# ARTIFICIAL INTELLIGENCE FOR MICROWAVE CIRCUIT SIMULATIONS

**SAM KAH SENG**

**Universiti Sains Malaysia**

**2017**

# ARTIFICIAL INTELLIGENCE FOR MICROWAVE CIRCUIT SIMULATIONS

**by**

**SAM KAH SENG**

**Thesis submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Electronic Engineering)**

**JUNE 2017**

# ACKNOWLEDGEMENTS

This thesis becomes a reality with the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

Foremost, I would like to express my deepest gratitude to my project supervisor, Dr. Patrick Goh Kuan Lye for his support and guidance throughout the research. Once again, I would like to thank him for imparting his knowledge and expertise in the field of signal integrity, microwave and artificial neural network. His continuous support leads me to the right way.

Besides, I would like to thank my project examiner, Professor. Dr Widad bt. Ismail for her positive feedback and suggestions in providing deeper understanding and approval of my work.

My sincere appreciation is extended to all of my friends for their invaluable support, advice and patience, who were always with me during the project.

Most of all, I am fully indebted to my family for the encouragement which helped me in the completion of this work.

# TABLE OF CONTENTS

**ACKNOWLEDGEMENTS** ..... ii

**TABLE OF CONTENTS** ..... iii

**LIST OF TABLES** ..... vi

**LIST OF FIGURES** ..... vii

**LIST OF ABBREVIATIONS** ..... ix

**LIST OF SYMBOLS** ..... xi

**ABSTRAK** ..... xiii

**ABSTRACT** ..... xiv

**CHAPTER 1 – INTRODUCTION** ..... 1

1.1 Research Background ..... 1

1.2 Problem Statement ..... 2

1.3 Objectives of Research ..... 3

1.4 Scope of Research ..... 3

1.5 Thesis Outline ..... 3

**CHAPTER 2 – LITERATURE REVIEW** ..... 5

2.1 Introduction ..... 5

2.2 Structure of Neural Network ..... 5

2.3 Neural Learning ..... 7

2.4 Training and Validation of Neural Network ..... 9

2.5 Training Algorithms ..... 10

2.6 Levenberg-Marquardt backpropagation ..... 12

2.7 Finding Number of Hidden Neurons ..... 13

2.8 Meander Lines/Serpentine Lines ..... 14

2.9 Differential Signalling With Meander Line ..... 16

2.10 Summary ..... 17

**CHAPTER 3 – METHODOLOGY** ..... 19

3.1 Introduction ..... 19

3.2 Project Flow ..... 19

3.3 Data Collection through Simulations ..... 23

3.4 Separation of Data into Subsets ..... 27

3.5 Multilayer Neural Network Configuration ..... 28

3.6 Mean Squared Error Performance Function (MSE) ..... 29

3.7 Determining Number of Hidden Neuron ..... 30

3.8 Neural Network Model Performance Evaluation ..... 30

3.9 Summary ..... 32

**CHAPTER 4 – RESULTS AND DISCUSSIONS** ..... 33

4.1 Introduction ..... 33

4.2 Simulation Using ADS Momentum ..... 33

4.3 Neural Network with Different Number of Hidden Neurons ..... 37

4.3.1 Case 1: Effect of Segment Length on Propagation Delay Time ..... 37

4.3.2 Case 2: Effect of Segment Spacing on Propagation Delay Time ..... 43

4.4 Neural Network with Different Number of Input Data Points ..... 49

4.4.1 Case 1: Effect of Segment Length on Propagation Delay Time ..... 49

4.4.2 Case 2: Effect of Segment Spacing on Propagation Delay Time ..... 50

4.5 Overall Outcome ..... 51

4.6 Summary ..... 52

**CHAPTER 5 – CONCLUSION AND FUTURE WORK** ..... 53

5.1 Introduction ..... 53

5.2 Conclusion ..... 53

5.3 Future Work ..... 54

**REFERENCES** ..... 56

**APPENDIX** ..... 59

MATLAB ANN script ..... 59

# LIST OF TABLES


| | | Page |
|---|---|---|
| Table 2.1 | MLP network training algorithms performance | 11 |
| Table 2.2 | Training and test performance errors for neural models trained by three algorithms | 11 |
| Table 2.3 | Time delay for different segment spacing | 16 |
| Table 4.1 | Results of neural network for Case 1 | 39 |
| Table 4.2 | Results of neural network for Case 2 | 44 |
| Table 4.3 | Comparison of NNs with different number of input data points for Case 1 | 49 |
| Table 4.4 | Comparison of NNs with different number of input data points for Case 2 | 51 |

# LIST OF FIGURES


| | | Page |
|---|---|---|
| Figure 2.1 | Structure of a MLP neural network | 7 |
| Figure 2.2 | Flowchart demonstrating neural-network training, neural model testing, and use of training, validation, and test data sets in ANN modelling approach | 10 |
| Figure 2.3 | Serpentine structure | 15 |
| Figure 2.4 | E-field at 0.5 mm above reference plane at 1 GHz, for (a) s = 0.15 mm and (b) s = 1.5mm, with segment length = 4.5mm | 16 |
| Figure 2.5 | Practical example of a differential serpentine delay microstrip line (DSDML) on an industrial printed circuit board (PCB) | 17 |
| Figure 3.1 | Data collection through simulation using ADS | 21 |
| Figure 3.2 | Artificial neural network implementation in this project | 22 |
| Figure 3.3 | Example of meander line | 23 |
| Figure 3.4 | Generated layout of meander line | 24 |
| Figure 3.5 | S-parameters from circuit simulation and momentum simulation | 25 |
| Figure 3.6 | Transient modelling of meander line | 26 |
| Figure 3.7 | NN training window | 29 |
| Figure 4.1 | Meander line schematic diagram | 34 |
| Figure 4.2 | Comparison between S-parameters from circuit simulation and momentum simulation | 35 |

| | | |
|---|---|---|
| Figure 4.3 | Delay time simulated using ADS Momentum | 36 |
| Figure 4.4 | Delay time simulated using circuit simulator | 36 |
| Figure 4.5 | A sample structure of the neural network for Case 1 | 38 |
| Figure 4.6 | Comparison of results from ANN and ADS during training for Case 1 | 40 |
| Figure 4.7 | Comparison of results from ANN and ADS during evaluation for Case 1 | 41 |
| Figure 4.8 | Lowest validation error versus epochs for Case 1 | 42 |
| Figure 4.9 | Highest validation error versus epochs for Case 1 | 42 |
| Figure 4.10 | A sample structure of the neural network for Case 2 | 45 |
| Figure 4.11 | Comparison of results from ANN and ADS during training for Case 2 | 46 |
| Figure 4.12 | Comparison of results from ANN and ADS during evaluation for Case 2 | 46 |
| Figure 4.13 | Lowest validation error versus epochs for Case 2 | 47 |
| Figure 4.14 | Highest validation error versus epochs for Case 2 | 48 |

