The Use of Inverse Neural Networks in the Fast Design of Printed Lens Antennas

by

Gurpreet Singh Gosal

A thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

Master of Applied Science
in Electrical & Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

ABSTRACT

In this thesis the major objective is the implementation of the inverse neural network concept in the design of printed lens (transmitarray) antenna. As it is computationally extensive to perform full-wave simulations for entire transmitarray structure and thereafter perform optimization, the idea is to generate a design database assuming that a unit cell of the transmitarray is situated inside a 2D infinite periodic structure. This way we generate a design database of transmission coefficient by varying the unit cell parameters. Since, for the actual design, we need dimensions for each cell on the transmitarray aperture and to do this we need to invert the design database.

The major contribution of this thesis is the proposal and the implementation of database inversion methodology namely inverse neural network modelling. We provide the algorithms for carrying out the inversion process as well as provide check results to demonstrate the reliability of the proposed methodology. Finally, we apply this approach to design a transmitarray antenna, and measure its performance.

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to Dr. Derek McNamara and Dr. Mustapha Yagoub for their support throughout the research work as my advisors. I am very thankful to the trust they have shown in me for taking up this challenging research problem and it is because of their guidance that I was able to tackle it successfully.

I am also very thankful to Dr. Qi-Jun Zhang from the Department of Electronics, Carleton University for providing me with the simulation tools essential as well as for his support to carry out this research.

I am very grateful to Dr. Eqab Almajali who despite being very busy in his own PhD research helped me whenever I faced any issue in my work and offered his insight to overcome them. This thesis being the continuation of his research work, I would like to extend my gratitude to Dr. Nicolas Gagnon from Communication research Center (CRC) who also helped me whenever I faced any glitches in the problem formulation.

I would especially express my gratefulness to my parents and sisters for their constant support and inspiration. Finally, I would thank my friends and lab-mates for expressing interest and enthusiasm in my work.

PUBLICATIONS

G.Gosal, D.A.McNamara & M.Yagoub, "The Use of Inverse Neural Networks in Transmitarray Antenna Design", *IEEE AP-S Int. Symp.*, Memphis, Tennessee, USA, July 2014.

TABLE OF CONTENTS

LIST OF FIGURES

KEYWORDS:

ANN (Artificial Neural Network)

INN (Inverse Neural Network)

Forward neural network

Design database

S-parameter database

Printed Lens

Transmitarray

Engineered Surfaces

Inversion

Oblique Incidence

Unit cell

Training data

Test data

Neural network learning

Electric field

Polarization

Transmission coefficient

Transmission phase

Transmission amplitude

Incidence angle

Feed

Aperture

CHAPTER 1
Introduction

1.1 TRANSMITARRAY ANTENNAS

A transmitarray (printed lens) antenna, depicted in figure 1.1-1, consists of layers of conducting shapes etched on multiple dielectric sheets. The conductor layers are divided into a lattice of cells, with a conducting shape in each cell. A feed horn (or other appropriate radiator) illuminates the input surface of the lens. There is thus an incident field distribution over the input surface that is not uniform in either amplitude or phase. We wish the transmitarray surface to transform the incoming feed field so that the field phase distribution over the output surface of the transmitarray is what we need to provide a particular radiation pattern[1]. The conducting shapes thus vary from cell to cell in order to provide a different transmission coefficient amplitude $|T_n|$ and phase $\angle T_n$ at each cell (say the n-th cell) to ensure the necessary transformation of the incident field at each cell. The feed plus the transmitarray surface forms the transmitarray antenna.

Although the techniques to be described in this paper can in principle be used with any transmitarray geometry, we illustrate their use by applying it to one consisting of three layers of square conducting patches separated by dielectric spacers, as sketched in cross-section in figure 1.1-2. Once we know what $\angle T_n$ must be for each transmittarray element (or cell) the dimensions of the conductors of each element must be selected. In the present case the dimensions a_1 and a_2 on the three conducting element layers constitute the geometrical feature of each lens element whose variation allows one to adjust the $\angle T_n$ values. The specific pair (a_1, a_2) chosen should be that which provides the desired $\angle T_n$ and a transmission coefficient amplitude $|T_n|$ that is as close

[1] In this thesis we will always wish the transmitarray antenna to have a pencil beam, and so the transmitarray surface must transform the incoming feed field so that the field phase distribution over the output surface of the transmitarray is as uniform as possible, and the amplitude at each cell as close to unity as possible.

to unity as possible (as explained in the footnote). As implied above, the $\angle T_n$ values vary as a function of position over the lens aperture, and so the elements are not all identical. The lens is therefore not a periodic structure, although the cell lattice is regular. Nevertheless, in order to determine the database of complex T_n values versus (a_1, a_2) the element is considered to reside in an infinite periodic array of identically-sized elements, the phase of the resulting surface's plane wave transmission coefficient computed, and thus considered to be the T_n for the n-th element. This is repeated for different element sizes and a database of T_n versus feature size (a_1, a_2) is constructed.

The output surface (that is, aperture) of the transmitarray antenna is shown in Fig.1.1-3. This is for illustration purposes only. In practice the transmitarray aperture would be divided into many more cells. The transmitarray can clearly be thought of as a planar array of cells, each of size $d_x \times d_y$. The centre of the m-th cell is (x_m, y_m). Each cell acts as a radiating element in the array. If we desire that the transmitarray antenna has a radiation pattern of some specified shape we can use array excitation synthesis to determine what the amplitude and phase must be at each cell, and then determine (using a pre-determined database) the dimensions of the conducting shapes in each cell needed to achieve this for a known feed incident field.

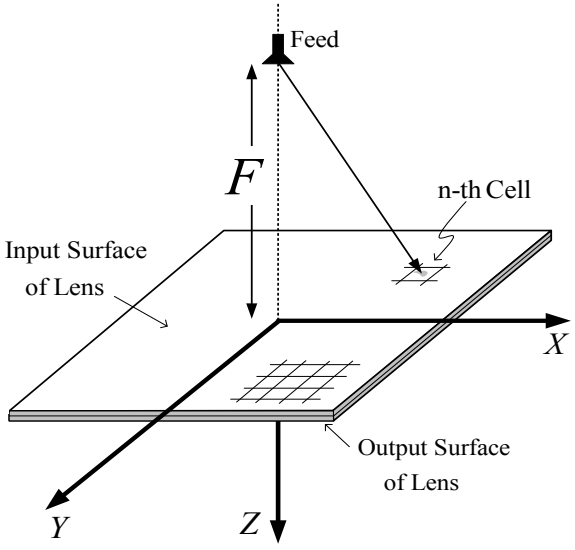


Fig.1.1-1 : Illustration of Printed Transmitarray Antenna Configuration.
(Courtesy of D.A.McNamara)

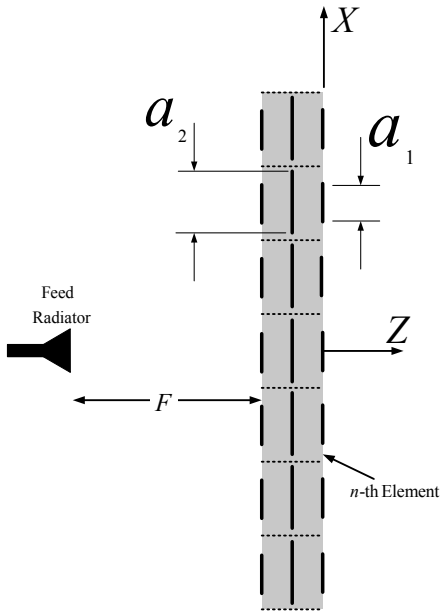


Fig.1.1-2 : Side View of the Horn-Fed Transmittarray Antenna, for the Case of a 3-Conducting-Layer Printed Surface

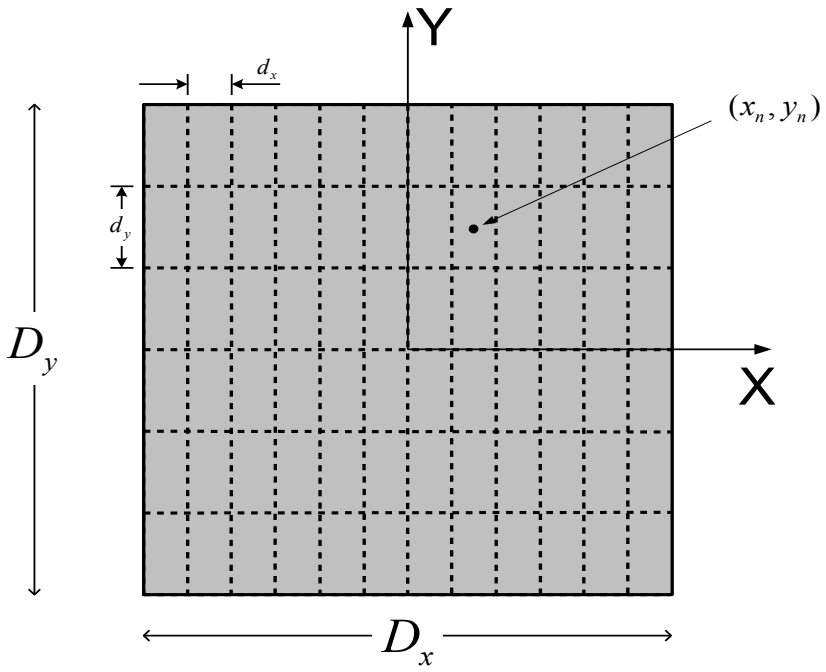


Fig.1.1-3 : Quantized Output Aperture of the Transmitarray Antenna.
(Courtesy of D.A.McNamara)

1.2 THE DATABASE INVERSION PROBLEM

The final step in the design of the transmitarray antenna involves the selection of the correct element size (a_1, a_2) for each unit cell to obtain the desired phase at points over the transmitarray output surface. This selection can be done relatively easily if there is a single geometrical feature that needs to be selected for each element in turn. However, if there is more than one feature, such as the two quantities a_1 and a_2 in the present case, the selection process (the process that needs to "invert" the database) is more difficult. It is the aspect that is of interest in this thesis. This inversion process will be accomplished through the use of inverse neural networks (INNs), something that has not yet been done for the transmitarray and other aperture antennas that use engineered surfaces.

1.3 OVERVIEW OF THE THESIS

The principal goal of this thesis is the implementation of the inverse neural network concept for the inversion of the design database used in the transmitarray design process. Chapter 2 first describes the design procedure usually used for transmitarray antennas, and explains the use of the design database of transmitarray elements. The work described here could only be undertaken after a thorough study of the existing neural network concepts, and in particular inverse neural network ideas. The second part of Chapter 2 therefore provides a review of these concepts.

The most significant contributions of this thesis are developed in Chapters 3, 4 and 5. Although inverse neural network (INN) ideas have been described and used by others for RF circuit design, they do not appear to have been used in detailed antenna design work. In Chapter 3 we therefore take existing neural network ideas and adapt them to the transmitarray type problem at hand. We show how that it is not practical to apply the INN concepts directly to the database of points obtained directly from full-wave simulations; too many such simulations would be needed. It is demonstrated how the distribution of these initial full-wave database points can be examined to decide on how to best use forward (as opposed to inverse) neural network modelling

to augment the number of full-wave database points by a factor of about fifty, to have sufficient training data for implementing INNs accurately. It is then shown that the highly non-linear and multivalued (that is, not one-to-one) nature of the data necessitates the use of inverse sub-models. The inverse sub-model theory available in the literature is altered slightly to suite some database inversion issues possibly unique to the antenna design problem. This latter extension offers some flexibility that might be of benefit in the design of this class of antennas in the future. All the algorithms needed are given in the form of flow-charts and detailed pseudo-code for easy implementation by others. This has not been available elsewhere in the literature.

Chapter 4 applies forward and inverse neural networks to the representation and inversion of transmitarray design databases when not only the transmitarray element conductor dimensions are variables, but also the incidence angle of the field incident on the transmitarray cell. Chapter 5 then actually applies the methods developed in Chapter 3 to design several transmitarrays. These designs are analysed using full-wave 3D electromagnetic models to demonstrate the effectiveness of the forward and inverse neural network models developed.

Finally, in Chapter 6, we conclude with a list of the contributions made by the work of this thesis. We will also comment on the future potential applications of inverse neural networks in the design of aperture antennas that utilize engineered surfaces.

CHAPTER 2
Review of Transmitarray Design Techniques & Neural Network Concepts

2.1 INTRODUCTION

This chapter introduces the concept of transmitarray antennas more thoroughly than it was done in Section 1.1. It describes the working of the transmitarray in detail, the parameters vital for its design, and how these parameters are involved in the design process in Section 2.2. Following that, possible *design approaches* are listed that can be employed to successfully design a transmitarray for given specifications, in Section 2.3. Then, in Section 2.4 the technique that we will use is introduced in terms of database construction that stores the variation of transmission coefficient with respect to the unit cell dimensions. Section 2.5 consequently describes how this database is used in the design process. Existing neural network *design approaches* in use for the design of printed high-directivity aperture antennas are reviewed in Section 2.6. Finally, the neural network modelling concept, as utilized in this thesis is discussed, and conclusions drawn, in Section 2.7 and Section 2.8, respectively.

2.2 TRANSMITARRAY ANTENNA DESIGN AS AN ARRAY PROBLEM

2.2.1 PRELIMINARIES

The essence of the transmitarray antenna is shown in Figure 2.2-1. It consists of layers of conducting shapes etched on multiple dielectric sheets. In other words, the resulting lens is made up of layers of conducting shapes separated by dielectric spacers, as illustrated in Figure 2.2-2. The conductor layers are divided into a rectangular lattice of cells, with a conducting shape in each cell. The cell size is kept to less than $\lambda_0/2$. We may refer to the first and last layers of the

lens as the input and the output surfaces, respectively. This assumes we will always consider the transmitarray antenna as if it were being used as a transmitting antenna; reciprocity ensures that there is no loss of generality in doing so. A feed horn (or any other appropriate radiator) illuminates the input surface. There is thus an incident field distribution over the input surface that is not uniform in either amplitude or phase. The purpose of the transmitarray structure is to correct this so that the field distribution over the output surface of the lens is what we need to provide a particular radiation pattern. The conducting shapes thus vary from cell to cell in order to provide a different transmission coefficient amplitude and phase at each cell to ensure the necessary transformation of the incident field by each cell, so that the field amplitude and phase at the output surface of each cell is what we want it to be. The feed plus the amplitude/phase transformation surface forms the transmitarray antenna.

The output surface (that is, aperture) of a transmitarray antenna is shown in Figure 2.2-3. This is for illustration purposes only. In practice the aperture would be divided into many more cells. The transmitarray can clearly be thought of as a planar array of cells, each of size $d_x \times d_y$. The centre of the m-th cell is (x_m, y_m). Each cell acts as a radiating element in the array. If we desire that the antenna have a radiation pattern of some specified shape we can use array excitation synthesis to determine what the amplitude and phase must be at each cell, and then determine (using a pre-determined database) the dimensions of the conducting shapes in each cell needed to achieve this for a known feed incident field. The rim of the transmitarray aperture can be of any shape. a circular aperture (that we used in this thesis) is illustrated in Figure 2.2-3, the decision to include or exclude a cell from the "parent" rectangular aperture being based on a criterion such as

$$\begin{aligned} &\sqrt{x_m^2 + y_m^2} \le \frac{D}{2} \quad \Rightarrow \quad \text{Include m-th Cell} \\ &\sqrt{x_m^2 + y_m^2} > \frac{D}{2} \quad \Rightarrow \quad \text{Exclude m-th Cell} \end{aligned} \tag{2.2-1}$$

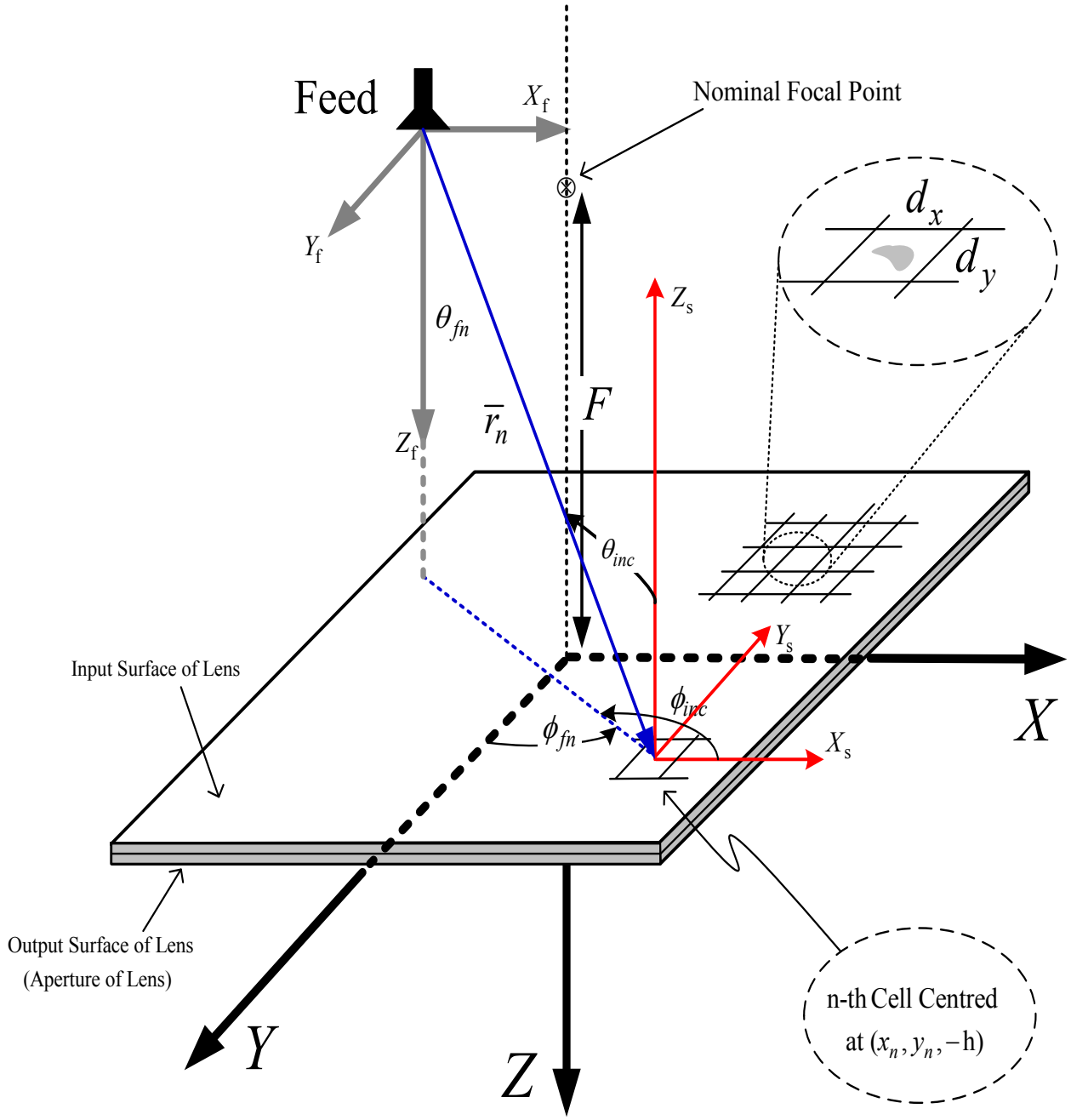


Figure 2.2-1 Transmitarray Configuration.

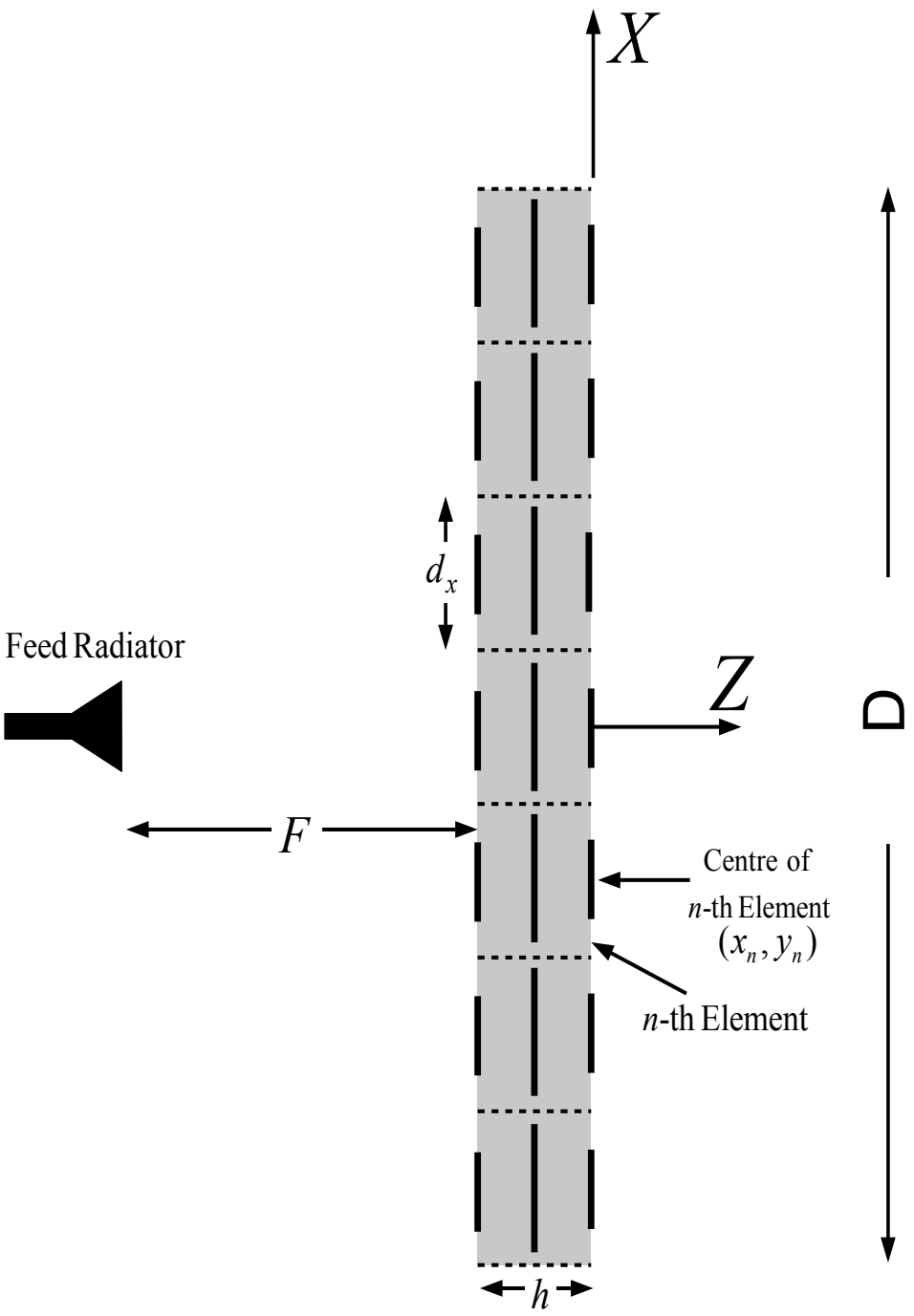


Figure 2.2-2 Cross-Section through a Typical Transmitarray.

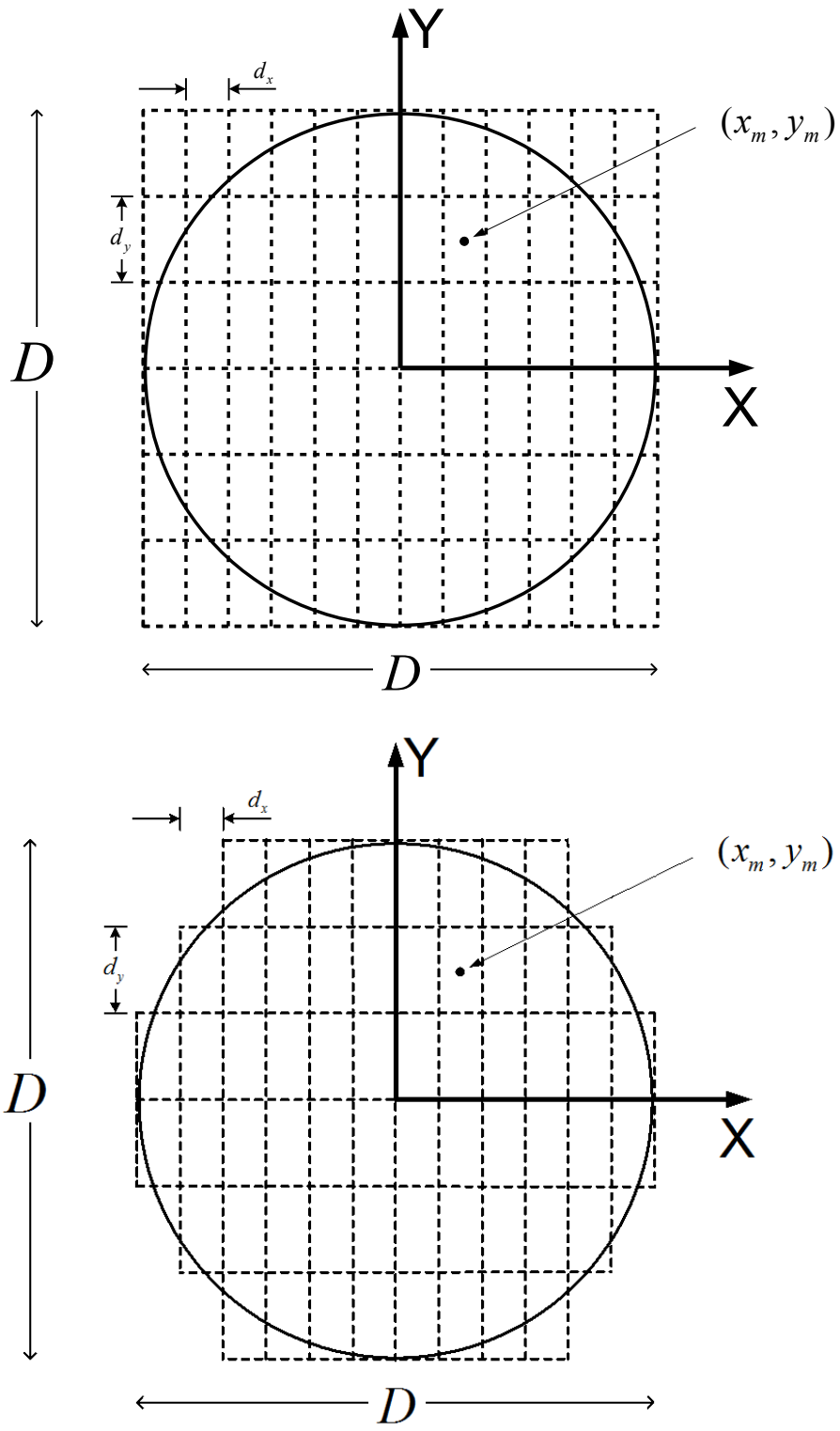


Figure 2.2-3 Quantized Output Aperture of the Transmitarray Antenna.

2.2.2 COORDINATE SYSTEMS

A) Definition of the Coordinate Systems

We refer to Figure 2.2-1, which shows three different coordinate systems, whose definitions are summarised in Table 2.2-1.

Table 2.2-1 The Three Coordinate Systems Used

Coordinate System	Description
X Y Z	Radiation pattern coordinate system for the transmitarray. The $z = 0$ plane is coincident with the output surface (that is, the aperture) of the transmitarray. This will be referred to as the global coordinate system
X_s Y_s Z_s	The coordinate system of the n-th cell (that is, the cell whose transmission coefficient T_n is required). It is centered on the n-th cell.
X_f Y_f Z_f	The feed coordinate system. It is the coordinate system for which we will typically compute the feed fields.

B) Relationships Between the Coordinate Systems

In the global coordinate system the centre of the n-th cell is at the point $(x_n, y_n, -h)$, and that element has the origin of its element coordinate system located there.

The global coordinates of the origin of the feed coordinate system are (x_o, y_o, z_o). In most cases (x_o, y_o, z_o) will be at the nominal focal point of the transmitarray, so that $(x_o, y_o, z_o) = (0, 0, -\mathrm{F} - \mathrm{h})$. Usually the feed phase centre will be located at this nominal focal point.

A point with coordinates (x_s, y_s, z_s) in the element coordinate system has global coordinates

$$x = x_s + x_n \tag{2.2-2}$$

$$y = y_s + y_n \tag{2.2-3}$$

and

$$z = z_s - h \tag{2.2-4}$$

with $(x_n, y_n, -h)$ the location of the element coordinate system origin expressed in global coordinates. The associated unit vectors are

$$\hat{x}_s = \hat{x} \tag{2.2-5}$$

$$\hat{y}_s = -\hat{y} \tag{2.2-6}$$

and

$$\hat{z}_s = -\hat{z} \tag{2.2-7}$$

A point with coordinates (x_f, y_f, z_f) in the feed coordinate system has global coordinates

$$x = x_o + x_f \tag{2.2-8}$$

$$y = y_o + y_f \tag{2.2-9}$$

and

$$z = z_f - (F + h + z_o) \tag{2.2-10}$$

The associated unit vectors are

$$\hat{x}_f = \hat{x} \tag{2.2-11}$$

$$\hat{y}_f = \hat{y} \tag{2.2-12}$$

and

$$\hat{z}_f = \hat{z} \tag{2.2-13}$$

C) Determination of the Feed Field Incidence Angles

We write down the basic relations

$$\overline{r}_n = (x_n - x_o)\hat{x} + (\mathrm{y}_n - y_o)\hat{x} + (\mathrm{z}_n - z_o)\hat{x} \tag{2.2-14}$$

$$|\overline{r}_n| = \sqrt{(x_n - x_o)^2 + (\mathrm{y}_n - y_o)^2 + (\mathrm{z}_n - z_o)^2} \tag{2.2-15}$$

$$\hat{r}_n = \frac{\overline{r}_n}{|\overline{r}_n|} \tag{2.2-16}$$

$$\overline{t}_n = (x_n - x_o)\hat{x} + (\mathrm{y}_n - y_o)\,\hat{\mathrm{y}} \tag{2.2-17}$$

and

$$\hat{t}_n = \frac{(x_n - x_o)\hat{x} + (\mathrm{y}_n - y_o)\,\hat{\mathrm{y}}}{\sqrt{(x_n - x_o)^2 + (\mathrm{y}_n - y_o)^2}} \tag{2.2-18}$$

The definition of the dot-product of two vectors then yields the relation

$$-\hat{t}_n \cdot \hat{x}_s = \cos\phi_{inc} \tag{2.2-19}$$

Use of (2.2-5) and (2.2-13) reduces this to

$$-\frac{(x_n-x_o)\hat{x}+(\mathrm{y}_n-y_o)\,\hat{\mathrm{y}}}{\sqrt{(x_n-x_o)^2+(\mathrm{y}_n-y_o)^2}}\cdot\hat{x}=-\frac{(x_n-x_o)}{\sqrt{(x_n-x_o)^2+(\mathrm{y}_n-y_o)^2}}=\cos\phi_{inc}$$

and so

$$\phi_{inc}=\cos^{-1}\left\{-\frac{(x_n-x_o)}{\sqrt{(x_n-x_o)^2+(\mathrm{y}_n-y_o)^2}}\right\} \tag{2.2-20}$$

Similarly, we have

$$-\hat{r}_n\cdot\hat{z}_s=\cos\theta_{inc} \tag{2.2-21}$$

and then the use of (2.2-7), (2.2-15), (2.2-16) and (2.2-17) gives

$$\frac{(\mathrm{z}_n-z_o)}{|\bar{r}_n|}=\cos\theta_{inc}$$

and hence

$$\theta_{inc}=\cos^{-1}\left\{\frac{(\mathrm{z}_n-z_o)}{|\bar{r}_n|}\right\} \tag{2.2-22}$$

Expressions (2-19) and (2-21) provide us with $(\theta_{inc},\phi_{inc})$. These are incidence angles of the feed field, assumed locally plane, on the n-th cell.

D) Determination of the Feed Pattern Angles

The definition of the dot-product of two vectors in this case supplies the relation

$$-\hat{t}_n\cdot\hat{x}_f=\cos\phi_{fn} \tag{2.2-23}$$

and because of (2.2-11) and (2.2-18) this becomes

$$\phi_{fn}=\cos^{-1}\left\{\frac{(x_n-x_o)}{\sqrt{(x_n-x_o)^2+(\mathrm{y}_n-y_o)^2}}\right\} \tag{2.2-24}$$

Comparison of (2.2-20) and (2.2-24) reveals that

$$\phi_{fn}+\phi_{inc}=\pi \tag{2.2-25}$$

Similarly, we have

$$\hat{r}_n \cdot \hat{z}_f = \cos\theta_{fn} \tag{2.2-26}$$

Then use of (2.2-13), along with (2.2-15) through (2.2-17) gives

$$\frac{(z_n - z_o)}{|\bar{r}_n|} = \cos\theta_{inc}$$

and so

$$\theta_{fn} = \cos^{-1}\left\{\frac{(z_n - z_o)}{|\bar{r}_n|}\right\} \tag{2.2-27}$$

Comparison of (2.2-22) and (2.2-27) reveals that

$$\theta_{fn} = \theta_{inc} \tag{2.2-28}$$

The correctness of (2.2-25) and (2.2-28) can be seen by inspection of Figure 2.2-1.

2.2.3 FEED FIELD ON THE INPUT SURFACE OF THE TRANSMITARRAY

A) Feed Field on the Input Surface of a Cell

The field of the feed in *the feed coordinate system*, is

$$\bar{E}^f(r_f,\theta_f,\phi_f) = E_r^f(r_f,\theta_f,\phi_f)\hat{r}_f + E_\theta^f(r_f,\theta_f,\phi_f)\hat{\theta}_f + E_\phi^f(r_f,\theta_f,\phi_f)\hat{\phi}_f \tag{2.2-29}$$

If we assume that the amplitude/phase transformation surface is in the far-zone of the feed, then we can instead write

$$\bar{E}^f(r_f,\theta_f,\phi_f) = \left\{F_\theta^f(\theta_f,\phi_f)\hat{\theta}_f + F_\phi^f(\theta_f,\phi_f)\hat{\phi}_f\right\}\frac{e^{-jkr_f}}{r_f} \tag{2.2-30}$$

The feed field incident on the input surface of the n-th cell is then given ***in the feed coordinate system*** by (2.2-30) with

$$r_f = |\bar{r}_n| = \sqrt{(x_n - x_o)^2 + (\mathrm{y}_n - y_o)^2 + (\mathrm{z}_n - z_o)^2} \tag{2.2-31}$$

$$\theta_f = \theta_{fn} = \cos^{-1}\left\{\frac{(\mathrm{z}_n - z_o)}{|\bar{r}_n|}\right\} \tag{2.3-32}$$

$$\phi_f = \phi_{fn} = \cos^{-1}\left\{\frac{(x_n - x_o)}{\sqrt{(x_n - x_o)^2 + (\mathrm{y}_n - y_o)^2}}\right\} \tag{2.3.33}$$

This can be converted into rectangular components ***in the feed coordinate system*** by the transformation

$$\begin{bmatrix} E_{xf} \\ E_{yf} \\ E_{zf} \end{bmatrix} = \begin{bmatrix} \sin\theta_{fn}\cos\phi_{fn} & \cos\theta_{fn}\cos\phi_{fn} & -\sin\phi_{fn} \\ \sin\theta_{fn}\sin\phi_{fn} & \cos\theta_{fn}\sin\phi_{fn} & \cos\phi_{fn} \\ \cos\theta_{fn} & -\sin\theta_{fn} & 0 \end{bmatrix} \begin{bmatrix} E_{rf} \\ E_{\theta f} \\ E_{\phi f} \end{bmatrix} \tag{2.2-34}$$

The relation between the feed coordinate system $\mathrm{X_f\,Y_f\,Z_f}$ and cell coordinate systems $\mathrm{X_s\,Y_s\,Z_s}$ is such that

$$E_{xs} = E_{xf} \tag{2.2-35}$$

and

$$E_{ys} = -E_{yf} \tag{2.2-36}$$

B) Some Background - Oblique Incidence Options

What is meant by oblique incidence? We can set $\phi_{inc} = 0^\circ$ and vary θ_{inc}, or have $\phi_{inc} = 90^\circ$ and vary θ_{inc}, or many other possibilities. The computed reflection phase versus θ_{inc} will not be the same in all cases. Access to some basic expressions for an incoming plane wave will allow us to provide some clarity on this issue, and so are provided next. A plane wave travelling towards the origin of the XsYsZs coordinate system from direction $(\theta_{inc}, \phi_{inc})$ has a propagation direction

vector $\hat{k}_i = -(\hat{x}_s \sin\theta_{inc}\cos\phi_{inc} + \hat{y}_s \sin\theta_{inc}\sin\phi_{inc} + \hat{z}_s \cos\theta_{inc})$. A plane wave travelling in the $\hat{k}_i$ direction has an electric field $\bar{E}_{inc} = \bar{E}_0\, e^{-jk_0 \hat{k}_{inc}\cdot\bar{r}_s}$, where $\bar{E}_0$ is the electric field value at the origin, expressed in the XsYsZs coordinate system and vector $\bar{r}_s = x_s\hat{x}_s + y_s\hat{y}_s + z_s\hat{z}_s$ as usual. Thus the incident plane wave can be written as

$$\bar{E}_{inc} = \bar{E}_0\, e^{-jk_0 e^{jk_0(x_s \sin\theta_{inc}\cos\phi_{inc} + + y_s \sin\theta_{inc}\sin\phi_{inc} + z_s\cos\theta_{inc})}} \tag{2.2-37}$$

The electric field vector of the plane wave must be orthogonal to $\hat{k}_i$, and as a consequence

$$\bar{E}_0 = \hat{x}_s \cos\theta_{inc}\cos\phi_{inc} + \hat{y}_s \cos\theta_{inc}\sin\phi_{inc} - \hat{z}_s \sin\theta_{inc} \tag{2.2-38}$$

θ-polarised incident wave and

$$\bar{E}_0 = -\hat{x}_s \sin\phi_{inc} + \hat{y}_s \cos\phi_{inc} \tag{2.2-39}$$

for a ϕ-polarised incident wave. What these expressions look like for the oblique incidence cases we are interested in here (namely in the x-z and y-z planes) are summarised in Table 3.2-1.

Table 2.2-2 : Terminology Used in Describing Oblique Incidence Cases

Wave Type	Alternative Terminology	Plane of Incidence	Incidence Type	Field Expressions
θ-polarised $\phi_{inc} = 0^\circ$	TM	x-z Plane	Oblique	$\bar{E}_{inc}(x,y,z) = (\hat{x}_s\cos\theta_{inc} - \hat{z}_s\sin\theta_{inc})\, e^{jk_0(x_s\sin\theta_{inc} + z_s\cos\theta_{inc})}$
			Normal	$\bar{E}_{inc}(x,y,z) = \hat{x}_s\, e^{jk_0 z_s}$
ϕ-polarised $\phi_{inc} = 90^\circ$	TE	y-z Plane	Oblique	$\bar{E}_{inc}(x,y,z) = -\hat{x}_s\, e^{jk_0(y_s\sin\theta_{inc} + z_s\cos\theta_{inc})}$
			Normal	$\bar{E}_{inc}(x,y,z) = -\hat{x}_s\, e^{jk_0 z_s}$

2.2.4 FIELD ON THE OUTPUT SURFACE OF THE TRANSMITARRAY

Figure 2.2-2 shows a sketch of a transmitarray that uses three-layer strip elements, for example. The properties of the elements are varied over the xy-plane so that the required pattern

is obtained. This allows us to model the lens antenna excitation synthesis problem as a planar array synthesis one.

The feed field at points on the input side of the cell, expressed in the cell coordinate system components (albeit in terms of the global system's spatial coordinate), will for convenience be rewritten as

$$E_x^{in}(\mathrm{x}_n, \mathrm{y}_n, -\mathrm{h}) = E_{xs}(\mathrm{x}_n, \mathrm{y}_n, -\mathrm{h}) \tag{2.2-40}$$

and

$$E_y^{in}(\mathrm{x}_n, \mathrm{y}_n, -\mathrm{h}) = E_{ys}(\mathrm{x}_n, \mathrm{y}_n, -\mathrm{h}) \tag{2.2-41}$$

We next **assume** that the fields on the output surface of the transmitarray taken to be (note the change in the value of the global z-coordinate from the one side of the expression to the other)

$$E_x^{out}(\mathrm{x}_n, \mathrm{y}_n, 0) = T_n^x E_x^{in}(\mathrm{x}_n, \mathrm{y}_n, -\mathrm{h}) \tag{2.2-42}$$

and

$$E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0) = T_n^y E_y^{in}(\mathrm{x}_n, \mathrm{y}_n, -\mathrm{h}) \tag{2.2-43}$$

Quantity T_n^x (T_n^x) is the complex transmission coefficient for the x-component (y-component) of the n-th element. They are assumed to be equal to the transmission coefficient of a plane wave that is incident, with incidence angles $(\theta_{inc}, \phi_{inc})$, on a structure composed of an infinite number of elements along the x-axis that are all identical to the n-th element (which reduces it to a doubly-periodic planar structure). This is of course an approximation, since the n-th element in the actual transmitarray will not be surrounded by elements identical to itself. The values of T_n^x and T_n^x will be functions of frequency, the properties of the substrate material, the widths of the conductors at each layer of the n-th cell, and the incidence angles $(\theta_{inc}, \phi_{inc})$. So we can write then as $T_n^x(a_1, a_2, \theta_{inc}, \phi_{inc}, f)$ and $T_n^y(a_1, a_2, \theta_{inc}, \phi_{inc}, f)$. Tabulated values of T_n^x and T_n^x are referred to as the design database for the particular transformation surface. In writing (2.3-1) we are implicitly assuming that ray-optic tracking of the field is valid; that the feed field travels along a ray from the focal point to the centre of the input surface of the n-th cell and then,

irrespective of $(\theta_{inc}, \phi_{inc})$, parallel to the z-axis through the said cell[2] to the output surface. This is of course not rigorously correct but is the conventional approach that has been used successfully in transmitarray.

2.2.5 TRANSMISSION COEFFICIENT REQUIRED FOR EACH CELL

We **assume** that the principal polarization of the feed is such that only E_y^{in} need be considered. The required distribution $E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)$ need to obtain a particular radiation pattern shape will have been determined using some synthesis procedure. Thus the amplitude distribution $\left|E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)\right|$ and phase distribution $\arg\left\{E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)\right\}$ over the output surface of the transmitarray are known.

If we take the modulus of both sides of (2.2-43) we have[3]

$$\left|E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)\right| = \left|T_n\right|\left|E_y^{in}(\mathrm{x}_n, \mathrm{y}_n, -\mathrm{h})\right| \tag{2.2-44}$$

and so the required magnitude of the transmission coefficient amplitude of the n-th cell is

$$\left|T_n\right| = \frac{\left|E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)\right|}{\left|E_y^{in}(\mathrm{x}_n, \mathrm{y}_n, -\mathrm{h})\right|} \tag{2.2-44}$$

Expression (2.2-43) can also be written as

$$\left|E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)\right| e^{j\arg\left\{E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)\right\}} = \left|T_n\right| e^{j\arg\{T_n\}} \left|E_y^{in}(\mathrm{x}_n, \mathrm{y}_n, -\mathrm{h})\right| e^{j\arg\left\{E_y^{in}(\mathrm{x}_n, \mathrm{y}_n, -\mathrm{h})\right\}} \tag{2.2-45}$$

This implies that

$$\arg\left\{E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)\right\} = \arg\{T_n\} + \arg\left\{E_y^{in}(\mathrm{x}_n, \mathrm{y}_n, -\mathrm{h})\right\} \tag{2.2-46}$$

and hence that

---

[2] As a normally incident plane wave would.

[3] Symbol T_n is being used instead of T_n^x for convenience.

$$\arg\{T_n\} = \arg\left\{E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)\right\} - \arg\left\{E_y^{in}(\mathrm{x}_n, \mathrm{y}_n, -\mathrm{h})\right\} \tag{2.2-47}$$

We earlier mentioned that the output phase distribution $\arg\left\{E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)\right\}$ will have been determined using some excitation synthesis technique. We can add to it any constant phase distribution (that is, we can add the same amount of phase to each element) without changing the resultant pattern; we will denote this constant offset by the symbol Ψ_{offset}, so that the desired phase distribution over the output surface of the transmitarray is actually

$$\arg\left\{E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)\right\} \rightarrow \arg\left\{E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)\right\} + \Psi_{offset} \tag{2.2-48}$$

Thus (2.2-47) can be rewritten as

$$\arg\{T_n\} = \arg\left\{E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)\right\} - \arg\left\{E_y^{in}(\mathrm{x}_n, \mathrm{y}_n, -\mathrm{h})\right\} + \Psi_{offset} \tag{2.2-49}$$

If we next **assume** expression (2.2-30) applies for the incident field model, then

$$\arg\left\{E_y^{in}(\mathrm{x}_n, \mathrm{y}_n, -\mathrm{h})\right\} = -k_0 r_n = -\frac{k_0 F}{\cos\theta_n} \tag{2.2-50}$$

Substitution of (5-8) into 2.2-49 yields

$$\arg\{T_n\} = \arg\left\{E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)\right\} + \frac{k_0 F}{\cos\theta_n} + \Psi_{offset} \tag{2.2-51}$$

If we next **assume** we want to obtain a pencil beam then $\arg\left\{E_y^{out}(\mathrm{x}_n, \mathrm{y}_n, 0)\right\} = 0$, and (2.2-51) reads

$$\arg\{T_n\} = \frac{k_0 F}{\cos\theta_n} + \Psi_{offset} \tag{2.2-52}$$

To tie this in with derivations done for conventional lenses, we let the constant phase offset be

$$\Psi_{offset} \rightarrow \Psi_{offset} + k_0 F \tag{2.2-53}$$

so that (2.2-52) is

$$\arg\{T_n\} = -k_0 F + \frac{k_0 F}{\cos\theta_n} + \Psi_{offset} \tag{2.2-54}$$

Recognising that

$$\left\{ k_0 F - \frac{k_0 F}{\cos\theta_n} \right\} = \frac{2\pi F}{\lambda_0} \left\{ \frac{\cos\theta_n - 1}{\cos\theta_n} \right\} \tag{2.2-55}$$

allows us to algebraically reduce this (2.2-54) to

$$\arg\{T_n\} = -\frac{2\pi F}{\lambda_0} \left\{ \frac{\cos\theta_n - 1}{\cos\theta_n} \right\} + \Psi_{offset} \tag{2.2-56}$$

The complex transmission coefficient T_n will always be associated with a corresponding reflection coefficient Γ_n. We remark that we will sometimes use the symbols S_{21} and S_{11} (the phase wave scattering parameters) in place of T_n and Γ_n, respectively.

2.2.6 FAR-ZONE RADIATION PATTERN OF THE TRANSMITARRAY

A) Conceptual Model for Radiation Pattern Determination

The transmitarray output surface (that is, aperture) is shown in Figure 2.2-4. For the purposes of far-zone radiation pattern computation we will view the transmitarray as a planar array with complex excitations[4]

$$a_n = E_y^{out}(x_n, y_n, 0) = T_n E_y^{in}(x_n, y_n, -h) \tag{2.2-57}$$

[4] In what follows we will be dealing with T_n^y only, and so will simply write it as T_n.

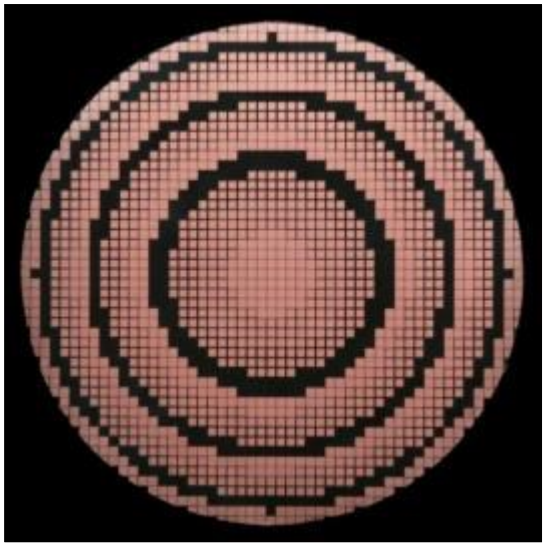


Figure 2.2-4 View of Output Surface (the Aperture) of a Transmitarray.

B) Planar Array Pattern Analysis Expressions

We are interested in the far-zone fields only, and can thus write the electric field of the antenna in the form

$$\bar{E}(r,\theta,\phi) = \frac{e^{jkr}}{r}\bar{F}(\theta,\phi) \qquad (2.2\text{-}58)$$

where r is the distance from the coordinate origin located in the vicinity of the array antenna, (θ,ϕ) is the usual spherical coordinate angle pair, and $k = 2\pi/\lambda = \omega\sqrt{\mu_0\varepsilon_0}$ is the free-space wavenumber. All quantities are in terms of the global coordinates. In previous expressions λ is the free space wavelength, $\omega = 2\pi f$ the radial frequency, and f the frequency in Hz. As in most antenna work the distance dependent term can be suppressed and we can work with the direction dependent term $\bar{F}(\theta,\phi)$ only, which will still be referred to it as the 'electric field', as is customary in antenna work. It is convenient to define the following quantities:

- A single index n will be used to number the elements of the array. The total number of elements in the array is N_e, so that $n = 1,2,\ldots..,\ N_e$.

♦ The location of the n-th element in the array, expressed in terms of the global coordinate systems, is $(x_n, y_n, 0)$.

♦ The relative complex excitation of the n-th element is

$$a_n = |a_n| e^{j\phi_n} \tag{2.2-59}$$

♦ The far-zone radiation pattern of the n-th element in the array, with $a_n = 1$, *when located at the origin of the coordinate system*, is

$$\bar{F}^{(n)}(\theta,\phi) = F_\theta^{(n)}(\theta,\phi)\hat{\theta} + F_\phi^{(n)}(\theta,\phi)\hat{\phi} \tag{2.2-60}$$

It then follows that by superposition the electric field of the entire array is [1]

$$\bar{F}(\theta,\phi) = \sum_{n=1}^{N_e} a_n \bar{F}^{(n)}(\theta,\phi) e^{jk x_n \sin\theta\cos\phi} e^{jk y_n \sin\theta\sin\phi} \tag{2.2-61}$$

If, as in most arrays, the elements are identical, then it is almost exactly true[5] that the radiation patterns of the individual elements are also identical (except for a phase factor dependent on its location in the array). We then can write $\bar{F}^{(n)}(\theta,\phi) = \bar{F}^e(\theta,\phi)$ for all *n*, and (2.2-4) becomes

$$\bar{F}(\theta,\phi) = \bar{F}^e(\theta,\phi)\sum_{n=1}^{N_e} a_n e^{jk x_n \sin\theta\cos\phi} e^{jk y_n \sin\theta\sin\phi} = \bar{F}^e(\theta,\phi) F_{AF}(\theta,\phi) \tag{2.2-62}$$

with

$$F_{AF}(\theta,\varphi) = \sum_{n=1}^{N_e} a_n e^{jk x_n \sin\theta\cos\varphi} e^{jk y_n \sin\theta\sin\varphi} \tag{2.2-63}$$

referred to as the array factor. Identical element patterns will indeed be assumed in this document.

[5] Elements near the edges of the array will in practice have radiation patterns that are slightly different from the rest even though they may be structurally identical. However, if the array is not too small, the effect of this difference between the *in situ* patterns of the elements is small. Then the radiation patterns of all elements can be assumed to be identical, except for the location-dependent phase factor in (2.2-4).

C) Model for the Normalized Feed Fields

Although we can have any feed, we adopted the model given in (2.2-30). We now go a step further and specifically assume a raised-cosine model for the feed, namely[6]

$$F_\theta^f(\theta_f,\phi_f)=\mathrm{C}_E(\theta_f)\cos\phi_f \qquad F_\phi^f(\theta_f,\phi_f)=-\mathrm{C}_H(\theta_f)\sin\phi_f \qquad \hat{x}_f\text{-Polarised On-Boresight Field}$$

$$F_\theta^f(\theta_f,\phi_f)=\mathrm{C}_E(\theta_f)\sin\phi_f \qquad F_\phi^f(\theta_f,\phi_f)=\mathrm{C}_H(\theta_f)\cos\phi_f \qquad \hat{y}_f\text{-Polarised On-Boresight Field}$$

(2.2-64)

$$\mathrm{C}_E(\theta_f)=(\cos\theta_f)^{q_E} \qquad \text{E-Plane Pattern} \tag{2.2-65}$$

$$\mathrm{C}_H(\theta_f)=(\cos\theta_f)^{q_H} \qquad H\text{-Plane Pattern} \tag{2.2-66}$$

This model is widely used in studying reflector and reflectarray antennas. It quite accurately describes realistic feed patterns in the feed main lobe region (which illuminates the transmitarray). It is of course not accurate over angular regions far from the feed main lobe. We refer to the situation shown in Figure 3.2-1.

Simple geometry allows us to write

$$\theta_{edge}=\tan^{-1}\left\{\frac{D/2}{F}\right\}=\tan^{-1}\left\{\frac{1}{2(F/D)}\right\} \tag{2.2-67}$$

and

$$r_{edge}=\sqrt{(D/2)^2+F^2} \tag{2.2-68}$$

[6] Y.Rahmat-Samii, "Reflector Antennas", Chapter 15 in J.L.Volakis (Edit.), *Antenna Engineering Handbook* (McGraw-Hill, 2007) 4th Edition. [2.2-1]

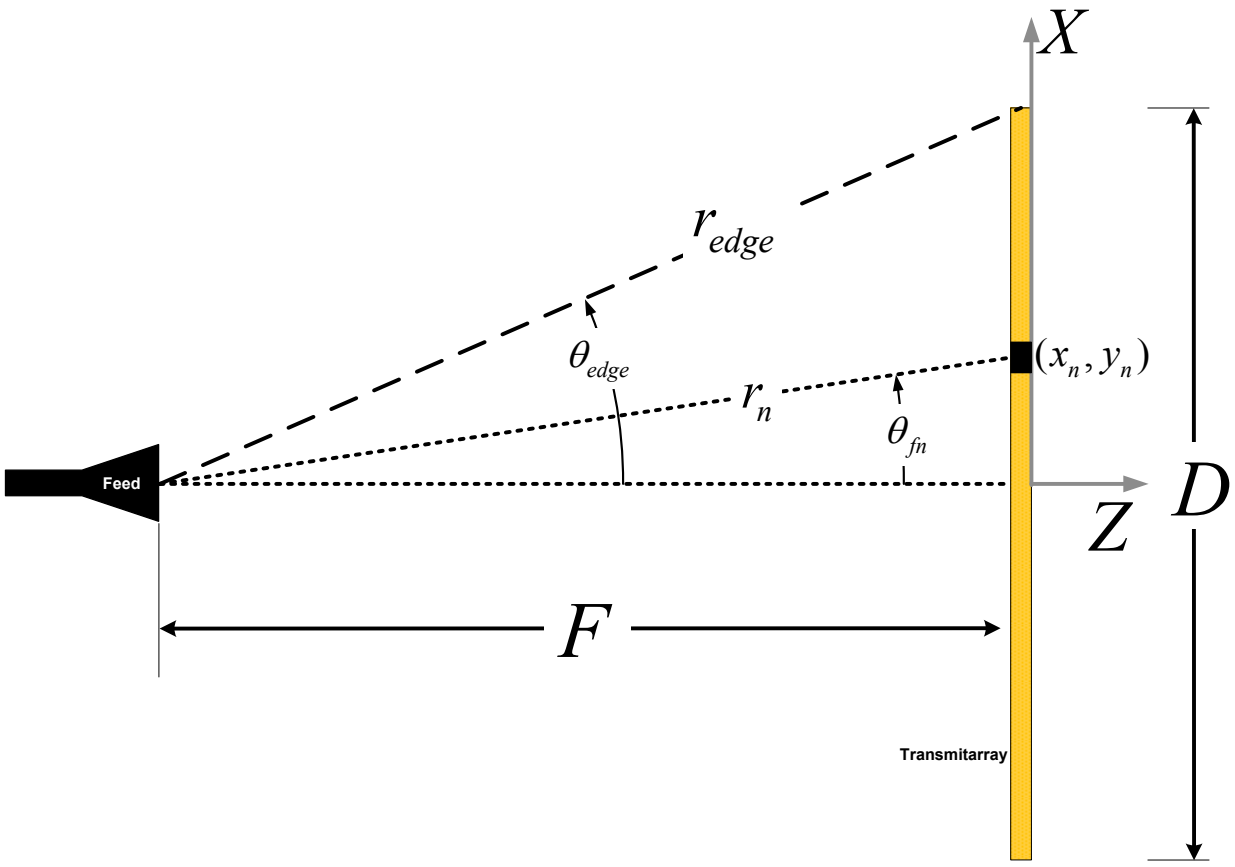


Figure 2.2-5 Relationship Between the Feed and the Transmitarray Geometry.

Using (2.2-30),

$$\left|\bar{E}^f(r_f,\theta_f,\phi_f)\right| = \frac{\sqrt{\left|F_\theta^f(\theta_f,\phi_f)\right|^2 + \left|F_\phi^f(\theta_f,\phi_f)\right|^2}}{r_f} \tag{2.2-69}$$

When the raised-cosine variation in (2.2-64) through (2.2-66) are inserted, this becomes

$$\left|\bar{E}^f(r_f,\theta_f,\phi_f)\right| = \frac{\sqrt{\left|C_E(\theta_f)\cos\phi_f\right|^2 + \left|C_H(\theta_f)\sin\phi_f\right|^2}}{r_f} \tag{2.2-70}$$

This implies that

$$\left|\bar{E}^f(r_f,\theta_f,\phi_f)\right|^{\max}=\frac{1}{F} \tag{2.2-71}$$

and so the normalized field amplitude on the input surface of the transmitarrray is

$$\frac{\left|\bar{E}^f(r_f,\theta_f,\phi_f)\right|}{\left|\bar{E}^f(r_f,\theta_f,\phi_f)\right|^{\max}}=\frac{\sqrt{\left|C_E(\theta_f)\cos\phi_f\right|^2+\left|C_H(\theta_f)\sin\phi_f\right|^2}}{F/r_f} \tag{2.2-72}$$

Thus the feed edge taper is

$$\frac{\left|\bar{E}^f(r_{edge},\theta_{edge},\phi_f)\right|}{\left|\bar{E}^f(r_{edge},\theta_{edge},\phi_f)\right|^{\max}}=\frac{\sqrt{\left|C_E(\theta_{edge})\cos\phi_f\right|^2+\left|C_H(\theta_{edge})\sin\phi_f\right|^2}}{F/r_{edge}} \tag{2.2-73}$$

If we assume that $q_E=q_H=q$, this simplifies to

$$\frac{\left|\bar{E}^f(r_{edge},\theta_{edge},\phi_f)\right|}{\left|\bar{E}^f(r_{edge},\theta_{edge},\phi_f)\right|^{\max}}=\frac{\cos^q\theta_{edge}}{F/r_{edge}} \tag{2.2-74}$$

If we wish to have a feed illumination taper[7] of C_0 dB say, this means we need to select q such that

$$C_0=-20\log\left\{\frac{F}{r_{edge}}\cos^q(\theta_{edge})\right\} \tag{2.2-75}$$

which gives

$$q=\frac{-(C_0/20)+\log(r_{edge}/F)}{\log\left\{\cos(\theta_{edge})\right\}} \tag{2.2-76}$$

[7] This means that, since the normalised feed field on axis is 0dB[0,1], that at the edge of the lens is $-C_0$ dB.

For example, if we have a printed lens with aperture dimension D = 152.4 mm with an $F/D = 0.5$. This means that F = 76.2mm. Thus we have $\theta_{edge} = 45^{\circ}$ and $r_{edge} = 107.76\,\text{mm}$. If we want $C_0 = 10\text{dB}$, then we must have q = 2.4.

The determination of feed fields on the input surface of the transmitarray could be more sophisticated than the raised cosine feed model used here. It could be extracted from the output of a detailed computational electromagnetics model of the feed, or it could be found from a near-field measurement of the feed fields back-projected to the plane with which the transmitarrray input surface eventually coincides.

2.3 DESIGN APPROACHES FOR TRANSMITARRAYS

Several approaches have been described in the literature for designing the class of antennas that can be described as aperture antennas that use printed engineered surfaces, which includes reflectarrays and transmitarrays. We will classify these, state their applicability, and then briefly describe each of them. The classification can be stated as follows:

(a). The Brute Force Approach
(b). Use of an Excitation/Pattern Synthesis, Followed by Use of a Design Database
(c). Use of a Design Database as Part of an Excitation/Pattern Synthesis Process

Option (a) has not yet been used as it is simply impractical. Option (c) has been used (we will mention this in Section 2.6) by one or more authors. Option (b) is the one most widely used with the class of antennas to which transmitarrays belong, and is of the same level of approximation (albeit sufficient for good designs) as Option (c). It is the option outlined in Section 2.2, and is the one used in this thesis.

