Advanced Neural-Based Model Generation and Extrapolation Techniques for Microwave Applications

by

Weicong Na, B.Eng

A book based on the thesis submitted to
The Faculty of Graduate and Postdoctoral Affairs
in fulfillment of the degree requirements of
Doctor of Philosophy

Microelectronics and Solid State Electronics, School of Microelectronics
Tianjin University, Tianjin, China

Ottawa-Carleton Institute for Electrical and Computer Engineering
Carleton University, Ottawa, Ontario, Canada

Abstract

Neural-based modeling techniques have been recognized as important vehicles in the microwave computer-aided design (CAD) area in addressing the growing challenges of designing next generation microwave device, circuits and systems. Artificial neural network models can be trained to learn electromagnetic (EM) and physics behavior. The trained neural network models can be used in high-level circuit and system design allowing faster simulation and optimization including EM and physics effects in components. The purpose of this thesis is to develop advanced neural-based model generation and extrapolation techniques for microwave applications. The proposed techniques take advantage of the high-efficiency of automated model generation algorithm, the cost-effective concept of knowledge-based neural network and the generalization capability of extrapolation techniques, to achieve reliable models for microwave applications.

To further speed up neural modeling process, an automated knowledge-based neural network model generation method using a new adaptive sampling technique for microwave applications is firstly proposed. The proposed method integrates all the subtasks involved in knowledge-based neural modeling, thereby facilitating a more efficient and automated model development framework. The new adaptive sampling technique incorporates interpolation techniques to determine the additional training samples required and their location in model input space. In this way, the proposed method can improve the efficiency and reduce the expense of knowledge-based neural model development.

For different modeling problems, the mapping structures in knowledge-based

models should be different. We propose a unified automated model structure adaptation algorithm for knowledge-based modeling using l_1 optimization to automatically determine the type and topology of the mapping structure in a knowledge-based model. A new unified knowledge-based model structure to encompass various types of mappings is proposed. Using the distinctive property for feature selection of l_1 optimization, the proposed method can automatically distinguish whether a mapping is needed or not and whether a mapping is linear or nonlinear. It is a more flexible and systematic technique and can further speed up the knowledge-based neural model development.

As a further advancement, we propose an advanced multi-dimensional extrapolation technique for neural-based microwave modeling to make the model can be more reliably used outside the training range. Grid formulation in the extrapolation region is introduced and the proposed extrapolation is performed over these grids. We present multi-dimensional cubic polynomial extrapolation formulation and propose to use optimization to obtain extrapolated values at grid points. By using the proposed extrapolation method, neural models become more robust and reliable when they are used outside the training range. The validity of the proposed extrapolation method is demonstrated by both EM optimization example and nonlinear microwave simulation examples.

KEY WORDS: Design automation, knowledge-based neural network, l_1 optimization, extrapolation, microwave modeling.

To my parents

Wenbo Na and Qiu Li

for all the love, understanding and support

Acknowledgements

First of all, I would like to express my sincere appreciation to my supervisor, Prof. Q. J. Zhang, for his professional guidance, invaluable inspiration and continuous support throughout the research work and the preparation of this thesis. This thesis would not have been completed without his patience, assistance, encouragement and faith of this study. It was my great honor to work under his supervision and guidance.

Special appreciation to my co-supervisor, Prof. Jianguo Ma for the invaluable counsel and knowledgeable instruction. His invaluable counsel provided a great help for me on my way pursuing my Ph.D degree. I would like to thank Dr. Michel Nakhla, Dr. Mustapha Yagoub and Dr. Pavan Gunupudi for their valuable suggestions and corrections for my thesis.

Many thanks to the School of Microelectronics, Tianjin University, where I have stayed for ten years accomplishing both my undergraduate and Ph.D studies. I would also like to thank the Department of Electronics, Carleton University for the Cotutelle program for my Ph.D study.

I would like to thank all my present and former colleagues in our research group, especially Dr. Lin Zhu, Dr. Shuxia Yan, Dr. Feng Feng, Chao Zhang, Dr. Wenyuan Liu, Jianan Zhang, Wei Zhang, Zhihao Zhao, Jing Jin, Dr. Venu-Madhav-Reddy Gongal Reddy, and others, for their collaboration, helpful discussions and valuable suggestions.

I would also like to thank my friends Chen Wu, Tingjun Liu, Ruixue Wang, Zhe Zhu, Qinwei Li, and others, too many to name here, for their lasting friendship and

hearty support in fulfilling my dreams.

Finally, I wish to thank my parents for their endless love, trust, support and encouragement throughout the years of my study. A special thank to my dear friend Yifan Jia for his constant accompany and encouragement.

Table of Contents

List of Figures

List of Tables

List of Symbols

$a_{it}^{(l)}$	The element in the ith row and tth column of submatrix $\boldsymbol{A}_l$
$\boldsymbol{A}$	The matrix containing the grid information
$\boldsymbol{A}_l$	The lth submatrix of $\boldsymbol{A}$
$\boldsymbol{A}_{new}$	The modified version of matrix $\boldsymbol{A}$ by canceling the step size factor in $\boldsymbol{A}$
$\boldsymbol{b}$	The vector containing the function value f and 1st order to nth order mixed partial derivative values of f w.r.t all input variables at N corner points of a cell
$b_{it}^{(l)}$	The element in the ith row and tth column of submatrix $\boldsymbol{B}_l$
$\boldsymbol{B}$	The matrix relating the derivatives of f to the model output values at the corner points of the current cell and its neighboring cells
$\boldsymbol{B}_l$	The lth submatrix of $\boldsymbol{B}$

$\boldsymbol{B}_{new}$	The modified version of matrix $\boldsymbol{B}$ by multiplying the step size factor to $\boldsymbol{B}$
$\boldsymbol{c}_k$	The kth intermediate vector containing 4^n elements during the extrapolation calculation
c_t	The coefficient evaluated according to the location of the corner point of a cell
C_n^k	The number of k-combinations from a set of n elements
$\boldsymbol{d}^{(a)}$	The vector containing the desired outputs of the model for the ath training sample $\boldsymbol{x}^{(a)}$
$\boldsymbol{d}^{(b)}$	The vector containing the desired outputs of the model for the bth testing sample $\boldsymbol{x}^{(b)}$
$\boldsymbol{D}_{ij}$	The matrix containing the intermediate 2nd order information for a particular cell
$e(\tilde{\boldsymbol{y}})$	The error function of model output values in $\tilde{\boldsymbol{y}}$
$e_j(\boldsymbol{y}_e)$	The error function of extrapolated output values in $\boldsymbol{y}_e$ for the jth cell
$E_(\boldsymbol{y}_e)$	The error function for the entire region during extrapolation
E_d	The user-desired model accuracy (testing error)

E_{In}^{k}	The interpolation error during the kth stage of the automated knowledge-based model development process
E_{train}^{k}	The training error during the kth stage of the automated knowledge-based model development process
E_{test}^{k}	The testing error during the kth stage of the automated knowledge-based model development process
$E(\boldsymbol{w})$	The error function of neural network internal weights $\boldsymbol{w}$.
f	The cubic polynomial function
$\boldsymbol{f}$	The entire knowledge-based model
$\boldsymbol{f}_{ANN}$	The neural network function
$\boldsymbol{f}_{E}$	The empirical model
$\boldsymbol{f}_{\text{map1}}$	The input space mapping neural network
$f_{\text{map1},i}$	The ith output of the input mapping neural network
f_{map2}	The frequency space mapping neural network
$\boldsymbol{f}_{\text{map3}}$	The output space mapping neural network
$f_{\text{map3},r}$	The rth output of the output mapping neural network

$\boldsymbol{F}$	The matrix correlating the parameters in quadratic function to the input/output information and the derivatives in the model-level extrapolation technique [11]
H_1	The number of hidden neurons in the input mapping neural network $\boldsymbol{f}_{\text{map1}}$
H_2	The number of hidden neurons in the frequency mapping neural network f_{map2}
H_3	The number of hidden neurons in the output mapping neural network $\boldsymbol{f}_{\text{map3}}$
$i_d(t)$	The time-domain drain current of a FET device
$i_g(t)$	The time-domain gate current of a FET device
I	The interpolation set
I_d	The drain current of a FET device
L	The training set
m	The number of output neurons
m_k	The number of grid points along the kth dimension of the input space
M	The unit-mapping set

n	The number of input neurons
N	The number of corner points of a cell
N_c	The number of cells in the entire region of the input space
N_E	The number of extrapolated data
N_g	The number of grid points in the entire region of the input space
N_L	The number of training data
N_V	The number of testing data
p_i	The index for the ith corner point of a cell
$\boldsymbol{P}$	The approximate mapping from the fine model parameter space $\boldsymbol{x}_f$ to the coarse model parameter space $\boldsymbol{x}_c$
$\boldsymbol{P}_E$	The incidence matrix to indicate the relationship between elements in $\tilde{\boldsymbol{y}}$ and elements in $\boldsymbol{y}_e$
$\boldsymbol{P}_L$	The incidence matrix to indicate the relationship between elements in $\tilde{\boldsymbol{y}}$ and elements in $\boldsymbol{y}_L$
$\boldsymbol{q}$	The vector containing the intermediate results calculated from $\boldsymbol{D}_{12}$ of every cell and $\boldsymbol{y}_L$
$\boldsymbol{Q}$	The matrix containing the intermediate results calculated from $\boldsymbol{D}_{11}$ of every cell

$\boldsymbol{r}$	The vector containing the input/output information and the derivatives in the model-level extrapolation technique [11]
$\boldsymbol{R}_c(\boldsymbol{x}_c)$	The corresponding responses of the coarse model
$\boldsymbol{R}_f(\boldsymbol{x}_f)$	The corresponding responses of the fine model
$s_k^{(i_k)}$	The step size along the kth dimension of the cell
$\boldsymbol{S}$	The diagonal matrix containing the step size factor
$\boldsymbol{t}_{\text{map1}}$	The vector of weights between the input neurons and hidden neurons in the input mapping neural network $\boldsymbol{f}_{\text{map1}}$
$t_{\text{map1},kj}$	The weight between the jth input neuron and the kth hidden neuron of the input mapping neural network $\boldsymbol{f}_{\text{map1}}$
$\boldsymbol{t}_{\text{map2}}$	The vector of weights between the input neurons and hidden neurons in the frequency mapping neural network f_{map2}
$t_{\text{map2},cb}$	The weight between the bth input neuron and the cth hidden neuron of the frequency mapping neural network f_{map2}
$\boldsymbol{t}_{\text{map3}}$	The vector of weights between the input neurons and hidden neurons in the output mapping neural network $\boldsymbol{f}_{\text{map3}}$
$t_{\text{map3},qp}$	The weight between the pth input neuron and the qth hidden neuron of the output mapping neural network $\boldsymbol{f}_{\text{map3}}$

T_r	The index set of training data
$\boldsymbol{u}$	The vector containing all weights for the presented direct connections between the input neurons and output neurons in all mapping neural networks
$\boldsymbol{u}_{\text{map1}}$	The vector of weights for the presented direct connections between the input neurons and output neurons in the input mapping neural network $\boldsymbol{f}_{\text{map1}}$
$u_{\text{map1},ij}$	The connection weight between the jth input neuron and the ith output neuron of the input mapping neural network $\boldsymbol{f}_{\text{map1}}$
$\boldsymbol{u}_{\text{map2}}$	The vector of weights for the presented direct connections between the input neurons and output neurons in the frequency mapping neural network f_{map2}
$u_{\text{map2},1b}$	The connection weight between the bth input neuron and the output neuron of the frequency mapping neural network f_{map2}
$\boldsymbol{u}_{\text{map3}}$	The vector of weights for the presented direct connections between the input neurons and output neurons in the output mapping neural network $\boldsymbol{f}_{\text{map3}}$
$u_{\text{map3},rp}$	The connection weight between the pth input neuron and the rth output neuron of the output mapping neural network $\boldsymbol{f}_{\text{map3}}$

$\boldsymbol{v}$	The vector containing all weights between the hidden neurons and output neurons in all mapping neural networks
$v_d(t)$	The time-domain drain voltage of a FET device
$v_g(t)$	The time-domain gate voltage of a FET device
$\boldsymbol{v}_{\text{map1}}$	The vector of weights between the hidden neurons and output neurons in the input mapping neural network $\boldsymbol{f}_{\text{map1}}$
$v_{\text{map1},ik}$	The weight between the kth hidden neuron and the ith output neuron of the input mapping neural network $\boldsymbol{f}_{\text{map1}}$
$\boldsymbol{v}_{\text{map2}}$	The vector of weights between the hidden neurons and output neurons in the frequency mapping neural network f_{map2}
$v_{\text{map2},1c}$	The weight between the cth hidden neuron and the output neuron of the frequency input mapping neural network f_{map2}
$\boldsymbol{v}_{\text{map3}}$	The vector of weights between the hidden neurons and output neurons in the output mapping neural network $\boldsymbol{f}_{\text{map3}}$
$v_{\text{map3},rq}$	The weight between the qth hidden neuron and the rth output neuron of the output mapping neural network $\boldsymbol{f}_{\text{map3}}$
V	The testing set
V_d	The drain voltage of a FET device

V_g	The gate voltage of a FET device
$\boldsymbol{V}_p$	The parameters in quadratic function in the model-level extrapolation technique [11]
$\boldsymbol{w}$	The vector of weights in the entire knowledge-based model
$\boldsymbol{w}^*$	The optimal solution after the training process containing the weight parameters of all the mapping neural networks
$\boldsymbol{w}_E$	The vector of weights in the empirical model
$\boldsymbol{w}_E^*$	The optimal solution of the empirical model optimization
$\boldsymbol{w}_{\text{map1}}$	The vector of weights in the input mapping neural network $\boldsymbol{f}_{\text{map1}}$
$\boldsymbol{w}_{\text{map2}}$	The vector of weights in the frequency mapping neural network f_{map2}
$\boldsymbol{w}_{\text{map3}}$	The vector of weights in the output mapping neural network $\boldsymbol{f}_{\text{map3}}$
$\boldsymbol{W}$	The weighting matrix to regulate the amount of influence of the base points in the model-level extrapolation technique [11]
$\boldsymbol{x}$	The vector of model inputs, containing the design parameters of a RF/microwave device/circuit
$\boldsymbol{x}_c$	The vector containing design parameters of the coarse model
$\boldsymbol{x}_E$	The vector containing the inputs of the empirical model

$\boldsymbol{x}_f$	The vector containing design parameters of the fine model
$x_k^{(i_k)}$	The i_kth grid point along the kth dimension of the entire region
$\Delta x_{E,i}$	The BP error at the output of the input mapping neural network
$\boldsymbol{y}$	The vector of model outputs, containing the response of the device/circuit under consideration
$\tilde{\boldsymbol{y}}$	The vector containing the model output values at the 4^n corner points of a given cell and its neighboring cells
$\boldsymbol{y}_e$	The vector containing the extrapolated target values
$\boldsymbol{y}_E$	The vector containing the outputs of the empirical model
$\Delta y_{E,p}$	The BP error at the pth output of the empirical model
$\boldsymbol{y}_L$	The vector containing the output values of given training data
$\Delta y_r^{(a)}$	The difference between the rth model output and the EM data output of the ath sample
$\boldsymbol{z}_j$	The jth sample of model inputs in the initial training data set L
$z_{\text{map1},k}$	The output of the kth hidden neuron of the input mapping neural network $\boldsymbol{f}_{\text{map1}}$
$z_{\text{map2},c}$	The output of the cth hidden neuron of the frequency mapping neural network f_{map2}

$z_{\text{map3},q}$	The output of the qth hidden neuron of the output mapping neural network $\boldsymbol{f}_{\text{map3}}$
$\Delta z_{\text{map1},k}$	The BP error back propagated from the output layer of the input mapping to the hidden neuron of the input mapping
$\Delta z_{\text{map2},c}$	The BP error back propagated from the output layer of the frequency mapping to the hidden neuron of the frequency mapping
$\Delta z_{\text{map3},q}$	The BP error back propagated from the output layer of the output mapping to the qth hidden neuron of the output mapping
$\boldsymbol{\alpha}$	The vector containing 4^n coefficients of the cubic polynomial function
β_k	The coefficient evaluated according to the location of a given corner point of the kth dimension of the cell
γ_i	The one-dimensional input variable for the switch function of the ith hidden neuron
$\gamma_{i\,\max}$	The maximum of γ_i
$\gamma_{i\,\min}$	The minimum of γ_i
γ_{jj}	The element at the jth row and jth column of matrix $\boldsymbol{S}$
$\lambda^1_{\text{map1},ik}$	The non-negative constant for the input mapping weight $v_{\text{map1},ik}$ during the first stage training process

$\lambda^1_{\text{map2},1c}$	The non-negative constant for the frequency mapping weight $v_{\text{map2},1c}$ during the first stage training process
$\lambda^1_{\text{map3},rq}$	The non-negative constant for the output mapping weight $v_{\text{map3},rq}$ during the first stage training process
$\lambda^2_{\text{map1},ij}$	The non-negative constant for the input mapping weight $u_{\text{map1},ij}$ during the second stage training process
$\lambda^2_{\text{map2},10}$	The non-negative constant for the frequency mapping weight $u_{\text{map2},10}$ during the second stage training process
$\lambda^2_{\text{map2},11}$	The non-negative constant for the frequency mapping weight $u_{\text{map2},11}$ during the second stage training process
$\lambda^2_{\text{map3},rp}$	The non-negative constant for the frequency mapping weight $u_{\text{map3},rp}$ during the third stage training process
$\sigma(\cdot)$	The sigmoid function
τ	Time delay parameter
ω	The frequency input of the entire knowledge-based model
ω_E	The frequency input of the empirical model
$\Delta\omega_E$	The BP error at the output of the frequency mapping neural network

$\lvert \cdot \rvert$	l_1 norm
$\lvert\lvert \cdot \rvert\rvert$	l_2 norm

Chapter 1

Introduction

1.1 Background and Motivation

Nowadays in microwave design, because of the increasing component complexity, tighter component tolerances and shorter design cycles, the industry's emphasis on time-to-market and low cost are placing enhanced demands on computer-aided design (CAD) methods and tools for microwave modeling and design. With the rapid development in the semiconductor industry, new devices continually evolve and new models are constantly needed. The challenges for CAD researchers are not only to develop more accurate models, but also to develop efficient and systematic CAD methods. This thesis aims to propose unified and automated CAD methods using neural network technique and advanced extrapolation technique to develop accurate models and speed up the model development process.

Artificial neural networks (ANN) have been recognized as powerful vehicles in microwave modeling and design [1]-[3]. ANNs learn electromagnetic (EM)/physics behaviors from microwave component data through training process and the trained

neural networks are then used as fast and accurate models for efficient high-level circuit and system design. Significant speed-up of CAD by using neural models in place of CPU-intensive EM/physics models resulted in a drive to develop advanced neural modeling techniques [4]-[6].

Neural network models can represent devices accurately by training them using an input/output data set. Usually, more training data leads to better model accuracy [7]. Two types of advanced approaches have been presented in the literature where good model accuracy can be achieved with less training data: one is automated model generation technique [4] and the other is knowledge-based neural network technique [1], [8]. These techniques are useful because less training data means faster model development. Automated model generation technique integrates all the subtasks involved in neural modeling, such as data generation, neural network selection, training and validation, thereby facilitating a efficient and automated model development framework. On the other hand, knowledge-based models incorporate application specific knowledge such as microwave empirical functions or equivalent circuits into neural network structure to reduce the amount of training data and enhance the model generalization performance. However, in the existing literature, the knowledge-based neural model development is typically a manual process. The data sampling process and the knowledge-based neural model structure selection still depend on designers' experience. The automation of the knowledge-based neural model development process still remains an open topic.

In neural modeling for microwave applications, there is another important aspect to consider. In general, a well-trained neural network model has good accuracy

and can represent the component behavior it learned within the training region [1]. However, the accuracy of the neural network model outside the training region decreases and this creates limitations when neural models are used to explore design solutions in a larger range than the model training range. For example, when a neural network model is used in the iterative computational loops during EM optimization or harmonic balance (HB) simulation, the iterative variables which are the inputs of the neural model may go beyond the neural model training range during the iterative process. The poor performance of the trained neural model outside the training range may mislead the iterative process into slow convergence or even divergence. Therefore, extrapolation methods on neural network models [9]-[12] have been described in the literature to address this issue. These model-level and neuron-level extrapolation methods use the neural model and its derivative information at the boundaries of the neural network training region, or use modified neuron activation function to help model extrapolation.

However, for extrapolation methods on neural network models, the smoothness of the neural model outside the training is very important in microwave simulation and optimization. The smooth tendency of the model outside the training range across different directions can make the neural model more robust and lead to faster convergence in microwave simulation and optimization. How to make the neural model smooth outside the training region across different directions still remains an open subject.

1.2 List of Contributions

The main objective of this thesis is to develop a unified and systematic automated modeling method for microwave applications using knowledge-based neural network with l_1 optimization, and propose an advanced multi-dimensional extrapolation technique for neural-based microwave modeling and design. In this thesis, the subtasks in generating a knowledge-based model like initial optimization, unit mapping, data generation, model structure adaptation, training and testing are integrated into a unified framework. A new adaptive sampling method is proposed to make the automated modeling process faster. A new structure adaptation algorithm with l_1 optimization method is proposed to automatically determine the type and the topology of the mapping structure in a knowledge-base neural model. In addition, a novel and efficient multi-dimensional extrapolation technique is proposed to make neural network based microwave models behave well outside the training region. In this thesis, the following significant contributions are made:

- An automated neural modeling method with a new adaptive sampling technique is proposed [13], [14]. The proposed method automates data generation, determination of data distribution, selection of the number of hidden neurons, and model training in a systematic framework. The proposed adaptive sampling technique incorporates efficient interpolation approaches to avoid intermediate training during the modeling process and speed up the model development. In this method, automated model generation is extended from generating pure neural network models to generating knowledge-based neu-

ral network models for the first time. Compared to existing knowledge-based modeling methods and existing automated model generation algorithms, the proposed method can achieve an accurate knowledge-based neural network model with less training data in a shorter modeling time.

- A unified automated model structure adaptation algorithm for knowledge-based modeling using l_1 optimization is proposed to automatically determine the type and the topology of the mapping structure in a knowledge-based neural network model [15], [16]. Knowledge-based neural network modeling techniques using space-mapping concept have been demonstrated in the existing literature as efficient methods to overcome the accuracy limitations of empirical/equivalent circuit models when matching new electromagnetic (EM) data. For different modeling problems, the mapping structure in the knowledge-based model should be different. By encompassing various types of mappings of the knowledge-based neural network model in the existing literature, we propose a new unified model structure and derive new sensitivity formulas for the training of the unified model. The proposed l_1 formulation of modeling can force some weights of the mapping neural networks to zeros while leaving other weights as nonzeros. We utilize this feature to allow l_1 optimization to automatically determine which mapping is necessary and which mapping is unnecessary. Using the proposed l_1 optimization method, the mapping structure can be determined to address different needs of different modeling problems. In this way, the proposed algorithm is more systematic and can further speed up the knowledge-based modeling process than existing

knowledge-based modeling algorithms.

- A novel and efficient multi-dimensional extrapolation method is proposed for neural-based microwave modeling and design [17] to make the model can be reliably used outside the training region. In this proposed technique, the given training data can be randomly distributed in the input space, and the boundaries of the training region can be arbitrary. The unknown target values of the model outside the training range are formulated as optimization variables and are determined by optimization, such that 1st order continuity of model outputs versus inputs is preserved and 2nd order derivatives are minimized everywhere. In this way, for locations in the input space where training data exist, the model will be trained to fit the data; for locations in the input space where no training data exist, the output values of the model will be optimized to preserve the tendency of the model from the training boundaries and make the model maximally smooth across all directions everywhere outside the training region. Compared to existing extrapolation methods for neural networks, the proposed extrapolation technique makes neural models more robust, resulting in faster convergence in microwave simulation and optimization involving neural model inputs as iterative variables.

1.3 Outline of the Thesis

The rest of the thesis is organized as follows:

In Chapter 2, a literature review of neural network applications for microwave

modeling and design is presented. An overview of basic concepts of artificial neural network (ANN), automated model generation, knowledge-based neural network using space mapping concept are presented. Classical space mappings in knowledge-based neural network model such as input space mapping, output space mapping and frequency space mapping are also reviewed. In addition, extrapolation problem of neural network models and recent extrapolation techniques for neural network models such as model-level extrapolation methods and neuron-level extrapolation methods are also reviewed.

Chapter 3 presents an advanced algorithm to automate the process of developing knowledge-based models with a new adaptive sampling technique for microwave applications. It integrates and automates data generation, determination of data distribution, selection of the number of hidden neurons and model training in a systematic framework. This proposed method uses less training data and shorter time model development time than existing techniques. This algorithm is demonstrated through a microwave filter modeling example.

Chapter 4 presents a more elegant and unified automated model structure adaptation algorithm for knowledge-based parametric modeling. The proposed method can automatically determine the type and topology of the mapping structure in a knowledge-based model. An accurate knowledge-based model with the most suitable and compact structure for a specific modeling problem can be achieved by using the proposed method. Compared to the existing literature on the model structure selection method, our proposed method is a more systematic technique and can further speed up the process of developing a knowledge-based neural network model.

This technique is illustrated by three microwave filter modeling examples.

Chapter 5 presents an advanced multi-dimensional extrapolation technique for neural-based microwave modeling and design to address the modeling challenges in microwave simulation and optimization, such as EM optimization and large-signal HB simulation. The proposed technique guarantees best model accuracy where training data is given and guarantees maximum smoothness where training data is not given. The proposed extrapolation technique makes neural models for both passive and active components more robust, resulting in faster convergence in microwave simulation and optimization involving neural model inputs as iterative variables. The validity of the proposed technique is demonstrated using both EM optimization example and nonlinear microwave simulation examples.

Finally, in Chapter 6, conclusions and discussions on possible directions for future work are presented.

Chapter 2

Literature Review

2.1 Review of Artificial Neural Network for Microwave Modeling and Design

Neural networks, also called Artificial Neural Networks (ANN), are information processing systems with their design inspired by the studies of human-brain learning and generalization abilities [1]-[3]. The fact that neural networks can be trained to learn any arbitrary nonlinear input-output relationships from corresponding data has resulted in their use in a number of areas such as pattern recognition, speech processing, control, bio-medical engineering etc. Nowadays, ANNs have been applied to RF and microwave computer-aided design (CAD) problems as well. Neural networks are first trained to model the electrical behavior of microwave components/circuits. These trained neural networks, often referred to as neural network models (or simply neural models), can then be used in high-level simulation and design, providing fast answers to the task they have learned [18]-[24]. In microwave modeling area, neural models are much faster than detailed EM/physics models

[1], [25]-[27], more accurate than polynomial and empirical models [28], allow more dimensions than table lookup models [29], and are easier to develop when a new device/technology is introduced [30].

Neural network techniques have been used for a wide variety of microwave applications such as transmission line components [5], [8], vias [31], bends [32], coplanar waveguide (CPW) components [33], spiral inductors [7], field effect transistor (FET) devices [25], [34], heterojuction bipolar transistor (HBT) devices [35], high electron mobility transistor (HEMT) devices [36], [37], filters [38]-[41], amplifiers [42]-[45], mixers [46], antennas [47], embedded passives [4], [26], [27], packaging and interconnects [48], etc. Neural networks have also been used in circuit simulation and optimization [3], [25], [49], signal integrity analysis and optimization of very-large-scale-integration (VLSI) interconnects [48], [50], microstrip circuit design [51], process design [52], microwave impedance matching [53], inverse modeling [54], measurements [55], synthesis [25], [56] and behavioral modeling of nonlinear RF/microwave subsystems [57].

An increased number of RF/microwave engineers and researchers have started taking serious interest in this emerging technology. A variety of ANN structures has been developed in recent years, such as multilayer perceptron (MLP) neural network [1], [57], radial basis function networks (RBF) [42], [58], recurrent neural network (RNN) [46], [59], time-delay neural networks (TDNN) [60], dynamic neural networks (DNN) [61], [62], knowledge-based neural networks (KBNN) [8] and state-space dynamic neural networks (SSDNN) [63]-[66]. In addition, automated model generation method [4] and extrapolation techniques [11], [12] for neural net-

work modeling have been described as recent developments in ANN techniques for microwave modeling.

2.2 Basic Concepts of Neural Network Modeling

2.2.1 Neural Network Structures

A neural network has at least two physical components, i.e., the processing elements and the connections between them. The processing elements are called neurons, and the connections between the neurons are called links. Every link has a weighting parameter associated with it. Each neuron receives the stimulus from the neighboring neurons which are connected to it, processes the information and generates an output response. Neurons who receive stimulus from the outside of the neural network (i.e., not from neurons of the neural network) are named input neurons. Neurons whose outputs are used externally are named output neurons. Neurons who receive stimulus from other neurons and whose output is a stimulus for other neurons in the neural network are named hidden neurons. A neural network structure defines how information is processed inside a neuron and how the neurons are connected. Different neural network structures can be constructed by using different processing elements and by the specific pattern how they are connected.

A variety of neural network structures have been developed for microwave modeling and design. Here we describe several neural network structures that are commonly used for microwave modeling and design [1], such as multilayer perceptrons (MLP), radial basis function networks (RBF), recurrent neural networks (RNN) and dynamic neural networks (DNN).

Multilayer perceptrons (MLP) [1], [57] are the basic and most frequently used structure of neural networks. In the MLP structure, the neurons are grouped into different layers. The first layer is called input layer and the last layer is called output layer. The remaining layers between these two layers are called hidden layers. Input layer neurons simply relay the external inputs to the neural network. Hidden layer neurons have smooth switch-type activation functions. Output layer neurons can have simple linear activation functions. In general, an MLP consists of one input layer, one or more hidden layers, and one output layer, as shown in Fig. 2.1.

According to the universal approximation theorem for MLP proved by Cybenko [67] and Hornik et al. [68], a three layer perceptron (a perceptron is defined as an algorithm for supervised learning) with enough hidden neurons is able to approximate an arbitrary nonlinear, continuous, multi-dimensional function with any desired accuracy. In practice, how to choose the precise number of hidden neurons required for specific modeling problem remains an open question. The number of hidden neurons depends on the degree of nonlinearity and dimensionality of the original problem. The ongoing research in this direction includes algorithms such as constructive algorithms [69], network pruning [70], regularization [71], and automated model generation [4].