# LIST OF ABBREVIATIONS

| | |
|---|---|
| 2.5D | 2.5 Dimensional |
| 3D | 3 Dimensional |
| ADS | Advanced Design System |
| ANN | Artificial Neural Network |
| CAD | Computer Aided Design |
| CGP | Conjugate gradient with Polak–Ribiere updates |
| EM | Electromagnetic |
| EMI | Electromagnetic Interference |
| GD | Gradient descent |
| HEMT | High electron mobility transistor |
| IC | Integrated Circuit |
| LM | Levenberg–Marquardt |
| MLP | Multilayer Perceptron |
| MSE | Mean Square Error |
| NN | Neural Network |
| PA | Power Amplifier |
| PCB | Printed Circuit Board |
| RAM | Random access memory |
| RF | Radio Frequency |
| S2P | 2 port S-parameter file block |
| SLFN | Single Hidden Layer Feedforward Network |
| SNR | Signal to Noise Ratio |
| TLFN | Two Hidden Layer Feedforward Network |
| trainbfg | BFGS Quasi-Newton Backpropagation |

| | |
|---|---|
| traincgb | Conjugate Gradient with Powell/Beale Restarts |
| traincgf | Fletcher-Powell Conjugate Gradient |
| traincgp | Polak-Ribiére Conjugate Gradient |
| traingdx | Variable Learning Rate Backpropagation |
| trainlm | Levenberg-Marquardt Backpropagation |
| trainoss | One Step Secant |
| trainrp | Resilient Backpropagation |
| trainscg | Scaled Conjugate Gradient |

# LIST OF SYMBOLS

| | |
|---|---|
| $E_{Tr}$ | Total means square error for training |
| $J^T$ | Transpose of Jacobian Matrix |
| $N_h$ | Number of hidden neurones |
| $N_{Tr}$ | Length of training data |
| $N_o$ | Number of outputs |
| $N_s$ | Distinct samples |
| $N_{ub}$ | Upper bound of hidden number of neurone |
| $R_{Ev}^2$ | Evaluation performance |
| $R_{Tr}^2$ | Training performance |
| $R_{Tst}^2$ | Testing performance |
| $t_{Ev,ij}$ | Evaluation target |
| $t_{Tst,ij}$ | Testing target |
| $t_{ij}$ | Training target |
| $w_{ji}^h$ | Weight of the link from $i$th neuron in layer $h$ to $j$th neuron in layer $h+1$ |
| $x_j^{h+1}$ | Total input for the neurone in $j$ layer $h+1$ |
| $y_{Ev,ij}$ | Evaluation output |
| $y_{Tst,ij}$ | Testing output |
| $y_i^h$ | State of the $i$th neuron in the preceding $h$th layer |
| $y_{ij}$ | Training output |
| $z_N^{h-1}$ | Input from the previous neurone |
| $\varepsilon_r$ | Relative dielectric constant |
| $\theta_j^{h+1}$ | Threshold of the $j$th neuron in layer $h+1$ |

| | |
|---|---|
| $\mu_{Ev,i}$ | Mean of $t_{Ev,ij}$ |
| $\mu_{Tst,i}$ | Mean of $t_{Tst,ij}$ |
| $\mu_i$ | Mean of $t_{ij}$ |
| GHz | Gigahertz |
| L | Neural network layer |
| M | Number of output |
| mm | millimetre |
| ps | Picosecond |
| s | Segment spacing |
| T | Target |
| V | Volt |
| X | Input |
| Y | Output |
| Ω | Ohm |
| $H$ | Hessian matrix |
| $J$ | Jacobian Matrix |
| $ceil$ | Rounding towards positive infinity |
| $e$ | Vector of network errors |
| $f(X)$ | Input function |
| $g$ | Gradient |

# KECERDASAN BUATAN UNTUK SIMULASI LITAR GELOMBANG MIKRO

## ABSTRAK

Sejak beberapa dekad yang lalu, Jaringan Saraf Tiruan (ANN) muncul sebagai alat yang popular untuk menganalisis pelbagai parameter dalam pemodelan peranti gelombang frekuensi radio (RF) dan gelombang mikro. Analisis ini berjaya membuktikannya cepat dan tepat dalam teori dan eksperimen daripada kerja yang lalu. Pada masa kini, talian kelewatan dan integriti isyarat main peranan yang penting dalam peranti gelombang mikro. Dalam penyelidikan ini, talian Serpentine atau talian Meander akan disiasat berdasarkan hubungan antara lengah perambatan dan parameter fizikal talian mikro-jalur. Parameter kritikal yang dipertimbangkan ialah kelebaran jalur, kepanjangan jalur, jarak antara jalur, bilangan selekoh dan jenis selekoh. Parameter ini akan dijadikan sebagai pasangan input-sasaran data untuk diberikan kepada rangkaian neural untuk sesi latihan dan sesi pengesahan. Momentum dalam Advanced Design System (ADS) digunakan untuk mensimulasikan talian Meander kerana ia merupakan penyelesai elektromagnet, bagi mendapatkan S-parameters dan menjanakan susun atur litar. S-parameter yang dihasilkan akan digunakan dalam model transient untuk menentukan lengah perambatan dalam talian meander. MATLAB digunakan dalam penyelidikan ini untuk mewujudkan model rangkaian neural yang boleh meramal lengah perambatan. Kaedah penghampiran diguna untuk menentukan bilangan neuron yang tersembunyi untuk mengoptimumkan prestasi rangkaian neural dari segi kelajuan melatih dan ketepatan hasilnya. Akhirnya, simulasi ADS dan ANN akan dibandingkan untuk mengesahkan prestasi dan mewajarkan ketepatan analisis yang dicadangkan. Keputusan menunjukkan ANN mencapai ketepatan melebihi 0.995 (dengan rujukan 1.0) dan masa latihan yang kurang daripada satu saat.

# ARTIFICIAL INTELLIGENCE FOR MICROWAVE CIRCUIT SIMULATIONS

## ABSTRACT

For the past decades, the Artificial Neural Networks (ANNs) have emerged as a popular tool for analysing the various type of parameters in radio frequency (RF) and microwave device modelling and designing. This type of analysis has been shown to be fast and accurate, both theoretically and experimentally from the past work. In today high-speed digital world delay lines and signal integrity plays an important role in microwave devices. In this work, serpentine lines or meander lines will be investigated base on their correlation between the propagation delay and the physical parameters of the microstrip lines. Some of the critical parameters to be considered while designing the microstrip line are serpentine line width, length, spacing, the number of bends and types of bends. These parameters will be the input-target pairs of data to be fed to the neural network for training session and validation purpose later on. The Momentum simulator in Advanced Design System (ADS) will be used to simulate the meander lines as it is an electromagnetic solver, to get the S-parameters and generating layout. The S-parameters generated will be used to create the transient modelling which can determine the propagation delay in meander lines. MATLAB is used in this research to create the desired neural network model for propagation delay prediction. Approximation method is employed to find the best number of hidden neurones which can optimise the performance of the neural network in term of the training speed and the accuracy. Finally, both the ADS and ANN results for simulated delay times of meander lines are compared to validate the performance and to justify the exactitude of proposed analysis. The results indicate that the ANN achieves an accuracy of above 0.995 (with a reference of 1.0) with a training time of less than a second.

# CHAPTER 1

# INTRODUCTION

## 1.1 Research Background

Artificial Neural Network (ANN) is an information-processing system that is modelled after the neuronal structure that is based on the studies of the human brain. ANN resembles the brain in two aspects [1]: (1) the ability to acquire knowledge through a learning process by the network, and (2) the knowledge is stored in the interneuron connection strengths known as synaptic weights. These remarkable attributes of the ANN which lead to other processing characteristics such as nonlinearity, high parallelism, fault and noise tolerance, learning and generalisation capabilities make it attractive to the researcher to use it in a variety of fields [2].