A) The Brute Force Approach

This method would involve setting up a 3D full-wave model of the complete transmitarray plus the feed. The full-wave model must be capable of rigorously computing the far-field

radiation pattern once the dimensions of the conducting shapes in all layers of all the cells have been specified; in other words it must be able to perform a rigorous analysis. An objective function is then set up to describe the desired radiation pattern, with all the cell conducting shape dimensions as optimization variables. Some numerical optimization routine is then used to extremise the objective function. This method might possibly be used for computationally small structures (that is, those that are not electrically large) but is not viable for electrically large structures such as transmitarrays; it would require years of computations to come up with a viable prototype[8].

B) Use of an Excitation/Pattern Synthesis, Followed by Use of a Design Database

In order to determine what the transmission phase of each element must be, some sort of excitation/pattern synthesis that uses as its basis the approximate pattern analysis of Section 2.26 must be used. Once this is known the database of element characteristics can be used to determine what the dimensions of the conducting shapes must be on each of the individual cells of the transmitarray. This is the approach used in this thesis.

C) Use of a Design Database as Part of Excitation/Pattern Synthesis Process

In this approach the design database is used as an integral part of the excitation/pattern synthesis process. In other words, the cell conductor dimensions would be used as the variables in the synthesis process (that would use as its basis the approximate pattern analysis of Section 2.26), rather than the required transmission phase. Thus the synthesis procedure would access the database of element characteristics directly. However, there is not advantage with respect to accuracy or rigor compared to that in part B.

[8] The rigorous 3D full-wave analysis model will be used in Chapter to analyse transmitarrays that have already been design. Such analyses take many hours of computation. So they not viable as part of an optimisation loop in which these analyses have to be repeated at each step.

2.4 DESIGN DATABASE CONSTRUCTION AND APPLICATION

Databases of element characteristics can be constructed in two ways. In both cases each of the many points in the database, each corresponding to an element with a particular set of cell conducting shape dimensions, would be found by assuming the cell of that particular set of dimensions to lie in an infinite periodic structure of identical cells. One can attempt to derive an approximate equivalent circuit model for this situation and use it to compute database points, or perform a full-wave analysis of the particular infinite periodic structure (repeatedly of course) to compute the database points.

2.4.1 Equivalent Circuit Model for Database Generation

In this research we first attempted the approximate equivalent circuit approach, since this would have made database generation very rapid. We were interested in a 3-layered transmitarray, and so this 3 layered unit cell is assumed to be situated in a 2D periodic structure of identical cells. Now, if we consider each layer of unit cells in the periodic structure separately, it can be represented by a metallic screen. The idea is, if we can find the equivalent circuit representation of each layer, and later on cascade those circuits to form the integrated unit cell, then it would simplify the database generation stage since there wouldn't be any necessity for full wave 3D electromagnetic models of the "assembled" 3-layer structure.

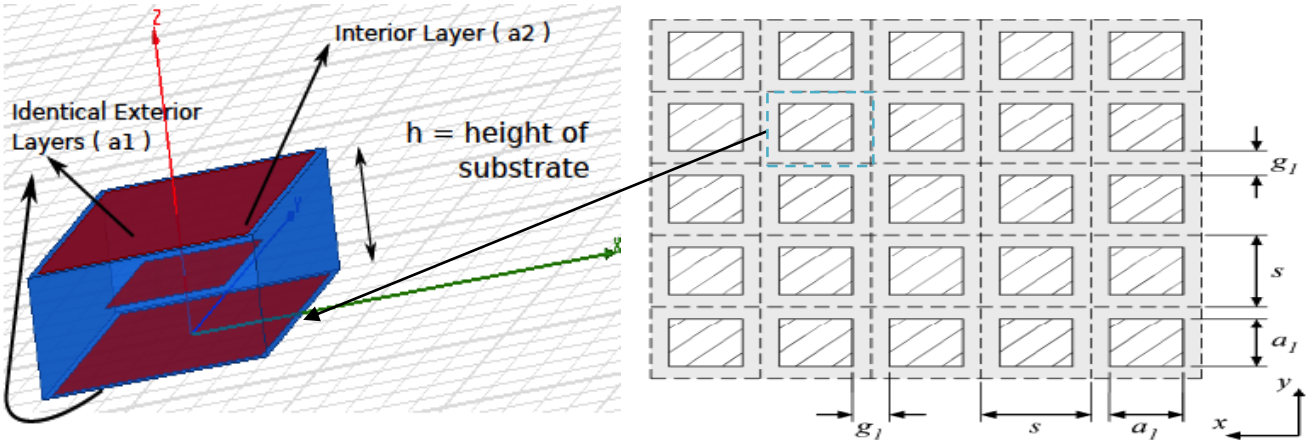


Figure 2.4-1 (a) Unit Cell (b) Unit Cell Situated in an Infinite Periodic Structure

Zarillo-Lee [2.4-1] gave approximate equivalent circuit models for planar, very thin metallic screens and quantified the lumped elements' values. Accordingly, the capacitive screen Figure 2.4-3 (a) and inductive screen in Figure 2.4-3 (b) can be characterized by an equivalent circuit for a normally incident wave with vertical polarization in the manner shown in Figure 2.4-2:

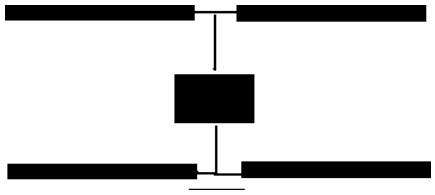


Figure 2.4-2 Transmission Line Model for Metallic Screen

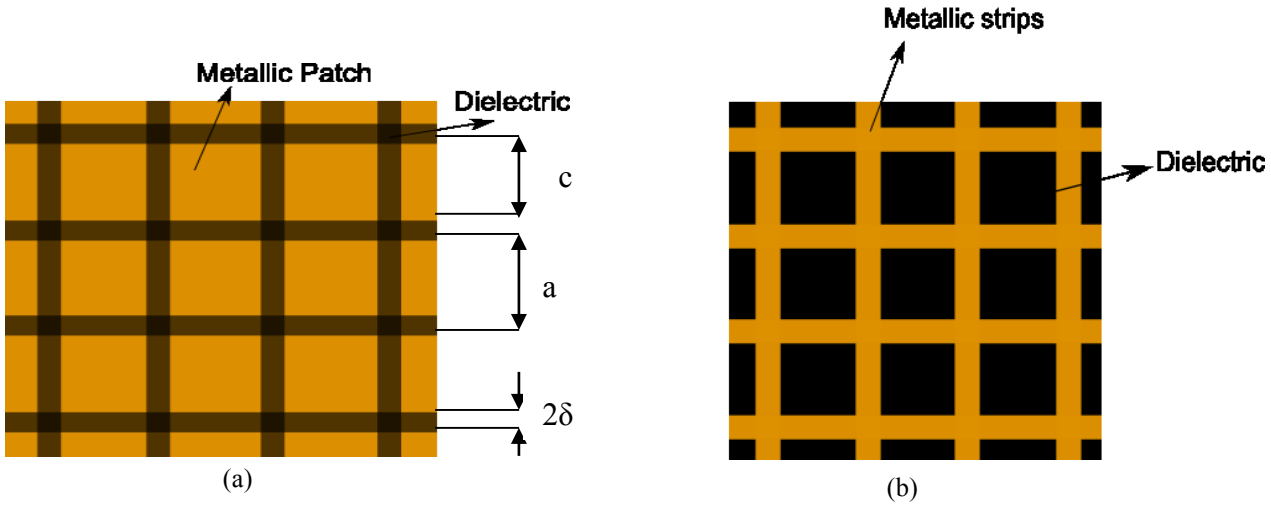


Figure 2.4-3 (a) Capacitive Screen (b) Inductive Screen

Approximate Formulas of admittance for capacitive (Y_{cap}) and inductive (Y_{ind}) screens, as derived in [2.4-1], are

$$Y_{ind} = \frac{1}{Y_{cap}} \approx (-\mathrm{j})(\beta - \beta^{-1}) \frac{\left[\left(\frac{a}{c} \right) + 0.5 \left(\frac{a}{\lambda} \right)^2 \right]}{\ln\left(\csc\left(\frac{\pi}{2} \frac{\delta}{2} \right) \right)} \tag{2.4-1}$$

where

$$\beta = \frac{\left(1 - 0.41 \frac{\delta}{a} \right)}{\left(a / \lambda \right)} \tag{2.4-2}$$

Using this equivalent circuit model for each layer and accounting for the inter-layer interaction, we conjectured the equivalent circuit shown in Figure 2.4-4

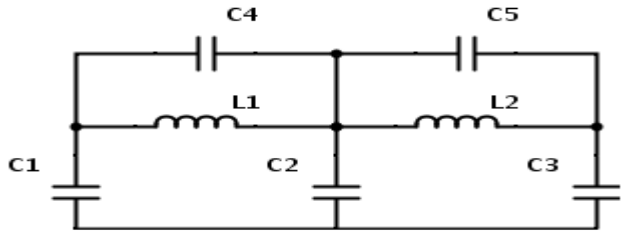


Figure 2.4-4 Equivalent Circuit for the Transmitarray Unit Cell Lying in an Infinite Periodic Structure

C1, C2 and C3 are included to account for interaction between conductive patches of cells on the same metallic sheet. C4 and C5 were included to account for evanescent field interaction between patches of different sheets. C1 and C3's value depend on 'a1', C2's value depend on 'a2', and L1 and L2 represent inductance due to currents flowing along the surface. C1, C2 and C3 are found using equations (2.4-1) and (2.4-2). Value of C4 and C5 had to be determined iteratively; optimum values for C4 and C5 were selected such that the full-wave and circuit response are approximately the same.

To check whether this model is an accurate description of a transmitarray unit cell in a periodic structure, some numerical tests were carried out. First of all, single thin metallic screens were tested and results using full-wave EM analysis and the equivalent circuit model were compared. Secondly, 2-layered structures with dielectric material sandwiched between two capacitive metallic screens were analyzed. Cell size s in Figure 2.4-1 in all the following cases is 3 mm.

Figure 2.4-5 to Figure 2.4-7 suggest that as we increase the number of layers of capacitive screens, the accuracy of the equivalent circuit model decreases. It is due to the fact that with increasing number of layers higher order evanescent modes are generated in the proximity of the screen apertures; although these modes don't propagate they introduce significant coupling between the closely-spaced sheets that is not well modelled by lumped circuit elements C4 and C5. This approach was abandoned as part of the research of this thesis.

a) 1D model: Single Capacitive Layer

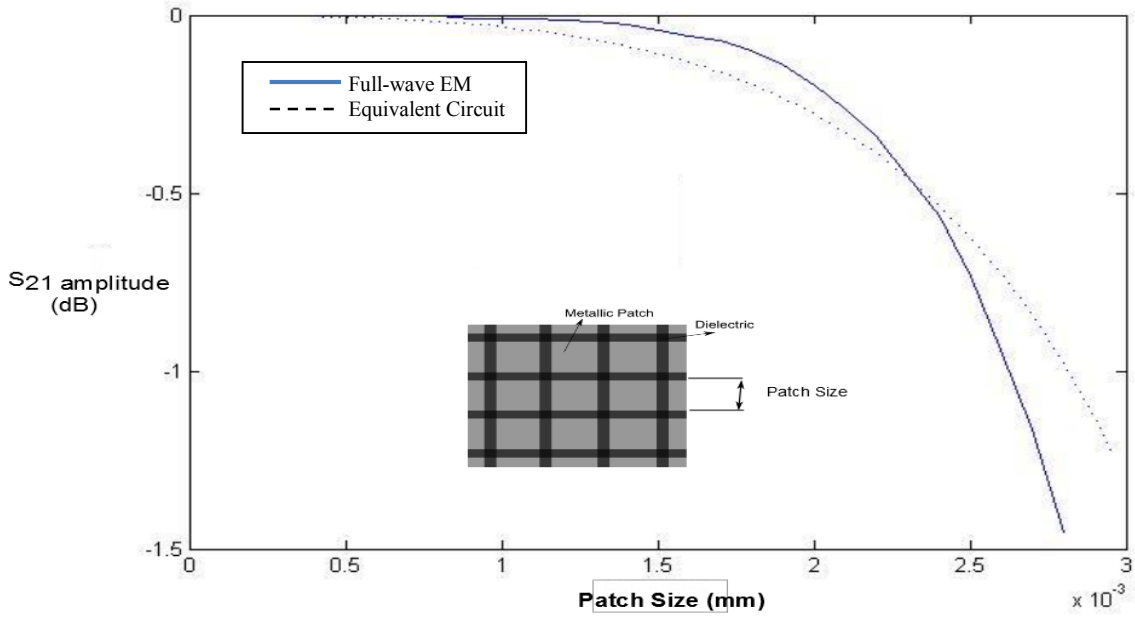


Figure 2.4-5 Transmission Amplitude vs Patch Size for a Single Capacitive Sheet

b) Two identical capacitive sheets with dielectric in the middle:

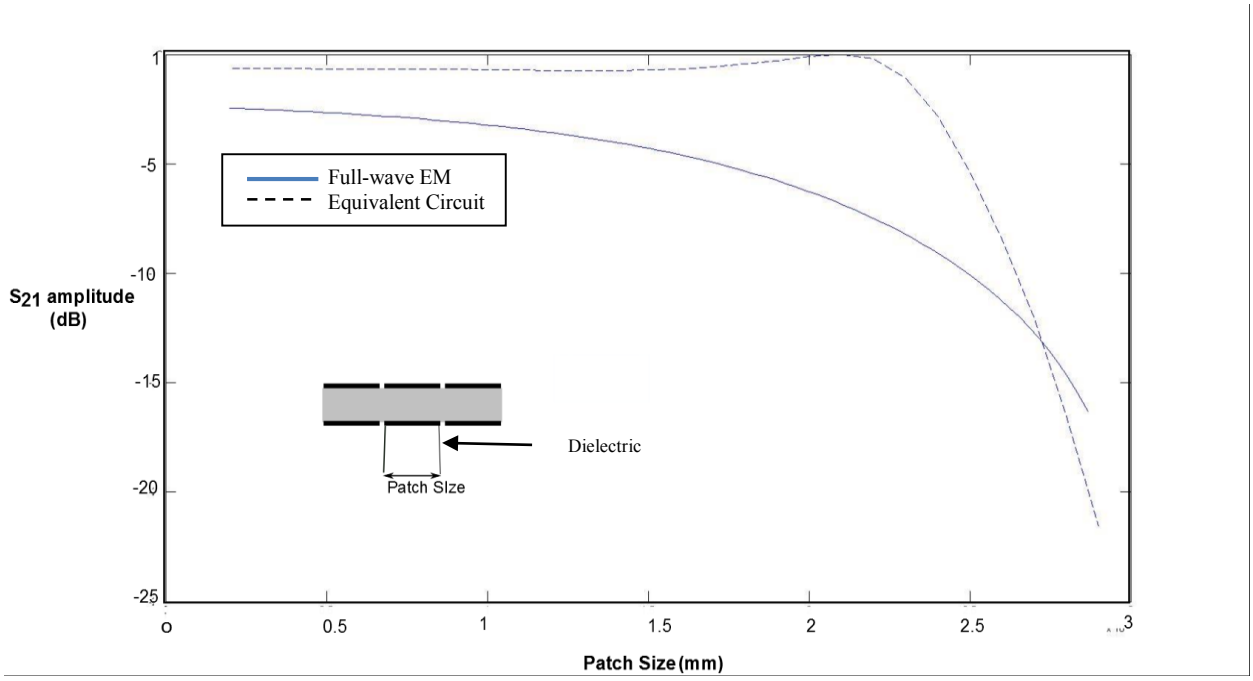


Figure 2.4-6 Transmission Amplitude vs Patch Size for Two Capacitive Sheets

c) *Three capacitive screens with two identical exterior layers and independent middle layer (analogous to the transmitarray of interest):*

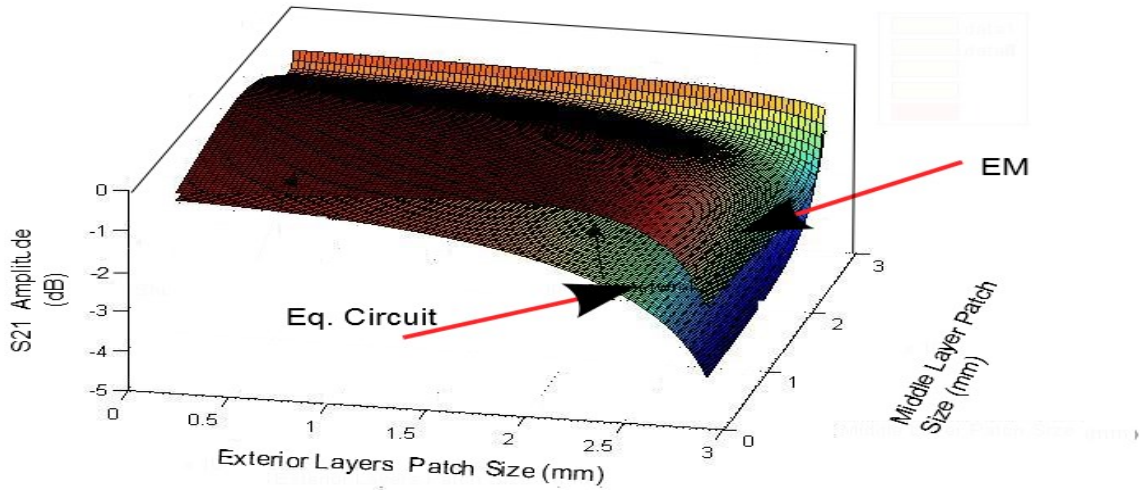


Figure 2.4-7 Transmission Amplitude vs Patch Size for 3-Layered Capacitive Sheets

2.4.2 Full-Wave Modelling Approach to Database Generation

Instead of using approximate equivalent circuit approach, which we have seen does not provide sufficient accuracy, we can set up the database using full-wave analysis that utilizes the infinite periodic structure route. This in essence reduces to a 2D computation, and so the computational burden, heavy as it is, is very much lighter than one would need have for the brute force approach mentioned in PART A of Section 2.3. Once we have created this database, we use the expression from Section 2.2.5 to determine the phase shift $\arg\{T_n\}$ required at each particular location. The final step is to "invert" the database; values of any $\arg\{T_n\}$ are specified and we wish to know the conductor dimensions (of the element) needed to realize these required $\arg\{T_n\}$ values. It is this last step that will be done using inverse neural networks in this thesis.

2.5 NEURAL NETWORK IDEAS

2.5.1 Artificial Neural Networks (ANN)

A) The Role of ANNs in Electromagnetic Engineering

Artificial neural network (ANN) modeling for analysis and design of RF integrated circuits, microwave components, and antenna design can exponentially reduce the full-wave simulation time needed. Once we have a certain amount of data from full-wave simulations, called training data (i.e. response of the system for a set of input parameters' values), an ANN model can be trained to approximate, to a very high degree of accuracy, the response of the system for input datasets differing from the training datasets. Therefore, ANNs can reduce the number of full wave model simulations needed when repetitive evaluation is required; an ANN model can be used over and over again.

Once ANNs learn device data through the training process, the trained neural networks are then use as fast and accurate models for efficient high level microwave circuit and component design. The universal approximation theorem [2.5-1] provides the foundation for ANN operation, stating that a neural network with at least one hidden layer can approximate any nonlinear continuous multidimensional function to any desired level of accuracy. Considering these advantages, ANNs have been implemented in diverse areas such as antenna applications [2.5-2, 2.5-3], microwave design applications [2.5-4, 2.5-5, 2.5-17, 2.5-18], parasitic modeling [2.5-6], non-linear microwave circuit optimization [2.5-7, 2.5-8, 2.5-16], non-linear device modeling [2.5-9, 2.5-10], and waveguide filters [2.5-11].

B) Neural Network Versus Only Full-Wave Modeling

If the ANN approach is compared with conventional modeling techniques a better understanding can be achieved. The first approach is the detailed modeling approach namely electromagnetic models. The detailed models are accurate, but are computationally expensive.

The ANN approach, however, once trained with exact full-wave input-output maps, can deliver results with very high degree of accuracy without having to perform further full-wave modelling.

C*)* Idea of NN Modeling

The design of an ANN model is synonymous to the ability of the brain to learn from observations and to generalize by abstraction. Typically a NN comprises two basic components, the processing elements and the interconnections between them. The processing elements are known as neurons and the connections between the neurons are known as links, which have associated with them a weight parameter [2.5-12]. Each neuron receives stimuli from other neurons connected to it, then processes the information, and produces an output. Neurons that receive stimuli from outside the network are termed as *input neurons*, while neurons which provide output to external components are called *output neurons*. The neurons lying in between the input and output neurons are known as hidden neurons. Input neurons, output neurons and hidden neurons can be interconnected in various topologies, leading to numerous spatial structures.

Let *n* and *m* represent the number of input and output neurons of a neural network. Let *x* be a n-element vector containing the external inputs to the neural network, *y* be a m-vector containing the outputs from the output neurons, and *w* be a vector containing all the weight parameters representing the various interconnections in the neural network. The definition of *w*, and the manner in which *y* is computed from *x* and *w*, determine the structure of the neural network.

Let inputs and outputs of a device (eg. Antenna) are *x* and *y,* respectively, where

$$x = [\, x_1, x_1, x_1 \ldots\ldots\ldots x_n \,]^T \tag{2.5-1}$$

$$y = [\, y_1, y_1, y_1 \ldots\ldots\ldots y_m \,]^T \tag{2.5-2}$$

and let the inputs be mapped to outputs by a multi-dimensional non-linear function f that represents the analytical (physical) relationship between x and y such that

$$y = f(x) \tag{2.5-3}$$

Corresponding neural network representation can be written as

$$y = f(x, w) \tag{2.5-4}$$

Now, the goal of the NN is to approximate this input-output mapping for a given set of training data, and generalize the relationship for any given inputs with minimal error. After learning the original input-output relationship through a process called training, the NN in equation (2.5-4) can represent the physical behavior in (2.5-3). Multiple (x - y) samples, called training data, need to be generated from the full-wave simulator[9] in the desired range of operation of the device. The objective of training is to adjust neural-network weights w such that the neural model outputs best match the training data outputs.

D) Multi-Layered Perceptron (MLP)

1) Topology: MLP is a widely used neuron topology [2.5-21]. In the MLP neural network, the neurons are grouped into layers. Typically, an MLP consists of an input layer, one or more hidden layers, and an output layer, as shown in Figure 2.5.1. For example, an MLP neural network with an input layer, one hidden layer, and an output layer, is referred to as a three-layer MLP (or MLP3).

Let's assume the total number of layers are L. The first layer is the input layer, the L^{th} layer is the output layer, and layers 2 to *L-1* are hidden layers. If the number of neurons in the l^{th} layer are N_l such that $l = 1,2,3.....,L$. Let w^l_{ij} represent the weight of the link between the jth neuron of the l-*1* th layer and the ith neuron of the lth layer. Let x_i represent the ith external input to the MLP and z_i^l be the output of the i^{th} neuron of the l^{th} layer. There is an additional weight parameter for each neuron (w^l_{i0}) representing the bias for the i^{th} neuron of the l^{th} layer. In other words, $w = [w^2_{10}\ \ w^2_{11}\ \ w^2_{12}\ \ \ldots\ldots\ldots\ w^L_{NL\,NL\text{-}1}]^T$. The parameters in this weight vector are real numbers, which are initialized before MLP training. Using various optimization algorithms the weights are updated during the training process.

[9] In some applications this training data could be obtained from measurements, but this is not relevant to the type of problem of interest in this thesis.

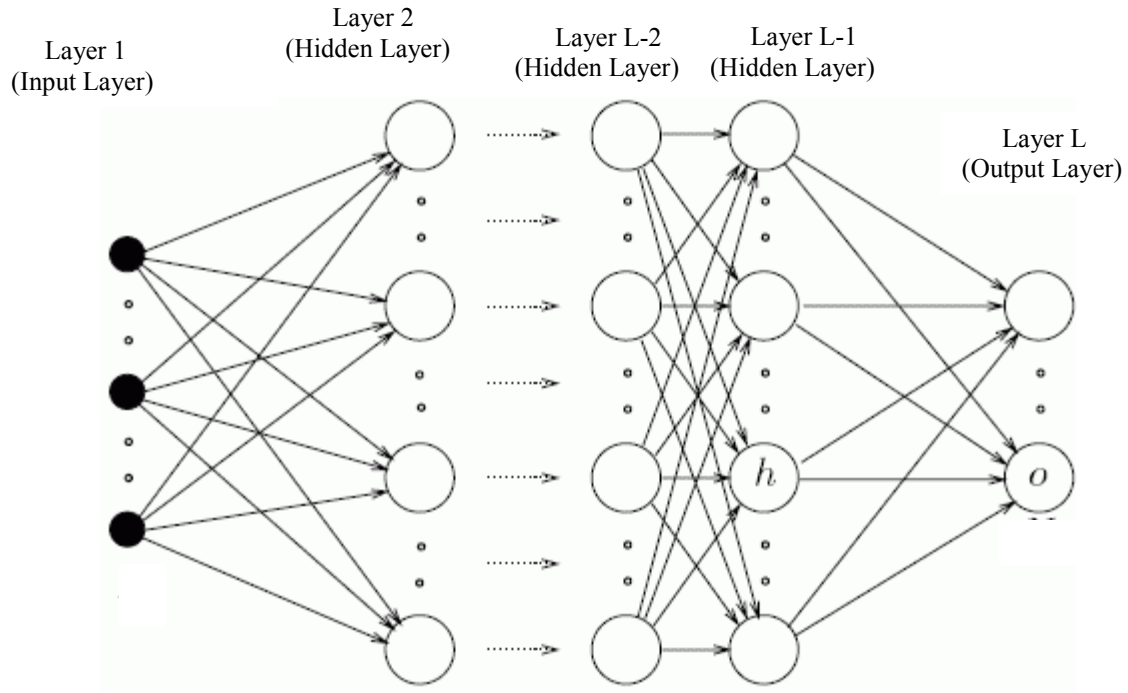


Figure 2.5-1 General MLP with L Layered Topology

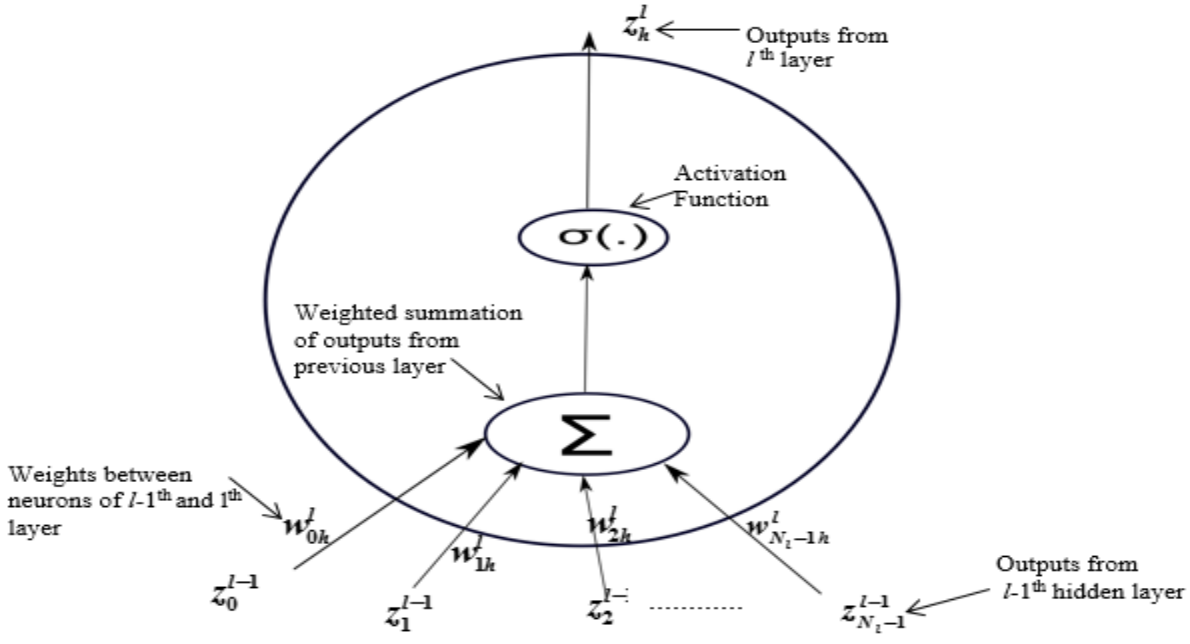


Figure 2.5-2 Neuron- Processing Unit of ANN for the l^{th} Layer [2.5-21]

After the termination of the training process weight vector is finalized and it is this final version of the weight vector which is utilized for the further usage of neural network model.

2) *Composition of the Neurons:* In the MLP neural network, each neuron is connected to other neurons and is fed by the information received from the connected neurons and finally processes and relays this information to the neurons in the subsequent layer. Activation function is the function inherent to all the neurons, and it is through this function that all the information is processed and the processed information becomes the output of the respective neurons. Next we consider a neuron in the l^{th} layer and this neuron receives stimuli from the neurons of the $(l-1)^{th}$ layer, i.e., $z_1^{l-1}, z_2^{l-1}, z_3^{l-1} \ldots\ldots\ldots, z_{N_{l-1}}^{l-1}$ as shown in Figure 2.5-2. The i^{th} neuron in the l^{th} layer processes this information in two steps. Initially, each of the inputs is multiplied by the corresponding weight parameter and the products are added to produce a weighted sum γ_i^l namely [2.5-12]

$$\gamma_i^l = \sum_{j=0}^{N_{l-1}} w_{ij}^l z_j^{l-1} \tag{2.5-5}$$

The weighted sum in equation (2.5-5) is in turn used to activate the neuron's activation function $\sigma(.)$ to produce the final output of the neuron $z_i^l = \sigma(\gamma_i^l)$. There are several functions that can be used as the function for a neuron, but in the literature, most commonly used hidden neuron activation function is the sigmoid function given by

$$\sigma(\gamma) = \frac{1}{(1+e^{-\gamma})} \tag{2.5-6}$$

The ability of MLP to handle and model the non-linear relationship between inputs and outputs makes it such a powerful approximation tool. But it becomes possible only if we choose an appropriate activation function. For the back-propagation learning process the activation function must be differentiable. For the output neurons, activation functions should be such that they are suited to the outputs in the training/test data. If the neural network training/ test data outputs are continuous (non-discrete) and are bounded, then sigmoid functions are very useful.

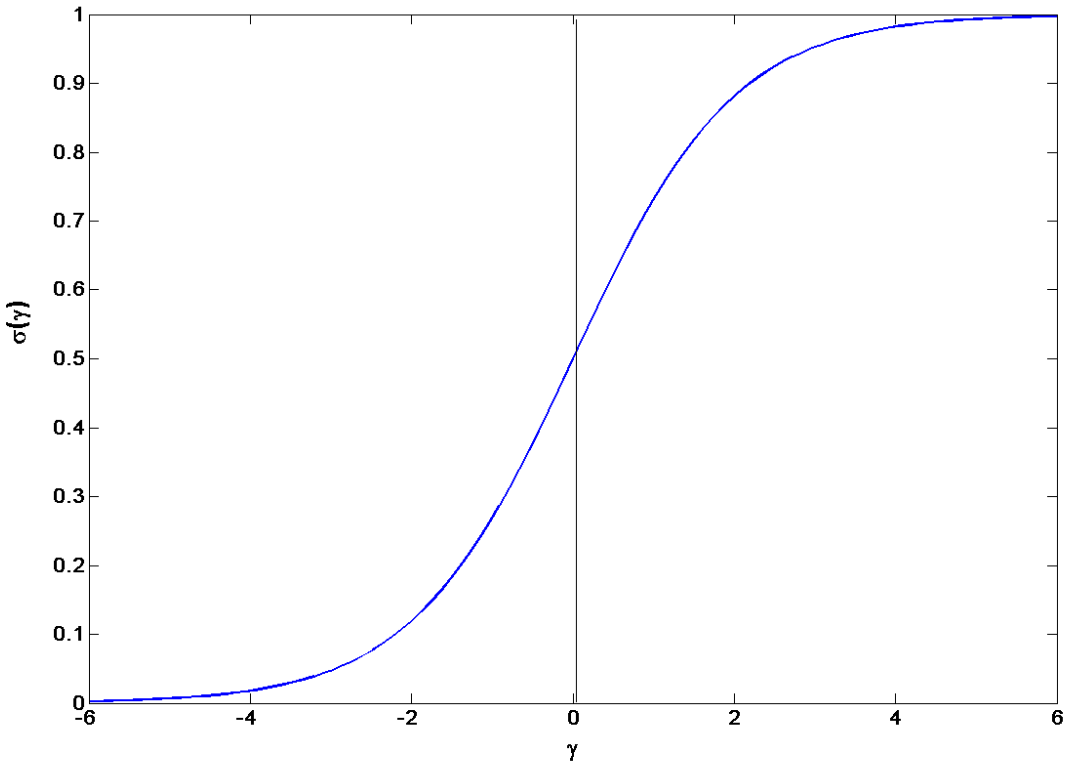


Figure 2.5-3 Sigmoid Function

Input neurons simply relay the external stimuli to the hidden layer neurons using a linear relay activation function , that is $z_i^1 = x_i,\ i=1,2,\ldots,n$. The x_i are those in equation (2.5-1). In the case of neural networks for electromagnetic design, where the purpose is to model continuous electromagnetic parameters, a linear activation function can be used for output neurons. At the output neuron, the computation can be expressed as

$$\sigma(\gamma) = \gamma \quad \text{(Linear Activation Function)}$$
$$\sigma(\gamma_i^L) = \gamma_i^L = \sum_{j=0}^{N_{L-1}} w_{ij}^L z_j^{L-1} \tag{2.5-7}$$

3) Feedforward Computation: Feedforward computation is a process through which computation of the output vector $y = [y_1, y_2, y_4 \ldots\ldots y_m]^T$ is done for given inputs of the form $x = [x_1, x_2, x_3 \ldots\ldots x_n]^T$ and the weight vector w. The feedforward computation is useful not only during neural-network training, but also during the usage of the trained neural model. The external inputs are first fed to the neurons in the input layer, that is Layer 1, and the outputs from

the input neurons are fed to the hidden neurons of the second layer, that is Layer 2. We proceed in this way and finally the outputs of the L-1th layer neurons are fed to the output layer neurons, that is Layer L. For a given epoch (iteration) of feedforward computations weight vector *w* remains constant. The computation of the output is done by propagating the input through all the layers and applying the activation function and summation for each neuron and this computation is detailed below.

For the Layer 1, the output is computed using the relay function described in equation (2.5-7) and is done in the following manner

$$z_i^1 = x_i \ , \qquad \text{i=1,2,….,N}_1 \tag{2.5-8}$$

where N_1 = number of neurons in the input layer or Layer 1, and this output is fed to Layer 2 and computation of output in Layer is given by

$$z_i^2 = \sigma(\sum_{j=0}^{N_{l-1}} w_{ij}^2 z_j^1) \quad \text{i=1,2,….,N}_2 \text{ and } \text{j = 1,2……N}_1 \tag{2.5-9}$$

Where N_2 = number of neurons in Layer 2, and this output is fed to Layer 3.

This process is repeated for all the hidden layers until and for the l^{th} hidden layer computation of output is given by

$$z_i^l = \sigma(\sum_{j=0}^{N_{l-1}} w_{ij}^l z_j^{l-1}) \ , \text{i=1,2,……,N}_{l-1} \qquad l = 3,4\ldots,L \tag{2.5-10}$$

where N_{l-1} = number of neurons in the Layer l-1

In the final step, outputs are extracted from the output neurons in layer L as follows:

$$y_i = z_i^L, \qquad \text{i=1,2,…….,N}_L \ \& \ \text{m=N}_L \tag{2.5-11}$$

Equations (2.5-7) – (2.5-11) can approximate any nonlinear relationship (x-y) with very high degree of accuracy.

E) Training of Neural Network

One of the most critical steps in NN model development is training. The NN needs to be trained with RF/microwave component's input-output map. The data, called *training data*, are pairs of model input-output (i.e., *x* - *y*) data generated either from detailed microwave simulation or measurement. Let d_k be the k^{th} sample of *y* in the training data. We define the neural network training error as

$$E(w) = \frac{1}{2} \sum_{k \in G_{tr}} \sum_{j=1}^{m} | y_j(x_k, w) - d_{jk} |^2 \qquad (2.5\text{-}12)$$

where d_{jk} is the j^{th} element of the data vector d_k, y_j (x_k, w) is the j^{th} neural network output for the input $\boldsymbol{x}_k$, G_{tr} is the index set of all training data, and *w* is the vector of neural network weight parameters.

Fundamentally the purpose of neural network training is to adjust 'w' such that the error function E(w) is minimized. Since E(w) is a non-linear function of the modifiable weight parameters w, optimization algorithms are used to optimize *w* beginning with an initial guess and then iterative updates.

During the training process, the error function is computed between the output from the model and measured/simulated output. This error function is then propagated through the various layers (starting from the output through the hidden layer to the input layer). In each layer the Jacobian of the error function w.r.t. the weight vector is computed in each layer. Once this information is obtained, it can either be used directly to update the value of each weight parameter in the neural network the back-propagation training algorithm [2.5-13], or used as gradient information for gradient based algorithms such as conjugate gradient, Levenberg-Marquardt or quasi-Newton [2.5-14] methods. However, the quasi-Newton, Levenberg-Marquardt and conjugate gradient methods are now the best choices for a majority of neural network training problems for microwave modeling, because they produce models with lower training error and at significantly faster speed of training than the back- propagation method. Sometimes we can encounter challenges to evade local minima in the optimization loop, which can prevent effective training. In that case global optimization algorithms, such as genetic algorithms and particle swarm

optimization [2.5-15] can be utilized, but these routines suffer from elongated training times. Detailed training procedure utilized in this work can be found in [2.5-12]

F) Neural Network Microwave Modeling Procedure

The idea of neural network modeling is to use the simple formulas described by (2.5.1)–(2.5.3) to approximate multidimensional nonlinear relations between x and y. By simply adding input neurons, a neural network can deal with multidimensional input variables (x) much more easily than polynomial, rational, or table models for interpolation. By adding more hidden neurons, a neural network can handle arbitrary nonlinearity between x and y more efficiently than other approximation approaches.

We can summarize the steps to generate a ANN model for a given MW/ RF problem as follows:

Step1 ➔ Define the input and output variables of a device or a structure.
Step2 ➔ Generate IO data using EM simulation, physics-based simulation, or measurement.

Step3 ➔ The generated data, i.e., training data, are used to train the neural network.

Step4 ➔ Once the model is trained, it can be incorporated into a circuit simulator for fast and accurate simulation and optimization.

G) Self-Organizing Maps (SOM)

In some instances the training of the NN as outlined in part E Section 2.5.1 is insufficient to provide the accuracy required. The self-organizing map (SOM) approach can then be used [2.5-22, 3.3-1]. In the neural network modelling, if the number of samples is huge and highly nonlinear, the training process becomes difficult and it has been tested that if we divide the neural network sample space into sub-spaces and model these sub-spaces individually and later connect them, the we can achieve satisfactory training of neural networks. In Self-Organizing Map (SOM), the large input space is divided into smaller sub-spaces and each sub-space is modelled using MLP neural network. SOM is a clustering algorithm that facilitates automatic

decomposition through processing and learning of training data (explained in Section 2.5.1 (E)). A self-organizing map consists of components called nodes or neurons. Associated with each node is a weight vector of the same dimension as the input data vectors, and a position in the map space. The usual arrangement of nodes is a two-dimensional regular spacing in a hexagonal or rectangular grid. The self-organizing map describes a mapping from a higher-dimensional input space to a lower-dimensional map space. The procedure for placing a vector from data space onto the map is to find the node with the closest (smallest distance metric) weight vector to the data space vector. The total number of cluster centers to be determined from training data is equal to the total number of neurons in SOM. In the sample distribution each neuron represents a cluster center. Further details of the SOM approach, and reasons why it is necessary, will be provided when it is implemented for the transmitarray database in Section 3.3.1.

2.5.3 Inverse Neural Networks

A) Introduction to Inverse Modelling approach

A typical ANN as discussed in previous Section has input to output mapping where geometrical / physical parameters (guide length, substrate thickness, characteristic impedence etc.) form the input data set and output data set, electrical parameters or characteristic response, is sought. A forward Neural Network model is obtained this way by adjusting inter-neuron weights iteratively. But the design problems encountered in RF circuit and EM design are typically, if not exclusively, 'inverse'. 'Inverse' stands for the mapping from electrical parameters (as inputs) to geometrical parameters (as outputs) – although there can be a combination of geometrical and electrical parameters as inputs and outputs as well. This inverse problem can be solved either by implementing optimization routines or EM simulator/ forward model can be evaluated repeatedly (which can be an onerous task) to find the optimal solution. Details of such process are given in [2.5-19].

The fundamental premise for the inverse problem which is to find geometrical parameters from a given a set electrical parameters, is an arduous task if we go for analytical approach. Therefore, neural

network becomes a rational choice in such matters. In this approach geometrical parameters become inputs and electrical parameters become outputs. Now, in order to obtain training data, inputs and outputs are swapped. This method is termed as 'Direct Inverse Modelling' and it can provide solutions instantaneously once trained.

To elucidate, let's consider design of a generic microwave (MW) device with physical dimensions (a_1,a_2), substrate height h and let the corresponding response in terms of electrical parameters be S_{11} and S_{21}. Therefore a forward ANN model and corresponding (one of the possible) inverse NN model that can approximate this device would have the topology shown in Figure 2.5-4.

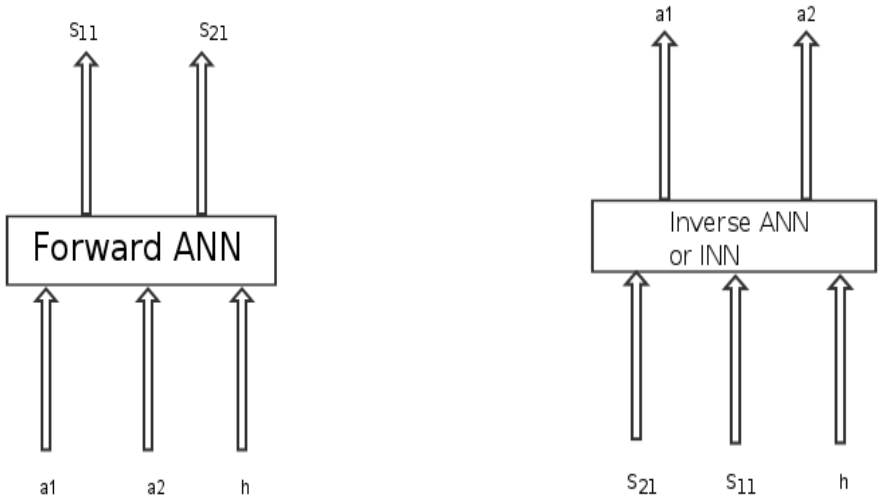
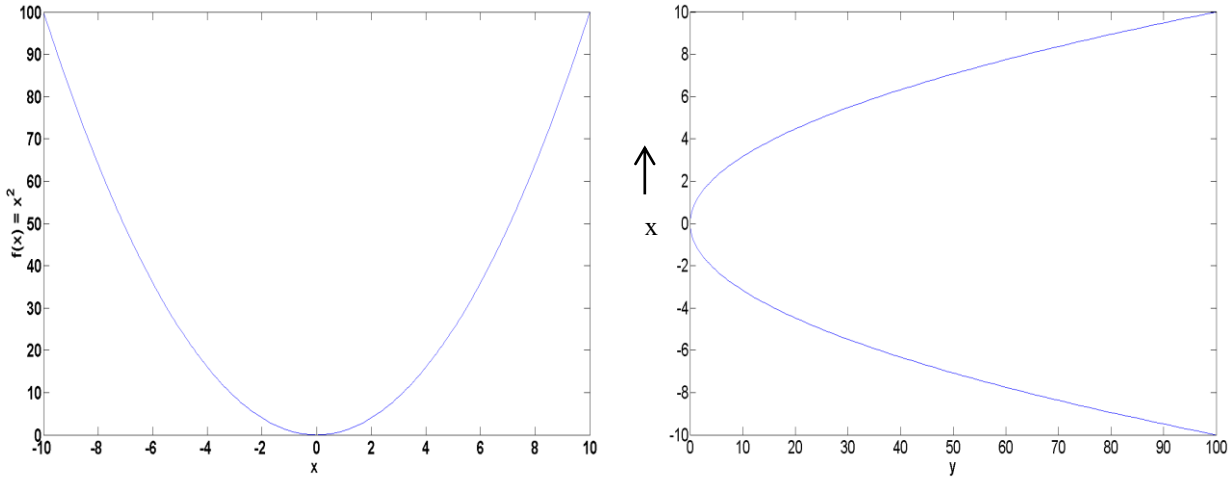


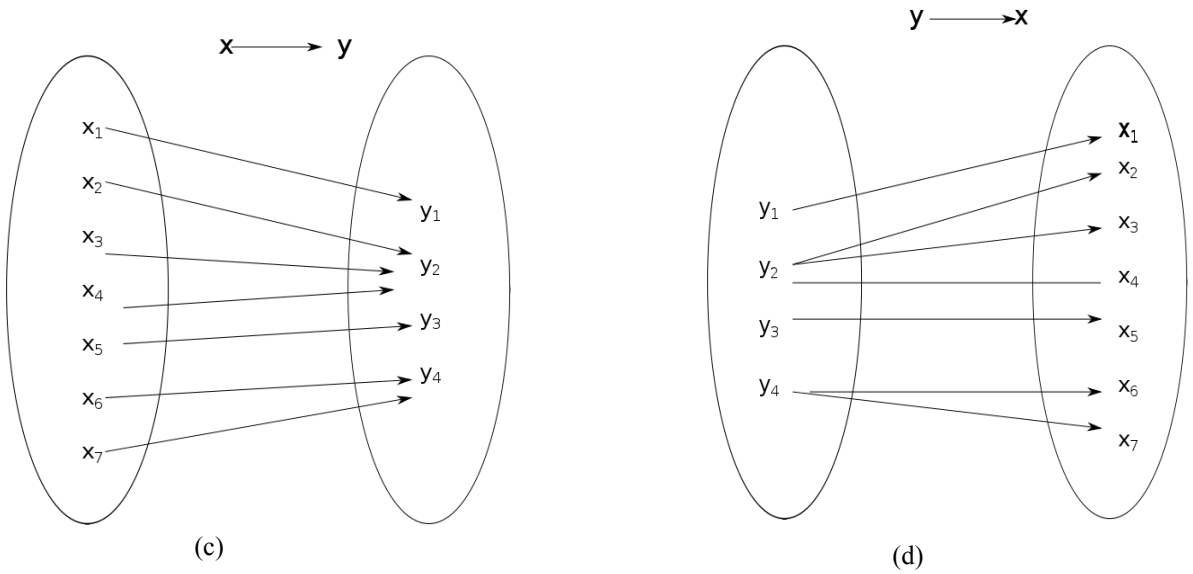
Figure 2.5-4 Forward Neural Network and Alongside one of the possible inverse Neural Network Configuration

In the inverse model we can see that some of the outputs and inputs are flipped but h (substrate height) is still an input parameter. But such a model (let us call it the direct inverse model) often faces a stumbling block in terms of non-uniqueness of the x➔y mapping (input-output relationship) or multivalued solutions. This means that multiple inputs of the forward model can have the same outputs; this means that one input value of the inverse model can produce multiple outputs. As a consequence, training an inverse model can become challenging. This problem is illustrated in Figure 2.5-5 for the simple scalar input-output map of $f(x) = x^2$.

Figure 2.5.5b



(a) (b)



(c) (d)

Figure 2.5-5 (a) , (c) Represent Forward model $y=x^2$ with many-to-one mapping. (b), (d) Represent Inverse model $x=\pm\left|\sqrt{y}\right|$ with non-unique multivalued solutions

Figure 2.5-5(a) shows plot of $y=x^2$; as it can be seen for two different values of x the value of function can be the same, that is this function is many-to-one. Figure 2.5-5(b) and 2.5-5(d) show the plot of the corresponding inverse function $x=\pm\left|\sqrt{y}\right|$ and its map, respectively. As we can

see a single input for the inverse model has multiple possible outputs. To overcome this predicament a novel methodology was developed in [2.5-20].

The proposed technique initially finds out whether there is non-unique input-output map. If so, then so-called adjoint neural network which is the derivative information of the neural network outputs w.r.t. inputs. Using these derivative criteria, training data is partitioned various groups based upon the slope. Each group of data is used to train a separate inverse model, which is called the 'inverse sub-model'. Since a sub-model individually doesn't suffer from multivalued solutions, the modeling good accuracy.

B) Inverse Modeling Formulation

Let the forward modeling problem be represented as in equation (2.5-3), namely

$$y = f(x) \tag{2.5-13}$$

where x is a *n-element input vector* and y is a *m-element output vector.* Therefore, $x = [x_1\ x_2 \ldots x_n]^T$, $y = [y_1\ y_2 \ldots\ldots\ y_m]^T$ and f represents the input-output relationship. Figure 2.5.4 shows a 3-input forward model and its corresponding inverse model with some inputs and outputs swapped. Now lets consider a scenario such that few of the inputs and outputs are swapped and say a subset of x and y is swapped thus forming an inverse model. Considering I_x to be an index set containing the indices of inputs of the forward model that are moved to the output of the inverse model

$$I_x = \{i \mid if\ x_i\ is\ swapped\ to\ become\ output\ of\ inverse\ model\} \tag{2.5-14}$$

Similarly, lets define another index set I_y containing indices of outputs of the forward model that are swapped and have now become inputs of the inverse model

$$I_y = \{i \mid if\ y_i\ is\ swapped\ to\ become\ input\ of\ inverse\ model\} \tag{2.5-15}$$

Also, let $\bar{x}$ and $\bar{y}$ be vectors of inputs and outputs of the inverse model. They differ from x and y in that some elements have been swapped. Therefore inverse model can be defined as

$$\bar{y} = \bar{f}(\bar{x}) \tag{2.5-16}$$

where $\bar{f}$ represents input-output relationship of the inverse model, $\bar{y}$ contains y_i if i$\in I_y$ and x_i if i $\square I_x$; $\bar{x}$ includes x_i *if* $i \notin I_x$ *and* y_i if $i \square I_y$. For instance lets again consider forward and corresponding inverse model for a MW device as represented in Figure 2.5.4, where we can write for :

Forward model - $x_1 = a1, x_2 = a2, x_3 = h$; also $y_1 = S_{11}$, $y_2 = S_{21}$ and

and,

Inverse model - $\bar{x}_1 = S_{11}$, $\bar{x}_2 = S_{21}$, $\bar{x}_3 = h$; *and* $\bar{y}_1 = a1$, $\bar{y}_2 = a2$

Thus,

$$I_x = \{1, 2\} \tag{2.5-17}$$

$$Iy = \{1, 2\} \tag{2.5-18}$$

$$\bar{x} = [\bar{x}_1, \bar{x}_2, \bar{x}_3]^T \tag{2.5-19}$$

$$= [y_1, y_2, x_3]^T \tag{2.5-19}$$

$$\bar{y} = [\bar{y}_1, \bar{y}_2]^T \tag{2.5-21}$$

$$= [x_1, x_2]^T \tag{2.5-22}$$

Once the inverse model topology is developed according to the methodology, next process is to train the inverse model. Ideally training data is generated in a similar way as for the forward model using EM/ Circuit Solver and finally inputs and outputs are swapped in accordance with the desired model. This neural model thus generated would be a direct inverse model which, as explained in Section *A,* is suitable when IO relationship is one to one or monotonous. Whereas if that's not the case strategies a explained in next Section are utilized.

C) To Check Whether Non-Uniqueness of IO Relationship Exists

As discussed in previous Section if we encounter forward IO map to be non-monotonic then non-uniqueness of inverse IO map causes a problem. To overcome that first of all we will need

to be sure whether non-unique input-output relationship exists for the inverse model under consideration. Now, if two different input values in the forward model lead to same output, then certainly there will be a contradiction in the inverse model as one input will have multiple value outputs. In that case training of the inverse model becomes problematic and it will lead to high training error which is undesirable. Therefore, detection of non-monotonic nature of forward input-output relationship becomes indispensable, and is determined next.

The procedure to accomplish this is proposed in [2.5-20] and we will describe it here. To determine whether the input-output map is non-monotonic, let's consider I_x and I_y index sets, as defined by equations (2.5-17) and (2.5-18), respectively. Both these index sets have equal number of elements and indices (i.e. elements) in I_x and I_y are in increasing order. Now, we define distance between two samples of training data, sample number *l* and *k,* as

$$d^{(k,l)} = \sqrt{\sum_{i=1}^{n} (\bar{x}_i^{(k)} - \bar{x}_i^{(l)})^2 / (\bar{x}_i^{\max} - \bar{x}_i^{\min})^2} \qquad (2.5\text{-}23)$$

(superscripts l, k depict the sample number in the training data wherein each sample contains input vector and corresponding output vector)

where $d^{(k,l)}$ symbolizes the normalized Euclidean distance between two samples namely sample number k and l ; $\bar{x}_i^{\max}$ and $\bar{x}_i^{\min}$ are the maximum and minimum value of $\bar{x}_i$ which is computed from the training data ; $\bar{x}_i^{(k)}$ and $\bar{y}_i^{(k)}$ represent values of $\bar{x}_i$ and $\bar{y}_i$ in the kth sample of training data, respectively. Now, we define neighborhood of sample $\bar{x}^{(l)}$ using a user defined threshold, ε (its value depends on step size of sample) , such that sample $\bar{x}^{(k)}$ lies in its neighborhood if $d^{(k,l)} < \varepsilon$. We then define maximum and minimum 'slope' for samples that reside within the neighborhood of $\bar{x}^{(l)}$ as

Given that a sample lies within a neighborhood of $\bar{x}^{(l)}$, we define maximum and minimum slope as

$$\psi_{\max}^{(l)} = \left[d^{(k,l)} \underset{k}{\overset{\max}{<}} \varepsilon \frac{\sum_{i=1}^{m} [(\bar{y}_i^{(k)} - \bar{y}_i^{(l)}) / (\bar{y}_i^{(\max)} - \bar{y}_i^{(\min)})]^2}{\sum_{i=1}^{n} [(\bar{x}_i^{(k)} - \bar{x}_i^{(l)}) / (\bar{x}_i^{(\max)} - \bar{x}_i^{(\min)})]^2} \right]^{\frac{1}{2}} \tag{2.5-24}$$

$$\psi_{\min}^{(l)} = \left[d^{(k,l)} \underset{k}{\overset{\min}{<}} \varepsilon \frac{\sum_{i=1}^{m} [(\bar{y}_i^{(k)} - \bar{y}_i^{(l)}) / (\bar{y}_i^{(\max)} - \bar{y}_i^{(\min)})]^2}{\sum_{i=1}^{n} [(\bar{x}_i^{(k)} - \bar{x}_i^{(l)}) / (\bar{x}_i^{(\max)} - \bar{x}_i^{(\min)})]^2} \right]^{\frac{1}{2}} \tag{2.5-25}$$

Once these slopes are computed in the neighborhood for a given sample, we check whether multivalued solutions exist in the neighborhood for the sample under consideration ($\bar{x}^{(l)}$). If in the neighborhood of $\bar{x}^{(l)}$, the slope is larger than the maximum allowed slope or ratio of maximum to minimum slope is larger - than the maximum allowed slope change, we can ascertain that there exist multivalued solutions for the current sample. This criteria for detection can be expressed as:

If,
$$\psi_{\max}^{(l)} > \psi_M \tag{2.5-26}$$

and

$$\psi_{\max}^{(l)} / \psi_{\min}^{(l)} > \psi_R \tag{2.5-27}$$

then multivalued solutions exist. Quantities ψ_M and ψ_R are user-selected values and the selection criteria is discussed next.

Now the question arises how to decide the neighborhood (depicted by ε) for a given sample. It has been observed that if there are K samples in training data and *N* number of outputs then,

$$\varepsilon > \frac{2 \times \sum_{i=1}^{K-1} \delta_i}{K-1} \tag{2.5-28}$$

where

$$\delta_i = \sqrt{\sum_{j=1}^{N} (y_j^{(i+1)} - y_j^{(i)})^2} \qquad i = 1,2,\ldots\ldots\text{K-1} \tag{2.5-29}$$

Equation (2.5-28) and (2.5-29) state that ε must be greater than twice the average training data step size for y.

ψ_R can be approximated by computing slope for the forward model, using same principles as equation (2.5-24) and (2.5-25), and taking the inverse of it. ψ_M can be application dependent but usually its value can be around 1.

D) Method to Divide Inverse Model Training Data if Multivalued Solutions Exist

Inverse model training data need to be preprocessed once its been ascertained that multivalued solutions exist. Then the next objective is to segment the training data into various sets such that each set doesn't suffer from non-uniqueness of solutions problem. So, now the challenge remains, what criteria must we utilize in order to partition the data?

As an illustrative example, we look at the division of data for a one-dimensional (1 input and 1 output) neural network model. Let's again consider the input-output relationship depicted in Figure (2.5-5), i.e.

$$y' = f(x') = (x')^2 \tag{2.5-30}$$

Note that neural network model's output representation y is written as y' for this specific example to avoid confusion with the generalized case. Now, let's examine the forward model whose analytical form is given by equation (2.5-30). It can be observed that; if we segment the input domain into two segments such that for one segment the slope is negative while for the other slope is positive, the two segments (groups of input) thus generated, have corresponding inverse sub-models with unique solutions. This way we can evade the complication of multivalued solutions. In mathematical terms:

If for the k^{th} training sample

$$\left(\frac{dy'}{dx'}\right)_k < 0 \qquad (2.5\text{-}31)$$

then

$$[y'_k, x'_k] \in H_1 \qquad (2.5\text{-}32)$$

otherwise

$$[y'_k, x'_k] \in H_2 \qquad (2.5\text{-}33)$$

where k = 1,2,….K(number of samples) , H_1 and H_2 are the training data sets for inverse sub-model number 1 and number 2.

In this case, ψ_R is not needed since we have graph for the example and ψ_M is taken to be 1.