MLPs with at least one hidden layer are necessary and sufficient for arbitrary nonlinear function approximation. Practically, three-layer and four-layer perceptrons (i.e., one or two hidden layers) are most commonly used in neural network applications. In practice, four-layer perceptrons have better performances in modeling higher nonlinear problem than three-layer perceptrons which may need too

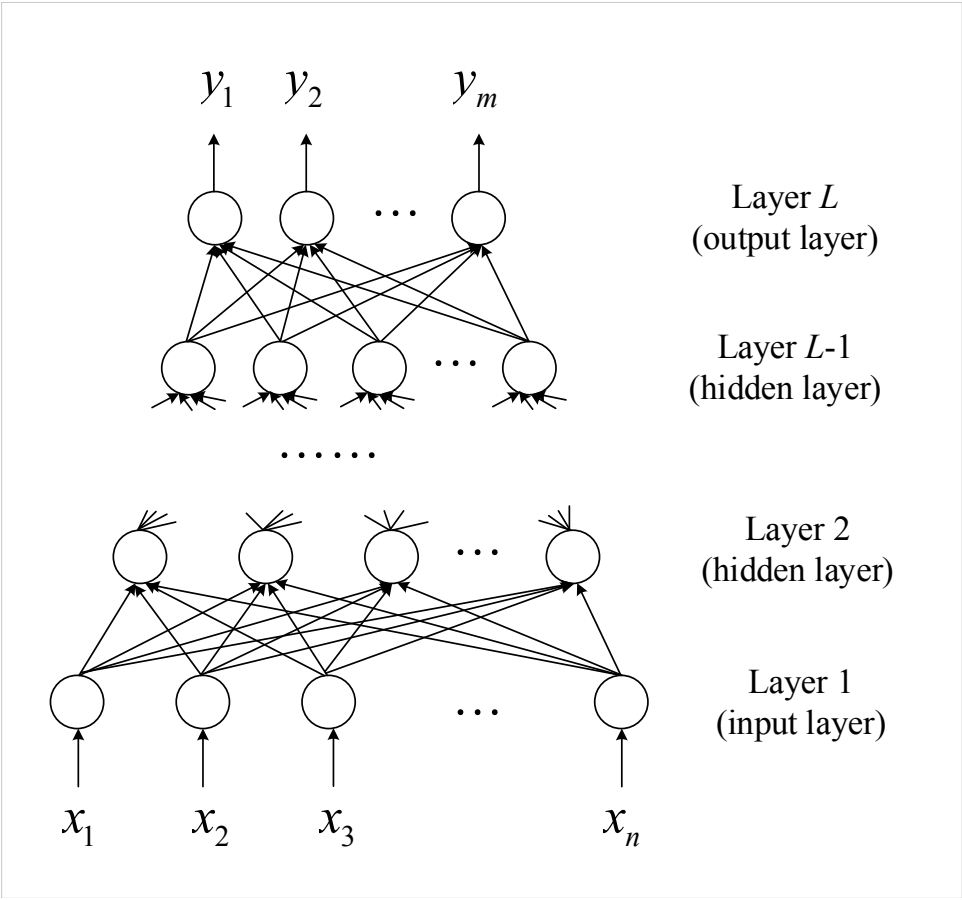


Figure 2.1: Multilayer perceptrons (MLP) structure containing one input layer with n input neurons, one output layer with m output neurons, and several hidden layers.

many hidden neurons. Three-layer perceptrons are usually preferred when generalization capability of the function approximation is a major concern [72], because fewer hidden neurons are needed in the resulting neural network. On the other hand, four-layer perceptrons perform better in pattern classification problems [73], because more layers allow more effective representation of hierarchical information in the original problem.

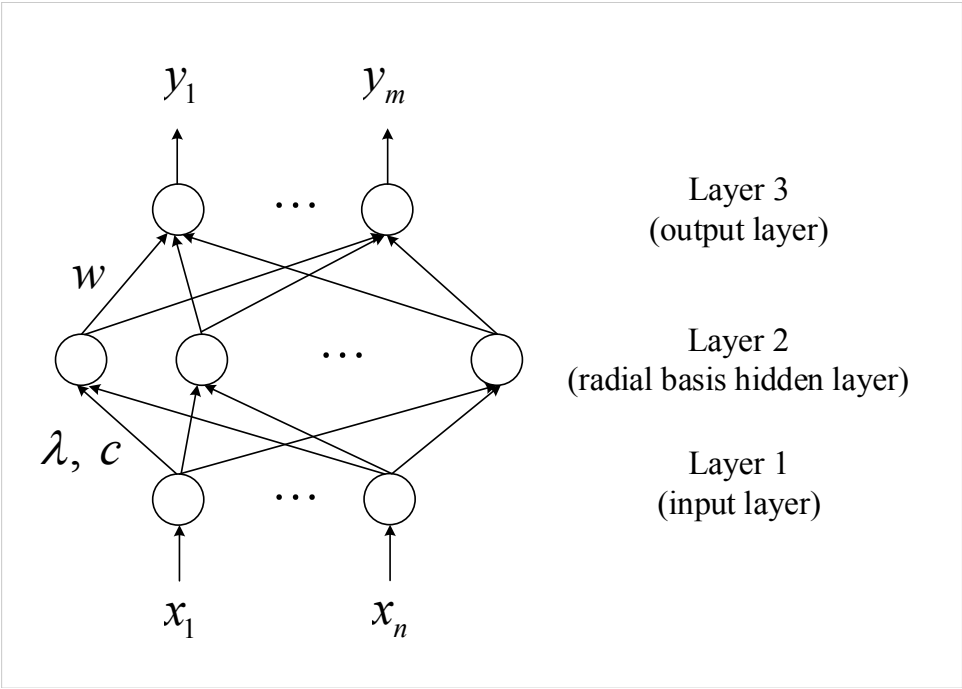


Figure 2.2: The structure of radial basis function (RBF) neural network.

Radial basis function (RBF) neural network [42], [58] is another popular neural network structure which has an input layer, a hidden layer using radial basis activation functions for hidden neurons and an output layer[58], [74]. A typical RBF neural network is illustrated in Fig. 2.2. The parameters c and λ are centers and standard deviations of radial basis activation functions. The Gaussian and multiquadratic functions are the most commonly used radial basis activation functions in RBF neural networks. According to the universal approximation theorem for RBF networks by Park and Sandberg [75], an RBF neural network with a sufficient number of hidden neurons can approximate any given nonlinear function to any degree of accuracy.

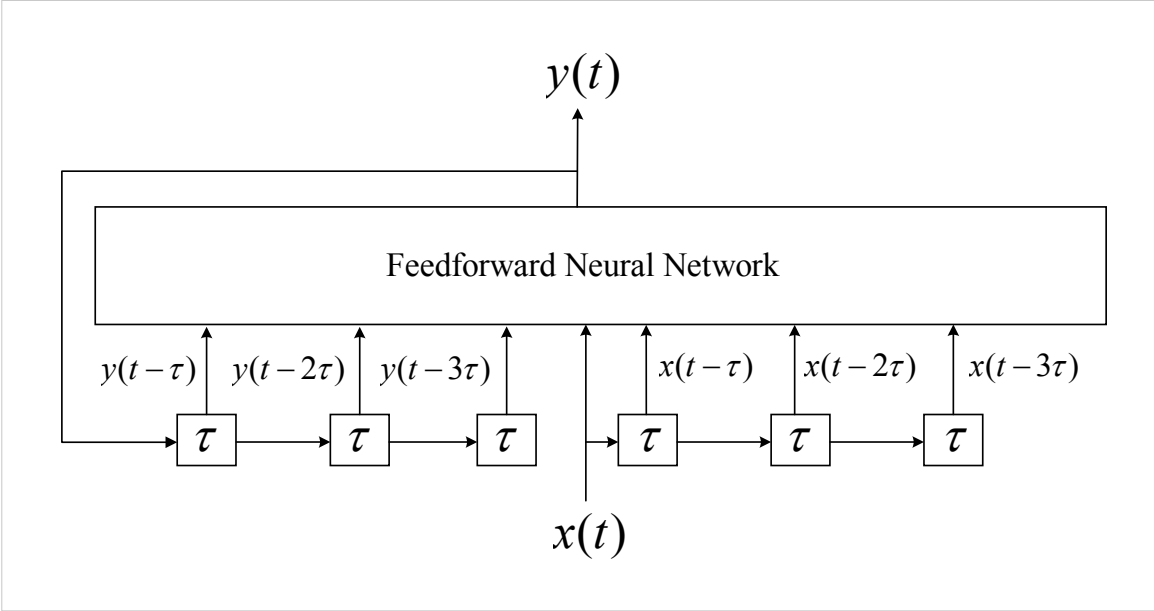


Figure 2.3: A recurrent neural network (RNN) structure using three delays for output feedback.

Both MLP and RBF belong to a general class of neural networks called feedforward networks, where the neural network accepts the input information sent to input neurons and proceeds to produce the response at the output neurons. These neural network structures cannot allow time-domain behaviors of a dynamic system to be modeled. It is because that the outputs of a dynamic system depend not only on the present inputs, but also on the history of the system states and inputs. In order to model such behaviors, a recurrent neural network (RNN) structure with feedback of delayed neural network outputs is described in [76]-[78], as shown in Fig. 2.3. The feedforward network together with the delay and feedback mechanisms results in a RNN structure. The feedforward network module in a RNN could be any of the standard feedforward neural network structures (e.g., MLP, RBF).

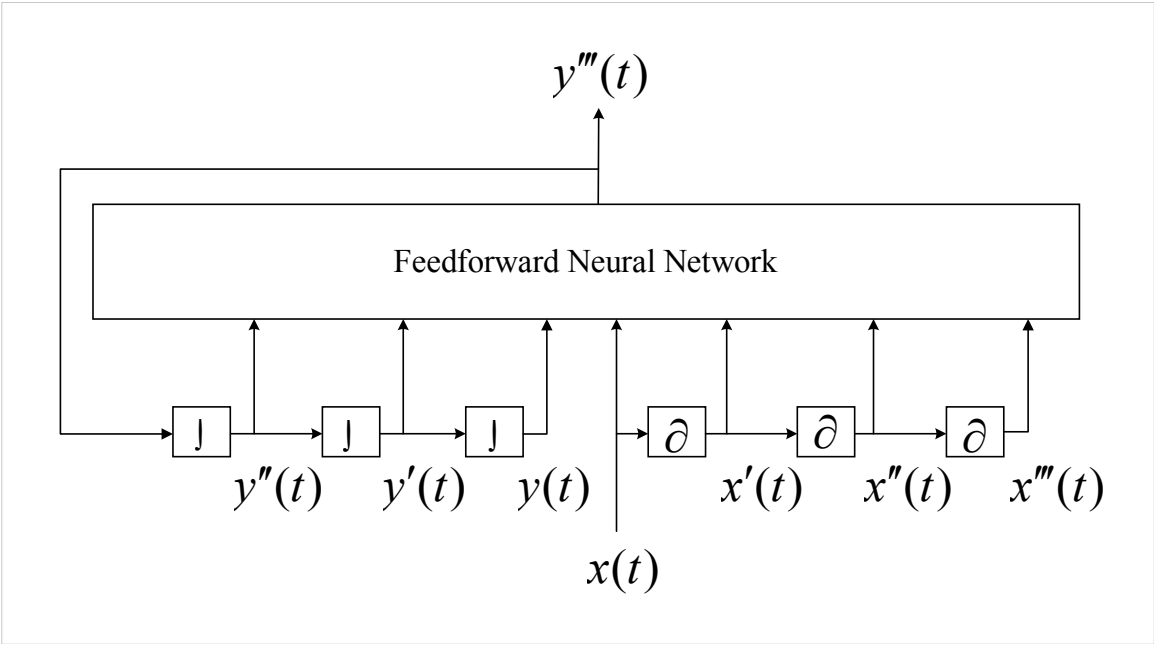


Figure 2.4: The structure of dynamic neural network.

While the RNN approach achieves a discrete time-domain model for nonlinear microwave modeling problems, another specific type of neural network called dynamic neural network (DNN) is described in [61], [62] to obtain models for continuous time-domain dynamic systems. The structure for a DNN is illustrated in Fig. 2.4. The DNN model directly represents the dynamic input-output relationship of the problem. The input-output signals and their time derivatives are related through a feedforward network. Similar to RNN, the feedforward network module in a DNN could be any of the standard feedforward neural network structures (e.g., MLP, RBF).

2.2.2 Neural Network Model Development

A neural network cannot represent any RF/microwave device/circuit behavior unless it is well trained with corresponding data. A systematic neural model development process includes data generation, data scaling, training, validation, and testing.

Let $\boldsymbol{x}$ represent an n-vector containing the design parameters of a RF/microwave device/circuit (e.g., gate voltage and drain voltage of an FET, width and length of an EM structure). Let $\boldsymbol{y}$ represent an m-vector containing the response of the device/circuit under consideration (i.e., drain current of an FET, or S-parameters of an EM structure). The theoretical EM/physics relationship between inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$ can be represent by a neural network function $\boldsymbol{f}_{ANN}$, formulated as

$$\boldsymbol{y} = \boldsymbol{f}_{ANN}(\boldsymbol{x}, \boldsymbol{w}) \tag{2.1}$$

where $\boldsymbol{w}$ represents a vector containing all trainable neural network weights. A fast and accurate neural model is developed by training the neural network function $\boldsymbol{f}_{ANN}$ with a set of simulated/measured data called the training data. The training data is denoted by input-output sample pairs $\{(\boldsymbol{x}_k, \boldsymbol{d}_k), k \in T_r\}$, where $\boldsymbol{d}_k$ represents the measured/simulated output data for the input $\boldsymbol{x}_k$, and T_r represents the index set of training data.

For training purpose, an error function $E(\boldsymbol{w})$ is defined as,

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{k \in T_r} \| \boldsymbol{f}_{ANN}(\boldsymbol{x}_k, \boldsymbol{w}) - \boldsymbol{d}_k) \|^2 \tag{2.2}$$

The primary objective of neural network training is to adjust the weigh parameters

$\boldsymbol{w}$ such that $E(\boldsymbol{w})$ is minimized. Multiple training algorithms are developed for RF/microwave neural modeling such as backpropagation [79], conjugate gradient [80], Quasi-Newton [81], Levenberg-Marquardt [82], Genetic Algorithms [83] and Simulated Annealing [84]. Once trained, an independent set of input-output samples, called test data, is used to test the accuracy of the neural network model. Normally, the test data should be generated within the same input range as the training data but contains the input-output samples which are never used in training stage. If both training and testing results are satisfied with the use desired accuracy, the neural network model are well trained and can be used in circuit simulation and design optimization. A flow chart of neural network training, validation and testing is shown in Fig. 2.5.

2.2.3 Automated Model Generation for Neural Network Models

As mentioned in the previous subsection, the overall neural model development involves several subtasks like data generation, neural network selection, training and testing. Conventionally, all these subtasks are manually carried out in a sequential manner independent of one another, which requires intensive human effort. To improve modeling efficiency, an automated neural model development algorithm is presented in [4] to integrates all subtasks in neural modeling into one unified task. Starting with minimal amounts of training data and a three-layer MLP network, automated model generation algorithm performs dynamic sampling and neural-network structure adaptation (i.e., selection of the number of hidden neurons in

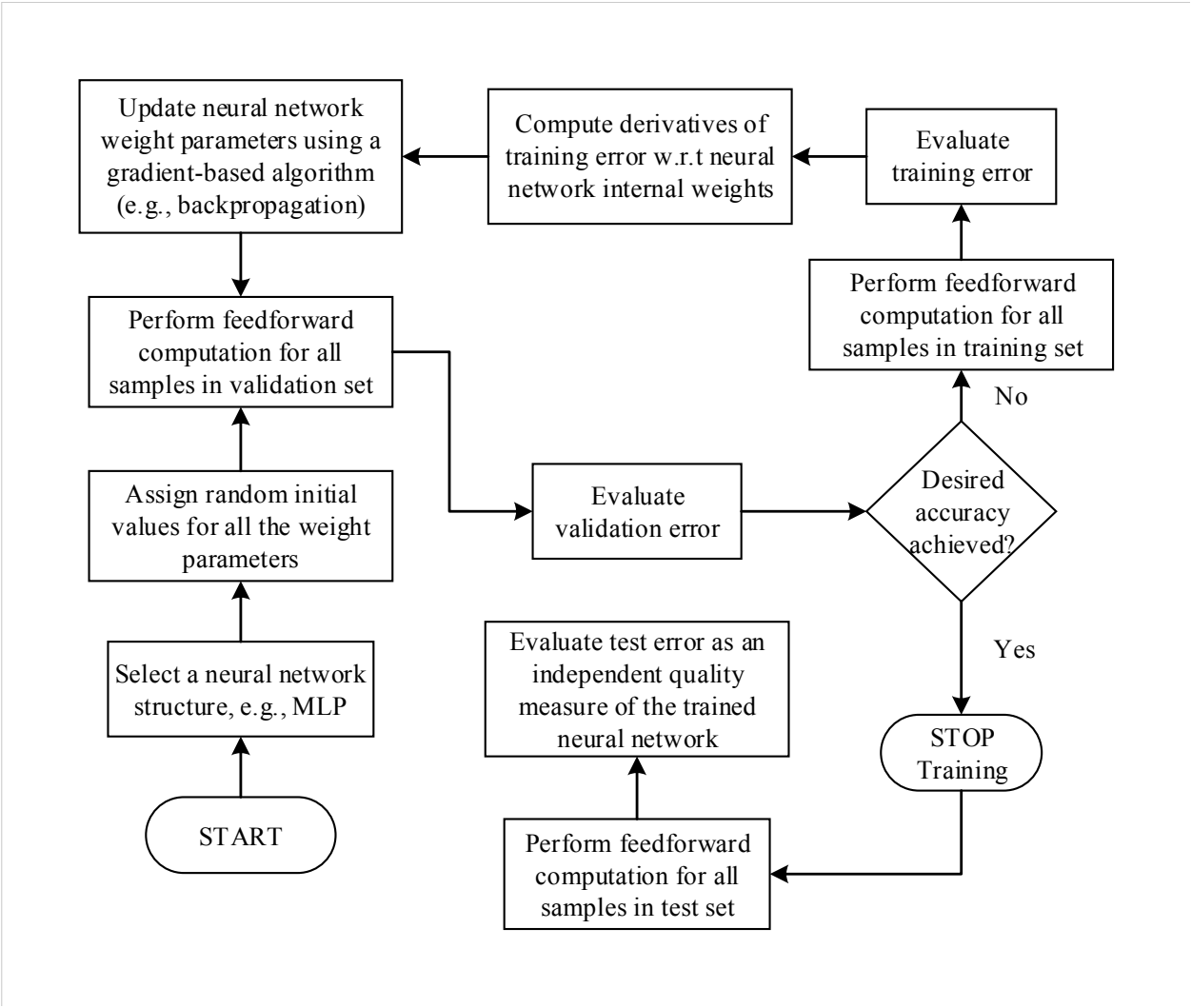


Figure 2.5: The flowchart of neural network training, validation and testing.

the MLP). It uses an adaptive data sampling algorithm to automatically drive the simulators to generate new data during training, and adjusts the neural-network size (i.e., add more neurons in the hidden layer) based on the neural network training error. In this way, automated model generation algorithm effectively converts

the heavy human workload into intensive computation by computer algorithm, thus realizing the automation of microwave neural network modeling.

2.3 Knowledge-Based Neural Network Using Space Mapping Concept

2.3.1 Knowledge-Based Neural Network

The knowledge-based neural network (KBNN) approach [8] is a modeling method exploiting existing knowledge in the form of empirical or analytical approximations models together with neural networks. This method enhances neural model accuracy, especially for the data not seen during training (generalization capability), and reduces the need for a large amount of training data. The comprehensive structure of the knowledge-based neural network is shown in Fig. 2.6.

In KBNN, the microwave knowledge is embedded as part of the neural network internal structure. There are six layers in the KBNN structure, namely the input layer, the knowledge layer, the boundary layer, the region layer, the normalized region layer, and the output layer. The input layer accepts the external inputs to the model. The knowledge layer is the place where microwave knowledge resides, complementing the ability of learning and generalization of neural networks. The boundary layer incorporates knowledge in the form of problem dependent boundary functions. The region layer contains neurons for constructing regions from boundary neurons. The normalized region layer contains rational function based neurons to normalize the outputs of the region layer. The output layer consists of second-order neurons combining knowledge neurons and normalized region neurons.

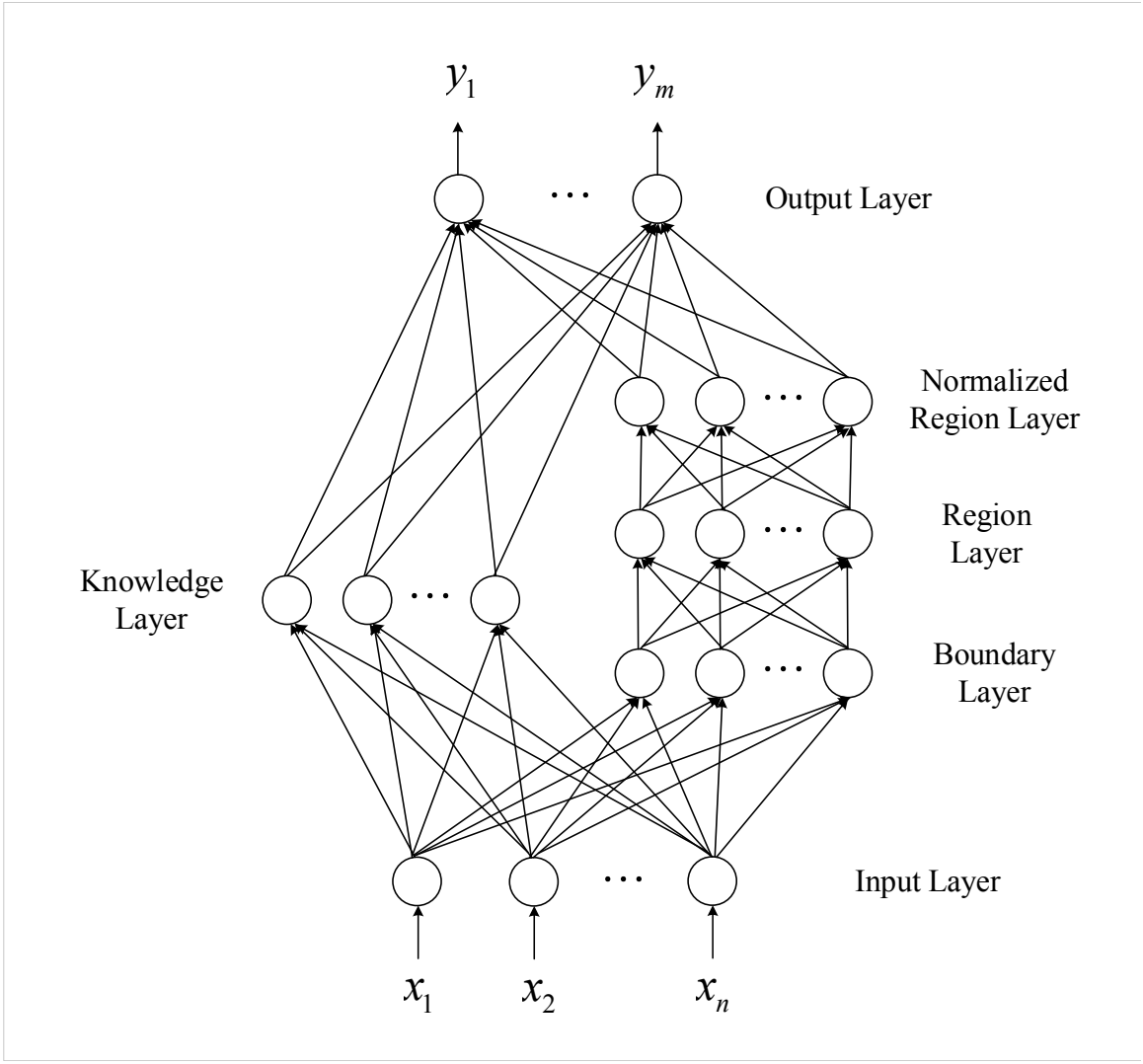


Figure 2.6: Structure of a knowledge-based neural network [8].

The KBNN approach combines microwave empirical experience with the power of learning of neural networks to develop faster and more accurate models. Extrapolation capability of the KBNN is also enhanced by embedding knowledge in the model. Another microwave modeling and optimization technique with prior knowl-

edge, using an advanced optimization concept named space mapping, is discussed in next subsection.

2.3.2 Space Mapping Concept and Space-Mapped Neuromodeling

The space mapping technique was originally developed by Bandler *et al.* [85]. Space mapping is a concept for circuit design and optimization, which combines the computational efficiency of coarse models with the accuracy of fine models. The coarse models are typically empirical functions or equivalent circuit models, which are computationally very efficient, but often valid only in a limited range of input space, beyond which the model predictions become inaccurate. On the other hand, detailed or "fine" models can be provided by an EM simulator, or even by direct measurements. The detailed models are very accurate, but can be expensive (e.g., CPU-intensive simulations). The space mapping technique establishes a mathematical link between the coarse and the fine models and directs the bulk of the CPU-intensive computations to the coarse model, while preserving the accuracy offered by the fine model.

Let $\boldsymbol{x}_c$ and $\boldsymbol{x}_f$ represent the input parameters of the coarse and fine models, respectively, and let $\boldsymbol{R}_c(\boldsymbol{x}_c)$ and $\boldsymbol{R}_f(\boldsymbol{x}_f)$ represent the corresponding model responses. $\boldsymbol{R}_c$ is much faster to calculate, but less accurate than $\boldsymbol{R}_f$. The aim of space mapping is illustrated in Fig. 2.7, i.e., to find an appropriate mapping $\boldsymbol{P}$ from the fine model input space $\boldsymbol{x}_f$ to the coarse model input space $\boldsymbol{x}_c$

$$\boldsymbol{x}_c = \boldsymbol{P}(\boldsymbol{x}_f) \tag{2.3}$$

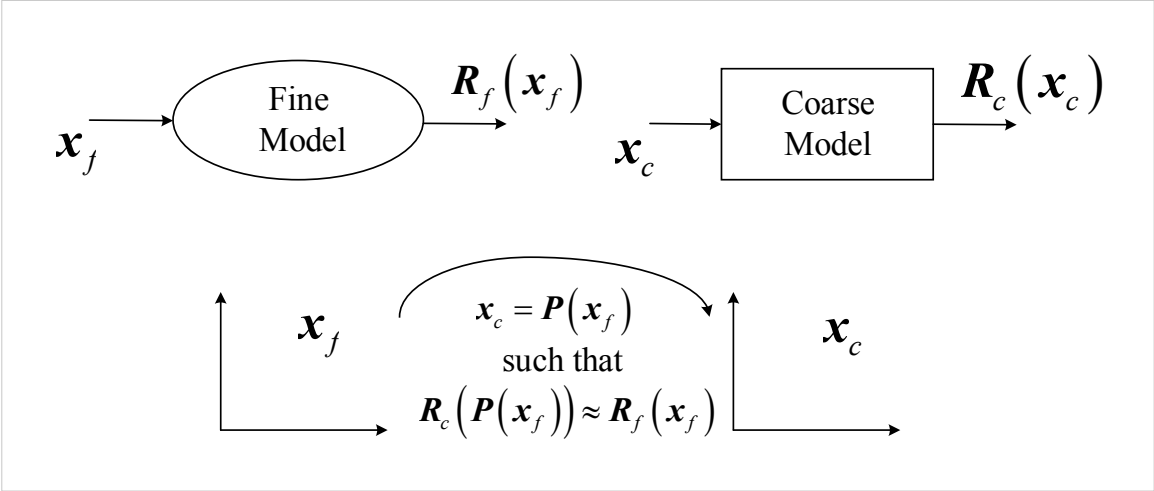


Figure 2.7: Illustration of the aim of space mapping.

such that

$$\boldsymbol{R}_c\left(\boldsymbol{P}\left(\boldsymbol{x}_f\right)\right) \approx \boldsymbol{R}_f\left(\boldsymbol{x}_f\right). \tag{2.4}$$

Once the mapping is found, the coarse model can be used for fast and accurate simulations.

The space-mapped neural network approach was firstly proposed in [32] based on the space mapping concept [85] in microwave optimization area. Recently, innovative schemes combining space mapping technology and neural network for microwave modeling are presented. The fundamental idea is to construct a nonlinear multidimensional vector mapping function $\boldsymbol{P}$ from fine to coarse input space, or from coarse to fine output space using one or more neural network. This can be done in a variety of ways, to make better use of the coarse model information for developing the neuromodel. The implicit knowledge in the coarse model not only allows us to decrease the number of training data needed, but also to reduce the

complexity of the neural model topology and to improve the generalization performance. A number of papers cover different kinds of space mappings in the literature, including input space mapping [86]-[88], frequency space mapping [89], and output space mapping [90], [91]. In the following subsections, these different kinds of space mappings are reviewed.

2.3.3 Input Space Mapping

The input space mapping method [88] establishes a mathematical link between the design spaces of both the coarse model and the fine model. In microwave modeling, the input space mapping is used to modify the values of the model inputs (e.g., the geometrical design parameters) to a different set of values to be supplied to the coarse model, so that the modified coarse model response can match the fine model response. The structure of a neural model with input mapping is illustrated in Fig. 2.8. Vectors $\boldsymbol{x}_c$ and $\boldsymbol{x}_f$ represent the design parameters of the coarse model and the fine model, respectively, whose corresponding model responses are in vectors $\boldsymbol{R}_c$ and $\boldsymbol{R}_f$. Let ω_c and ω_f represent the frequency variable of the coarse model and the fine model. The difference between the available coarse model and fine model is used to train the corresponding neural network.