In the last few decades, many ANN applications in microwave engineering have been reported. The reason to these emerging technologies were ANNs have the ability to model poorly know physical processes in some specific structures (neural networks are trained to model the behaviour of the circuit) [3]. A more common reason was ANN able to accelerate the optimisation and modelling of microwave and RF circuits.

In consideration of full wave modelling with the help of computer-aided design (CAD) tools can give accurate results but still require a lot of time and high computing power. This is true as the technologies keep evolving with increasing complexity of microwave and RF devices, there is a need for a faster method to design this kind of circuits. Thus, there is a need for researchers to look for alternative approximation techniques in modelling such simulation.

ANN are often trained to capture arbitrary input-output relationship to any degree of accuracy [4]. A well trained neural network can be used repetitively which is more time

efficient compared to conventional CAD method. The trained neural network model delivers the predicted output parameters very quickly. CAD models using ANN techniques also require reduced memory requirements compared to conventional approaches and allows for near real-time control.

## 1.2 Problem Statement

Microwave/RF devices are widely used in today's digital world. The full-wave electromagnetic (EM) solvers have been utilised to design these kinds of circuits for a long time. Generally, a lot of simulations are required to meet the specifications which take a considerable amount of time [5]. In order to achieve first pass success with only small adjustment in the manufacturing process, precise EM modelling is a vital condition.

The design process usually involves iterating the design parameters until the final result is realised in spite of the only slight change in the design specifications. The modelling time increases as the transfer function order increase. For example, modelling meander lines required varying a lot of parameters such as lines width, length, the number of bends, type of bends, substrate parameters and etc. When the number parameters are more, the EM solver tends to become slow [6]. With the increasing complexity of the EM circuits, there is a need for a faster method to design this kind of circuits.

Thus, ANN techniques can be applied in cases like this in which device physics are not fully understood, but device output for specified input is known which make it a popular choice among the designers. Thus, this will ease the work of the researcher to quickly get any input-output data without knowing the complex details of the actual structure. Once a model is developed it can be used over and over again. The trained model delivers the output parameters very quickly. This avoids any EM simulation where a simple change in the physical dimension requires a complete simulation. Therefore, the new method is

proposed in this thesis where the physical dimensions of the circuit are used as the input for the ANN model to predict the propagation delay times of meander lines.

## 1.3 Objectives of Research

The objectives of this project are:

1. To investigate the applications of artificial neural network for meander lines modelling.
2. To apply and validate the artificial neural network for propagation delay prediction of meander lines.

## 1.4 Scope of Research

In this project, multilayer perceptron (MLP) neural network will be implemented which is commonly used in microwave design. The inputs for the neural network build are the physical parameters of the meander lines also known as serpentine lines in printed circuit board (PCB) such as the width and length of the microstrip lines. The propagation delay times will be measured through the CAD simulation and will be the target data for the neural network model. All the input-output pairs data will be collected and used as the training and validation data to check the performance of the neural network model. Furthermore, several neural network models with different training algorithm will be trained to predict the propagation delay and their comparison will be analysed.

## 1.5 Thesis Outline

Generally, the entire content of this thesis is divided into three main chapters. The report begins with the introduction in Chapter 1. This chapter discusses background, problem statement, project objectives, project scope and the thesis outline.

Chapter 2 covers the relevant theoretical background involved in this project. The literature review describes the related analysis, interpretations and summary of past works.

Furthermore, the choices of ANN configurations with the advantages and disadvantages of the selected topology from other researchers will be reviewed.

Chapter 3 emphasised on the methodology of the project which involves the selection of proper training method for the desired neural network model. The project flowchart is also included as graphically represent all the logical decisions and progression of each step involved to complete the project. Calculation and analysis will be done to evaluate the performance of the trained neural network model.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Introduction

Over the last few decades, there has been much research directed at building a network that able to solve many forecasting and decision making modelling problems. This research has led to many developments in forecasting methods. It stands to reason that computing systems that attempt similar task as what human can do will profit tremendously from understanding how humans can perform these tasks and simulating processes to the extent allowed of physical limitations. Hence, this necessity the study and developing artificial neural network (ANN) [7]. Neural networks are attractive alternatives to conventional methods such as numerical modelling methods, which could be computationally expensive, or analytical methods which could be difficult to obtain for new complex devices that do not fully understand the circuit behaviour [8].

## 2.2 Structure of Neural Network

A simplify neural network will be consist of 3 layers that are the input layer, hidden layers and output layer. The input layer is the first layer where the neurone receives the first information or stimulus that are not transferred from a neighbouring neurone. The following layer is the hidden layers that consist of (L-2) layers where the neurone will be receiving their information from the input layer, process it and pass it to connected neighbouring neurones. The output layer is the final layer where neurones will produce an output after the information had gone through the whole processing system. This best describes the most popular structure of the neural network that is multilayer perceptron (MLP) which is commonly used in microwave research.

Based on [9] and the illustrated network in Figure 2.1, they define the total input $x_j^{h+1}$, received by the neuron $j$ in layer $h+1$ is defined as

$$x_j^{h+1} = \sum y_i^h w_{ji}^h - \theta_j^{h+1} \tag{2.1}$$

where $y_i^h$ is the state of the $i$th neuron in the preceding $h$th layer, $w_{ji}^h$ is the weight of the connection link from the $i$th neuron in layer $h$ to the $j$th neuron in layer $h+1$, and $\theta_j^{h+1}$ is the threshold of the $j$th neuron in layer $h+1$. For instance, a neuron from the $h$th layer receives the stimuli from the neurons of $(h-1)$th layer, i.e., $z_1^{h-1}$, $z_2^{h-1}$,…, $z_N^{h-1}$ [8]. Output from a neuron in any layer other than input layer is a monotonic nonlinear function of its total input and is written as

$$y_j^h = \frac{1}{1+e^{-x_j^h}} \tag{2.2}$$

All neurones within a layer, other than the input layer, have their conditions set by (2.1) and (2.2) in parallel while other layers have their states set sequentially until the states of the neurones in the output layer H are determined.

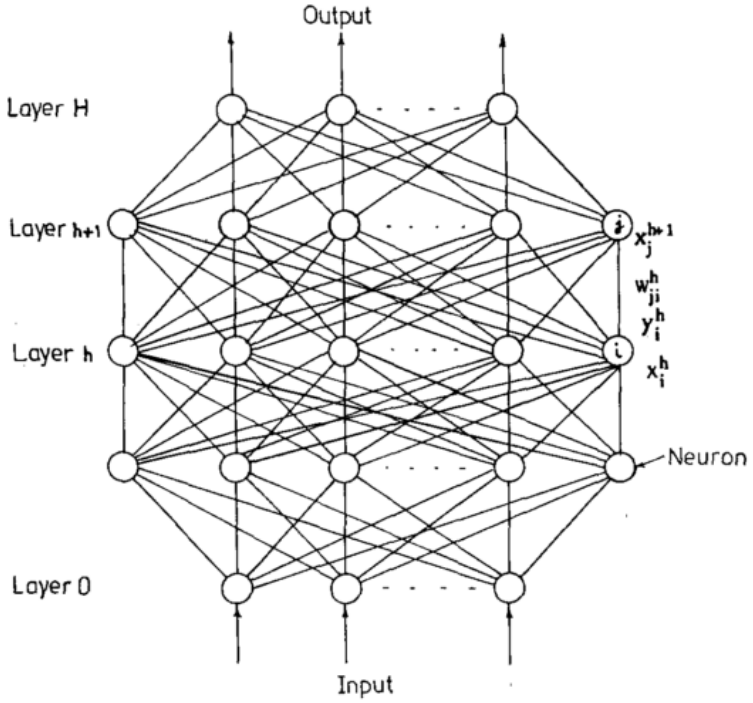



Figure 2.1: Structure of a MLP neural network [9]

In most of the microwave research, a simple MLP with a hidden layer is sufficient to model highly non-linear functions for complex problems. Multiple papers have published on such design; investigate parallel coupled microstrip lines, the small signal and large signal of High-electron-mobility transistor (HEMT), modelling relative intensity noise and terminal electrical noise, determining resonate behaviour of a split ring resonator and modelling power amplifier (PA) memory effect [10]-[14].