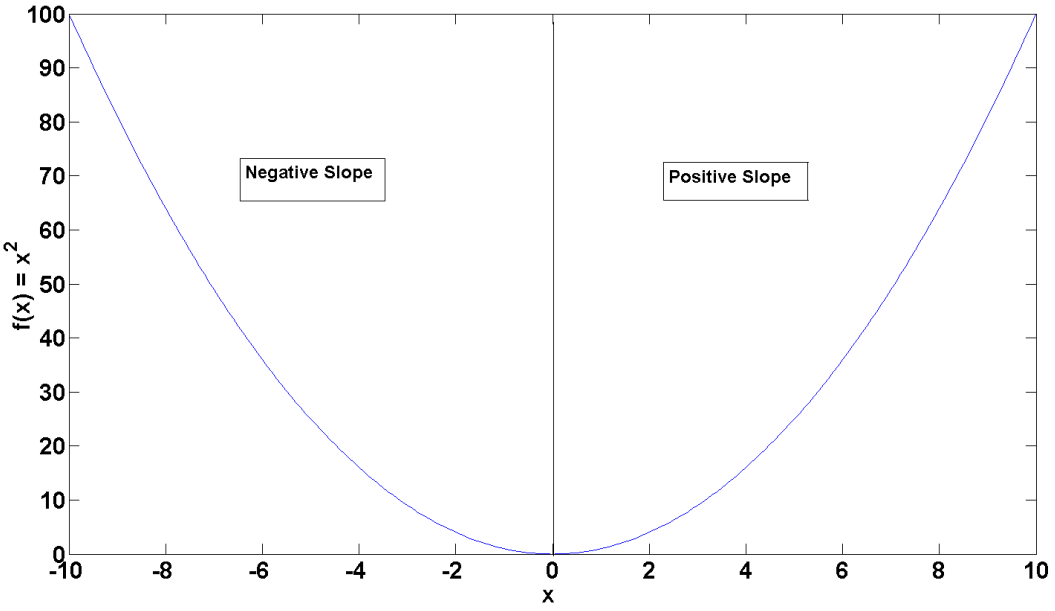


Figure 2.5-6 Forward Model and Domain Bifurcation According to Sign of the Slope

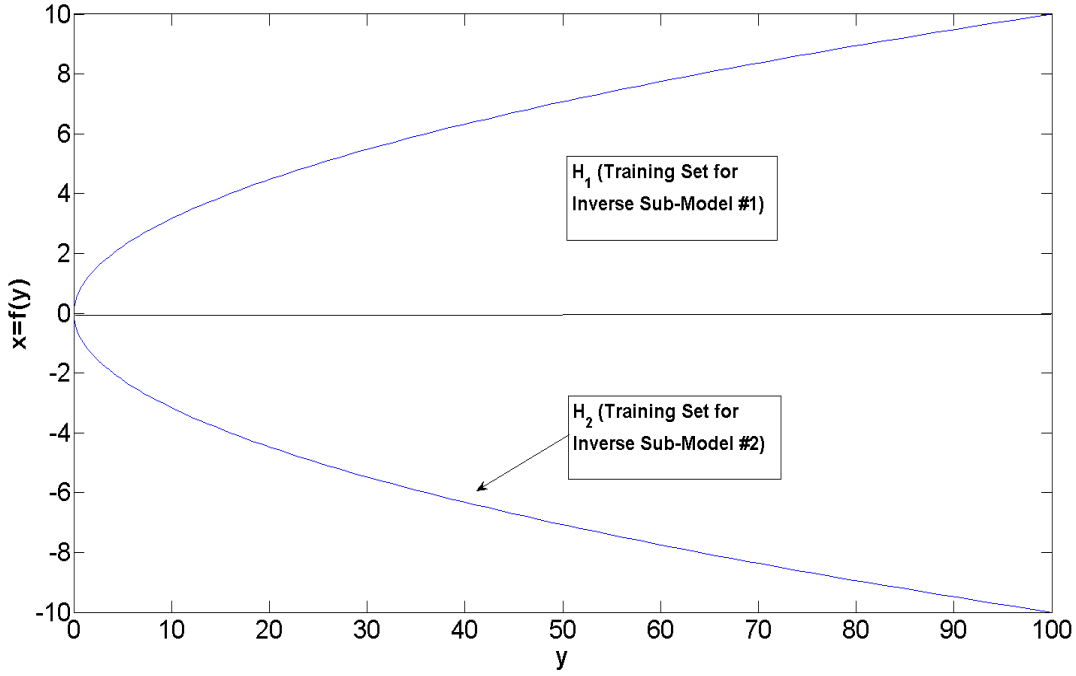


Figure 2.5-7 Corresponding Inverse Model Divided into Two Inverse Sub Models thus Eliminating Multi-Valued Solutions

The approach described above is for 1 input, 1 output model but in RF and MW design we encounter multivariable models and therefore we need to extend this formulation to multiple dimensions. To do so we need to compute derivatives of outputs versus inputs for forward model. Derivative information is formulated for only those inputs and outputs pairs that have been exchanged (to form the inverse model). Again considering a generalized neural network with n-element input vector x and m-element output vector y with I_x and I_y index sets as defined in Part B of Section 2.5.3-B. Therefore derivative computed at each training sample, for the inputs and outputs swapped to form inverse model, can be expressed as:

$$\frac{\partial y_i}{\partial x_j} | \, x = x^{(k)}, \qquad i \in I_y \ \ and \ \ j \in I_x \tag{2.5-34}$$

where k = 1,2,3,……..,K, and K is the number of samples in training data. Once we have this derivative information, training data is divided into two sub-groups for each pair of one element from I_x and one from I_y. This is done as follows:

If for k^{th} sample

$$\frac{\partial y_i}{\partial x_j} | \, x = x^{(k)} \; < \; \delta \tag{2.5-35}$$

then current sample is placed in a different group as opposed to the sample for which

$$\frac{\partial y_i}{\partial x_j} | \, x = x^{(k)} \; > \; \delta \tag{2.5-36}$$

Hence, this sample will be placed in a distinct group. Ideally δ is equal to zero but we can assign it a vary small positive value and that will introduce overlapping connection at the break point between the two groups. The choice of δ only affects the accuracy of the model in the overlapping region and rest of it remains unaltered.

If ,

Number of elements in set I_x = p
Number of elements in set I_y = q

Then,

Minimum number of *inverse-sub models* required = $\mathrm{p} \times \mathrm{q}$

After finishing this task we end up with at least $\mathrm{p} \times \mathrm{q}$ inverse sub-models. Now, the problem arises in the fact that, when the user submits an input to the inverse sub-models how to decide which sub-model to use. Henceforth, a mechanism has to be developed to choose the sub-model that gives the most accurate result. This technique will be discussed in the next chapters in detail against the backdrop of inverse neural network applied to design the transmitarray.

2.6 EXISTING USE OF NEURAL NETWORK CONCEPTS IN THE DESIGN OF PRINTED HIGH-DIRECTIVITY ANTENNAS

Existing applications of neural network concepts to high-directivity reflectarray antennas is described in this section. We note at the outset that in all these applications the neural network concepts have (in the terminology of this thesis) been used in as "forward" neural networks establish augmented databases. Also, these are all for the design of reflectarrays rather than transmitarrays[10]. In reflectarrays only the reflection coefficient phase is important; the magnitude is always unity. In transmitarrays both the phase and the amplitude of the element transmission coefficients are of interest. In order to be able to place this work in the context of the present thesis, it will be necessary to refer back to earlier sections of this chapter, as well as forward to sections still to come.

Reference [2.6-1] uses so-called Minkowski elements for the reflectarray unit cells. Three geometrical parameters, and the frequency, are the inputs to neural network model, and the reflection coefficient phase is the output. Thus four inputs are mapped to one output. The FDTD-based commercial electromagnetic simulation code *CST Microwave Studio* is the full-wave analysis used to generate accurate data to be used in the training of the neural network, which is used as an interpolator to augment the number of points available in the database. By comparing r*econstructed data* (obtained using the neural network) and additional full-wave data, it is established that the generalized regression neural network (GRNN) model can reliably characterize the reflection coefficient phase. Compared to the MLP neural network with feed forward computation and quasi-Newton as the training algorithm that we have utilized in setting up the neural network models in Section 3.3 of this thesis, GRNN has an advantage that it doesn't require multiple iterations before converging to the optimal set of weight vectors. GRNN only requires a single pass (one iteration) to 'learn'. The major disadvantage of GRNN compared to quasi-Newton based neural network models is that it requires substantial computation time (and hence resources) to evaluate new points once learning is complete. Given that, in the present work, we need to repeatedly access the forward neural network model relationship[11] that

[10] To the best of this author's knowledge, use of the superb capabilities of neural networks in transmitarray design has not yet been reported by others.
[11] This will be apparent in Algorithm-6 and Algortihm-11 in Sections 3.5 and 4.5.

characterizes input-output relationship of transmitarray unit cells, use of the GRNN would lead to enormous computational cost associated with each design iteration. Parallelization can be implemented to combat this bottleneck when using the GRNN approach, but it leads to complexity in algorithm formulation.

Reference [2.6-2] describes the generation of the augmented reflection phase database for reflectarrays that use Minkowski elements, but using the multilayer perceptron (MLP) neural network approach described in Part D of Section 2.5.1. In order to "invert" the augmented database during the reflectarray design direct use is made of a numerical optimisation routine (particle swarm optimization or PSO). This approach does have one excellent characteristic, namely that element dimensions are selected according to the sensitivity of the resulting elements' reflection phase to small dimensional inaccuracies and frequency. The first stage of the approach in the present thesis is similar to that in [2.6-2] in the use of the MLP route for generation of the augmented database (albeit for a transmitarray surface rather than a reflectarray). However, use of the PSO-algorithm approach for inversion of the database not only requires long computing times for each element[12] but also provides only one solution. As we will see in Section 3.5, we need multiple possible element dimension outputs (the data are multivalued) for a given transmission coefficient phase value in order to select the best possible amplitude. Hence, the PSO approach was deemed unsuitable for the inverting process in the present thesis. In [2.6-3] similar methods are applied to the design of an X-band reflectarray by the same group which authored [2.6-2].

The MLP neural network topology is employed in [2.6-4] to augment the database for the reflection coefficient phase of reflectarrays, for four different types of elements: 1-layer square patch, 1-layer modified Maltese cross, 2-layer stacked patch, and 3-layer square patch. This validates the appropriateness of the MLP model for database "forward" augmentation. The MLP model will also be used in this thesis, albeit for transmitarray elements rather than reflectarray ones. Reference [2.6-4] does not discuss the inversion of the database.

[12] And there are more than 2000 elements for each transmitarray.

The design of a reflectarray antenna with a contoured-beam radiation pattern is the subject of [2.6-5]. A neural network model, trained using full-wave data points as usual, is used to augment the design database, and its accuracy is confirmed through comparison to additional full-wave points. In order to reduce the overall time in the neural network modelling symmetry is exploited, and multi-step training along with other improvisations is used. However, only forward neural network modelling is used. In Section 3.3 we will show that the use of self-organizing map (SOM) based clustering in neural network generation can achieve similar accuracy.

2.7 CONCLUSIONS

Section 2.2 provided a discussion of the assumptions used in the design of transmitarrays. The approximate (yet accurate) techniques used for computing transmitarray radiation patterns was also provided, as well as the basics of the transmitarray design procedure that will be used in this thesis. Although this is not new, it is not available elsewhere in the collated form given here. We have also used some structured notation for increased clarity. Section 2.3 briefly discusses the approaches one could think of for transmitarray design, and explains why one has been chosen here (as in fact already outlined in Section 2.2.4). Two ways of generating initial design database points are mentioned in Section 2.4, with reasons given for choosing one of these in later chapters.

As the thesis proceeds it will become increasingly clear that the use of full-wave analysis alone for database generation is not computationally efficient. It will also become clear that inversion of the database, which is necessary for completing the transmitarray design, is a difficult problem. This thesis will show that neural networks can be used to overcome both these difficulties. Section 2.5 therefore provides an introduction to the neural network concept, with some emphasis on the inverse neural network ideas that have previously not been used in antenna work, and which will be developed for such application in Chapter 3.

Details of existing usage of neural network modelling in the reflectarray (as opposed to transmitarray) antennas are described in Section 2.6. We found that whereas “forward” neural networks have been used for these antennas, the inverse neural network concept has not.

CHAPTER 3
Design Database Construction & Inversion Using Neural Network Characterization of Computational Electromagnetic Data

3.1 PRELIMINARY REMARKS

Figure 3.3-1 summarizes the complete transmitarraya design procedure mentioned in Section 2.4. In this chapter, in Section 3.2, the database is generated and owing to some shortcomings identified here, in Section 3.3, we discuss the need for artificial neural network characterization of transmitarray antenna element sizes and corresponding transmission characteristics. Along with that the process of forward neural network model generation using self-organizing maps is elucidated.. In Section 3.4 the subsequent inverse model generation is explained to invert the mapping between inputs and outputs of transmitarray elements, and the next step is to extract output from the trained inverse neural network model, which itself is a complex procedure, is proposed in Section 3.5. Finally, in Sections 3.6 and 3.7, we establish the accuracy of the neural network model thus generated, and conclude the chapter, respectively.

3.2 DESIGN DATABASE

Database generation consists of various stages starting from the unit cell characterization according to the transmitarray to be designed, setting up the full wave EM simulation of the unit cell using Floquet port analysis and periodic boundary conditions, and storing the relevant S-parameters.

3.2.1 Unit Cell Specifications

In this thesis, the transmitarray under consideration is a 3-layered structure as shown in Figure 3.2-1 where the exterior layer of each cell has identical square conducting patch dimensions a_1 and the interior layer of the unit cell, the layer embedded in the dielectric, has square conducting patch dimension a_2, which is independent of the exterior layer. Dielectric material with dielectric constant ε_r is filled between the adjacent layers and electromagnetic field impinges upon this structure by virtue of the presence of feed horn.

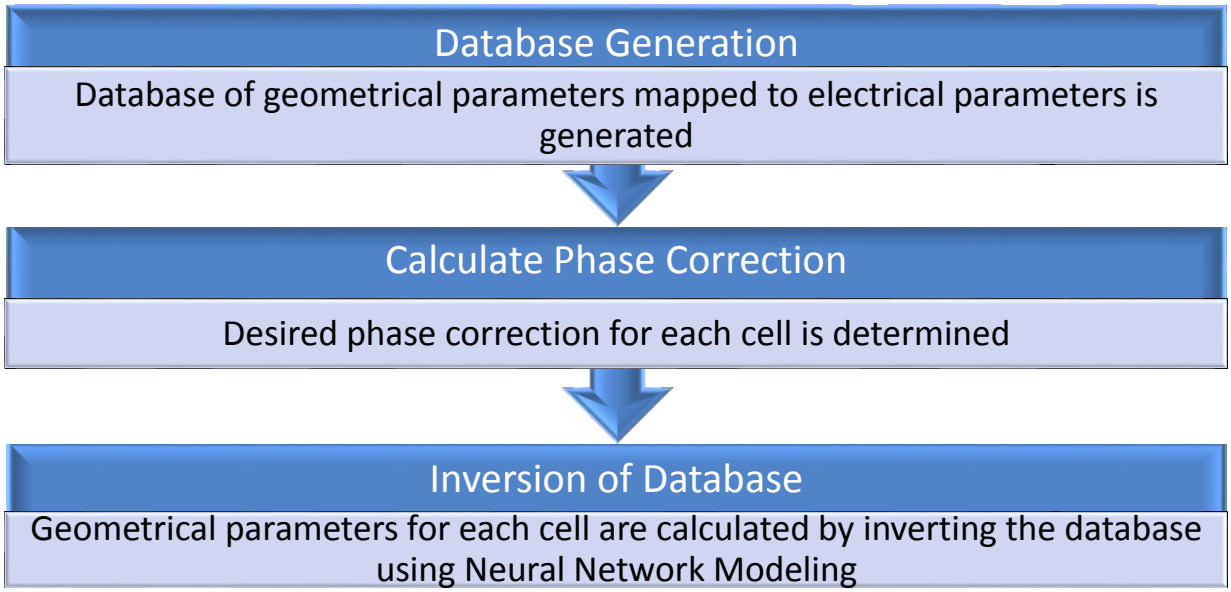


Figure 3.1-1 Transimitarray Design Approach

3.2.2 Periodic Structure Analysis

As discussed before, each unit cell of the transmitarray has unique dimensions depending upon the desired phase shift and for the periodic structure analysis of a particular cell it is assumed that the particular cell (let's say n^{th} cell) is lying in an infinite periodic structure in the x-y plane as shown in Figure 3.2-2

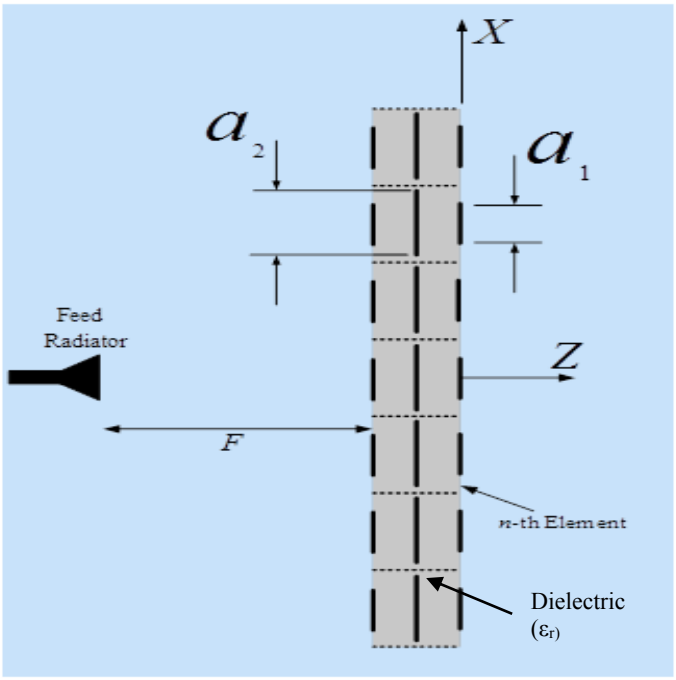


Figure 3.2-1 Side view of the transmitarray showing geometrical dimensions associated with each cell

Figure 3.2-2 depicts the periodic structure analysis procedure, where nth element/cell's geometry is shown (b) and it is assumed to be residing in the periodic structure in (d). Now, consider Figure 3.2-2 (d) where the n^{th} cell is lying in the periodic structure; 'g_l' represents inter-element spacing and 's' is the cell size. We want to compute the S-parameters for this periodic structure, Due to the infinite structure assumption, appropriate periodic boundary conditions reduce the problem to a field analysis for a single cell. But to facilitate that, following conditions needs to be met:

$$s < \frac{\lambda_0}{1 + |\operatorname{Sin}\theta_{\max}|} \tag{3.2-1}$$

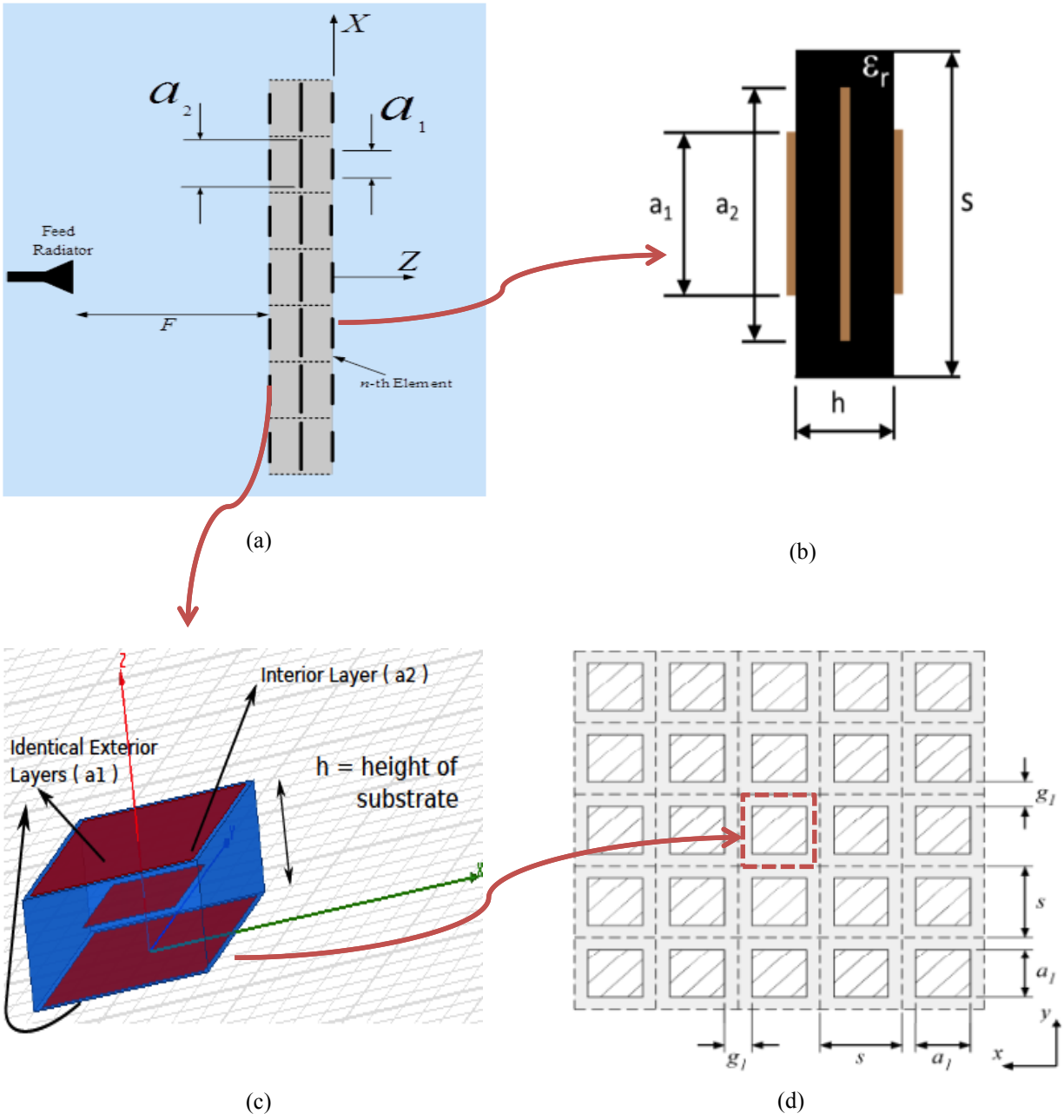


Figure 3.2-2: (a) Transmitarray side view, (b) Side of the nth cell, (c) HFSS Geometry for the nth cell, (d) nth cell assumed to be lying in a periodic structure for analysis

s being the unit cell size, λ_0 is the free space wavelength of the incident electromagnetic wave and θ_{max} is the maximum angle of the incidence which will be for the cell located farthest from the center of the transmitarray. Therefore, for the case when $\theta_{max} = 0^o$ i.e. normally incident wave, s < λ_0 whereas for $\theta_{max} = 90^o$, s < $\lambda_0/2$. In the periodic structure analysis carried out in this work, we have used the lower threshold which is s < $\lambda_0/2$ to account for the fact that cells lying far away from the center (given that we are considering an infinite periodic structure), the incident wave would have a large non-zero incident angle.

We have used a square unit cell in our design, of size 's', as illustrated in Figure 3.2-2(d). The full wave electromagnetic computations were performed using Ansys HFSS, a commercial full wave simulator employing finite element method based method. If E_y^{in} and E_y^{out} are the incident and transmitted fields, respectively, which are defined at the nth element's center, then as described in Section 2.2-4 they are related as

$$E_y^{out}(x_n, y_n, 0) = T_n^y E_y^{in}(x_n, y_n, -h) \qquad (3.2\text{-}2)$$

where quantity T_n^y is the complex transmission coefficient for the y-component of the field through the nth element assumed to be equal to the transmission coefficient of a plane wave that is incident, with incidence angles $(\theta_{inc}, \phi_{inc})$ in Figure 2.2-1, on a structure composed of an infinite number of elements in the x-y plane that are all identical to the nth element (which reduces it to a doubly-periodic planar structure). This is of course an approximation, since the n-th element in the actual transmitarray will not be surrounded by elements identical to itself. The value of T_n^y will be a function of frequency, the properties of the substrate material, the widths of the conductors at each layer of the n-th cell, and the incidence angles $(\theta_{inc}, \phi_{inc})$. So we can write them as $T_n^y(a_1, a_2, \theta_{inc}, \phi_{inc}, f, \varepsilon_r)$. In what follows we fix the material parameter (ε_r, h) and frequency (f) as well as assume that the plane wave is normally incident[13], and therefore T_n^y is

[13] Oblique incidence will be considered in Chapter 4

only a function of a_1 and a_2 namely T_n^y (a_1, a_2). Because of the 90^o symmetry of the particular structure under consideration, $T_n^x = T_n^y$.

Tabulated values of T_n^x and T_n^y are referred to as the ***design database*** for the particular transformation surface (transmitarray in this case). To emulate the infinite periodic structure perfect electric conductors (PEC) are placed along the x-axis, and perfect magnetic conductors (PMC) are placed along y-axis, on the boundaries of the square unit cell as shown in Figure 3.2-3. This imposes a field polarization in the y-direction (vertical polarization) for the incident electromagnetic wave. This is automatically accomplished by HFSS when being used to analyze plane wave scattering from infinite periodic structures. The square unit cell is chosen because it can be used for dual-polarization or circular polarization applications as well, since a square's order of rotational symmetry is 4 (90^0 rotational symmetry). Hence any of the two orthogonal polarizations would be supported.

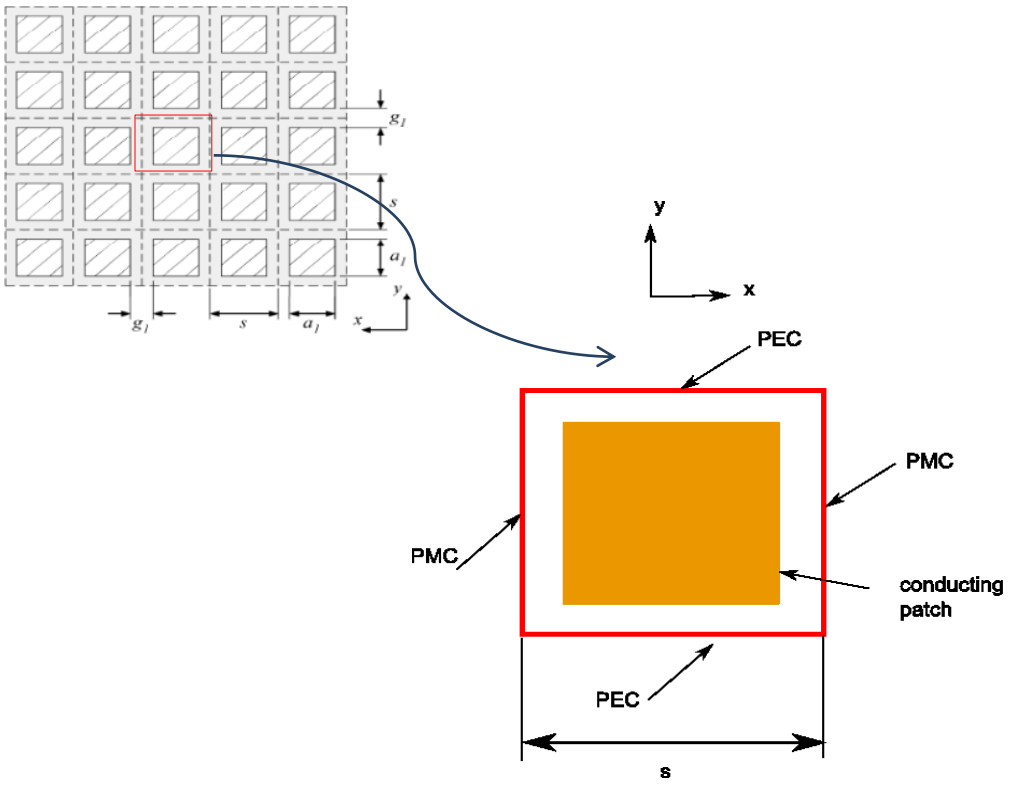


Figure 3.2-3 Top View of the Unit Cell with Boundary Conditions Pertaining to Infinite Periodic Structure

Using this unit cell in the finite element computations using HFSS [3.2-1], we generate a database of transmission coefficients as indicated above. We only vary the dimensions of the conducting patches of the unit cell (a_1, a_2) and keep other parameters fixed as[14]:

s = 3 mm	cell size
h = 1 mm	height of substrate
ε_r = 2.2	relative permittitivity
f = 30 GHz	frequency

's' is so chosen that it is smaller than half of the free space wavelength in accordance with equation (3.2-1) but large enough to be above the tolerance of photolithographic etching process used to fabricate the structure. The frequency of operation was chosen to lie in a band that will enable us to compare the results in this thesis with work on transmitarrays done by others.

3.2.3 Database Generation

We vary a_1 and a_2 and fix all other variables and find the corresponding transmission coefficient for the n^{th} element lying in an infinite periodic structure in the x-y plane. A non-uniform sample space for (a_1, a_2) was chosen based upon the observed sensitivity in the periodic structure response to changes in (a_1, a_2). In regions of more rapid variation in the response the higher the sample density used.as depicted in Figure 3.2-4. In total 4893 sample points (a_1, a_2) are selected, and a full wave analysis is performed for each point.

As precisely noted in Section 2.2-6, the transmission coefficient (T_n^y) will be referred to as S_{21}. We describe S-parameter database as:

$$\left(arg\{ S_{21}\} \ , \ |S_{21}| \right) = g_{database} \left(a_2, \ a_2 \right) \qquad (3.2\text{-}3)$$

[14] Reasoning behind the selection of theses parameters is provided in Chapter 5.

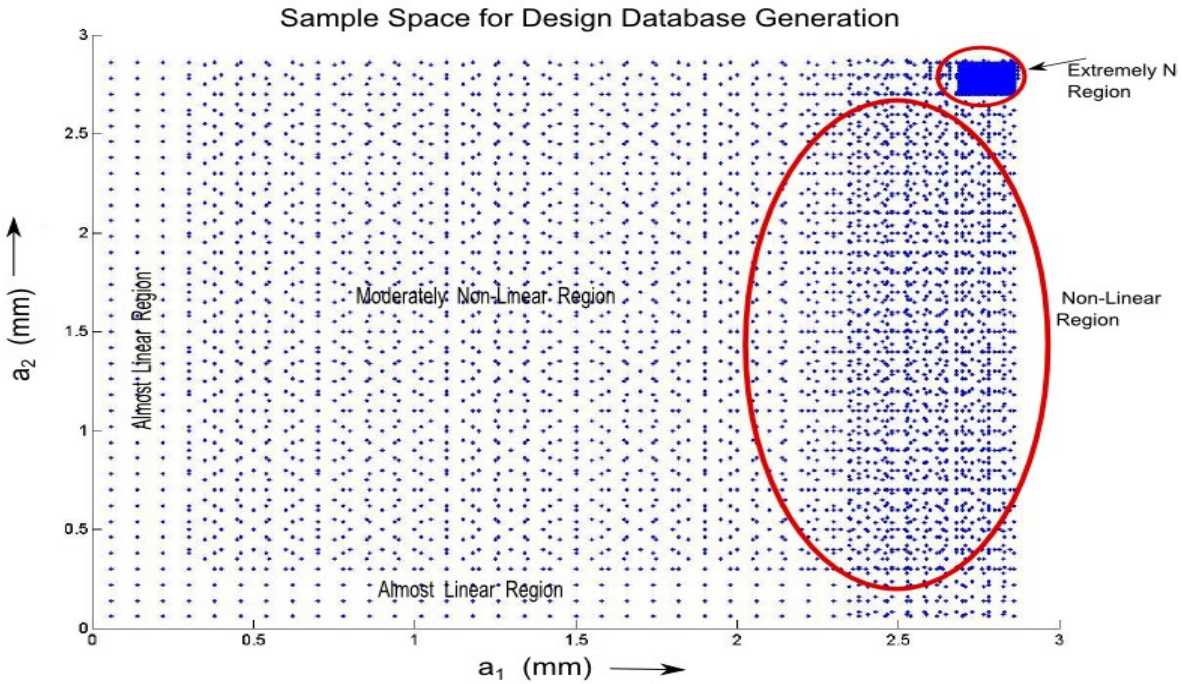


Figure 3.2-4 Design Database Sample Space Illustrating Non-Uniformity of Samples Selected based upon Degree of Non-Linearity of Response

We feed the samples (a1 , a_2) to the full wave 3-D electromagnetic simulator and store the corresponding phase ($arg\{S_{21}\}$) and amplitude ($|S_{21}|$) of the transmission coefficient. This provides the S-parameter databases shown in the next few figures.

In Figure 3.2-5, the range of phase values was from -270^o to 37^o which shows not the entire phase range (360^o) is covered. It is due to physical limitations of the unit cell structure used in this design. Figures 3.2-5 and 3.2-6 show the transmission phase and amplitude, respectively, with respect to all the combinations (a_1, a_2) shown in Figure 3.2-4. Figure 3.2-7 however, demonstrates the possible transmission amplitudes available for each transmission phase value. Now, ideally we would want unit transmission amplitude for each phase value (0 dB) but due to computational limitations, as well as the physical limitations of the particular structure, that's not possible. Therefore, in the optimal case we should be able to get transmission amplitudes marked by the red envelope in Figure 3.2-8 for each phase value.

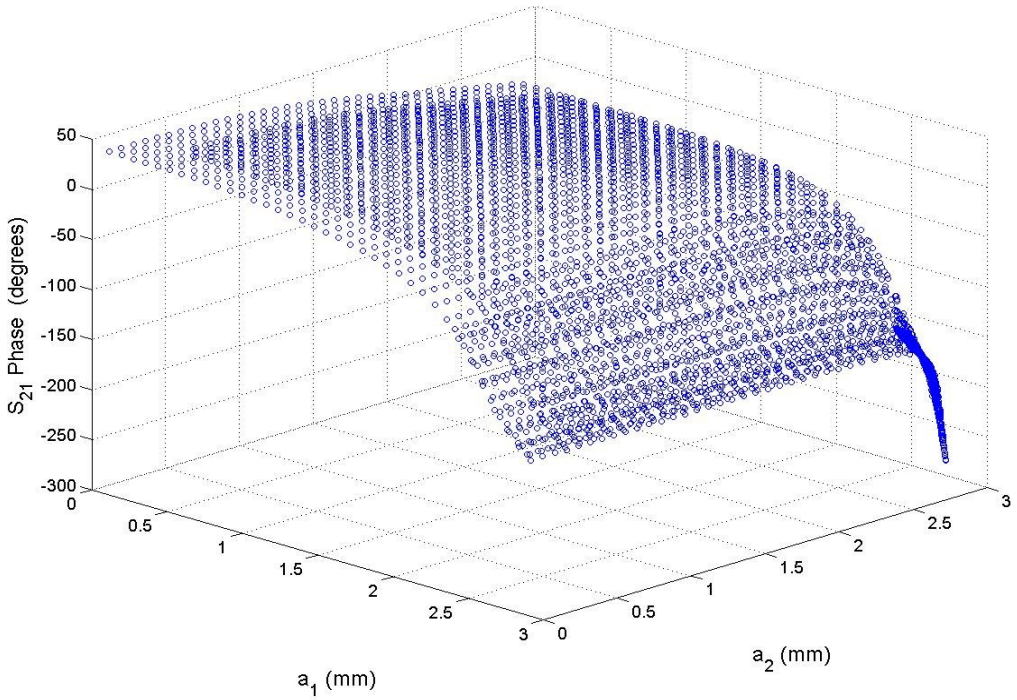


Figure 3.2-5 Transmission Coefficient Phase Versus (a_1 , a_2)

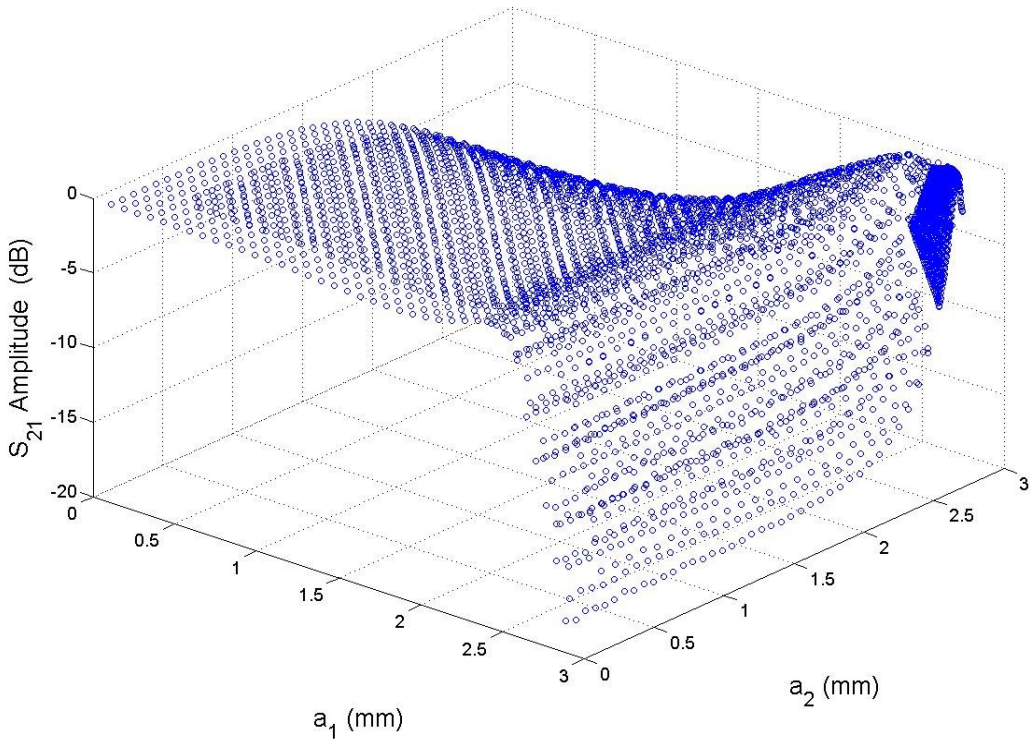


Figure 3.2-6 Transmission Coefficient Amplitude Versus (a_1 , a_2)

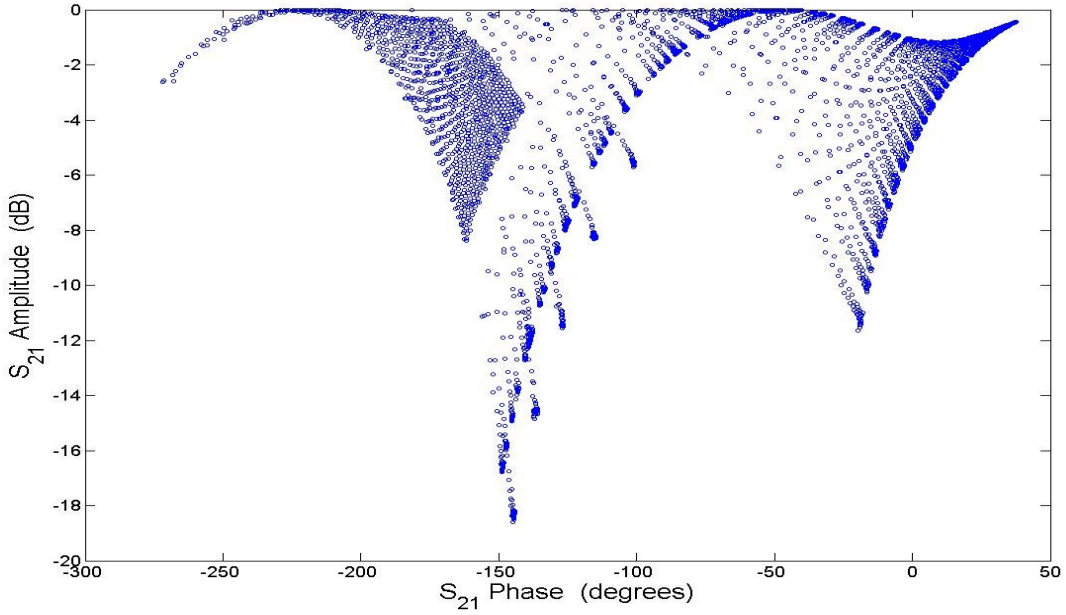


Figure 3.2-7 Transmission Coefficient Amplitude Versus Phase

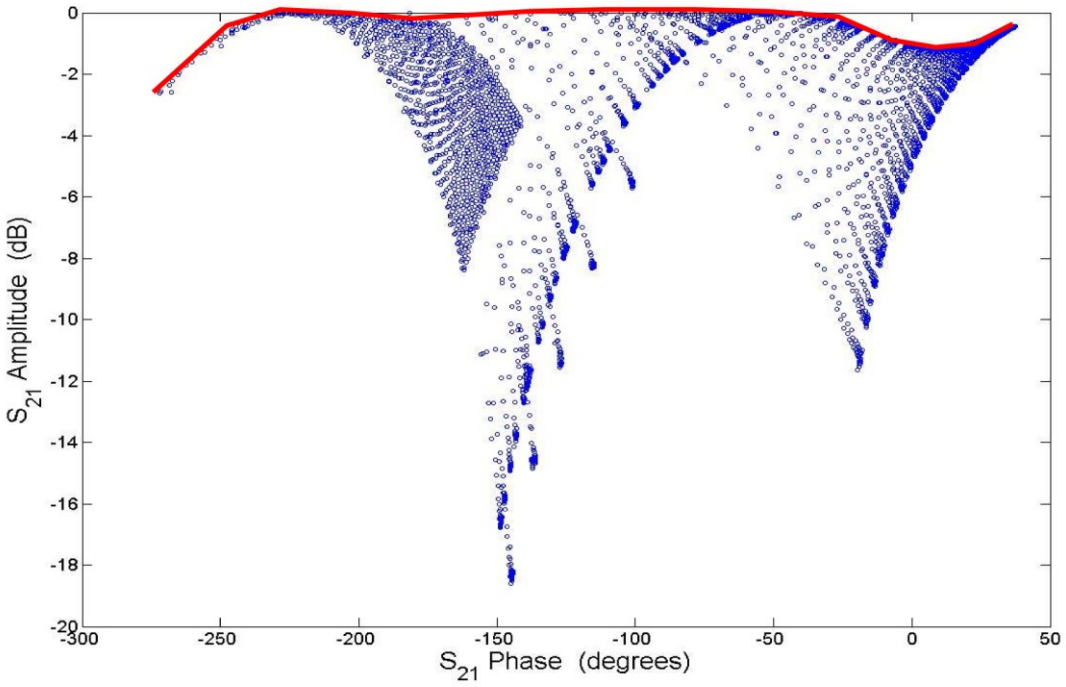


Figure 3.2-8 Optimal Case where we get Maximum Possible Transmission Amplitude for a Given Phase

3.3 ARTIFICIAL NEURAL NETWORK (ANN) CHARACTERIZATION OF TRANSMITARRAY CELL RESPONSES

The transmitarray design process (Figure 3.1-1) involves inversion of the database in step-3. In other words, the $g_{database}$ map in equation (3.2-3) needs to be inverted i.e. outputs become inputs and vice versa. Due to the highly non-linear nature of $g_{database}$, evident in Figures 3.2-5 and 3.2-6, the inverse map represented by $g_{database}^{-1}$ is even more non-linear. We therefore, would need large amount of data samples (of the order of 100,000 but currently $g_{database}$ contains only 4893 samples) for *inverse neural network modelling* which would be impractical given the sheer amount of computation time and extensiveness in the full wave electromagnetic simulations. Therefore, we first use a 'forward' artificial neural network (ANN) to accurately characterize $g_{database}$ using the 4893 full wave points in order to accelerate the response time for the calculation of transmission coefficient at many more (a_1, a_2) values, for use in the inversion process.

Therefore, two forward ANN models are created using a 3-Layered MLP[15] neural network topology; one for *S_{21} phase* and another *S_{21} amplitude.* These models are generated as follows :

Forward model 1 or forward phase model, $f_{ann,1}^{forward}$, maps (a_1, a_2) to *S_{21} phase*

$$\arg\{S_{21}\} = f_{ann,1}^{forward}(a_1, a_2) \tag{3.3-1}$$

Forward model 2 or forward amplitude model, $f_{ann,2}^{forward}$, maps (a_1, a_2) to *S_{21} amplitude*

$$|S_{21}| = f_{ann,2}^{forward}(a_1, a_2) \tag{3.3-2}$$

where,

$$x = \{a_1, a_2\} \tag{3.3-3}$$

represents neural network input vector and,

[15] See Section 2.2-5

$$y = \{S_{21}\} \qquad \left\{ \begin{array}{ll} y = \arg\{S_{21}\} & \forall\ f_{ann,1}^{forward} \\ y = |S_{21}| & \forall\ f_{ann,2}^{forward} \end{array} \right\} \tag{3.3-4}$$

represents the neural network output vector. Given the highly nonlinear nature of the function $S_{21}(a_1, a_2)$, apparent in Figure 3.2-5 and Figure 3.2-6, forward neural network training process becomes difficult to do accurately due to the limited amount of training data samples. In order to possibly surmount this problem, the *SOM* method is next implemented.

3.3.1 Self-Organizing Map (SOM)

A) SOM Topology

The SOM approach was briefly introduced in Part G of Section 2.5-1. We place a higher dimensional data space onto the lower dimensional map by finding the node with the closest (smallest distance metric) weight vector to the data space vector. The total number of cluster centers to be determined from training data is equal to the total number of neurons in SOM. In the sample distribution each neuron represents a cluster center. The SOM network is created from a 2D abstract lattice of 'nodes', each of which is fully connected to the input layer. Figure 3.3-1 shows the SOM network of 3 X 3 nodes (in the computational layer) connected to the input layer representing a two dimensional vector.

Each node has a specific topological position (a point in the abstract lattice[16]) and contains a vector of weights of the same dimension as the input vectors. That is to say, if the training data consists of vectors of dimensions *m* such as:

$$V = [V_1, V_2, V_3 \ldots V_m]$$

Then each node will contain a corresponding weight vector of dimensions *m*:

$$W = [W_1, W_2, W_3 \ldots W_m]$$

[16] We emphasize the adjective 'abstract' to avoid confusion with use of the word lattice when referring to the physical transmitarray structure.

In our case, each input vector V has 3 dimensions (m = 3) and so does the weight vector W for each node[17]

$$V = [\ a_1\ \ a_2\ \ \arg\{S_{21}\}\ ] \quad \text{for } f_{ann,1}^{forward} \qquad (3.3\text{-}5)$$

and

$$V = [\ a_1\ \ a_2\ \ |S_{21}|\ ] \qquad \text{for } f_{ann,2}^{forward} \qquad (3.3\text{-}6)$$

The output of the SOM is the distribution of input sample space into discrete clusters according to the criteria in Algorithm-1.

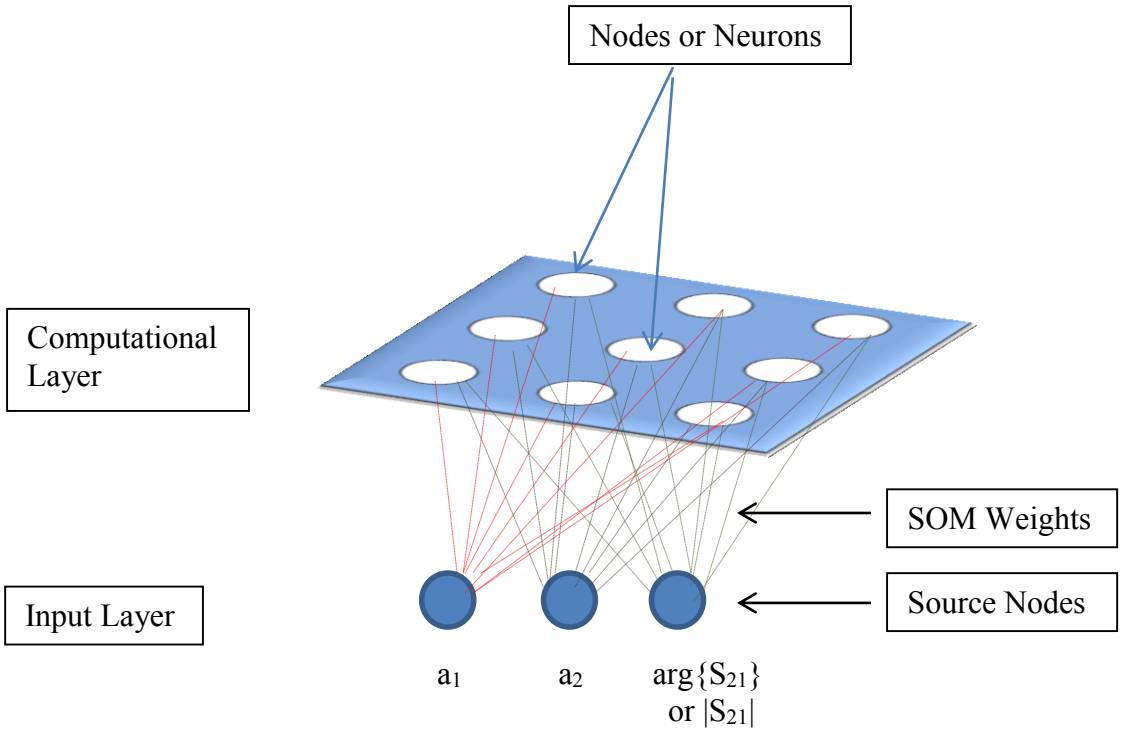


Figure 3.3-1 2D Self-Organizing Map

[17] Input vector V in the case of SOM is different from input vector 'x' in equations (3.3-1) and (3.3-2)

B) SOM Learning

For SOM training, the weight vector associated with each neuron moves to become the center of a cluster of input vectors. In addition, neurons that are adjacent to each other in the topology should also move close to each other in the input space, therefore it is possible to visualize a high-dimensional input space in the two dimensions of the network topology. The default SOM topology is hexagonal but we have used a rectangular topology. Figure 3.3-1 shows the neuron locations in the topology, and indicates how many of the training data are associated with each of the neurons (cluster centers). The topology is a 3-by-3 rectangular grid, so there are 9 neurons. Training data is divided into 9 clusters using SOM, as shown in Figure 3.3-1. Basically it's a 2-D matrix with each element representing a cluster. Figure 3.3-2 and 3.3-3 show the SOM topology and distribution of training samples (input vectors) among the clusters (nodes) respectively for the phase model (same procedure is followed for the amplitude model). 4893 samples are distributed among 9 clusters based upon criteria in ***Algorithm-1***. The generalized SOM Learning Algorithm is as follows, details of which can be found in [3.3-1]:

Algorithm-1

Training occurs in several steps and over many iterations:

1.) Each node's weights are initialized.
2.) A vector is chosen at random from the set of training data and presented to the lattice.
3.) Every node is examined and the one for which weights are closest to the input vector is the winning node The winning node is commonly known as the **Best Matching Unit (BMU).**
4.) The radius of the neighborhood of the BMU is now calculated. It is initiated with a large value, typically set to the 'radius' of the lattice, but diminishes with each time-step. Any nodes found within this radius are deemed to be inside the BMU's neighborhood.
5.) Each neighboring node's (the nodes found in step 4) weights are so adjusted that they get closer to the input vector. The closer a node is to the BMU, higher is the degree to which its weights are altered.
6.) Repeat step 2 for N (number of training samples) iterations.

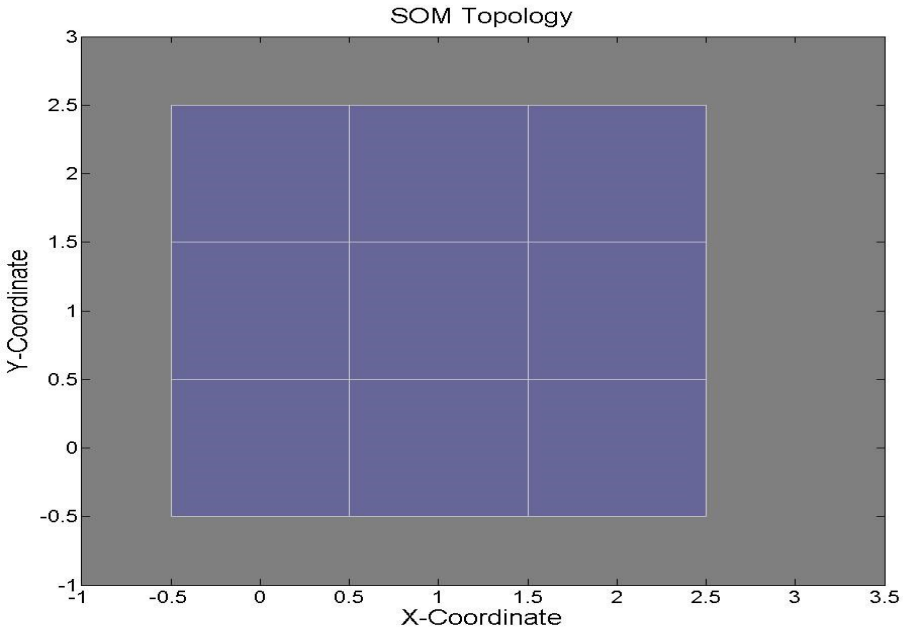


Figure 3.3-2 3X3 Rectangular Grid Representing Nodes/ Neurons in the Computational Layer of the SOM

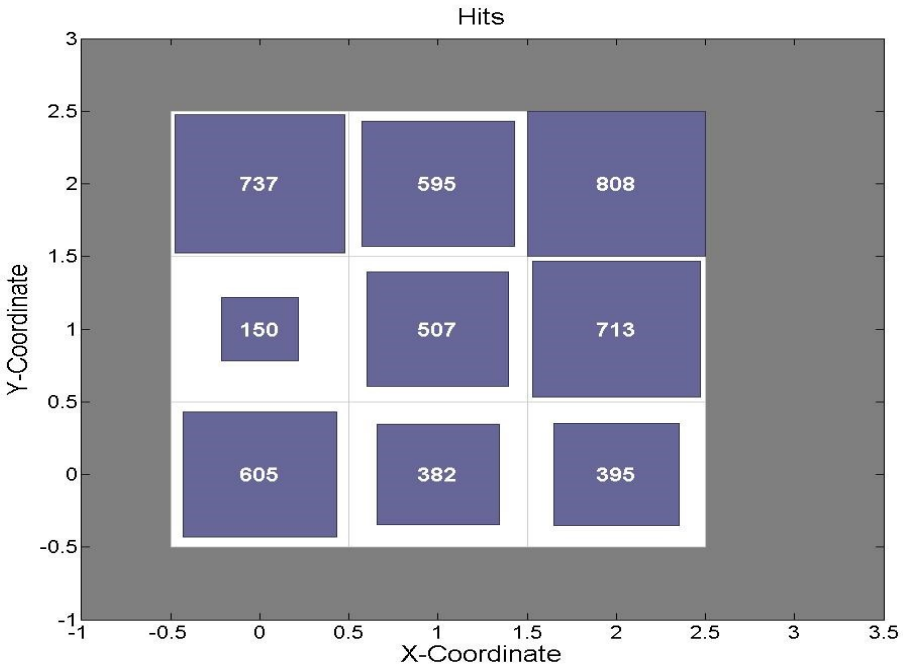


Figure 3.3-3 Number of **Hits** for each Cluster i.e. Number of Input Vector Samples Associated with Each Cluster Center for the Phase Model $f_{ann,1}^{forward}$

C) Sub-Model Generation for each SOM Cluster

Once all the input samples of the form V (V is the input vector for SOM learning process which is formed by combining the inputs x and outputs y of the forward neural network in equation (3.3-1)) are fed to the SOM and consequently clusters are formed. Nine distinct sub-forward neural network models were generated for the *forward phase* neural network model (3.3-1) - one model for each SOM cluster. Each of the nine models have input vector of two elements [a_1 a_2] and an output vector of one element [arg{S_{21}}] . Mathematically, the variable vector space of $f_{ann,1}^{forward}$ (3.3-1) is divided into nine *sub-vector spaces,* and each sub-vector space is used to generate *sub-models* given by,

$$ {}^{i}f_{ann,1}^{forward} \qquad \text{where} \quad i = 1,2,3\ldots.9 \qquad (3.3\text{-}7) $$

Each of the 9 sub-neural network models were generated using a 3 layered multi-layer perceptron (MLP) as the topology and were trained separately using the corresponding dataset from the related SOM cluster. Finally, the outputs were combined to obtain a larger database.

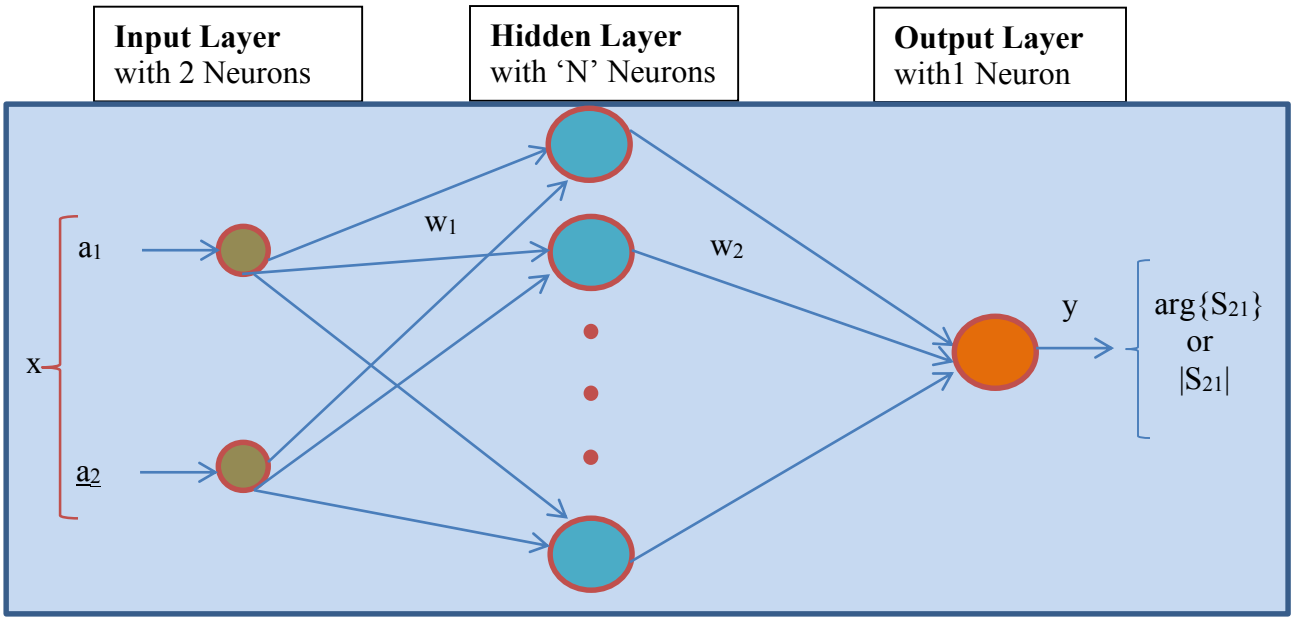


Figure 3.3-4 Multi-layer Perceptron Topology Used for Forward Neural Network Sub-Model Generation for Phase ($f_{ann,1}^{forward}$) and amplitude ($f_{ann,2}^{forward}$) for each SOM Cluster

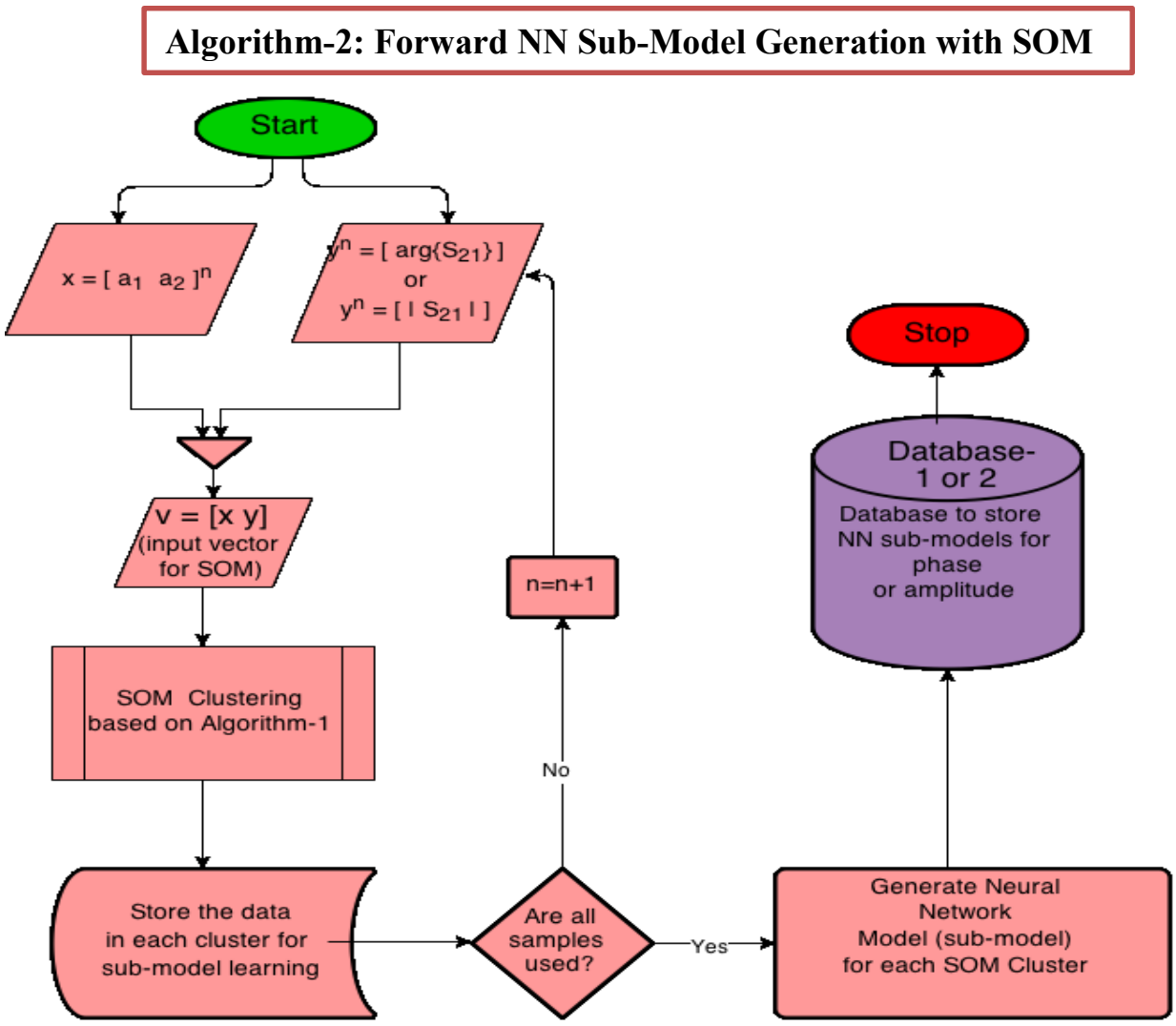


Figure 3.3-5 Algorithm for Forward Sub-Model Generation Process Employing SOM Clustering

Figure 3.3-5 delineates the forward neural network sub-model generation process using the SOM clustering explained in Section 3.3-1(B). Here, *x* and *y* represent the input vectors for the complete forward model for phase and amplitude as given in equations (3.3-1) and (3.3-2), and they are combined to form V, the input vector for the SOM. This way sub-models are *trained* and stored in a *database* (***database-1*** for phase models and ***database-2*** for amplitude models). It is important to note that sample space for each of the sub-models is determined by the SOM as per the distribution shown in Figure 3.3-3. In essence, *database-1* represents collection of data structures (in this case neural networks) of the form:

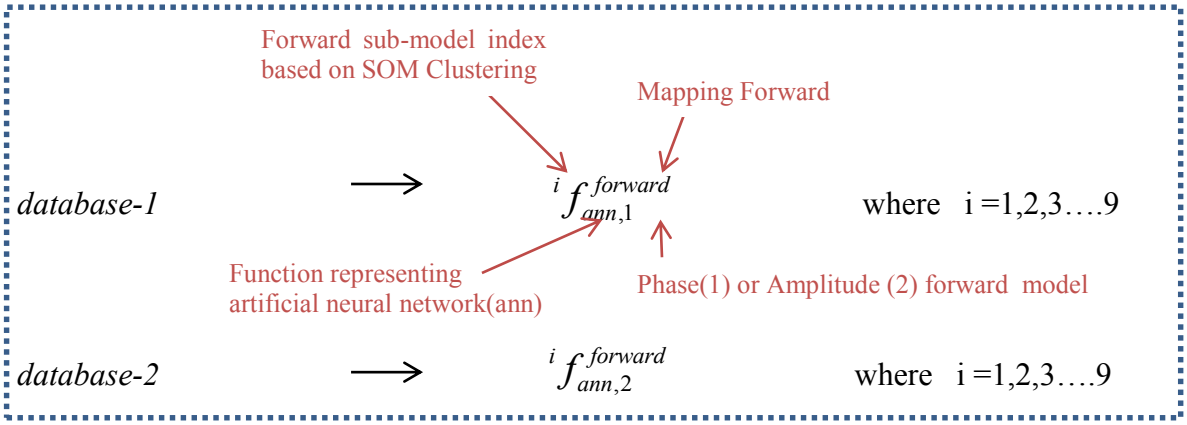


Figure 3.3-6 Definition of the Neural Network Databases and Explanation of the Symbols Used for Sub-Model Representation

A quasi-newton algorithm [2.5-14] was used to train the forward phase/amplitude sub-models according to the process flow described in Figure 2.5-3, and equations (2.5-5) to (2.5-10) in Section 2.5 are used for feedforward computations, with the sigmoid activation function utilized in the process. During the training, weight vector w = [w_1 w_2] (shown in Figure 3.3-4) is updated using aforementioned optimization algorithm and to summarize the neural network training and testing procedure lets define some terminology here. Since, we have clustered the complete dataset stored in $g_{database}$ into 9 clusters, we will show the **training** and **testing** conventions utilized in our model generation for one SOM cluster and it would be identical for other 8 clusters. Let's consider a forward sub-model from *database-1* for $^{i}f^{forward}_{ann,1}$ (i = 1),

N → represents the total number of data samples in this cluster

I_s → represents the index set for all the samples in the cluster, I_s = [1 2 … N]

I_r → represents the index set of training data as well as validation data

I_e → represents the index set of test data

Training and testing datasets are formed by dividing the complete data set (for the sub-model in question) by a given proportion. In our case we split I_s into two sets with 75%-25% split such that I_r contains 75% of the samples for training and I_e has a total share of 25% of the samples used for testing. We define training error which quantifies the error for the current set of weights

and is calculated again after every update of weight vector. Now, using the generalized equation for quantifying training error and modifying it for the training of ${}^{i}f_{ann,1}^{forward}$ (i=1) model, we get

$$E_{training}(\mathrm{w}) = \frac{1}{2}\left[\sum_{n\in I_r}\sum_{j=1}^{1}(y_j(\mathrm{x}_n,\mathrm{w}) - y'_{n,j})^2\right]^{\frac{1}{2}} \tag{3.3-8}$$

where,

- n = 1,2,3…N (sample index)
- x_n = n^{th} input vector sample such that, $x_n = [\, a_1, a_2 \,]_n$
- $y_j(x_n,w)$ = j^{th} component of the output vector y computed for the n^{th} sample and current weight vector ($y = \arg\{S_{21}\}$)
- $y'_{n,j}$ = actual output from the dataset used for training
- j = index for the output vector

Since the output, for the model in question, is in fact scalar; therefore, j=1 is the only possibility in equation(3.3-8) and one of the summations can be neglected. Thus, equation (3.3-8) can be rewritten as[18]:

$$E_{training}(\mathrm{w}) = \frac{1}{2}\left[\sum_{n\in I_r}(y_1(\mathrm{x}_n,\mathrm{w}) - y'_{n,1})^2\right]^{\frac{1}{2}} \tag{3.3-9}$$

Once the training error E_{training} is below the prescribed limit, the training is terminated. The next step is neural network testing, and we utilize the dataset ascribed specifically to testing using the indices in the testing index set I_e. The average test error is quantified as:

$$E_{test}^{average} = \frac{1}{\overline{\overline{I_e}}\,2}\left[\sum_{n\in I_e}\left(\frac{(y_1(\mathrm{x}_n,\mathrm{w}) - y'_{n,1})}{y'_{\max} - y'_{\min}}\right)^2\right]^{\frac{1}{2}} \tag{3.3-10}$$

In equation (3.3-10) $\overline{\overline{I_e}}$ is the number of elements in the index set I_e. Similarly, the worst case test error is calculated as:

---

[18] In this case we are using Euclidean Norm for the error computation but norm of any order can be used.

$$E_{test}^{worst\ case} = \max_{n \in I_e} \frac{(y_1(\mathrm{x}_n, \mathrm{w}) - y'_{n,1})}{y'_{\max} - y'_{\min}} \tag{3.3-11}$$

These error measures defined above are calculated for each of the 9 clusters for both the phase and amplitude models, and the mean of these errors tabulated as shown below.

