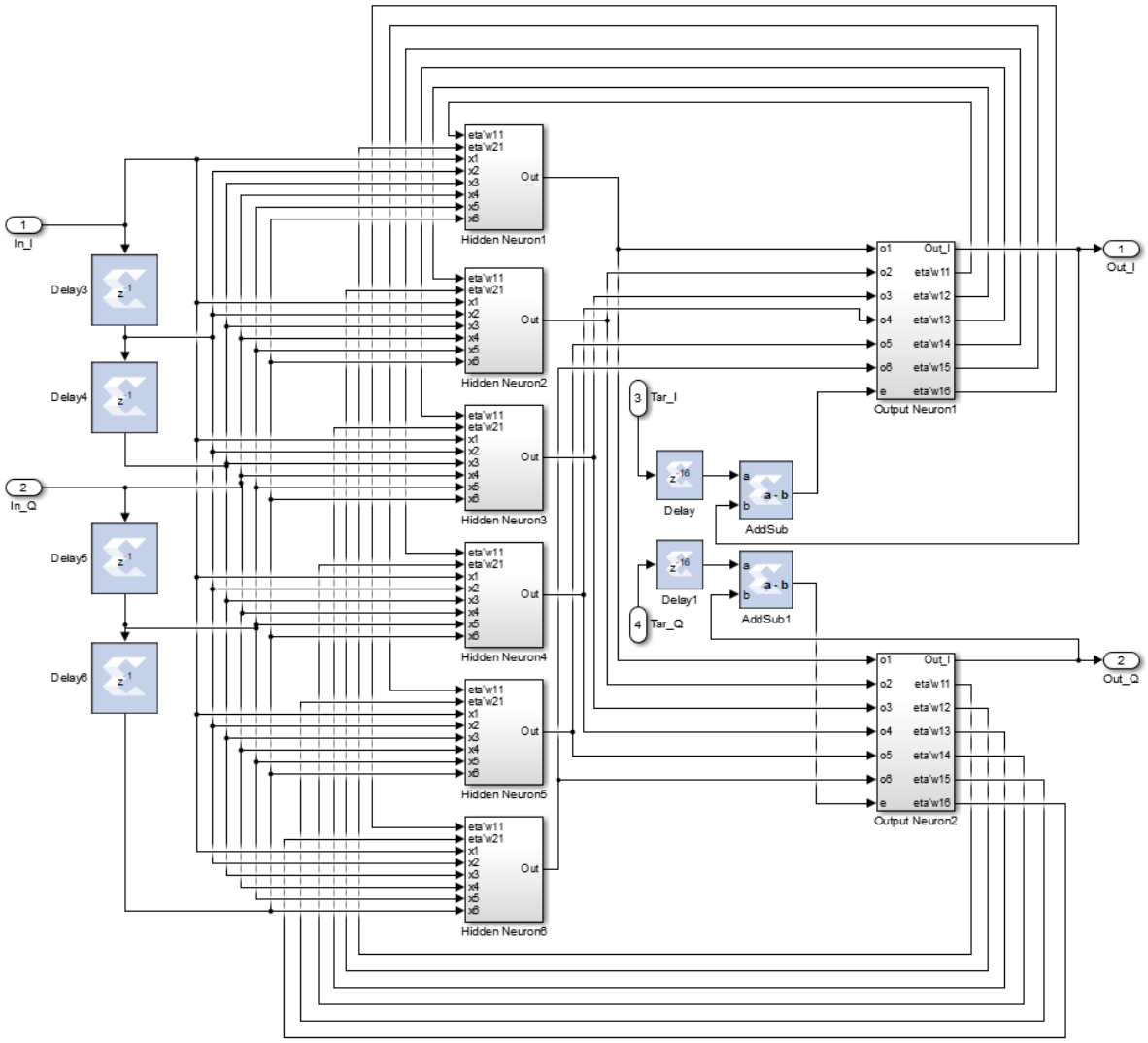

Figure 28: Sub-Structure of Pipelined-RVTDNN6 block.