## 2.3 Neural Learning

In ANN, learning can be referring to the methods of modifying the weights of connection links between the nodes of the specified network. Learning is the process of adapting random value parameters, weight and bias of the neural network through a continuous process of simulation within the environment set. There are three major learning paradigms which are supervised learning, unsupervised learning and reinforcement learning.

The most common method of machine learning, deep or not, is the supervised learning. Supervised learning involves training of an ANN with the right answers (i.e., target outputs) for each example given, and using the deviation (error) of the ANN solution from corresponding target values to determine the required amount by which each weight should be adjusted. To properly adjust the weight vector, the learning algorithm will compute a gradient vector. For each weight, it will decide by what amount the error would increase or decrease if the weight were increased by a slight amount. The weight vector is then adjusted in the opposite direction to the gradient vector [15]. For example; you use an algorithm to learn the mapping function from the input (X) to the output (Y).

$$Y = f(X) \tag{2.3}$$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

The researchers have introduced the unsupervised learning procedures that could create layers of feature detectors without requiring labelled data, which means that this technique does not require any correct answers for the training process. This network will learn to categorise the input into groups (cluster) without being told. This method has the ability to explore the uniqueness of the data and the correlation between the input samples based on their similarity and dissimilarity. Unsupervised learning is normally used to encode raw incoming data such as video or audio [16].

Reinforcement learning is a type of learning that can be considered as an intermediate form of the supervised learning and unsupervised learning. Here the system must discover how to interact with a dynamic, initially unknown environment to maximise their expected cumulative reward signals by doing some action on the environment and gets a

feedback response from the environment. This learning system will grade its action good (reward) or bad (punish) accordingly to adjust its parameters [17].

## 2.4 Training and Validation of Neural Network

The most particular characteristic of a neural network system is its ability to learn through iterations. In the context of artificial systems, learning can be defined as the process of updating the internal representation of the system in response to external stimuli so that the performance of a specific task is improved. The learning algorithm defines how network weights are adjusted between successive training cycles or epochs. Although a number of learning strategies have been developed for multi-layer perceptron model (MLP), the most popular are the backpropagation learning algorithm, also called the generalised delta rule [18]. In this method, the weights are adjusted to minimise the squared difference between the model output and the desired output for an observation in the data set. The squared error is then propagated backwards through the network and used to adjust the weights and biases. Figure 2.2 show a very systematic flowchart of the complete training process for MLP neural network.

Parallel stochastic weight perturbation is a popular algorithm that will be usually employed. In this algorithm, all weights are perturbed simultaneously by a random vector. Then the mean squared error (MSE) is evaluated on the entire training set. If the error decreases, the new vector of weights is accepted; otherwise, it is discarded. This algorithm, however, suffers from a high likelihood of convergence to a local minimum. Hence, training may need to be performed several times before a good solution is found [19]. As the weight vectors move closer to the correct orientation both the MSE may decrease.

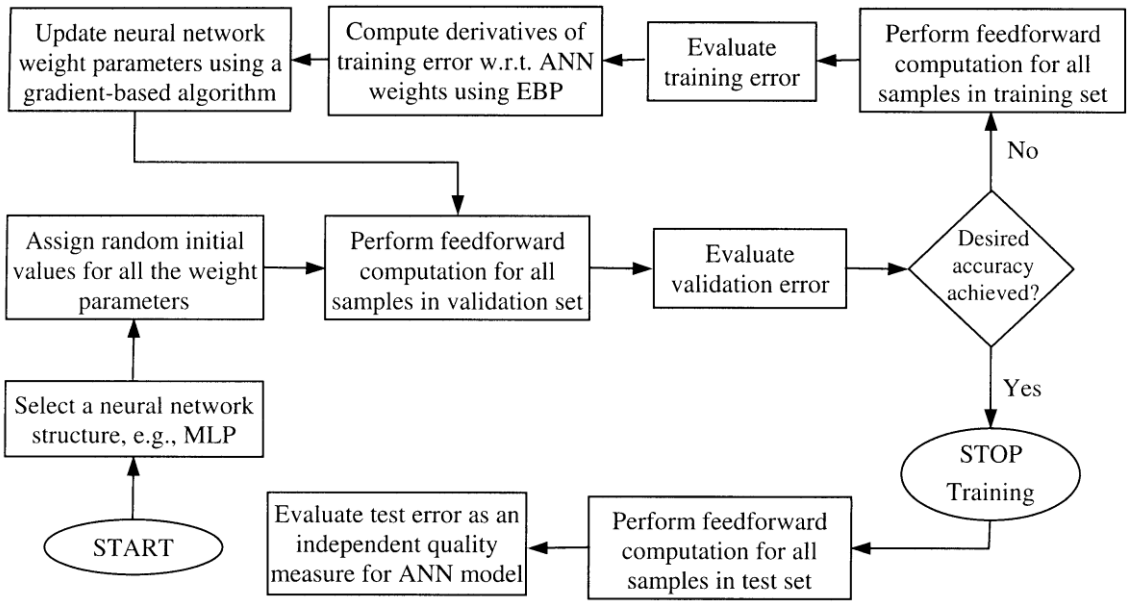



Figure 2.2: Flowchart demonstrating neural-network training, neural model testing, and use of training, validation, and test data sets in ANN modelling approach [8].

## 2.5 Training Algorithms

It is very difficult to know which training algorithm will be the fastest for a given set of problems. It depends on several factors, including the complexity of the problem, the number of data points in the training set, the number of weights and biases in the network, the error goal, and whether the network is being used for pattern recognition (discriminant analysis) or function approximation (regression). There are several algorithms that have been developed from the past work such as Levenberg-Marquardt Backpropagation (trainlm), BFGS Quasi-Newton Backpropagation (trainbfg), Resilient Backpropagation (trainrp), Scaled Conjugate Gradient (trainscg), Conjugate Gradient with Powell/Beale Restarts (traincgb), Fletcher-Powell Conjugate Gradient (traincgf), Polak-Ribiére Conjugate Gradient (traincgp), One Step Secant (trainoss), and Variable Learning Rate Backpropagation (traingdx). All of the algorithms shown above are available in Matlab neural network toolbox. In [20], a comparison of networks with a variety of different architectures and complexities are used, and the networks are also trained to a range of different accuracy levels.