Table 3.3-1 Average Test Error

Forward Model	Average Test Error	
	No SOM clustering	SOM Clustering (mean error of all sub models)
$f_{ann,1}^{forward}$ (phase)	4 %	0.8 %
$f_{ann,2}^{forward}$ (amplitude)	11 %	2.3 %

Table 3.3-2 Worst Case Error

Forward Model	Worst Case Test Error	
	No SOM clustering	SOM Clustering (mean error of all sub models)
$f_{ann,1}^{forward}$ (phase)	10%	4 %
$f_{ann,2}^{forward}$ (amplitude)	19 %	9 %

Significant improvement in the neural network model performance is made having implemented SOM as is evident in Table 3.3-1 and Table 3.3-2. These forward sub-models are then combined to recreate original forward models $f_{ann,1}^{forward}$ and $f_{ann,2}^{forward}$. In this way 250,000 database points were generated to be utilized for inverse modeling. Figure 3.3-7 represents the augmented input sample space which was finally used for database inversion process, and Figure 3.3-8 shows the corresponding output sample space. These sample spaces result from the combined sub-spaces of the SOM clustered ANN models.

C) Output Extraction from *database-1* and *database-2*

Figure 3.3-7 demonstrates the procedure used to extract the phase and amplitude information from the *database* of sub-models (generated in Algorithm-2), given the input vector x = [a_1 a_2].

The process is identical for both amplitude and phase extraction, and this is how we are able to increase the sample density for $g_{database}$. The database in Figure 3.3-7 can either be *database-1 or database-2,* depending upon the output parameter sought, namely transmission phase or transimission amplitude, respectively. Therefore, if we want transmission phase for the given input x = [a_1 a_2] then, *database-1*($^{i}f_{ann,1}^{forward}$ for i = 1,2 …..9) is utilized, and if transmission amplitude is to be calculated, *database-2* (collection of $^{i}f_{ann,2}^{forward}$ for i = 1,2 …..9), is invoked in Algorithm-3**.**

Algorithm-3: Output Extraction from Database of Forward Sub-Models

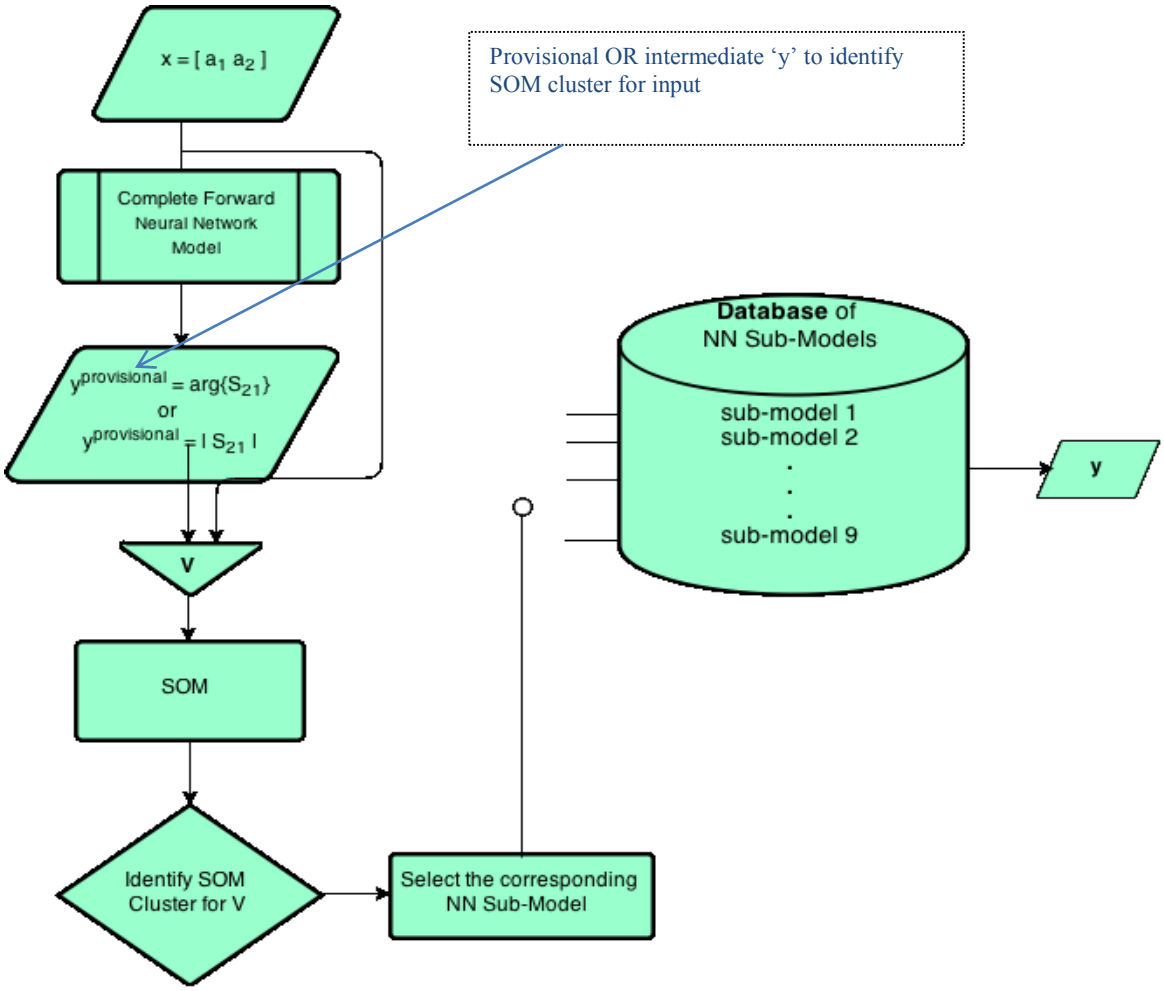


Figure 3.3-7 Algorithm to Extract Transmission Phase and Amplitude Using NN Sub-Models Using SOM Mapping

The above process is repeated for N_s = 250,000 samples and the output is stored in the ***augmented*** $g_{database}$. The augmented sample space is represented in Figure 3.3-8, and Figure 3.3-9 depicts the output amplitude and output phase for each of the inputs in Figure 3.3-8 obtained utilizing Algorithm-3.

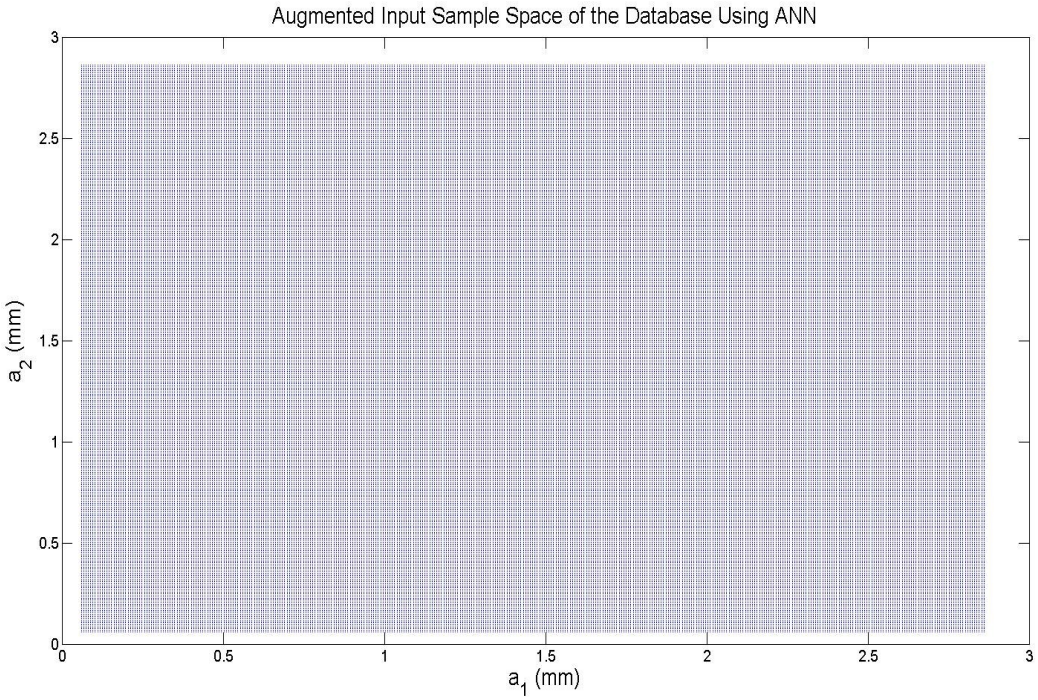


Figure 3.3-8 Input Sample Space of Combined ANNs

The following are the advantages of forward ANN characterization of transmitarray database:

1.) It allows for regeneration of thousands, or even millions, of sample points of transmission coefficient which can be utilized in the later *inversion* process which, otherwise, can be almost impossible due to computational limitations. Thus it allows for a larger concentration or density of sample points available

2.) Given the instantaneous response of the ANNs, it reduces the computational time for the optimization process involved in the Inverse Neural Network modelling since the forward neural network models form an integral part of the *inversion process* to be described in Section 3.4.

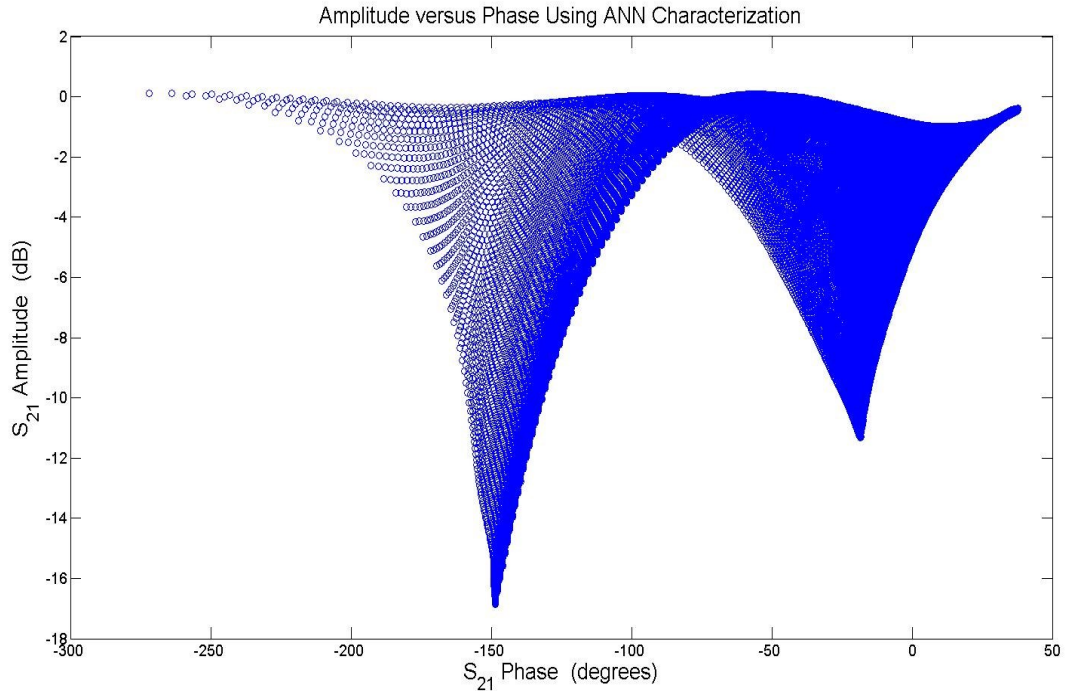


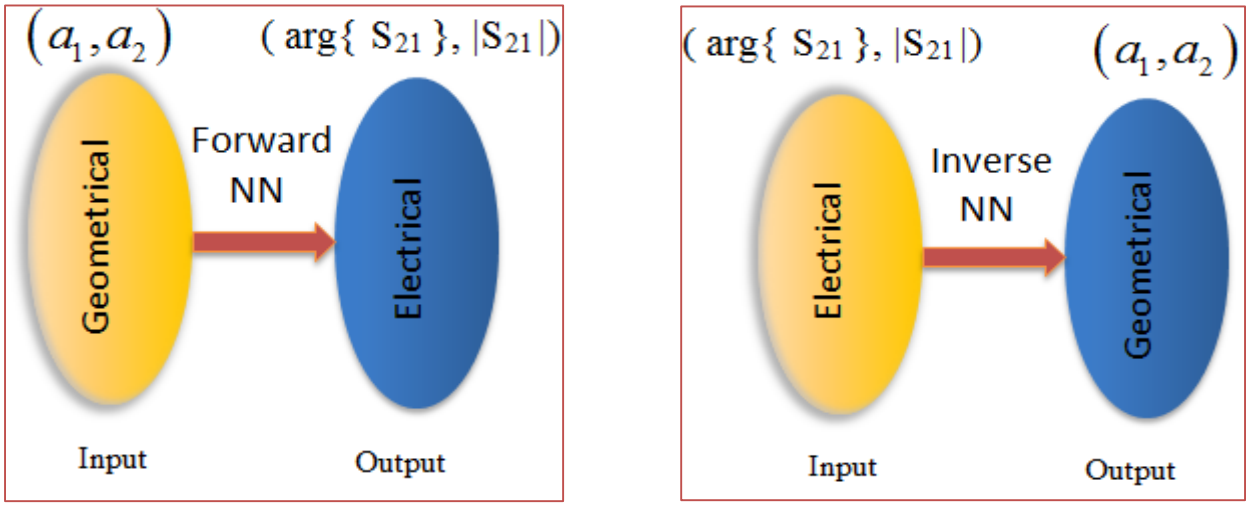
Figure 3.3-8 Output Sample Space of Combined ANNs for Input (a_1, a_2) Pairs Shown in Figure 3.3-3

3.4 INVERSE NERUAL NETWORK MODEL GENERATION

We refer to Figure 3.1-1 again. In Section 3.2 Step-1 was undertaken. Step-3, which is the final step in the process, will be addressed here[19]. In order to accomplish that step, we need to generate *inverse models*. The *inverse problem,* which is to find geometrical parameters from a given a set electrical parameters, is an arduous task if we attempt to use conventional analytical approach. Based on observations of inversion methods mentioned in Section 1.2 and Section 2.6, we have pursued the inverse neural network (INN) approach due to its greater flexibility compared to other approaches. Although the INN approach has been applied to microwave filter [2.5-20], it does not appear to have been used in the design of high-directivity aperture antennas comprised of engineered electromagnetic surface.

[19] Step-2 is discussed in Chapter-5, it is not directly related to database generation or inversion.

In this approach geometrical parameters become inputs, and electrical parameters become outputs. We already have neural network models, the forward neural network models both for transmission amplitude in equation (3.3-1) and phase in equation (3.3-2), mapped from geometrical parameters (a_1, a_2) to electrical parameters (arg{S_{21}} and | S_{21} |), as shown in Figure 3.4-1. But for the inverse problem inputs and outputs are inverted partially or completely, as indicated in equations (2.5-15) through (2.5-21).



(a) (b)

Figure 3.4-1 (a) Forward Neural Network I-O Map, (b) Inverse Neural Network I-O Map

The design process employed in this research only demands the inversion of transmission phase forward neural network model, and doesn't require the amplitude neural network model to be inverted, as indicated in the design process flow diagram Figure 3.4-2. The amplitude neural network model is only required when we need to select the best possible (a_1, a_2) pair for which max transmission amplitude is obtained. This is demonstrated in Figure 3.4-2. We seek the element parameters (a_1, a_2) for a specified phase arg{S_{21}}.

It will be helpful to provide some further information on the need for the inverse model, and why only the phase model needs to be inverted. Consider the forward phase model[20] as given by equation (3.3-1), namely

[20] Note that equation (3.3-1) is the composite forward phase model but in the design process we use the sub-forward phase models as described in Figure 3.3-6

$$\arg\{S_{21}\} = f_{ann,1}^{forward}(a_1, a_2) \tag{3.4-1}$$

Here,

$$x = [\, a_1,\ a_2\,] \tag{3.4-2}$$

is the input to the neural network model

$$y = [\, \arg\{S_{21}\}\,] \tag{3.4-3}$$

represents the output space of the model

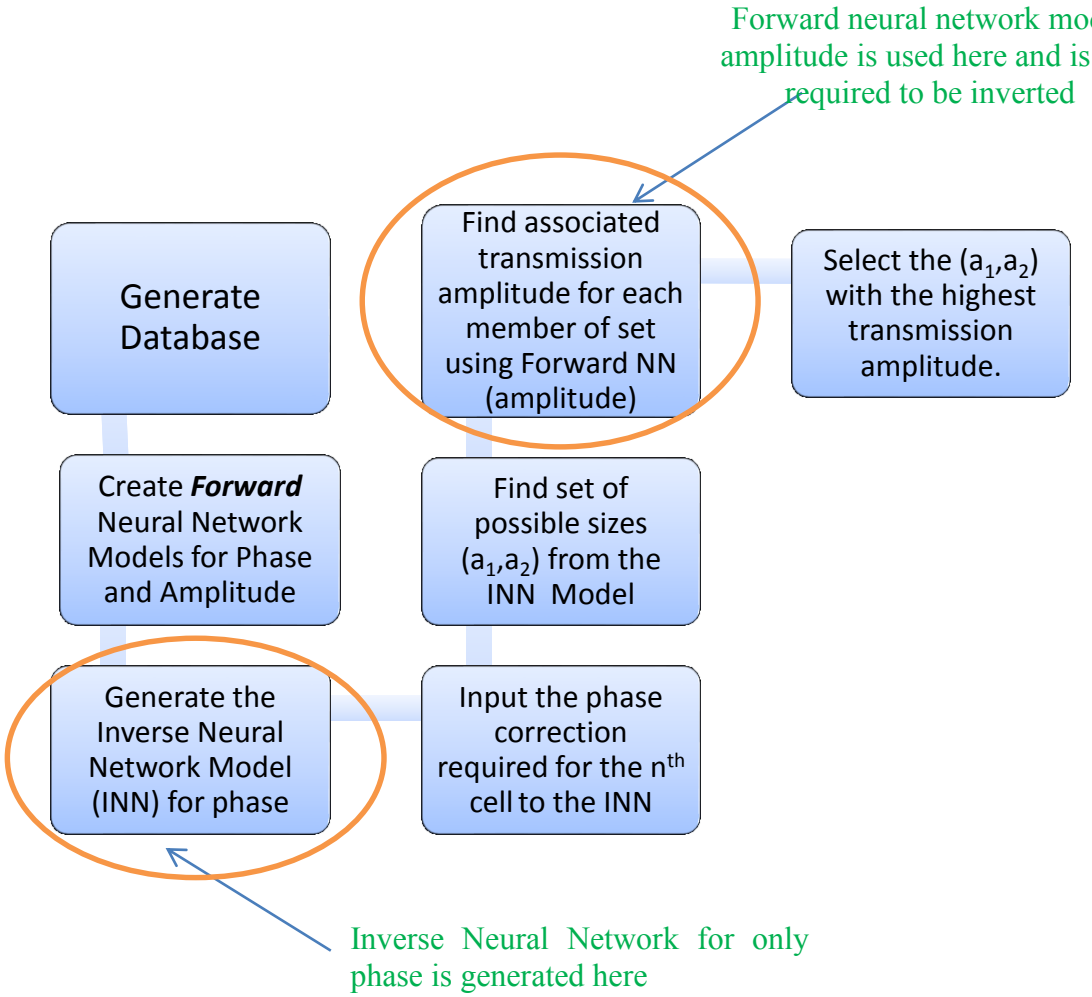


Figure 3.4-2 Transmitarray Design Process Highlighting the Steps where Phase and Amplitude Neural Network Models are Used

$f_{ann,1}^{forward}$ is the map $x \rightarrow y$. Therefore, for the inverse model, we need to invert this map to $y \rightarrow x$.

For convenience, we will let $\bar{x}$ and $\bar{y}$ be the input and output vectors for the inverse model. Thus the inverse map is $\bar{x} \rightarrow \bar{y}$, and the inverse model can be defined as

$$\bar{y} = \bar{f}(\bar{x}) \tag{3.4-4}$$

where,

$$\bar{x} = [\arg\{S_{21}\}] \tag{3.4-5}$$

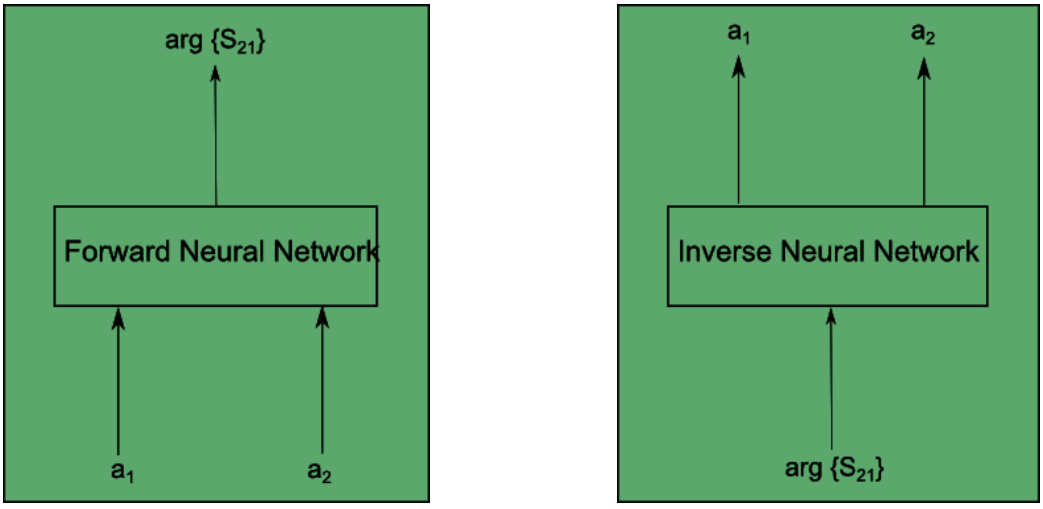
$$\bar{y} = [a_1 \; a_2] \tag{3.4-6}$$

and

$$\bar{f} = \left(f_{ann,1}^{forward} \right)^{-1} \quad or \quad \bar{f} = f_{ann,1}^{inverse} \tag{3.4-7}$$

Therefore, we can represent the INN as

$$(a_1, a_2) = f_{ann,1}^{inverse}(\arg\{S_{21}\}) \tag{3.4-8}$$



(a) (b)

Figure 3.4.3 Neural Network Model Representation for (a) Forward Phase Model, (b) Inverse Phase Model

As described in Section 2.5.3, inverse neural network models, such as in equation (3.4-8) suffer from multivalued solutions due to the fact that the forward models are not one-to-one. Thus, the training of the inverse neural model becomes difficult. Therefore, the complete inverse model's *(direct inverse model's)* sample space is divided into *inverse sub-models,* each of which denotes part of the sample space of the original inverse model.

In order to do that, first of all we need to check whether multivalued solution exist, as is the case for one dimensional input-output relationship in Figure 2.5-5. Using equations (2.5-22) to (2.5-28) it was established that multivalued solutions do exist. Hence, there is a need to create inverse sub-models according to the criteria discussed in what follows.

We need to compute derivatives of outputs versus inputs for the forward model. Derivative information is formulated for only those input and output pairs that have been exchanged (to form the inverse model). In this case, since the two inputs are swapped with the single output, as evident in equations (3.4-5) and (3.4-6), the partial derivative of the output of $f_{ann,1}^{forward}$ (the forward model) with respect to both inputs of $f_{ann,1}^{forward}$ must be determined, and based upon this derivative criteria, the inverse model is segmented.

Since, for the forward model $f_{ann,1}^{forward}$ input vector 'x' contains two elements as shown in equation (3.4-2), whereas the output contains one element as shown in equation (3.4-3), there are two partial derivatives[21] possible according to which segmentation can be performed, namely

Derivative-1 $$\frac{\delta(\arg\{S_{21}\})}{\delta a_1}\Big|_{x=x^{(k)}} \tag{3.4-9}$$

Derivative-2 $$\frac{\delta(\arg\{S_{21}\})}{\delta a_2}\Big|_{x=x^{(k)}} \tag{3.4-10}$$

where k = 1,2,3,……K, with K the number of samples, and $x^{(k)}$ the k^{th} sample.

[21] These partial derivatives are in accordance with the generalized partial derivatives defined in equation (2.5-33)

Recall that in section 3.2-3 we used the ANN characterization of $g_{database}$ to generate 250,000 sample points. These now act as the sample space for the inverse model in equation (3.4-8). In other words, we have, K= 250,000[22].

Accordingly, if for the k[th] sample

$$\frac{\delta(\arg\{S_{21}\})}{\delta a_1} \mid x = x^{(k)} < \delta \tag{3.4-11}$$

in which case the k[th] sample belongs to *sub-model space-1,* as opposed to the case where for the k[th] sample

$$\frac{\delta(\arg\{S_{21}\})}{\delta a_1} \mid x = x^{(k)} > \delta\,, \tag{3.4-12}$$

then current sample belongs to *sub-model space-2.*The choice of δ was discussed in Part D of Section 2.5-3.

This way now the large sample space for the complete inverse model is divided into two inverse sub-models using Derivative-1. To simplify, we can refer to the inverse sub-model space-1 by the term Group-1, and sub-model space-2 by the term Group-2. We next let K_1 and K_2 be the number of samples in Group-1 and Group-2. Samples from each group are then considered separately.

Derivative-2 is used for further segmentation. If, for the k[th] sample in Group-1, we have

$$\frac{\delta(\arg\{S_{21}\})}{\delta a_2} \mid x = x^{(k)} < \delta \tag{3.4-13}$$

then

$$x^{(k)} \in \text{Group-1.1}$$

otherwise, if

[22] The inputs and outputs of augmented $g_{database}$ are flipped to form the inverse vector space as indicated in equations (3.4-4) through (3.4-8)

$$\frac{\delta(\arg\{S_{21}\})}{\delta a_2} \mid x = x^{(k)} > \delta \qquad (3.4\text{-}14)$$

then

$$x^{(k)} \in \text{Group-1.2}$$

where k = 1,2,3,…….K_1

Similarly, Group-2.1 and Group-2.2 are generated by further segmentation of Group-2 using Derivative-2. We then have four *inverse sub-models,* each of which represent partial sample spaces of the inverse model equation (3.4-8) and are used in conjunction.

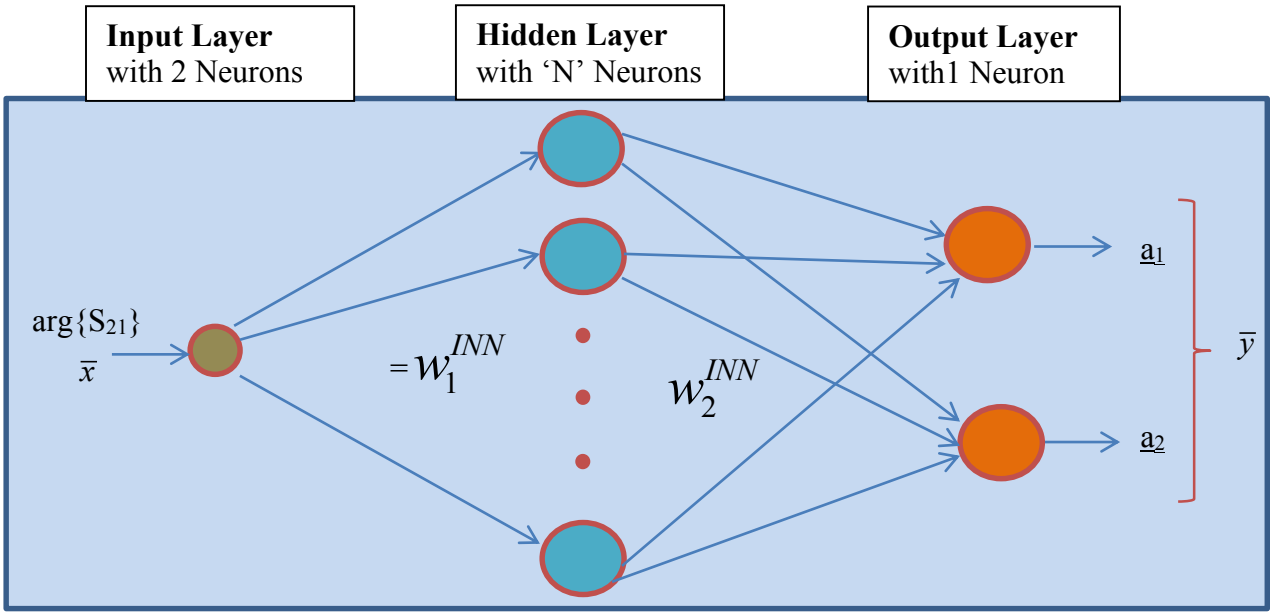


Figure 3.4-4 Multi-layer Perceptron (MLP) Topology Used for Inverse Neural Network(INN) Sub-Model Generation for $f_{ann,1}^{inverse}$

The above describes the procedure for tackling the problem of multivalued solutions by dividing the sample space in to sub-models. We next describe the algorithm (namely Algorithm-4) for implementing this. We then discuss the performance of Algorithm-4, and its shortcomings, and the resolution of these shortcomings through the use of Algorithm-5.

Algorithm 4: To segment the direct inverse model space into inverse sub-models using adjoint of the neural network

Input: entire sample space of inverse *direct inverse model*

Output: Direct inverse model space segmented into *inverse sub-models*

begin

δ → zero by default but small positive value is chosen

Ns → total number of samples also called training samples in direct inverse model

comment: small positive or negative value for δ allows for adjacent sub-model overlap

for iterate through all direct inverse model samples (k ← 1) to (k ← Ns)

comment: compute partial derivatives of output w.r.t. inputs for forward ANN model

comment: $x^{(k)}$ denotes the k^{th} training sample

if $\frac{\delta(\arg\{S_{21}\})}{\delta a_1} | x = x^{(k)} < \delta$ **comment**: computed at k^{th} sample

if $\frac{\delta(\arg\{S_{21}\})}{\delta a_2} | x = x^{(k)} < \delta$

$x^{(k)} \in$ Group-1.1

elseif $\frac{\delta(\arg\{S_{21}\})}{\delta a_2} | x = x^{(k)} > \delta$

$x^{(k)} \in$ Group-1.2

end

elseif $\frac{\delta(\arg\{S_{21}\})}{\delta a_1} | x = x^{(k)} > \delta$

if $\frac{\delta(\arg\{S_{21}\})}{\delta a_2} | x = x^{(k)} < \delta$

$x^{(k)} \in$ Group-2.1

elseif $\frac{\delta(\arg\{S_{21}\})}{\delta a_2} | x = x^{(k)} > \delta$

$x^{(k)} \in$ Group-2.2

end

end

end

return Groups

After iterating through all the 250,000 samples and feeding these samples to Algorithm-4, we obtain 4 inverse sub-models. Then, each sub-model is trained using the Levenberg-Marquadt training algorithm [2.5-14] according to the process flow described in Figure 2.5-3, and equations (2.5-5) to (2.5-10) in Section 2.5 are used for feedforward computations. The sigmoid activation function in equation 2.5-6 is utilized in the process. During the training, weight vector $w^{INN} = [w_1^{INN}, w_2^{INN}]$ shown in Figure 3.4-4 is updated using above mentioned optimization algorithm. To summarize the inverse neural network training and testing procedure we define some terminology here. We have created 4 inverse sub-models, but will illustrate the training and testing conventions utilized in our model generation for just one sub-model; it would be identical for the other 3 sub-models. Let's consider an inverse sub-model corresponding to Group-1.1

$N_{1.1}$ → represents the total number of data samples in a Group-1.1 inverse sub-model sample space, in this case $N_{1.1} = 96{,}000$

I_s^{INN} → represents the index set for all the samples in the Group-1.1, $I_s^{INN} =$ [1 2 … $N_{1.1}$]

I_r^{INN} → represents the index set of training data as well as validation data

I_e^{INN} → represents the index set of training data

Training and testing datasets are formed by dividing the complete data set (for the sub-model in question) by a given proportion. In our case we split I_s^{INN} into two sets with a 75-25% split such that I_r^{INN} contains 75% of the samples for training, whereas I_e^{INN} has a total share of 25% of the samples. We define training error, which quantifies the error for the current set of weights and is calculated again after every update of the weight vector. Now, using the generalized equation for quantifying training error and considering the inverse model defined in equations (3.4-4) through (3.4-8), where

$$E_{training}^{INN}(\mathrm{w}) = \frac{1}{2}\left[\sum_{n \in I_r^{INN}} \sum_{j=1}^{2} (\bar{y}_j(\bar{\mathrm{x}}_n, \mathrm{w}^{INN}) - \bar{y}'_{n,j})^2 \right]^{\frac{1}{2}} \qquad (3.4\text{-}15)$$

where

- n = 1,2,3…N_1 (sample index)
- $\bar{x}_n$ = n^{th} input vector sample such that, $\bar{x}_n$ = [arg $\{S_{21}\}$]^n
- $\bar{y}_j(\bar{x}_n, w)$ = j^{th} component of the output vector $\bar{y}$ computed for the n^{th} sample and current weight vector $\bar{y}$ = [a_1, a_2]
- $\bar{y}'_{n,j}$ = actual output from the dataset used for training
- j = index for the output vector, j = 1, 2
- w^{INN} = weight vector associated with each synapse (connection between two neurons)

Once the training error $E_{training}^{INN}$ is below the prescribed limit, the training is terminated. The next step is neural network testing, and we utilize the dataset ascribed specifically to testing using the indices in the testing index set I_e. Average test error is quantified as:

$$^{average}E_{test}^{INN} = \frac{1}{\overline{\overline{I_e^{INN}}}\,2}\left[\sum_{n\in I_e}\left(\frac{(\bar{y}_j(\bar{x}_n, w) - \bar{y}'_{n,j})}{\bar{y}'_{j,max} - \bar{y}'_{j,min}}\right)^2\right]^{\frac{1}{2}} \tag{3.4-16}$$

where

$\overline{\overline{I_e^{INN}}}$ = size of index set I_e^{INN}

Similarly, the worst case test error is calculated as

$$^{worst\,case}E_{test}^{INN} = \max_{n\in I_e^{INN}} \max_{j=1}^{2} \frac{(\bar{y}_j(\bar{x}_n, w) - \bar{y}'_{n,j})}{\bar{y}'_{j,max} - \bar{y}'_{j,min}} \tag{3.4-17}$$

Having computed training and test errors for all four of the inverse sub-models (INN sub-models), it was observed that the average training error is still very high (around 13 %). In Figure 3.4-5 and Figure 3.4-6, linear regression plots are presented demonstrating the erroneous training of the inverse models for Group-1.1 using the test data for this group; the plots basically depict the accuracy of the inverse neural network training process in terms of linear regression which should be unit ideally. On the vertical axis we have the actual known values of the inverse neural network outputs while on the horizontal axis we have output from the neural network for

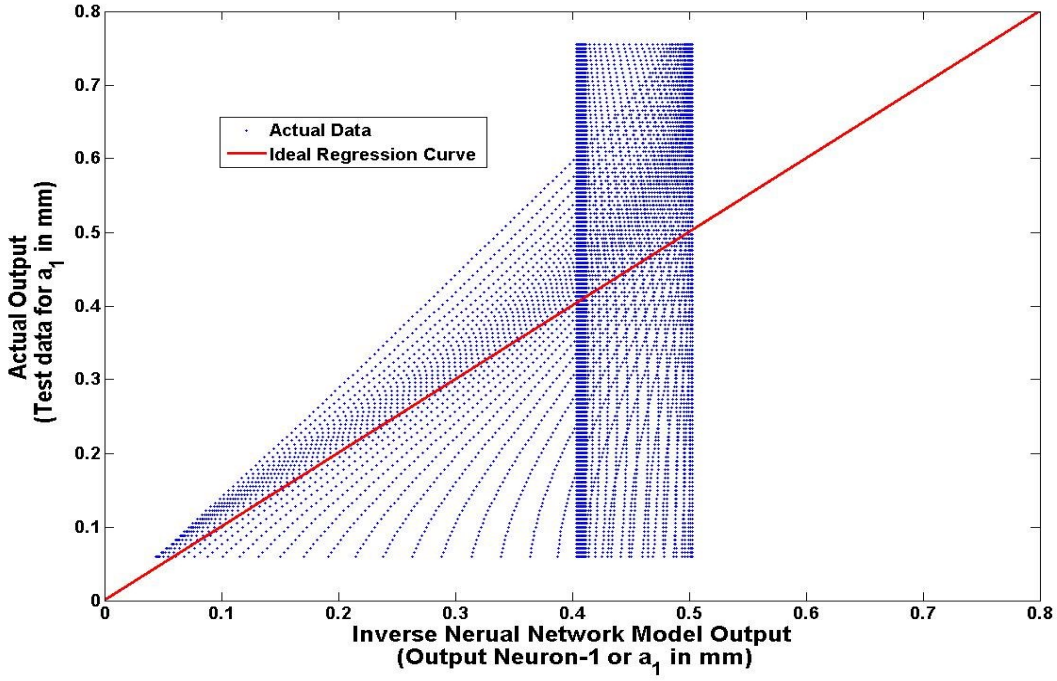


Figure 3.4-5 Regression Plot for Inverse Neural Network Sub-Model Group 1.1 Testing for One of the Outputs 'a_1' such that $\bar{y}=[a_1\ a_2]$

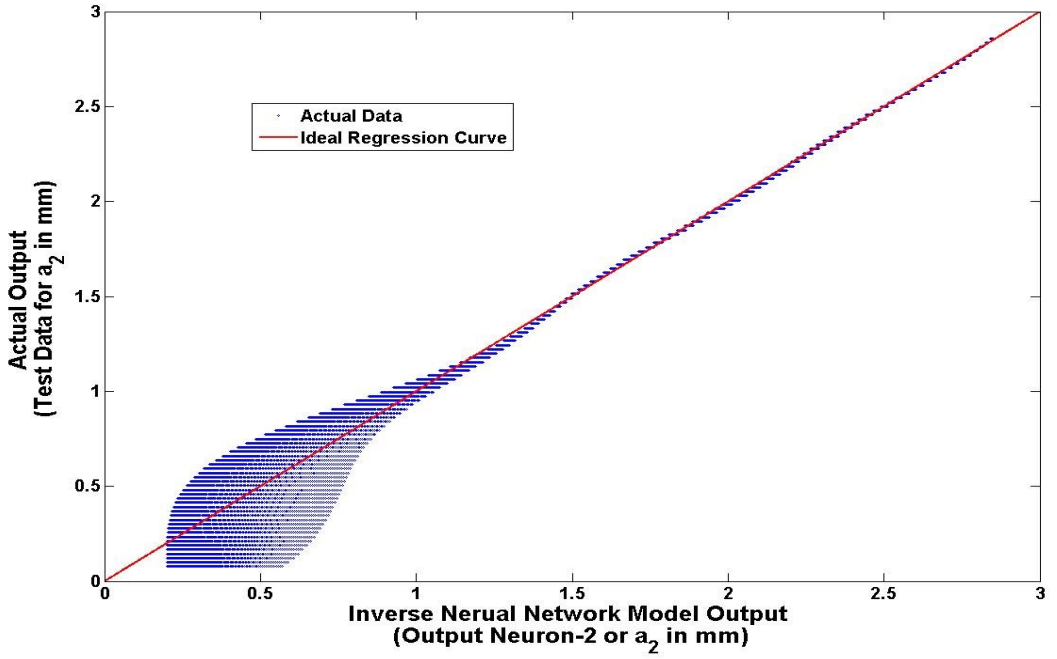


Figure 3.4-6 Regression Plot for Inverse Neural Network Sub-Model Group 1.1 Testing for One of the Outputs 'a_2' such that $\bar{y}=[a_1\ a_2]$

the test data inputs. Ideally all the points must lie on the y=x line shown in red but that is not the case as we can witness; instead the points are scattered and regression is lower than unity. For other inverse model groups (Group-1.2, Group-2.1 and Group-2.2) similar observations were made. To further confirm the *inaccuracy* of the 4 sub-models, training and test error plots are presented in Figures 3.4-7 and 3.4-8. In order to overcome these shortcomings, the situation demands further refining of the inverse model generation strategy. This is achieved via Algorithm-5, which is discussed next.

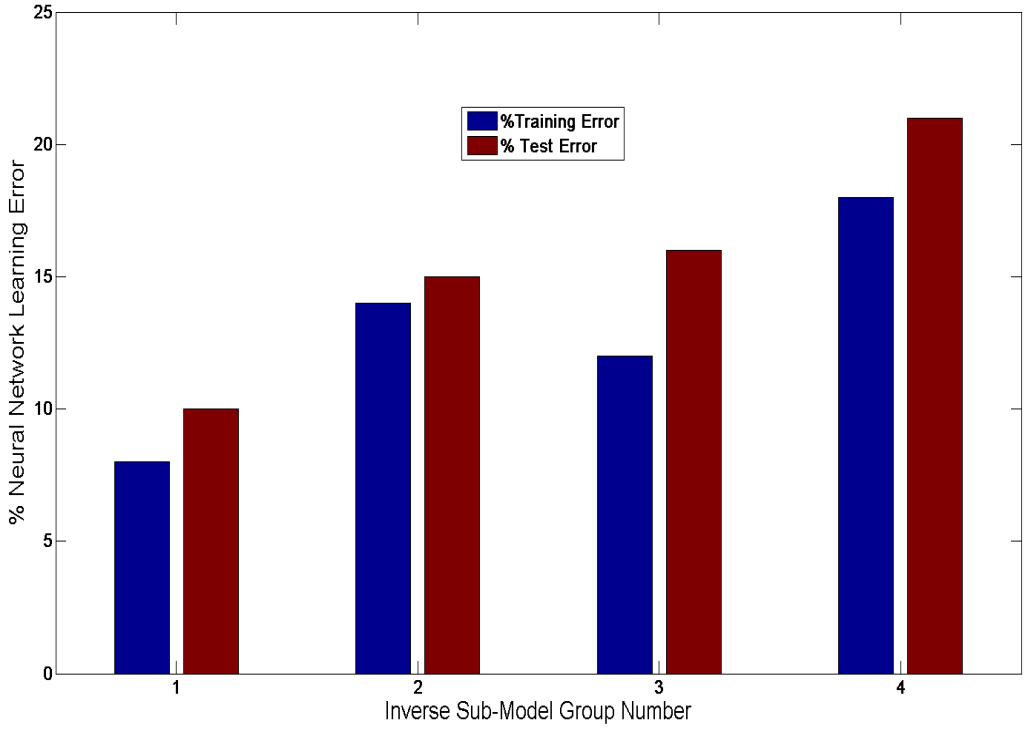


Figure 3.4-7 Average Training and Test Error for each of the 4 Groups After Slope Based Segmentation of the Direct Inverse Model Sample Space

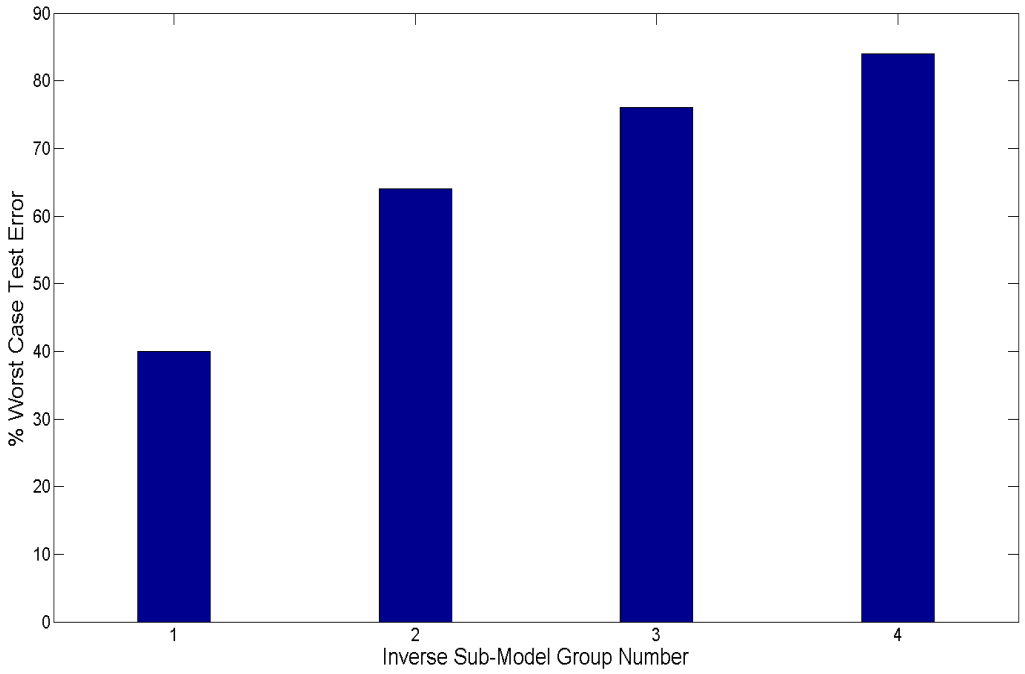


Figure 3.4-8 Worst Case Test Error for each of the 4 Groups After Slope Based Segmentation of the Direct Inverse Model Sample Space

Similar to what was suggested in [2.5-20], we segment the direct inverse model sample space (using Algorithm-5) into several segments (sub-segments) disregarding the derivative criteria (slope based segmentation). We then apply Algorithm-4 to each sub-segment. There are 4 groups for each sub-segment. This reduces the size of the sample space for each inverse sub-model significantly, and limits the training error tremendously. The details of Algorithm-5 are shown in the form of pseudo-code and a flow chart in Figure 3.4-9.

After implementing Algorthm-4 in conjunction with Algorithm-5, 2250 INN sub-models were generated. Figure 3.4-10 shows the *test error* **(**error measured once neural network learning is finished to ascertain accuracy of training**),** calculated for each of the sub-models. The sub-models are represented as:

Algorithm-5: Further Segmentation and Inverse-Sub Model Generation

Input: entire sample space of inverse *direct inverse model*
Output: Direct inverse model space segmented into *inverse sub-models* and stored in an ***INN Repository***
begin

Ns → total number of samples also called training samples in direct inverse model where inputs and outputs are in accordance with equation 3.4-8
comment: accuracy of an inverse model in each of the steps stands for the training and testing error
comment: training is undertaken at each step when number of segments increase for each segment individually and thus, we have a corresponding inverse sub-model for that segment

if $f_{ann,1}^{inverse}$ (direct inverse neural network model) is accurate
comment: terminate (store the direct inverse model in the INN repository)
else
comment: start segmentation of $f_{ann,1}^{inverse}$ model's sample space
comment: segment direct model into 'S' segments disregarding *slope* criteria

if sub-models are accurate
comment: terminate (store 'S' segments in INN repository)
terminate
else
comment: check for non-unique solutions using eqs. 2.5-22 to 2.5-28

if non-unique solutions exist
comment: feed each of 'S' segments to *Algorithm 1* iteratively which will *return* 4 sub-segments based upon the slope criteria

convergence = false
while **convergence = false** **do**
for **i = 1 to S**
Function Call: Algorithm-1 (segment number 'i')
comment: Algorithm-1 will return 4 sub-segments or sub-models for i^{th} segment

i ← i +1
comment: there will be S*4 sub-models in the INN repository at this stage

if all the sub-models are accurate
convergence = true
terminate (store the sub-models in the *INN repository***)**
else
convergence = false
comment: Increase the number of segments from S to 2*S i.e. reduce the number of samples in each segment by half
S ← 2*S
end
end

```
        else
            comment: if there are no non-unique solutions then increase the number of segments of direct
            inverse model by 2 twice.

            convergence = false
            while convergence = false do

                S ← 2*S
                if sub-models are accurate
                    terminate (process finished and store all the sub-models in the repository)
                end
            end

        end

    end
  end
return INN Sub-Models
```

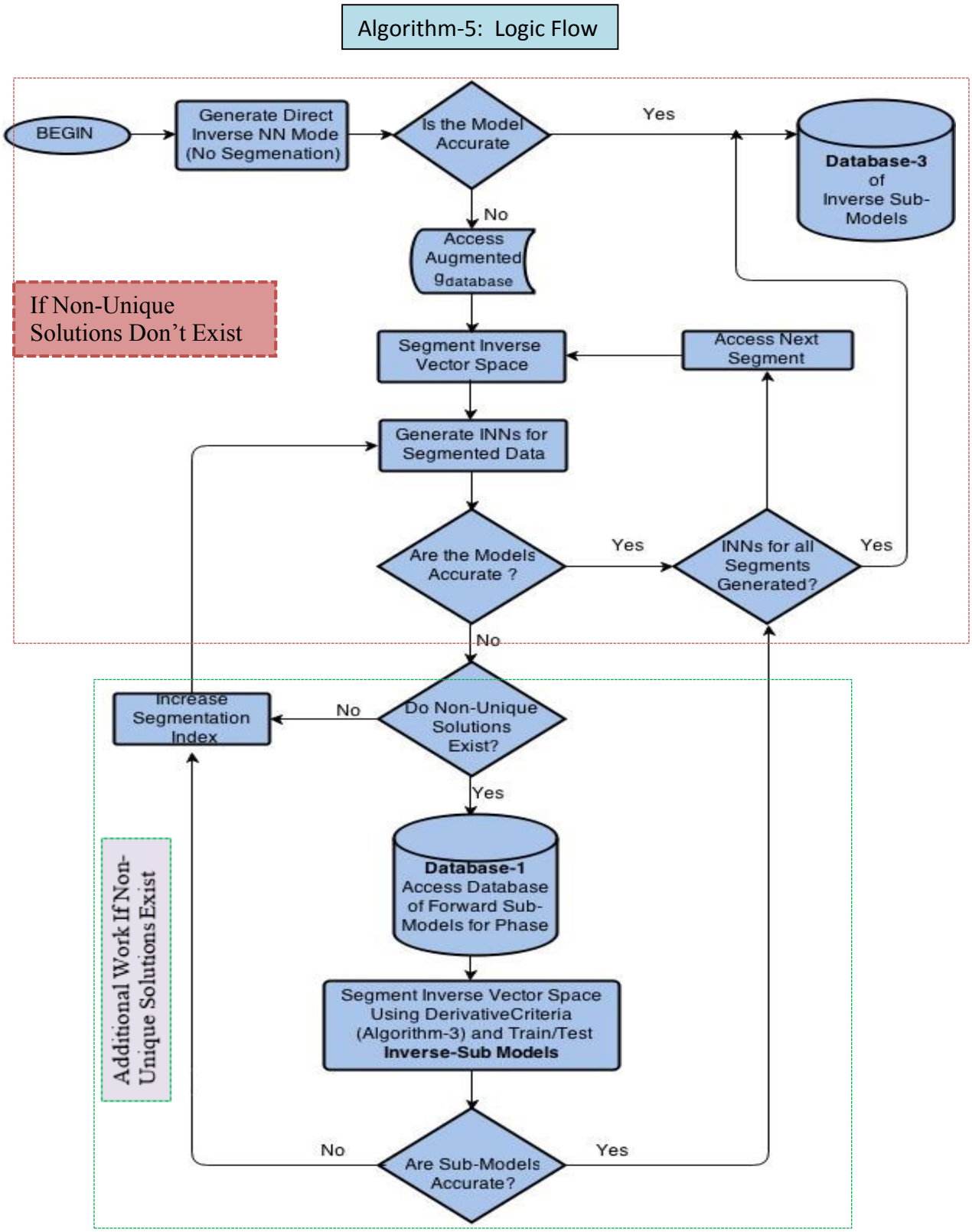


Figure 3.4-9 Logic Flow Diagram for Algorithm-5[23]

[23] Algorithm-5 is adapted from [2.5-20]

$$\bar{y} = {}^{i}f_{ann}^{inverse}(\bar{x}) \qquad \forall i = 1,2,3....m,.....M \ (M = 2250) \tag{3.4-18}$$

where, $\bar{y} = [a_1 \ a_2]$ and $\bar{x} = \arg\{S_{21}\}$

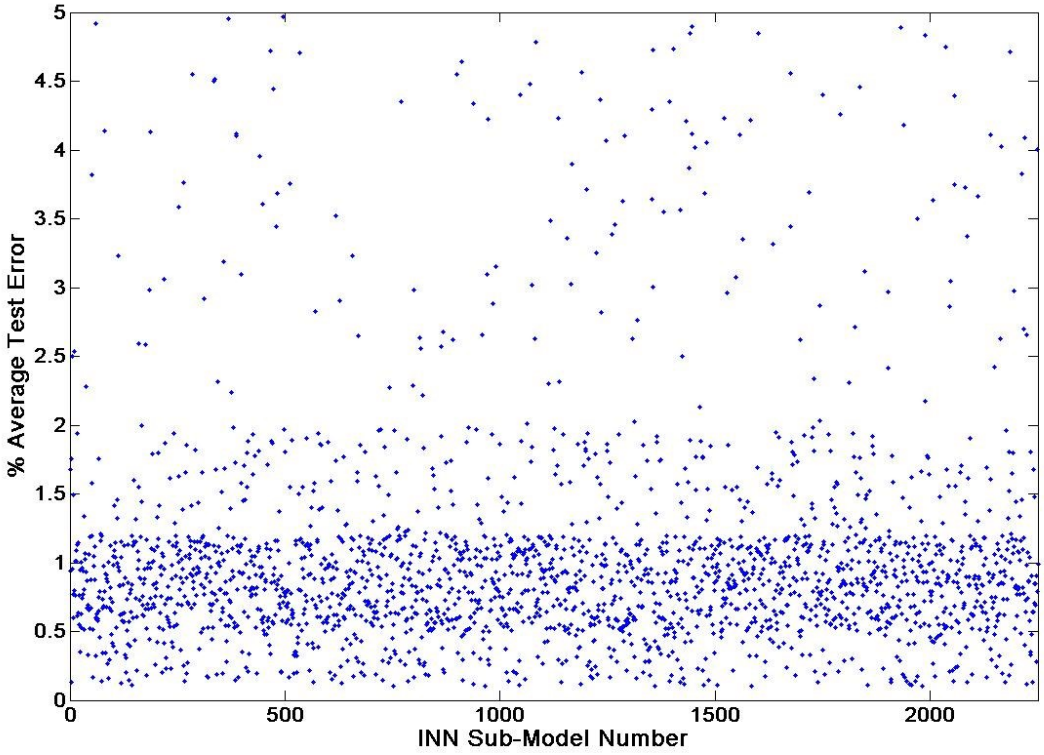


Figure 3.4-10 Percent Test Error versus the INN Sub-Model Number

It was observed (see Figure 3.4-10) that most of the INN-Sub Models had an average test error under 1.2 %, which is an acceptable number. Also, the maximum error that resulted was 4.97%, for model number 496. A histogram of the test error frequency distribution among the inverse sub-models is plotted in Figure 3.4-10, showing the average test error of most of the models well within the 2% range. Therefore, it is established that implementation of Algorithm-5 along with Algorithm-4 for inverse sub-model generation was successful, resulting in 2250 sub-models

(stored in *database-3*). The next hurdle is how to seek output from Database-3. This will be discussed in Section 3.5[24].