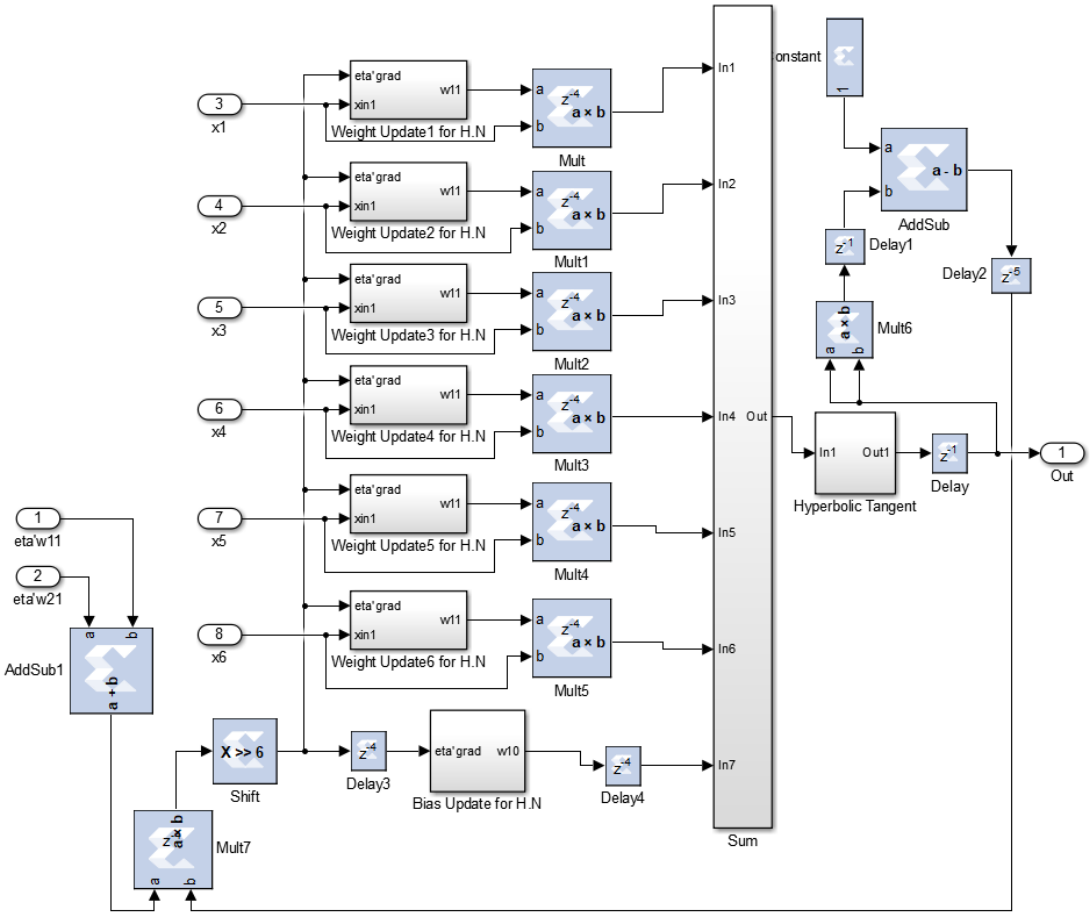

Figure 29: Sub-Structure of Hidden Neuron block.

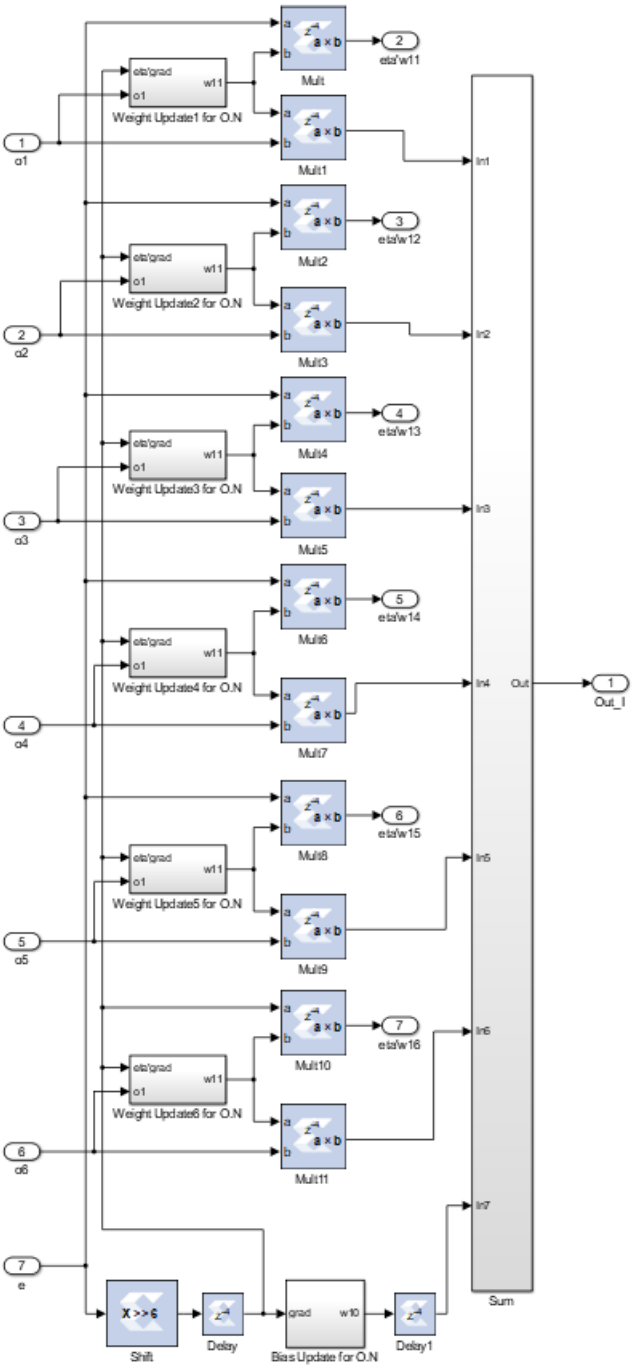

Figure 30: Sub-Structure of Output Neuron block.