2.3.4 Frequency Space Mapping

For those cases where the shapes of the fine model and coarse model responses are nearly identical, but shifted in frequency, the frequency mapped neuromodeling technique [89] simulates the coarse model with the same values of inputs used by

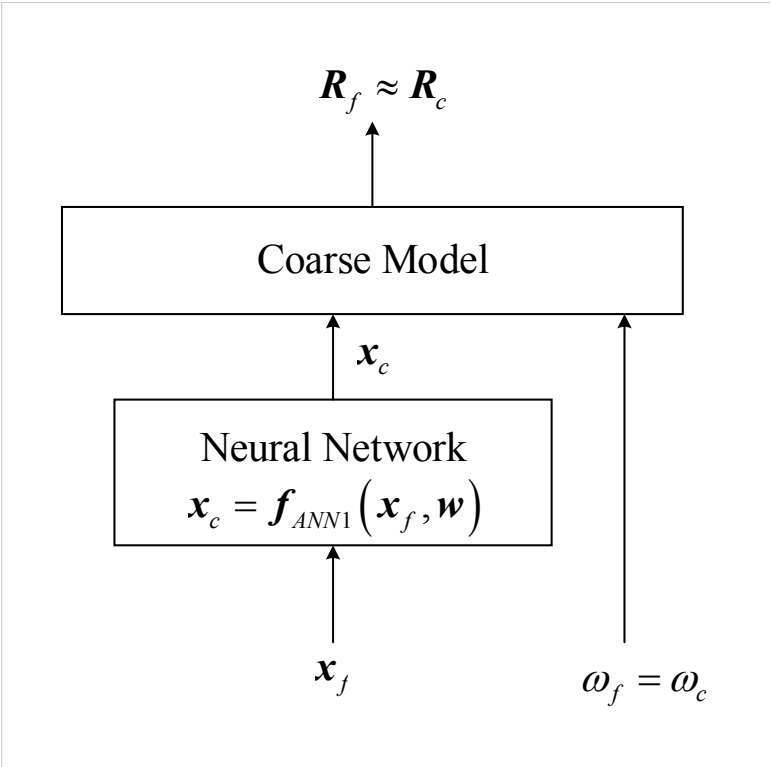


Figure 2.8: Structure of a neural model with input mapping.

the fine model, but at a different frequency to align both responses. The structure of a neural model with frequency mapping is illustrated in Fig. 2.9.

2.3.5 Output Space Mapping

Output space mapping is a technique introduced to enhance the robustness of the space mapping modeling process in case other space mappings cannot provide sufficient matching with the EM data [92], [93]. Typically, output space mapping improves the performance of the space-mapping algorithm when applied as an auxiliary mapping. If the range of the coarse model is substantially different from the

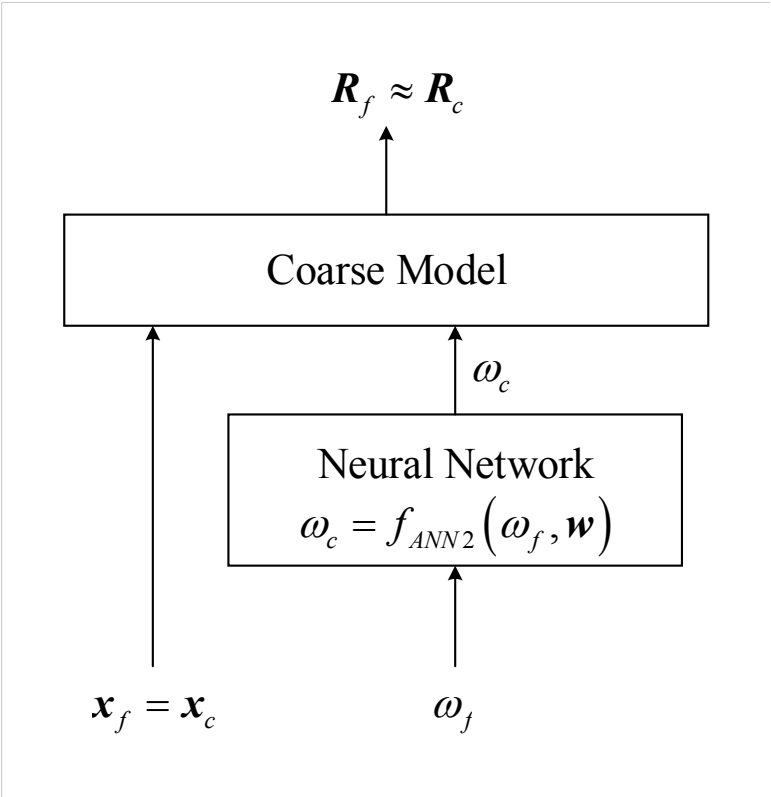


Figure 2.9: Structure of a neural model with frequency mapping.

range of the fine model, output space mapping is practically mandatory to make the space mapping algorithm perform reasonably well. The structure of a neural model with output mapping is illustrated in Fig. 2.10.

2.4 Extrapolation Problem of Neural Network Model

In general, a neural network model, after being trained for a particular range of data, is very good at representing the original problem within the training region. However, outside this region, the accuracy of the model deteriorates very rapidly

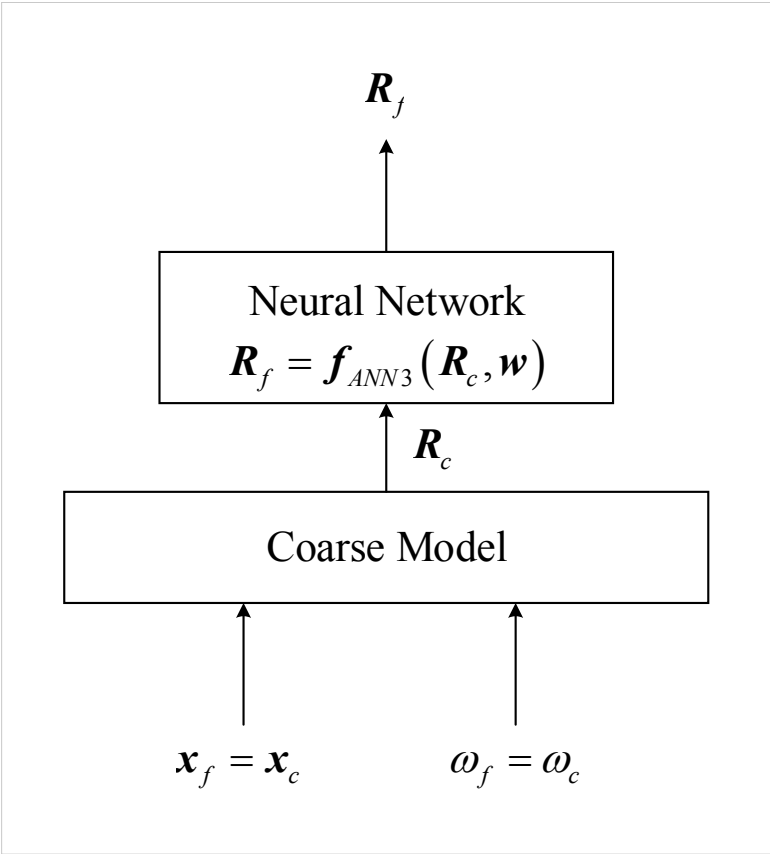


Figure 2.10: Structure of a neural model with output mapping.

because of saturation of the activation functions in the hidden layer of the neural network structure. This creates limitations for use of neural models in iterative computational loops such as EM optimization and harmonic balance (HB) simulation where the range of the iterative variables may need to be larger than the neural model training range. This is an important issue for microwave design involving physical/geometrical design parameters and nonlinear circuit simulation. The poor performance of conventional neural model outside the training range may mislead the iterative process into slow convergence or even divergence.

A variety of studies have been conducted on the extrapolation [9] of neural networks, such as knowledge-based neural models [8], neural network model-level extrapolation methods [10], [11], and neuron-level extrapolation method [12]. A general review of the existing extrapolation methods on neural network modeling will be presented in the following section.

2.5 Extrapolation Methods on Neural Network Models

To address the extrapolation problem on neural network models, knowledge-based neural network [8] use microwave empirical formulas with neural networks to help model extrapolation. Mathematical extrapolation methods [10], [11] have been developed to address the issue when empirical models are not available for the modeling problem. These model-level methods are based on the model and its derivative information at the boundary of the neural network training range in a multi-dimensional input space. Recently, a simple neuron-level extrapolation method [12] is presented to address the problem in case when the training boundaries are irregular.

2.5.1 Model-Level Extrapolation Method

The model-level extrapolation method was firstly developed in [11] to address the task of using microwave neural models far beyond their training range. It develops a training process to formulate a set of base points to represent a training region. An adaptive base point selection method is developed to identify the most significant

subset of base points upon any given values of model input. Combining quadratic approximation [94], [95] with the information of the model at these base points including both the input/output behavior and its derivatives, the model-level extrapolation technique is able to reliably extrapolate the performance of the model from training range to a much larger region.

Given inputs $\boldsymbol{x}$, the model-level extrapolation technique will firstly search the set of basis points to find several base points closest to $\boldsymbol{x}$. Then a smooth quadratic function will be used to best match the behavior of these base points, including both the input/output behavior and their derivatives. A weighting matrix $\boldsymbol{W}$ is defined to regulate the amount of influence of the base points. The quadratic approximation using $\boldsymbol{W}$ is formulated as

$$\boldsymbol{W} \cdot \boldsymbol{F} \cdot \boldsymbol{V}_p = \boldsymbol{W} \cdot \boldsymbol{r} \tag{2.5}$$

where $\boldsymbol{V}_p$ represents the parameters in quadratic function, and $\boldsymbol{F}$, $\boldsymbol{r}$ represent the input/output information and the derivatives, provided by the adjoint neural network [96]. Least square method [95] is applied to solve (2.5). The computation of the model-level extrapolation is illustrated in Fig. 2.11

The model-level extrapolation technique also has limitations. In microwave modeling such as time-delay neural network for nonlinear transistor and power amplifier modeling [97], the multi-dimensional training boundaries can be irregular or unknown. In this case, existing model-level extrapolation technique becomes less effective or complicated.

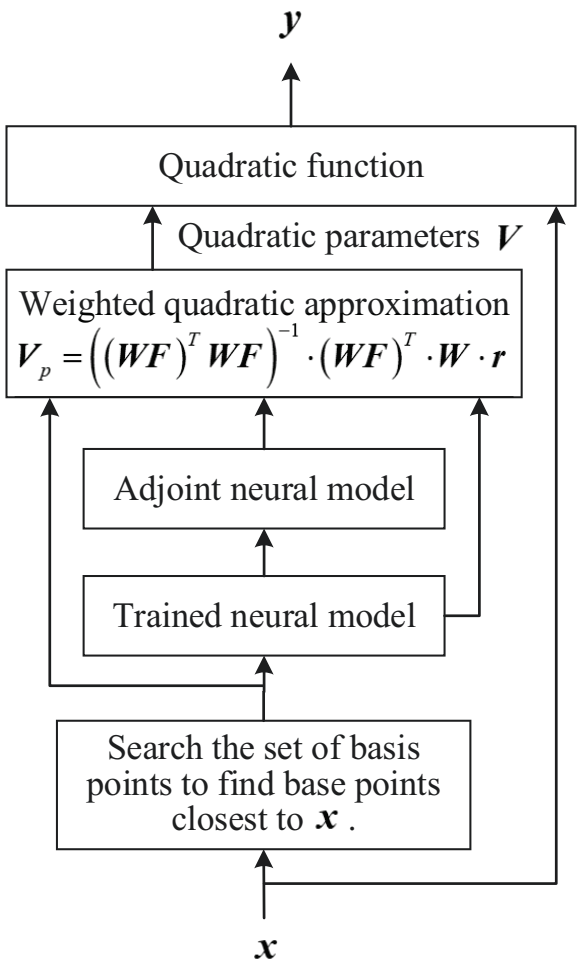


Figure 2.11: Flow-chart of the model-level extrapolation [11]. This process is done during the use of trained neural models.

2.5.2 Neuron-Level Extrapolation Method

Recently, a simple and effective neuron-level extrapolation method [12] is presented to address the problem in case when the training boundaries are irregular. It detects

the necessity of extrapolation inside each hidden neuron and performs extrapolation using a modified neuron activation function. It simplifies the problem into a combination of several one-dimensional 1st order extrapolations along the neuron switch directions of a set of hidden neurons. This method is easy to implement.

Let $\sigma(\gamma_i)$ represent the neuron activation function, e.g., sigmoid function [1], for the ith hidden neuron. Let γ_i be the one-dimensional input variable for the switch function of the ith hidden neuron. The neuron-level extrapolation method first evaluates the switch variable γ_i from all training samples and obtain the minimum and maximum of γ_i, i.e., $\gamma_{i\min}$ and $\gamma_{i\max}$. The modified neuron activation function with extrapolation is

$$z_i = \begin{cases} \sigma(\gamma_{i\min}) + \sigma(\gamma_{i\min}) \cdot (1 - \sigma(\gamma_{i\min})) \cdot (\gamma_i - \gamma_{i\min}), & \text{if } \gamma_i < \gamma_{i\min} \\ \sigma(\gamma_i), & \text{if } \gamma_{i\min} \leqslant \gamma_i \leqslant \gamma_{i\max} \\ \sigma(\gamma_{i\max}) + \sigma(\gamma_{i\max}) \cdot (1 - \sigma(\gamma_{i\max})) \cdot (\gamma_i - \gamma_{i\max}), & \text{if } \gamma_i > \gamma_{i\max} \end{cases} \tag{2.6}$$

where the term $\sigma \cdot (1 - \sigma)$ is the derivative of the function, i.e., the sigmoid function. Fig. 2.12 illustrates the neuron-level extrapolation for the ith hidden neuron of the neural network. If the neural network has only one input, the effective range of the extrapolation is equal to that of the 1st order Taylor expansion at the boundary of the training range. If the neural network has multiple inputs, the neuron-level extrapolation technique simplifies the problem into a combination of several one-dimensional 1st order extrapolations along the neuron switch directions of a set fo hidden neurons.

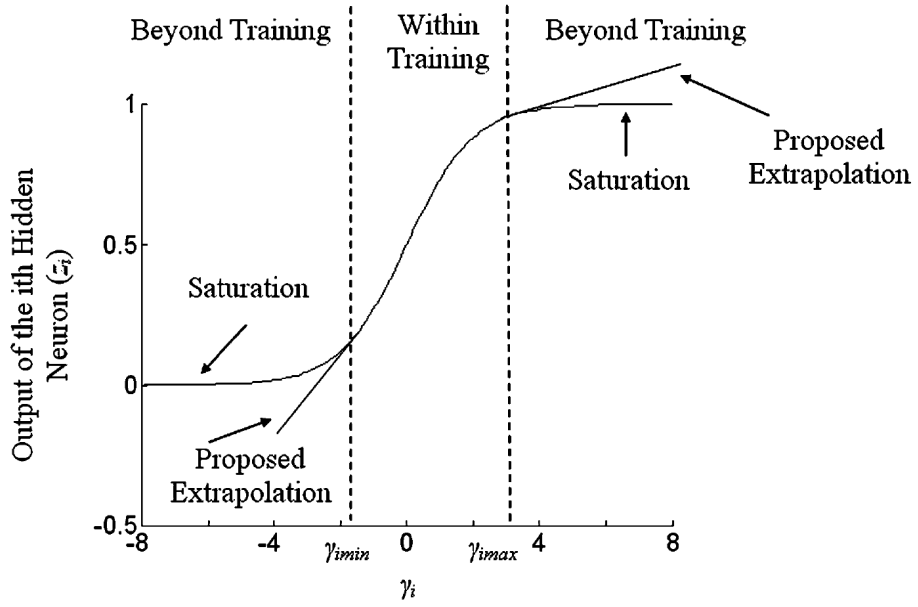


Figure 2.12: Illustration of the neuron-level extrapolation technique [12] for a hidden neuron. The saturation region outside training range is replaced by the linear approximation extended from the γ-boundary of the hidden neuron.

The existing neuron-level extrapolation technique [12] works well in low dimensions or with few hidden neurons. With increased dimension or increased number of hidden neurons, the hidden neuron switch directions (along which extrapolations are made) are not always aligned well with the normal directions of the training boundaries in the input space, affecting the quality of extrapolation.

2.6 Conclusion

In this chapter, existing neural network based methods for microwave modeling and design are reviewed. The basic concepts of neural network methods, i.e., neural

network structure and neural network model development, have been discussed. Automated neural model generation technique and knowledge-based neural network using space mapping concept, which are relevant to this thesis work, have been reviewed. The knowledge-based neural network has been discussed as an advanced type of neural networks with prior knowledge as part of internal neural network structure. Space mapping methods such as input space mapping, frequency space mapping and output space mapping combine space mapping technology and neural network for microwave modeling to improve model accuracy and modeling efficiency. Further, extrapolation problem of neural network model and recent extrapolation methods on neural network model such as the model-level extrapolation method and neuron-level extrapolation method are also reviewed. In the next chapter, an advanced automated knowledge-based neural network modeling algorithm with a new adaptive sampling method for microwave applications is proposed.

Chapter 3

Automated Neural Modeling Algorithm with A New Adaptive Sampling Technique

This chapter describes an automated knowledge-based neural network modeling algorithm with a new adaptive sampling technique. In this chapter, automated model generation (AMG) method is extended from generating pure neural network models to generating knowledge-based neural network models for the first time. Knowledge-based models have been demonstrated in the existing literature to use less data over pure neural network models while maintaining good accuracy. The proposed method automates data generation, determination of data distribution, model structure adaptation, and model training in a systematic framework. A new adaptive sampling technique incorporating efficient interpolation approaches is proposed to make the automated modeling process much faster. The proposed method can further reduce the number of training data, shorten the model development time over existing automated model generation methods and existing knowledge-based

modeling methods, and ensure the accuracy of the final model at the same time. The algorithm is demonstrated through a microwave modeling example.

3.1 Introduction

In neural network modeling, more training data usually leads to better model accuracy. Two types of advanced approaches, i.e., automated model generation (AMG) [4] and knowledge-based neural network modeling [8], have been presented in the literature where good model accuracy can be achieved with less training data. In [4], multilayer perceptron (MLP) modeling technique with adaptive sampling algorithm was introduced to turn the conventional manual modeling process into the AMG process. However, in the existing literature, the knowledge-based model development is typically a manual process. The number of training data, the distribution of data in the input space and the size of the NN are generally unknown and usually determined by experience.

In this chapter, an advanced algorithm to automate the process of developing knowledge-based models with a new sampling technique for microwave applications is proposed. It integrates all the subtasks in generating a knowledge-based model like initial optimization, unit mapping, data generation, model structure adaptation, training and testing into a unified framework. The proposed algorithm applies adaptive sampling to knowledge-based modeling, and an efficient interpolation approach is added into the AMG process, avoiding training intermediate models, further speeding up model development. Therefore, the number of training data and their distribution in the input space are automatically determined by the algorithm.

It uses less training data, shorter model development time than existing automated model generation techniques and existing knowledge-based manual training methods.

3.2 Proposed Automated Neural Modeling Algorithm with A New Adaptive Sampling Technique

3.2.1 Proposed Model Structure

While the model structure in existing AMG algorithms [4], [14] is a pure neural network, i.e., the 3-layer MLP, the knowledge-based model structure used in our proposed algorithm is formulated using space mapping concept [3], [32] shown in Fig. 3.1.

Let $\boldsymbol{x}$ represent an input vector containing physical parameters of a microwave device, and $\boldsymbol{y}$ represent a vector of the knowledge-based model outputs. The input mapping function is realized by the MLP in the knowledge-based model. We define ω to represent the frequency input, and $\boldsymbol{x}_E$ to represent the outputs of the MLP. The inputs of the empirical model are

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{x}_E^T & \omega \end{bmatrix}^T = \begin{bmatrix} \boldsymbol{f}_{\text{map1}}\left(\boldsymbol{x}, \boldsymbol{w}_{\text{map1}}\right)^T & \omega \end{bmatrix}^T \tag{3.1}$$

In the form of input mapping, the knowledge-based model can be described by the

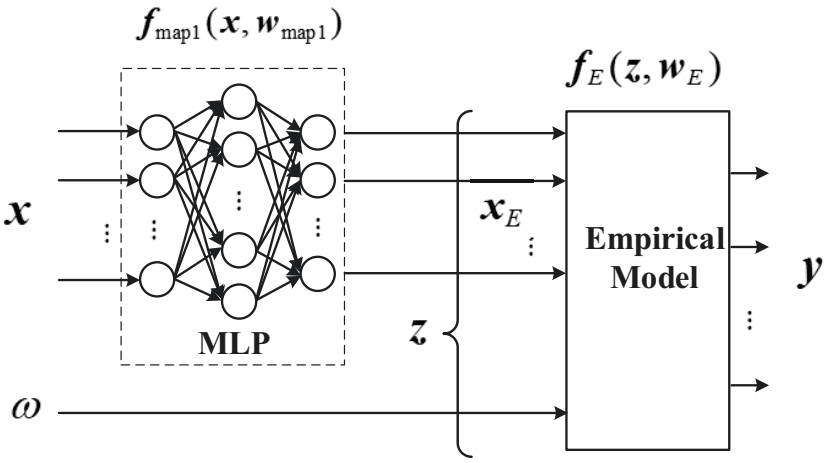


Figure 3.1: The knowledge-based model used in the proposed algorithm. It consists of an MLP $\boldsymbol{f}_{\text{map1}}(\boldsymbol{x}, \boldsymbol{w}_{\text{map1}})$ as the input mapping and an empirical model $\boldsymbol{f}_E(\boldsymbol{z}, \boldsymbol{w}_E)$.

relationship between inputs and outputs as

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{f}\left(\begin{bmatrix}\boldsymbol{x}^T & \omega\end{bmatrix}^T, \boldsymbol{w}\right) \\ &= \boldsymbol{f}_E(\boldsymbol{z}, \boldsymbol{w}_E) \\ &= \boldsymbol{f}_E\left(\begin{bmatrix}\boldsymbol{f}_{\text{map1}}(\boldsymbol{x}, \boldsymbol{w}_{\text{map1}})^T & \omega\end{bmatrix}^T, \boldsymbol{w}_E\right) \end{aligned} \tag{3.2}$$

where $\boldsymbol{f}$, $\boldsymbol{f}_E$ and $\boldsymbol{f}_{\text{map1}}$ represent the entire knowledge-based model, the empirical model and the input space mapping MLP, respectively. $\boldsymbol{w}_E$ is a vector of parameters in the empirical model, $\boldsymbol{w}_{\text{map1}}$ is a vector of weights in the input space mapping MLP, and $\boldsymbol{w}$ is a vector combining $\boldsymbol{w}_E$ and $\boldsymbol{w}_{\text{map1}}$. The MLP modifies the inputs of the empirical model, so that the outputs of the knowledge-based model match the

training data. Compared to pure neural network models, knowledge-based models can achieve good accuracy with less training data [8].

3.2.2 Proposed Adaptive Sampling Process

For the proposed algorithm, we use four sets of data, i.e., unit-mapping set M, training set L, testing set V and interpolation set I. Unlike existing AMG algorithms which use only training and testing data, unit-mapping data is used in our proposed algorithm to initialize the input space mapping MLP and provide good starting point before training the model. Interpolation data is used to create local models in subregions of the model input space to assess the adequacy of training data during training process.

The proposed algorithm is carried out in a stage-wise manner and the algorithm regards the original input space as one region at first. If the test error in the region is large, the region would be divided and new data are generated. In each stage, the algorithm compares the test error in each subregion. The subregion with the maximum test error is chosen as the worst region. Then the worst region would be further divided into 2^n new subregions in the next stage. In order to avoid training intermediate neural networks in each stage, we propose to evaluate the test error of various subregions using interpolation approaches during each stage of AMG [14]. Different subregions have different interpolations. In this way, we take advantage of availability of training data to produce localized interpolation function, avoiding the training of neural network during intermediate stages of AMG, and speeding up model development. Neural network training is performed only during the last few

stages of AMG.

3.2.3 Proposed Automated Neural Modeling Method with A New Adaptive Sampling Technique

The proposed algorithm is carried out in a stage-wise manner. In the first stage, the existing empirical model is optimized, so that the model could approximate the training data as much as possible. The solution of this first optimization is

$$\boldsymbol{w}_E^* = \arg\min_{\boldsymbol{w}_E} \sum_{\boldsymbol{z}^{(a)} \in L} \left(\frac{1}{2} \left\| \boldsymbol{f}_E \left(\boldsymbol{z}^{(a)}, \boldsymbol{w}_E \right) - \boldsymbol{d}^{(a)} \right\|^2 \right) \tag{3.3}$$

where $\boldsymbol{z}^{(a)}$ is the ath sample of model inputs in the initial training data set L, and $\boldsymbol{d}^{(a)}$ is the corresponding sample of model outputs in the training data.

The algorithm then performs the second optimization, i.e., the unit mapping, formulated as the training of $\boldsymbol{f}_{\text{map1}} \left(\boldsymbol{x}, \boldsymbol{w}_{\text{map1}} \right)$ with $\boldsymbol{w}_{\text{map1}}$ as variables, such that for various samples of $\boldsymbol{x}$

$$\boldsymbol{x} = \boldsymbol{f}_{\text{map1}} \left(\boldsymbol{x}, \boldsymbol{w} \right) \tag{3.4}$$

In this way, the entire knowledge-based model equals the empirical model at the beginning of the next training process. In subsequent stages, the algorithm performs adaptive data generation and MLP structure adaptation. The training error E_{train}^k during the kth stage of the automated knowledge-based model development process is

$$E_{\text{train}}^k \left(\boldsymbol{w}_{\text{map1}} \right) = \sum_{\left[\boldsymbol{x}^{(a)^T} \ \omega^{(a)} \right]^T \in L} \left(\frac{1}{2} \left\| \boldsymbol{f}_E \left(\left[\boldsymbol{f}_{\text{map1}} \left(\boldsymbol{x}^{(a)^T}, \boldsymbol{w}_{\text{map1}} \right)^T \quad \omega^{(a)} \right]^T, \boldsymbol{w}_E^* \right) - \boldsymbol{d}^{(a)} \right\|^2 \right) \tag{3.5}$$

where $\left[\boldsymbol{x}^{(a)^T}\ \omega^{(a)}\right]^T$ is the ath sample of model inputs in the training data set L. The tst error E^k_{test} is defined similarly to (3.5) except that the set L is replaced by V. The interpolation error E^k_{In} is computed similarly as (3.5) except that L is replaced by I and $\boldsymbol{f}_E$ is replaced by interpolation formulas [14].

Let E_d represent user-desired model accuracy (test error). In the first few stages, we use interpolation error E^k_{In} to replace training error E^k_{train}, to avoid training the intermediate knowledge-based model and shorten the training time. If $E^k_{In} > E_d$, the knowledge-based model is detected as over-learning, and more data will be generated in the next stage. Interpolation error E^k_{In} is used to estimate the location of the most nonlinear subregion (where E^k_{In} is high) in the input space. New data from this subregion will be generated, and data sets L and V are updated. The training process proceeds stage by stage. In each stage, the remaining most nonlinear subregion is identified and corresponding training and testing data are generated. The interpolation error reduces stage by stage and data generation stops when E^k_{In} in all subregions are below required error criteria.

By creating training samples only where data is mostly needed, the proposed algorithm uses data more effectively than manual knowledge-based model training where grid data is typically used. In other words, the proposed method can sample data densely in highly nonlinear subregions, and sparsely in smooth subregions of the modeling input space. In this way, the proposed algorithm can achieve good accuracy with less training data than manual knowledge-based model training. On the other hand, with the incorporation of knowledge, our proposed knowledge-based automated model generation algorithm uses less training data than existing

automated model generation methods.

Next, the algorithm enters into neural network training and the model size adjustment stage. It performs the third type of optimization, i.e., training of the knowledge-based model with input space mapping MLP weights $\boldsymbol{w}_{map1}$ as variables. If $E^k_{\text{train}} > E_d$, the knowledge-based model is detected as under-learning, and more hidden neurons will be added into the input space mapping MLP structure to give the MLP more freedom to achieve better mapping of the empirical model towards training data. Otherwise, if $E^k_{\text{test}} < E_d$ and $E^k_{\text{train}} < E_d$, the knowledge-based model is detected as good-learning, and the number of hidden neurons in the MLP structure will be reduced to get a more compact MLP structure. The goal of the algorithm is to automatically carry out stage-wise knowledge-based model development process until $E^k_{\text{test}} < E_d$ is achieved with fewest hidden neurons in MLP structure.

In the last stage, the algorithm performs the fourth type of optimization. The entire knowledge-based model including both the empirical part and the MLP part is optimized to refine the final modeling result. The optimization is formulated as

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{\left[\boldsymbol{x}^{(a)T} \;\; \omega^{(a)}\right]^T \in L} \left(\frac{1}{2} \left\| \boldsymbol{f}\left(\left[\boldsymbol{x}^{(a)T} \;\; \omega^{(a)}\right]^T, \boldsymbol{w} \right) - \boldsymbol{d}^{(a)} \right\|^2 \right) \tag{3.6}$$

A flowchart of the proposed algorithm for automated knowledge-based model development is shown in Fig. 3.2.

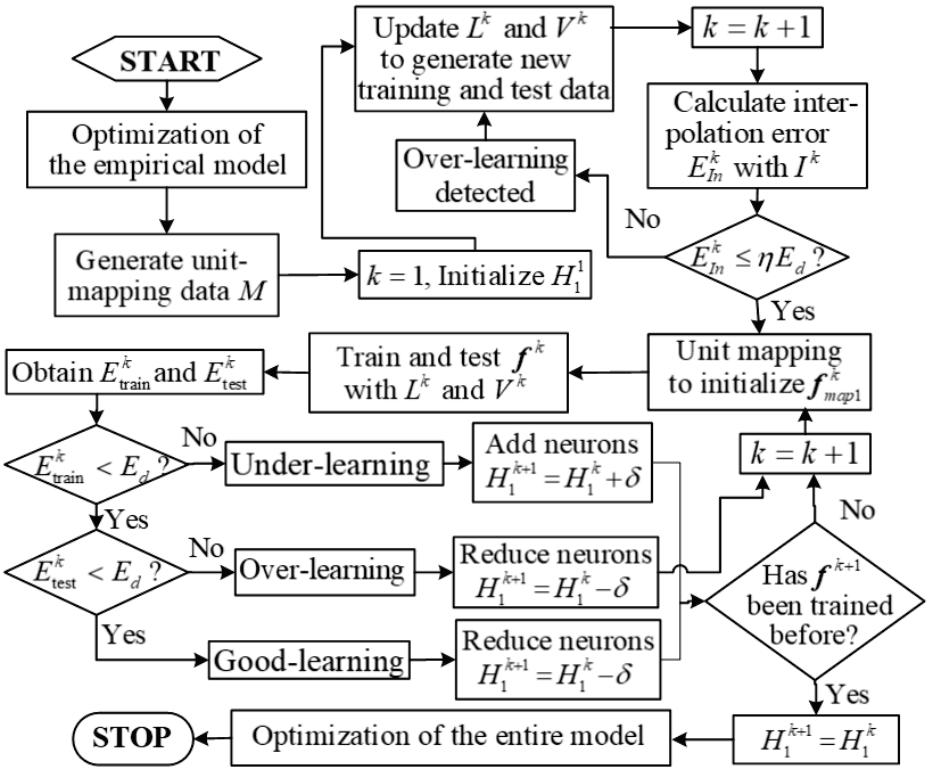


Figure 3.2: Flowchart of the proposed automated knowledge-based model generation algorithm. H_1 represents the number of hidden neurons.