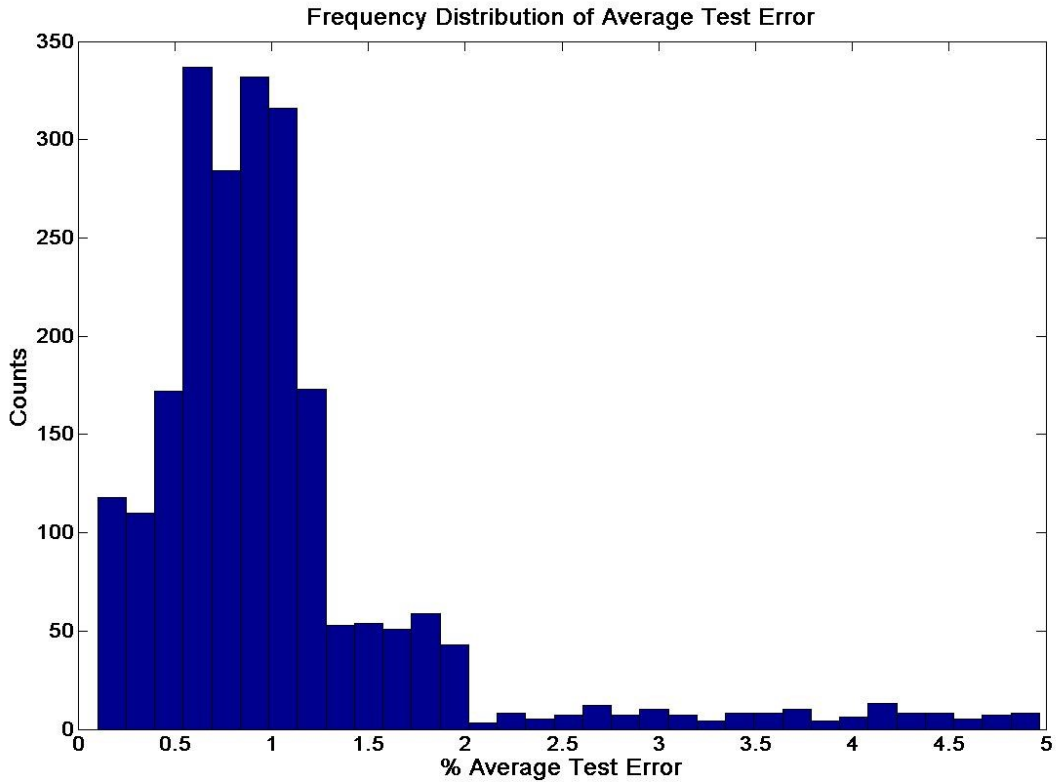


Figure 3.4-11 Histogram of Average Test Error for Inverse Sub-Models

3.5 OBTAINING OUTPUT FROM THE INVERSE SUB-MODEL DATABASE

The next step in Inverse Neural Network implementation for a general microwave modelling is the **derivation of output** from the database of inverse sub-models generated in Section 3.4. Although it might seem obvious to just feed the desired phase to the database-3 and seek the corresponding dimensions for the transmitarray element, but 'which inverse model to choose from the database?' is a conundrum. To resolve this, Algorithm 6 was used. If we look at Figure

[24] It is appropriate to restate where things stand: Full-wave simulation was used to produce 4893 points in the initial database. The forward ANN process used this to produce an augmented database of 250,000 points. Algorithm-5 and Algorithm-4 resulted in 2250 INN sub-models to invert this augmented database. Database-1 is the collection of forward sub ANN models for $\arg\{S_{21}\}$. Database-2 is the collection of forward sub ANN models for $|S_{21}|$. Database-3 is that for the overall INN process, consisting of 2250 sub-model. Section 3.5 deals with the procedure to actually access and use database-3.

3.4-2, which delineates the entire methodology of transmitarray design, and divide the whole process into Sections, we can obtain a better topological awareness as to where each Algorithm (described before and to be presented in next Sections) is used as shown in Figure 3.5-1.

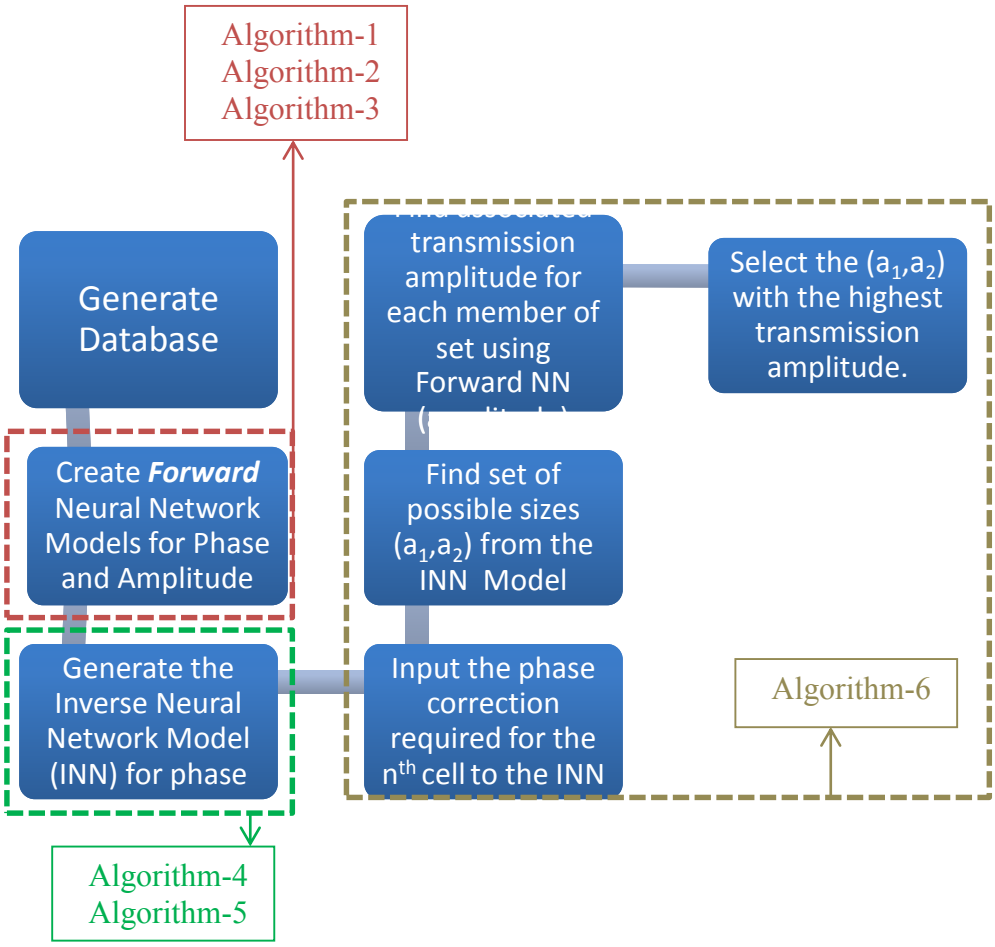


Figure 3.5-1 Transmitarray Design Process Flow Discretized into Sections According to the Algorithms Used (Annotated version of Figure 3.4-2)

Now that INN model is complete we could proceed to design the transmitarray based upon Algorithm-6, which we discuss next. However, we only use it for design in Chapter-5. Here we will check the accuracy of the INN modelling. To do this we will consider an arbitrarily selected set of *transmission phase values*, and will proceed to derive the corresponding dimensions for associated elements using Algorithm-6. The algorithm is first listed in pseudo-code and then in flowchart form. After that its implementation is discussed. It is verified in Section 3.6.

Algorithm-6: Output Derivation from the Database of Inverse Sub-Models (Database-3)

comment: to determine patch dimensions for n^{th} element for specified transmission phase and maximum transmission amplitude

comment: Database-1 → database of forward neural network sub-models for **phase** generated in Algorithm-3
Database-2 → database of forward neural network sub-models for **amplitude** generated in Algorithm-3
Database-3 → database of inverse neural network sub-models for generated in Algorithm-5

Input: Transmission Phase ($\arg\{S_{21}\}^n$) required at the center of n^{th} element on the tranmistarray aperture

Output: Patch Dimensions [a_1, a_2]^n for the current element that result in the given transmission phase

begin

$\bar{x}$ - input for inverse model , and $\bar{y}$ - output for inverse model

x - input for forward model , and **y** - output for forward model

comment: access Database-3

- *feed the desired phase $\bar{x}^n = arg\{S_{21}\}^n$ at the n^{th} element to all the inverse sub-models stored in Database-3 (let the number of models be 'M' and therefore, M = 2250)*
- *store the output (n^{th} element size) from each of the 'M' inverse sub-models*

$$(\bar{y}_i^n = [a_1, a_2]_i^n) = {}^i f_{ann}^{inverse}(\bar{x}^n = \arg\{S_{21}\}^n)$$
$$\forall i = 1, 2, 3 \ldots\ldots M$$

comment: now that we have 'M' possible values for the n^{th} element size, we need to ascertain which outputs are closer to the actual output by the process of elimination based on an Error function.

Comment: 'M' **outputs** are considered as **inputs** to the forward neural network model.

- *feed the 'M' possible outputs to the 'forward neural network model for phase' following the output extraction procedure described in Algorithm-3 from the forward sub-models using SOM utilizing database-1*

$$(y_i = \arg\{S_{21}\}_i) = {}^{SOM} f_{ann,1}^{forward}(\bar{y}_i^n = [a_1, a_2]_i^n) \qquad \forall i = 1, 2 \ldots M$$

- *Compute the error 'E_i' associated with each of the' M' inverse models*

$$E_i = \frac{1}{2}\left\{ \frac{[(y_i = \arg\{S_{21}\}_i) - (\bar{x}^n = \arg\{S_{21}\}^n)]^2}{[\max(\bar{x}) - \min(\bar{x})]^2} \right\}^{\frac{1}{2}} \qquad \forall i = 1, 2 \ldots M$$

comment : ideally we would select the m^{th} inverse sub-model given that 'E_i' is minimum for i = m but we also need to consider the associated transmission amplitude with the n^{th} element's dimensions

comment: hence, we select top 20 models with least error E_i and compute the transmission amplitudes associated with each one of them using database-2

comment: let 'P' represents the index set for 20 inverse sub-models selected by virtue of least E_i such that

- $$i \in P \equiv E_i \text{ is amongst least 20 error values}$$

$$(|S_{21}|_p) = {}^{SOM}f^{forward}_{ann,2}(\bar{y}^n_p = [a_1, a_2]^n_p)$$
$$\forall \quad p \in P$$

comment: select top10 models with the maximum transmission amplitude from set 'P' and store their indices in set 'Q' such that

- $$Q \subseteq P$$

$$Q \cap P := \{q: \ |S_{21}|_q \text{ is amongst top 10 transmission amplitude values}\}$$

comment: for all the inverse sub-models in set Q with index q (${}^{q}\mathrm{f}^{inverse}_{ann}$) such that $q \in Q$, the training range is defined as the range of each element of the output vector of the q^{th} model i.e. if for the q^{th} model, inputs are mapped to the outputs as

$$(\bar{y}_q = [a_1, a_2]_q) = {}^{i}\mathrm{f}^{inverse}_{ann}(\bar{x}_q = \arg\{\mathrm{S}_{21}\}_q)$$
$$\forall q \in Q$$

comment: then, training range is expressed for each element of the output vector as;

$$\max({}^{q}a_1) - \min({}^{q}a_1) \quad \&$$
$$\max({}^{q}a_2) - \min({}^{q}a_2)$$

comment: for an inverse model, the output is considered valid if and only if it lies within its training range

comment: determine index for the inverse sub-model in set 'Q' for which the output dimensions ($\bar{y}^n_q = [a_1, a_2]^n_q$) lie within the training range of the inverse sub-model ${}^{q}\mathrm{f}^{inverse}_{ann}$ for $q \in Q$

- *while j <= #(Q) [cardinality of Q]*
  if (${}^{Q_j}_{n}a_1$ - max(${}^{1}\bar{y}_{Q_j} = {}^{Q_j}a_1$))<0
  if (${}^{Q_j}_{n}a_1$ - min(${}^{1}\bar{y}_{Q_j} = {}^{Q_j}a_1$))> 0

  comment: select the output $\bar{y}^n_j = [a_1, a_2]^n_j$ from the inverse model ${}^{j}\mathrm{f}^{inverse}_{ann}$ to be the final output

  break
  end
  end
  j++
  end

comment: this way inverse sub-model with index 'q = Q_j' is selected which satisfies all the criterion above and its output is used to determine geometry of the n^{th} element of transmitarray

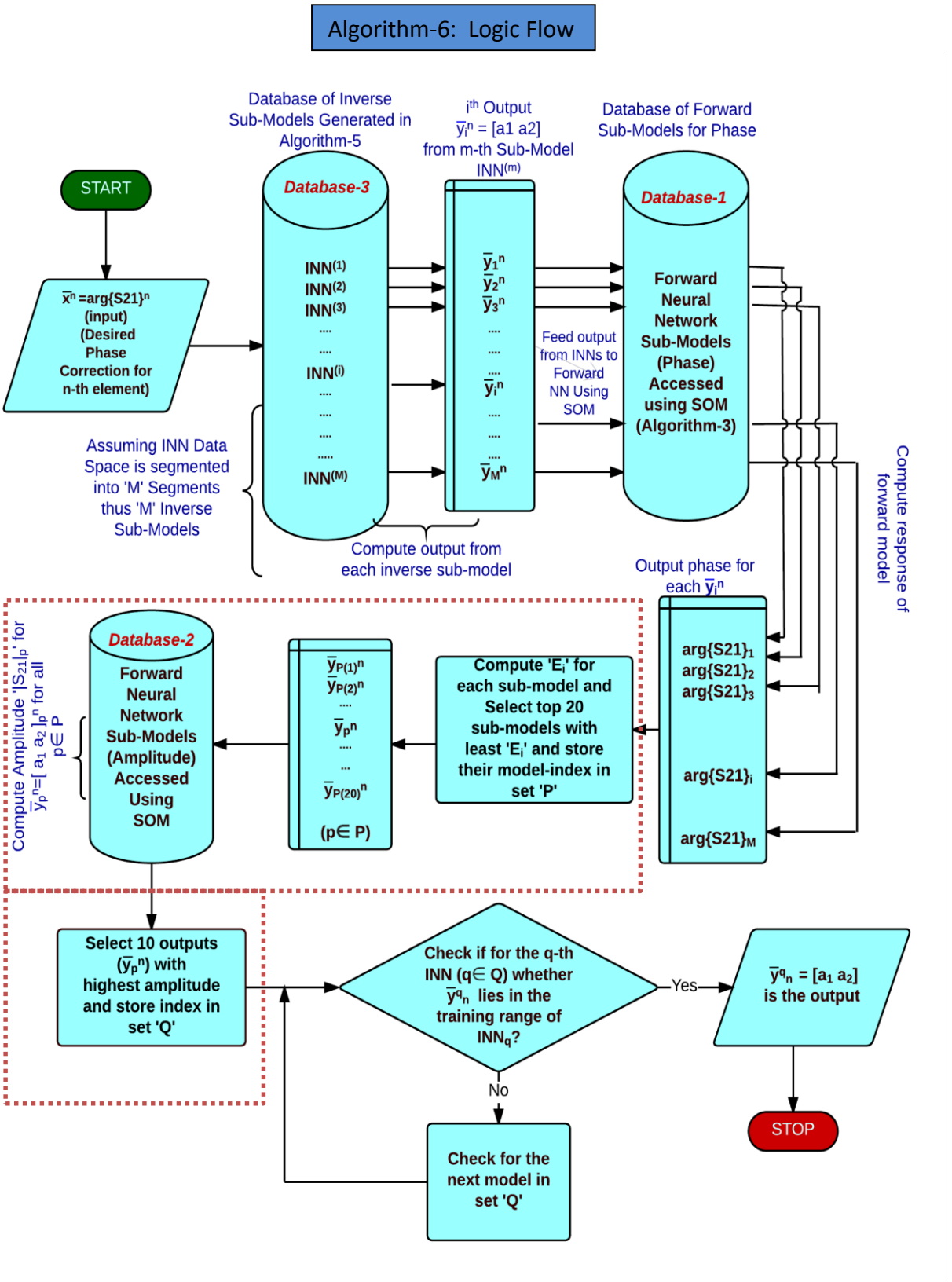


Figure 3.5-2 Algorithm to Find the Dimensions of n^{th} Element of the Transmitarray for a Given Transmission Phase. The Portion Shown in Dashed Lines Differs from that Suggested by other Authors.

We will divide Alogirthm-6 into sections, have a detailed look at the role that each part plays in the overall logic flow. Figure 3.5-3 shows the ***Stage-1*** in which we feed the input, ($\bar{x}^n$ = arg{S21}^{n}, desired transmission phase at the nth cell/ element) to the Database-3 of inverse sub-models and subsequently, we have to identify which inverse sub-model's output is correct? Therefore, we in turn feed each output vector ($\bar{y}_i^n = [a_1\ a_2]_i^n$), the output from each of the M inverse sub-models (where M=2250 the number of sub-models in Database-3) to the forward neural network. Recall that we created databases of forward neural networks using self organizing maps as explained in Algorithm-2; namely Database-1 (phase) and Database-2 (amplitude) which store forward neural network sub-models. Also note that to extract the output from database-1 and database-2, we need to implement Algorithm-3. Thus, we feed each output vector to Algorithm-3 and seek the corresponding phase (arg{S21}$_i$ where i=1,2,3….M) from database-1. To determine which of the M inverse sub-models model gives the best output, we will move to the ***Stage-2*** of Algorithm-6, depicted in Figure 3.5-4.

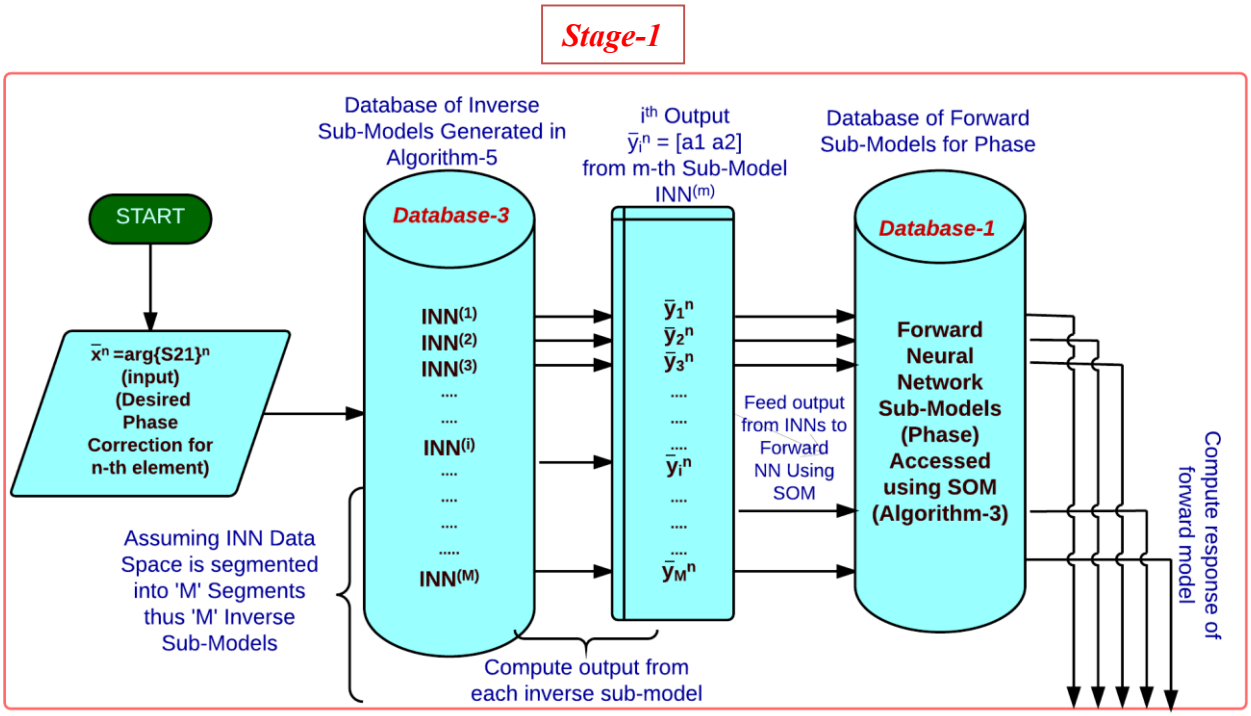


Figure 3.5-3 Stage-1 of Algorithm-6 where we Seek M Possible Outputs; One from each the M Inverse Sub-Models in Database-3

In Stage-2, the output from database-1, the M phase values, ($\arg\{S_{21}\}_i \ \forall$ i= 1,2,3…..M) are considered, and compared with the original phase value, $\bar{x}^n = \arg\{S_{21}\}^n$, and error '$E_i$' is computed. Then the 20 models with the least error 'E_i' are chosen and their index 'i' is stored in the set 'P'. As discussed in Section 3.2 and illustrated in Figure 3.2-8, ideally we would choose dimensions $[a_1\ a_2]^n$ for the n^{th}-element for which unity transmission amplitude is achieved. Since it's not possible to achieve unity amplitude for all the transmission phase values[25], we would want the highest transmission amplitude associated with each phase value. This is the essence of the next step, where we feed the outputs from 20 inverse sub-models, $\bar{y}_i^p = [a_1\ a_2]_i^p\ (\ \forall\ p \in P\ )$, to Database-2 that stores forward neural network sub-models for amplitude $|S_{21}|$. Using Algorithm-3, the amplitude associated with each of the 20 outputs is computed. We then reach the final stage of Algorithm-6 which is *Stage-3*, shown in Figure 3.5-5.

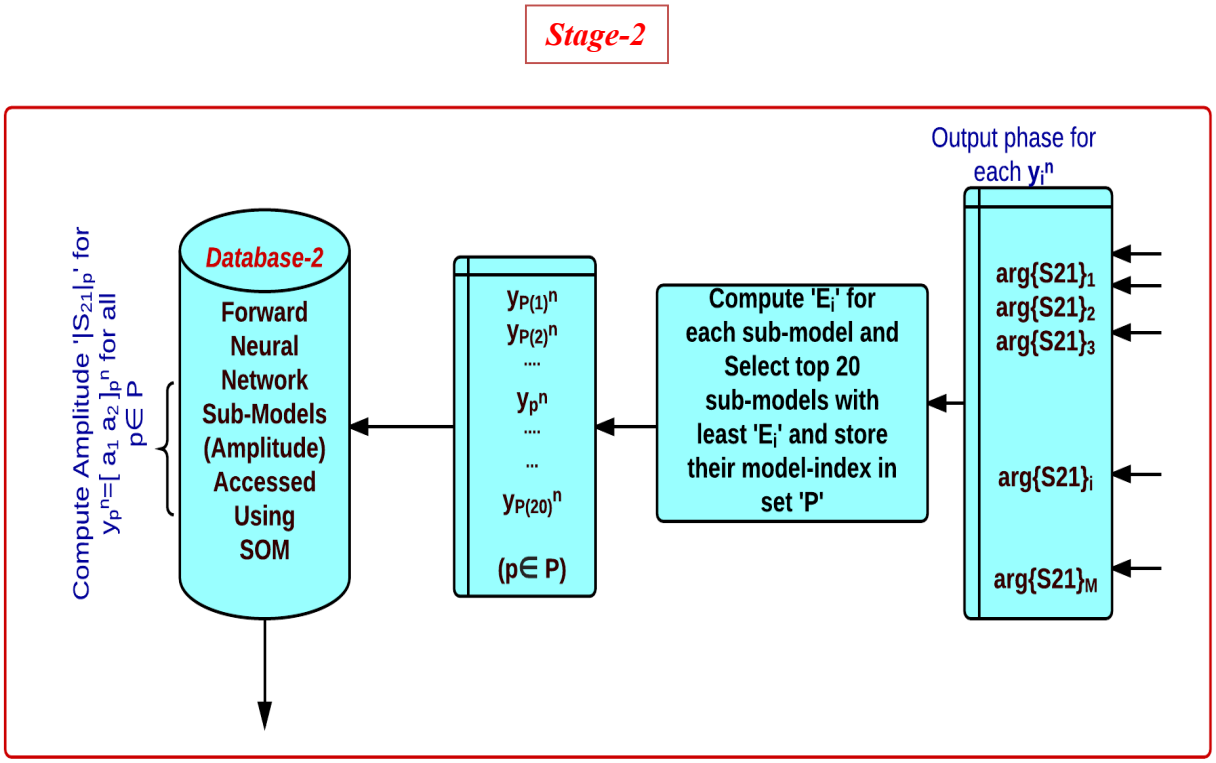


Figure 3.5-4 Stage-2 Of Algorithm-6 Where We Compute Error Associated With The Output Of Each Inverse Sub-Model And Also The Transmission Amplitude Associated With The Selected Models

[25] A physical limitation of the selected transmitarray element geometry, as previously pointed out in Figure 3.2-8

Only 10 models are retained, namely those with the highest amplitudes from among the 20 models, and their index is stored in set 'Q'. After filtering the outputs for best transmission amplitude, we also need to check whether the outputs from each of these inverse sub-models, $\bar{y}_i^q = [\mathrm{a}_1\ \mathrm{a}_2]_i^q\ (\ \forall\ q \in Q\ )$, lie within the range of outputs for that model. In order to check this for the q^{th} inverse sub-model, we refer to the training/ learning data used to train the q^{th} inverse sub-model. We scrutinize both the elements of the q^{th} output vector a_1^q as well as a_2^q; both of these must lie within the training range (details being shown in the pseudo-code of Algorithm-6). Any of the inverse sub-model outputs meeting this criterion can be considered as the final dimension to be used for n^{th} element. This is where this method contrasts with traditional INN algorithms (eg. [2.5-20]) as well as the algorithms in the present literature (discussed in Section 2.6) as it allows freedom to select from multiple possible $[a_1\ a_2]$ values.

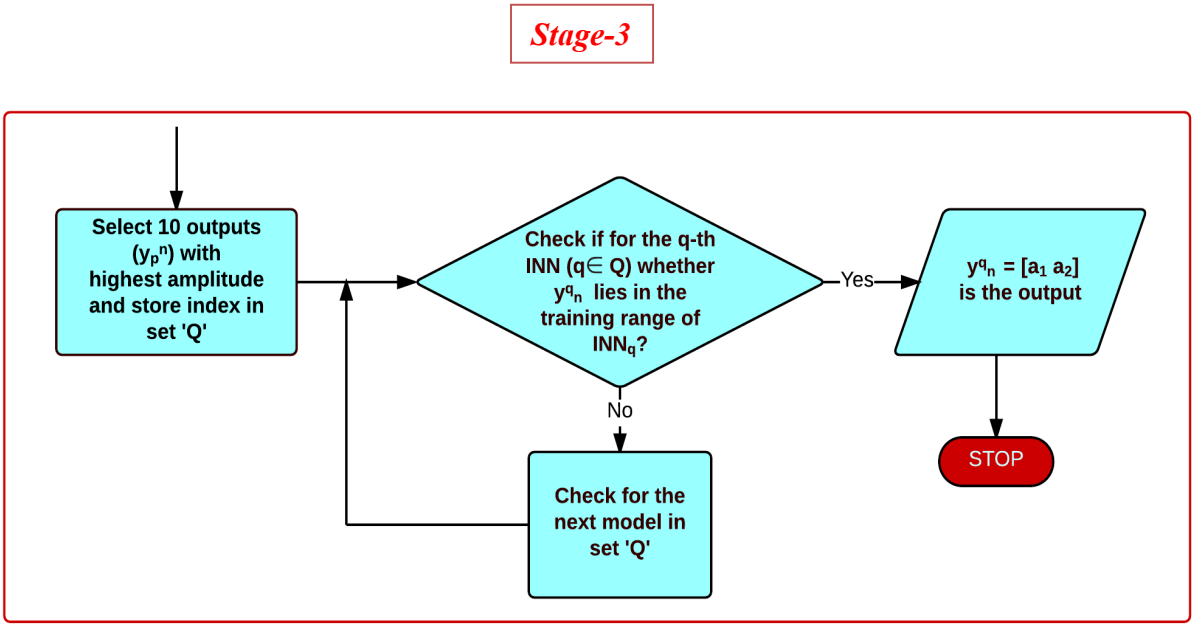


Figure 3.5-5 Stage-3 Of Algorithm-6 Where We Check Whether A Selected Model's Outputs Satisfy The Proximity Criteria Based Upon Triaining Range

This is a very important aspect as far as inverting databases for the design of transmitarrays and other aperture antennas based on engineered surfaces. In the present case being used as an example, the 10 sets of $[a_1, a_2]$ values that provide the highest $|S_{21}|$ values for the specified desired $\arg\{S_{21}\}$ is only one type of possible selection process that can be implemented with the

INN approach under discussion. Alternatively, we might wish to accept all $[a_1, a_2]$ values that provide a $|S_{21}|$ larger than a certain value (eg. this might be larger than 10), and then select those elements that are geometrically closest in conductor size to elements (of different $\arg\{S_{21}\}$) that would be adjacent to the element in question in a typical transmitarray environment. Or indeed we might use any other design-dictated selection criteria.

3.6 INN MODEL OUTPUT VERIFICATION

We use Algorithm-6, as suggested before, to derive the dimensions $[a_1, a_2]$ of n^{th} transmitarray element for a given transmission phase value. This transmission phase value $\arg\{S_{21}\}$ of each element is computed using equation (2.2-56) derived in Section 2.2. But for the time being we just want to test the *accuracy* of our INN modelling procedure.Thus, we consider a set 'F' of arbitrarily selected transmission phase values ($\arg\{S_{21}\}$), assuming that they range from -270^o to 37^o because that is the range of transmission phase values possible with 3-layered square patch transmitarray evident in Figure 3.2-8. We consider around 200 different $\arg\{S_{21}\}$ values within this range in 'F' and find the corresponding dimensions $[a_1\ a_2]$ employing Algorihm-6, given that we have already implemented Algorithm-1 through Algorithm-5, and generated Database-1, Database-2 and Database-3. Once this has been done, the next step is to determine whether the $[a_1\ a_2]$ computed through this process are accurate. In order to accomplish that, we will feed each output from Algorithm-6 that corresponds to a particular transmission phase value back into Algorithm-3 to determine the associated transmission phase, and finally compare this phase value with the original specified input phase[26]. We have condensed this procedure to evaluate the performance of inverse modelling into Algorithm-7.

As indicated in the pseudocode for Algorithm-7, error E_{ANN}^{k} pertaining to the actual specified transmission phase input $\arg\{S_{21}\}^{k}$, and the output phase $\arg\{S_{21}\}_{EM}^{k}$ realized using the dimensions rendered by Algorithm-6, is computed. We also compare the output transmission

[26] For each of the 200 $\arg\{S_{21}\}$ arbitrarily selected values referred to in the previous paragraph.

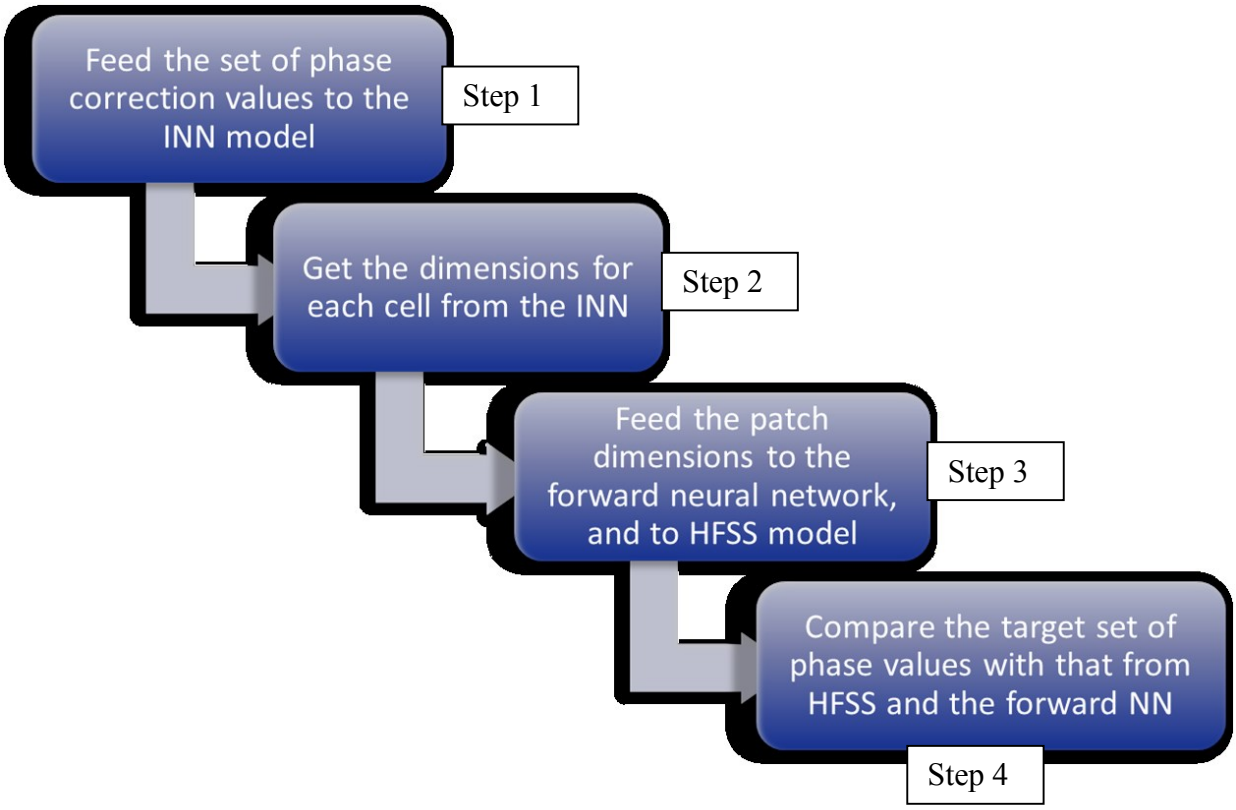


Figure 3.6-1 Summary Approach Employed to Authenticate the Output of INN Model

phase $\arg\{S_{21}\}^k_{EM}$ computed using the full wave 3-D model in HFSS, for all the output dimensions of the form $[a_1 \quad a_2]^k$ (such that, $\arg\{S_{21}\}^k \in F \quad \forall k = 1,2,3....200$) with that of the actual transmission phase input $\arg\{S_{21}\}^k$ and compute E^k_{EM} . Thus, when we do this for the 200 arbitrarily selected $\arg\{S_{21}\}$ values we obtain the cumulative error values (defined in Algorithm-7 pseudocode) of $E_{ANN} = 0.026$ % and $E_{EM} = 0.15$ %.

Vectors U_1, U_2, V_1 and V_2 are 'returned' by Algorithm-7. U_1 and U_2 contain transmission phase and amplitude respectively, computed using Algroithm-3 for the dimensions rendered by the inverse modelling process in Algorithm-6. Likewise, transmission phase and amplitude values computed using HFSS (full-wave 3D model) corresponding to the element dimensions calculated

Algorithm-7: INN Output Authentication Using Forward NN and Full-wave EM Model (HFSS)

comment: Set 'F' is created containing 200 elements representing fictional phase values required for different transmitarray elements with range -270° to 37°

Input: Set 'F' of Fictional Transmission Phase values ($\arg\{S_{21}\}\ \forall n=1,2,3$) presumably required at the center of various elements on the tranmistarray aperture.

Output: i) Error 'E_{ANN}' the Normalized Euclidean Norm of the difference between output phase values from forward neural network and actual input phase values in set F

ii) Error 'E_{EM}' the Normalized Euclidean Norm of the difference between output phase values from HFSS and actual input phase values in set F

iii) Vectors of phase values corresponding to output dimensions computed by Algorithm-6 and HFSS.

begin

k=1

while k <= cardinality of set F

function call - *Algorithm-6*(***argument*** = $\arg\{S21\}^k$)

comment: such that, $\arg\{S_{21}\}^k \in F$

comment: store output [a1 a2]^k

function call- *Algorithm-3(* ***argument*** = [a1 a2]^k)

comment: store output $arg\{S_{21}\}^k_{ANN}$ in vector 'U_1' and output $|S_{21}|^k_{ANN}$ in vector 'U_2'

function call- *full-wave EM Model in HFSS* ***(argument*** = [a1 a2]^k ***)***

comment: store output $arg\{S_{21}\}^k_{EM}$ in vector 'V_1' and output $|S_{21}|^k_{EM}$ in vector 'V_2'

comment: compute E^{ANN} and E^{EM}

$$E^k_{ANN} = \left[\frac{\arg\{S_{21}\}^k_{ANN} - \arg\{S_{21}\}^k}{\max(\arg\{S_{21}\}^k) - \min(\arg\{S_{21}\}^k)}\right]^2 \qquad k \text{ is the index for elements in set } F$$

$$E^k_{EM} = \left[\frac{\arg\{S_{21}\}^k_{EM} - \arg\{S_{21}\}^k}{\max(\arg\{S_{21}\}^k) - \min(\arg\{S_{21}\}^k)}\right]^2 \qquad k \text{ is the index for elements in set } F$$

k = k+1

end

$$E_{ANN} = \frac{1}{n(\mathrm{F})}\left[\sum_{k=1}^{n(\mathrm{F})} E^k_{ANN}\right]^{\frac{1}{2}}, \qquad E_{EM} = \frac{1}{n(\mathrm{F})}\left[\sum_{k=1}^{n(\mathrm{F})} E^k_{EM}\right]^{\frac{1}{2}}$$

Return: E_{ANN} , E_{EM}, U_1, U_2, V_1, V_2

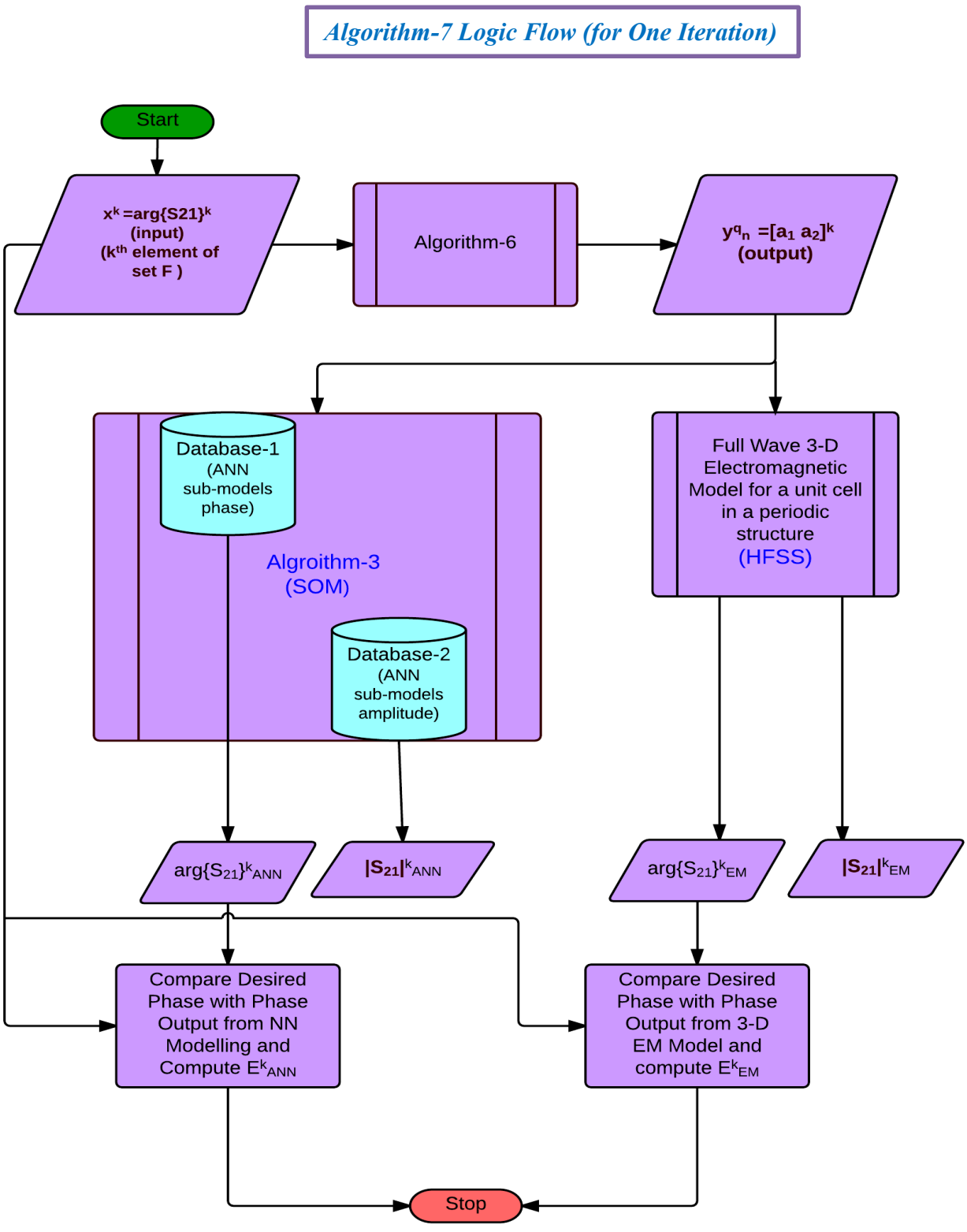


Figure 3.6-2 Logic Flow of Algorithm-7 for Single Iteration (k^{th} iteration) Demonstrating the Procedure Employed to Check the Authenticity of the Output from the INN Modelling

using Algorithm-6 (for desired phase values in set F) are stored in vectors V_1 and V_2 respectively. The results of these tests are encapsulated in Figures 3.6-3 and 3.6-4 and in Table 3.6-1.

The plots in Figure 3.6-3 and 3.6-4 are linear regression curves to implement Step 4 in Figure 3.6-1, where we have known/ actual transmission phase ($\arg\{S_{21}\}$) values on the horizontal axis (from the set F), and on the vertical axis, in Figure 3.6-3 and Figure 3.6-4, we have transmission phase values computed using Algorithm-3 and full-wave HFSS model respectively. This is depicted in Step 3 in Figure 3.6-1. The transmission phase values on the vertical axis correspond to the [a_1, a_2] dimensions output by Algorithm-6 for the required phase values in F (Step 2 in Figure 3.6-1).

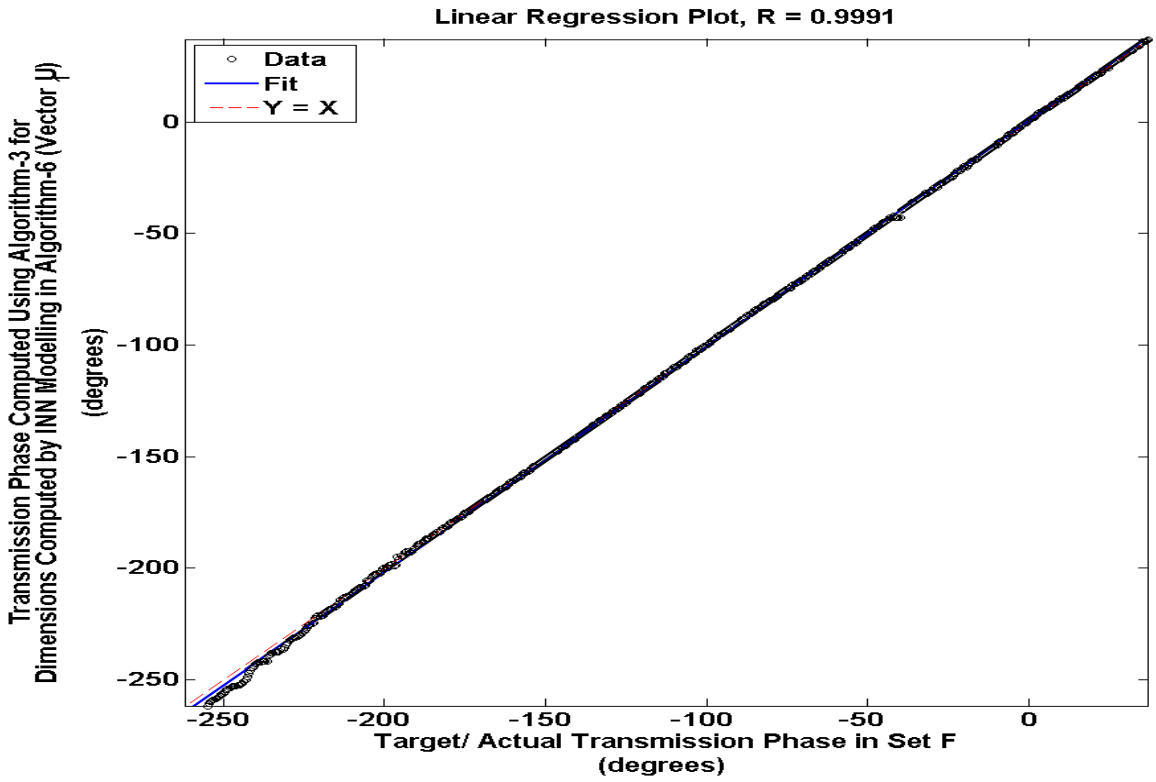


Figure 3.6-3 Linear Regression Plot Comparing Desired Phase Values in Set F and Phase Output given by SOM based Forward Neural Network Model Using Algorithm-3 for the Output [a_1, a_2] Dimensions Computed Using Algorithm-6

After analyzing all these validation check results, it can be seen that when neural network output is compared with the actual transmission phase values in set F (Figure 3.6-3), a very high degree of accuracy seems to be achieved since R =0.9991 but when we analyze regression in Figure 3.6-4 it appears INN output seems to be less accurate. To resolve this confusion, we need to appreciate the fact that when we train the forward neural network sub-models in Algorithm-2 using S-parameter database, there is an error associated with training and this error is propagated into INN models and therefor, there is a discrepancy between full wave results and neural network model results when we compare regression values in Figures 3.6-3 and 3.6-4.

Table 3.6-1 Error Values and Regression Associated from Algorithm-7

Validation Using Neural Network Model		**Validation Using EM Model**	
E_{ANN}	R_{ANN} (Regression)	E_{EM}	R_{EM} (Regression)
0.026 %	0.9991	0.15 %	0.9929

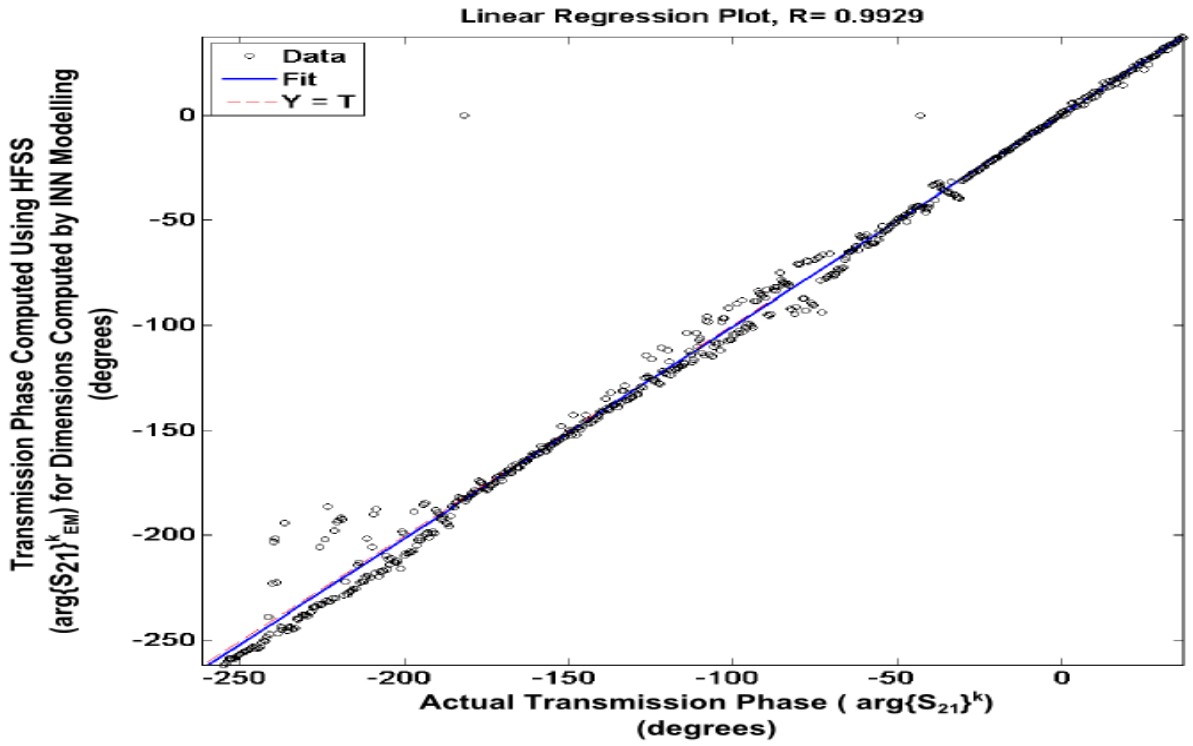


Figure 3.6.4 Linear Regression Plot Comparing Desired Phase Values in Set F and Phase Output given by Full-Wave 3D EM Model in HFSS for the Output [a_1, a_2] Dimensions Computed Using Algorithm-6

Moreover, it also remains to be confirmed that whether the transmission amplitude, associated with each element dimension $[a_1\ a_2]^k$ corresponding to the desired phase $\arg\{S_{21}\}^k$ ($\forall k \in F$), is large enough for considerable transmission. Ideally, we would want the amplitude to be unity but due to physical limitations it is not possible. Hence, a threshold is fixed, in this case -3 dB, such that only transmission amplitude above this threshold is considered viable. This concept was also utilized in the determination of dimensions in Algorithm-6. The plot in Figure 3.6-4 was generated to validate that transmission amplitudes for selected $[a_1\ a_2]^k$ pairs is in fact above -3 dB threshold.

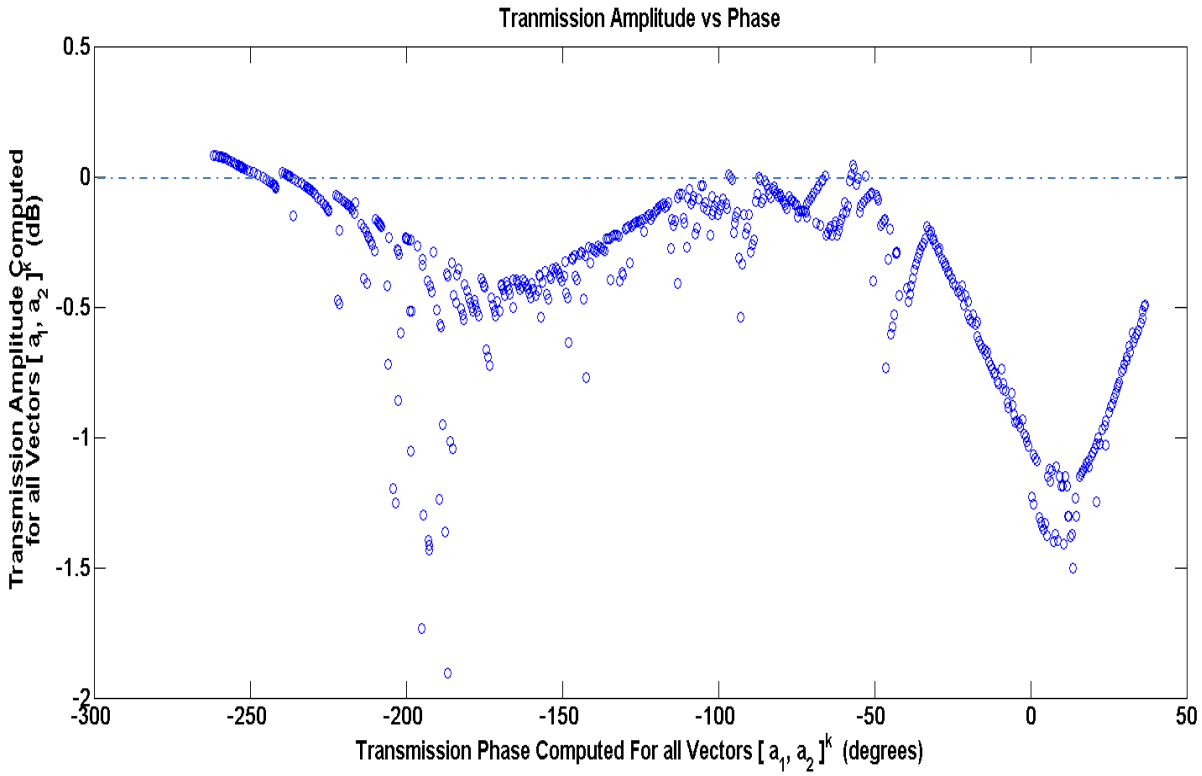


Figure 3.6-5 Transmission Coefficient Amplitude vs Phase for Various $[a_1\ a_2]$

In the transmission amplitude versus transmission phase plot in Figure 3.6-5, for some transmission phase values the amplitude is above 0 dB, which is of course physically unfeasible but due to the error in the training of forward amplitude neural network sub-models stored in

Database-2, there is a possibility that output amplitude become greater than 0 dB. Since, the average test error for the sub-models in Database-2 is 2.3%, it is safe to assume that the actual amplitude corresponding to transmission phase values would be below 0 dB.

3.7 CONCLUDING REMARKS

Several new contributions, clarifications and complete implementational details have been offered in this chapter. Although inverse neural network (INN) ideas have been described and used by others for RF circuit design, they do not appear to have been used in detailed antenna design work. In Chapter 3 we have therefore taken existing neural network ideas and adapted them to the transmitarray design type problem. We have shown that it is not practical to apply the INN concepts to the database of points obtained directly from full-wave simulations; too many such simulations would be needed. In Section 3.2 it was demonstrated how the distribution of these initial full-wave database points can be examined to decide on how to best use forward (as opposed to inverse) neural network modelling to augment the number of full-wave database points by a factor of about fifty, in order to have sufficient training data for implementing INNs accurately. Details on the algorithms (Algorithm-1 through Algorithm-3) used in such forward neural network modelling to obtain the augmented database ("forward" Database-1 and Database-2 for the transmission phase and amplitude) are provided in Section 3.3. It is then shown in Section 3.4 that the highly non-linear and multivalued (not one-to-one) nature of the data necessitates the use of inverse sub-models. Algorithm-4 and Algorithm-5 have been given for such an implementation. The inverse sub-model theory available in the literature is altered slightly in Section 3.5 to suite some database inversion issues (and decision-making) possibly unique to the present type of antenna design problem. This latter extension, implemented via Algorithm-6 (to create "inverse" Database-3) also offers some flexibility that might be of benefit in the design of this class of antennas in the future. The work in Section 3.6, implemented via Algorithm-7, presents solid verification of the effectiveness of the INN-based inversion of the augmented design database. All the algorithms needed are given in the form of flow-charts and complete pseudo-code for easy implementation by others; such detailed implementational details have not been available elsewhere in the literature.

CHAPTER 4

Design Database Construction and Inverse Neural Network Model Generation Including Incidence Angle Effects

4.1 PRELIMINARY REMARKS

We characterized the transmitarray unit cell response using ANNs in Section 3.3 & Section 3.4. Subsequently, in Section 3.5 & Section 3.6 we generated the INN to invert the S-parameter database. The implicit assumption was that for each cell on the transmitarray aperture, the feed field is incident normally. But that's not precisely the case in reality; instead, the angle of incidence of the incident wave depends on the centroid coordinates of each cell, and so changesfrom cell to cell. This is depicted in Figure 1.1-3 and Figure 2.2-1; θ_{inc} is the angle of incidence symbol used there for a given cell. We therefore wanted to investigate the impact of incidence angle inclusion in the design database.In order to accomplish this we regenerated these databases but this time taking into account the incidence angle. Therefore, the modelling of unit cell residing in a 2D infinite periodic structure was carried out by incorporating the oblique incidence in the process described in Section 3.2.

In this chapter, in Section 4.2, we will delineate the procedure used to generate the S-parameter database for oblique incidence S-parameters will be calculated for varying unit cell dimensions and a separate S-parameter database will be generated for each of the discrete incidence angle cases. Thereafter, in Section 4.3, we proceed to characterize these S-parameter databases using ANN modelling on the same note as was undertaken in Section 3.3 and neural network databases will be created. Then we will move on to the INN model generation in Section 4.4 and the procedure to integrate these discrete inverse models – corresponding to various incidence angles – will be outlined and implemented. To validate the inverse modelling procedure assume a set of transmission phase values required on the aperture of transmitarray and will seek the output from inverse models and authenticate it. Finally, we will conclude this chapter in Section 4.6.

4.2 S-PARAMATER DATABASE GENERATION FOR OBLIQUE INCIDENCE CASE

4.2.1 Database Generation for Oblique Incidence Cases Simulated Using Full-Wave Electromagnetic Analysis

There are various stages of database construction such as unit cell characterization, setting up the full wave EM simulation of the unit cell using Floquet port analysis and periodic boundary conditions, and storing the relevant S-parameters. Using the same principles as described in Section 3.2, a design database for S-parameters is generated taking into account aforementioned *oblique incidence* angle values. The unit cell specifications as described in Section 3.2-1, as well as periodic structure analysis for the unit cell presented in Section 3.2-2, are used exactly in the same way. However, while setting up the simulations in HFSS, the periodic boundaries are specified in such a way that the impact of incidence angle is considered in the computation of the periodic structure response. For each discrete value of the incidence angle, a separate design database is generated. We will use the symbol θ_i to denote te incidence angle value.

Because full wave electromagnetic simulation for the infinite periodic structure is computationally very extensive, we only consider five different oblique incidence cases, represented by a set

$$\xi = [0^o \;\; 5^o \;\; 20^o \;\; 35^o \; 45^o] \qquad (4.2\text{-}1)$$

$$\text{such that } \theta_i \in \xi \quad \forall \; i = [1,5]$$

For each θ_i value we vary a_1 and a_2 (as shown in Figure 3.2-2) and fix all other variables as in Section 3.2.

Table 4.2-1 Unit Cell Specifications

s = 3 mm cell size
h = 1 mm height of substrate
ε_r = 2.2 (relative permittivity)
f = 30 GHz (frequency)
$\theta_{inc} = \theta_i$

We then proceed to find the corresponding transmission coefficient for n^{th} element lying in an infinite periodic structure in the x-y plane. A non-uniform sample space for (a_1, a_2) was chosen based upon the associated non-linearity in the periodic structure response. The more rapidly varying the response, the larger the number of samples chosen in that region, as depicted in the Figure 3.2-4. This approach was followed for each discrete incidence angle. This way five different S-parameter design databases ($h^i_{s-database}$ with $i = 1, 2 \ldots 5$), namely

$$[\arg\{S_{21}\}, |S_{21}|]^i = h^i_{s\text{-}database}(\theta_i, a_1, a_2) \tag{4.2-2}$$

We feed the samples (a_1, a_2) to the full wave 3-D electromagnetic simulator and store the corresponding phase ($arg\{S_{21}\}$) and amplitude ($|S_{21}|$) of the transmission coefficient and this process is repeated for each incidence angle in set ξ . Plots in Figure 4.2-1 to Figure 4.2-5 are generated for transmission phase and transmission amplitude for all the five S-parameter databases corresponding to different incidence angles. Each S-parameter database has approximately 3000 points.

We observed that for a given pair of (a_1, a_2), the variation of $\arg\{S_{21}\}$ and $|S_{21}|$ as function of θ_i follow a predictable pattern, and therefore we can use a polynomial as an interpolant to predict the values of S_{21} for additional values θ_i such that we can increase the number of elements in ξ which would facilitate higher accuracy in the final transmitarray design. When designing the transmitarray, for a given cell center, the incident angle θ_{inc} can have a value absent from set ξ, thus begging to be approximated to the nearest θ_i available. Therefore, denser the set ξ is, the higher will be the accuracy of the final design. This will become clear in Section 4.5, as well as chapter 5 where we will actually design a transmitarray.