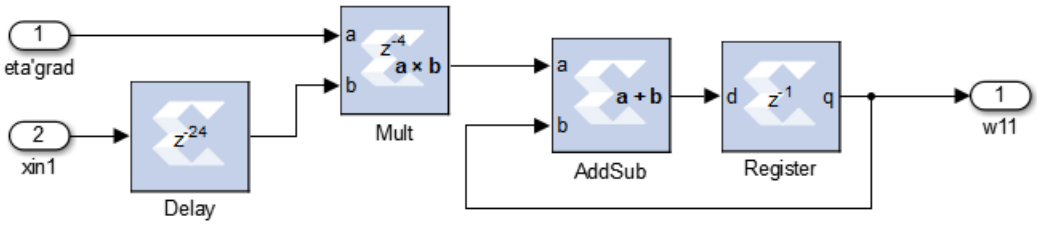

Figure 31: Sub-Structure of Weight Update block.

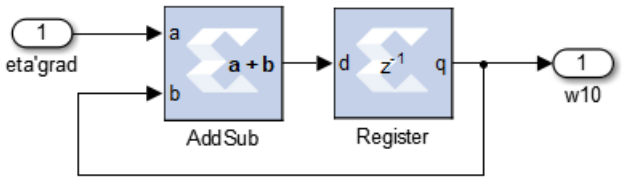

Figure 32: Sub-Structure of Bias Update block.

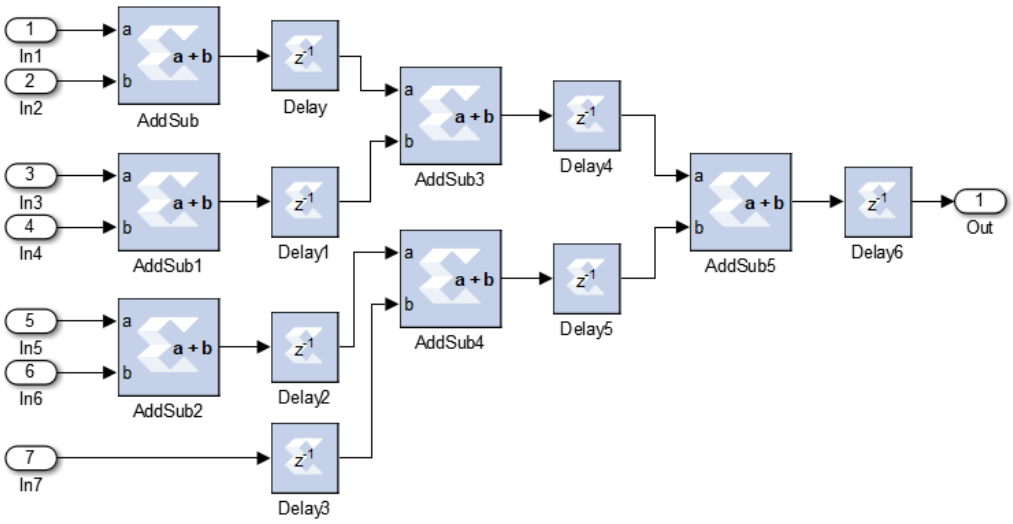

Figure 33: Sub-Structure of Sum block.