3.2.4 Discussion

If the quality of the empirical model is high, the final knowledge-based model can be achieved with less data and fewer hidden neurons. If the quality of the empirical model is low, more data and more hidden neurons are needed to get successful results. The knowledge-based model structure in this chapter is based on input-

mapping concept. In next chapter, we expand the automated model generation approach to other types of knowledge-based model structures, such as output mapping and frequency mapping.

3.3 Application Example - Automated Neural Modeling for a Bandpass High-Temperature Superconducting (HTS) Microstrip Filter

In this example, we develop a parametric knowledge-based model of a high-temperature superconducting (HTS) quarter-wave parallel coupled-line microstrip filter [32]. The model inputs are $\boldsymbol{x} = [L_1, L_2, L_3,\ \ S_1, S_2, S_3]^T$, where L_1, L_2 and L_3 are the lengths of the parallel coupled-line sections and S_1, S_2 and S_3 are the gaps between the sections.

The user-desired input space is bounded by $L_1 = 172.3$ to 182.9 mil, $L_2 = 186.6$ to 198.1 mil, $L_3 = 173.1$ to 183.8 mil, $S_1 = 19.5$ to 23.8 mil, $S_2 = 78.4$ to 95.8 mil and $S_3 = 74.7$ to 91.2 mil. The frequency range is from 3.8 to 4.2 GHz. The model outputs are $\boldsymbol{y} = [|S_{11}|, |S_{21}|]^T$.

Training data is generated using *CST Microwave Studio*. Compared to a similar example in [14], a denser mesh for EM simulation is used here. Our program automatically drives *CST* to read input samples, execute EM simulation and return EM simulation results. When a subregion is defined, the proposed algorithm can automatically drive *CST* to generate new training data wherever needed. The empirical model is an equivalent circuit consisting of six coupled microstrip lines [32]. We use $C{+}{+}$ to program and incorporate the equivalent circuit of the bandpass

filter in the *NeuroModelerPlus*. The equivalent circuit alone cannot match EM data to achieve required accuracy especially if geometrical parameters vary. Therefore, the equivalent circuit is combined with MLP to form the entire knowledge-based model.

For comparison purpose, we perform MLP manual training, knowledge-based model manual training, existing MLP automated model generation algorithm and proposed knowledge-based automated model generation algorithm. These four models are all further tested with the same set of test samples. The results are listed in Table 3.1 and shown in Fig. 3.3. In manual methods, the models are manually trained several times with different number of hidden neurons, and the model meeting the desired accuracy with fewest hidden neurons is selected. In manual methods, the CPU time includes data generation time, training time and testing time. The proposed algorithm costs 27.6 h to achieve a knowledge-based model with 0.83% error.

The results show that because of the addition of equivalent circuit, knowledge-based modeling techniques can achieve better accuracy than MLP modeling with same amount of training data. Table I also illustrates that automated model generation algorithms use less data and takes less time than manual training because of the adaptive sampling process.

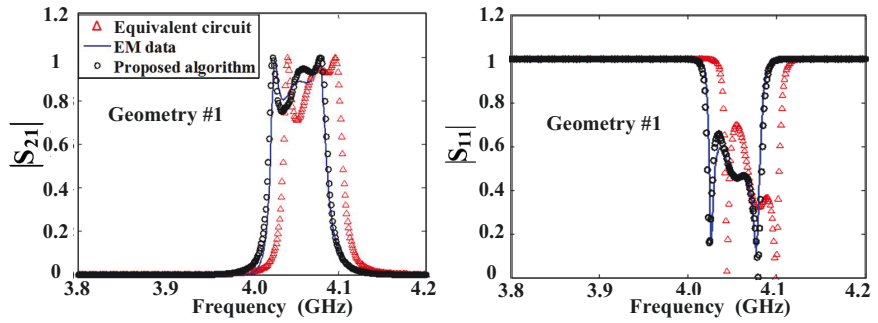
3.4 Conclusion

In this chapter, an automated neural modeling algorithm with a new adaptive sampling technique is proposed. Combining the advantages of both knowledge-based

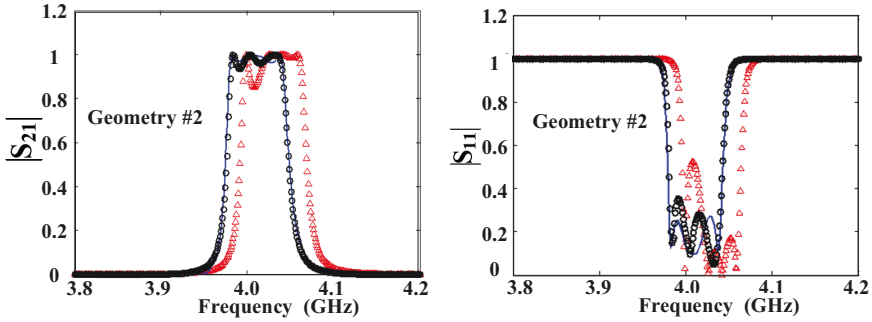
Table 3.1: Comparisons Between Proposed Knowledge-Based Model Automated Generation Algorithm (AMG) and Existing Algorithms for the Bandpass Filter Example

Model Development Method	No. of Hidden Neurons	No. of Training Data	Test Error	CPU Time
Equivalent Circuit	/	/	7.66%	/
MLP Manual	33	4096	1.53%	147h
Knowledge-Based Manual	13	4096	0.90%	138h
Knowledge-Based Manual	13	2304	1.15%	90h
MLP AMG	33	1394	1.22%	55.2h
Knowledge-Based AMG	13	729	0.83%	27.6h

techniques and automated model generation methods, our proposed knowledge-based automated model generation algorithm provides further reduction of data over existing automated model generation method and knowledge-based manual training. It uses least amount of training data and shortest time to obtain the most accurate and most compact model among all the four neural modeling techniques



(a)



(b)

Figure 3.3: The modeling result of the bandpass filter at two different sets of geometrical values (a) $\boldsymbol{x} = [175.0, 189.5, 181.1, 20.5, 91.5, 78.8]^T$ and (b) $\boldsymbol{x} = [180.3, 195.3, 175.7, 22.7, 82.8, 82.9]^T$. The behaviors of the equivalent circuit are added as references.

Chapter 4

A Unified Automated Parametric Modeling Algorithm Using Knowledge-Based Neural Network and l_1 Optimization

Knowledge-based neural network modeling techniques using space-mapping concept have been demonstrated in the previous chapter and other existing literature as efficient methods to overcome the accuracy limitations of empirical/equivalent circuit models when matching new electromagnetic (EM) data. For different modeling problems, the mapping structures can be different. In this chapter, we present a unified automated model generation algorithm which uses l_1 optimization to automatically determine the type and the topology of the mapping structure in a knowledge-based neural network model. By encompassing various types of mappings of knowledge-based neural network model in existing literature, we present a new unified model structure and derive new sensitivity formulas for the training of the unified model. The proposed l_1 formulation of modeling can force some weights

of the mapping neural networks to zeros while leaving other weights as non-zeros. We utilize this feature to allow l_1 optimization to automatically determine which mapping is necessary and which mapping is unnecessary. Using the proposed l_1 optimization method, the mapping structure can be determined to address different needs of different modeling problems. The structure of the final knowledge-based model can be flexible combinations of some or all of linear mapping, nonlinear mapping, input mapping, frequency mapping and output mapping. In this way, the presented algorithm is more systematic and can further speed up the knowledge-based modeling process than existing knowledge-based modeling algorithms. The presented method is illustrated by three microwave filter modeling examples.

4.1 Introduction

ANNs learn EM data through the training process and the trained neural networks are then used as fast and accurate models for efficient high-level circuit and system design. In microwave modeling area, knowledge-based neural network modeling approaches using space-mapping concept [8], [20], [32] have been described in the literature for obtaining a better model with limited data. The idea of knowledge-based model is to exploit existing knowledge in the form of empirical or equivalent circuit models together with neural networks to develop a faster and more accurate model. For microwave design, there exists many empirical or equivalent circuit models which are computationally efficient and widely used in practical design. However, such models are often valid only in a limited parameter range, beyond which the model predictions become inaccurate. As with the space mapping concept, the em-

pirical/equivalent circuit models are considered as "coarse model" [87], [98] and the EM data is considered as "fine data". The space mapping technique is developed to use linear mappings to establish a mathematical link between the coarse model and the fine data [87], [98], and to obtain a faster and more accurate model. However, when the modeling range becomes large, linear mappings only are not enough. The neural networks are used to provide nonlinear computation approach to bridge the gap between the empirical/equivalent circuit model and the new EM simulation data [32]. This is achieved with space mapping concept by using neural networks to represent the nonlinear mappings between the empirical/equivalent circuit model and the EM data. Extrapolation capability is also enhanced because of the embedded knowledge in the model [8]. For simplicity and convenience of the ongoing descriptions, we will use the term "empirical model" to imply the empirical and/or equivalent circuit models in the subsequent part of this chapter.

By taking advantage of the vast set of empirical models already available, space mapping-based neural network models decrease the number of EM simulations for training, improve model accuracy and generalization ability, and reduce the complexity of the ANN topology with respect to the classical neural network modeling approach [32], [85]. A number of papers cover different kinds of space mapping, including input space mapping [85], [86], frequency space mapping [89] and output space mapping [90]. Input space mapping is used to modify the values of the geometrical design variables to a different set of values to be supplied to the empirical model, so that the modified empirical model response can match the EM simulation data [87], [88]. For those cases where the shapes of the EM simulation and the em-

pirical model response are nearly identical, but shifted in frequency, the frequency space map-ping is used to align both responses [89]. Output space mapping is used to enhance the robustness of the modeling process in case other space mappings cannot provide sufficient matching with the EM data [90], [91]. Any of these three mappings can be either linear or nonlinear. For simple modeling problems within a small geometrical parameter range, linear mappings [98], [99] are used to obtain a good match be-tween the model and the training data. For complicated modeling problems within a large geometrical parameter range, nonlinear mappings [100] are necessary in order to obtain an accurate model.

Several approaches for the structure selection of the knowledge-based model are described in the existing literature [13], [14], [99], [101]. In [99], both input and output space mappings are used in the knowledge-based model for microwave device optimization. However, all the mappings in [99] are linear mappings, and the decision about whether to use input map-ping or output mapping is made manually with the designer's knowledge of the problem and engineering experience. The method in [101] uses genetic algorithm to find suitable combinations of the mapping structure and the empirical model, but it cannot distinguish between linear mapping and nonlinear mapping.

In [4], automated model generation method is described to automate the structure selection process of the pure neural network modeling. In previous chapter, automated model generation method is expanded from generating pure neural network models to generating knowledge-based models. It integrates all the subtasks in generating a knowledge-based model into one framework and further reduces the

number of training data required. However, the structure of the knowledge-based model in previous chapter is fixed at a pre-determined structure and the automated algorithm is more focused on the automation of data sampling process. The model structure in previous chapter is a combination of an empirical model and a nonlinear input mapping which is realized by a three-layer MLP neural network. However, the selection of different mapping structures is not addressed in previous chapter. For some complicated EM modeling problems, input mapping only may not be good enough to achieve an accurate model. On the other hand, for some simple modeling problems, a nonlinear input mapping is redundant and only one simple linear mapping may be good enough to meet the accuracy requirement.

Different mapping structures should be needed for different modeling problems. The mapping structure depends on many factors, such as the complexity of the specific modeling problem, the quality of the empirical model, and the modeling range of geometrical parameters. For example, for a given set of EM data, different empirical models may need different mapping structures. For another example, if an empirical model is to be mapped to different sets of EM data with different ranges of geometrical parameters, the mapping structures in the final model must be different. Since the mapping structure is problem-dependent, the development of the automated model structure adaptation algorithm is very important.

Most recently, preliminary work for a relatively flexible automated model structure adaptation method for knowledge-based model development is described in [16]. It takes both input and output mapping, linear and nonlinear mapping into consideration during the modeling process. The final knowledge-based model can be any

combination of the empirical model and the mapping neural networks. However, when determining the mapping structure, the algorithm is based on a brute force sequential trial and error mechanism, first trying input mapping and then output mapping, first trying linear mapping and then nonlinear mapping. It compares various combinations of mappings and finds a suitable knowledge-based neural network model. The process involves many trial and error computation steps and is usually time-consuming.

In this chapter, we describe a more elegant and unified automated model structure adaptation algorithm for knowledge-based parametric modeling. The presented technique is a substantial advance over [16]. We present a new formulation using l_1 optimization to automatically determine all the mappings in the knowledge-based model. We present an extended and unified model structure to encompass various types of mappings. New sensitivity formulas for the training of the unified model are also presented and derived in this chapter. The use of the l_1 optimization, based on its theoretical properties [102], [103], is a critical part of the presented training algorithm and optimization process. The l_1 optimization has the distinctive property for feature selection within the training process. At the end of l_1 optimization, some weights in the mapping neural networks are zeros while others remain non-zeros. Zero weights mean that the corresponding parts of the mapping can be ignored and deleted. Using this property, we formulate l_1 optimization solutions to indicate whether a mapping is linear or nonlinear, and whether a mapping should be input mapping, frequency mapping or output mapping. Compared to traditional knowledge-based models with fixed mapping structure, the presented method can

automatically adjust the mapping structure to achieve an accurate model with the most suitable and compact structure. Compared to existing literature on model structure selection method, our presented method is a more systematic technique and can further speed up the process of developing a knowledge-based neural network model.

4.2 Proposed Unified Knowledge-Based Model Structure

Let $\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$ represent a vector of the inputs of the presented knowledge-based model. We define n as the number of the model inputs. $\boldsymbol{x}$ contains the physical geometrical parameters of a microwave device, such as the length and the width of an EM structure. Let $\boldsymbol{y} = [y_1, y_2, \cdots, y_m]^T$ represent a vector of the outputs of the knowledge-based neural network model, such as S-parameters. We define m as the number of the model outputs. Empirical models often exist to represent the relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$. However, the accuracy of the empirical model is usually limited, especially when the values of the EM geometrical parameters change. The mapping method using neural networks is utilized to address the situation when an existing empirical model cannot fit the new EM data well. The mapping structures are affected by many factors and the determination of the mapping structures is not straight forward. In [16], the knowledge-based model structure combines the empirical model with one input mapping, one frequency mapping and one output mapping. The three mappings can be either a two-layer MLP or a three-layer MLP, which represents linear mapping or nonlinear mapping respectively. In this chapter,

we present a new unified knowledge-based model structure which is expanded from [16]. This is done by expanding the mapping neural network from MLP to include an additional direct connection between the input layer and the output layer of the neural network. This structure is a composite two-layer and three-layer MLP, which allows the neural network to have explicit linear/nonlinear mapping. Thus, the presented unified model has three mixed linear and nonlinear mapping neural networks which are the input mapping, the frequency mapping and the output mapping. The structure of the presented unified knowledge-based model is shown in Fig. 4.1.

We define the various symbols needed to represent various aspects of the mapping structure, modeling and the training. Let $\boldsymbol{f}_{\text{map1}}$, f_{map2} and $\boldsymbol{f}_{\text{map3}}$ represent the mapping functions for input mapping, frequency mapping and output mapping, respectively. Each mapping function is a neural network. Let H_1, H_2 and H_3 represent the number of hidden neurons in $\boldsymbol{f}_{\text{map1}}$, f_{map2} and $\boldsymbol{f}_{\text{map3}}$ respectively. We define $\boldsymbol{u}_{\text{map1}}$, $\boldsymbol{u}_{\text{map2}}$ and $\boldsymbol{u}_{\text{map3}}$ as vectors of the weights for the presented direct connections between the input neurons and output neurons in the input mapping $\boldsymbol{f}_{\text{map1}}$, frequency mapping f_{map2} and output mapping $\boldsymbol{f}_{\text{map3}}$ respectively. Let $\boldsymbol{u} = [\boldsymbol{u}_{\text{map1}}^T \ \boldsymbol{u}_{\text{map2}}^T \ \boldsymbol{u}_{\text{map3}}^T]^T$ be defined as a vector containing all weights for the presented direct connections between the input neurons and output neurons in all mapping neural networks. We define $\boldsymbol{t}_{\text{map1}}$, $\boldsymbol{t}_{\text{map2}}$ and $\boldsymbol{t}_{\text{map3}}$ as vectors of the weights between the input neurons and hidden neurons in $\boldsymbol{f}_{\text{map1}}$, f_{map2} and $\boldsymbol{f}_{\text{map3}}$ respectively. We also define $\boldsymbol{v}_{\text{map1}}$, $\boldsymbol{v}_{\text{map2}}$ and $\boldsymbol{v}_{\text{map3}}$ as vectors of the weights between the hidden neurons and output neurons in $\boldsymbol{f}_{\text{map1}}$, f_{map2} and $\boldsymbol{f}_{\text{map3}}$ respectively. Let $\boldsymbol{v} = [\boldsymbol{v}_{\text{map1}}^T \ \boldsymbol{v}_{\text{map2}}^T \ \boldsymbol{v}_{\text{map3}}^T]^T$ be defined as a vector containing all weights between the

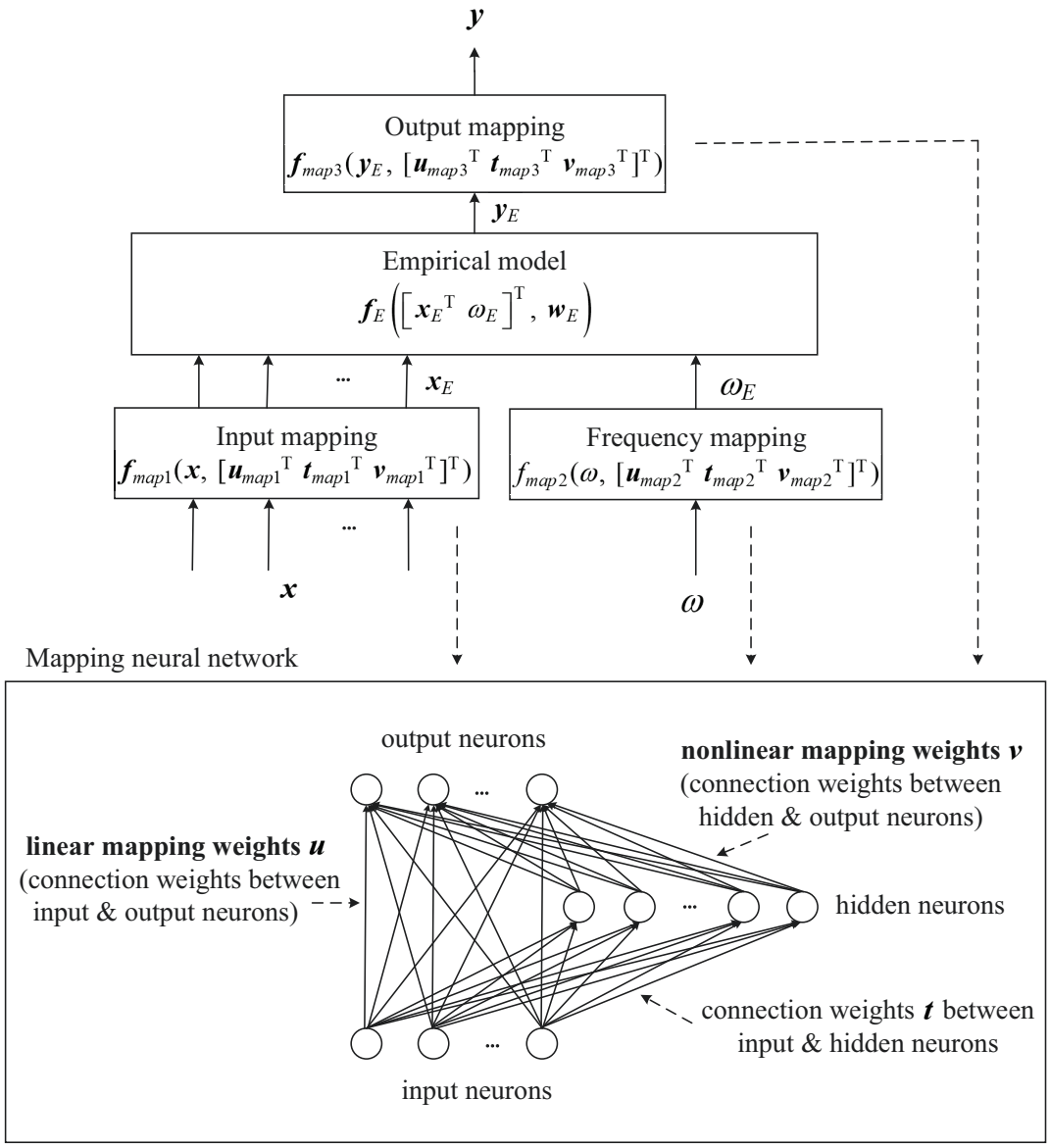


Figure 4.1: Proposed unified knowledge-based model structure. It combines the empirical model with three mapping neural networks. Each mapping neural network contains linear mapping and nonlinear mapping.

hidden neurons and output neurons in all mapping neural networks. Therefore, in our presented unified model structure, the weights in $\boldsymbol{u}$ represent the linear mapping weights and the weights in $\boldsymbol{v}$ represent the nonlinear mapping weights.

We define $\boldsymbol{x}_E = [x_{E1}, x_{E2}, \cdots, x_{En}]^T$, ω_E and $\boldsymbol{y}_E = [y_{E1}, y_{E2}, \cdots, y_{En}]^T$ as the geometrical inputs, the frequency input and the outputs of the empirical model, respectively. The empirical model is represented as

$$\boldsymbol{y}_E = \boldsymbol{f}_E\left([\boldsymbol{x}_E^T\ \omega_E]^T, \boldsymbol{w}_E\right) \tag{4.1}$$

where $\boldsymbol{w}_E$ is a vector of parameters in the empirical model. Various empirical models have been developed in the past. They are computationally efficient but the accuracy is limited when matching new EM data. In other words, if we directly supply the geometrical parameter values $\boldsymbol{x}$ to the empirical model, the outputs of the empirical model may not sufficiently match the outputs $\boldsymbol{y}$ of the EM data, i.e.,

$$\boldsymbol{y} \neq \boldsymbol{f}_E\left([\boldsymbol{x}^T\ \omega]^T, \boldsymbol{w}_E\right) \tag{4.2}$$

The proposed method addresses the problem when the existing empirical model cannot fit the new EM data well. We use mappings to alter the relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$, therefore altering the model. The input mapping is used to modify the values of the geometrical parameters to a new set of values to be supplied to the empirical model. The purpose to modify (map) the values of the geometrical parameters is to make the subsequent response of the empirical model better match the EM simulation data. $\boldsymbol{x}$ represent the original values of the geometrical parameters and $\boldsymbol{x}_E$ represent the modified values of the geometrical parameters to be supplied

to the empirical model. Because the relationship between $\boldsymbol{x}$ and $\boldsymbol{x}_E$ is unknown, we use a neural network to represent this relationship, i.e., the input mapping function. In the presented algorithm, the input mapping neural network, which has N input neurons and N output neurons, is defined as

$$\boldsymbol{x}_E = \boldsymbol{f}_{\text{map1}}(\boldsymbol{x}, \boldsymbol{w}_{\text{map1}}) \tag{4.3}$$

where $\boldsymbol{w}_{\text{map1}} = [\boldsymbol{u}_{\text{map1}}^T \ \boldsymbol{t}_{\text{map1}}^T \ \boldsymbol{v}_{\text{map1}}^T]^T$ represents the weights in the input mapping neural network $\boldsymbol{f}_{\text{map1}}$ The presented algorithm determines the structure of the input mapping, whether linear or nonlinear, or no mapping. Here we formulate the internal structure of the input mapping neural network to include linear and nonlinear mappings as follows. The ith output of the input mapping is computed as

$$f_{\text{map1},i} = \sum_{j=0}^{n}(u_{\text{map1},ij} \cdot x_j) + \sum_{k=0}^{H_1}(v_{\text{map1},ik} \cdot z_{\text{map1},k}), \quad i = 1, 2, \cdots, n \tag{4.4a}$$

$$z_{\text{map1},k} = \begin{cases} \sigma\left(\sum_{j=0}^{n} t_{\text{map1},kj} \cdot x_j\right), & k \neq 0 \\ 1, & k = 0 \end{cases} \tag{4.4b}$$

$$x_0 = 1 \tag{4.4c}$$

where $z_{\text{map1},k}$ is defined as the output of the kth hidden neuron of the input mapping neural network. To accommodate the convenient representation of the bias parameters in neural networks, we assume the fictitious neurons in the input mapping as x_0 and $z_{\text{map1},0}$ [1]. $\sigma(\cdot)$ is the sigmoid function. $u_{\text{map1},ij}$, $v_{\text{map1},ik}$ and $t_{\text{map1},kj}$ represent the connection weight between the jth input neuron and the ith output neuron, the weight between the kth hidden neuron and the ith output neuron, and

the weight between the jth input neuron and the kth hidden neuron of the input mapping neural network, respectively. $u_{\text{map1},ij}$ represents the linear mapping weight and $v_{\text{map1},ik}$ represents the nonlinear mapping weight in the input mapping neural network. In this way, the presented input mapping can encompass linear mapping, nonlinear mapping or no mapping by one neural network.

The frequency mapping is usually used for those cases where a shift in frequency exists between the empirical model response and the EM data. The presented algorithm determines the structure of the frequency mapping, whether linear or nonlinear, or no mapping. The internal structure of the frequency mapping neural network, which has one input neuron and one output neuron, is defined as

$$\begin{aligned}\omega_E &= f_{\text{map2}}(\omega,\ \boldsymbol{w}_{\text{map2}}) \\ &= f_{\text{map2}}(\omega,\ [\boldsymbol{u}_{\text{map2}}^T\ \boldsymbol{t}_{\text{map2}}^T\ \boldsymbol{v}_{\text{map2}}^T]^T) \\ &= \sum_{b=0}^{1}(u_{\text{map2},1b}\cdot\omega_b) + \sum_{c=0}^{H_2}(v_{\text{map2},1c}\cdot z_{\text{map2},c})\end{aligned} \tag{4.5a}$$

$$z_{\text{map2},c} = \begin{cases} \sigma\left(\sum_{b=0}^{1} t_{\text{map2},cb}\cdot\omega_b\right), & c\neq 0 \\ 1, & c=0 \end{cases} \tag{4.5b}$$

$$\omega_0 = 1 \tag{4.5c}$$

where $\boldsymbol{w}_{\text{map2}} = [\boldsymbol{u}_{\text{map2}}^T\ \boldsymbol{t}_{\text{map2}}^T\ \boldsymbol{v}_{\text{map2}}^T]^T$ represents the weights in the frequency mapping neural network f_{map2}. $z_{\text{map2},c}$ is defined as the output of the cth hidden neuron of the frequency mapping neural network. In order to accommodate the convenient representation of the bias parameters in neural networks, we assume the fictitious neurons in the frequency mapping as ω_0 and $z_{\text{map2},0}$ [1]. $u_{\text{map2},1b}$, $v_{\text{map2},1c}$ and $t_{\text{map2},cb}$

represent the connection weight between the ath input neuron and the output neuron, the weight between the cth hidden neuron and the output neuron, and the weight between the bth input neuron and the cth hidden neuron of the frequency mapping neural network, respectively. $u_{\text{map2},1b}$ represents the linear mapping weight and $v_{\text{map2},1c}$ represents the nonlinear mapping weight in the frequency mapping neural network. Therefore, the presented frequency mapping can encompass linear mapping, nonlinear mapping or no mapping by one neural network.

The output mapping is used to enhance the robustness of the modeling process. In the presented method, the output mapping neural network, which has M inputs and M outputs, is defined as

$$\boldsymbol{y} = \boldsymbol{f}_{\text{map3}}(\boldsymbol{y}_E, \boldsymbol{w}_{\text{map3}}) \tag{4.6}$$

where $\boldsymbol{w}_{\text{map3}} = [\boldsymbol{u}_{\text{map3}}^T \ \boldsymbol{t}_{\text{map3}}^T \ \boldsymbol{v}_{\text{map3}}^T]^T$ represents the weights in the output mapping neural network $\boldsymbol{f}_{\text{map3}}$. Similarly as (4.4a) and (4.5a), here we formulate the internal structure of the output mapping neural network to include linear and nonlinear mappings as follows. The rth output of the output mapping is computed as

$$f_{\text{map3},r} = \sum_{p=0}^{m}(u_{\text{map3},rp} \cdot y_{E,p}) + \sum_{q=0}^{H_3}(v_{\text{map3},rq} \cdot z_{\text{map3},q}), \quad r = 1, 2, \cdots, m \tag{4.7a}$$

$$z_{\text{map3},q} = \begin{cases} \sigma\left(\sum_{p=0}^{m} t_{\text{map3},qp} \cdot y_{E,p}\right), & q \neq 0 \\ 1, & q = 0 \end{cases} \tag{4.7b}$$

$$y_{E,0} = 1 \tag{4.7c}$$

where $z_{\text{map3},q}$ is defined as the output of the qth hidden neuron of the output map-

ping neural network. To accommodate the convenient representation of the bias parameters in neural networks, we assume the fictitious neurons in the output mapping as $y_{E,0}$ and $z_{\text{map3},0}$ [1]. $u_{\text{map3},rp}$, $v_{\text{map3},rq}$ and $t_{\text{map3},qp}$ represent the connection weight between the pth input neuron and the rth output neuron, the weight between the qth hidden neuron and the rth output neuron, and the weight between the pth input neuron and the qth hidden neuron of the output mapping neural network, respectively. $u_{\text{map3},rp}$ represents the linear mapping weight and $v_{\text{map3},rq}$ represents the nonlinear mapping weight in the output mapping neural network. In this way, the output mapping can encompass linear mapping, nonlinear mapping, or no mapping by one neural network.