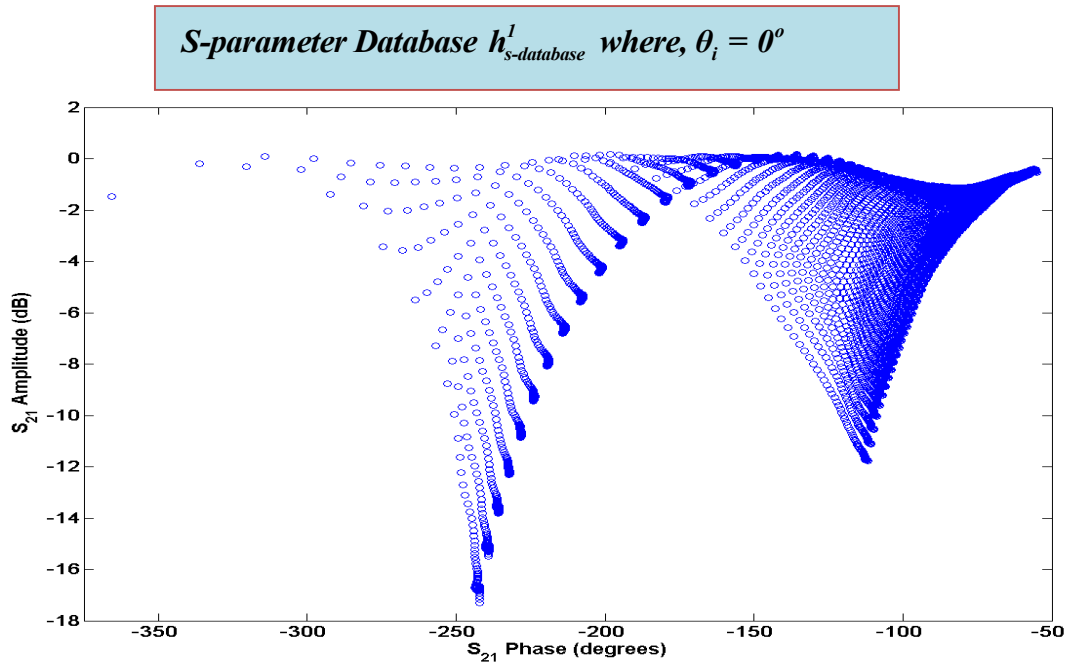


Figure 4.2-1 (a)

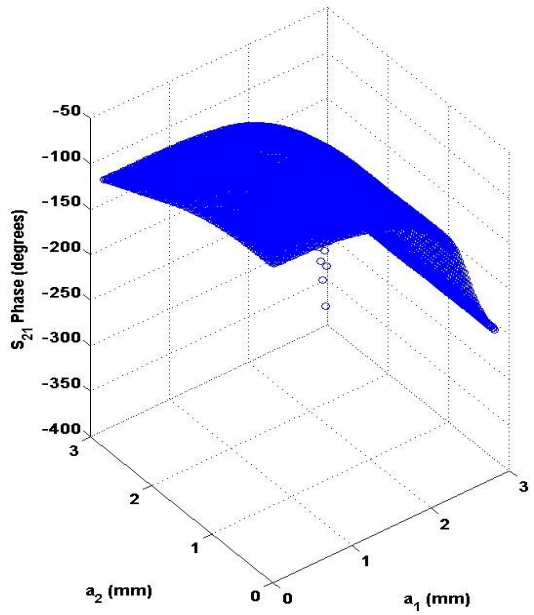


Figure 4.2-1 (b)

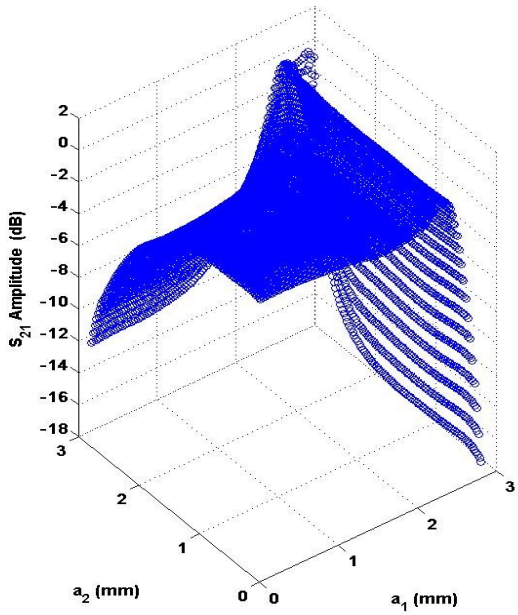


Figure 4.2-1 (c)

Figure 4.2-1 (a) Represents Amplitude versus Phase for Each Combination of (a_1, a_2), (b) and (c) Depict Transmission Phase and Amplitude, Respectively, for Database $h^{1}_{s\text{-}database}$

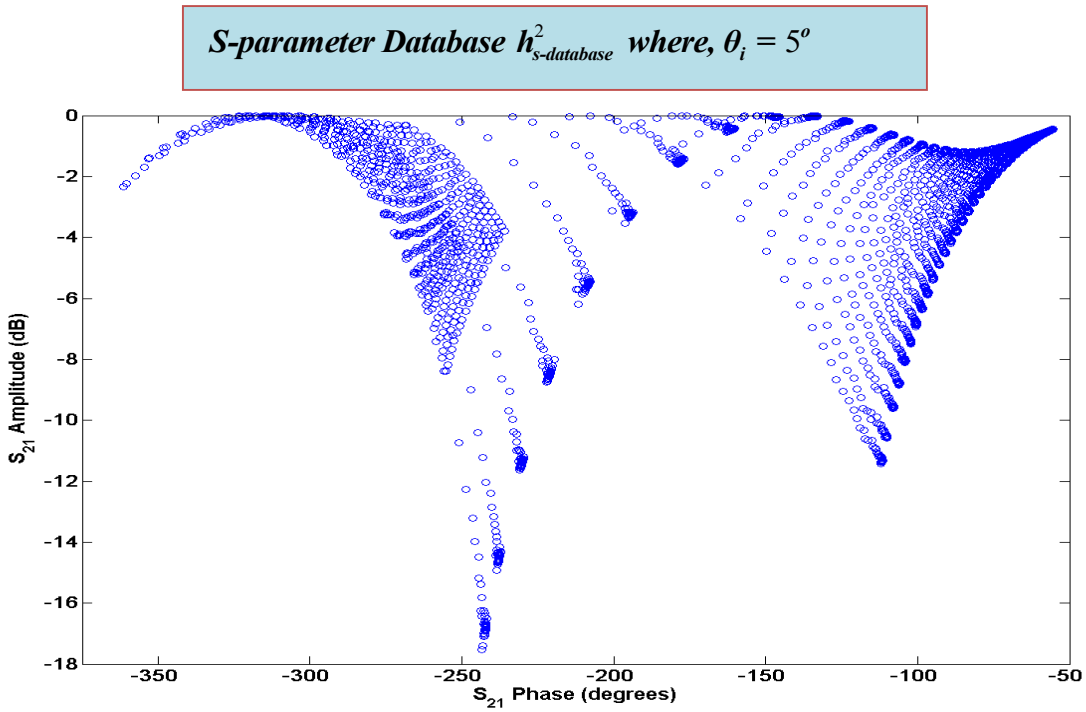


Figure 4.2-2 (a)

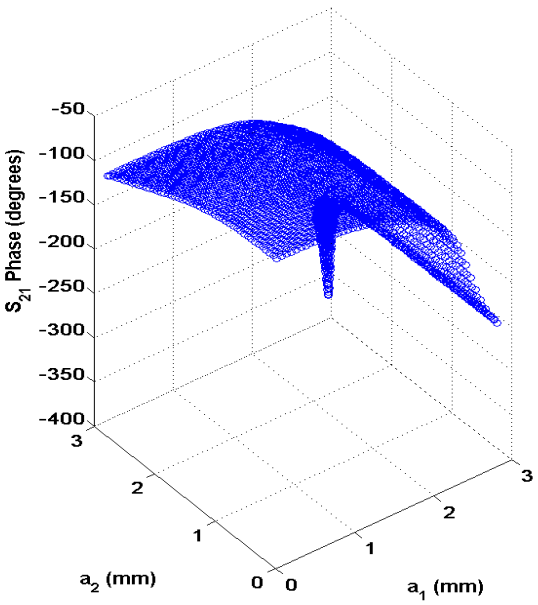


Figure 4.2-2 (b)

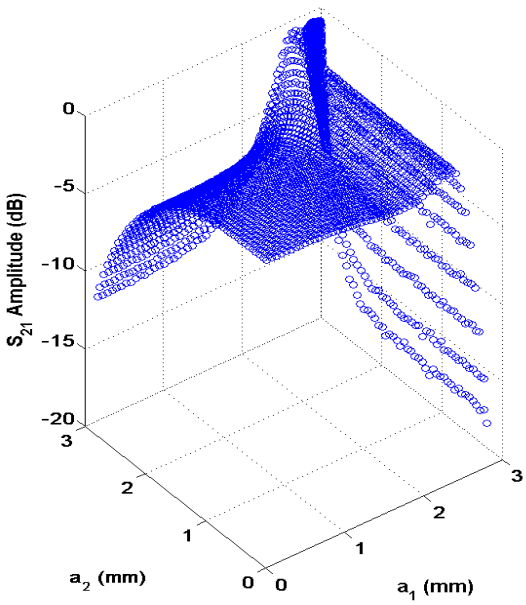


Figure 4.2-2 (c)

Figure 4.2-2 (a) Represents Amplitude versus Phase for Each Combination of (a_1, a_2), (b) and (c) Depict Transmission Phase and Amplitude, Respectively, for Database $h^2_{s\text{-}database}$

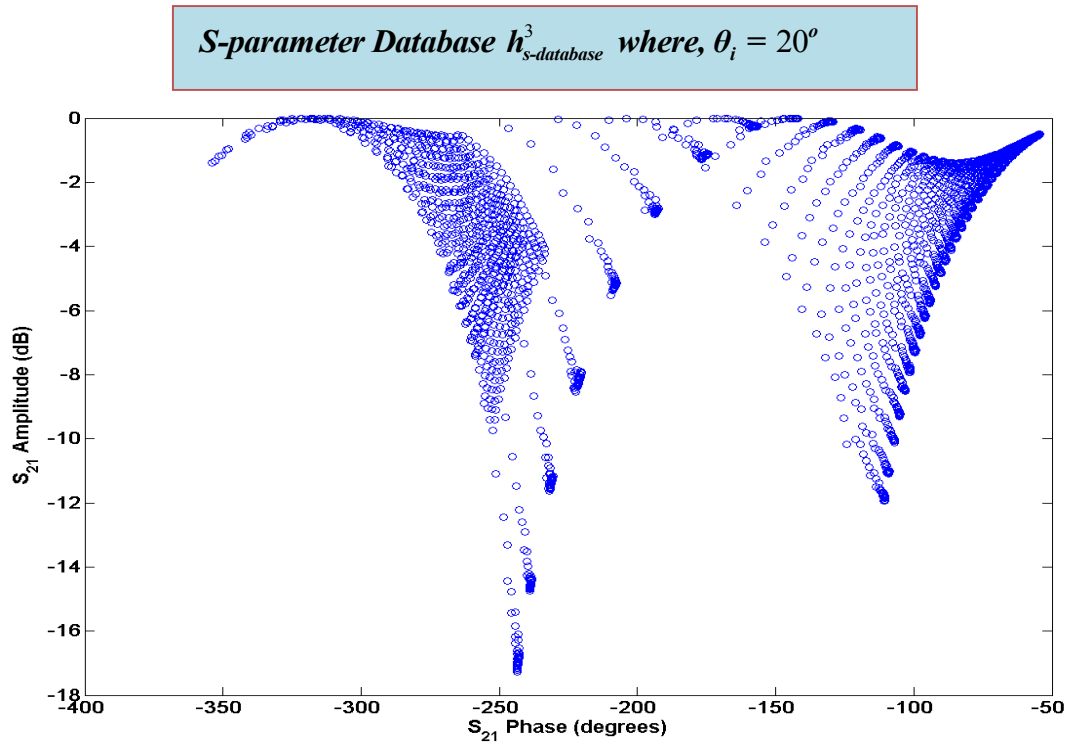


Figure 4.2-3 (a)

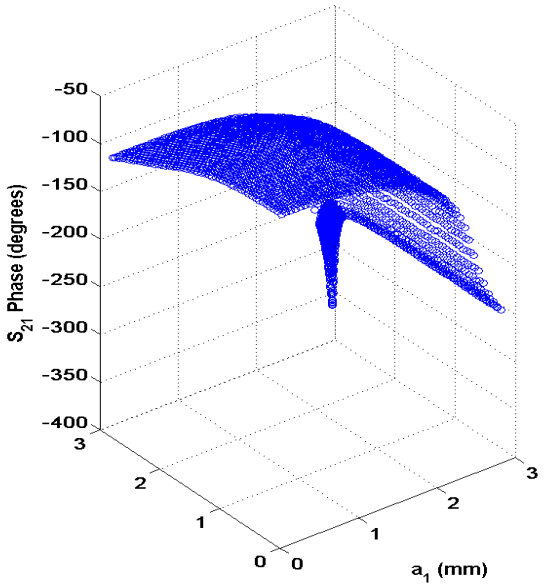


Figure 4.2-3 (b)

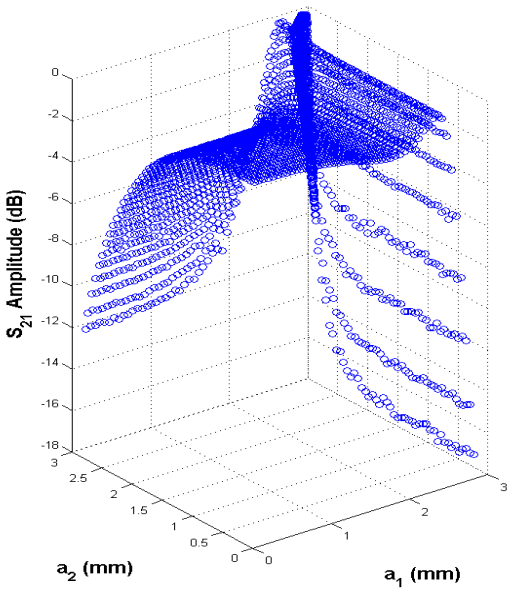


Figure 4.2-3 (c)

Figure 4.2-3 (a) Represents Amplitude versus Phase for Each Combination of (a_1, a_2), (b) and (c) Depict Transmission Phase and Amplitude, Respectively, for Database $h^3_{s\text{-}database}$

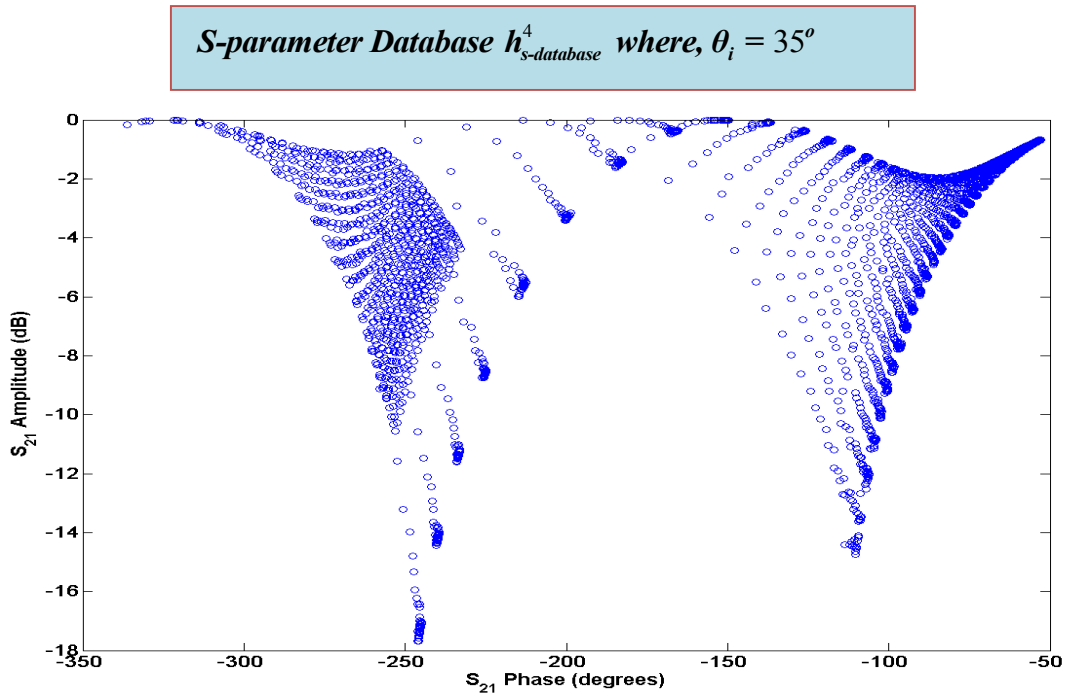


Figure 4.2-4 (a)

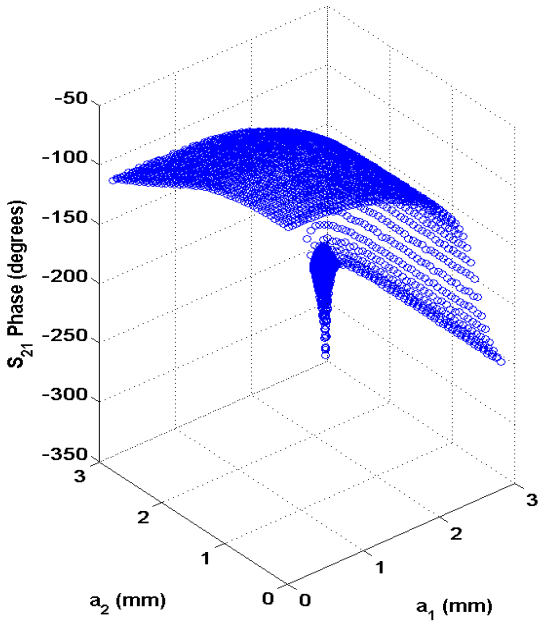


Figure 4.2-4 (b)

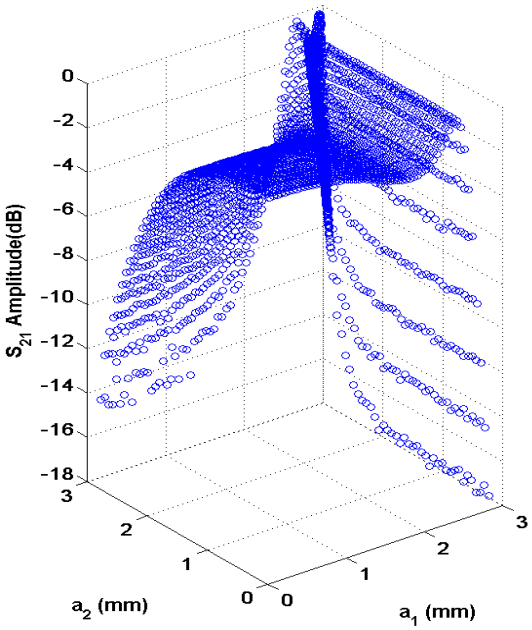


Figure 4.2-4 (c)

Figure 4.2-4 (a) Represents Amplitude versus Phase for Each Combination of (a_1, a_2), (b) and (c) Depict Transmission Phase and Amplitude, Respectively, for Database $h^4_{s\text{-}database}$

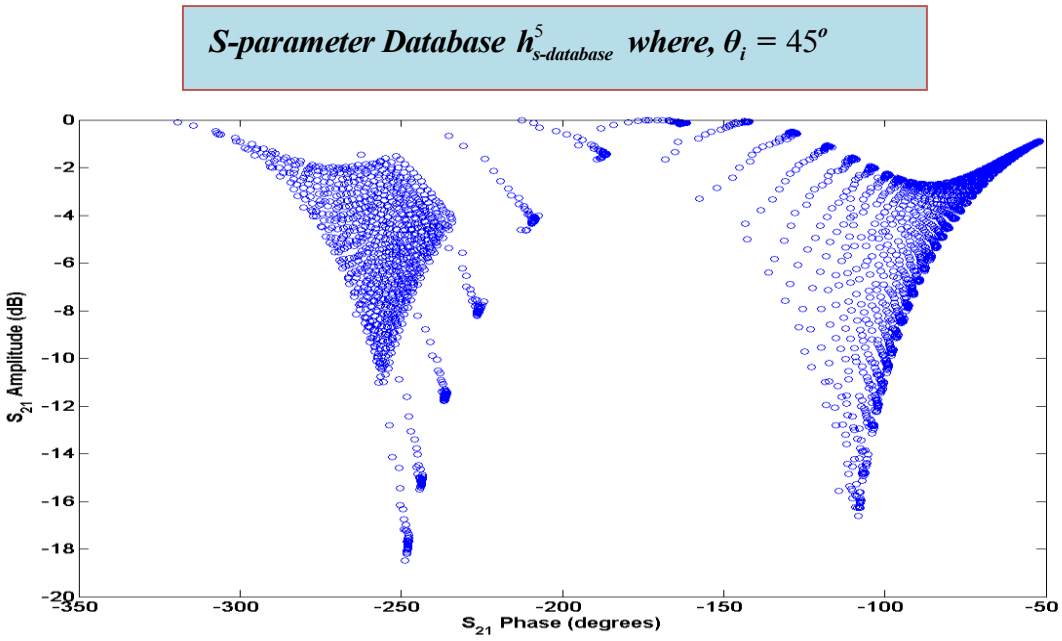


Figure 4.2-5 (a)

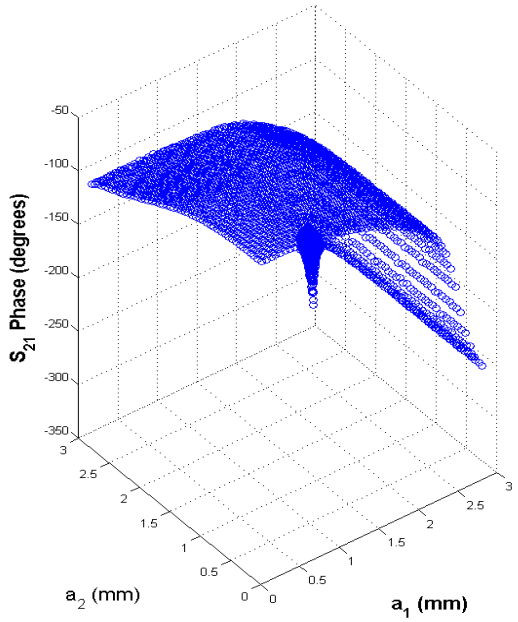


Figure 4.2-5 (b)

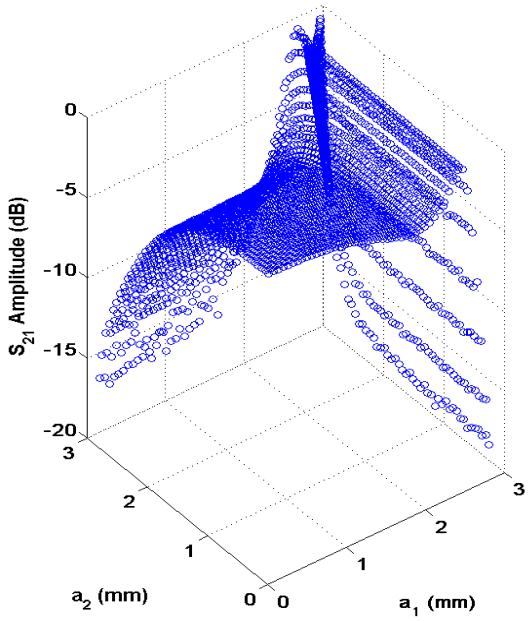


Figure 4.2-5 (c)

Figure 4.2-5 (a) Represents Amplitude versus Phase for Each Combination of (a_1, a_2), (b) and (c) Depict Transmission Phase and Amplitude, Respectively, for Database $h^5_{s\text{-}database}$

4.2.2 Database Generation for Oblique Incidence Cases Generated Using Interpolation

As mentioned before, the predictability of arg{S_{21}} and $|S_{21}|$ with respect to the variation I θ_i can be exploited to interpolate the transmission coefficient values for values θ_i not present in set ξ, and Figures 4.2-6 to Figure 4.2-11 demonstrate this. Figures 4.2-6 to 4.2-8 represent phase variation, while Figures 4.2-9 to 4.2-11 represent amplitude variation.

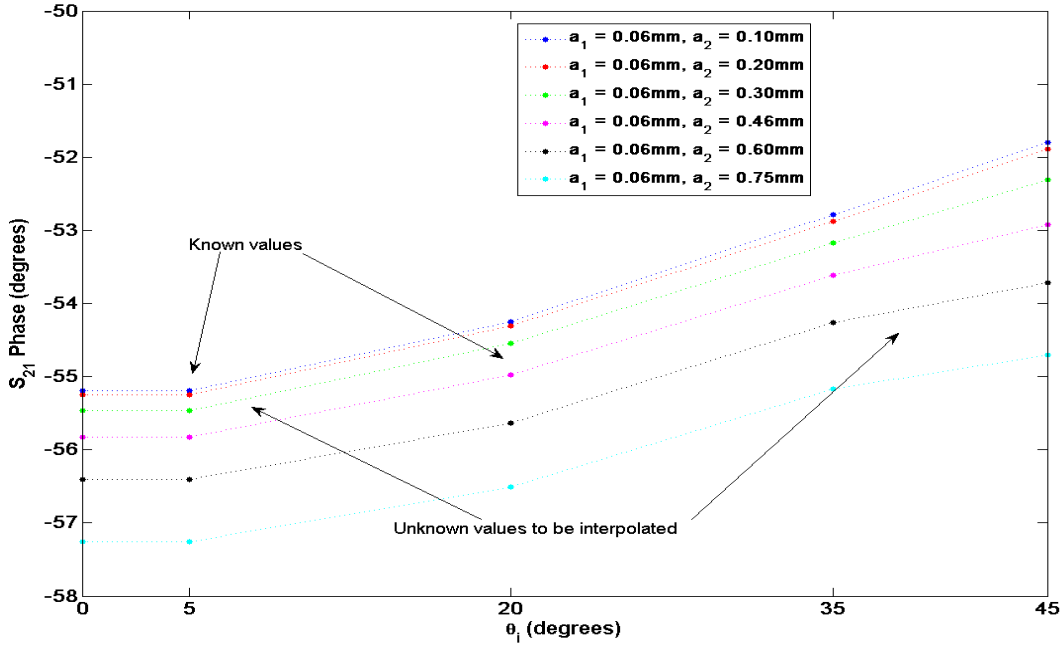


Figure 4.2-6 Transmission Phase Variation w.r.t. to Discrete Incidence Angle Values θ_i for 6 Different (a_1, a_2) Pairs

Since, we only have transmission phase and amplitude values for five different values $\theta_i \in \xi$, the dots in the plots represent the known transmission phase and amplitude, whereas dashes represent the unknown values which we will interpolate. These plots only show a few (a_1, a_2) combinations from among the approximately 3000 samples. The vital observation that needs to be made here, while analyzing Figure 4.2-6 and the following figures, is that the curve representing the variation of transmission phase and amplitude for a fixed (a_1,a_2) follows a function that can be described using a polynomial fit.

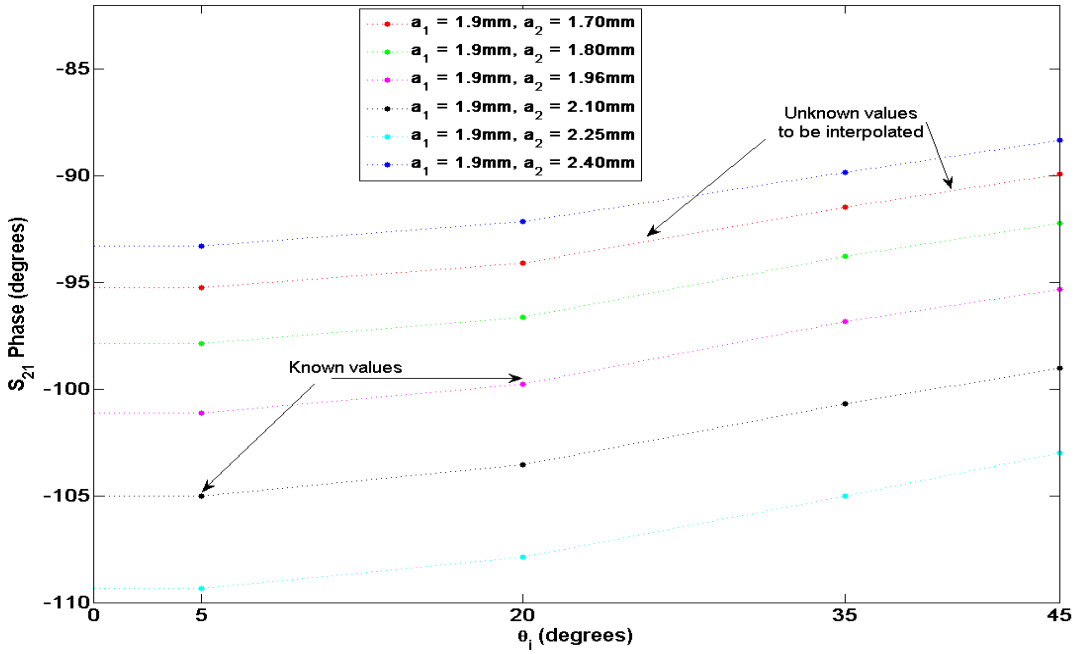


Figure 4.2-7 Transmission Phase Variation w.r.t. to Discrete Incidence Angle Values θ_i for another 6 Different (a_1, a_2) Pairs

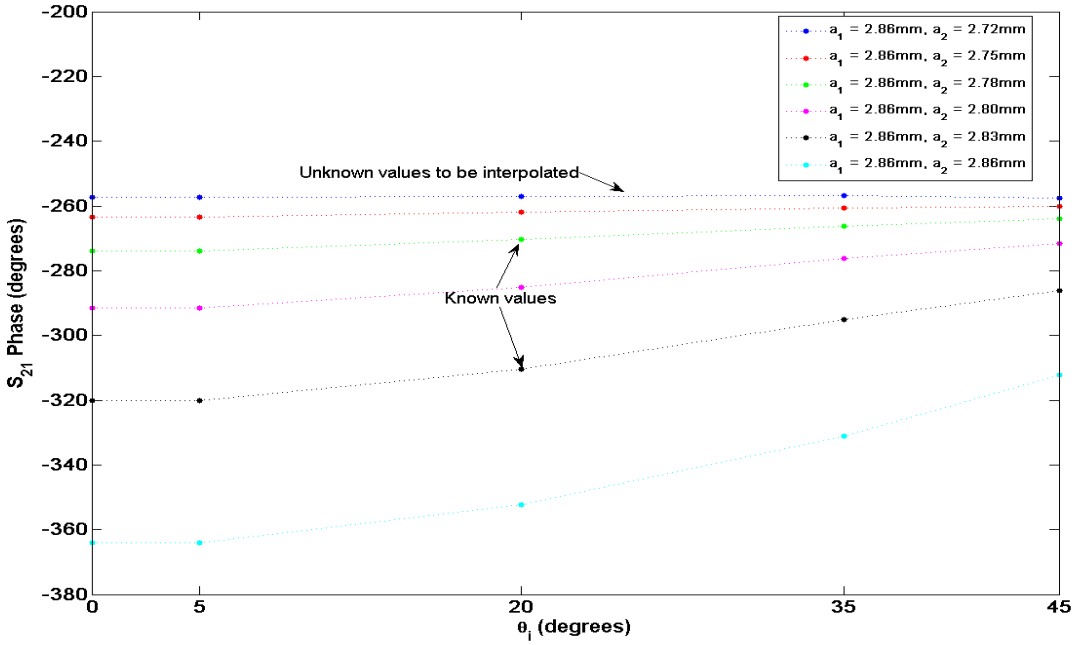


Figure 4.2-8 Transmission Phase Variation w.r.t. to Discrete Incidence Angle Values θ_i for another 6 Different (a_1, a_2) Pairs

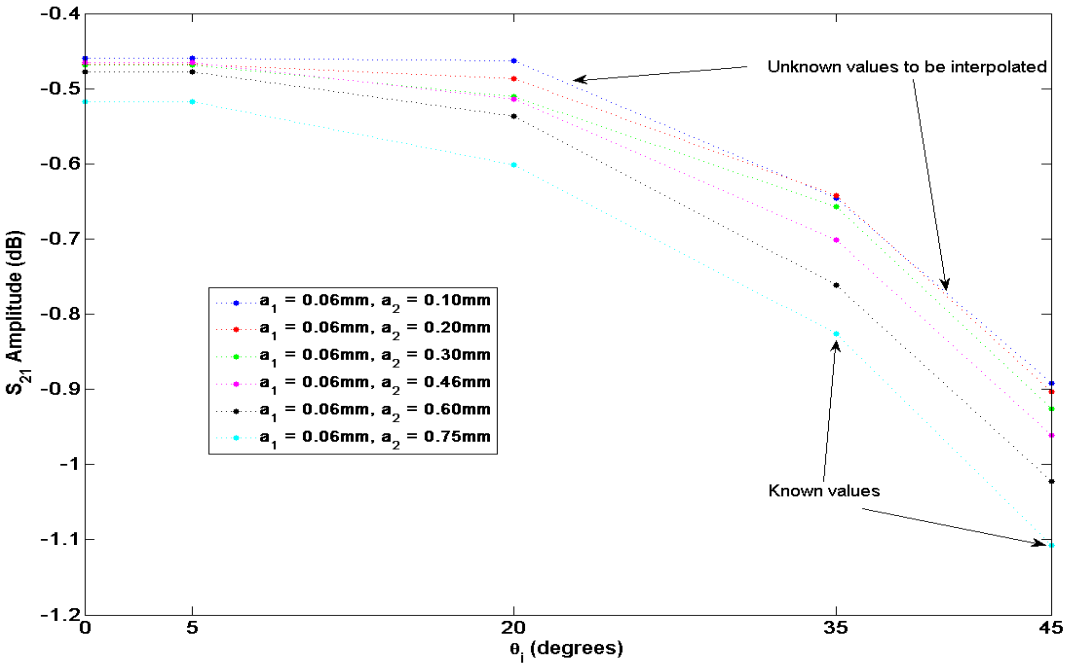


Figure 4.2-9 Transmission Amplitude Variation w.r.t. Discrete Incidence Angle Values θ_i for 6 Different (a_1, a_2) Pairs

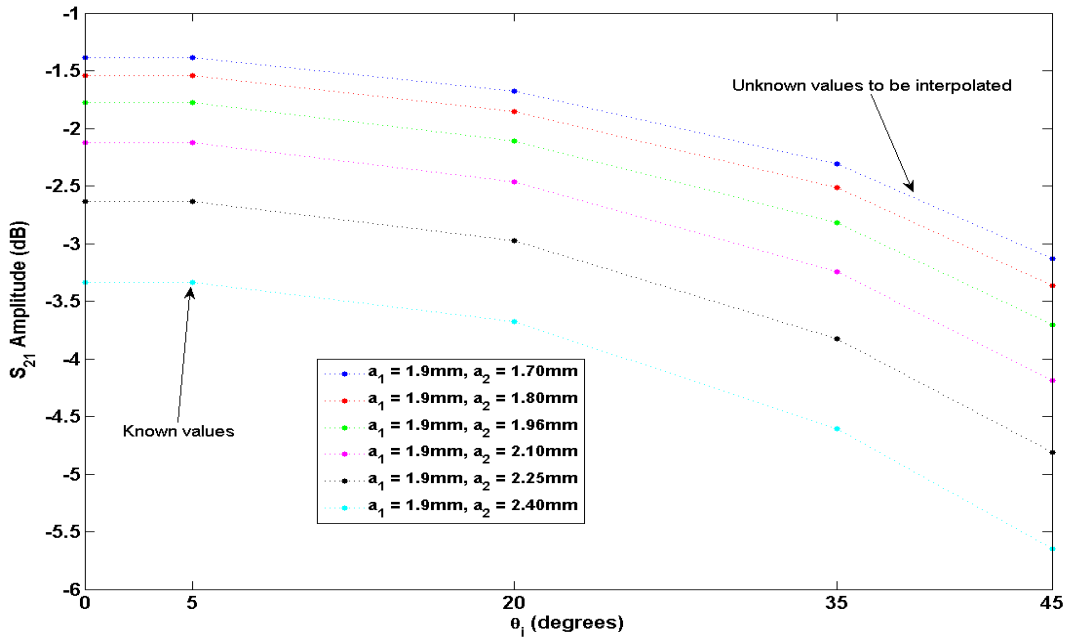


Figure 4.2-10 Transmission Amplitude Variation w.r.t. Discrete Incidence Angle Values θ_i for another 6 Different (a_1, a_2) Pairs

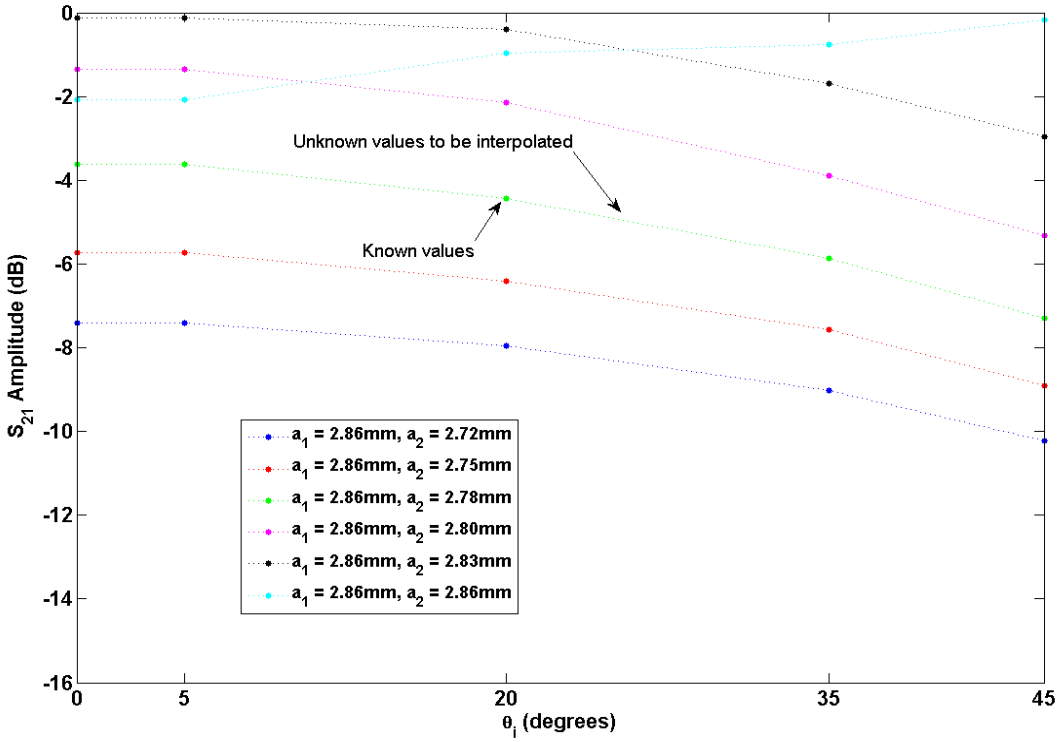


Figure 4.2-11 Transmission Amplitude Variation w.r.t. Discrete Incidence Angle Values θ_i for another 6 Different (a_1, a_2) Pairs

The interpolation was performed for both transmission phase and amplitude curves. I^{interp} is the polynomial function to interpolate for transmission phase for different values of angle of incidence given by θ_j in set ξ' such that

$$\begin{aligned} \xi' &= [10^o\ 15^o\ 25^o\ 30^o\ 40^o] \\ \xi &= [0^o\ 5^o\ 20^o\ 35^o\ 45^o] \end{aligned} \tag{4.2-3}$$

The interpolation process angle is varied (θ_j) but $(a_1,a_2)^k$ is fixed represented as

$$(\arg\{S_{21}\}, |S_{21}|)^{(a_1,a_2)^k} = I^{interp}(\theta_j) \quad \forall\ \theta_j \in \xi' \tag{4.2-4}$$

for each value of k such that $k \in [1, 3000]$

After interpolation we generated transmission phase and amplitude for 3000 different (a_1, a_2) pairs for all the angles of incidence in set ξ'. Figures 4.2-12 (a) and 4.2-13 illusstrate this process for 6 different 'k' values or (a_1, a_2)^k pairs.

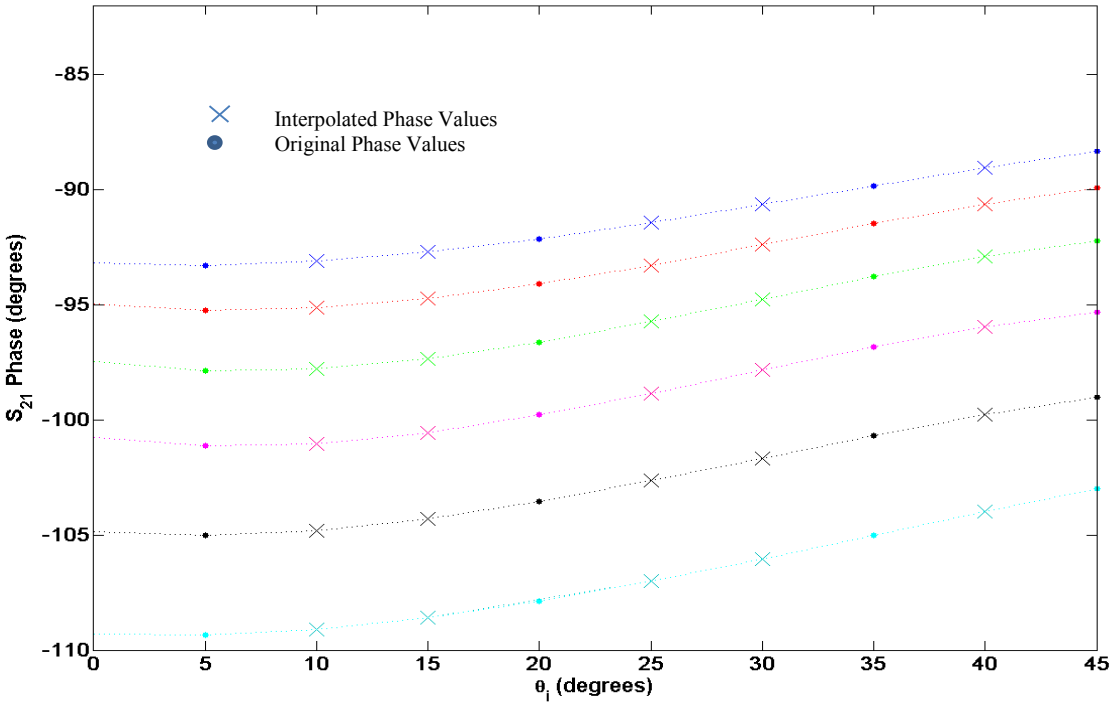


Figure 4.2-12 Transmission Phase Versus Incident Angle to Represent Interpolated Phase Values for 6 Different (a_1, a_2) Pairs Corresponding to Figure 4.2-7

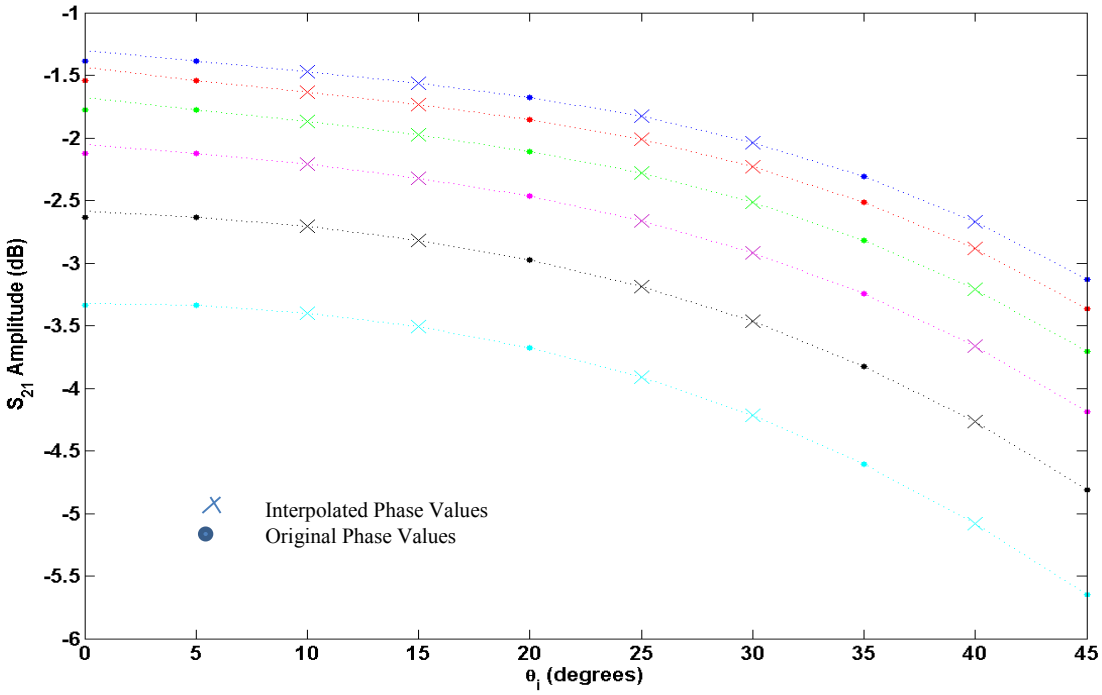


Figure 4.2-12 Transmission Amplitude Versus Incident Angle to Represent Interpolated Phase Values for 6 Different (a_1, a_2) Pairs Corresponding to Figure 4.2-10

This way we have additional S-parameter databases along with the ones represented by equation 4.2-2. We denote the additional databases by the symbol p, as follows:

$$[\arg\{S_{21}\},|S_{21}|] = p^{j}_{s\text{-}database}(\theta_j, a_1, a_2)$$

where $j \in [6,10]$ and $\theta_j \in \xi'$ (4.2-5)

with $\theta_{j=6} = 10^o, \theta_{j=7} = 15^o, \theta_{j=8} = 25^o, \theta_{j=9} = 30^o, \theta_{j=5} = 40^o$

Hence, we have ten S-parameter databases in total after combining equations 4.2-2 and 4.2-5. We carried out further analysis of the variation[27] of transmission phase and amplitude versus angle of incidence and we came across very interesting findings. First of all, the variation of transmission phase versus incident angle was not consistent for all (a_1, a_2) pairs i.e. for some pairs the variation was significant but for other pairs it was negligible.

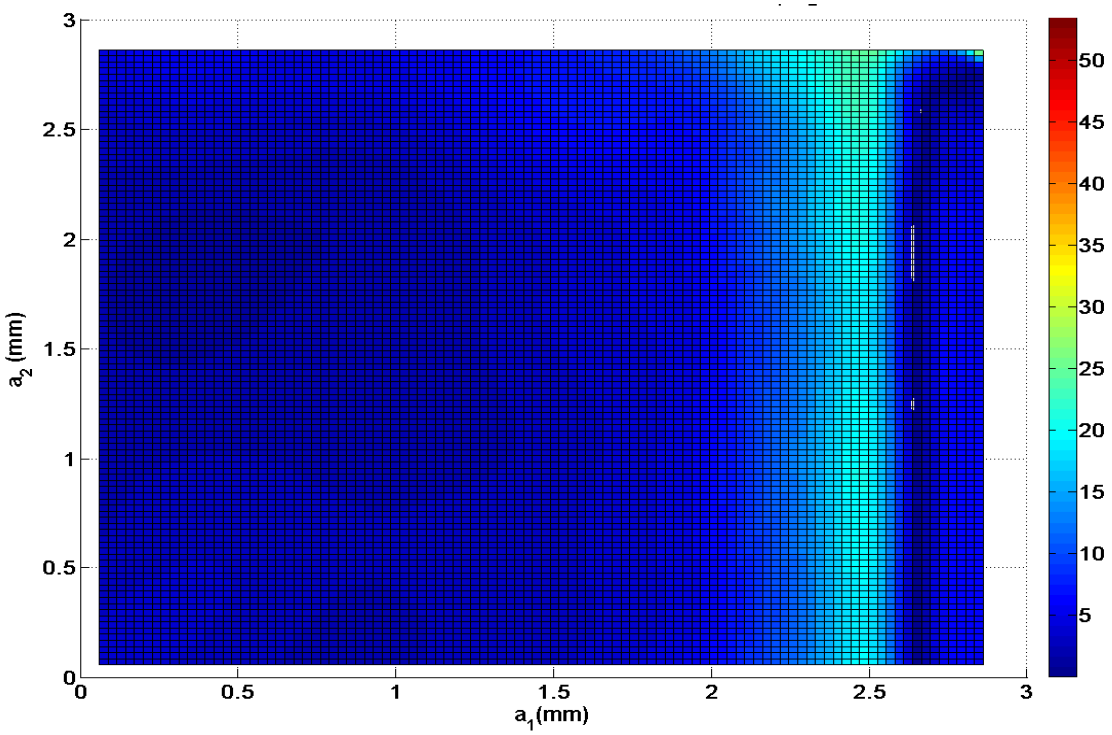


Figure 4.2-13 Intensity Plot of $V^t\,|_{(a_1,a_1)^t}$ Over the Entire Domain of (a_1,a_2)

[27] We are investigating this variation of transmission phase versus incidence angle when (a_1,a_2) pair is fixed because first of all, we needed to make a decision whether to generate incidence angle dependent databases (as it turns out we do need these since, in certain regions the variation is significant). We also wanted to check how much error has propagated into the design procedure of transmitarray when incidence angle is not considered. Finally, it can be concluded that if we avoid the values of (a_1,a_2) in the region in light blue shade (high variation) in Figure 4.2-13, then it may be possible to design a transmitarray considering normal incidence database.

The plot in Figure 4.2-14 represents the maximum variation/difference between two transmission phase values for varying incidence angle θ_{inc} but constant (a_1, a_2) and this variation V^t can be expressed as the maximum possible difference between two transmission phase values for variable incidence angle such that (a_1, a_2) is constant.

$$V^t \big|_{(a_1,a_1)^t} = \max \left| \arg\{S_{21}\}^{\theta_m} - \arg\{S_{21}\}^{\theta_n} \right|_{(a_1,a_2)^t} \quad (4.2\text{-}6)$$

where

$$t \in [1, 3000] \quad i.e. \text{ number of pairs } of \ (a_1, a_2)$$

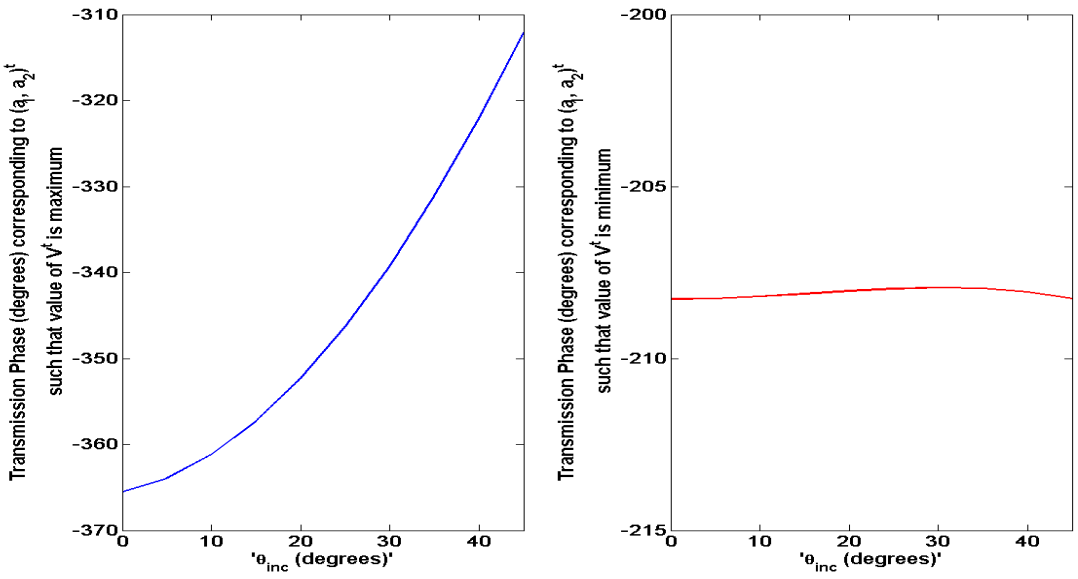
and

$$\theta_m, \theta_n \in \xi \cup \xi'$$

The following observations were made from the plot in Figure 4.2-13:

- Generally, the larger the dimensions of the unit cell patches (a_1, a_2) relative to the cell size 's' (Figure 3.2-3), the higher is the value of V^t, as indicated in Figure 4.2-14
- The maximum value of V^t was observed to be 53°

Now, we can make further observations to find the values of $(a_1, a_2)^t$ such that V^t is maximum or minimum. We identified these points in the (a_1, a_2) domain and here plotted them as in Figure 4.2-15. These plots show that V^t was maximum for $(a_1, a_2)^t$ = (2.859 mm, 2.859 mm), and minimum for $(a_1, a_2)^t$ = (2.64 mm, 1.9 mm).



(a) (b)

Figure 4.2-15 (a) Maximum and (b) Minimum Variation of $\arg\{S_{21}\}$ versus θ_{inc} for a Particular $(a_1, a_2)^t$

4.3 ARTIFICIAL NEURAL NETWORK BASED AUGMENTATION OF S-PARAMETER DATABASES $h^i_{s\text{-}database}$ AND $p^j_{s\text{-}database}$

As was the case for the S-parameter database $g_{database}$ in equation (3.2-3), the number of points, approximately 3000, in S-parameter databases $h^i_{s\text{-}database}$ and $p^j_{s\text{-}database}$ created in Section 4.2 are insufficient for satisfactory performance of inverse neural network thus generated. The idea is to augment these S-parameter databases' sample space and therefore we characterize the input-output mapping of these samples for each of the ten databases individually using neural network models (we again call it forward neural network modelling). Following the same procedure as in Section 3.3 we set up neural network models through the implementation of self-organizing maps for each of above S-parameter databases separately.

We feed the sample space for each of the ten databases ($h^{i}_{s\text{-}database}$ and $p^{j}_{s\text{-}database}$ $\forall$ i = [1,5] and j = [1,5]) to Algorithm-2 in Section 3.3, and we store the neural network models for phase and amplitude thus generated, as the output of this algorithm, in the neural network databases. In the present scenario, Algorithm 2`s input x, y and output databases can be represented as described in what follows.

There are roughly 3000 samples for each of the ten S-parameter databases. We run Algorithm-2 iteratively for ten times, for each incident angle, keeping the value of incidence angle constant for a given iteration. This operation is performed in Algorithm-8 and we end up with

$$x^{n}|_{\theta_q} = (a_1, a_2)^{n}|_{\theta_q} \tag{4.3-1}$$

$$y^{n}|_{\theta_q} = \begin{Bmatrix} \arg\{S_{21}\}^{n}_{\theta_q} & \forall\ phase \text{ model generation} \\ |S21|^{n}|_{\theta_q} & \forall\ amplitude \text{ model generation} \end{Bmatrix} \tag{4.3-2}$$

where,

$$n = 1, 2, 3\ldots\ldots \approx 3000 \quad \text{and}$$

$$\theta_q \in \xi \cup \xi' \text{ and } q=1,2,\ldots..10$$

Algorithm-8 clusters the sample space of 3000 points into nine sub-groups for each oblique incidence angle database, and then training of the neural network takes place with each sub-space acting as training and testing data sets. At the output we obtain two databases; one acts as a repository of forward neural network sub-models of phase, and the other for forward neural network sub-models of amplitude. This process is repeated for all ten values of θ_q. The resulting databases can be represented as follows:

- Database-α^{q} → a repository for forward phase NN sub-models of the form

$$\underset{u}{\overset{\theta_q}{f}}_{1,ann}^{forward}$$

where,

$$\theta_q (\in \xi \cup \xi') = 0^{o}, 5^{o}, 10^{o}.....45^{o} \text{ and}$$

$u \in [1,9]$ Index to denote the SOM cluster number

- Database-$\beta^q \rightarrow$ a repository for forward amplitude NN sub-models of the form

$$ {}^{\theta_q}_{u} f^{forward}_{2,ann} $$

where,

$\theta_q (\in \xi \cup \xi') = 0^o, 5^o, 10^o .....45^o$ and

$u \in [1,9]$ Index to denote the SOM cluster number

Finally, we end up with 20 databases of NN sub-models. For instance if we consider $\theta_{q=4} = 15^o$, then following the above notation, corresponding forward model databases will be represented as Database-α^4 and Database-$\beta^{4.}$, namely

Database-α^4 is a repository for $[\ {}^{15^o}_{1} f^{forward}_{1,ann} \quad {}^{15^o}_{2} f^{forward}_{1,ann} \quad {}^{15^o}_{3} f^{forward}_{1,ann} \quad \ldots\ldots \quad {}^{15^o}_{9} f^{forward}_{1,ann} \ ] \ \forall q = 4$

and

Database-β^4 is a repository for $[\ {}^{15^o}_{1} f^{forward}_{2,ann} \quad {}^{15^o}_{2} f^{forward}_{2,ann} \quad {}^{15^o}_{3} f^{forward}_{2,ann} \quad \ldots\ldots \quad {}^{15^o}_{9} f^{forward}_{2,ann} \ ] \ \forall q = 4$

We will use these databases of neural network sub-models to augment the sample space of S-parameter databases - $h^i_{s\text{-}database}$ and $p^j_{s\text{-}database}$ ($\forall$ i = [1, 5] and j = [6, 10]), and also in the generation of inverse models.

Given that there is a minor modification in the execution of Algorithm-2 for this case, we can summarize this procedure in a pseudocode as Algorithm-8, adapted from Algorithm-2. Thus, for each value of q an S-parameter database is selected for phase and amplitude, and 9 SOM clusters are generated according to Algorithm-1. Thus 9 sub-neural network models are generated each for phase and amplitude.

Algorithm 8: Forward NN Sub-Model Generation using SOM given S-parameter databases for 10 oblique incidence angles.

Input: sample space of each of the 10 **S-parameter databases** represented by $h^{i}_{s\text{-}database}$ and $p^{j}_{s\text{-}database}$ $\forall$ $i \in [1, 5]$ and $j \in [6,10]$

Output: 2 databases containing neural network sub-models (for both amplitude and phase)

begin

for q =1 to 10

comment: Access sample space of S-parameter databases

comment: the selection of $h^{i}_{s\text{-}database}$ or $p^{j}_{s\text{-}database}$ as well i and j depend upon the value of q

if q = [1, 5]

comment: then select $h^{i}_{s\text{-}database}$ with i = q

elseif q = [6, 10]

comment: select $p^{j}_{s\text{-}database}$ with j=q

comment: Let SS^{q}_{phase} and SS^{q}_{amp} represent sample space of the q[th] S-parameter database for phase and amplitude respectively, such that,

comment: $x^{n}|_{\theta_q} = (a_1, a_2)^{n}|_{\theta_q}$, $y^{n}_{\theta_q} = \arg\{S_{21}\}^{n}_{\theta_q}$) $\in SS^{q}$ $\forall$ $n = 1, 2, \ldots \approx 3000$

comment: for each value of q an S-parameter database is selected for phase and amplitude and 9 SOM clusters are generated according to Algorithm-1 and thus 9 sub-neural network models are generated each for phase and amplitude

[Database-α^{q} , Database-β^{q}] = ***function call :*** *Algorithm-2 (*SS^{q}_{phase} , SS^{q}_{amp}*)*

end

stop

Next, we will look at training and test errors associated with the generation of each of these sub-models stored in databases α^q and β^q where, $q \in [1, 10]$. Average training and test error are computed for each of the 20 databases given by α^q and β^q

$$q = [1, 10] \text{ and } \therefore \theta_q (\in \xi \cup \xi') = [0^o, 5^o, 10^o, 15^o, 20^o, 25^o, 30^o, 35^o, 40^o, 45^o],$$

Databases considered: α^q (containing 9 phase sub-models) and,
β^q (containing 9 amplitude sub-models)

The average training and test error of 9 sub-NN models (both phase and amplitude) for all values of q are presented in Figure 4.3-1 and 4.3-2.