4.1.6 Training of Neural Network by Hit and Trial method to detect Harmonics in wave in MATLAB

Code

```
clc;
clear all;
close all;

fs = 10000; % Sampling Frequency
ts = 1/fs; % Sampled Time
t = 0:ts:200-ts; % Making intervals of the wave or sampling the wave.
f = 50; % Fundamental Frequency = 50 Hz.
h1 = 1; % 1st Harmonic
h3 = 3; % 3rd Harmonic
h5 = 5; % 5th Harmonic
h7 = 7; % 7th Harmonic
h9 = 9; % 9th Harmonic
h11 = 11; % 11th Harmonic
h13 = 13; % 13th Harmonic
h15 = 15; % 15th Harmonic

A = randi(5, 1, 10000);
partitions = 10000;

v1 = 1*sin(2*pi*h1*f*t);
```

```
for i = 1:partitions;

    temp = ((i-1)*200) + 1;

    for j = temp:temp + 199

        v1_new(j) = A(i)*v1(j);
        end;

end;

v3 = 1*sin(2*pi*h3*f*t);
for i = 1:partitions;

    temp = ((i-1)*200) + 1;

    for j = temp:temp + 199

        v3_new(j) = A(i)*v3(j);
        end;

end;

v5 = 1*sin(2*pi*h5*f*t);
for i = 1:partitions;

    temp = ((i-1)*200) + 1;

    for j = temp:temp + 199

        v5_new(j) = A(i)*v5(j);
        end;

end;

v7 = 1*sin(2*pi*h7*f*t);
for i = 1:partitions;

    temp = ((i-1)*200) + 1;

    for j = temp:temp + 199

        v7_new(j) = A(i)*v7(j);
        end;

end;

v9 = 1*sin(2*pi*h9*f*t);
for i = 1:partitions;

    temp = ((i-1)*200) + 1;

    for j = temp:temp + 199

        v9_new(j) = A(i)*v9(j);
        end;

end;

v11 = 1*sin(2*pi*h11*f*t);
for i = 1:partitions;

    temp = ((i-1)*200) + 1;

    for j = temp:temp + 199

        v11_new(j) = A(i)*v11(j);
```

```
        end;

end;

v13 = 1*sin(2*pi*h13*f*t);
for i = 1:partitions;

    temp = ((i-1)*200) + 1;

    for j = temp:temp + 199

        v13_new(j) = A(i)*v13(j);
        end;

end;

v15 = 1*sin(2*pi*h15*f*t);
for i = 1:partitions;

    temp = ((i-1)*200) + 1;

    for j = temp:temp + 199

        v15_new(j) = A(i)*v15(j);
        end;

end;

v_sum = v1 + v3 + v5 + v7 + v9 + v11 + v13 + v15;
v_sum_new = v1_new + v3_new + v5_new + v7_new + v9_new + v11_new + v13_new + v15_new;

for l = 1:200000
    v_sum_new_sampled(l) = v_sum_new(10*l);
end;

for m = 1:20

    for n = 1:10000
        Final_input(m, n) = v_sum_new_sampled(20*(n-1) + m);
    end;
end;

for b = 1:8
    for v = 1:10000
        Final_output(b, v) = A(v);
    end;
end;

% Plotting the Graphs of Waves.

figure (1)
subplot (2, 2, 1)
plot (t, v1)
hold on
plot (t, v1_new)
title('1^{st} Fundamental Harmonic')
xlabel('Time')
ylabel('v_1')

subplot (2, 2, 2)
plot (t, v3)
hold on
plot (t, v3_new)
title('3^{rd} Fundamental Harmonic')
xlabel('Time')
ylabel('v_3')
```

```
subplot (2, 2, 3)
plot (t, v5)
hold on
plot (t, v5_new)
title('5^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_5')

subplot (2, 2, 4)
plot (t, v7)
hold on
plot (t, v7_new)
title('7^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_7')

suptitle('1^{st}, 3^{rd}, 5^{th} and 7^{th} Harmonics') % We can use sgtitle also but it comes
in the 2018th version of MATLAB.


figure (2)
subplot (2, 2, 1)
plot (t, v9)
hold on
plot (t, v9_new)
title('9^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_9')

subplot (2, 2, 2)
plot (t, v11)
hold on
plot (t, v11_new)
title('11^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_11')

subplot (2, 2, 3)
plot (t, v13)
hold on
plot (t, v13_new)
title('13^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_13')

subplot (2, 2, 4)
plot (t, v15)
hold on
plot (t, v15_new)
title('15^{th} Fundamental Harmonic')
xlabel('Time')
ylabel('v_15')
ylabel('v_7')

suptitle('9^{th}, 11^{th}, 13^{th} and 15^{th} Harmonics')

figure (4)
plot (t, v_sum)
hold on
plot (t, v_sum_new)
title('Overall sum of the wave containing 1st, 3rd, 5th, 7th, 9th, 11th, 13th and 15th
Harmonics')
xlabel('Time')
ylabel('v_{sum}')
```

Graphs obtained from Code and Neural Network results

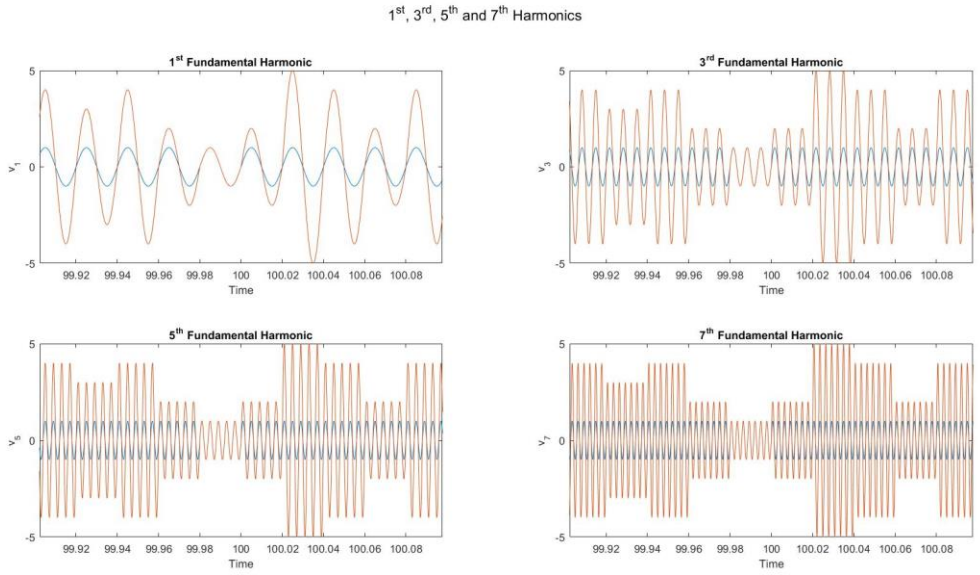

Figure 34: Amplitude variation of 1st, 3rd, 5th and 7th.