Table 2.1: MLP network training algorithms performance [20]

| | MLP Training Algorithm Types | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Trainlm | Trainrp | Trainbfg | Trainscg | Traincgb | Traincgf | Traincgp | Trainoss |
| Hidden Nodes | 5 | 20 | 4 | 14 | 6 | 22 | 7 | 19 |
| Training (%) | 100 | 98 | 100 | 100 | 100 | 100 | 100 | 98 |
| Validation (%) | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Testing (Ideal) (%) | 98.00 | 88.00 | 98.00 | 92.00 | 98.00 | 96.00 | 98.00 | 96.00 |
| Testing (SNR of 30) (%) | 86.80 | 84.56 | 88.72 | 85.84 | 89.44 | 87.68 | 87.92 | 86.24 |

From the result, less hidden nodes mean less time consumption needed to train the input data and more efficient the system. However, based on their findings traincgb have the best performance among the training algorithms with the highest score percentage in ideal and signal to noise ratio (SNR) of 30% testing.

In another paper, microwave imaging of dielectric cylinders from experimental scattering data is a study based on three different MLP training algorithms. The chosen algorithms are conjugate gradient with Polak–Ribiere updates (CGP), Levenberg–Marquardt (LM) and gradient descent (GD) are used to train the ANN. The comparison performance among these 3 algorithms is shown in Table 2.2.

Table 2.2: Training and test performance errors for neural models trained by three algorithms [21]

| Training Algorithms | Performance | |
|---|---|---|
| | Training data | Test data |
| CGP | 0.0845 | 0.1873 |
| LM | 0.0083 | 0.0274 |
| GD | 0.2831 | 0.3165 |

The findings from this paper clearly show that training function Levenberg–Marquardt (LM) has the lowest error in both training and test validation. There are several algorithm

characteristics that can be deduced from the experiments described. In general, on function approximation problems, for networks that contain up to a few hundred weights, the Levenberg-Marquardt algorithm will have the quickest convergence. This advantage is particularly noticeable if very accurate training is needed. In many cases, trainlm is able to obtain lower mean square errors than any of the other algorithms tested. However, the storage requirements (memory) for trainlm are larger than the other algorithms tested. The conjugate gradient algorithms, in particular, trainscg seem to perform well over a wide variety of problems, notably for networks with a large number of weights. The SCG algorithm is almost as fast as the LM algorithm on function approximation problems (faster for large networks).

## 2.6 Levenberg-Marquardt backpropagation

Generally, LM is one of the most popular tools for non-linear minimum mean squares problems. LM algorithm also approximates to the Newton method and has been also used for ANN training. The Newton method approximates the error of the network with a second order expression, which contrasts to the former category which follows a first-order expression. Because of the properties of fast convergence and stability, the method has been employed in several microwave modelling [22]. When the performance function has the form of a sum of squares (as is typical in training feedforward networks), then the Hessian matrix can be approximated as

$$H = J^T J \tag{2.4}$$

and the gradient can be computed as

$$g = J^T e \tag{2.5}$$

where $J$ is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and $e$ is a vector of network errors. The Jacobian matrix

can be computed through a standard backpropagation method [23] that is much less complex than computing the Hessian matrix.

The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$X_{k+1} = X_k - [J^T J + \mu I]^{-1} J^T e \tag{2.6}$$

When the scalar $\mu$ is zero, this is same as Newton's method, using the approximate Hessian matrix. When $\mu$ is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum. Therefore, the aim is to shift toward Newton's method as quickly as possible. Thus, $\mu$ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function is always reduced at every iteration of the algorithm.

## 2.7 Finding Number of Hidden Neurons

It is very important to choose proper topology for the neural network. Determining the optimal number of hidden nodes is the most challenging aspect of Artificial Neural Network (ANN) design. To date, there are still no reliable ways of determining this a priori, because it depends on so many domain specific factors. Number of neurones in input layer must be the same as the number of input values, and the number of neurones in the output layer is the same as the number of output values. Determination of an optimal number of neurones in the hidden layer is one of the major difficulties faced by researchers in this field. If the number of neurones in the hidden layer is too small, the network may not be powerful enough to meet the desired requirements. On another hand, a large number of hidden neurones can cause very long training and recall time [24].

Hence in practice, the optimal means the 'correct' balance between the number of hidden nodes and generalisation error.

There are several methods proposed to find the optimal number of hidden neurones from past years. One of the methods proposed is the coarse to fine search method to find the number of neurones. First, the number of hidden neurones is initially set using the binary search mode such as l, 2, 4, 8, 16, 32, 64 and 128. Then a sequential fine search will be used in the neighbourhood of the number of hidden neurones that gave the lowest MSE [25].

In [26], they lowered these bounds proving that a single hidden layer feedforward network (SLFN) with at most hidden neurones, $N_h$ can learn distinct samples, $N_s$ with zero error. This is true for any bounded, non-linear activation function which has a limit at one infinity. Thus, the upper bound for SLFNs is given as

$$N_h \leq N_s \tag{2.7}$$

Later as ANN has improved, work have been done to increase the number of hidden layers by proving that the upper bound on the number of hidden nodes $N_h$ for two hidden layer feedforward network (TLFNs) with sigmoid activation function is given by equation (2.8), where $N_o$ is the number of outputs. These can learn at least $N_s$ distinct samples with any degree of precision [27].

$$N_h \leq 2\sqrt{(N_o + 2)N_s} \tag{2.8}$$

## 2.8 Meander Lines/Serpentine Lines

Serpentine lines have been widely used in printed circuit boards (PCB) to provide the required timing delays for timing margin management. Nevertheless, as the processor speed and clock frequency increases, conventional circuits such as delay lines have to be

considered as distributed elements. Based on a full-wave simulation tool such as CAD, some critical parameters that affect the delay characteristic of a serpentine line are carefully studied and analysed.

Layout implementation of serpentine routeing without careful design consideration may lead to timing violation due to unwanted delay variation. Some of the critical parameters considered while designing the serpentine structure include the serpentine line length, line width, spacing, number of bends, and types of bends. Furthermore, the strip line thickness, substrate thickness, dielectric of the substrate, and rise time of signals are another factor that will affect the propagation delay as well. Figure2.3 show an example of a top view of the serpentine strip line structure.