We define $\boldsymbol{w} = [\boldsymbol{w}_{\text{map1}}^T \ \boldsymbol{w}_{\text{map2}}^T \ \boldsymbol{w}_{\text{map3}}^T]^T$ as a vector containing all the neural network weights which are treated as optimization variables during the training process. Combining all the mappings with the empirical model, the overall relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$, i.e., between the inputs and outputs of the overall model, can be represented as

$$\begin{aligned}
\boldsymbol{y} &= \boldsymbol{f}_{\text{map3}}(\boldsymbol{y}_E, \ \boldsymbol{w}_{\text{map3}}) \\
&= \boldsymbol{f}_{\text{map3}}\left(\boldsymbol{f}_E\left(\left[\boldsymbol{x}_E^T \ \boldsymbol{\omega}_E\right]^T, \boldsymbol{w}_E\right), \boldsymbol{w}_{\text{map3}}\right) \\
&= \boldsymbol{f}_{\text{map3}}\left(\boldsymbol{f}_E\left(\left[\boldsymbol{f}_{\text{map1}}^T(\boldsymbol{x}, \boldsymbol{w}_{\text{map1}}) \ f_{\text{map2}}(\omega, \boldsymbol{w}_{\text{map2}})\right]^T, \boldsymbol{w}_E\right), \boldsymbol{w}_{\text{map3}}\right).
\end{aligned} \tag{4.8}$$

This input and output relationship depends on not only the empirical model, but also the various mappings. Therefore, changing the mappings (including changing the mapping structure and changing the values of neural network weights in the mappings) will allow us to change the model.

This unified knowledge-based model includes all cases of mappings, which are no mapping, linear mapping and nonlinear mapping for each of input mapping, frequency mapping and output mapping. Usually, a modeling problem with a small range of geometrical parameters needs linear mappings and a modeling problem with a large range of geometrical parameters needs nonlinear mappings. However, a quantitative decision of when to use linear mapping or when to use nonlinear mapping is problem-dependent and is unknown in advance. The input mapping, frequency mapping and output mapping are not necessarily all linear or all nonlinear. They can be different combinations of linear and nonlinear functions because they have different effects on the modeling behavior.

Existing automated model structure adaptation techniques for knowledge-based model development use a brute force sequential trial and error mechanism to try different combinations of space mapping structures and compare to determine the most suitable knowledge-based neural network structure. Here we utilize the new unified knowledge-based model structure and present a new training method to automatically determine the mapping structure in the final model.

4.3 Proposed Automated Model Generation Algorithm Using l_1 Optimization

4.3.1 Proposed Training Method with l_1 Optimization

In order to train the model, we generate training data. We define the training data as pairs of $\left(\boldsymbol{x}^{(a)}, \boldsymbol{d}^{(a)}\right)$, $a = 1, 2, \cdots, N_L$, where $\boldsymbol{d}^{(a)}$ is a vector containing the desired outputs of the overall knowledge-based neural network model for the ath training

sample $\boldsymbol{x}^{(a)}$, and N_L is the total number of training samples. For training purpose, the error function $E(\boldsymbol{w})$ in the existing knowledge-based neural network literature e.g., [1], is denoted as

$$E(\boldsymbol{w}) = \sum_{a=1}^{N_L} \left(\frac{1}{2} \left\| \boldsymbol{y}(\boldsymbol{x}^{(a)}, \boldsymbol{w}) - \boldsymbol{d}^{(a)} \right\|^2 \right) \tag{4.9}$$

which represents the difference between the model outputs and the EM data outputs over all the training samples.

In the presented method, we perform model training with l_1 optimization. The training process is divided into two stages. In the first stage, the presented algorithm determines whether to use linear mapping or nonlinear mapping in various parts of the model. In the second stage, the algorithm determines whether a linear mapping, if exists, can be removed or not. After these two stages, the presented method automatically produces the mapping structure of the knowledge-based model for the modeling problem.

For the first stage, we present a new formulation of the error function for training. We add the l_1 norms of the neural network weights between hidden neurons and output neurons to the traditional training error function in (4.9). The new training

error function in the first stage of the training process is formulated as

$$\begin{aligned} E^1_{\text{train}}(\boldsymbol{w}) = & \sum_{a=1}^{N_L} \left(\frac{1}{2} \left\| \boldsymbol{y}(\boldsymbol{x}^{(a)}, \boldsymbol{w}) - \boldsymbol{d}^{(a)} \right\|^2 \right) \\ & + \sum_{i=1}^{n} \sum_{k=0}^{H_1} \lambda^1_{\text{map1},ik} \left| v_{\text{map1},ik} \right| \\ & + \sum_{c=0}^{H_2} \lambda^1_{\text{map2},1c} \left| v_{\text{map2},1c} \right| \\ & + \sum_{r=1}^{m} \sum_{q=0}^{H_3} \lambda^1_{\text{map3},rq} \left| v_{\text{map3},rq} \right| \end{aligned} \tag{4.10}$$

where $\lambda^1_{\text{map1},ik}$, $\lambda^1_{\text{map2},1n}$ and $\lambda^1_{\text{map3},rq}$ are all non-negative constants during the first stage training process. The training problem of the first stage is to minimize the training error by adjusting $\boldsymbol{w}$ and force as many nonlinear mapping weights in $\boldsymbol{v}$ to zeros as possible.

The use of the l_1 norm as compared to the other l_p norms with $p > 1$ has the distinctive property that some large components of nonlinear mapping weight vector $\boldsymbol{v}$ can be ignored during the optimization process [102], [103], i.e., at the solution there may well be a few nonlinear mapping weights v's which are non-zeros while others are zeros. This means that, the important components of nonlinear mapping weight vector $\boldsymbol{v}$ can be automatically selected by the l_1 norm. The robustness of the l_1 optimization in dealing with large components of nonlinear mapping weight vector $\boldsymbol{v}$ is the result of a mathematical property related to the necessary conditions for optimality [104]. The solution of the first stage training is usually situated at a point where one or more of the v's equal zero while some large v's are in effect ignored completely by optimization process.

The formulation in (4.10) has two properties. The purpose of the first part in the error function is to optimize the model to match the training data. By adding the remaining parts to the error function, we penalize the error function for any large values of the nonlinear mapping weights in $\boldsymbol{v}$. Since the l_1 norm is used, one or a few large v's are still allowed. In this way, the model optimization can get good training accuracy as well as many v's to zeros. After the first stage training, the nonzero v's represent the selected nonlinear mapping function and nonlinear mapping structure. If some of the nonlinear mapping weights v's are non-zeros, the mapping is nonlinear mapping; if all nonlinear mapping weights v's are zeros, the nonlinear mapping can be deleted, and the remaining mapping structure is linear mapping. In this way, the presented algorithm automatically detects and determines whether to use linear mapping or nonlinear mapping.

We have three mapping neural networks in our presented knowledge-based model and each mapping neural network can be linear or nonlinear. If all these three mappings remain nonlinear mappings after the first stage training, the algorithm will stop. Otherwise, the algorithm proceeds to the second stage of the training process. The three mappings in the first stage solution can be all linear, or all nonlinear, or a mixture of some linear and some nonlinear. For the linear mapping, we present to further retrain the neural network weights during the second stage of training process. For the nonlinear mapping, the neural network weights are set as constants during the second stage training. In the second stage of the training process, our presented algorithm determines whether the linear mappings of the first stage training solution can be further simplified into unit mapping or not. Unit

mapping means the values of the variables before mapping and after mapping are the same. If a mapping is unit mapping, it means this mapping is unnecessary and can be removed. Therefore, the second stage of the training process can determine whether a linear mapping is needed or not. The presented l_1 training error function in the second stage is formulated as

$$\begin{aligned} E^2_{\text{train}}(\boldsymbol{u}) = & \sum_{a=1}^{N_L} \left(\frac{1}{2} \left\| \boldsymbol{y}(\boldsymbol{x}^{(a)}, \boldsymbol{w}) - \boldsymbol{d}^{(a)} \right\|^2 \right) \\ & + \sum_{i=1}^{n} \sum_{j=0, j\neq i}^{n} \lambda^2_{\text{map1},ij} \left| u_{\text{map1},ij} \right| \\ & + \sum_{i=1}^{n} \lambda^2_{\text{map1},ii} \left| u_{\text{map1},ii} - 1 \right| \\ & + \lambda^2_{\text{map2},10} \left| u_{\text{map2},10} \right| + \lambda^2_{\text{map2},11} \left| u_{\text{map2},11} - 1 \right| \\ & + \sum_{r=1}^{m} \sum_{p=0, p\neq r}^{m} \lambda^2_{\text{map3},rp} \left| u_{\text{map3},rp} \right| \\ & + \sum_{r=1}^{m} \lambda^2_{\text{map3},rr} \left| u_{\text{map3},rr} - 1 \right| \end{aligned} \tag{4.11}$$

where $\lambda^2_{\text{map1},ij}$, $\lambda^2_{\text{map2},10}$, $\lambda^2_{\text{map2},11}$ and $\lambda^2_{\text{map3},rp}$ are all non-negative constants during the second stage training process. $\lambda^2_{\text{map1},ij}$ are set to be zeros if the input mapping is nonlinear mapping. Similarly, $\lambda^2_{\text{map2},10}$ and $\lambda^2_{\text{map2},11}$ (or $\lambda^2_{\text{map3},rp}$) are set to be zeros if the frequency mapping (or the output mapping) is nonlinear mapping. In the second stage training solution, if

$$\begin{cases} u_{\text{map1},ij} = 0, & i \neq j \\ u_{\text{map1},ij} = 1, & i = j \end{cases} \tag{4.12}$$

the input mapping is unit mapping, i.e.,

$$\boldsymbol{x}_E = \boldsymbol{x} \tag{4.13}$$

which means the input mapping is not needed in the final knowledge-based model. Similarly, if

$$\begin{cases} u_{\text{map2},10} = 0 \\ u_{\text{map2},11} = 1 \end{cases} \tag{4.14}$$

it means the frequency mapping is not needed in the final model. If

$$\begin{cases} u_{\text{map3},rp} = 0, & r \neq p \\ u_{\text{map3},rp} = 1, & r = p \end{cases} \tag{4.15}$$

the output mapping is unit mapping which means the output mapping is not needed in the final model. Otherwise, the linear mappings are necessary in the final model. In this way, the presented algorithm can automatically determine whether a mapping is needed or not.

4.3.2 Proposed Computation of the Error Derivatives in the Proposed Training Method with l_1 Optimization

During the training process, the derivatives of the error functions in (4.10) and (4.11) with respect to the weights in all three mapping neural networks are needed. Due to the complexity of the presented unified model covering all possibilities of mappings, the brute force derivation for derivatives of the presented model structure is complicated. Here we present a simple and elegant approach based on the back propagation (BP) concept for MLP [79], extended to our unified knowledge-based

structure and l_1 training error function. We define the error at the output of the overall model as

$$\Delta {y_r}^{(a)} = {y_r}^{(a)} - {d_r}^{(a)} \tag{4.16}$$

which represents the difference between the rth model output and the EM data output of the ath sample. Then the presented BP algorithm will start from $\Delta {y_r}^{(a)}$ and propagate this error backwards from the outputs of the output mapping neural network through the hidden layer of the output mapping to the inputs of the output mapping. Let $\Delta z_{\text{map3},q}$ be defined as the BP error back propagated from the output layer of the output mapping to the qth hidden neuron of the output mapping. The calculation of $\Delta z_{\text{map3},q}$ can be derived as

$$\Delta z_{\text{map3},q} = \sum_{r=1}^{m} \left[\Delta {y_r}^{(a)} \cdot v_{\text{map3},rq} \cdot z_{\text{map3},q} \cdot (1 - z_{\text{map3},q}) \right] \tag{4.17}$$

where $\Delta {y_r}^{(a)}$ is the solution from (4.16).

The error BP continues from the output mapping to the empirical model. Let $\Delta y_{E,p}$ be defined as the BP error at the pth output of the empirical model. The calculation of $\Delta y_{E,p}$ can be derived as

$$\Delta y_{E,p} = \sum_{r=1}^{m} \left(\Delta y_r^{(a)} \cdot u_{\text{map3},rp} \right) + \sum_{q=0}^{H_3} \left(\Delta z_{\text{map3},q} \cdot t_{\text{map3},qp} \right) \tag{4.18}$$

where $\Delta {y_r}^{(a)}$ and $\Delta z_{\text{map3},q}$ are the solutions from (4.16) and (4.17), respectively.

Then the error BP continues from the empirical model to the output layer of the input mapping. Let $\Delta x_{E,i}$ be defined as the BP error at the output of the input

mapping neural network, which can be derived as

$$\Delta x_{E,i} = \sum_{p=1}^{m} \left(\Delta y_{E,p} \cdot \frac{\partial y_{E,p}}{\partial x_{E,i}} \right) \tag{4.19}$$

where $\Delta y_{E,p}$ is the solution from (4.18). Let $\Delta z_{\text{map1},k}$ be defined as the BP error back propagated from the output layer of the input mapping to the hidden neuron of the input mapping. The calculation of $\Delta z_{\text{map1},k}$ can be derived as

$$\Delta z_{\text{map1},k} = \sum_{i=1}^{n} \left[\Delta x_{E,i} \cdot v_{\text{map1},ik} \cdot z_{\text{map1},k} \cdot (1 - z_{\text{map1},k}) \right] \tag{4.20}$$

where $\Delta x_{E,i}$ is the solution from (4.19).

The error BP also continues from the empirical model to the output layer of the frequency mapping. Let $\Delta \omega_E$ be defined as the BP error at the output of the frequency mapping neural network. It can be derived as

$$\Delta \omega_E = \sum_{p=1}^{m} \left(\Delta y_{E,p} \cdot \frac{\partial y_{E,p}}{\partial \omega_E} \right) \tag{4.21}$$

where $\Delta y_{E,p}$ is the solution from (4.18). Let $\Delta z_{\text{map2},c}$ be defined as the BP error back propagated from the output layer of the frequency mapping to the hidden neuron of the frequency mapping. The calculation of $\Delta z_{\text{map2},c}$ can be derived as

$$\Delta z_{\text{map2},c} = \Delta \omega_E \cdot v_{(map2,)1c} \cdot z_{\text{map2},c} \cdot (1 - z_{\text{map2},c}) \tag{4.22}$$

where $\Delta \omega_E$ is the solution from (4.21).

After finishing the error BP process, the presented training algorithm starts to calculate the derivatives of the training error with respect to the weights of the three mapping neural networks. Even though the unified model structure is

complicated, the final result of the sensitivity formula is surprisingly simple and elegant as described next. For the first stage of the training process, the error derivatives with respect to the weights in the output mapping can be derived as

$$\frac{\partial \left(E^1_{\text{train}}\right)^{(a)}}{\partial \left(u_{\text{map3},rp}\right)} = \Delta y_r^{(a)} \cdot y_{E,p} \tag{4.23a}$$

$$\frac{\partial \left(E^1_{\text{train}}\right)^{(a)}}{\partial \left(v_{\text{map3},rq}\right)} = \Delta y_r^{(a)} \cdot z_{\text{map3},q} \pm \lambda^1_{\text{map3},rq} \tag{4.23b}$$

$$\frac{\partial \left(E^1_{\text{train}}\right)^{(a)}}{\partial \left(t_{\text{map3},qp}\right)} = \Delta z_{\text{map3},q} \cdot y_{E,p}. \tag{4.23c}$$

If $v_{\text{map3},rq} \geq 0$, the "$\pm$" symbol in (4.23b) is replaced by the "$+$" symbol, otherwise is replaced by the "$-$" symbol. For the neural network weights in the input mapping, the error derivatives can be derived as

$$\frac{\partial \left(E^1_{\text{train}}\right)^{(a)}}{\partial \left(u_{\text{map1},ij}\right)} = \Delta x_{E,i} \cdot x_j \tag{4.24a}$$

$$\frac{\partial \left(E^1_{\text{train}}\right)^{(a)}}{\partial \left(v_{\text{map1},ik}\right)} = \Delta x_{E,i} \cdot z_{\text{map1},k} \pm \lambda^1_{\text{map1},ik} \tag{4.24b}$$

$$\frac{\partial \left(E^1_{\text{train}}\right)^{(a)}}{\partial \left(t_{\text{map1},kj}\right)} = \Delta z_{\text{map1},k} \cdot x_j. \tag{4.24c}$$

If $v_{\text{map1},ik} \geq 0$, the "$\pm$" symbol in (4.24b) is replaced by the "$+$" symbol, otherwise is replaced by the "$-$" symbol. For the neural network weights in the frequency mapping, the error derivatives can be derived as

$$\frac{\partial \left(E^1_{\text{train}}\right)^{(a)}}{\partial \left(u_{\text{map2},1b}\right)} = \Delta \omega_E \cdot \omega_b \tag{4.25a}$$

$$\frac{\partial \left(E^1_{\text{train}}\right)^{(a)}}{\partial \left(v_{\text{map2},1c}\right)} = \Delta \omega_E \cdot z_{\text{map2},c} \pm \lambda^1_{\text{map2},1c} \tag{4.25b}$$

$$\frac{\partial \left(E^1_{\text{train}}\right)^{(a)}}{\partial \left(t_{\text{map2},cb}\right)} = \Delta z_{\text{map2},c} \cdot \omega_b. \tag{4.25c}$$

If $v_{\text{map2},1c} \geq 0$, the "$\pm$" symbol in (4.25b) is replaced by the "+" symbol, otherwise is replaced by the "$-$" symbol.

For the second stage of the training process, the error derivatives with respect to the linear weights (i.e., $u_{\text{map3},rp}$) in the output mapping can be derived as

$$\frac{\partial \left(E^2_{\text{train}}\right)^{(a)}}{\partial \left(u_{\text{map3},rp}\right)} = \Delta y_r^{(a)} \cdot y_{E,p} \pm \lambda^2_{\text{map3},rp}. \tag{4.26}$$

If $u_{\text{map3},rp} \geq 0$, the "$\pm$" symbol in (4.26) is replaced by the "+" symbol, otherwise is replaced by the "$-$" symbol. For the linear neural network weights (i.e., $u_{\text{map1},ij}$) in the input mapping, the error derivatives can be derived as

$$\frac{\partial \left(E^2_{\text{train}}\right)^{(a)}}{\partial \left(u_{\text{map1},ij}\right)} = \Delta x_{E,i} \cdot x_j \pm \lambda^2_{\text{map1},ij}. \tag{4.27}$$

If $u_{\text{map1},ij} \geq 0$, the "$\pm$" symbol in (4.27) is replaced by the "+" symbol, otherwise is replaced by the "$-$" symbol. For the linear neural network weights (i.e., $u_{\text{map2},1m}$) in the frequency mapping, the error derivatives can be derived as

$$\frac{\partial \left(E^2_{\text{train}}\right)^{(a)}}{\partial \left(u_{\text{map2},1m}\right)} = \Delta \omega_E \cdot \omega_m \pm \lambda^2_{\text{map2},1m}. \tag{4.28}$$

If $u_{\text{map2},1m} \geq 0$, the "$\pm$" symbol in (4.28) is replaced by the "+" symbol, otherwise is replaced by the "$-$" symbol.

Using above derivative information for training, the pro-posed algorithm proceeds to find a training solution which determines the mapping structure. In the presented algorithm, we do not need to manually determine whether the mappings

are linear or nonlinear, whether an input, frequency or output mapping is needed or not. The model structure can be automatically determined by training with l_1 optimization. The nonzero v's represent the selected nonlinear mapping function and nonlinear mapping structure. If the nonlinear mapping weights v's are nonzeros, the mapping is nonlinear mapping; if the nonlinear mapping weights v's are all zeros, the mapping is linear mapping. The nonzero linear mapping coefficients u's represent the selected linear mapping.

4.3.3 Simple Illustration Examples of the Mapping Type for One Mapping Neural Network

We take the input mapping as an example. If

$$\begin{cases} u_{\text{map1},ij} = 0, & i \neq j \\ u_{\text{map1},ij} = 1, & i = j \\ \boldsymbol{v}_{\text{map1}} = \mathbf{0} \end{cases} \tag{4.29}$$

the input mapping function (4.4a) is simplified as

$$f_{\text{map1},i} = x_j, \quad i = j = 1, 2, \cdots, N \tag{4.30}$$

which means the input mapping is unit mapping and there is no need for the input mapping in the final model.

If

$$\begin{cases} u_{\text{map1},ij} \neq 0, & i \neq j \\ \boldsymbol{v}_{\text{map1}} = \mathbf{0} \end{cases} \tag{4.31}$$

the input mapping function (4.4a) is simplified as

$$f_{\text{map1},i} = \sum_{j=0}^{n} u_{\text{map1},ij} x_j, \quad i = 1, 2, \cdots, n \tag{4.32}$$

which means the input mapping $\boldsymbol{f}_{\text{map1}}$ is linear mapping.

If

$$\boldsymbol{v}_{\text{map1}} \neq \mathbf{0} \tag{4.33}$$

the input mapping function $\boldsymbol{f}_{map1}$ is nonlinear mapping.

The determinations for the frequency mapping and the output mapping neural networks are similar to that for the input mapping. In this way, the presented algorithm uses l_1 optimization to automatically obtain a most suitable knowledge-based model.

4.3.4 Simple Illustration Examples of the Overall Model Structure Including Various Mappings

In this subsection, we show several examples of different model structures to illustrate that the final model can be various combinations of mapping neural networks after using optimization to obtain the knowledge-based model.

1) Model Structure Example 1: If after l_1 optimization, the nonlinear input mapping weights $\boldsymbol{v}_{\text{map1}} \neq \mathbf{0}$, the nonlinear frequency mapping weights $\boldsymbol{v}_{\text{map2}} = \mathbf{0}$, the linear frequency mapping weights $u_{\text{map2},10} = 0$, $u_{\text{map2},11} = 1$, the nonlinear output mapping weights $\boldsymbol{v}_{\text{map3}} = \mathbf{0}$, and the linear output mapping weights $u_{\text{map3},rp} =$

$0, (r \neq p), u_{\text{map3},rp} = 1, (r = p)$, (4.8) will be simplified as

$$\boldsymbol{y} = \boldsymbol{f}_E \left(\left[\boldsymbol{f}_{map1}^T \left(\boldsymbol{x}, \boldsymbol{w}_{\text{map1}} \right) \ \omega \right]^T, \boldsymbol{w}_E \right) \tag{4.34}$$

which means the final knowledge-based model has only a nonlinear input mapping. There is no need for frequency mapping and no need for output mapping in the final model.

2) Model Structure Example 2: If the nonlinear input mapping weights $\boldsymbol{v}_{\text{map1}} \neq \boldsymbol{0}$, the nonlinear frequency mapping weights $\boldsymbol{v}_{\text{map2}} \neq \boldsymbol{0}$, the nonlinear output mapping weights $\boldsymbol{v}_{\text{map3}} = \boldsymbol{0}$, and the linear output mapping weights $u_{\text{map3},rp} = 0, (r \neq p), u_{\text{map3},rp} = 1, (r = p)$, the final model will have a nonlinear input mapping and a nonlinear frequency mapping. There is no need for output mapping in the final model. Then (4.8) will be simplified as

$$\boldsymbol{y} = \boldsymbol{f}_E \left(\left[\boldsymbol{f}_{map1}^T \left(\boldsymbol{x}, \boldsymbol{w}_{\text{map1}} \right) \ f_{\text{map2}} \left(\omega, \boldsymbol{w}_{\text{map2}} \right) \right]^T, \boldsymbol{w}_E \right). \tag{4.35}$$

3) Model Structure Example 3: If the nonlinear input mapping weights $\boldsymbol{v}_{\text{map1}} \neq \boldsymbol{0}$, the nonlinear frequency mapping weights $\boldsymbol{v}_{\text{map2}} = \boldsymbol{0}$, the linear frequency mapping weights $u_{\text{map2},10} \neq 0$, $u_{\text{map2},11} \neq 1$, and the nonlinear output mapping weights $\boldsymbol{v}_{\text{map3}} \neq \boldsymbol{0}$, the final model will contain a nonlinear input mapping, a linear frequency mapping and a nonlinear output mapping. Then the model will be

$$\boldsymbol{y} = \boldsymbol{f}_{\text{map3}} \left(\boldsymbol{f}_E \left(\left[\boldsymbol{f}_{map1}^T \left(\boldsymbol{x}, \boldsymbol{w}_{\text{map1}} \right) \ f_{\text{map2}} \left(\omega, \boldsymbol{w}_{\text{map2}} \right) \right]^T, \boldsymbol{w}_E \right), \boldsymbol{w}_{\text{map3}} \right). \tag{4.36}$$

Using l_1 optimization, the presented method can automatically determine the combination of input mapping, frequency mapping and output mapping. Each

mapping can be linear or nonlinear mapping or no mapping.

4.3.5 Proposed Automated Modeling Algorithm

The presented unified automated modeling algorithm uses the training method with l_1 optimization to train a unified knowledge-based model structure, forcing the model to determine the type and topology of space mapping structure automatically. The testing error $E_{\text{test}}(\boldsymbol{w}^*)$ is defined as

$$E_{\text{test}}(\boldsymbol{w}^*) = \sum_{b=1}^{N_V} \left(\frac{1}{2} \left\| \boldsymbol{y}(\boldsymbol{x}^{(b)}, \boldsymbol{w}^*) - \boldsymbol{d}^{(b)} \right\|^2 \right) \tag{4.37}$$

where $\boldsymbol{w}^*$ is the optimal solution after the training process containing the weight parameters of all the mapping neural networks. $\boldsymbol{d}^{(b)}$ represents the testing data output of the bth input sample $\boldsymbol{x}^{(b)}$ and N_V represents the number of testing data.

We define E_d as the user-defined threshold. During the first stage of the training process, once $E_{\text{test}} \leq E_d$ is detected, the algorithm checks the values of the trained nonlinear mapping weights $\boldsymbol{v}$ in all three mapping neural networks. If all three mappings are nonlinear mappings, the algorithm stops. Otherwise, the algorithm proceeds to the second stage of the training process to determine whether the linear mapping is needed or not. The algorithm will retrain the weights in the linear mapping until $E_{\text{test}} \leq E_d$ is achieved again.

The presented algorithm can be summarized as follows.

Step 1) Perform EM simulation to generate training and testing data using Designed of Experiments (DoE) sampling method [105].

Step 2) Optimize the existing empirical model to make the model approximate the

training data as much as possible.

Step 3) Test the optimized empirical model. If the empirical model satisfies the accuracy requirement, which means the empirical model is good enough, stop the modeling process. Else, we add mappings to the empirical model to improve the model in subsequent steps.

Step 4) Create initial mapping neural networks $\boldsymbol{f}_{\text{map1}}, f_{\text{map2}}$ and $\boldsymbol{f}_{\text{map3}}$ with five hidden neurons in each neural network, and initialize the weight parameters of the mapping neural networks.

Step 5) Perform the first stage training with l_1 optimization, using the training error function in (4.10) to determine whether the mappings are linear or nonlinear.

Step 6) Test the model. If $E_{\text{train}} > E_d$, the model is detected as under-learning [13]; add one more hidden neuron and go back to Step 5). Else if $E_{\text{train}} \leq E_d$ and $E_{\text{test}} > E_d$, the model is detected as over-learning [13]; delete one hidden neuron in the nonlinear neural networks and go back to Step 5). Else, go to Step 7).

Step 7) If $\boldsymbol{v}_{\text{map1}} \neq \boldsymbol{0}, \boldsymbol{v}_{\text{map2}} \neq \boldsymbol{0}$ and $\boldsymbol{v}_{\text{map3}} \neq \boldsymbol{0}$, all mappings are nonlinear and all mapping weights in the neural networks are determined. Stop the modeling process. Else, at least one mapping is linear. Set the neural network weights in the nonlinear mappings as constants, and leave the neural network weights in the linear mappings as variables. Perform the second stage training with l_1 optimization until $E_{\text{test}} \leq E_d$, using the training error function in (4.11).

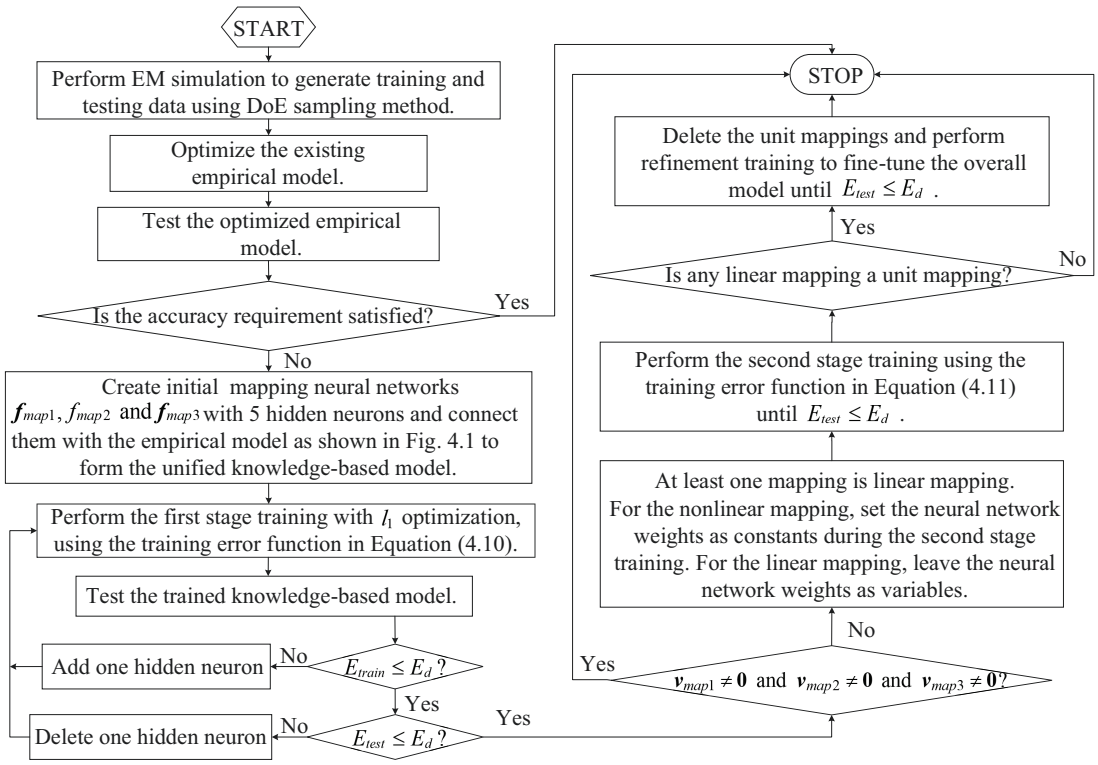


Figure 4.2: Block diagram of the proposed unified automated knowledge-based neural network modeling method using l_1 optimization.

Step 8) If one or more linear mappings are unit mapping, which means the linear mappings are unnecessary, delete the unnecessary linear mappings and perform refinement training to fine-tune the overall model until $E_{\text{test}} \leq E_d$. Then stop the modeling process. Else, the linear mappings are needed in the final model. Stop the modeling process.

Fig. 4.2 shows the flow diagram of our presented unified automated modeling algorithm using knowledge-based neural network and l_1 optimization.