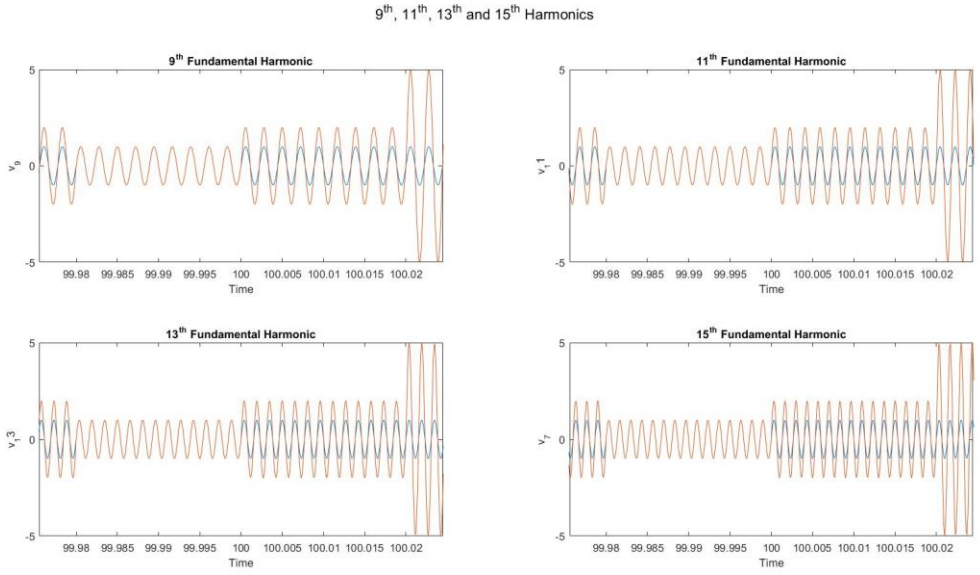

Figure 35: Amplitude variation of 9th, 11th, 13th and 15th.

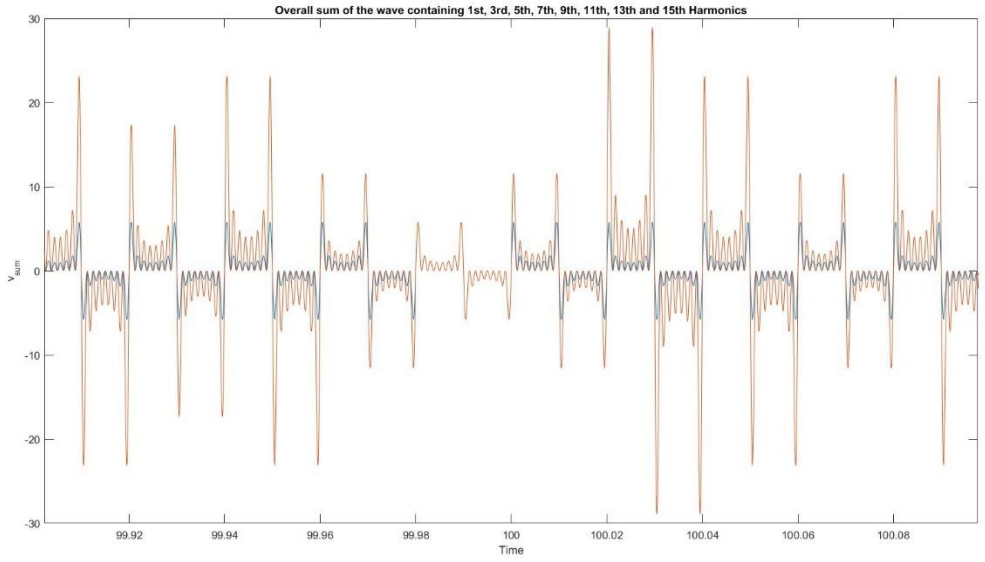

Figure 36: Total summation of all the Harmonics having Varying Amplitude vs Constant Amplitude.