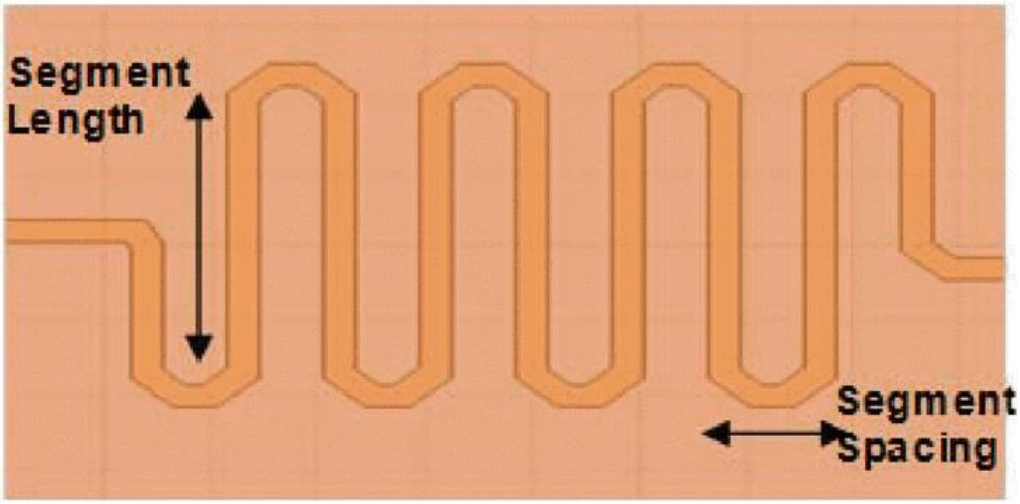


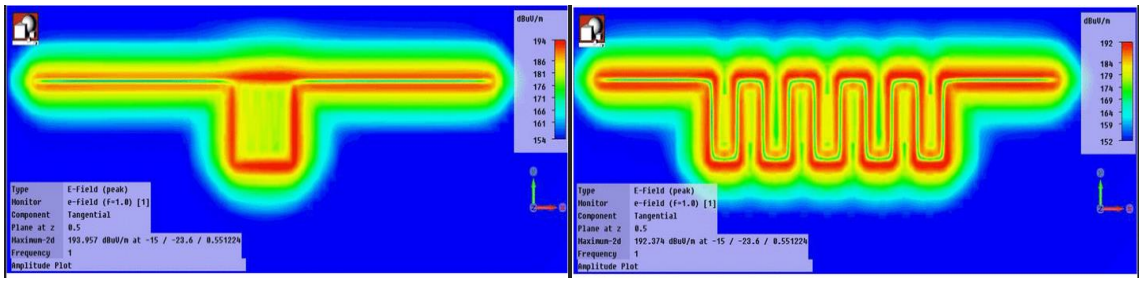

Figure 2.3: Serpentine Structure [28]

From Table 2.3 it can be stated that the straight line has more delay compared to serpentine segments designed with the same conductor length. The primary reason being cross-talk coupling between the segments. In case, of straight line structures, the electric fields are terminated to the ground, on the contrary in serpentine segments, due to cross-talk effect electric fields get coupled to the adjacent segment causing the reduction in delay. The width of the pulse is generally understood to be twice the propagation time through the coupled region.

Table 2.3: Time delay for different segment spacing [28]

| Rise Time | Segment Spacing | | | |
|---|---|---|---|---|
| | $15um$ | $30um$ | $45um$ | Straight Line |
| 25ps | 67ps | 70ps | 72ps | 77ps |
| 50ps | 74ps | 78ps | 79ps | 84ps |
| 100ps | 84ps | 87ps | 88ps | 94ps |
| 200ps | 101ps | 108ps | 108ps | 113ps |

Another paper has studied on the near field distribution on the serpentine delay line as shown in Figure 2.4 (a) and (b) respectively. For the case of segment spacing, s = 0.15mm, beside the field distribution along the serpentine line, very strong field coupling is observed across the adjacent segments. It indicates that besides the usual signal propagation path along the serpentine line, there is another alternative shorter path through the strong coupling between segments, which results in shorter propagation delay of the overall signal. For another case where s = 1.5 mm, the field coupling between segments is rather weak, and hence, the signal propagation is dominated by the path along the serpentine line. It explains why the delay for the serpentine line with s = 1.5mm is much closer to the propagation delay of a straight microstrip line of identical length [29].



(a) (b)

Figure 2.4: E-field at 0.5 mm above reference plane at 1 GHz, for (a) s = 0.15 mm and (b) s = 1.5mm, with segment length = 4.5mm [29]

## 2.9 Differential Signalling With Meander Line

Implementation of meander line in differential signalling is a very popular choice to be used in the electronic industry because this technique able to resists electromagnetic noise compare to one conductor with an unpaired reference(ground). Being dependent on the

difference in the signal level on the paired lines, the differential circuits are relatively insensitive to noises such as the ground bounce that may exist on the power and/or ground plane and to the common-mode signals that may appear equally on each line. Moreover, the differential signals are somewhat immune from the electromagnetic interference (EMI). The employment of differential signalling may lessen the occurrence of crosstalk and improve the signal integrity. Therefore, the differential signalling gradually becomes a common routeing scheme in the PCB layout design rather than the single-ended signalling [30]. A practical example of differential serpentine delay microstrip lines implemented on a PCB in the industry.

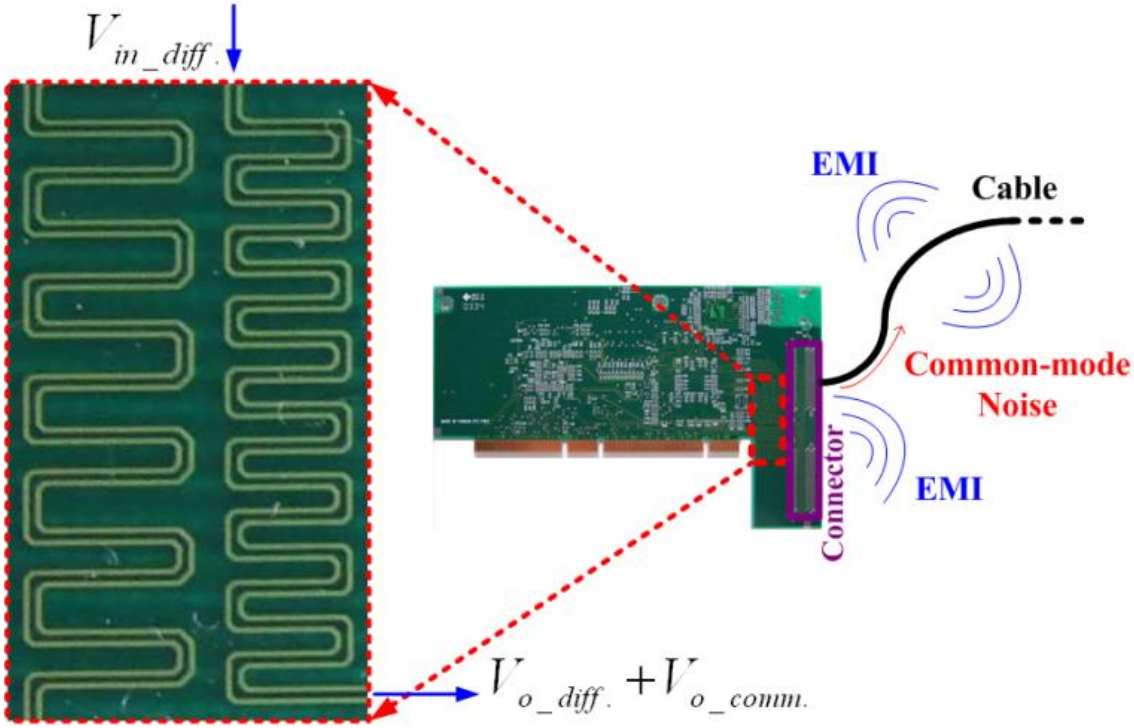



Figure 2.5: Practical example of a differential serpentine delay microstrip line (DSDML) on an industrial printed circuit board (PCB) [31]

## 2.10 Summary

In this chapter, the background of artificial neural network (ANN) and its application in microwave and RF circuits modelling are reviewed. Based on those past work, many researchers have conducted or propose their design of ANN by testing out various architectures, algorithms and configurations to improve the performance of the ANN and

have been applied in a variety of field. Both advantages and disadvantages of ANN compare with the conventional method of using CAD are discussed. Several ANN training algorithms are discussed and compared over each of their characteristics. Finally, the background of meander lines or serpentine lines is discussed which include several factors that need to be considered during the designing stage to get the desired delay time. For this project, there are previous works analysing on meander lines but there are not much ANN work that can be compared to previous researches because using ANN to predict the delay time of meander lines is considered quite new.

# CHAPTER 3

# METHODOLOGY

## 3.1 Introduction

This methodology of this entire project discusses the implementation of ANN in microwave application which is a study on the meander lines by modelling it using ANN. Furthermore, every stage of the process will be discussed such as data collection and the type of microstrip lines to be modelled. The process of setting up the ANN to imitate the behaviour of the meander line will be presented. Finally, the performance of the ANN will be evaluated and analysed.