4.3.6 Discussion

$\lambda^1_{\text{map1},ik}$, $\lambda^1_{\text{map2},1c}$, $\lambda^1_{\text{map3},rq}$ in (4.10) and $\lambda^2_{\text{map1},ij}$, $\lambda^2_{\text{map2},11}$, $\lambda^2_{\text{map3},rp}$ in (4.11) are user-defined factors. The values of these factors are either zero or non-zero. Zero λ's mean the l_1 norms of the corresponding parameters in the mapping neural network are not included in the training error functions. Non-zero λ's mean that large values in the corresponding parameters of the mapping neural network are penalized using l_1 optimization. For non-zero λ's, we use 10 as the default value. In (4.10), if the values of the λ's are large, the training process will emphasize more on forcing as many v's to zeros as possible, while emphasize less on optimizing the model to match the training data. On the other hand, if the values of the λ's in (4.10) are small but not equal to zero, the training error of the model will tend to be small but we cannot guarantee that the values of v's in the model are as small as possible. The λ's in (4.11) work in similar way as those in (4.10).

An important difference between the presented method and the previous automated modeling method [4] is that our algorithm finds best knowledge-based models while the previous method produces pure neural network models. Knowledge-based models have better extrapolation capability than pure neural network models as originally demonstrated in [8]. By using the knowledge-based model formulation in this chapter, the presented training method enables the extrapolation capability of the models.

This chapter focuses on proposing a computational method for parametric modeling to represent the change of EM behavior versus the change in geometrical parameters. We perform validation by using independent EM data which are never

used in training of the model. For validation of EM simulation data including measurement validation, we refer to existing works such as [106] and [107].

4.4 Application Examples

4.4.1 Parametric Modeling of Two-Section Low-Pass Elliptic Microstrip Filter

In this example, we develop a knowledge-based parametric model for a two-section low-pass elliptic microstrip filter [40], [108], [109], as shown in Fig. 4.3. The training and testing data are obtained from EM simulator *CST Microwave Studio* using DoE sampling method [105]. In Fig. 4.3, H is the thickness of the alumina substrate and ϵ_r is the dielectric constant of the substrate. $H = 0.508$ mm and $\epsilon_r = 9.4$. The geometrical input parameters of the model are $\boldsymbol{x} = [L_1\ L_2\ L_{c1}\ L_{c2}\ W_c\ G_c]^T$, chosen based on the sensitivity data of the above low-pass elliptic filter example. The model output is the magnitude of S_{21}. The existing empirical model is the equivalent circuit for the low-pass filter using simple transmission lines, which is shown in Fig. 4.4 [86], [99]. We build our empirical model using formulas based on *Keysight Advanced Design System (ADS)* [109], [110]. After optimization, the empirical model still cannot satisfy the accuracy requirement, therefore we proceed to add mappings using our presented method. *NeuroModelerPlus* software is used to drive EM simulators for data generation, program the empirical model, implement unified mapping structures and perform the knowledge-based model training and testing. The new training error functions in (4.10) and (4.11) are implemented through Huber functions in *NeuroModelerPlus*. The values of $\lambda^1_{\text{map1},ik}$, $\lambda^1_{\text{map2},1c}$ and

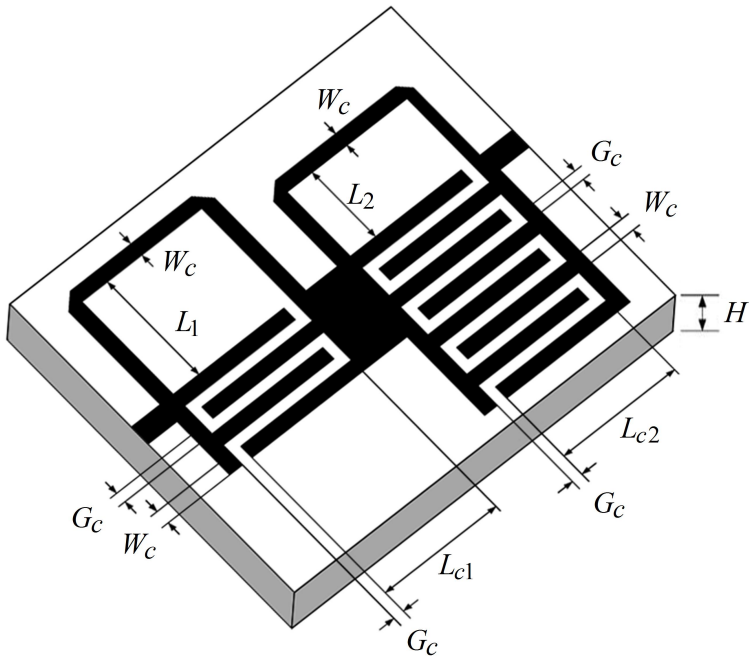


Figure 4.3: Low-pass elliptic microstrip filter example: EM simulation for data generation.

$\lambda^1_{\text{map3},rq}$ in (4.10) are set to be 10.

For comparison purposes, we develop knowledge-based models for two different cases. The range of the geometrical parameters for Case 1 is small, while the range of the geometrical parameters for Case 2 is large. These two user-desired modeling ranges are shown in Table 4.1. We set the testing error threshold for this modeling example as 2%.

The modeling results are listed in Table 4.2 and shown in Fig. 4.5. For Case 1, after using l_1 optimization, a linear input map-ping is good enough to obtain an accurate knowledge-based model with about 2% testing error. The values of $\lambda^2_{\text{map1},ij}, \lambda^2_{\text{map2},10}, \lambda^2_{\text{map2},11}$ and $\lambda^2_{\text{map3},rp}$ in (4.11) are equal to 10. All nonlinear map-

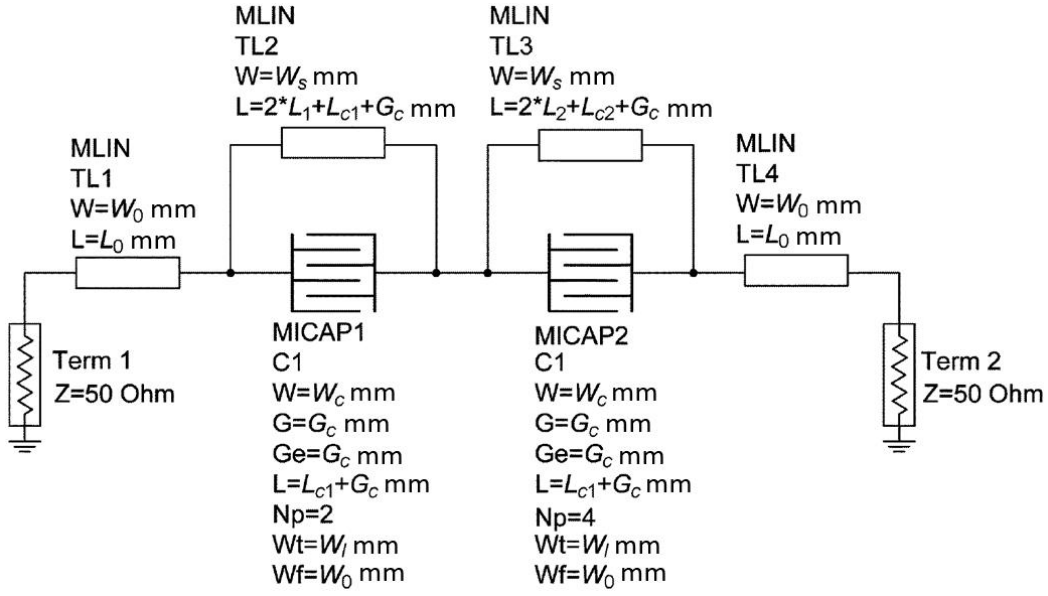


Figure 4.4: Low-pass elliptic microstrip filter example: the empirical model.

ping weights $\boldsymbol{v}_{\text{map1}}, \boldsymbol{v}_{\text{map2}}$ and $\boldsymbol{v}_{\text{map3}}$ in mapping neural networks are almost zeros in the final knowledge-based model, therefore all mappings in the final model are linear mappings. Besides, the frequency mapping and output mapping are unit mapping, which means frequency mapping and output mapping are not needed. This result is consistent with our observation that since the range of the modeling space is small, a linear mapping is sufficient.

For Case 2, we also perform l_1 optimization during the model development. After l_1 optimization, the nonlinear mapping weights v_{map1} and v_{map2} of the input mapping and frequency mapping are non-zeros and the nonlinear mapping weights v_{map3} of the output mapping are zeros. The values of $\lambda^2_{\text{map1},ij}$, $\lambda^2_{\text{map2},10}$ and $\lambda^2_{\text{map2},11}$ in (4.11) are equal to 0, while the values of $\lambda^2_{\text{map3},rp}$ in (4.11) are equal to 10. Besides, the output mapping is unit mapping. This modeling result means the presented

Table 4.1: Training Data and Testing Data for Parametric Modeling of the Low-Pass Filter Example

Input	Training Data		Testing Data	
	Min	Max	Min	Max
	Case 1			
L_1 (mm)	1.14	1.24	1.15	1.23
L_2 (mm)	3.3	3.5	3.31	3.49
L_{c1} (mm)	4.06	4.26	4.07	4.25
L_{c2} (mm)	1.14	1.24	1.15	1.23
W_c (mm)	0.16	0.2	0.165	0.195
G_c (mm)	0.05	0.07	0.052	0.069
Freq (GHz)	1	4	1	4
	Case 2			
L_1 (mm)	1.02	1.27	1.03	1.26
L_2 (mm)	3.18	3.68	3.2	3.65
L_{c1} (mm)	3.94	4.44	3.96	4.42
L_{c2} (mm)	1.0	1.27	1.02	1.25
W_c (mm)	0.15	0.25	0.16	0.24
G_c (mm)	0.05	0.09	0.053	0.085
Freq (GHz)	1	4	1	4

Table 4.2: Comparison of the Modeling Results for Low-Pass Filter with Two Different Modeling Ranges

	Case 1	Case 2
Mapping Structure of the Final Model	Linear input mapping	Nonlinear input mapping + Nonlinear frequency mapping
No. of Training Data	81*101	81*101
No. of Testing Data	64*101	64*101
Training Error	1.89%	1.63%
Testing Error	1.92%	1.97%

algorithm with optimization automatically chooses a nonlinear input mapping and a nonlinear frequency mapping to map the difference between the empirical model

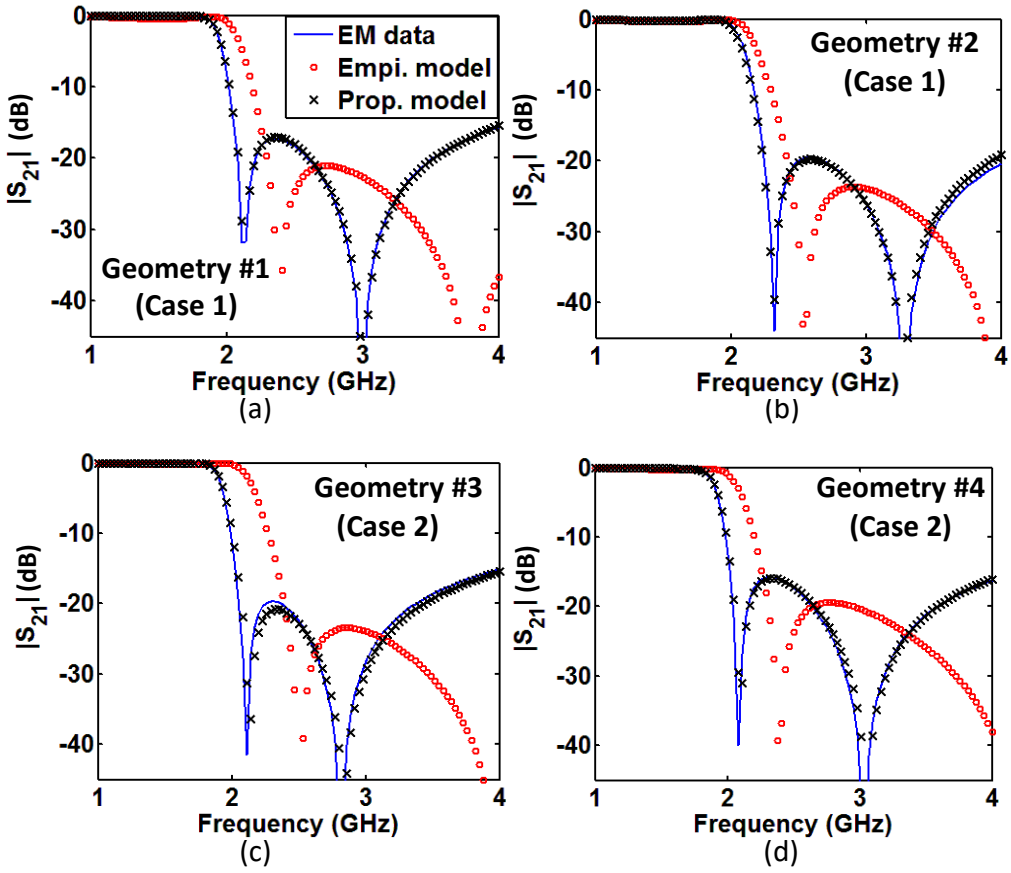


Figure 4.5: Low-pass elliptic microstrip filter example: modeling results at four different geometrical values (a) $\boldsymbol{x} = [1.15,\ 3.49,\ 4.25,\ 1.16,\ 0.18,\ 0.06]^T$, (b) $\boldsymbol{x} = [1.23,\ 3.47,\ 4.08,\ 1.17,\ 0.19,\ 0.069]^T$, (c) $\boldsymbol{x} = [1.03,\ 3.52,\ 4.29,\ 1.22,\ 0.23,\ 0.058]^T$ and (d) $\boldsymbol{x} = [1.13,\ 3.46,\ 4.35,\ 1.10,\ 0.21,\ 0.06]^T$. The solid line, "o" and "x" in the figures represent the EM simulation data, the empirical model response and the presented model response, respectively.

and EM simulation. The output mapping is not needed. For Case 2, we have also generated another set of testing data which is beyond the training range to test the presented model, i.e., L_1: $0.96 - 1.32$ mm, L_2: $3.05 - 3.81$ mm, L_{c1}: $3.81 - 4.32$

mm, L_{c2}: $0.96 - 1.32$ mm, W_c: $0.12 - 0.28$ mm and G_c: $0.045 - 0.096$ mm. The testing error is 2.35% which means the presented model also has reasonable accuracy beyond the training range because of the embedded knowledge in the model. From these two modeling examples in Case 1 and Case 2, it is demonstrated that the presented algorithm uses l_1 optimization to determine whether to use linear mapping or nonlinear mapping for the specific modeling problem and determine the types of mapping structures automatically.

We compare the modeling results using existing brute force knowledge-based modeling algorithm and the presented automated model generation algorithm, shown in Table 4.3. In the brute force knowledge-based modeling method, we have to try different mapping structures in order to choose an accurate knowledge-based model. Here Table 4.3 shows only some examples of the trials in the existing brute force modeling algorithm. In reality, the number of the possible combinations of the mappings in a knowledge-based model is more than that listed in Table 4.3. We also compare the modeling time between the existing method in [16] and our presented method, as shown in Table 4.4. The presented method increases the modeling efficiency and develops the model in a shorter time than the brute force modeling approaches.

To further compare between the model produced by the presented method and the fixed structured model, we consider a fixed model structure where all neural network mappings are nonlinear (refer to as "three nonlinear mapping structure"). We use less training data to develop models for Case 1 and Case 2. The results are shown in Table 4.5. The testing error of the model with three nonlinear mappings

Table 4.3: Comparison Between the Model from the Proposed Automated Modeling Method and the Models from Existing Brute Force Modeling Algorithms with Different Combinations of Mappings for the Low-Pass Filter Modeling Problem

Model Structure	Testing Error	
	Case 1	Case 2
Optimal Model Structure by Proposed Method: **Empirical Model + Linear Input Mapping**	**1.92%**	/
Optimal Model Structure by Proposed Method: **Empirical Model + Nonlinear Input Mapping + Nonlinear Frequency Mapping**	/	**1.97%**
Empirical Model Alone	16.43%	14.33%
Empirical Model + Linear Input Mapping	1.92%	2.38%
Empirical Model + Nonlinear Input Mapping	1.88%	2.04%
Empirical Model + Linear Output Mapping	24.95%	13.80%
Empirical Model + Nonlinear Output Mapping	7.73%	9.10%
Empirical Model + Linear Input Mapping + Linear Frequency Mapping + Linear Output Mapping	1.89%	2.17%
Empirical Model + Nonlinear Input Mapping + Nonlinear Frequency Mapping + Nonlinear Output Mapping	1.85%	1.96%

is larger than that of the presented model, and cannot satisfy the user-desired accuracy. In this situation, the status of training the model with three nonlinear

Table 4.4: Comparison of the Modleing Time for Low-Pass Filter Using the Existing Method in [16] and the Proposed Method

	Case 1	Case 2
Existing Method in [16]	4.5 h	4.3 h
Propose Method	3.3 h	3.35 h

Table 4.5: Comparison of the Modeling Results for Low-Pass Filter Using the Three Nonlinear Mapping Structure and the Proposed Method with Two Different Modeling Ranges

	Case 1		Case 2	
Mapping Structure of the Final Model	3 nonlinear mappings (fixed)	Linear input mapping (proposed)	3 nonlinear mappings (fixed)	Nonlinear input mapping (proposed)
No. of Training Data	49*101		49*101	
No. of Testing Data	64*101		64*101	
Training Error	1.21%	1.45%	1.87%	1.74%
Testing Error	2.82%	1.92%	2.43%	1.98%

mappings is over-learning. More training data are needed to obtain an accurate model satisfying the user-desired accuracy. Because the presented algorithm with l_1 optimization forces the model to be as simple as possible while preserving accuracy, less data are needed by the presented method to develop the knowledge-based model. Therefore, the presented technique is more efficient than simply choosing three nonlinear mappings when the number of training data is limited. In addition, the model produced by the presented method is simpler and more compact than the model with three nonlinear mappings. The resulting compact model is easier to use and easier to be incorporated into circuit simulators.

4.4.2 Parametric Modeling of Bandpass HTS Microstrip Filter

Consider the parametric modeling of an HTS quarterwave parallel coupled-line microstrip filter [40], [89], [90], illustrated in Fig. 4.6. In the figure, L_1, L_2 and L_3 are the lengths of the parallel coupled-line sections, and S_1, S_2 and S_3 are the gaps between the sections. L_0 is the length of the input and output transmission line feeding the coupled line filter. The width $W = 0.635$ mm is assumed to be the same for all the sections. A lanthanum-aluminate substrate with thickness $H = 0.508$ mm and dielectric constant $\epsilon_r = 23.425$ is used.

The model input variables $\boldsymbol{x} = [L_1\ L_2\ L_3\ S_1\ S_2\ S_3]^T$ for this example are chosen based on the sensitivity data and the model output is the magnitude of S_{21}. The testing error criteria for this modeling example is 4%. Training and testing data generation is performed by *CST Microwave Studio* EM simulator using DoE sampling method [105]. We build the empirical model using formulas based on *Keysight ADS* [110], [111], shown in Fig. 4.7 [90]. After the optimization of the empirical model, the error is still beyond our threshold. Therefore we proceed to add mappings using our presented method. The values of $\lambda^1_{\text{map1},ik}$, $\lambda^1_{\text{map2},1c}$ and $\lambda^1_{\text{map3},rq}$ in (4.10) are set to be 10.

We develop knowledge-based bandpass filter models for two different sets of geometrical parameter ranges, shown in Table 4.6. For Case 1, the range of the geometrical parameters is small, while the range of the geometrical parameters for Case 2 is large. The modeling results are listed in Table 4.7 and Fig. 4.8 shows the comparison of the empirical model, the presented knowledge-based model and the

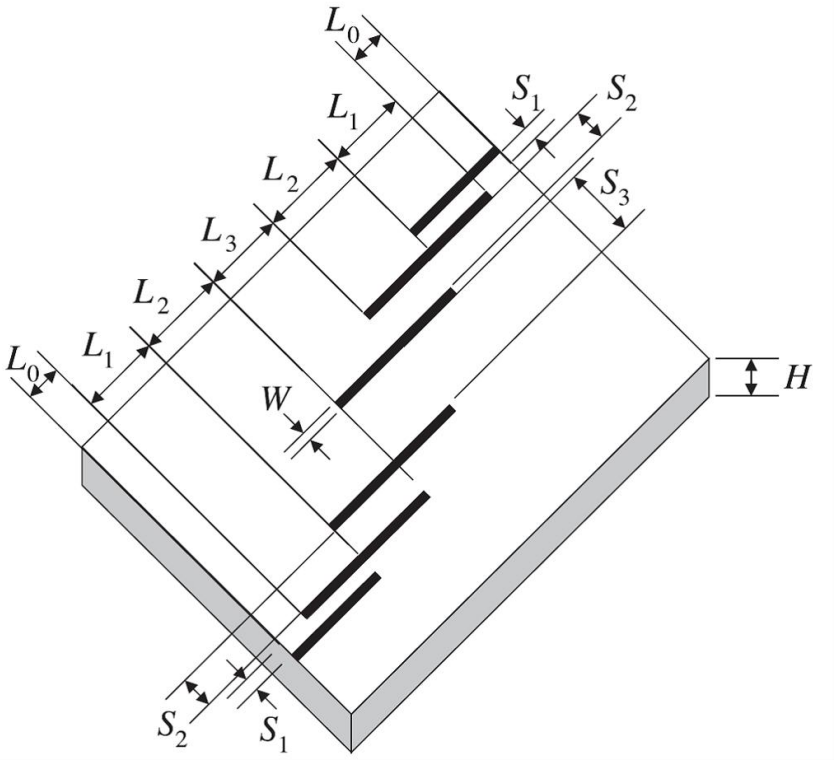


Figure 4.6: Bandpass HTS microstrip filter example: EM simulation for data generation.

EM simulation at four different sets of geometrical values.

For Case 1, the modeling range is relatively small. We perform l_1 optimization to develop the model. After l_1 optimization, the nonlinear mapping weights $\boldsymbol{v}_{\text{map1}}, \boldsymbol{v}_{\text{map2}}$ and $\boldsymbol{v}_{\text{map3}}$ are almost zeros, which means all mappings are linear mapping. The values of $\lambda^2_{\text{map1},ij}, \lambda^2_{\text{map2},10}, \lambda^2_{\text{map2},11}$ and $\lambda^2_{\text{map3},rp}$ in (4.11) are equal to 10. Besides, the linear frequency mapping and the linear output mapping are unit mapping, which means these two mappings are not needed. Therefore, the presented algorithm with l_1 optimization uses only a linear input mapping to obtain

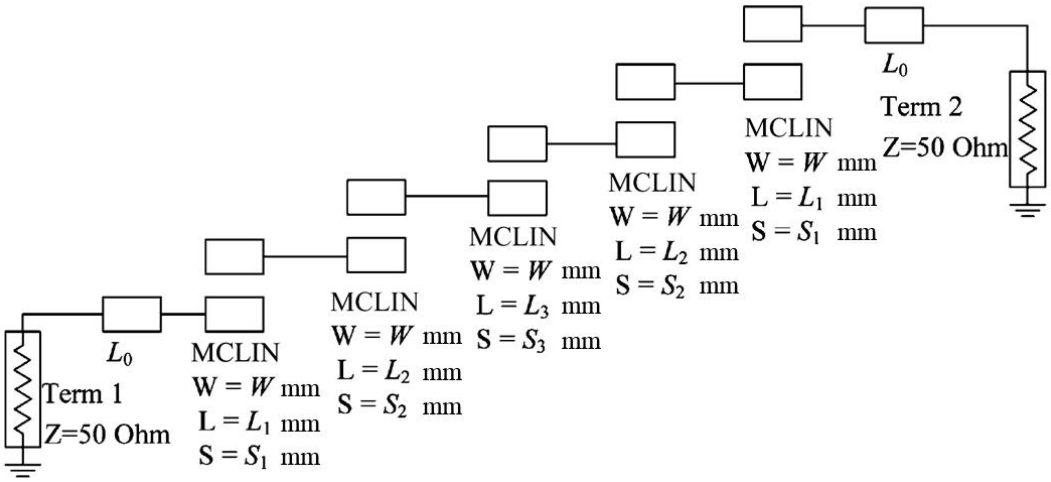


Figure 4.7: Bandpass HTS microstrip filter example: the empirical model.

a knowledge-based model which meets the user-desired accuracy. This final model structure from l_1 optimization is consistent with observation that a linear mapping is sufficient for the modeling problem with a relatively small modeling range.

For Case 2, after l_1 optimization, the nonlinear weights $\boldsymbol{v}_{\text{map2}}$ of the frequency mapping are all zeros while $\boldsymbol{v}_{\text{map1}}$ and $\boldsymbol{v}_{\text{map3}}$ in the input mapping and output mapping are non-zeros. Therefore the final model contains a nonlinear input mapping, a linear frequency mapping and a nonlinear output mapping. The values of $\lambda^2_{\text{map1},ij}$ and $\lambda^2_{\text{map3},rp}$ in (4.11) are equal to 0, while the values of $\lambda^2_{\text{map2},10}$ and $\lambda^2_{\text{map2},11}$ in (4.11) are equal to 10. For Case 2, we have also generated another set of testing data which is beyond the training range to test the presented model, i.e., L_1: $4.47 - 5.08$ mm, L_2: $4.72 - 5.33$ mm, L_3: $4.47 - 5.08$ mm, S_1: $0.44 - 0.58$ mm, S_2: $1.72 - 2.03$ mm and S_3: $1.98 - 2.29$ mm. The testing error is 5.54% which means the extrapolation capability of the presented model is reasonable because of the embedded knowledge

Table 4.6: Training Data and Testing Data for Parametric Modeling of the Bandpass Filter Example

Input	Training Data		Testing Data	
	Min	Max	Min	Max
		Case 1		
L_1 (mm)	4.56	4.78	4.58	4.76
L_2 (mm)	4.82	5.03	4.84	5.01
L_3 (mm)	4.57	4.78	4.58	4.76
S_1 (mm)	0.4	0.6	0.41	0.59
S_2 (mm)	2.03	2.24	2.04	2.22
S_3 (mm)	2.28	2.48	2.3	2.47
Freq (GHz)	3.8	4.2	3.8	4.2
		Case 2		
L_1 (mm)	4.57	4.98	4.6	4.95
L_2 (mm)	4.82	5.23	4.85	5.2
L_3 (mm)	4.57	4.98	4.6	4.95
S_1 (mm)	0.46	0.54	0.47	0.53
S_2 (mm)	1.78	1.98	1.79	1.97
S_3 (mm)	2.03	2.24	2.04	2.22
Freq (GHz)	3.8	4.2	3.8	4.2

Table 4.7: Comparison of the Modeling Results for Bandpass Filter with Two Different Modeling Ranges

	Case 1	Case 2
Mapping Structure of the Final Model	Linear input mapping	Nonlinear input mapping + Linear frequency mapping + Nonlinear output mapping
No. of Training Data	81*101	81*101
No. of Testing Data	64*101	64*101
Training Error	4.13%	3.87%
Testing Error	3.96%	3.87%

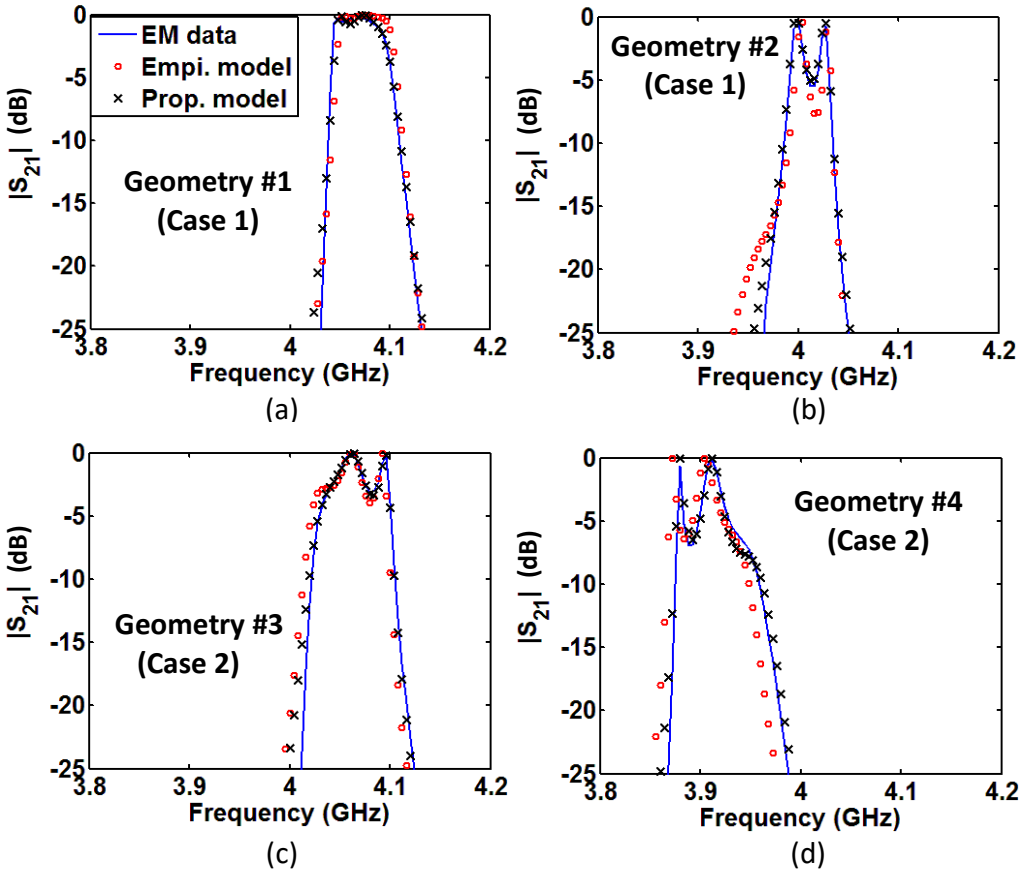


Figure 4.8: Bandpass HTS microstrip filter example: modeling results at four different geometrical values (a) $\boldsymbol{x} = [4.58,\ 4.86,\ 4.61,\ 0.47,\ 2.12,\ 2.4]^T$, (b) $\boldsymbol{x} = [4.76,\ 5.0,\ 4.64,\ 0.42,\ 2.12,\ 2.45]^T$, (c) $\boldsymbol{x} = [4.65,\ 4.9,\ 4.6,\ \ 0.5,\ 1.97,\ 2.1]^T$ and (d) $\boldsymbol{x} = [4.8,\ 5.0,\ 4.9,\ 0.47,\ 1.96,\ 2.12]^T$. The solid line, "o" and "x" in the figures represent the EM simulation data, the empirical model response and the presented model response, respectively.

in the model. The model structure in Case 2 is more complicated than that in Case 1, which is consistent with the fact that the modeling range is larger and the nonlinearity of the space mapping is stronger for Case 2 than for Case 1. The comparison

of the modeling results between the existing brute force knowledge-based modeling methods and the presented automated modeling algorithm is listed in Table 4.8. In reality, the number of the possible combinations of the mappings in a knowledge-based model is more than that listed in Table 4.8. We also compare the modeling time between the existing method in [16] and our presented method, as shown in Table 4.9.