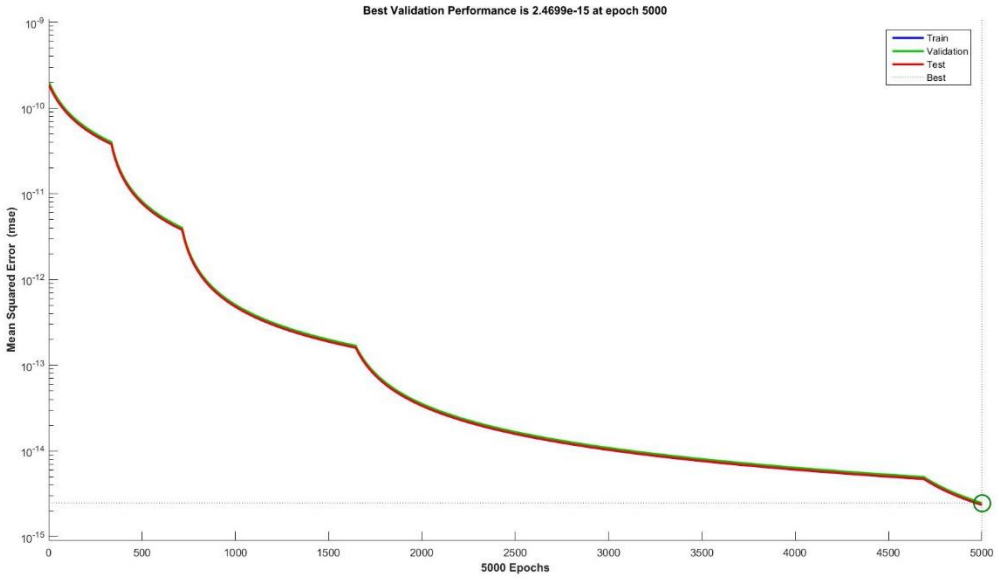

Figure 37: Performance of the Neural Network from Hit and Trial Method.

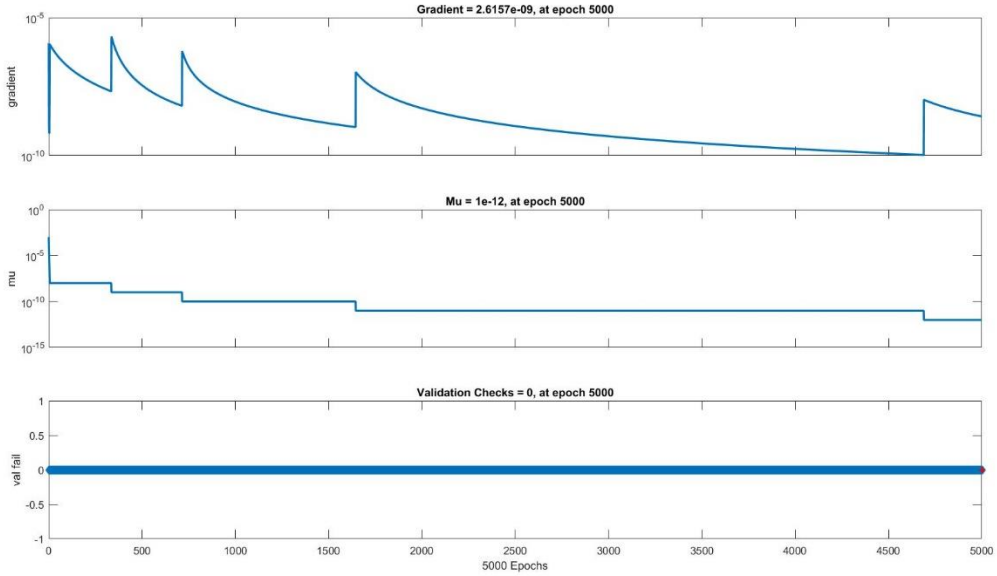

Figure 38: Training State of Neural Network from Hit and Trial method.

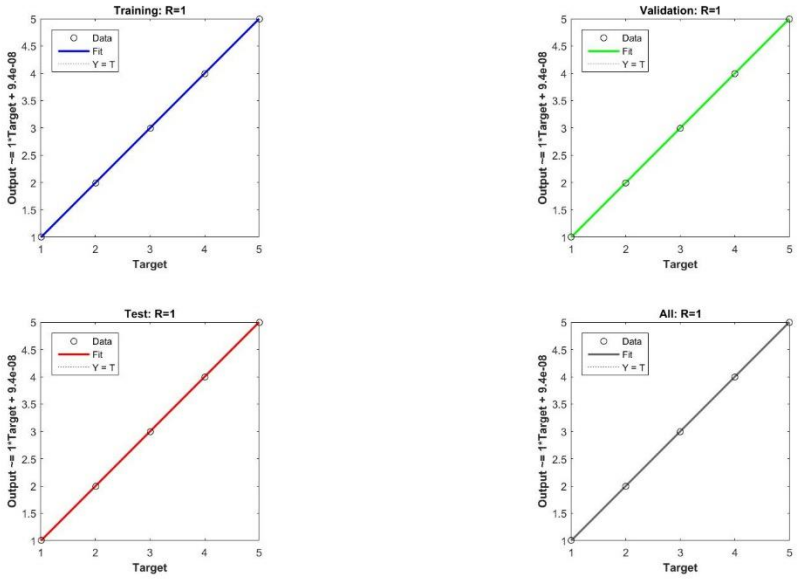

Figure 39: Regression of Neural Network from Hit and Trial method.