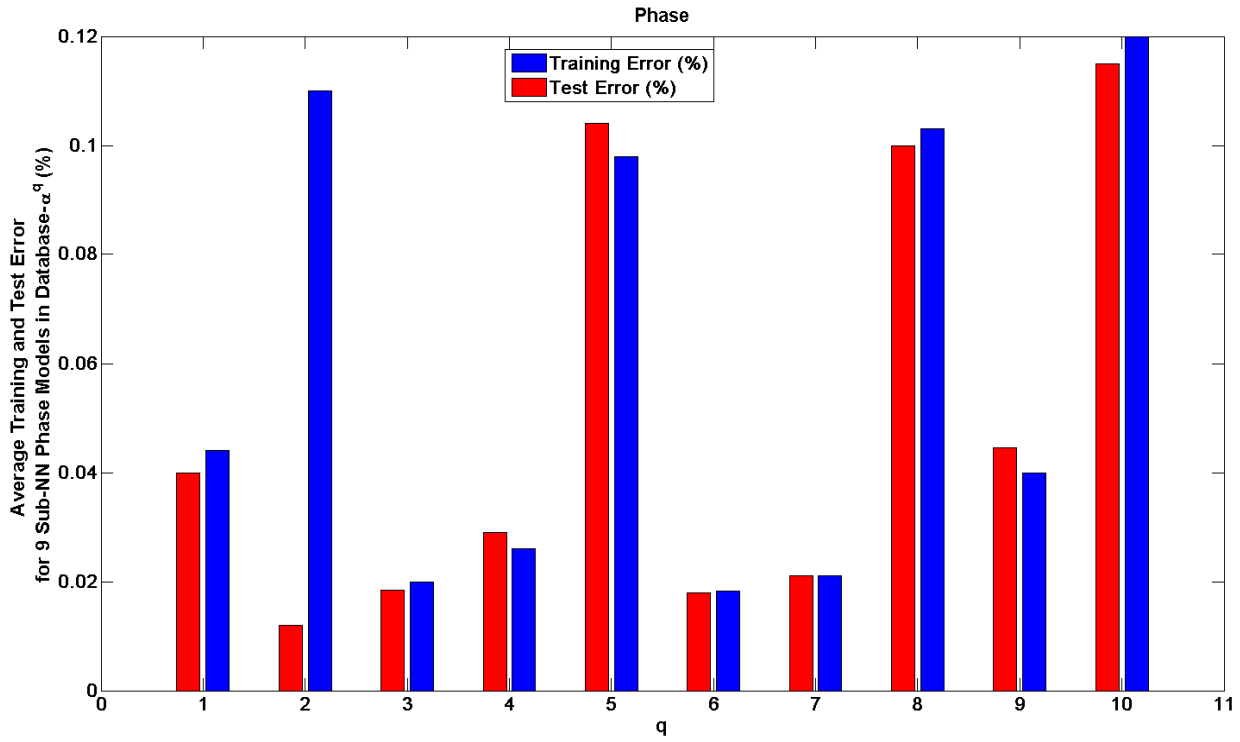


Figure 4.3-1 Average Training and Test Error for 9 Sub-NN Models for the Phase Stored in the Database-α^q with $q \in [1,10]$

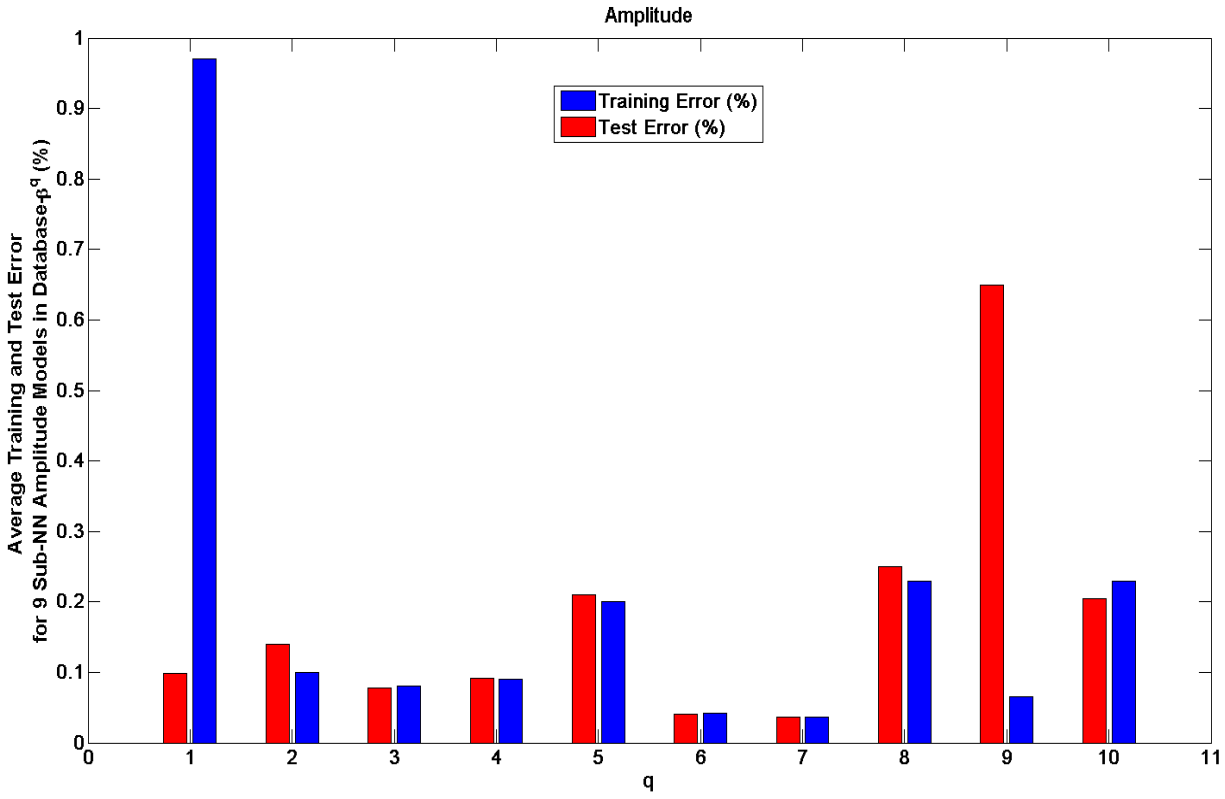


Figure 4.3-2 Average Training and Test Error for 9 Sub-NN Models for the Amplitude Stored in the Database-β^q with q $\in$ [1,10]

It can be observed that the training and test error for all the sub-NN models residing in databases are less than 1%, which is an acceptable value. It can be concluded that neural network characterization of S-parameter databases $h^i_{s\text{-}database}$ and $p^j_{s\text{-}database}$ is successful. Hence, we can create larger S-parameter sample spaces as compared to the sample spaces with 3000 points presented in Figures 4.2-1 to 4.2-5.

The next step is to use the databases generated in Algorithm-8 and create 250,000 samples S-parameter versus (a_1, a_2) for each of the angle of incidence value in set $\xi \cup \xi'$. To do this we use the identical procedure as we did in Algorithm-3 in Section 3.3 but in this case it has to be repeated for all the oblique incidence cases. This methodology is delineated as the pseudocode shown as Algorithm-9.

Algorithm 9: Output Extraction from Database of Forward Sub-Models Taking Oblique Incidence into Account

Input: $\left(a_1, a_2\right)_q^n$, where, q represents the databases of NN sub-models to be selected (α^q and β^q) based upon value of θ_q ($q \in [1,10]$) and n signifies the sample number such that $n \in [1, 250000]$

comment: Since we want to augment the sample space of each database to 250,000 samples (as we did in Section 3.3 in Figure 3.3-7 for normal incidence case), for each value of q we need to run iteratively run Algorithm-3 250,000 times and seek the transmission phase and amplitude corresponding to $\left(a_1, a_2\right)_q^n$

Output: $\arg\{S_{21}\}_q^n \; and \; |S_{21}|_q^n$

begin

for q =1 to 10

for n = 1 to 250,000

$[\arg\{S_{21}\}_q^n] = \text{Algorithm}-3(\ (a_1, a_2)_q^n, database-\alpha^q\ )$

$[\ |S_{21}|_q^n\ ] \quad = \text{Algorithm}-3(\ (a_1, a_2)_q^n, database-\beta^q\ )$

comment: a particular sub-NN model is selected by Algorithm-3 from among a set a sub-NN models in database-α^q and database β^q

end

end

stop

Implementation of Algorithm-9 gives an augmented sample space for each of the oblique incidence cases, which facilitates the inverse neural network generation process. Unit cell dimensions $(a_1, a_2)_q^n$ are used as shown in Figure 3.3-7, and corresponding transmission coefficient values are computed using Algorithm-9 for all the values of q. This way we now have 250,000 sample points for each oblique incidence case for $\theta_q \in \xi \cup \xi'$. If we look at Figure 3.3-8, which represents the augmented S-parameter sample space for normal incidence, similar plots can be constructed for all the oblique incidence cases. For example, databases shown in Figure 4.2-1 to Figure 4.2-5 with only 3000 points can be shown to have denser sample space with 250,000 points but to avoid repetition we won't provide those. The next step is to generate INN models.

4.4 INVERSE NERUAL NETWROK MODEL GENERATION FOR OBLIQUE INCIDENCE CASE

The transmitarray design process, when we take into account the oblique incidence in the generation of S-parameter database, remains identical to as described for normal incidence case in Figure 3.4-2. The exception here is that the process of forward neural network and inverse neural network generation needs to be repeated for all the selected discrete incidence angle value. If we look at the summary of the design process portrayed in Figure 4.4-1, in Section 4.2 we implemented Step 1 and Section 4.3 contains implementation of Step 2. After storing the forward neural network models for transmission phase and amplitude in databases the next task is INN modelling to be addressed here.

Following the lead from Section 3.4, where we generated INN model only considering normal incidence in the design process, here we will regard each incidence angle database created in previous Section separately and create an INN model for it and finally we will combine them using a specific algorithm (which will be described later). The reason behind the generation of INN's is to find the physical dimensions for a particular cell at a given location to achieve desired transmission phase. We need to invert the mapping between I/O for the database-α^q $\forall$ q=

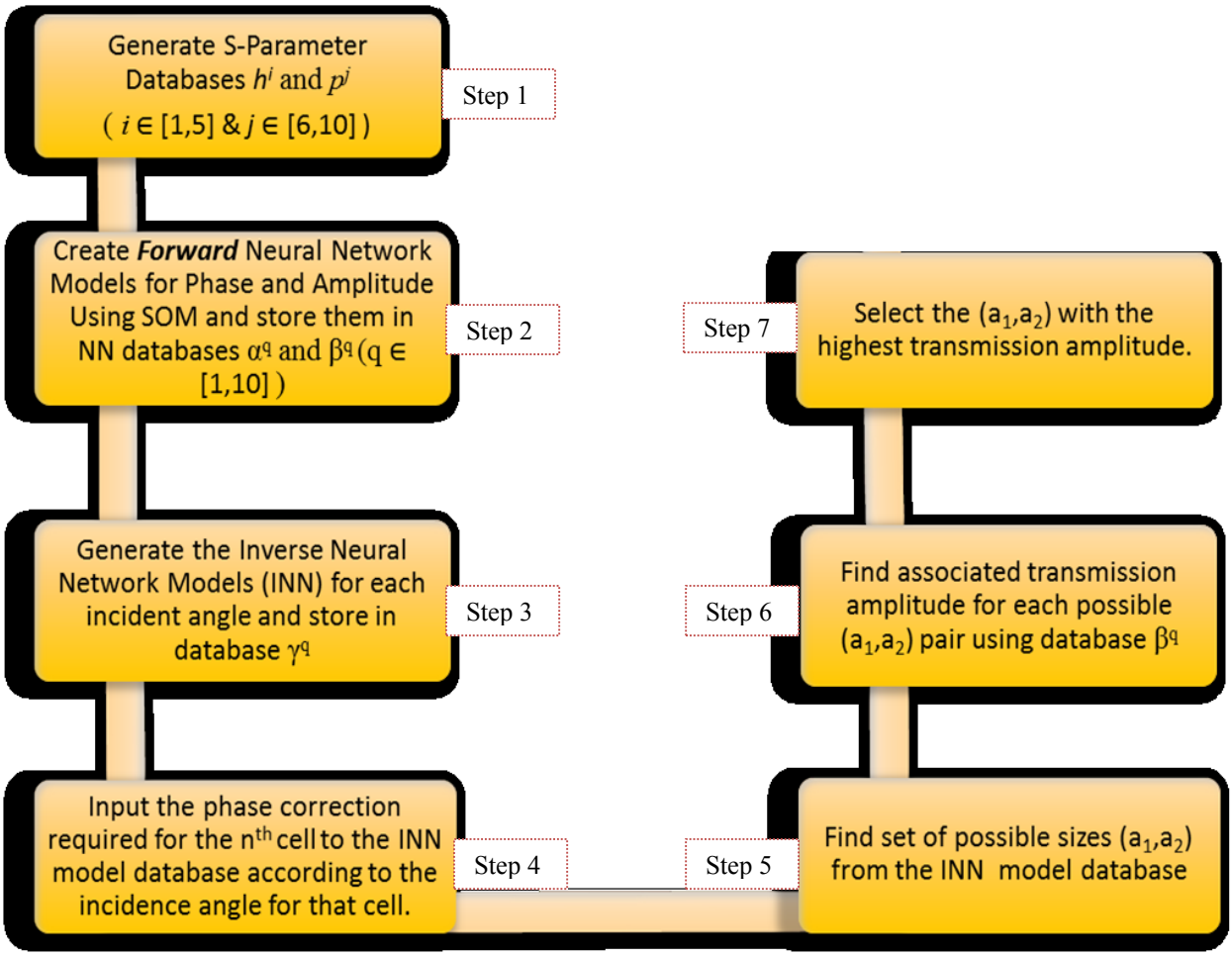


Figure 4.4-1 Overall Transmitarray Design Methodology Taking into Account the Oblique Incidence

1, 2 ….10 (defined in Section 4.3). Note that we will generate a new set of databases represented by 'γ^q' which will store INN sub-models for each value of q (based upon angle of incidence in consideration).In the hierarchical visualization of the overall design process in Figure 4.4-1, Step-3 is the essence of what we are going to describe in this Section pertaining to the INN model generation. Since, there are ten different incident angles under consideration as follows,

$$\theta_q \quad | \forall \quad q=[1,10]$$

If for the n^{th} cell located position (x_n, y_n) on the transmitarray aperture, the transmission phase required is $\arg\{S_{21}\}^n$ and the angle of incidence is θ_q (will remain constant for this cell), then inverse modelling problem can be expressed as:

$$[a_1 , a_2]^n = {}^{\theta_q} f_{ann}^{inverse}(\theta_q, \arg\{S_{21}\}^n) \quad \forall q = [1,10] \tag{4.4-1}$$

Equation 4.4-1 is essentially represents the inverse of the forward neural network function in equations 4.2-2 and 4.2-5 except that θ_q is fixed and $|S_{21}|$ is not required to be involved in the inverse neural modelling. It is only when we extract the output from the inverse model databases that $|S_{21}|$ will be considered to ensure that transmission amplitude is above a certain threshold for the selected dimensions. Now, 250,000 samples that were generated, using forward neural network model database-α^q, for each of the oblique incidence angle in Section 4.3 are utilized as a sample space for the generation of inverse models by flipping the inputs and outputs (similar to the case of normal incidence in Section 3.4 expressed mathematically in equations 3.4-1 to 3.4-8).

We 'SS^q' to represent sample space of the q^{th} INN in equation 4.4-1 and next, we implement Algorithm-5 to each of the ten sample spaces separately and generate corresponding INN database-γ^q. Algorithm-5 needs to be applied iteratively and this process is portrayed in Algorithm-10. Therefore, at the output we obtain ten different databases and we call them **INN databases** represented by,

$$database - \gamma^q = ({}^{\theta_q}_{1} f_{ann}^{inverse}, {}^{\theta_q}_{2} f_{ann}^{inverse}, {}^{\theta_q}_{1} f_{ann}^{inverse}, \ldots\ldots\ldots {}^{\theta_q}_{K_q} f_{ann}^{inverse}) \tag{4.4-2}$$

Each of the inverse databases store K_q inverse sub-models; the value of K_q isn't predetermined but it is determined by the algorithm and is contingent upon the sample space. The reason behind the generation of inverse sub-models is that we need to segment the sample space to preclude the possibility of multivalued solutions in the same sample space leading to a contradiction; the detailed theory behind segmentation was provided in Section 3.4.

After implementing Algorithm-10, we check the accuracy of the inverse models and to do so we look at the training and test error averaged over all of the inverse sub models for each of the ten INN database]. Figure 4.4-2 and Figure 4.4-3 show the test error and regression for each of the INN databases. Figure 4.4-2 (a) to Figure 4.4-2 (h) show linear regression and test

Algorithm 10: INN Model Generation for each of the Incidence Angle Values

Input: SS^q ($\forall q = [1,10]$) – sample space generated using forward NN in Algorithm-9 for each of the incidence angle with inputs and outputs in accordance with equation 4.4-1

Output: database-γ^q which stores inverse sub-models for each value of q and INN_error which represents the error associated with inverse neural network learning.

comment: we iteratively feed the q^{th} sample space (SS^q) to the Algorithm-5 and seek the inverse sub models generated using multi-layer perceptron neural network topolgy

```
begin

    for q =1 to 10
            [database-γ^q, INN_error ] = Algorithm-5( SS^q )
    end

stop
```

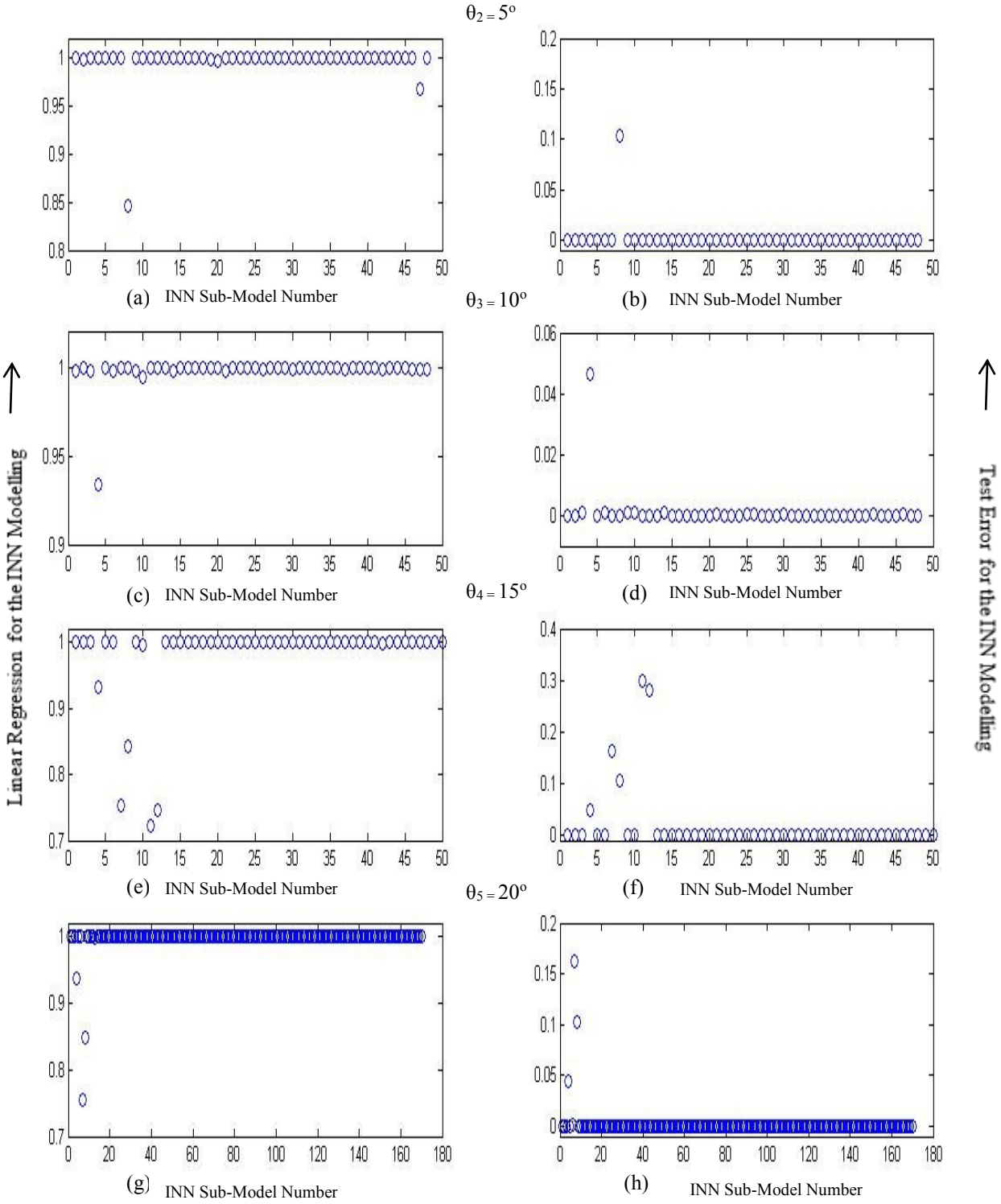


Figure 4.4-2 Linear Regression and Test Error of Inverse Sub-Models for each INN Database γ^q with Angle of Incidence (θ_q) with q = 2,3,4 and 5

error. Note that plots for q=1 are presented, since, for q=1; $\theta_q = 0^o$, which is the case of normal incidence and plots for this case are already provided in Figure 3.4-9 and Figure 3.4-10. Also, it is important to note that the number of sub-models (K_q) vary for each incidence angle case. Similarly, we will provide plots for linear regression and percent average test error for q=6, 7, 8 and 9 in Figure 4.4-4 and for q= 10 in Figure 4.4-3. Now, if we analyze Figure 4.4-2, Figure 4.4-3 and Figure 4.4-4, for most of the INN sub-models linear regression is very close to the ideal value of unity. Alongside, test error is well below the acceptable 1% mark for most of the sub-models. To summarize these data we take the average over the entire INN sub-model values for each of the incidence angle INN databases and the corresponding plots are provided in Figure 4.4-4 and Figure 4.4-5.

Average of the linear regression for all the sub-models is given in Figure 4.4-5 (a) for each INN database and it can be seen that the for all the cases linear regression is approximately equal to 1 except for q=4 (INN database-γ^q) where it is 0.98 which is still acceptable. Likewise, if we observe Figure 4.4-5 (b) %average test error is below the acceptable threshold 1% except for q=4 where it is approximately 1.8%.

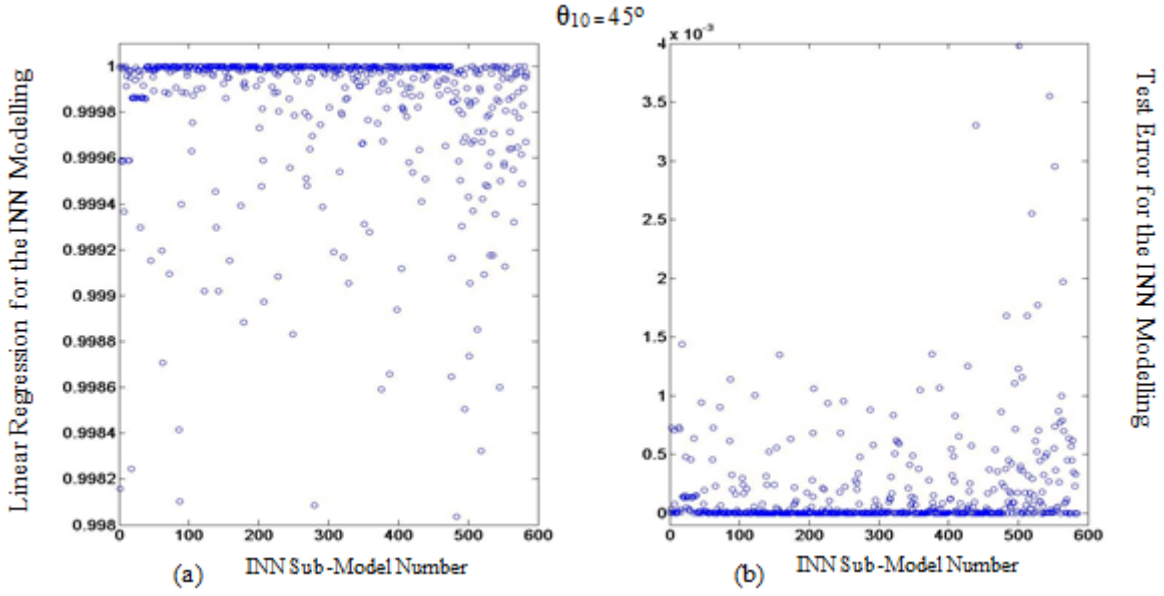


Figure 4.4-3 Linear Regression and Test Error of Inverse Sub-Models for INN Database γ^{10} with Angle of Incidence ($\theta_{q)} = 45^o$ with q = 10

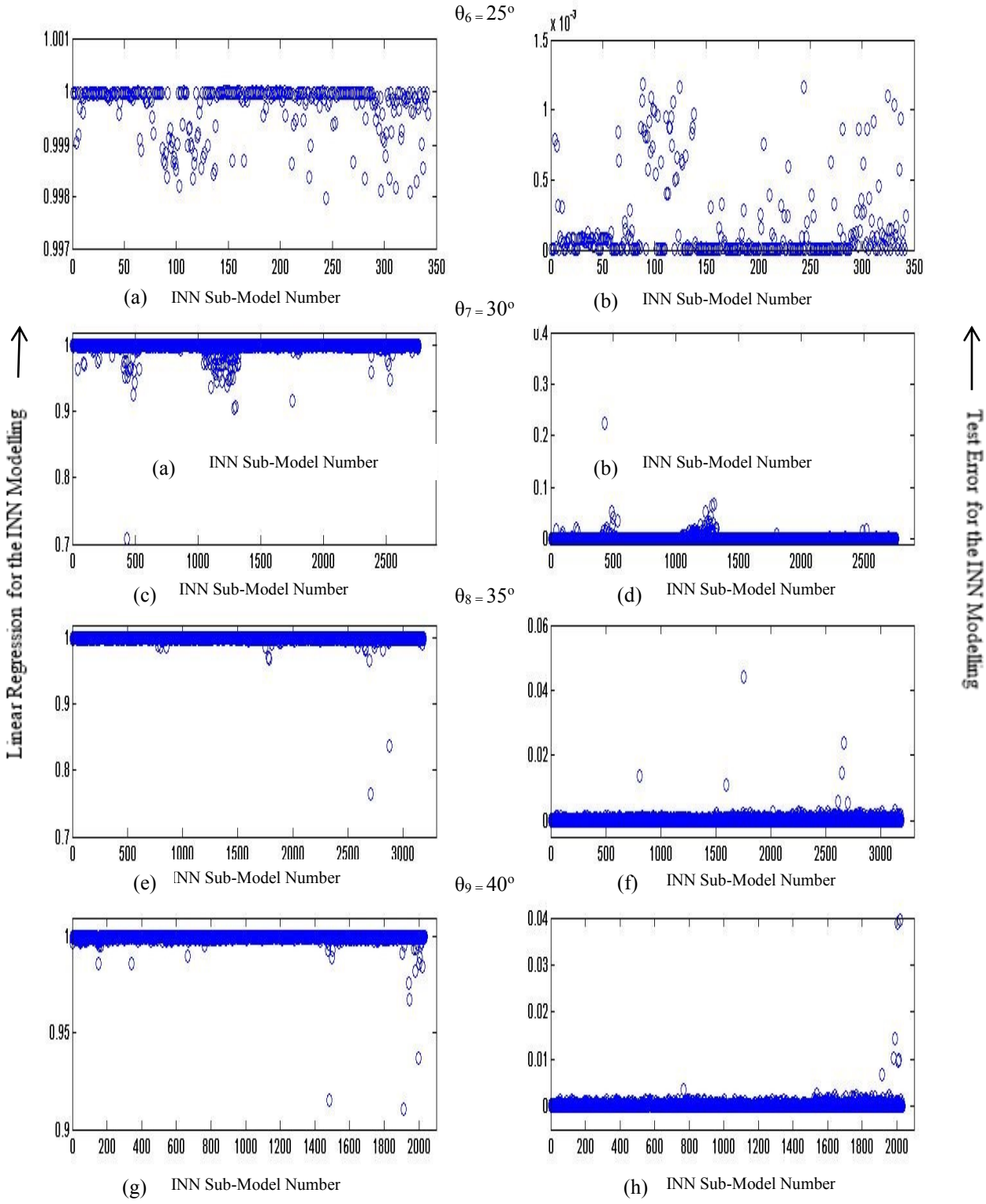
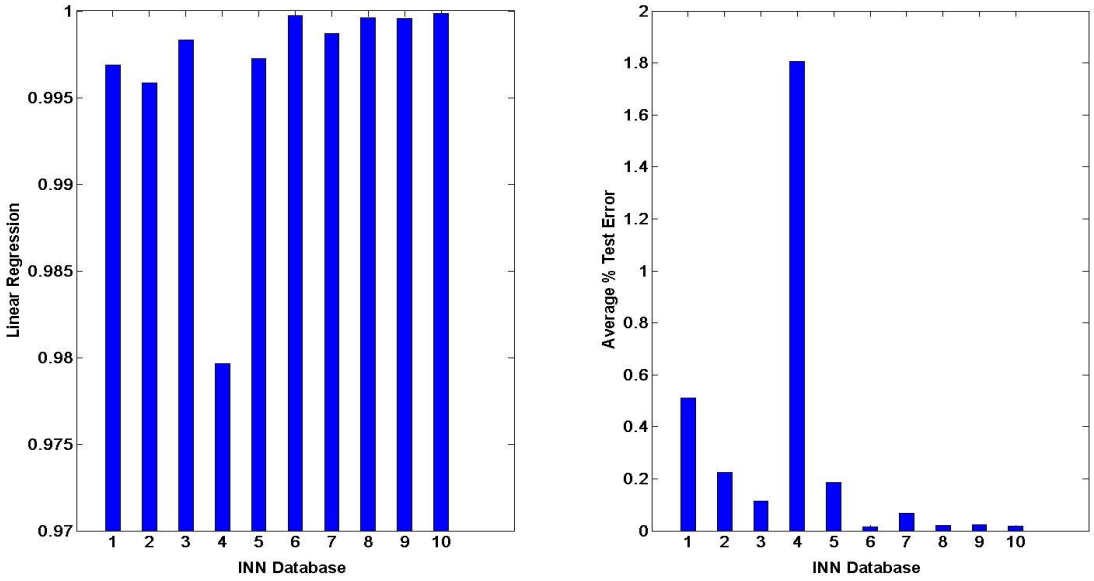


Figure 4.4-4 Linear Regression and Test Error of Inverse Sub-Models for each INN Database γ^q with Angle of Incidence (θ_q) with q = 6, 7, 8 and 9



(a) (b)

Figure 4.4-5 (a) Average Linear Regression and (b) Average %Test Error of Inverse Sub-Models average for each INN Database γ^q with Angle of Incidence

Having established the accuracy of INN modelling, now, we will look at the methodology and algorithm to extract output from the INN databases constructed in this Section.

4.5 OUTPUT DERIVATION FROM THE INN DATABASES

In this Section we will explain the methodology employed in extracting the output from the INN databases. It is basically the implementation of Step-4 to Step-7 in the process flow described in Figure 4.4-1. We will input the desired transmission phase at a given location for a particular cell as well the incidence angle for that cell to the algorithm and the algorithm in turn will select the appropriate INN database based upon the incidence angle and finally, following the same procedure as described in Algorithm-6 in the Section 3.5, the optimum dimensions will be sought at the output. The issue here is how do we decide which INN database is to be selected? Since, we only have ten databases corresponding to discrete angles of incidence i.e. for 0^o, 5^o, 10^o, …..45^o and quantities in between these discrete values are missing. So, if for a given

cell the angle of incidence turns out to be absent from this set, we select the nearest value from the set $\xi \cup \xi'$.

This procedure is delineated in Algorithm-11 along with the overall process employed to compute the required dimensions. In this algorithm we input the transmission phase $\arg\{S_{21}\}^n$ required at the center of the n^{th} cell as shown in Figure 4.5-1. θ_n is the angle of incidence for the n^{th} cell under consideration and we find the nearest angle of incidence value for which INN database is available, let's say θ_q. Therefore, q^{th} INN database is used for determining n^{th} cell dimensions using Algorithm-6. Along with that we also pass the forward neural network databases α^q and β^q as an argument to A;lgorithm-6 from Algorithm-11 which are also very vital in the computation.

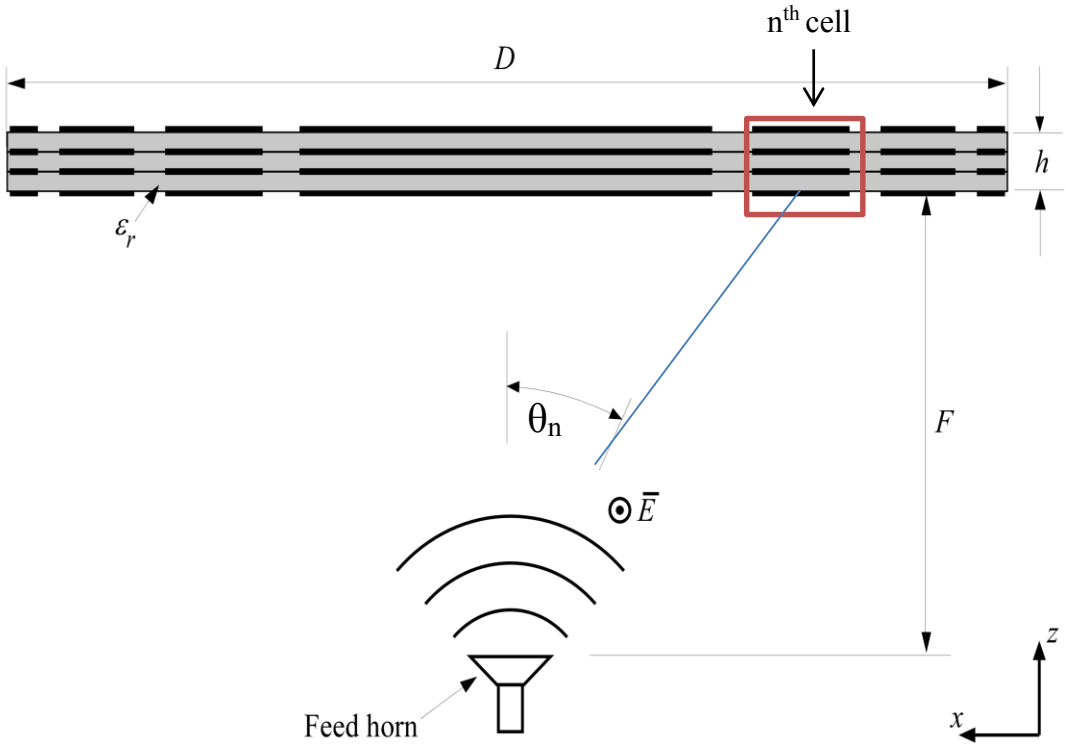


Figure 4.5-1 Side View of the Transmitarray Fed with Horn Antenna Portraying Plane Wave Incident on the n^{th} Cell at θ_n

Algorithm 11: Output Extraction from INN Databases for a Given Transmission Phase value at a particular Cell over the Aperture of the Transmitarray

comment: to determine the patch dimensions for the n^{th} element on the transmitarray aperture for a specified transmission phase and maximum transmission amplitude with θ_n as the angle of incidence for that cell

comment: Database-α^q → database of forward neural network sub-models for **phase** generated in Algorithm-8
Database-β^q → database of forward neural network sub-models for **amplitude** generated in Algorithm-8
Database-γ^q → database of inverse neural network sub-models generated in Algorithm-10

Input: Transmission Phase ($\arg\{S_{21}\}^n$) required at the center of n^{th} element on the tranmistarray aperture and angle of incidence θ_n at the center of the n^{th} cell.

Output: Patch Dimensions $[\ a_1,\ a_2\ ]^n$ for the current element that result in the given transmission phase **and** trnmission amplitude above the threshold

begin

comment: first step is to approximate incidence angle θ_n to the nearest available discrete angle of incidence values $\theta_q \in \xi \cup \xi'$

for u = 1 to number of elements in set $\xi \cup \xi'$ (which is 10)

minimize ($|\ \theta_n - \theta_q\ |$)

end

comment: select θ_q such that above criteria is met and therefore, INN database of interest will be **database-γ^q** and pass it to Algorithm-6 along with forward sub-model databases.

$[\ a_1, a_2\ ]^n$ = Algorithm-6 ($\arg\{S_{21}\}^n$, database-γ^q, database-α^q, database-β^q)

stop

4.6 INN DATABASE OUTPUT VERIFICATION

To check the accuracy of the INN modelling output, we use a fictional set of transmission phase values assuming these are the required transmission phases at various cell centers on the aperture of transmitarray. This procedure is identical to the one employed in Section 3.6 and it is summarized in Figure 4.6-1. We will implement this procedure through an algorithm on the same note as Algorithm-7 with minor variations. Let's set F consists of transmission phase values required at the center of various cells along with the incidence angle corresponding to each cell. We will feed the elements in F to Algorithm-11 and seek the output dimensions and in turn feed those output dimensions to the forward neural network databases and compute the transmission phase. Then the input transmission phase is compared with this output and error is computed.

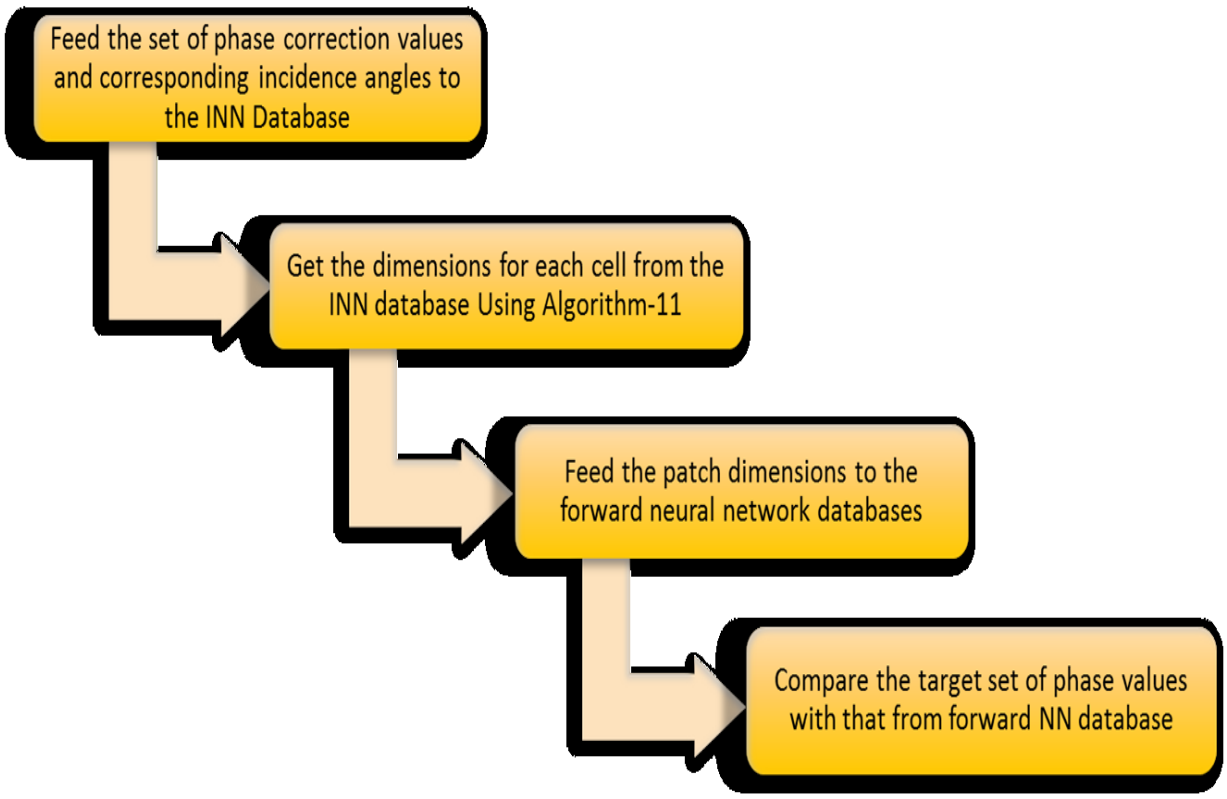


Figure 4.6-1 Summary of the Approach Employed to Authenticate the Output of INN Databases

This process is carried out in Algorithm-12 in the following manner:

The elements in F are defined as,

$$(\arg\{S_{21}\}^{u}, \theta_{u}) \in F \quad \forall u = [1, 210] \tag{4.6-1}$$

Where, $\arg\{S_{21}\}^{u}$ is the required transmission phase and θ_u is the corresponding incidence angle for that particular cell. All these elements of F are fed to Algorithm-11 iteratively and in each iteration we have to choose the closest value of incidence angle θ_q for which we have INN database available and we do this by minimizing the difference between $\theta_{q \text{ and }} \theta_u$. This step is already performed in Algorithm-11 and it is done as follows:

$$\begin{aligned} &\{\theta_q \,|\, \min(|\theta_u - \theta_q|)\ \} \\ &\therefore \theta_q \in \xi \cup \xi' \end{aligned} \tag{4.6-2}$$

Next we feed u^{th} element of F to INN database-γ^q assuming ${}^{\gamma^q} f_{ann}^{inverse}$ represents this database of inverse sub-models, therefore output dimensions $(a_1, a_2)^u$ are calculated as,

$$[a_1, a_2]^u = {}^{\gamma^q} f_{ann}^{inverse}(\arg\{S_{21}\}^{u}, \theta_u) \tag{4.6-3}$$

Now, to check if the output $[a_1, a_2]^u$ by the γ^q in equation 4.6-3 is accurate, we feed this output to the forward databases α^q and β^q and seek the output phase and amplitude as follows:

$$\arg\{S_{21}\}_{ANN}^{u} = {}^{\alpha^q} f_{ann}^{forward}([a1, a2]^u) \tag{4.6-4}$$

$$|S_{21}|_{ANN}^{u} = {}^{\beta^q} f_{ann}^{forward}([a1, a2]^u) \tag{4.6-5}$$

We need to compare $\arg\{S_{21}\}_{ANN}^{u}$ with the original input phase $\arg\{S_{21}\}^{u}$ and compute the associated error E_{ANN}^{u} as follows:

$$E^{u}_{ANN} = \left[\frac{\arg\{S_{21}\}^{u}_{ANN} - \arg\{S_{21}\}^{u}}{\max(\arg\{S_{21}\}^{u}) - \min(\arg\{S_{21}\}^{u})} \right]^2 \tag{4.6-6}$$

The composite error for the entire set F is calculated by taking average of error associated with all elements.

$$E_{ANN} = \frac{1}{n(\mathrm{F})} \left[\sum_{u=1}^{n(\mathrm{F})} E^{u}_{ANN} \right]^{\frac{1}{2}} \tag{4.6-7}$$

Now, let's compare the desired transmission phase values in F and values actually rendered on the basis of the dimensions computed using INN modelling stored in U_1 by plotting Linear Regression Curve in Figure 4.6-2. We obtained a linear regression R to be 0.997 which is an acceptable value and the overall error E_{ANN} = 0.12 %.

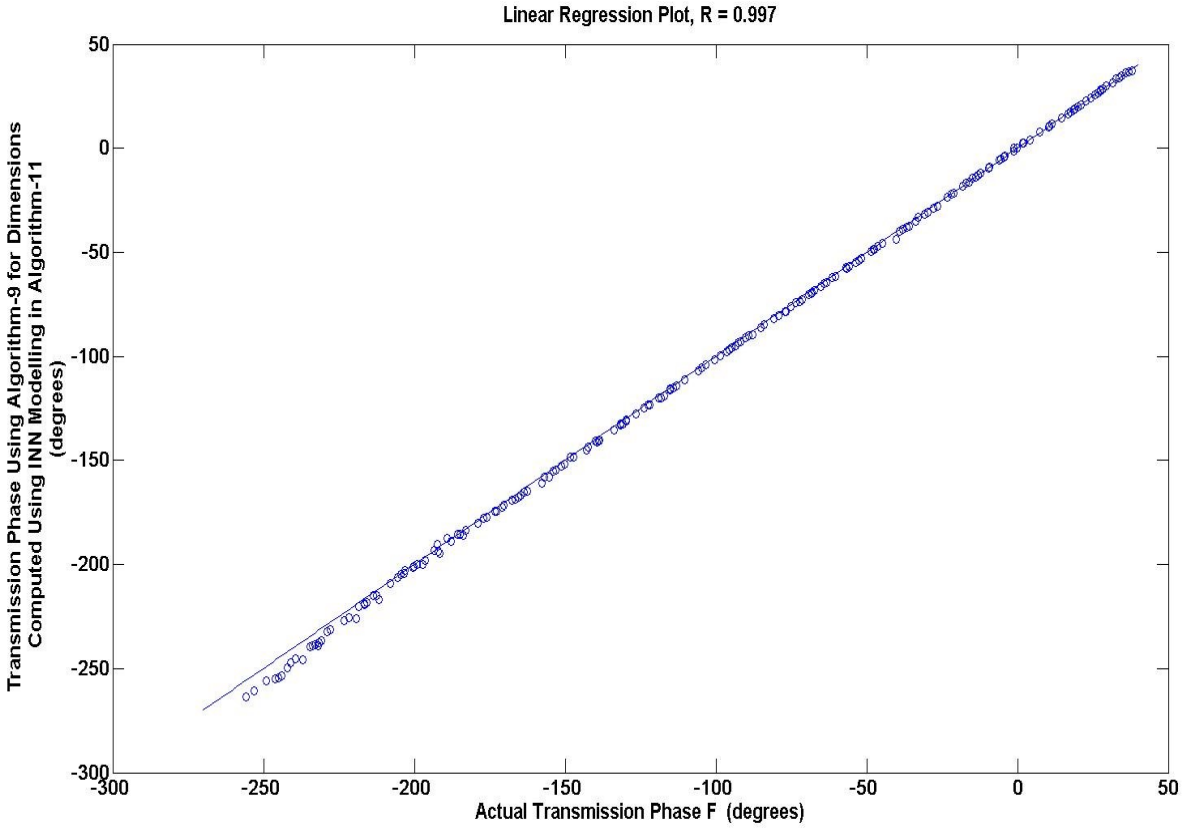


Figure 4.6-2 Linear Regression Plot Comparing Desired Phase Values in Set F and Phase Output given by Forward Neural Network Database Using Algorithm-11 in Set U_1

Algorithm 12: INN Output Authentication Using Inverse and Forward NN Databases in the Oblique Incidence Case

comment: Set 'F' is created containing 210 elements representing fictional phase values required for different transmitarray elements

Input: Set 'F' of Fictional Transmission Phase values ($\arg\{S_{21}\}^u \quad \forall n = 1,2,3$) presumably required at the center of various elements on the transmitarray aperture.

Output: i) Error 'E_{ANN}' the Normalized Euclidean Norm of the difference between output phase values from forward neural network and actual input phase values in set F

ii) Vectors of phase values corresponding to output dimensions computed by Algorithm-11.

begin

k=u

while u <= cardinality of set F

[(a_1, a_2), q]^u = ***function call*** - *Algorithm-12*(***argument*** = $\arg\{S_{21}\}^u$) , θ_u)

comment: such that, $\arg\{S_{21}\}^u \in F$

function call- *Algorithm-9(* ***argument*** = [a1 a2]^u, q)

comment: store output $arg\{S_{21}\}^u_{ANN}$ in vector 'U_1' and output $|S_{21}|^u_{ANN}$ in vector 'U_2

comment: compute E_{ANN}

$$E^u_{ANN} = \left[\frac{\arg\{S_{21}\}^u_{ANN} - \arg\{S_{21}\}^u}{\max(\arg\{S_{21}\}^u) - \min(\arg\{S_{21}\}^u)} \right]^2$$

u = u+1

end

$$E_{ANN} = \frac{1}{n(\mathrm{F})} \left[\sum_{u=1}^{n(\mathrm{F})} E^u_{ANN} \right]^{\frac{1}{2}},$$

Return: E_{ANN}, U_1, U_2

stop

Finally, we need to ensure that transmission phase associated with each cell's dimension $|S_{21}|^{u}_{ANN}$ stored in U_2 is above -3dB threshold so that there is significant transmission. Figure 4.6-3 represents the transmission coefficient amplitude (U_2) vs phase (U) plot and it can be seen that amplitude is well above -3dB. Therefore, we can conclude that the output cell dimensions from the INN database, for a given set of desired transmission phase, are accurate and satisfy the criteria for optimum performance by a transmitarray.

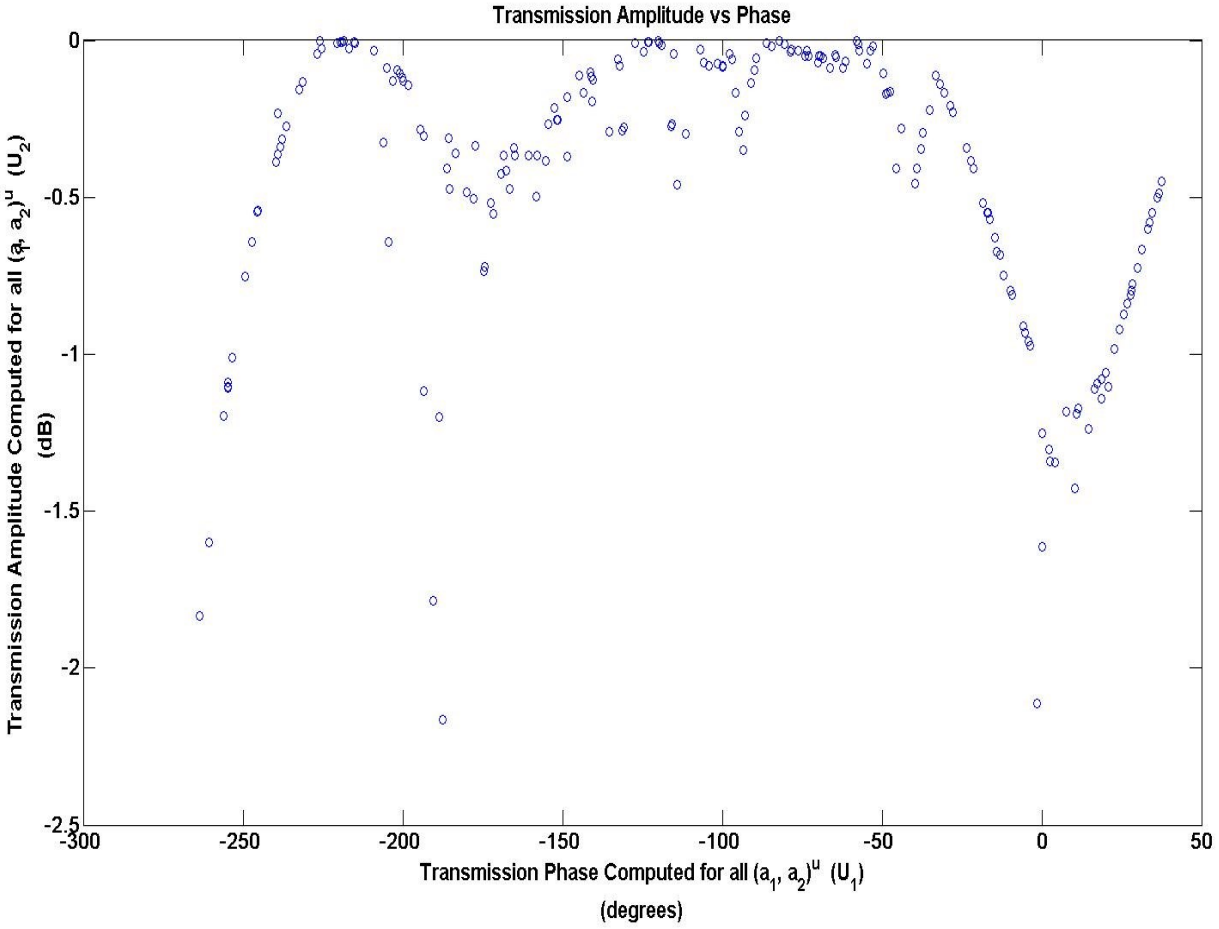


Figure 4.6-3 Transmission Coefficient Amplitude Output vs Phase Output for Various $[a_1\ a_2]^u$ Computed Using the INN database

4.7 CONCLUDING REMARKS

The objective of this chapter is to investigate how the oblique incidence affects the transmission coefficient for the transmitarray unit cell and after detailed analysis we established that the variation of the transmission phase w.r.t. the incidence angle is not consistent (for fixed cell dimensions). This is the major contribution of this Chapter, as to our knowledge in the current literature, this variation of transmission phase versus incidence angle hasn`t been investigated and included in the transmitarray design process. In Section 4.2, we generate the S-parameter design database by considering the oblique incidence, and store the transmission phase and amplitude in separate databases, one for each of the discrete incidence angle values. In Section 4.3 we characterize these databases using ANNs to augment the sample space, which is vital for the efficient INN model generation. Second major contribution in this chapter is the application of the INN modelling to each of the oblique incidence databases individually and consequently combining these INN models to generate the final design. This task is undertaken in Section 4.4, and we provide detailed pseudo-codes for all the algorithms used which are not available in the literature. Another complication of this procedure, that is when we consider oblique incidence in the INN model generation, is how we merge all the individual INN models, and this is done successfully in Section 4.5 and in Section 4.6 we present the check results to determine the accuracy of this procedure. We successfully demonstrated that incidence angle can be included in the design database to generate INNs and satisfactory results can be achieved in terms of the performance of this approach.

CHAPTER 5
Application of Inverse Neural Networks in Transmitarray Antenna Design

5.1 PRELIMINARY REMARKS

The goal of this thesis has been the development of a fast means of designing transmitarray antennas. This required the development of approaches for the computationally efficient construction of design databases and, most importantly, associated methods for the inversion of such databases during the actual design procedure. This has all been successfully achieved in Chapters 3 and 4, which describe the principal contributions of the thesis. Although the thesis is of its very nature a theoretical one, the foundations that it lays are for use in antenna design. It is thus appropriate to demonstrate the outcome of the use of the processes in Chapters 3 and 4.

5.2 TRANSMITARRAY PROTOTYPE DESIGN USING INN METHODOLOGY

Now that we have generated and validated the INN models for the design of transmitarray antenna in Chapter 3 and Chapter 4, next task is to design an actual prototype to further establish the accuracy and robust nature of the proposed design approach as compared to other available techniques in the literature. Note that in this chapter the prototype that we will design is based upon the methodology adopted in Chapter 3 i.e. in the process we only consider the normal incidence case and ignore the fact that cells lying far away from the center experience oblique incidence. But still we got satisfactory results from the final design which will be apparent in next Sections. The overall design procedure can be summarized as shown in Figure 5.2-1.

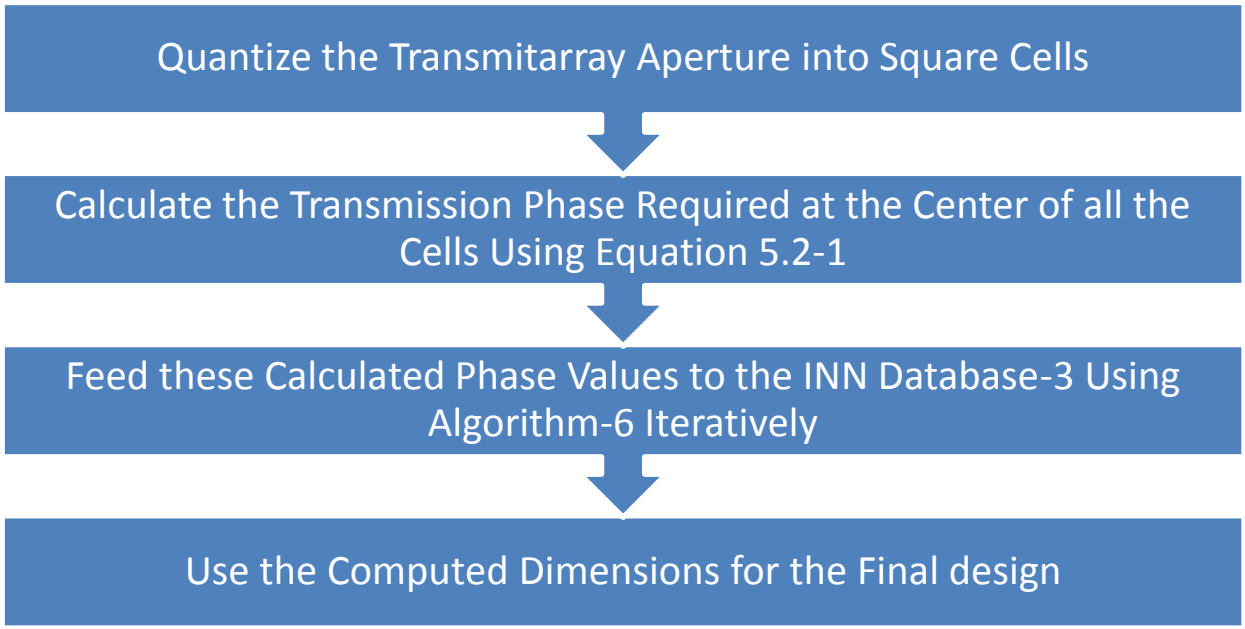


Figure 5.2-1 Transmitarray Prototype Design Procedure

5.2.1 **Transmitarray Quantization and Physical Specifications**

In our work we designed a 3-layered circular transmitarray with diameter 'D'. The very first step in the design procedure is to discretize the circular aperture of the tranmitarray into uniform square cells of side 's' (the cell size selection criteria is addressed in the Section 3.2.2). In this work we used D to be equal to 152.4mm and s to be 3mm. This way we end up with 2025 cells. Figure 5.2-2 represents this discretization and the center of the n^{th} cell with coordinates (x_n, y_n) is highlighted. Note that since we are quantizing the aperture into square cells, therefore, d_x is equal to d_y.

Now, Figure 5.2-2 is only a model to represent the quantization process but the actual quantization is represented in Figure 5.2-3 where red circles are the corners of each cell and green dots are the corresponding centers. The inset shows the magnified version of one of the cells with white region being the dielectric and black region is the metallic patch.

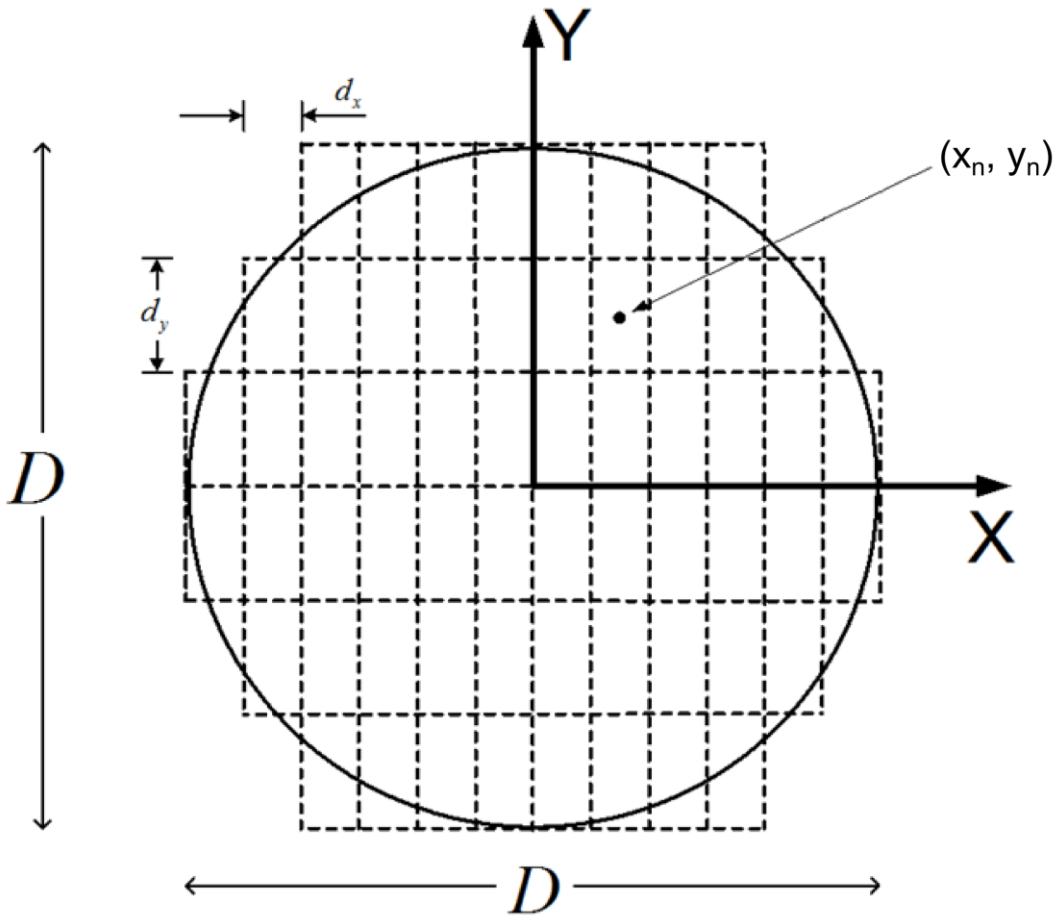


Figure 5.2-2 Quantized Transmitarray Aperture into Cells

Next, we will discuss the transmitarray parameters we used for the design. The frequency of operation 'f' was chosen to be 30 GHz. The rationale behind this choice was that we wanted the transmitarray to exhibit significant improvement over the conventional technologies in terms of size, profile, weight as well as the cost. The chosen frequency lies in the Ka band and at this chosen frequency above criterion were met. For example, in this band the size is not too large, a good trade off can be achieved between lower material cost and higher concomitant parts' cost.