The modeling results illustrate that our presented automated knowledge-based model generation algorithm with l_1 optimization has the ability to distinguish the linearity of specific modeling example. For different modeling examples, it can develop knowledge-based models with different mapping structures. The final model is forced to be as compact as possible by l_1 optimization property to force as many weights to zeros as possible. In addition, the first part in the l_1 error function makes the final model satisfy the user-required accuracy at the same time. Our presented algorithm increases the modeling efficiency and reduces the modeling time compared with the existing method.

4.4.3 Parametric Modeling of Bandstop Microstrip Filter with Open Stubs

The third example is the parametric model development for a bandstop microstrip filter with quarter-wave resonant open stubs [40], [90], [98], illustrated in Fig. 4.9. In the figure, L_1 and L_2 are the open stub lengths, and W_1 and W_2 are the open stub widths. L_0 is the interconnecting transmission line length between the two open stubs. W_0 is the width of a 50 Ω feeding line and $W_0 = 0.635$ mm. An alumina

Table 4.8: Comparison Between the Model from the Proposed Automated Modeling Method and the Models from Existing Brute Force Modeling Algorithms with Different Combinations of Mappings for the Bandpass Filter Modeling Problem

Model Structure	Testing Error	
	Case 1	Case 2
Optimal Model Structure by Proposed Method: **Empirical Model + Linear Input Mapping**	**3.96%**	/
Optimal Model Structure by Proposed Method: **Empirical Model + Nonlinear Input Mapping + Linear Frequency Mapping + Nonlinear Output Mapping**	/	**3.87%**
Empirical Model Alone	9.43%	12.11%
Empirical Model + Linear Input Mapping	3.96%	9.16%
Empirical Model + Nonlinear Input Mapping	4.06%	4.66%
Empirical Model + Linear Output Mapping	9.25%	11.91%
Empirical Model + Nonlinear Output Mapping	9.02%	11.59%
Empirical Model + Linear Input Mapping + Linear Frequency Mapping + Linear Output Mapping	3.94%	6.24%
Empirical Model + Nonlinear Input Mapping + Nonlinear Frequency Mapping + Nonlinear Output Mapping	3.87%	3.48%

Table 4.9: Comparison of the Modeling Time for Bandpass Filter Using the Existing Method in [16] and the Proposed Method

	Case 1	Case 2
Existing Method in [16]	6.5 h	6.75 h
Propose Method	4.32 h	5.03 h

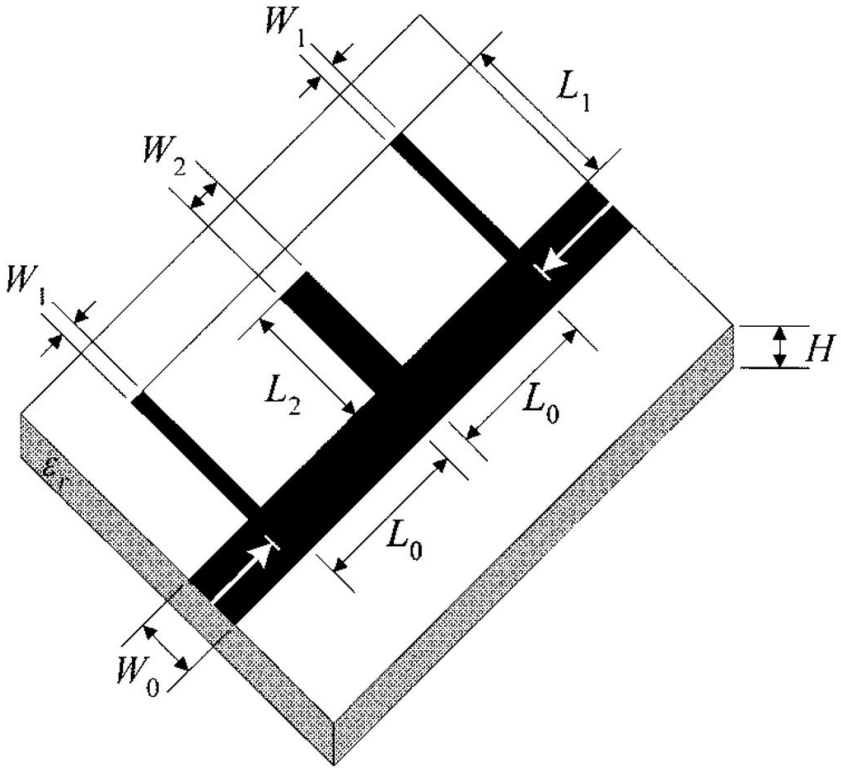


Figure 4.9: Bandstop microstrip filter example: EM simulation for data generation.

substrate with thickness of $H = 0.635$ mm and dielectric constant $\epsilon_r = 9.4$ is used.

The input variables of the model $\boldsymbol{x} = [W_1\ W_2\ L_0\ L_1\ L_2]^T$ are chosen based on the sensitivity data. The model output is the magnitude of S_{21}. Data generation is performed by *CST Microwave Studio* EM simulator using DoE sampling method

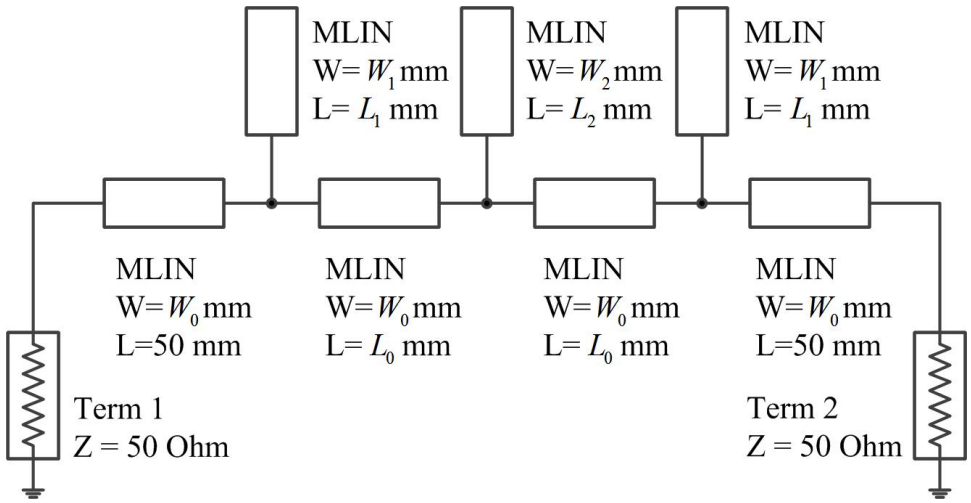


Figure 4.10: Bandstop microstrip filter example: the empirical model.

[105]. The empirical model is the equivalent circuit for the bandstop filter using simple transmission lines, which is shown in Fig. 4.10 [90]. We build our empirical model using formulas based on *Keysight ADS* [110]. The testing error threshold for this bandstop model is set as 2%. After the optimization of the empirical model, the error is still beyond our threshold. Therefore we proceed to add mappings using our presented method. The values of $\lambda^1_{\text{map1},ik}$, $\lambda^1_{\text{map2},1c}$ and $\lambda^1_{\text{map3},rq}$ in (4.10) are set to be 10. By using l_1 optimization, the presented method can force the weights of the unnecessary mappings to zeros. Therefore we can remove the unnecessary mappings and produce the most compact mapping structure for the final knowledge-based model to satisfy the accuracy requirement.

For comparison, we develop knowledge-based models for two different sets of geometrical parameter ranges using l_1 optimization, as shown in Table 4.10. For Case 1, the range of the geometrical parameters is small, while the range of the

Table 4.10: Training Data and Testing Data for Parametric Modeling of the Bandstop Filter Example

Input	Training Data		Testing Data	
	Min	Max	Min	Max
	Case 1			
W_1 (mm)	0.15	0.19	0.155	0.185
W_2 (mm)	0.23	0.27	0.232	0.268
L_0 (mm)	2.54	3.35	2.59	3.3
L_1 (mm)	2.54	3.35	2.59	3.3
L_2 (mm)	2.54	3.35	2.59	3.3
Freq (GHz)	5	15	5	15
	Case 2			
W_1 (mm)	0.075	0.23	0.08	0.225
W_2 (mm)	0.11	0.34	0.12	0.33
L_0 (mm)	2.0	3.35	2.05	3.3
L_1 (mm)	2.0	3.35	2.05	3.3
L_2 (mm)	2.0	3.35	2.05	3.3
Freq (GHz)	5	15	5	15

geometrical parameters for Case 2 is large. Table 4.11 and Fig. 4.11 show the comparison of the modeling results for bandstop filter with two different modeling ranges.

In Case 1, after l_1 optimization, the nonlinear mapping weights $\boldsymbol{v}_{\text{map1}}, \boldsymbol{v}_{\text{map2}}$ and $\boldsymbol{v}_{\text{map3}}$ in three mappings of the knowledge-based model are almost zeros, therefore all three mappings of the final model are linear mapping. The values of $\lambda^2_{\text{map1},ij}, \lambda^2_{\text{map2},10}, \lambda^2_{\text{map2},11}$ and $\lambda^2_{\text{map3},rp}$ in (4.11) are equal to 10. Besides, the frequency mapping and the output mapping are unit mapping. The presented algorithm with l_1 optimization uses only a linear input mapping to obtain a knowledge-based model which meets the required accuracy. There is no need for frequency

Table 4.11: Comparison of the Modeling Results for Bandstop Filter with Two Different Modeling Ranges

	Case 1	Case 2
Mapping Structure of the Final Model	Linear input mapping	Nonlinear input mapping + Linear frequency mapping + Nonlinear output mapping
No. of Training Data	81*101	81*101
No. of Testing Data	64*101	64*101
Training Error	2.0%	2.07%
Testing Error	1.92%	1.8%

mapping and output mapping in the final model. This modeling result and the observation that a linear mapping is sufficient for a modeling problem with small modeling range are consistent.

For a large modeling range (Case 2), the presented algorithm with l_1 optimization develops a knowledge-based model with a nonlinear input mapping, a linear frequency mapping and a nonlinear output mapping because after l_1 optimization, the nonlinear weights $\boldsymbol{v}_{\text{map1}}$ and $\boldsymbol{v}_{\text{map3}}$ of the input mapping and output mapping are non-zeros while $\boldsymbol{v}_{\text{map2}}$ in the frequency mapping are all zeros. The values of $\lambda^2_{\text{map1},ij}$ and $\lambda^2_{\text{map3},rp}$ in (4.11) are equal to 0, while the values of $\lambda^2_{\text{map2},10}$ and $\lambda^2_{\text{map2},11}$ in (4.11) are equal to 10. For Case 2, we have also generated another set of testing data which is beyond the training range to test the presented model, i.e., W_1: $0.06 - 0.24$ mm, W_2: $0.1 - 0.35$ mm, L_0: $1.9 - 3.43$ mm, L_1: $1.9 - 3.43$ mm and L_2: $1.9 - 3.43$ mm. The testing error is 4.53%. Within the training range, the presented model have small testing error which satisfy the user-desired accuracy.

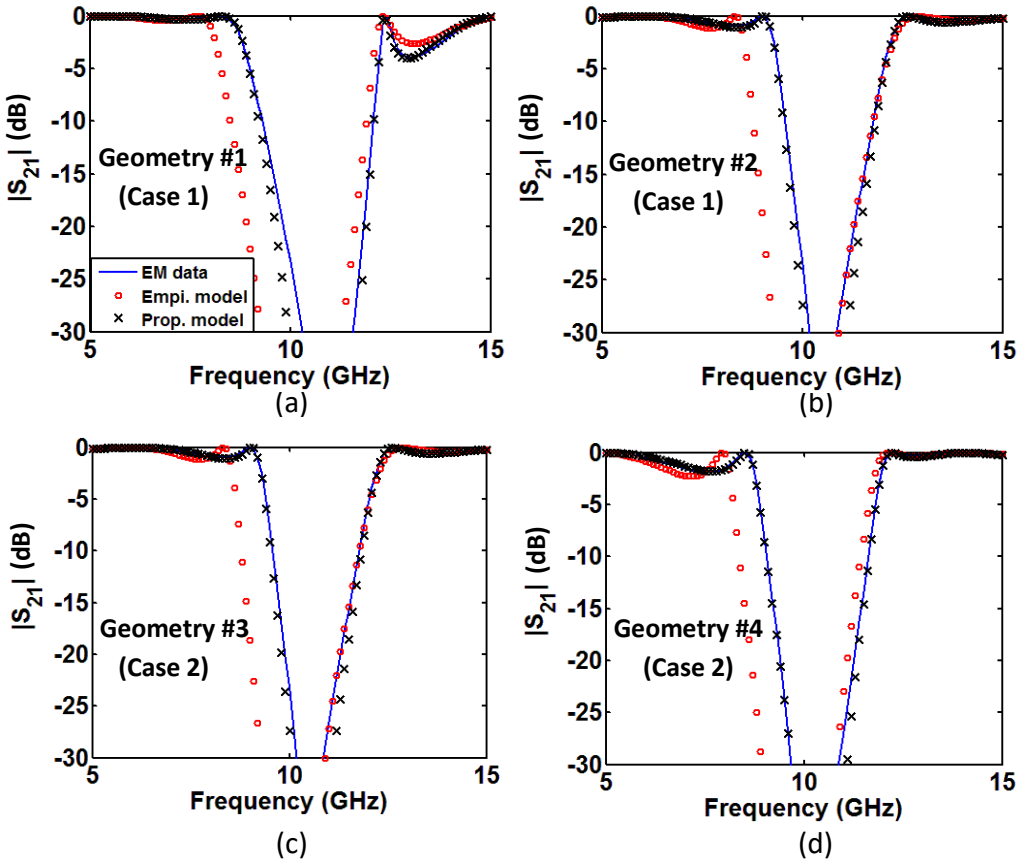


Figure 4.11: Bandstop microstrip filter example: modeling results at four different geometrical values (a) $\boldsymbol{x} = [0.185,\ 0.26,\ 2.69,\ 2.59,\ 2.79]^T$, (b) $\boldsymbol{x} = [0.19,\ 0.27,\ 2.59,\ 2.9,\ 3.2]^T$, (c) $\boldsymbol{x} = [0.11,\ 0.17,\ 2.29,\ 2.79,\ 2.79]^T$ and (d) $\boldsymbol{x} = [0.17,\ 0.14,\ 2.92,\ 2.92,\ 2.67]^T$. The solid line, "o" and "x" in the figures represent the EM simulation data, the empirical model response and the presented model response, respectively.

Beyond the training range, the presented model still has reasonable accuracy which means the extrapolation capability of the model is retained because of the embedded knowledge.

Table 4.12 shows the comparison between the presented automated modeling method and the existing brute force modeling method with different combination of input, frequency and output mapping neural networks. In reality, the number of the possible combinations of the mappings in a knowledge-based model is more than that listed in Table 4.12. We also compare the modeling time between the existing method in [16] and our presented method, as shown in Table 4.13. The modeling results demonstrate that for different modeling examples, our presented method with l_1 optimization can develop knowledge-based models with different combinations of mappings, which makes the model development process very flexible. The presented algorithm increases the modeling efficiency and develops a knowledge-based model with the compact mapping structure in a shorter time than the existing knowledge-based modeling algorithms.

4.5 Conclusion

In this chapter, a unified automated parametric modeling algorithm using combined knowledge-based neural network and l_1 optimization has been presented. In our method, we present a new unified knowledge-based model structure and new sensitivity formulas. The presented knowledge-based model combines the learning ability of the neural network with the convenience and simplicity of the empirical model. We use the properties of l_1 norm to present a training method which can automatically determine whether a mapping is necessary, or whether a mapping is linear or nonlinear. Instead of sequential trials of linear and nonlinear mappings every time for every mapping structure as in the existing literature, the presented

Table 4.12: Comparison Between the Model from the Proposed Automated Modeling Method and the Models from Existing Brute Force Modeling Algorithms with Different Combinations of Mappings for the Bandstop Filter Modeling Problem

Model Structure	Testing Error	
	Case 1	Case 2
Optimal Model Structure by Proposed Method: **Empirical Model + Linear Input Mapping**	**1.92%**	/
Optimal Model Structure by Proposed Method: **Empirical Model + Nonlinear Input Mapping + Linear Frequency Mapping + Nonlinear Output Mapping**	/	**1.8%**
Empirical Model Alone	21.51%	20.31%
Empirical Model + Linear Input Mapping	1.92%	3.62%
Empirical Model + Nonlinear Input Mapping	1.97%	2.31%
Empirical Model + Linear Output Mapping	17.2%	18.13%
Empirical Model + Nonlinear Output Mapping	17.0%	16.87%
Empirical Model + Linear Input Mapping + Linear Frequency Mapping + Linear Output Mapping	1.79%	3.3%
Empirical Model + Nonlinear Input Mapping + Nonlinear Frequency Mapping + Nonlinear Output Mapping	1.39%	1.7%

Table 4.13: Comparison of the Modeling Time for Bandstop Filter Using the Existing Method in [16] and the Proposed Method

	Case 1	Case 2
Existing Method in [16]	4.4 h	4.13 h
Propose Method	2.43 h	3.3 h

algorithm uses l_1 optimization to automatically determine the final model structure. Compared to existing knowledge-based modeling methods, the presented algorithm increases the knowledge-based modeling efficiency and speeds up the model development process.

Chapter 5

Advanced Extrapolation Technique for Neural Based Microwave Modeling and Design

In this chapter, an advanced multi-dimensional extrapolation technique for neural-based microwave modeling and design is proposed to address the modeling challenges in microwave simulation and optimization, such as EM optimization and large-signal harmonic balance (HB) simulation. A standard neural model is accurate only within the particular range where it is trained by training data, and is unreliable if used outside this range. Our proposed method aims to address this issue by extrapolation to provide information and guide the neural network outside the training range. The given training data can be randomly distributed in the input space, and the boundaries of the training region can be arbitrary. The unknown target values of the model outside the training range are formulated as optimization variables and are determined by optimization, such that 1st order continuity of model outputs versus inputs is preserved and 2nd order derivatives are

minimized everywhere. Formulas for 1st order continuity and 2nd order derivatives are derived through cubic polynomial functions. In this way, the formulation of the proposed method guarantees good model accuracy inside the training region, and makes the model maximally smooth across all directions everywhere outside the training region. Compared to existing extrapolation methods for neural networks, the proposed extrapolation technique makes neural models more robust, resulting in faster convergence in microwave simulation and optimization involving neural model inputs as iterative variables. The validity of the proposed technique is demonstrated using both EM optimization example and nonlinear microwave simulation examples.

5.1 Introduction

In general, a well-trained neural network model has good accuracy and can represent the component behavior it learned within the training region [1]. However, the accuracy of the neural network model outside the training region decreases and this creates limitations when neural models are used to explore design solutions in a larger range than the model training range. For example, when a neural network model is used in the iterative computational loops during EM optimization or HB simulation, the iterative variables which are the inputs of the neural model may go beyond the neural model training range during the iterative process. The poor performance of the trained neural model outside the training range may mislead the iterative process into slow convergence or even divergence.

Several extrapolation methods [9] on neural network models have been described in the literature to address this issue, such as knowledge-based neural models [8],

[15], [98], neural network model-level extrapolation [10], [11] and neuron-level extrapolation [12]. Knowledge-based neural models [8] exploit existing knowledge in the form of empirical or equivalent circuit models together with neural networks to help model extrapolation. The extrapolation capability of the knowledge-based neural model is enhanced because of the embedded empirical models. Mathematical extrapolation methods [10], [11] have been developed to address the issue when empirical models are not available for the modeling problem. These model-level extrapolation methods use the neural model and its derivative information at the boundaries of the neural network training region to help model extrapolation. Recently, a simple neuron-level extrapolation method [12] is presented to address the problem in case when the training boundaries are irregular. It detects the necessity of extrapolation inside each hidden neuron and performs extrapolation using a modified neuron activation function. It simplifies the problem into a combination of several one-dimensional 1st order extrapolations along the neuron switch directions of a set of hidden neurons. This method is simple and easy to use.

However, the existing extrapolation methods on neural network models mentioned above also have limitations. Knowledge-based neural network [8], [15], [98], depends on the availability of empirical or equivalent models. In the absence of such a model, knowledge-based neural network cannot be used. In microwave modeling such as time-delay neural network (TDNN) for nonlinear transistor and power amplifier modeling [97], the multi-dimensional training boundaries can be irregular or unknown. In this case, existing model-level extrapolation approaches [10], [11] become less effective or complicated. This is because that the extrapolation is con-

strained by fixed formulas which give smoothness along the extrapolation direction away from the boundary but do not guarantee smoothness across different extrapolation directions. The simple combination of several one-dimensional 1st order extrapolations in neuron-level extrapolation method [12] works well in low dimensions or with few hidden neurons. With increased dimension or increased number of hidden neurons, the hidden neuron switch directions (along which extrapolations are made) are not always aligned well with the normal directions of the training boundaries in the input space, affecting the quality of extrapolation.

For extrapolation methods on neural network models, the smoothness of the neural model outside the training range is very important in microwave simulation and optimization. The smooth tendency of the model outside the training range across different directions can make the neural model more robust and lead to faster convergence in microwave simulation and optimization. However, how to make the neural model smooth outside the training range across different directions still remains an open subject.

In this chapter, we propose a novel and efficient multi-dimensional extrapolation method for neural-based microwave modeling and design. The multi-dimensional training region can be irregular. Under our proposed formulation, the final neural model will fit training data wherever data are given inside the training region, and will be continuous and maximally smooth across different directions wherever data are not given outside the training region. Starting from randomly distributed training samples, we discretize the entire region of model inputs into grids where the entire region includes training region and extrapolation region. Extrapolation

is then performed over the grids. Target values of the model at some of the grid points are known, i.e., where training data are available, while target values of the model at remaining grid points are unknown, i.e., in the extrapolation region. These unknown target values are formulated as optimization variables and are determined by optimization such that the model preserves 1st order continuity and guarantees maximum smoothness at each extrapolation point. We achieve smoothness by minimizing the 2nd order derivatives everywhere in the entire region. The equations for 1st order and 2nd order derivatives at each point are derived through the cubic polynomial functions [112] over the grids. The minimization of 2nd order derivatives everywhere is a high-dimensional optimization problem with unknown target values of the model in various extrapolation grids as variables. Analytical derivation is also presented in this chapter to simplify this high-dimensional optimization problem into a large sparse system of linear equations, and we propose to use the Conjugate Gradient (CG) method [113] to calculate the numerical solutions of the resulting large sparse system of linear equations. Such numerical solutions become the target values of the model outside the training range, and will be used to guide the neural network training beyond the training range. In other words, we use optimization to add extrapolated data outside the training range.

After obtaining the extrapolated data in the extrapolation region, both the training data and extrapolated data are used to train the neural network to achieve neural-based models with good accuracy inside the training range and smooth tendency outside the training range. Compared to the existing extrapolation methods for neural networks, the proposed extrapolation technique makes neural models

more robust, resulting in faster convergence in microwave simulation and optimization involving neural model inputs as iterative variables.

5.2 Proposed Multi-Dimensional Extrapolation Method for Neural Models with Given Training Data

5.2.1 Formulation of Grids in the Entire Region

For neural network extrapolation, the given training data can be randomly scattered and not necessarily in a grid form. The boundaries of the training region can be arbitrary. Starting from the given training data, we formulate grids in both the training region and the extrapolation region. Let $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$ represent a vector of n inputs of a neural network model. Typical elements of $\boldsymbol{x}$ are geometrical parameters such as length and width of an EM structure, or voltage/current variables such as gate voltage and drain voltage of a transistor. We discretize the input space (i.e., the $\boldsymbol{x}$-space) into n-dimensional grids using a set of grid points along each dimension of the space. The set of grid points along each dimension is determined by the coordinates/locations of training data inside the training region and extra user-defined points outside the training region.

For example, suppose the distribution of 12 scattered training data in a two-dimensional (2-D) input space (x_1, x_2) and the user-defined extrapolation range are shown in Fig. 5.1(a). Based on the coordinates/locations of all training data, the nonuniform grids inside the training region are determined. The locations of nonuniform grids along the 1st dimension of the training region are 0, 1, 2, 4, 6, 8, and along the 2nd dimension of the training region are 0, 2, 3, 4, 5, 7, 8, 10, as

shown in Fig. 5.1(b). In addition, user-defined grids outside the training region are added, i.e., -4, -3, -2, -1, 9, 10, 11, 12 for the 1st dimension of the input space and -3, -2, -1, 11, 12, 13 for the 2nd dimension of the input space, respectively. The nonuniform grids in the entire region are shown in Fig. 5.1(c).

After formulating the nonuniform grids, the number of grids along each dimension and the step size of each grid along each dimension become available. The entire region is divided into many n-dimensional cells. For example, a cell is a rectangular area for $n = 2$ and a cubic region for $n = 3$. We define m_k to represent the number of grid points along the kth dimension, where $k = 1, 2, \ldots, n$. Let N_c and N_g represent the number of cells and the number of grid points in the entire region, respectively, i.e.,

$$N_c = \prod_{k=1}^{n} (m_k - 1) \tag{5.1}$$

and

$$N_g = \prod_{k=1}^{n} m_k \tag{5.2}$$

For the example in Fig. 5.1(c), $m_1 = 14$, $m_2 = 14$, and the entire region is divided into 169 cells (i.e., 169 rectangles). The number of cells is $N_c = 169$, and the number of grid points is $N_g = 256$. We define $x_k^{(i_k)}$ to represent the i_kth grid point along the kth dimension of the entire region, where $i_k = 1, 2, \ldots, m_k - 1$ and $k = 1, 2, \ldots, n$. A cell is defined as a n-dimensional cubic space $\left\{ x_k | x_k^{(i_k)} \leqslant x_k \leqslant x_k^{(i_k+1)}, \text{where } k = 1, 2, \ldots, n \right\}$. We define the origin of this cell as $\left(x_1^{(i_1)}, x_2^{(i_2)}, \ldots, x_k^{(i_k)}, \ldots, x_n^{(i_n)} \right)$. Therefore, the index set $(i_1, i_2, \ldots, i_n)$ represents the origin of the cell. Different values of $(i_1, i_2, \ldots, i_n)$ will indicate different cells. We also define $s_k^{(i_k)}$ to represent the step

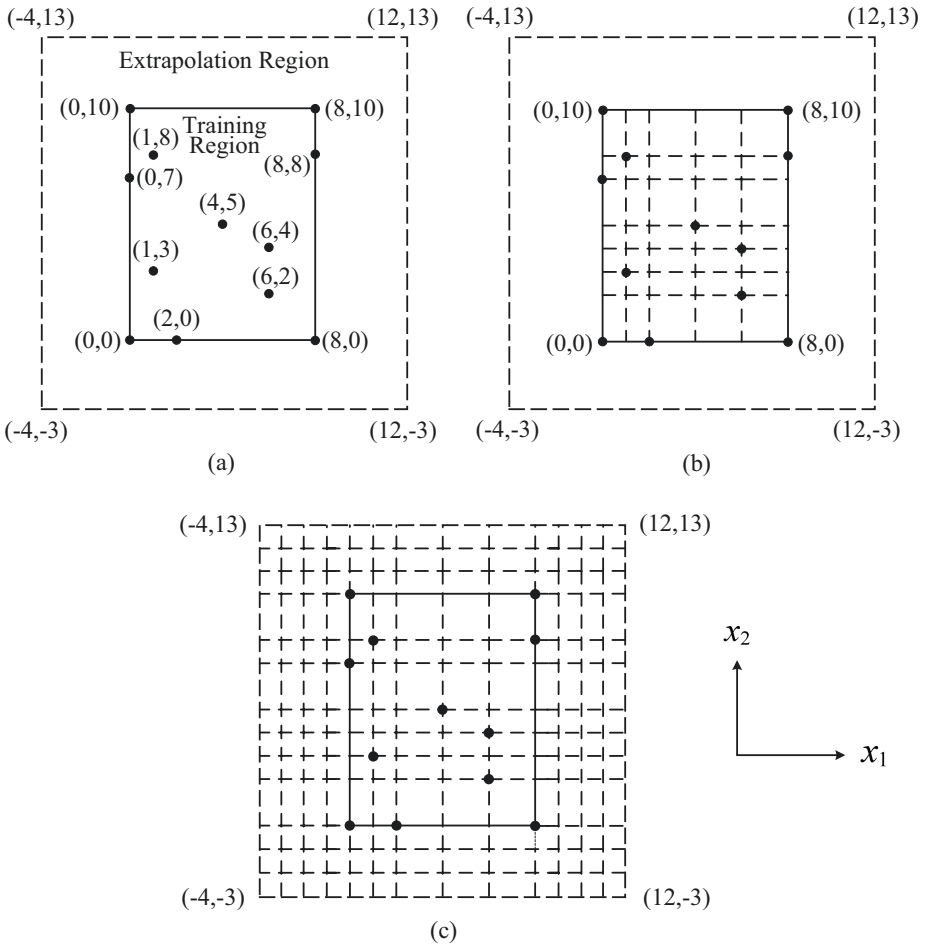


Figure 5.1: (a) Distribution of 12 scattered training data in a 2-D input space (x_1, x_2) and the user-defined extrapolation range. "•" represents the scattered training data. (b) The nonuniform grids of the training region determined by the distribution of the training data. (c) The nonuniform grids of the extrapolation region and training region.

size along the kth dimension of the cell, i.e.,

$$s_k^{(i_k)} = x_k^{(i_k+1)} - x_k^{(i_k)} \tag{5.3}$$

In this way, the nonuniform grids of the entire region are formulated. At some of these grid points, the target values of the model are known, i.e., the training data, while at other grid points, the target values of the model are unknown as shown in Fig. 5.1(c). These unknown target values are model extrapolation values to be determined using the proposed method. Let N_E and N_L represent the number of extrapolated data and the number of training data respectively, i.e., $N_E + N_L = N_g$. Let $\boldsymbol{y}_e$ and $\boldsymbol{y}_L$ represent the vector containing the extrapolated target values and the vector containing the output values of given training data, respectively. More specifically, $\boldsymbol{y}_e$ contains the extrapolated target values at the N_E grid points, and $\boldsymbol{y}_L$ contains the output values of available training data at the other N_L grid points.