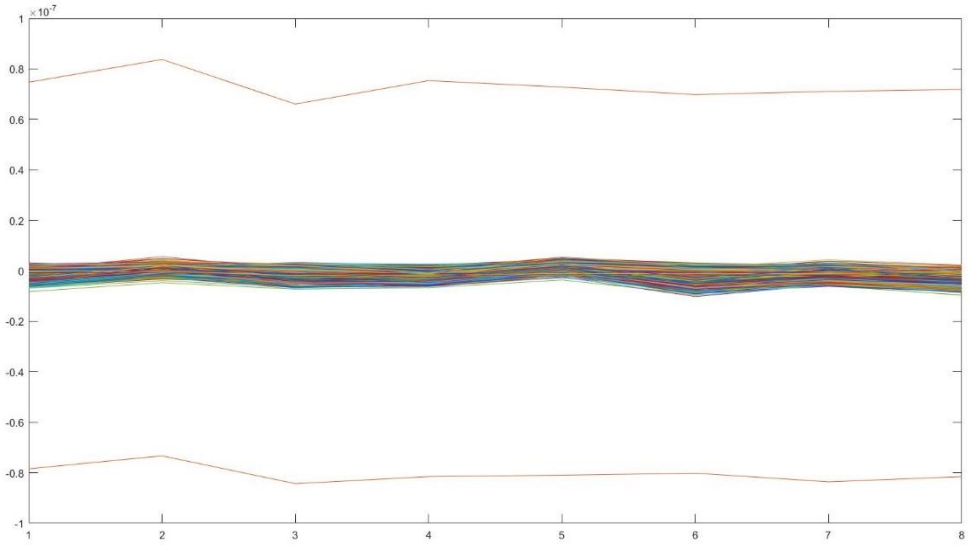

Figure 40: Amount of Error in the Neural Network.