## 3.2 Project Flow

The software implementation is one of the essential parts in designing the meander lines. It involves designing the schematic and generating layout. The main project flow starts with data collections by using simulation tools in ADS which will be fed to the ANN build later on. This process is done by varying the physical/geometrical parameters of the microstrip lines to generate the S-parameters which will be used in transient modelling. In the transient modelling, the delay times will be recorded. This whole process will be repeated again until sufficient data is collected as shown in Figure 3.1 and more detail processes will be explained later on.

Then, the data collected will be separated into training data, validation data, test data, and evaluation data. After that, the architecture of the neural network and its training algorithm are configured. The best training algorithm will be selected based on trial and error to obtain a good performance which is suitable for this project. Then the neural network will be trained and evaluated using training data and evaluation data respectively. If the neural network does not reach the specified mean square error goal, the whole

process will be reiterated again. This means that a new batch of data may need to be collected again by varying the parameters during serpentine line simulation. Figure 3.2 shows a general flowchart of the method used in this thesis to train a neural network.

The simulation software that is used to design the meander lines is Advanced Design System by Keysight Technologies. The behaviour of S-parameters can be simulated and observed carefully. It can be used to identify parasitic coupling between components which is important in the design to get an accurate result compare to normal circuit simulator without electromagnetic field solver.

All the neural network experiments were carried out using the MATLAB R2016a neural network toolbox. This well-known software is relatively easy to use and implement for first time user trying on artificial neural network.