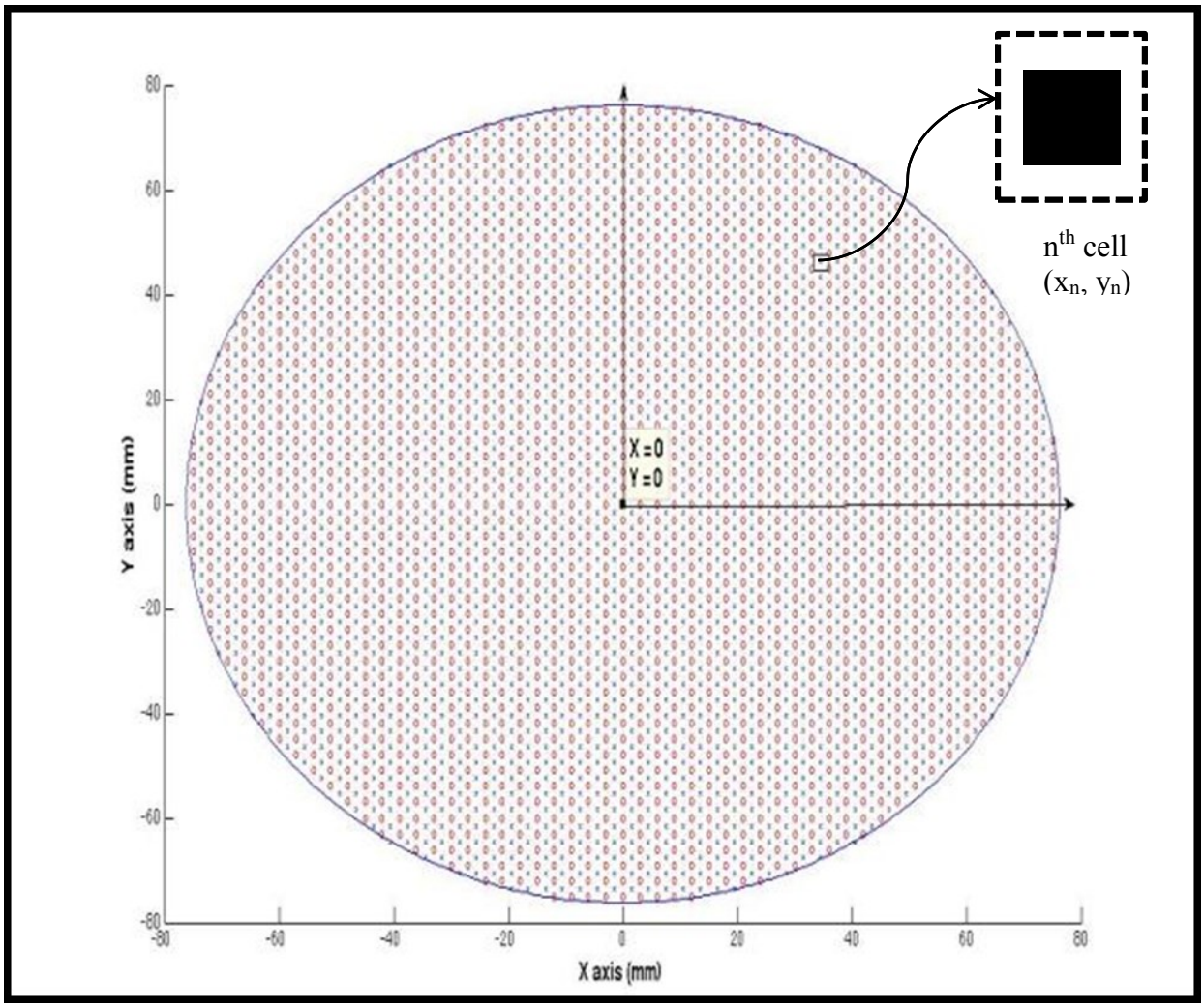


Figure 5.2-3 Quantized Aperture of the Transmitarray Under Design with One of the Cell Highlighted

Also, this band includes the satellite communication (SATCOM) bands; hence, this transmitarray can be utilized in such applications.

Secondly, the size of the square cells ‘s’ on the aperture of transmitarray is also an important parameter and its selection is based upon several factors that we are going to discuss now. If we recall from Chapter 3 and Chapter 4, while generating the design databases, we assumed that each cell on the transmitarray aperture is residing in an infinite 2D periodic structure. We used appropriate electromagnetic boundary conditions around a single unit cell thus size of the

problem is truncated tremendously as the unknowns in the electromagnetic simulaiton would be largely reduced. But to facilitate that, following condition needs to be met:

$$s < \frac{\lambda_0}{1+|\operatorname{Sin}\theta_{max}|} \tag{5.2-1}$$

s being the unit cell size, λ_0 is the free space wavelength (which is 10 mm corresponding to the frequency of operation f = 30 GHz) and θ_{max} is the maximum angle of the incidence which will be for the cell located farthest from the center of the transmitarray. Therefore, for the case when $\theta_{max} = 0^o$ i.e. normally incident wave, s < λ_0 whereas for $\theta_{max} = 90^o$, s < λ_0 /2. In the periodic structure analysis carried out in this work, we have used the lower threshold which is s < λ_0 /2 to account for the fact that cells lying far away from the center (given that we are considering an infinite periodic structure), the incident wave would have a large non-zero incident angle. This is the condition where a single main beam will be allowed in the physical space. We chose s = 3 mm which meets the criteria specified in equation (5.2-1).

The dielectric constant 'ε_r' for the material used is 2.2 and diameter 'D' of the aperture (Figure 5.2-2) is 152.4 mm (~ 15λ). As mentioned before, N = number of cells on the aperture is equal to 2025. All the specifications are summarized in Table 5.2-1.

Table 5.2-1 Transmitarray Specifications

Aperture:	Circular
Cell Size (s):	3 mm
Diameter (D):	152.4 mm ~ 15λ
Number of Elements (N):	2025
Frequency of Operation:	30 GHz
Relative permittivity of the dielectric:	2.2

Now, we will discuss the selection criteria for D. Actually the primary objective was to increase the phase range available and hence we had to optimize value of D for which maximum phase

range is possible. To do this we investigated the relationship between Phase Range and D in Figure 5.2-4. It can be observed that if we select diameter to be above 100 mm then for all F/D cases we have similar Phase Range values. Now, to further narrow down the value of D, we plotted Phase Range versus F/D for different values of D as shown in figure 5.2-5 and deduced that any value of D >120 mm will give satisfactory performance and finally, we chose D =152.4 mm keeping in mind that larger aperture would also ensure higher antenna gain.

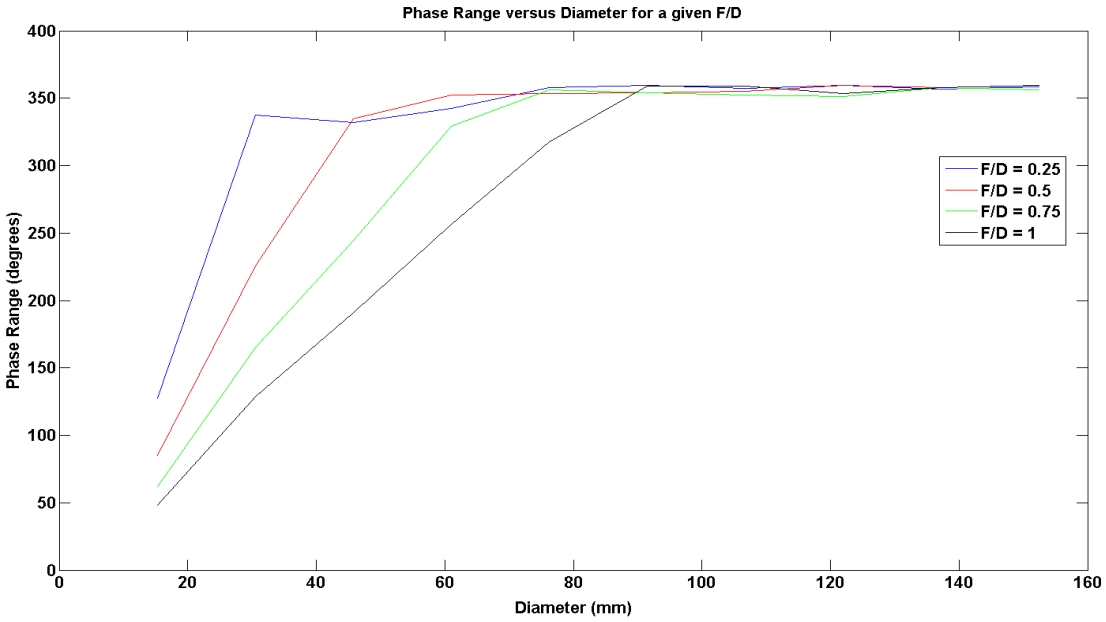


Figure 5.2-4 Phase Range versus Diameter D at Various F/D

5.2.2 Calculation of Required Transmission Phase at Each Cell

Next step is to find the transmission phase (arg{S_{21}}) required for all the cells on the transmitarray aperture. We use the equation 5.2-2 (derived in Section 2.2 as equation (2.2-56)) to determine the transmission phase required for the n^{th} cell. Although it was mentioned in Section 2.2 but we will remind the reader again that we intend to generate pencil beam radiation pattern on the output of the antenna and to do so it is necessary that all the cells have a uniform phase. We iterate through all the cells in the transmitarray and solve equation (5.2-2) for the required phase (implemented in Algorithm-13 in Section 5.2-3).

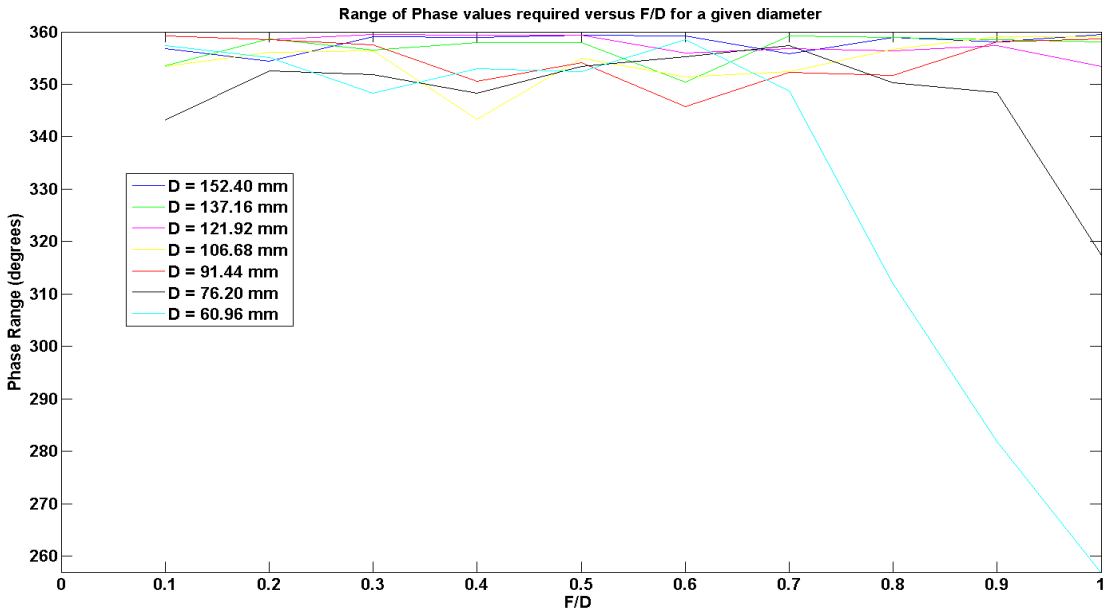


Figure 5.2-5 Phase Range versus Diameter F/D at Various D

$$\arg\{S_{21}\}^n = -\frac{2\pi F}{\lambda_0}\left\{\frac{\cos\theta_n - 1}{\cos\theta_n}\right\} + \Psi_{offset} \qquad \forall\, n = 1, 2, 3.....N = 2025 \qquad (5.2\text{-}2)$$

In the above equation F is the focal length, θ_n is the angle of incidence for the nth cell, λ_0 is the free space wavelength (refer to Figure 2.2-1) and Ψ_{offset} is a constant offset that can be assigned any phase value but it has to be consistent for all cells. In our design we consider two cases one with F = D and another with F= D/2 i.e. we shift the position of the feed horn antenna.

Let`s have a look at the transmission phase values required at the center of each of the 2025 cells. Figure 5.2-4 and Figure 5.2-5 represent these over the aperture. The range of the phase values required (F/D = 1) is 358.2^0 (from -320^0 to 38.2^0) but due to the physical limitations the actual transmission phase range available is 298^0 (from 38.2^0 to -260^0) and hence, we need to make approximations at the extremes of the phase values.

$$\arg\{S_{21}\}^{max}_{required} \text{ (Max. Phase Correction required)} = 38.10^0 \quad (5.2\text{-}3)$$

$$\arg\{S_{21}\}^{max}_{available} \text{ (Max. Phase Correction available)} = 38.10^0 \quad (5.2\text{-}4)$$

$$\arg\{S_{21}\}^{min}_{required} \text{ (Min. Phase Correction required)} = -321.90^0 \quad (5.2\text{-}5)$$

$$\arg\{S_{21}\}^{min}_{available} \text{ (Min. Phase Correction available)} = -260.00^0 \quad (5.2\text{-}6)$$

Due to limited phase range available, as it is evident from the equations (5.2-3) to (5.2-6), approximations made at the phase extremes are as follows;

If for the n^{th} cell,

$$\arg\{S_{21}\}^n < \arg\{S_{21}\}^{min}_{available} \quad (5.2\text{-}7)$$

and if,

$$\frac{(\arg\{S_{21}\}^{min}_{available} + \arg\{S_{21}\}^{min}_{required})}{2} \leq \arg\{S_{21}\}^n < \arg\{S_{21}\}^{min}_{available} \quad (5.2\text{-}8)$$

then,

$$\arg\{S_{21}\}^n = \arg\{S_{21}\}^{min}_{available} \quad (5.2\text{-}9)$$

otherwise,

$$\arg\{S_{21}\}^n = \arg\{S_{21}\}^{max}_{available} \quad (5.2\text{-}10)$$

Equations (5.2-6) to (5.2-10) are used to approximate for the transmission phase values which are not available in the design database. This procedure is implemented in Algorithm-13 (described in Section 5.2-3) but let's have a look the behavior of transmission phase required when plotted at the center of each cell; it is shown in Figure 5.2-6 (for F/D =1) and Figure 5.2-7 (for F/D = 0.5). Each colored ring represents a constant phase and it can be observed that when the feed is closer to the aperture i.e. when F/D = 0.5, the density of the rings is high as compared

to the F/D=1 case because a larger phase variation is required for this among the adjacent cells for optimum transmitarray operation.

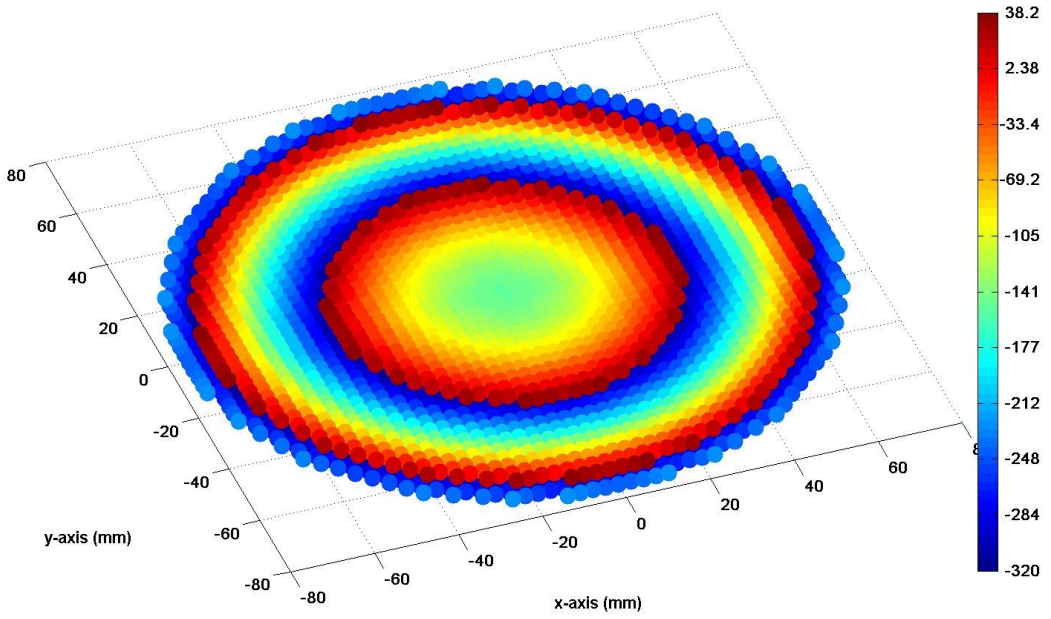


Figure 5.2-6 Pattern of the Transmission Coefficient Phase Required at each Cell for F/D =1

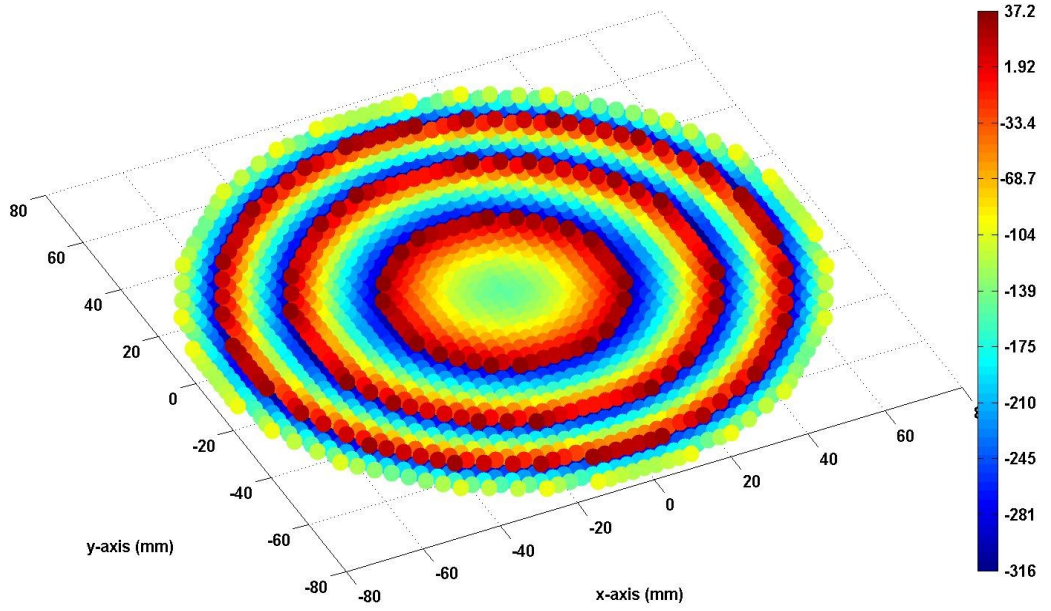


Figure 5.2-7 Pattern of the Transmission Coefficient Phase Required at each Cell for F/D =0.5

5.2.2 Algorithm Used for the Final Design

Once we have determined the phase required at each cell location the next step in the design process is to feed these phase values to the Algorithm-6 and seek the corresponding dimensions for the cell metallic patches. Note that the phase value required as shown in Figure 5.2-6 and Figure 5.2-7 are not approximated when they reside outside the phase range available. To do this we will use equations (5.2-6) to equation (5.2-9) to make approximations for both F/D =1 and F/D = 0.5 cases. Figure 5.2-8 shows the mechanism of this approximation process for 100 cells (the pattern repeats for all the remaining cells) where we can see the actual phase required in blue circles and the approximated values in the red crosses when the required phase is not available in the database. Figure 5.2-9 and Figure 5.2-10 show the location of cells for which approximation has been made in the case of F/D =1 and F/D =0.5 respectively. Centers of the cells for which approximation is made are represented by red circles and Flag of 1 while other cells are flagged 0. We can see for F/D=0.5 case in Figure 5.2-10, more cells require approximation. Next, we look at Algorithm-13 which basically makes a function call to Algorithm-6.

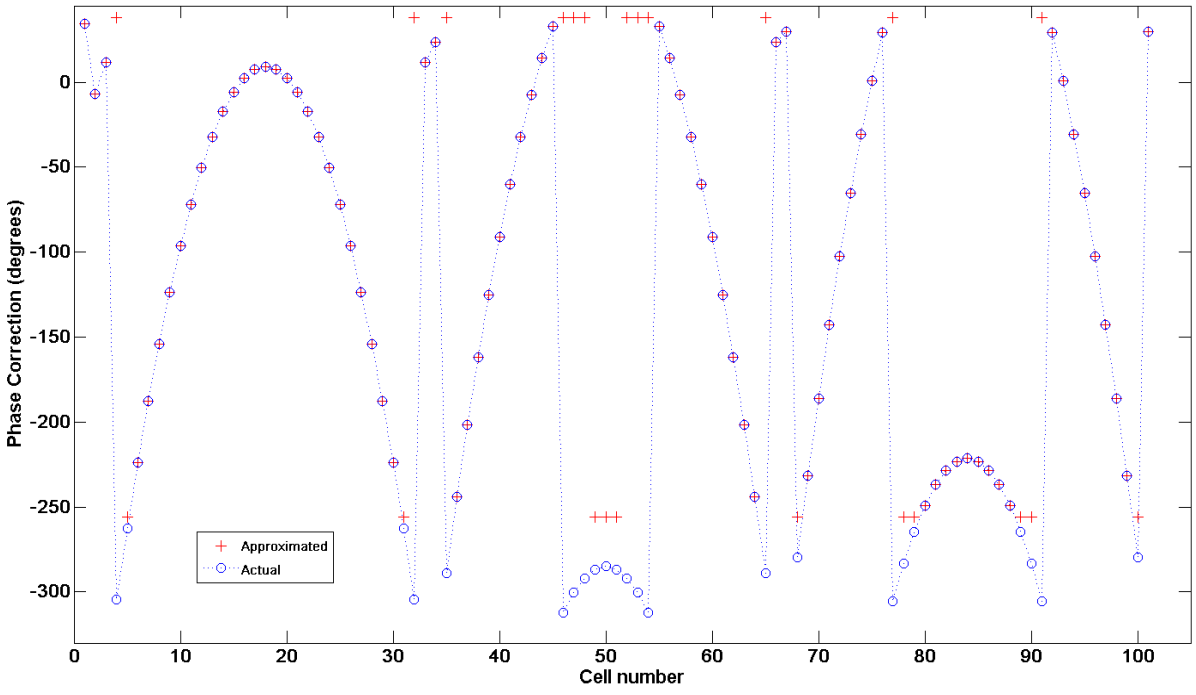


Figure 5.2-8 Transmission Phase Approximations Made for the Extreme Values

Algorithm 13 : Transmitarray Prototype Design Algorithm

comment: This Algorithm can be used for both F/D =1 and F/D =0.5 cases.

Input: Transmission phase values required at the center of the entire cell on the transmitarray aperture.

Output: Metallic patch dimensions (a_1, a_2) for all the cells

begin

for n <= 1 to N (number of elements on the aperture = 2025)

comment: compute $\arg\{S_{21}\}^n$ using equation (5.2-2)

comment: if $\arg\{S_{21}\}^n$ lies outside the phase range then approximate it using equations (5.2-6) to (5.2-9)

comment: record the coordinates of the cells for which approximation is made

$[(a_1, a_2)]^n$ = ***function call*** - *Algorithm-6*(***argument*** = $\arg\{S_{21}\}^n$) **)**

end

stop

Computation Details for Algorithm-13

Number of elements = 2025

Average time to compute patch dimensions of n^{th} celll = 22 seconds

Total time taken for entire transmitarray = 96 minutes

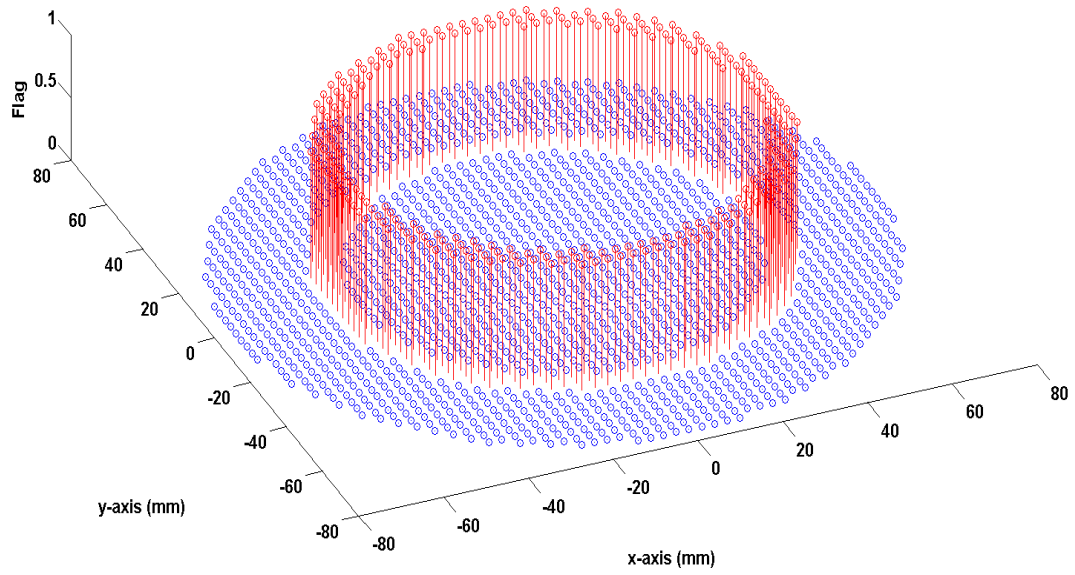


Figure 5.2-9 Location of Cells on the Aperture for which Approximation is Made (Highlighted with Red Flags) for F/D =1

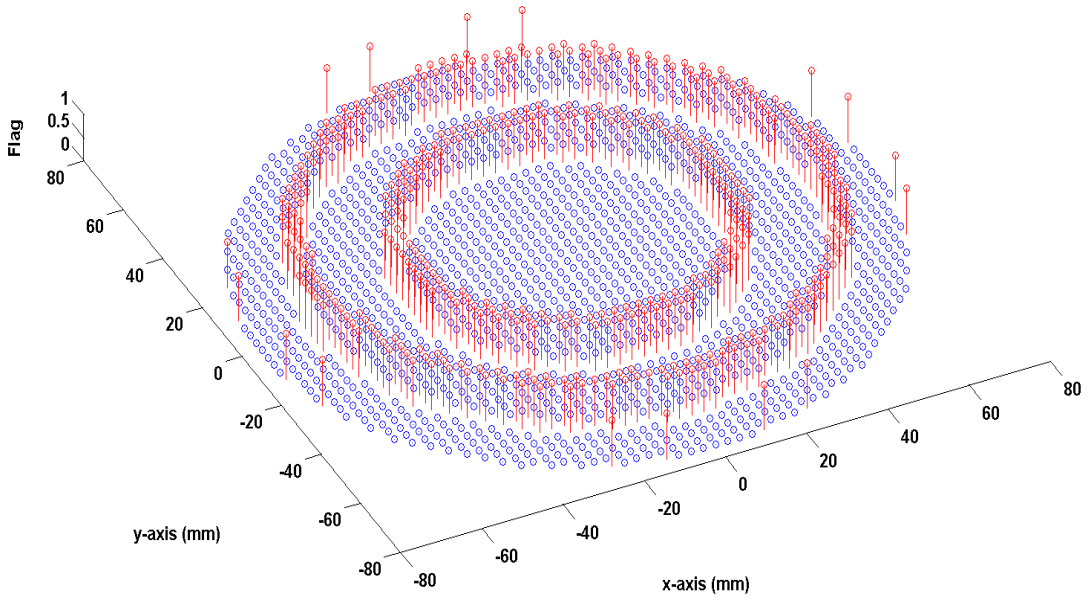


Figure 5.2-10 Location of Cells on the Aperture for which Approximation is Made (Highlighted with Red Flags) for F/D =0.5

5.3 VALIDATION OF THE FULL WAVE ANALYSIS OF THE TRANSMITARRAY

Before we present the results pertaining to the radiation performance of the transmitarray antenna prototype, it is vital to first of all validate the full wave electromagnetic simulation based analysis which we will be utilizing to characterize its performance. Given the theoretical nature of this thesis i.e. to propose a novel methodology to design a transmitarray antenna, we didn't manufacture a physical antenna; instead, we used a 3D full-wave model for the designed transmitarrays for analyzing the performance. This is done using HFSS, utilizing its FE-BI capability. The model includes a pyramidal horn feed as an intrinsic part of the model.

The validity of this approach of modelling transmitarrays has been validated in [5.3-1] through comparison with experimental data available for a specific transmitarray described in [5.3-2]. The transmitarray under consideration there has square metallic elements (similar to our design). It was designed and fabricated, its performance measured [5.3-2]. The same design was simulated according to the method proposed in [5.3-1] and the comparison in made in Figure 5.3-1 and Figure 5.3-2.

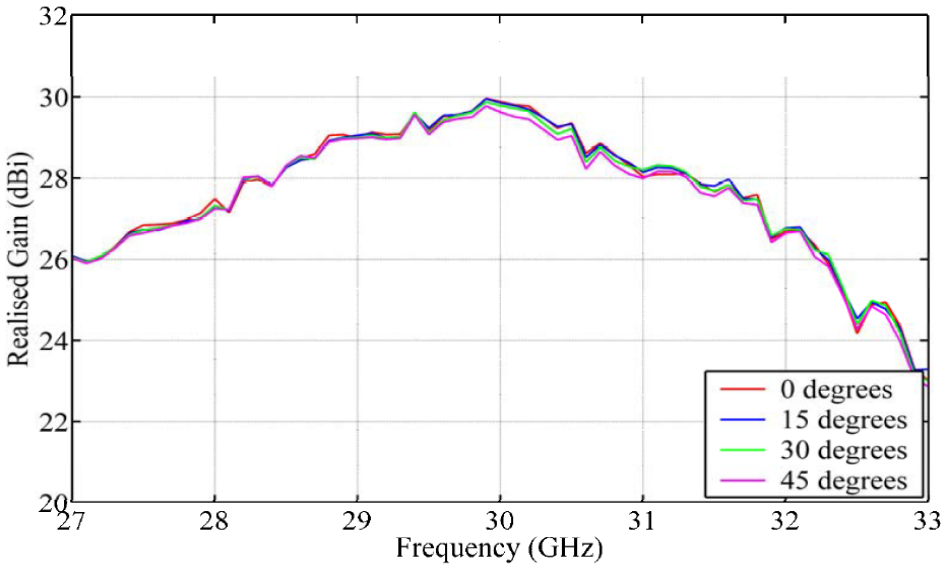


Figure 5.3-1 Measured Antenna Gain for Various Slant Linear Polarizations of the Horn Feed versus Frequency [5.3-2]

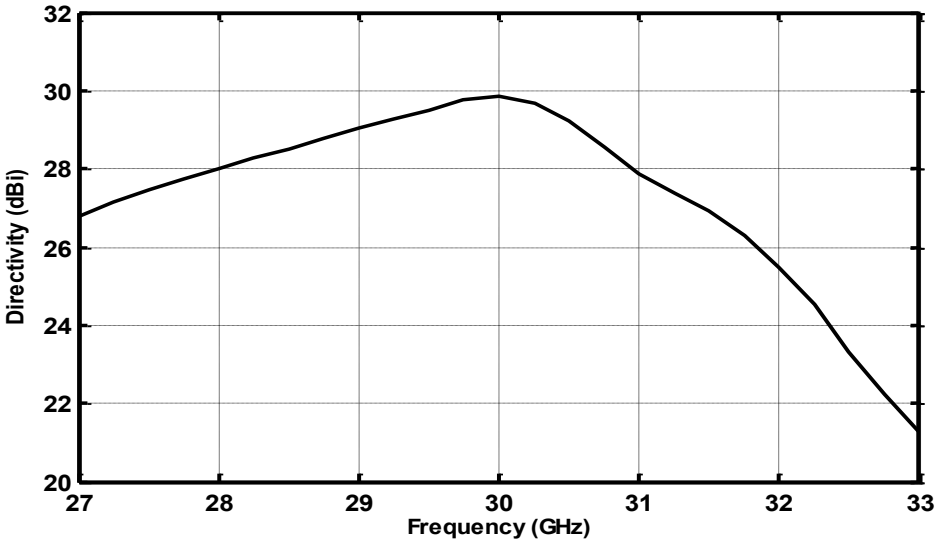


Figure 5.3-2 Computed Antenna Directivity versus Frequency [5.3-1]

The measured gain in Figure 5.3-1, and computed directivity in Figure 5.3-2, are within about 0.1 dB in the 29-31 GHz range. It can be concluded that the computational technique used in [5.3-1], and being used in the present chapter to simulate the complete transmitarray structure, is a reliable one.

5.4 COMPUTER PERFORMANCE OF THE TRANSMITARRAYS DESIGNED USING THE INN MODELS

Having established the accuracy of the computational model to simulate the complete designed transmitarray, we will next look at its radiation performance. As mentioned in Section 5.2.2, we have designed two transmitarray antenna prototypes by shifting feed location. In the first case the ratio of feed distance (focal length) to the diameter of the aperture F/D =1. In the second case F/D = 0.5. We will present the radiation patterns and associated gains of these designs separately in the next two sections.

5.4.1 Computed Performance for F/D = 1 Design

The radiation pattern is computed for the F/D =1 design using FE-BI based electromagnetic simulation technique [5.3-1] in the HFSS simulator, and Figure 5.4-1 shows the 3D radiation pattern at f = 29.75 GHz where the gain is maximum gain of 30.6 dBi is realized; Figure 5.4-2 shows the gain versus frequency plot. It is immediately clear that the use of the INN has been successful.

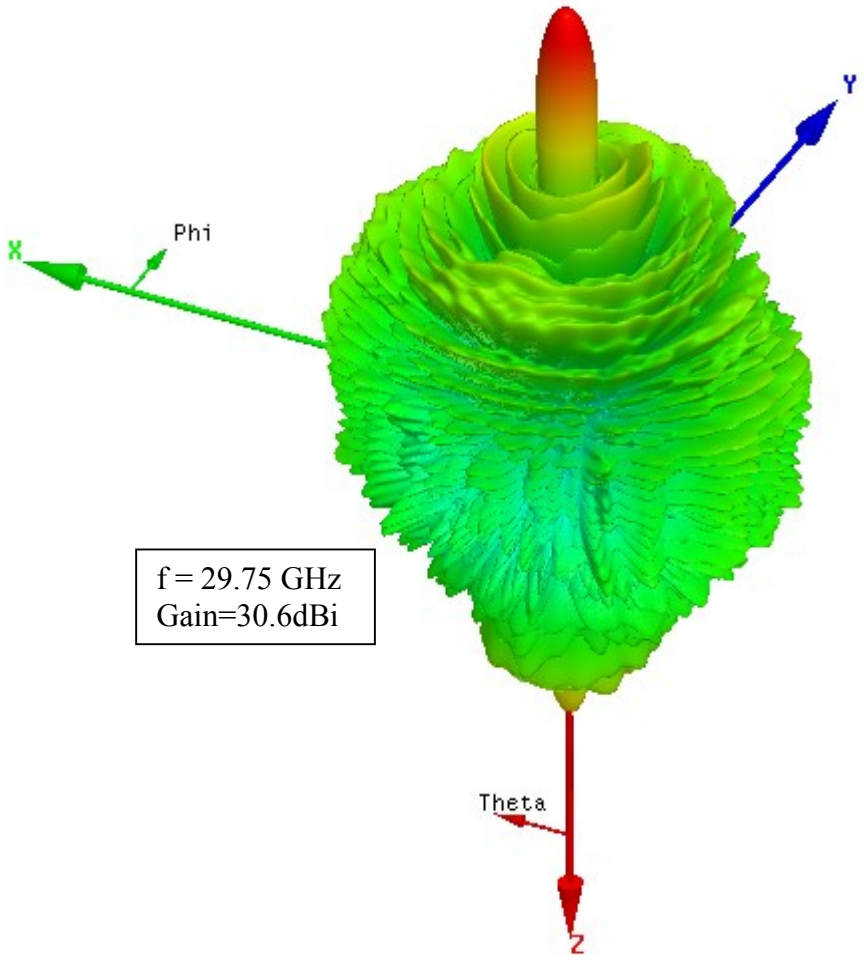


Figure 5.4-1 3D Radiation Pattern of the Transmitarray Prototype such that F/D =1

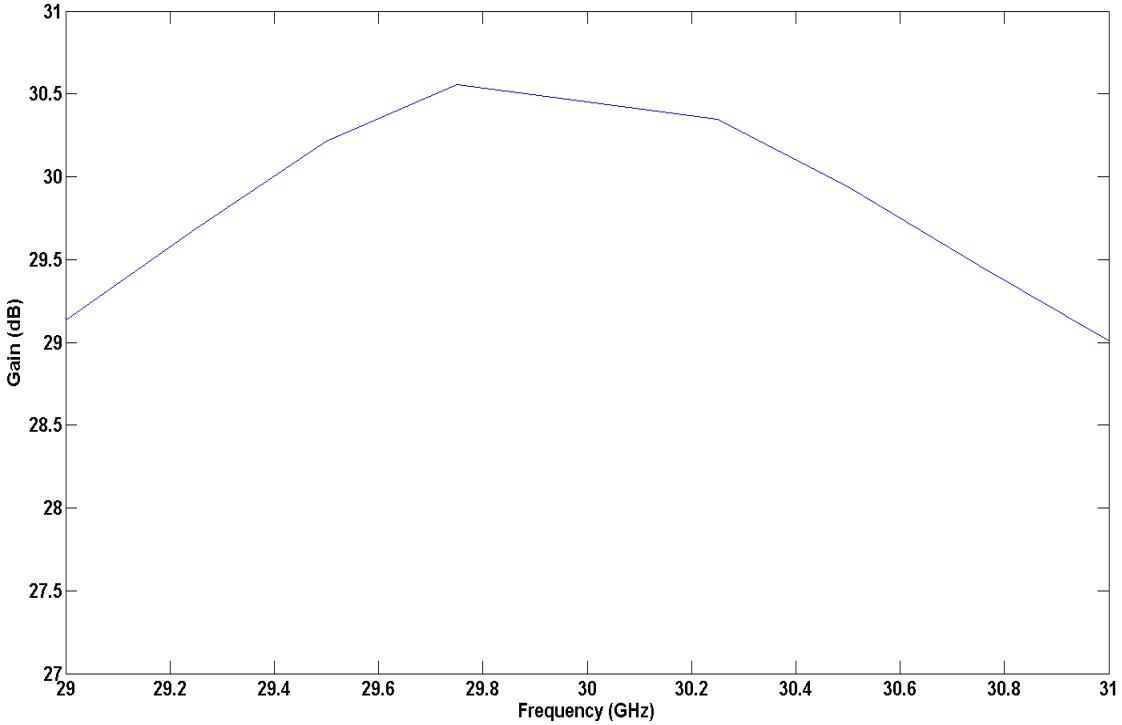


Figure 5.4-2 Gain versus Frequency for F/D = 1

5.4.2 Computed Performance for F/D = 0.5 Design

Next, we will present the results for the design in which the feed was brought closer to the aperture such that F/D = 0.5. Figure 5.4-3 represents the 3D radiation pattern at f = 30 GHz. It is at this frequency that maximum gain of 30.35 dBi was achieved. In figure 5.4-4, we present the gain versus frreuqnecy plot. It can be seen that there is a sharp drop in gain away from the central frequency.

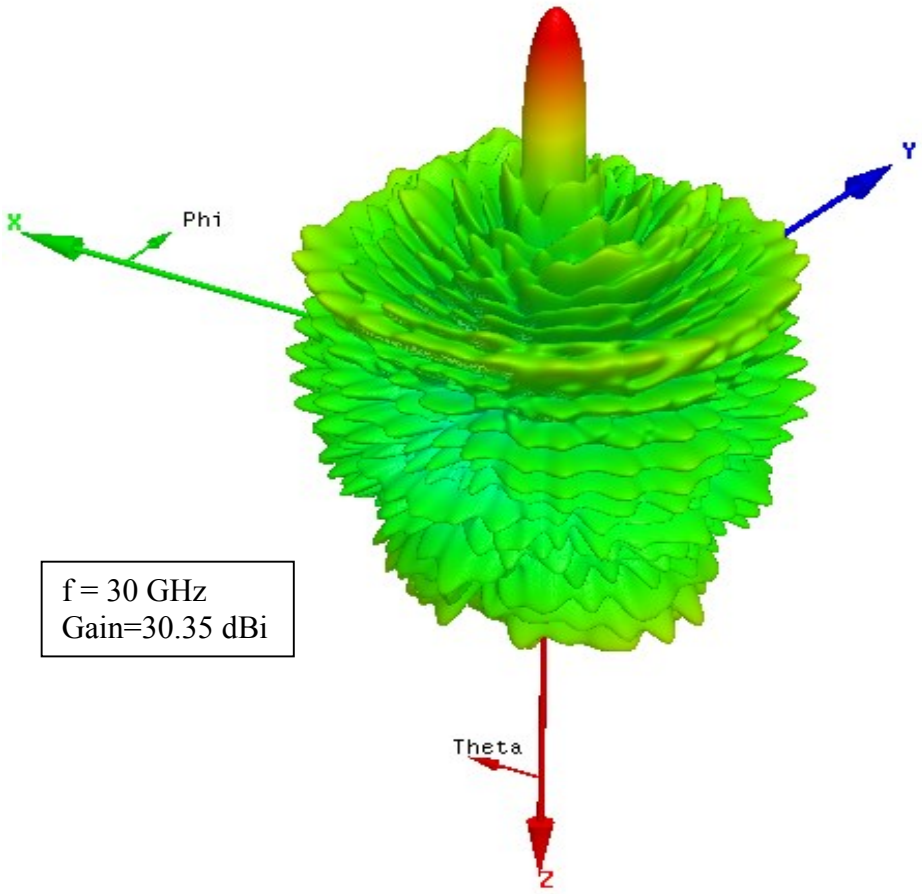


Figure 5.4-3 3D Radiation Pattern of the Transmitarray Prototype such that F/D =0.5

If we observe the gain versus frequency curve in Figure 5.4-2 it can be asserted that the transmitarray performs satisfactorily in the band 29-31 GHz as the gain is above 29 dB for the entire band. But if we compare gain versus frequency plots for F/D=1 and F/D =0.5 (Figure 5.4-3), it can be seen that the drop in gain is sharp when we move away from the frequency of operation (which is 30 GHz) in the case of F/D =0.5, as opposed to a relative smooth drop for F/D =1. Therefore, this type of transmitarray can be used for relatively wider band if the feed is farther from the aperture, and bandwidth will narrower in the case when feed is located closer to the aperture. On the other hand if we look at this from the perspective of the antenna's overall profile and size, the F/D = 1 design begs for larger physical size, and if this antenna is employed

for SATCOM applications – given the limited of space in such systems- it would be advisable to utilize the design with F/D = 0.5, albeit with narrow band performance. Most importantly, the success of the designprocess has demonstrated the ability of the INN model for database inversion.

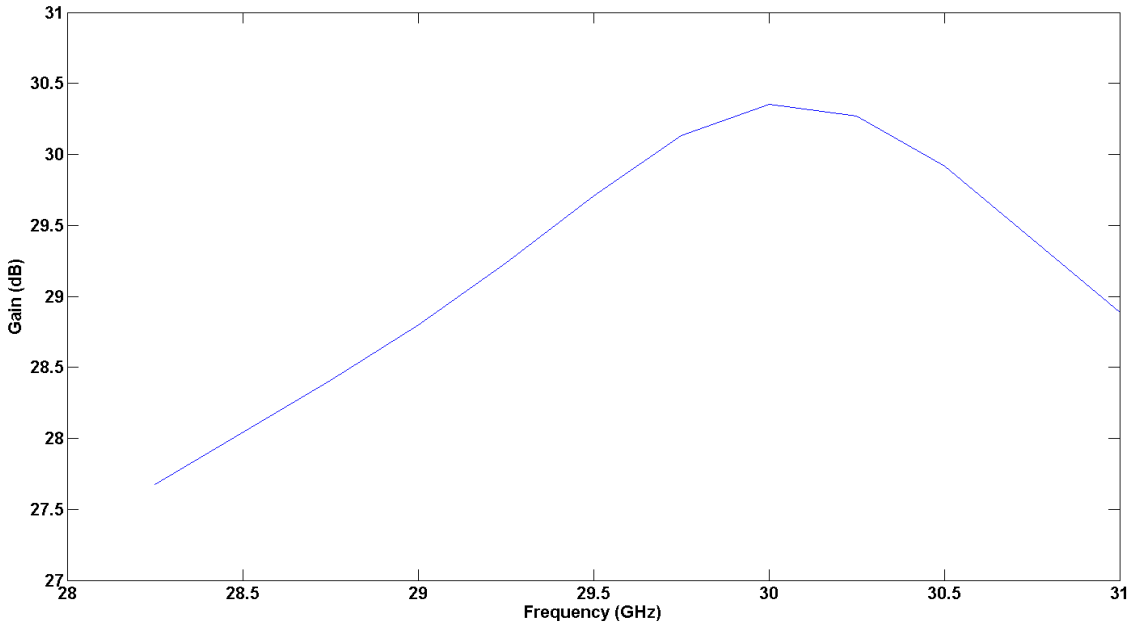


Figure 5.4-4 Gain versus Frequency for F/D = 0.5

We again remind the reader that the results provided here are for the case when the transmitarray was designed based on the approach detailed in Chapter 3 i.e. design was undertaken under the assumption that the transmission coefficient is independent of the incident angle. The methodology explained in Chapter 4, on the other hand, doesn't ignore the oblique incidence, and this could be used to include the impact of oblique incidence in the design process.

5.5 CONCLUDING REMARKS

In this chapter we applied the INN modelling approach introduced, explained and verified in Chapter 3, to design transmitarray antennas. First the specifications for the transmitarray prototype were presented and the criterion for selection of specific parameters was outlined. Then the approach used to approximate the transmisison phase for the extreme values for which transmission phase is not available in the database was explained and implemented. Next, we implemented Algorithm-13 to calculate the metallic patch dimensions for each of the 2025 cells. on the aperture for two different transmitarray designs. A previously validated computational approach for 3D electromagnetic modelling of compete transmitarrays (feasible only for analysis purposes, and not for design) was used to determine the performance of the two transmitarrays designed via the INN model. The expected performance was indeed obtained, demonstrating the utility of the INNdatabase inversion approach. The INN approach permits very rapid designs, allowing trade-off studies (eg. Should we use F/D =1 or F/d =0.5) to be performed. As mentioned in previous chapters, the INN approach allows even greater flexibility than has been exploited in the present chapter. We believe it is the first time the INN idea has been used for the design of aperture antennas comprised of engineered surfaces.

CHAPTER 6
General Conclusions

The principal contributions of this thesis are:

- We have shown that it is not practical to apply the INN concepts to the database of points obtained directly from full-wave simulations; too many such simulations would be needed. In Section 3.2 it was demonstrated how the distribution of these initial full-wave database points can be examined to decide on how to best use forward (as opposed to inverse) neural network modelling to augment the number of full-wave database points by a factor of about fifty, in order to have sufficient training data for implementing INNs accurately.

- We developed ANN models to characterize the relationship between input physical parameters (dimensions) and output electrical parameters (S-parameters). In this thesis we have applied neural network characterization for the first time to the transmitarray design in Chapter 3. The neural network characterization model thus generated to accomplish this is very fast and we demonstrated it to be an efficient approximant.

- It is then shown in Chapter 3 that the highly non-linear and multivalued (not one-to-one) nature of the data necessitates the use of inverse sub-models. Algorithm-4 and Algorithm-5 have been given for such an implementation. The inverse sub-model theory available in the literature is altered slightly to suite some database inversion issues (and decision-making) possibly unique to the present type of antenna design problem. This latter extension, implemented via Algorithm-6 (to create "inverse" Database-3) also offers some flexibility that might be of benefit in the design of this class of antennas in the future. The work in Section 3.6, implemented via Algorithm-7, presents solid verification of the effectiveness of the INN-based inversion of the augmented design database.

- All the algorithms needed are given in the form of flow-charts and complete pseudo-code for easy implementation by others; such detailed implementational details have not been

available elsewhere in the literature. There are specific tweaks and modifications that we made in the originally proposed algorithm in Chapters 3 and 4 in order to make it suitable and even more robust to facilitate its application in the transmitarray design.

- In Chapter 4, we investigated the impact of the oblique incidence on the transmission coefficient and incorporated it into the design databases and characterized the relationship between transmission coefficient and the oblique incidence using neural networks. INN models were generated for each incidence angle database separately and conflated to produce the final outputs. This methodology, although demonstrated and verified, hasn't been applied to an actual antenna design, but can be easily utilized in the design process. In the present literature the oblique incidence is neglected but we have shown that it can be easily included in the database and possibly leading to better performance of the antennas thus designed.

- All the algorithms that we derived from previous works and introduced minor alterations or algorithms that have been contrived particularly for this work have been explicitly presented in the form of pseudo-codes and/or flowcharts. Therefore, anyone can use these algorithms for related design challenges where we need to generate a database of electrical parameters and finally invert it for optimum physical parameters.

- A previously validated computational approach for 3D electromagnetic modelling of compete transmitarrays (feasible only for analysis purposes, and not for design) was used to determine the performance of the two transmitarrays designed via the INN model. The expected performance was indeed obtained, demonstrating the utility of the INN database inversion approach. The INN approach permits very rapid designs, allowing trade-off studies (eg. Should we use F/D =1 or F/d =0.5) to be performed. The INN approach allows even greater flexibility than has been exploited in the present application.

There remain some issues whose investigation in the future would prove useful:

- As implied earlier in this discussion, we are currently in the final phase of implementing the oblique incidence based INN model to generate the transmitarray antenna and

compare its performance to the one we designed in Chapter 6. It is very vital to probe the influence of the inclusion of incidence angle in the transmitarray database and consequently the antenna radiation and gain performance.

- The modified version of the output extraction algorithm that we proposed (Algorithm-6), allows us to select multiple possible output dimensions for a cell for a single input transmission phase. This aspect of the design process can allow further improvement in the performance of trasnsmitarrays if we choose (a_1, a_2) pairs such that the gradient w.r.t. to the surface is smooth. It is because if the gradient is small for the adjacent cells, for each cell, the neighborhood would appear to be as a 2D periodic structure. Since in the generation of design database the underlying assumption was that the unit cell of the transmitarray is situated inside an infinite 2D periodic structure, the smooth gradient would simulate conditions almost identical to an infinite periodic structure.

- In the future, it would be good to try and reduce the computational cost and time consumption of the INN modelling if it is to be used repeatedly. It is possible to refine the computations in the model INN to make them even more rapid (and yet significantly accurate) then they already are.

Appendix A

Flowchart Symbols

Flowchart Symbol	Name (Alternates)	Description
	Process	An operation or action step.
	Terminator	A start or stop point in a process.
	Decision	A question or branch in the process.
	Delay	A waiting period.
	Predefined Process	A formally defined sub-process.
	Alternate Process	An alternate to the normal process step.
	Data (I/O)	Indicates data inputs and outputs to and from a process.
	Document	A document or report.
	Multi-Document	Same as Document, except, well, multiple documents.
	Preparation	A preparation or set-up process step.
	Display	A machine display.
	Manual Input	Manually input into a system.
	Manual Operation	A process step that isn't automated.
	Card	A old computer punch card.
	Punched Tape	An old computer punched tape input.
	Connector	A jump from one point to another.
	Off-Page Connector	Continuation onto another page.
	Transfer	Transfer of materials.
	Or	Logical OR
	Summing Junction	Logical AND
	Collate	Organizing data into a standard format or arrangement.
	Sort	Sorting of data into some pre-defined order.
	Merge (Storage)	Merge multiple processes into one. Also used to show raw material storage.
	Extract (Measurement) (Finished Goods)	Extract (split processes) or more commonly - a measurement or finished goods.
	Stored Data	A general data storage flowchart symbol.
	Magnetic Disk (Database)	A database.
	Direct Access Storage	Storage on a hard drive.
	Internal Storage	Data stored in memory.
	Sequential Access Storage (Magnetic Tape)	An old reel of tape.
	Callout	One of many callout symbols used to add comments to a flowchart
	Flow Line	Indicates the direction of flow for materials and/or information

REFERENCES

[2.2-1] Y.Rahmat-Samii, “Reflector Antennas”, Chapter 15 in J.L.Volakis (Edit.), *Antenna Engineering Handbook* (McGraw-Hill, 2007) 4th Edition.

[2.4-1] S. W. Lee, G. Zarrillo, C. L. Law, "Simple formulas for transmission through periodic metal grids or plates", *IEEE Transactions on Antennas and Propagation*, vol.30 (5), pp. 904 – 909 Sept. 1982.

[2.5-1] K. Hornik, M. Stinchcombe, and H. White, “Multilayer Feedforward Networks are Universal Approximators,” *Neural Network*, vol. 2, no. 5, pp. 359–366, 1989.

[2.5-2] J. P. Garcia, F. Q. Pereira, D. C. Rebenaque, J. L. G. Tornero, and A. A. Melcon, “A Neural-Network Method for the Analysis of Multilayered Shielded Microwave Circuits,” *IEEE Trans. Microwave Theory Tech.*, vol. 54, no. 1, pp. 309–320, Jan. 2006.

[2.5-3] Y. Kim, S. Keely, J. Ghosh, and H. Ling, “Application of Artificial Neural Networks to Broadband Antenna Design Based on a Parametric Frequency Model,” *IEEE Trans. Antennas Propagat.*, vol. 55, no. 3, pp. 669–674, Mar. 2007.

[2.5-4] P. M. Watson and K. C. Gupta, “Design and Optimization of CPW Circuits Using EM-ANN Models for CPW Components,” *IEEE Trans. Microwave Theory Tech.*, vol. 45, no. 12, pp. 2515–2523, Dec. 1997.

[2.5-5] C. Ydiz and M. Turkmen, “Very Accurate and Simple CAD Models Based on Neural Networks for Coplanar Waveguide Synthesis,” *Int. J. RF Microwave Computer.-Aided Eng.*, vol. 15, no. 2, pp. 218–224, Mar. 2005.

[2.5-6] P. Sen, W. H. Woods, S. Sarkar, R. J. Pratap, B. M. Dufrene, R. Mukhopadhyay, C. Lee, E. F. Mina, and J. Laskar, “Neural Network Based Parasitic Modeling and Extraction Verification for RF/Millimeter-Wave Integrated Circuit Design,” *IEEE Trans. Microwave Theory Tech.*, vol. 54, no. 6, pp. 2604–2614, June 2006.

[2.5-7] V. Rizzoli, A. Costanzo, D. Masotti, A. Lipparini, and F. Mastri, “Computer-Aided Optimization of Nonlinear Microwave Circuits With the Aid of Electromagnetic Simulation,” *IEEE Trans. Microwave Theory Tech.*, vol. 52, no. 1, pp. 362–377, Jan. 2004.

[2.5-8] V. Rizzoli, A. Neri, D. Masotti, and A. Lipparini, “A New Family of Neural Network-Based Bidirectional and Dispersive Behavioral Models for Nonlinear RF/microwave Subsystems,” *Int. J. RF Micro- wave Comput.-Aided Eng.*, vol. 12, no. 1, pp. 51–70, Jan. 2002.

[2.5-9] D. M. M.-P. Schreurs, J. Verspecht, S. Vandenberghe, and E. Van- damme,

"Straightforward and Accurate Nonlinear Device Model Parameter-Estimation Method Based on Vectorial Large-Signal Measurements," *IEEE Trans. Microwave Theory Tech.*, vol. 50, no. 10, pp. 2315–2319, Oct. 2002.

[2.5-10] V. B. Litovski, J. I. Radjenovic, Z. M. Mrcarica, and S. L. Milenkovic, "MOS Transistor Modeling Using Neural Network," *Electron. Lett.*, vol. 28, no. 18, pp. 1766–1768, Aug. 1992.

[2.5-11] G. L. Creech, B. J. Paul, C. D. Lesniak, T. J. Jenkins, and M. C. Calcatera, "Artificial Neural Networks for Fast and Accurate EM-CAD of Microwave Circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 45, no. 5, pp. 794–802, May 1997.

[2.5-12] Qi-Jun Zhang, Kuldip C. Gupta and Vijay K. Devabhaktuni*,''* Artificial Neural Networks for RF and Microwave Design—From Theory to Practice*'', IEEE Trans. Microwave Theory Tech* ,vol. 51, no. 4, April 2003

[2.5-13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing*, vol. 1, D. E. Rumelhart and J. L. McClelland, Eds. Cam- bridge, MA: MIT Press, 1986, pp. 318–362.

[2.5-14] S. Haykin, "Neural Networks: A Comprehensive Foundation," New York, NY: IEEE Press, 1994.

[2.5-15] H. Ninomiya, S. Wan, H. Kabir, X. Zhang, and Q. J. Zhang, "Robust Training of Microwave Neural Network Models Using Combined Global/Local Optimization Techniques," in *IEEE MTT-S Int. Microwave Symp. Dig.*, Atlanta, GA, June 2008, pp. 995–998.

[2.5-16] V.Rizzoli,A.Costanzo,D.Masotti,A.Lipparini,andF.Mastri,"Computer Aided Optimization of Nonlinear Microwave Circuits with the Aid of Electromagnetic Simulation," *IEEE Trans. Microw. Theory Tech.*, vol. 52, no. 1, pp. 362–377, Jan. 2004.

[2.5-17] P. M. Watson and K. C. Gupta, "Design and Optimization of CPW Circuits Using EM-ANN Models for CPW Components," *IEEE Trans. Microw. Theory Tech.*, vol. 45, no. 12, pp. 2515–2523, Dec. 1997.

[2.5-18] P. M. Watson and K. C. Gupta, "EM-ANN Models for Microstrip Vias and Interconnects in Dataset Circuits," *IEEE Trans. Microw. Theory Tech.*, vol. 44, no. 12, pp. 2495–2503, Dec. 1996.

[2.5-19] M. M. Vai, S. Wu, B. Li, and S. Prasad, "Reverse Modeling of Microwave Circuits with Bidirectional Neural Network Models," *IEEE Trans. Microw. Theory Tech.*, vol. 46, no. 10, pp. 1492–1494, Oct. 1998.

[2.5-20] H. Kabir, Y. Wang, M. Yu, Q. J. Zhang, "Neural Network Inverse Modeling and Applications to Microwave Filter Design," *IEEE Trans. Microw. Theory Tech.* , vol. 56, no. 4, April 2008

[2.5-21] H. Kabir, L. Zhang, M. Yu, P. H. Aaen, J. Wood, Q. J. Zhang, "Smart Modeling of Microwave Devices," *IEEE Microwave Magazine,* May 2010.

[2.5-22] Q. J. Zhang and K. C. Gupta, "Neural Networks for RF and Microwave Design," Boston: Artech House, 2000

[2.6-1] S. Nesil, F. Gunes, U. Ozkaya and B. Turetken, "Generalized Regression Neural Network based Phase Characterization of Reflectarray Employing Minkowski Element of Variable Size," URSI, 2011.

[2.6-2] F. Gunes, S. Nesil and S. Demirel, ''Design and Analysis of Minkowski Reflectarray Antenna Using 3-D CST Microwave Studio-Based Neural Network Model with Particle Swarm Optimization,'' *Int. J. RF Microwave Computer.-Aided Eng.*, Vol. 23, No. 2, Mar. 2013.

[2.6-3] S. Nesil, F. Gunes and G. Kaya,'' Analysis and Design of X-Band Reflectarray Antenna using 3-D EM – based Artificial Neurall Network Model,*'' IEEE International Conference on Ultra-Wideband, ICUWB 2012,* pp. 532-536, Sept. 2012.

[2.6-4] A. Freni, M. Mussetta, P. Pirinoli, '' Neural Network Characterization of Reflectarray Antennas,'' *Hindawi International Journal of Antennas and Propagation,* vol. 2012,Article ID 541354, 2012.

[2.6-5] P. Robustillo, J. Zapata, J.A. Encinar and J. Rubio, ''ANN Characterization of Mulit-Layer Reflectarray Elements for Contoured-Beam Space Antennas in the Ku-Band,'' *IEEE Trans. on Antenna and Propagation,* vol. 60, issue 7, pp. 3205-3214, 2012.

[3.2-1] *HFSS*, Ansoft Product Suite, Ansys Inc., USA (www.ansoft.com)

[3.3-1] Kohonen, T., Self Organized Formulation of Topologically Correct Feature Maps, *Biological Cybermatics,* Vol. 43, 1982, pp. 59–69.

[5.3-1]E.Almajali, N.Gagnon, D.A.McNamara & A.Petosa, "Remarks on the Feasibility of Full-Wave Analyses of Printed Lens/Transmitarray Antennas", *IEEE AP-S Int. Symp*., Memphis, Tennessee, USA, July 2014.

[5.3-2]N. Gagnon, “Phase Shifting Surface (PSS) and Phase and Amplitude Shifting Surface (PASS) for Microwave Applications,” PhD Thesis, University of Ottawa, Ottawa, ON, Canada, 2011. Available for download at www.ruor.uottawa.ca/fr/handle/10393/19826