Basic structure of our Neural Network

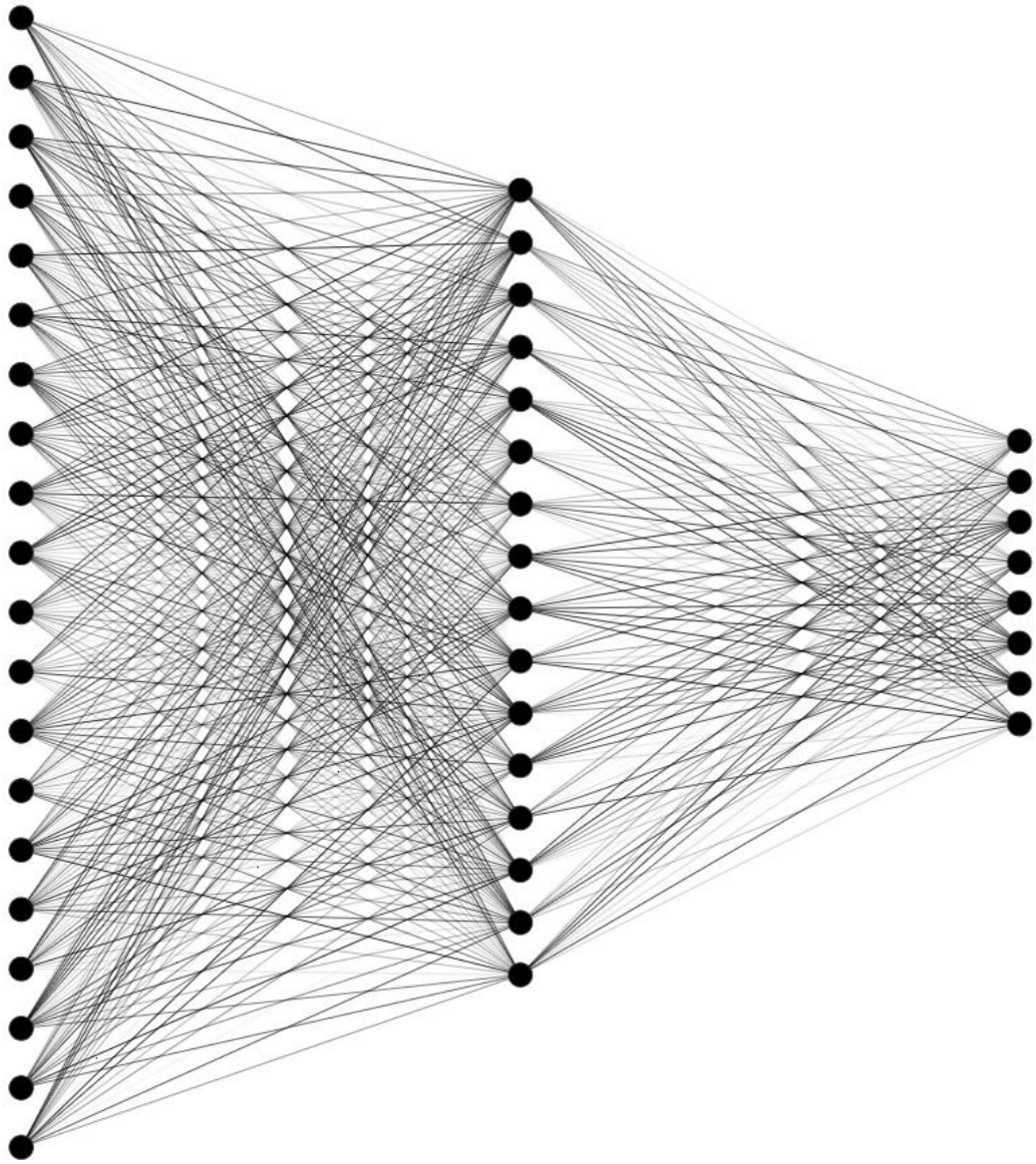

Figure 41: Neural Network pseudo structure from Hit and Trial Method.

5 Chapter 5:

5.1 Conclusions and Future Work

At last, we have tried to create a neural network model on the software which can estimate the harmonics in the power supply with better accuracy as compared to the initially used method of FFT, then we use Hit and Trial method which is better than the former. This project mainly focuses the software aspect till now, since the work on the Hardware prototype is not completed yet. The upcoming task is that we will train the Neural Network manually by giving weights to it.

Since the hardware prototype has a problem with it that it cannot support the non-linear functions to train, which limits our accuracy of the Neural Network, that's why we have tried different methods for detecting the Harmonics components of the simple wave structure by its Amplitude, Frequency or Phase etc.

We are going to implement this neural network on hardware on VIRTEX ML 605 next. We can also modify our neural network by changing:

1. The algorithm of the neural network like CNN, GAN etc.
2. We can try different functions instead of the hyperbolic function.
3. We can also modify the number of layers in the neural network or the number of nodes in each layer.
4. We can also train the neural network with bigger data sets.
5. We can also compare and study other methods harmonic estimation and try to improve our model based on that.

If this project creates the Hardware prototype in future as we have already discussed then with the help of Harmonics, we can control the parameters of the signal in the form of current, voltage etc.

By implementing all feasible steps and using a Harmonic Analyser, industries can greatly increase their energy efficiency and can replace their outdated equipment with newer models, with the help of early identification of the Harmonic voltage, they can minimise the possibility of further damage or equipment shutdown.

The real-world application of harmonics is in simplifying a complex process. For instance, 1^{st} Harmonic response gives you indication of the linear approximation. In power systems 3^{rd} Harmonics also play important role due to their sequence nature in three phase systems. In control systems non-linear models are simplified by linear approximation in which non linearity is approximated by 1^{st} Harmonic over a band of frequencies of harmonic excitation.

But the nature and scope of the ideas associated with harmonics is so vast that a comprehensive idea of application can come only after studying some applied Mathematics such as differential equations, pure Mathematics such as Fourier Analysis as well as Physics. Many of our

instruments and medical imaging actually is an application of Harmonics. Alternating current electricity is also one of the application of Harmonics. A lot of real-world application comes from deep results of pure mathematics which developed without any idea of the application. Much of this Mathematics developed while extending the ideas associated with Fourier series and hence ultimately harmonics. While harmonic analysis is still used for navigation, it also has a wide range of other, more surprising uses in signal processing, quantum physics, neuroscience, tomography, and other fields.

We hope that this report helps you to understand our work and progress till now.

Bibliography

[1] https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

[2] What is a Neural Network? AI and ML Guide - AWS (amazon.com)

[3] Deep Learning playlist overview & Machine Learning intro - YouTube

[4] What is Gradient Descent? | IBM

[5] Vanishing Gradient Problem, Explained - KDnuggets

[6] Exploding Gradient Problem Definition | DeepAI

[7] Standing wave - Wikipedia

[8] Difference Between Stationary and Non-Stationary Signals (askanydifference.com)

[9] Causes and Effects of Harmonics in Electrical Power Systems | Fluke

[10] What Is Harmonic? Definition From WhatIs (techtarget.com)

[11] https://docs.xilinx.com/v/u/2016.3-English/ug888-vivado-design-flows-overview-tutorial

[12] https://www.researchgate.net/publication/299570650_Hardware_Implementation_of_Polyphase-Decomposition-Based_Wavelet_Filters_for_Power_System_Harmonics_Estimation

[13] https://www.researchgate.net/publication/236264061_FPGA-implementation_of_dynamic_time_delay_neural_network_for_power_amplifier_behavioral_modeling

[14] https://www.researchgate.net/publication/225683982_FPGA-Implementation_of_Parallel_and_Sequential_Architectures_for_Adaptive_Noise_Cancelation

[15] https://link.springer.com/article/10.1007/s10470-014-0263-7

[16] https://www.researchgate.net/publication/3218798_Intelligent_Neural_Network-Based_Fast_Power_System_Harmonic_Detection