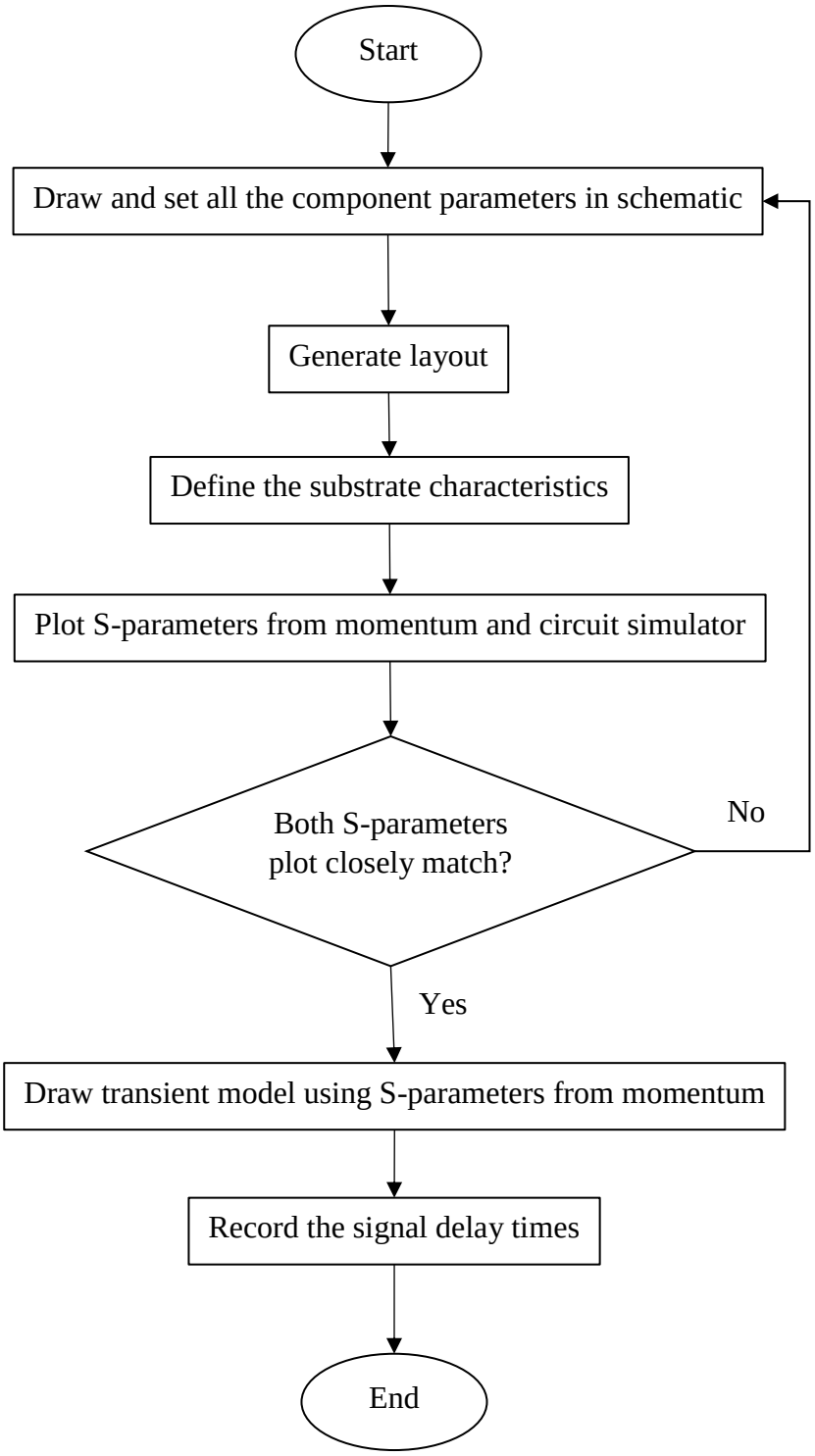



Figure 3.1: Data collection through simulation using ADS

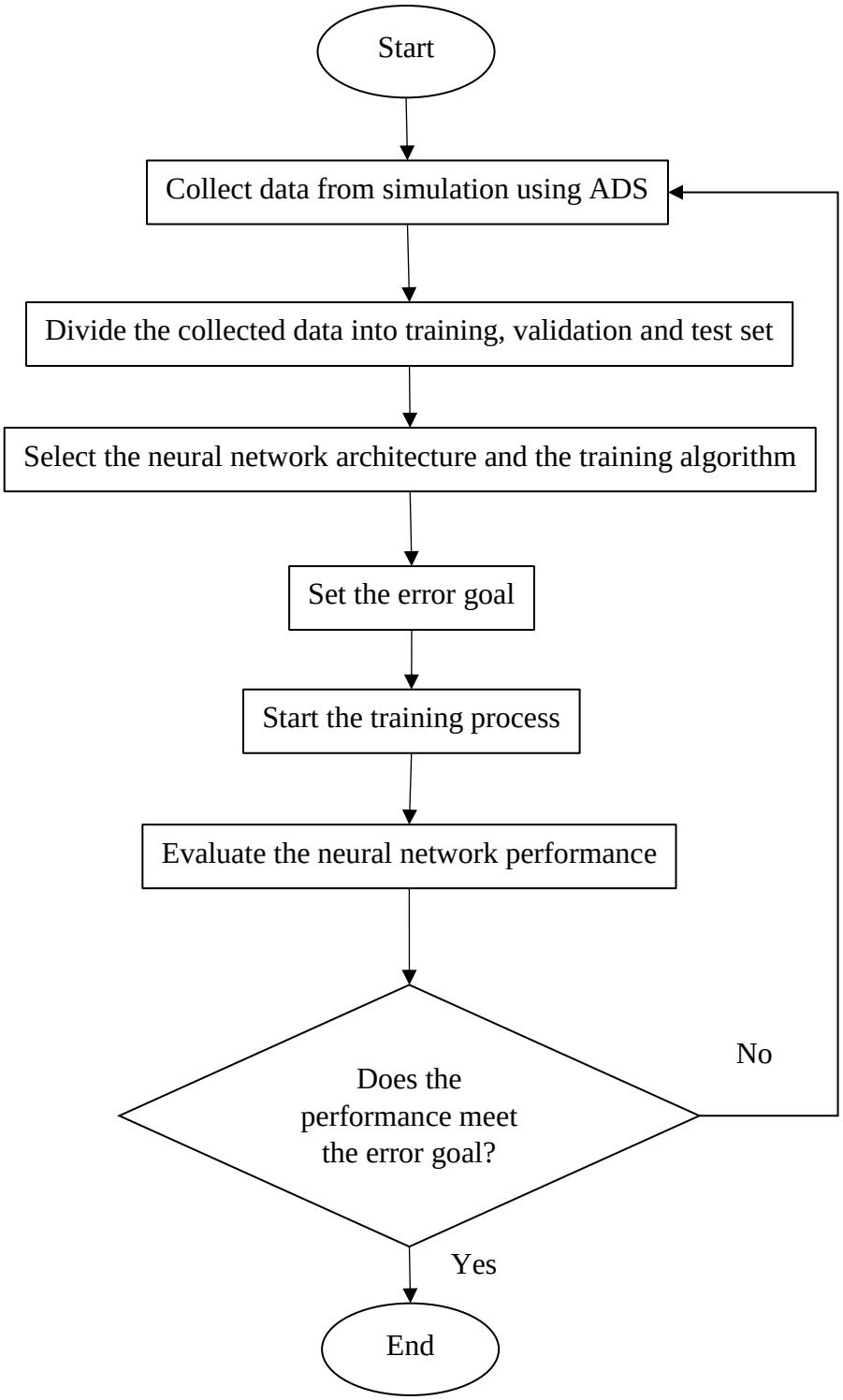



Figure 3.2: Artificial neural network implementation in this project

## 3.3 Data Collection through Simulations

As mention previously, the meander lines simulation is done using ADS and the results are recorded. The serpentine lines are drawn in schematic using the microstrip line which can configure the segment length and segment spacing.

In this project, the work is separated into 2 cases;

(a) Case 1: Effect of segment length on propagation delay time

(b) Case 2: Effect of segment spacing on propagation delay time

In all simulation, the impedances of both ports are specified at 50 Ω to emulate the ports of a network analyser. To match the impedance of the meander lines with the port, LineCalc is used to calculate the width of the lines. The width of the lines will be constant for all cases in this project. The number of turns for the meander lines in this project is set to 12 bends. In this project, the type of bend chosen is the right-angle bend (90 degrees) because it is more common to be used in research or experiment purpose compared to other types of bend such as curve bend and 45-degree bend [29]. Figure 3.3 shows an example of the serpentine line to be modelled in this thesis.

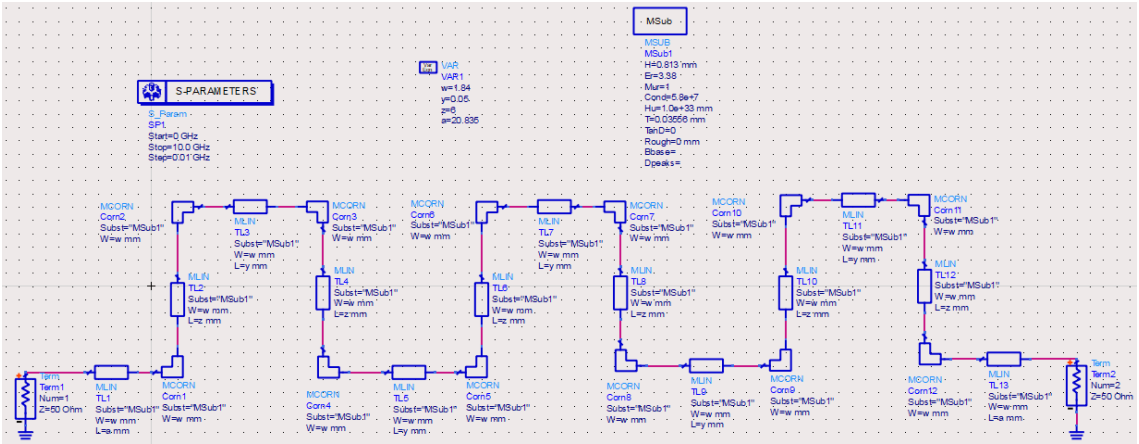



Figure 3.3: Example of meander line

RO4003 is chosen as the substrate due to its tight control on dielectric constant and low loss. Another advantage of this substrate is it offers a wide range of frequency stability. The microstrip substrate parameters are set where the substrate thickness is 0.813mm, the relative dielectric constant, $\varepsilon_r = 3.38$, relative permeability is 1 and the thickness of the conductor is 0.03556mm which assume to be very thin strip. Furthermore, the S-parameters are simulated at starting frequency of 0 GHz and stop at 10 GHz with every step of 0.01GHz. Generally, using a smaller frequency step will make simulation slower but this will give a better result which leads to better accuracy.

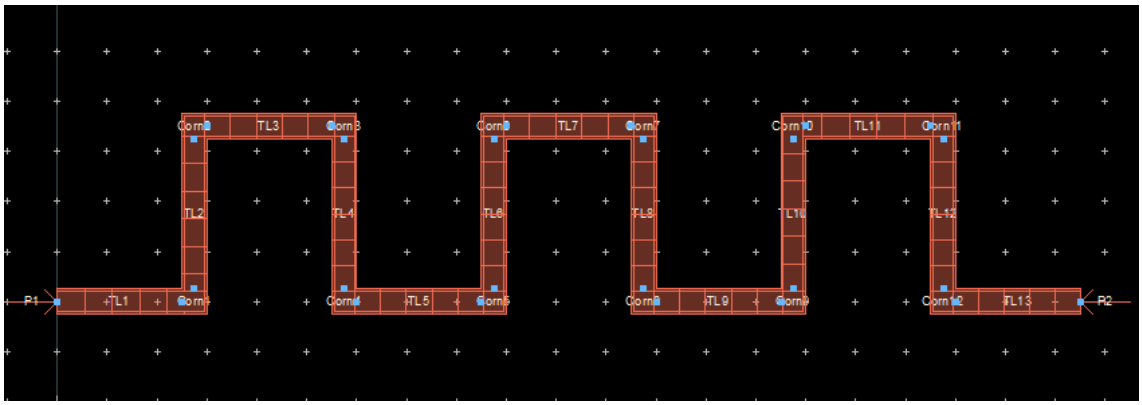



Figure 3.4: Generated layout of meander line

After that layout is generated as shown in Figure 3.4 and momentum is used. Momentum is a part of ADS and it is a simulation tool where it can evaluate and design modern communications systems products. Momentum is an electromagnetic simulator that computes S-parameters for general planar circuits, including microstrip, slotline, stripline, coplanar waveguide, and other topologies. Vias and airbridges connect topologies between layers, so multilayer RF/microwave printed circuit boards can be simulated, hybrids, multichip modules, and integrated circuits. Momentum gives a complete tool set to predict the performance of high-frequency circuit boards, antennas, and ICs. Momentum enables to simulate when a circuit model range is exceeded or the model does not exist. It also can identify parasitic coupling between components which make it a