5.2.2 Introduction of Cubic Polynomial Formulation

In the proposed extrapolation method, we use the information of given training data $\boldsymbol{y}_L$ to create new training data $\boldsymbol{y}_e$ in extrapolation region, and derive the values of these new training data through optimization. During optimization, the elements in $\boldsymbol{y}_e$ are the optimization variables. The 1st order continuity of model outputs versus inputs and maximum smoothness of the model across different directions in the entire region are the objectives of the optimization. The smoothness is represented by 2nd order derivatives. Formulas for 2nd order derivatives are derived based on the concept of cubic polynomial formulation [112]. In this section, we review the existing cubic polynomial formulation.

Let f represent the cubic polynomial function for the cell with the origin $(i_1, i_2, \ldots,$

$i_k, \ldots, i_n$). The cubic formulation [112] of f for a n-dimensional problem is

$$f = \sum_{j_1=0}^{3} \sum_{j_2=0}^{3} \cdots \sum_{j_n=0}^{3} \left[\alpha_t \prod_{k=1}^{n} \left(\frac{x_k - x_k^{(i_k)}}{x_k^{(i_k+1)} - x_k^{(i_k)}} \right)^{j_k} \right] \quad \text{for} \quad x_k^{(i_k)} \leqslant x_k \leqslant x_k^{(i_k+1)} \tag{5.4}$$

where t is defined as

$$t = 1 + 4^{n-1} \cdot j_1 + 4^{n-2} \cdot j_2 + \cdots + 4^0 \cdot j_n \tag{5.5}$$

In (5.4), α_t represent the unknown coefficients for the cell. The total number of such unknown coefficients for the cell is 4^n. Let $\boldsymbol{\alpha}$ be a vector containing these 4^n coefficients, i.e.,

$$\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_t, \ldots, \alpha_{4^n}]^T \tag{5.6}$$

For example, for a 2-D extrapolation problem (i.e., $n = 2$), if $x_1^{(i_1)} = 0$, $x_2^{(i_2)} = 0$, $x_1^{(i_1+1)} - x_1^{(i_1)} = 1$ and $x_2^{(i_2+1)} - x_2^{(i_2)} = 1$, then

$$\begin{aligned} f = {} & \alpha_1 + \alpha_2 x_2 + \alpha_3 {x_2}^2 + \alpha_4 {x_2}^3 \\ & + \alpha_5 x_1 + \alpha_6 x_1 x_2 + \alpha_7 x_1 {x_2}^2 + \alpha_8 x_1 {x_2}^3 \\ & + \alpha_9 {x_1}^2 + \alpha_{10} {x_1}^2 x_2 + \alpha_{11} {x_1}^2 {x_2}^2 + \alpha_{12} {x_1}^2 {x_2}^3 \\ & + \alpha_{13} {x_1}^3 + \alpha_{14} {x_1}^3 x_2 + \alpha_{15} {x_1}^3 {x_2}^2 + \alpha_{16} {x_1}^3 {x_2}^3 \end{aligned} \tag{5.7}$$

Therefore, $\boldsymbol{\alpha}$ is a vector containing 16 unknown coefficients for a 2-D problem.

Let N represent the number of corner points of a cell, i.e., $N = 2^n$. Let $p_1, p_2, \cdots, p_N$ represent the indices for the N corner points of the cell. According to existing cubic polynomial formulation [112], given the function value f and

the derivative values of f at every corner point of the cell, the unknown coefficients $\boldsymbol{\alpha}$ can be solved by

$$\boldsymbol{A}\boldsymbol{\alpha} = \boldsymbol{b} \tag{5.8}$$

where $\boldsymbol{b}$ is a vector containing the function value f and 1st order to nth order mixed partial derivative values of f with respect to all input variables at N corner points of the cell, and $\boldsymbol{A}$ is a sparse matrix containing the grid information. For a n-dimensional problem, the size of $\boldsymbol{A}$ matrix is $4^n \times 4^n$ and the size of $\boldsymbol{b}$ vector is 4^n. The detailed definitions of $\boldsymbol{A}$ and $\boldsymbol{b}$ are available from existing literature such as [112] and a summary of such detailed definitions is provided in Appendix A. For a 2-D example, the coordinates and the indices for the corner points of a cell are shown in Fig. 5.2. As shown in the figure, the number of corner points of a cell in the 2-D example is four, i.e., $N = 4$. The vector $\boldsymbol{b}$ contains the values of f, $\frac{\partial f}{\partial x_1}$, $\frac{\partial f}{\partial x_2}$ and $\frac{\partial^2 f}{\partial x_1 \partial x_2}$ at four corner points p_1, p_2, p_3 and p_4. The vector $\boldsymbol{\alpha}$ has 16 elements, i.e., $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_{16}]^T$ according to (5.7).

5.2.3 Formulation of Proposed Multi-Dimensional Cubic Polynomial Extrapolation for One Cell

In standard cubic interpolation [112], the elements in $\boldsymbol{b}$ are given data so that $\boldsymbol{\alpha}$ can be solved from (5.8). However, in our case, the 1st order to higher order derivative values of f required in $\boldsymbol{b}$ are not available from given data. Further, even the values of f at some grid points (i.e., extrapolated grid points) are not available. Take the parametric modeling of an EM structure as an example. Let $\boldsymbol{x}$ be the length and width parameters in the EM structure and f be the S-parameter. In this example, $\boldsymbol{b}$

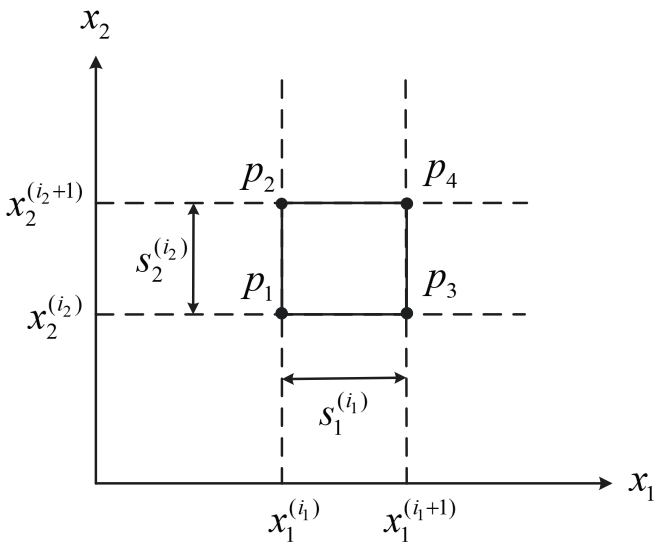


Figure 5.2: The coordinates and the indices for the corner points of a cell in a 2-D example. There are 4 corner points for a cell in the 2-D example, i.e., $N = 4$. x_1 and x_2 can represent geometrical parameters in an EM structure, such as length and width, or voltage/current variables in a nonlinear device, such as gate and drain voltages of a transistor.

contains the S-parameter values and the 1st order to higher order partial derivative values of the S-parameter with respect to length and width at different grid points in the length-width space. In our case, the training data for higher order derivatives of the S-parameter with respect to length and width are not available. Therefore, the values of $\boldsymbol{b}$ are not available. Furthermore, if a grid point is in the extrapolation region, even the S-parameter value at this grid point is not available. Therefore, we cannot use the method in standard cubic interpolation [112] as is. We need to derive new formulas.

In this chapter, we propose to use finite difference formulas between the corner points of a given cell and its neighboring cells in place of the vector $\boldsymbol{b}$. By doing this, we create a set of new linear equations based on the model output values at corner points of the given cell and its neighboring cells.

To facilitate the derivation, we introduce a new vector. We define $\tilde{\boldsymbol{y}}$ as a vector containing the model output values at the 4^n corner points of a given cell and its neighboring cells. As an example, we illustrate the definition of vector $\tilde{\boldsymbol{y}}$ for a 2-D problem. Fig. 5.3 shows a given cell and its neighboring cells for a 2-D case. The total number of corner points in the cell and its neighboring cells is 16 (i.e., 4^n). In this case, the vector $\tilde{\boldsymbol{y}}$ will contain the model output values at the 16 grid points in Fig. 5.3. Notice that these 16 grid points are the corner points of a given cell and its neighboring cells.

Next, we describe how to use $\tilde{\boldsymbol{y}}$ to represent the information in $\boldsymbol{b}$ through finite difference formulas. For example, suppose the extrapolation problem is a 2-D problem. The 1st order partial derivative of f with respect to x_1, the 1st order partial derivative of f with respect to x_2 and the 2nd order mixed partial derivative of f with respect to x_1 and x_2 at the corner point $(x_1^{(i_1)}, x_2^{(i_2)})$ are calculated using the values of f at the corner points of the cell and its neighboring cells, i.e.,

$$\begin{aligned}\left.\frac{\partial f}{\partial x_1}\right|_{\left(x_1^{(i_1)},x_2^{(i_2)}\right)} &= \frac{f|_{\left(x_1^{(i_1+1)},x_2^{(i_2)}\right)} - f|_{\left(x_1^{(i_1-1)},x_2^{(i_2)}\right)}}{x_1^{(i_1+1)} - x_1^{(i_1-1)}} \\ &= \left[0 \ \cdots \ 0 \ -\frac{1}{s_1^{(i_1)}+s_1^{(i_1-1)}} \ 0 \ \cdots \ 0 \ \frac{1}{s_1^{(i_1)}+s_1^{(i_1-1)}} \ 0 \ \cdots \ 0\right] \tilde{\boldsymbol{y}}\end{aligned} \tag{5.9a}$$

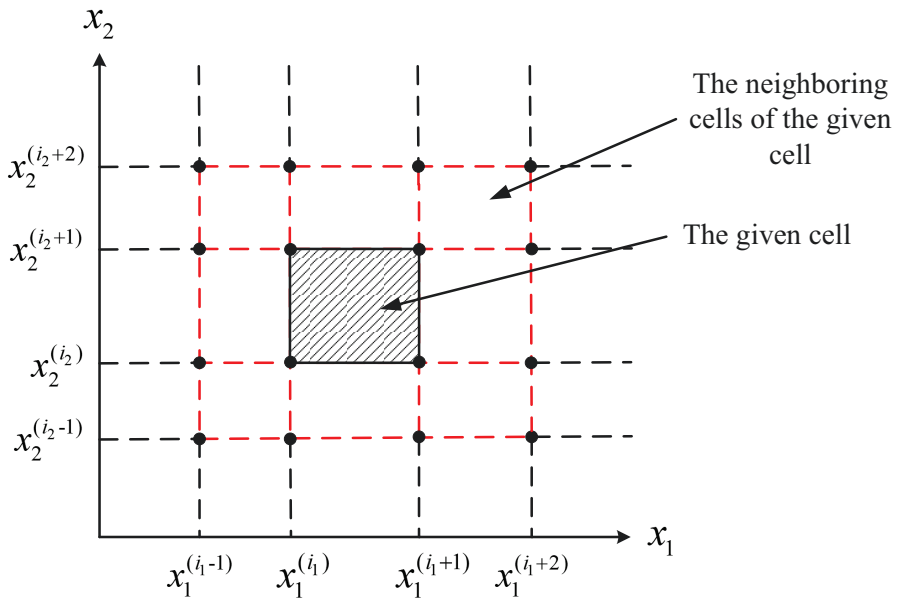


Figure 5.3: The given cell and its neighboring cells for a 2-D example. The shaded area represents the given cell. The 16 grid points indicated by "•" represent the 16 corner points of the given cell and its neighboring cells. The vector $\tilde{\boldsymbol{y}}$ will contain the model output values at these 16 grid points for the 2-D example. x_1 and x_2 can represent geometrical parameters in an EM structure, such as length and width, or voltage/current variables in a nonlinear device, such as gate and drain voltages of a transistor.

$$\begin{aligned}\left.\frac{\partial f}{\partial x_2}\right|_{\left(x_1^{(i_1)},x_2^{(i_2)}\right)} &= \frac{f|_{\left(x_1^{(i_1)},x_2^{(i_2+1)}\right)} - f|_{\left(x_1^{(i_1)},x_2^{(i_2-1)}\right)}}{x_2^{(i_2+1)} - x_2^{(i_2-1)}} \\ &= \left[0 \ \cdots \ 0 \ -\frac{1}{s_2^{(i_2)}+s_2^{(i_2-1)}} \ 0 \ \cdots \ 0 \ \frac{1}{s_2^{(i_2)}+s_2^{(i_2-1)}} \ 0 \ \cdots \ 0\right]\tilde{\boldsymbol{y}}\end{aligned} \tag{5.9b}$$

$$\left.\frac{\partial^2 f}{\partial x_1 \partial x_2}\right|_{\left(x_1^{(i_1)}, x_2^{(i_2)}\right)} = \frac{f|_{\left(x_1^{(i_1+1)}, x_2^{(i_2+1)}\right)} - f|_{\left(x_1^{(i_1-1)}, x_2^{(i_2+1)}\right)} - f|_{\left(x_1^{(i_1+1)}, x_2^{(i_2-1)}\right)} + f|_{\left(x_1^{(i_1-1)}, x_2^{(i_2-1)}\right)}}{\left(x_1^{(i_1+1)} - x_1^{(i_1-1)}\right)\left(x_2^{(i_2+1)} - x_2^{(i_2-1)}\right)}$$

$$= \left[0 \ \cdots \ 0 \ \frac{\left(s_2^{(i_2)} + s_2^{(i_2-1)}\right)^{-1}}{\left(s_1^{(i_1)} + s_1^{(i_1-1)}\right)} \ -\frac{\left(s_2^{(i_2)} + s_2^{(i_2-1)}\right)^{-1}}{\left(s_1^{(i_1)} + s_1^{(i_1-1)}\right)} \ 0 \ \cdots \ 0 \ -\frac{\left(s_2^{(i_2)} + s_2^{(i_2-1)}\right)^{-1}}{\left(s_1^{(i_1)} + s_1^{(i_1-1)}\right)} \ \frac{\left(s_2^{(i_2)} + s_2^{(i_2-1)}\right)^{-1}}{\left(s_1^{(i_1)} + s_1^{(i_1-1)}\right)} \ 0 \ \cdots \ 0\right] \tilde{\boldsymbol{y}} \tag{5.9c}$$

Similar to matrix $\boldsymbol{A}$, we define a new matrix $\boldsymbol{B}$ as a $4^n \times 4^n$ matrix

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_1 \\ \boldsymbol{B}_2 \\ \vdots \\ \boldsymbol{B}_N \end{bmatrix} \tag{5.10}$$

which relates the derivatives of f to the model output values at the corner points of the current cell with the origin $(i_1, i_2, \ldots, i_k, \ldots, i_n)$ and its neighboring cells. The size of each of the submatrices $\boldsymbol{B}_1$, $\boldsymbol{B}_2$, $\cdots$, $\boldsymbol{B}_N$ in $\boldsymbol{B}$ is $N \times 4^n$. These submatrices are defined such that

$$\boldsymbol{B}_1 \tilde{\boldsymbol{y}} = \begin{bmatrix} f|_{p_1} \\ f|_{p_2} \\ \vdots \\ f|_{p_N} \end{bmatrix} \tag{5.11a}$$

and

$$\boldsymbol{B}_l \tilde{\boldsymbol{y}} = \begin{bmatrix} \partial^m f / (\partial x_{u_1} \partial x_{u_2} \cdots \partial x_{u_m})|_{p_1} \\ \partial^m f / (\partial x_{u_1} \partial x_{u_2} \cdots \partial x_{u_m})|_{p_2} \\ \vdots \\ \partial^m f / (\partial x_{u_1} \partial x_{u_2} \cdots \partial x_{u_m})|_{p_N} \end{bmatrix} \tag{5.11b}$$

where $l = 2, 3, \cdots, N$, $m = 1, 2, \cdots, n$, and $u_1, u_2, \cdots, u_m$ are indices whose values

are selected from $u_1, u_2, \cdots, u_m \in \{1, 2, \cdots, n\}$ and $u_1 < u_2 < \cdots < u_m$. The detailed and complete definitions of $\boldsymbol{B}_1$, $\boldsymbol{B}_2$, $\cdots$, $\boldsymbol{B}_N$ are provided in Appendix B. For example, using (5.9a) to (5.9c), the submatrices $\boldsymbol{B}_1$, $\boldsymbol{B}_2$ and $\boldsymbol{B}_{n+2}$ can be derived as

$$\boldsymbol{B}_1 = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \ddots & & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix} \tag{5.12a}$$

$$\boldsymbol{B}_2 = \begin{bmatrix} 0 \cdots 0 & \frac{-1}{s_1^{(i_1)} + s_1^{(i_1-1)}} & 0 & 0 & \cdots & 0 & \frac{1}{s_1^{(i_1)} + s_1^{(i_1-1)}} & 0 & 0 & 0 \cdots 0 \\ 0 \cdots 0 & 0 & \frac{-1}{s_1^{(i_1+1)} + s_1^{(i_1)}} & 0 & \cdots & 0 & 0 & \frac{1}{s_1^{(i_1+1)} + s_1^{(i_1)}} & 0 & 0 \cdots 0 \\ \vdots & \vdots & \vdots & & \ddots & & \vdots & \vdots & & \ddots \quad \vdots \quad \vdots \\ 0 \cdots 0 & 0 & \cdots & 0 & \frac{-1}{s_1^{(i_1+1)} + s_1^{(i_1)}} & 0 & 0 & \cdots & \frac{1}{s_1^{(i_1+1)} + s_1^{(i_1)}} & 0 \cdots 0 \end{bmatrix} \tag{5.12b}$$

$\boldsymbol{B}_{n+2} =$

$$\begin{bmatrix} 0 & \cdots & \frac{\left(s_2^{(i_2)} + s_2^{(i_2-1)}\right)^{-1}}{\left(s_1^{(i_1)} + s_1^{(i_1-1)}\right)} & \frac{-\left(s_2^{(i_2)} + s_2^{(i_2-1)}\right)^{-1}}{\left(s_1^{(i_1)} + s_1^{(i_1-1)}\right)} & 0 & \cdots & 0 & \frac{-\left(s_2^{(i_2)} + s_2^{(i_2-1)}\right)^{-1}}{\left(s_1^{(i_1)} + s_1^{(i_1-1)}\right)} & \frac{\left(s_2^{(i_2)} + s_2^{(i_2-1)}\right)^{-1}}{\left(s_1^{(i_1)} + s_1^{(i_1-1)}\right)} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \frac{\left(s_2^{(i_2)} + s_2^{(i_2-1)}\right)^{-1}}{\left(s_1^{(i_1+1)} + s_1^{(i_1)}\right)} & \frac{-\left(s_2^{(i_2)} + s_2^{(i_2-1)}\right)^{-1}}{\left(s_1^{(i_1+1)} + s_1^{(i_1)}\right)} & \cdots & 0 & \frac{-\left(s_2^{(i_2)} + s_2^{(i_2-1)}\right)^{-1}}{\left(s_1^{(i_1+1)} + s_1^{(i_1)}\right)} & \frac{\left(s_2^{(i_2)} + s_2^{(i_2-1)}\right)^{-1}}{\left(s_1^{(i_1+1)} + s_1^{(i_1)}\right)} & 0 & \cdots & 0 \\ \vdots & & & \ddots & & \ddots & & & \ddots & & \ddots & \vdots \\ 0 & & \cdots & 0 & \frac{\left(s_2^{(i_2+1)} + s_2^{(i_2)}\right)^{-1}}{\left(s_1^{(i_1+1)} + s_1^{(i_1)}\right)} & \frac{-\left(s_2^{(i_2+1)} + s_2^{(i_2)}\right)^{-1}}{\left(s_1^{(i_1+1)} + s_1^{(i_1)}\right)} & 0 & \cdots & 0 & \frac{-\left(s_2^{(i_2+1)} + s_2^{(i_2)}\right)^{-1}}{\left(s_1^{(i_1+1)} + s_1^{(i_1)}\right)} & \frac{\left(s_2^{(i_2+1)} + s_2^{(i_2)}\right)^{-1}}{\left(s_1^{(i_1+1)} + s_1^{(i_1)}\right)} & \cdots & 0 \end{bmatrix} \tag{5.12c}$$

As shown from the examples in (5.12a) to (5.12c), the submatrices $\boldsymbol{B}_1$, $\boldsymbol{B}_2$, $\cdots$, $\boldsymbol{B}_N$ contain only the step information of cells and most of the elements in these

submatrices are 0. Hence, these submatrices are sparse and constant matrices. Therefore $\boldsymbol{B}$ is also a sparse and constant matrix. In this way, we obtain another system of linear equations for the cell with the model output values in $\tilde{\boldsymbol{y}}$, i.e.,

$$\boldsymbol{B}\tilde{\boldsymbol{y}} = \boldsymbol{b} \tag{5.13}$$

Using (5.13) to eliminate $\boldsymbol{b}$ in (5.8), we obtain a new set of linear equations

$$\boldsymbol{A}\boldsymbol{\alpha} = \boldsymbol{B}\tilde{\boldsymbol{y}} \tag{5.14}$$

where the unknown coefficients in $\boldsymbol{\alpha}$ can be solved from the model output values at grid points in $\tilde{\boldsymbol{y}}$ rather than the derivative values in $\boldsymbol{b}$. Therefore, the coefficients in $\boldsymbol{\alpha}$ can be expressed through the model output values at the corner points of the cell with the origin $(i_1, i_2, \ldots, i_k, \ldots, i_n)$ and its neighboring cells using the linear relationship

$$\boldsymbol{\alpha} = (\boldsymbol{A}^{-\mathbf{1}}\boldsymbol{B})\tilde{\boldsymbol{y}} \tag{5.15}$$

In this way, the unknown coefficients $\boldsymbol{\alpha}$ becomes a function of the output values in $\tilde{\boldsymbol{y}}$ which contains the model output values at corner points of a cell and its neighboring cells.

5.2.4 Discussion on Matrix $\boldsymbol{A}^{-1}$

In the computer programming implementation, we can further speed up the computation of (5.15) by taking the step size information out of matrix $\boldsymbol{A}$ and putting it into matrix $\boldsymbol{B}$. In this way, matrix $\boldsymbol{A}$ and matrix $\boldsymbol{A}^{-1}$ become constant matrices and are independent of the modeling problem. For each n, there is a fixed $\boldsymbol{A}^{-1}$ which

can be preprocessed and saved in pre-calculated data base. The implementation details of $(\boldsymbol{A}^{-1}\boldsymbol{B})$ and a mathematical proof are provided in Appendix C.

5.2.5 Optimization to Obtain Extrapolated Values at Grid Points in the Entire Region

The objective of the proposed extrapolation method is to calculate the target values of the model at extrapolated points through optimization and to make the model as smooth as possible in the entire region by training the model with these extrapolated values together with the given training data. We propose to achieve smoothness across different extrapolation directions by minimizing the 2nd order derivatives everywhere in the entire region.

For each cell, the 2nd order derivative of f with respect to the kth model input is derived as

$$\begin{aligned}\frac{\partial^2 f}{\partial {x_k}^2} &= \sum_{j_1=0}^{3}\cdots\sum_{j_k=2}^{3}\cdots\sum_{j_n=0}^{3}\left[\alpha_t\cdot\frac{j_k(j_k-1)}{\left(x_k^{(i_k+1)}-x_k^{(i_k)}\right)^2}\cdot\left(\frac{x_k-x_k^{(i_k)}}{x_k^{(i_k+1)}-x_k^{(i_k)}}\right)^{j_k-2}\cdot\prod_{\substack{l=1\\l\neq k}}^{n}\left(\frac{x_l-x_l^{(i_l)}}{x_l^{(i_l+1)}-x_l^{(i_l)}}\right)^{j_l}\right]\\ &= (\boldsymbol{c}_k)^T\boldsymbol{\alpha}\\ &= (\boldsymbol{c}_k)^T\boldsymbol{A}^{-1}\boldsymbol{B}\tilde{\boldsymbol{y}}\end{aligned} \tag{5.16}$$

We evaluate the 2nd order derivative in (5.16) at each corner point of a cell. In (5.16), $\boldsymbol{c}_k$ is defined as a vector, i.e.,

$$\boldsymbol{c}_k = [c_1, c_2, \cdots, c_t, \cdots, c_{4^n}]^T \tag{5.17}$$

where c_t is a coefficient evaluated according to the location of the corner point of a cell, i.e., according to whether $x_k = x_k^{(i_k)}$ or $x_k = x_k^{(i_k+1)}$, and whether $x_l = x_l^{(i_l)}$ or $x_l = x_l^{(i_l+1)}$ where $l = 1, 2, \cdots, k-1, k+1, \cdots, n$. Specifically, c_t is evaluated as

$$c_t = \begin{cases} 0 & \text{if } x_k = x_k^{(i_k)} \text{ and } j_k > 2 \\ & \text{if } x_l = x_l^{(i_l)} \text{ and } j_l > 0 \\ 0 & \text{for at least one value of } l, \\ & l = 1, 2, \cdots, k-1, k+1, \cdots, n \\ \dfrac{j_k(j_k-1)}{\left(x_k^{(i_k+1)} - x_k^{(i_k)}\right)^2} & \text{otherwise} \end{cases} \tag{5.18}$$

where $j_k = 0, 1, 2, 3$ and t is defined in (5.5). For each corner point of the cell (e.g., for the corner point p_i), the 2nd order derivatives of f and the vector $\boldsymbol{c}_k$ are represented as $\left.\frac{\partial^2 f}{\partial {x_k}^2}\right|_{p_i}$ and $\boldsymbol{c}_k|_{p_i}$, respectively. By using (5.16), the 2nd order derivatives of f at a corner point become a function of the model output values at all the corner points of that cell and its neighboring cells using finite difference method.

In order to achieve smoothness across all directions in the $\boldsymbol{x}$-space, we propose a new optimization with a new error function for each cell, defined as

$$\begin{aligned} e\left(\tilde{\boldsymbol{y}}\right) &= \sum_{k=1}^{n} \sum_{i=1}^{N} \left(\left.\frac{\partial^2 f}{\partial {x_k}^2}\right|_{p_i} \right)^2 \\ &= \tilde{\boldsymbol{y}}^T \sum_{k=1}^{n} \sum_{i=1}^{N} \left[\left(\boldsymbol{A}^{-1}\boldsymbol{B}\right)^T \left(\boldsymbol{c}_k|_{p_i} \right) \left(\boldsymbol{c}_k|_{p_i} \right)^T \left(\boldsymbol{A}^{-1}\boldsymbol{B}\right) \right] \tilde{\boldsymbol{y}} \end{aligned} \tag{5.19}$$

where e is an error function of model output values in $\tilde{\boldsymbol{y}}$. The 2nd order derivatives

of model output with respect to all inputs at all corner points in a cell are considered in e.

As described in Subsection 5.2.1, $\boldsymbol{y}_e$ represents the vector containing the extrapolated target values in the overall extrapolation region. $\boldsymbol{y}_L$ represents the vector containing the output values of given training data in the overall training region. $\tilde{\boldsymbol{y}}$ represents the vector containing the output values at corner points of one cell and its neighboring cells. Therefore, the vector $\boldsymbol{y}_e$ represents global information which means that $\boldsymbol{y}_e$ contains the model output values at all grid points in the extrapolation region. Similarly, the vector $\boldsymbol{y}_L$ also represents global information which means that $\boldsymbol{y}_L$ contains the values of all training data at grid points in the training region. On the other hand, the vector $\tilde{\boldsymbol{y}}$ represents local information which means that the elements in $\tilde{\boldsymbol{y}}$ are the model outputs at the grid points locally around a cell, i.e., the corner points of one cell and its neighboring cells. If a particular corner point is in the extrapolation region, then the corresponding element in $\tilde{\boldsymbol{y}}$ will appear globally as an element in $\boldsymbol{y}_e$. If a particular corner point is in the training region, then the corresponding element in $\tilde{\boldsymbol{y}}$ will appear globally as an element in $\boldsymbol{y}_L$. We define $\boldsymbol{P}_E$ as an incidence matrix to indicate the relationship between elements in $\tilde{\boldsymbol{y}}$ and elements in $\boldsymbol{y}_e$. Similarly, we define $\boldsymbol{P}_L$ as an incidence matrix to indicate the relationship between elements in $\tilde{\boldsymbol{y}}$ and elements in $\boldsymbol{y}_L$. More specifically, $\boldsymbol{P}_E$ is an incidence matrix to relate the local indices of corner points of a cell plus its neighboring cells with the global indices of all grid points in the entire extrapolation region. $\boldsymbol{P}_L$ is an incidence matrix to relate the local indices of corner points of a cell plus its neighboring cells with the global grid indices of all given training data

in the entire training region. In this way, the output values $\tilde{\boldsymbol{y}}$ at corner points of a cell and its neighboring cells can be selected from extrapolated output values in $\boldsymbol{y}_e$ and/or training data outputs in $\boldsymbol{y}_L$ by $\boldsymbol{P}_E$ and $\boldsymbol{P}_L$, i.e.,

$$\tilde{\boldsymbol{y}} = [\boldsymbol{P}_E \ \boldsymbol{P}_L] \begin{bmatrix} \boldsymbol{y}_e \\ \boldsymbol{y}_L \end{bmatrix} \tag{5.20}$$

Substituting (5.20) into (5.19), we obtain

$$\begin{aligned} e(\boldsymbol{y}_e) &= \begin{bmatrix} \boldsymbol{y}_e{}^T & \boldsymbol{y}_L{}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{D}_{11} & \boldsymbol{D}_{12} \\ \boldsymbol{D}_{21} & \boldsymbol{D}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_e \\ \boldsymbol{y}_L \end{bmatrix} \\ &= \boldsymbol{y}_e{}^T \boldsymbol{D}_{11} \boldsymbol{y}_e + \boldsymbol{y}_L{}^T \left(\boldsymbol{D}_{21} + \boldsymbol{D}_{12}^T \right) \boldsymbol{y}_e + \boldsymbol{y}_L{}^T \boldsymbol{D}_{22} \boldsymbol{y}_L \\ &= \boldsymbol{y}_e{}^T \boldsymbol{D}_{11} \boldsymbol{y}_e + 2 \cdot \boldsymbol{y}_L{}^T \boldsymbol{D}_{12}^T \boldsymbol{y}_e + \boldsymbol{y}_L{}^T \boldsymbol{D}_{22} \boldsymbol{y}_L \end{aligned} \tag{5.21}$$

where

$$\begin{aligned} \boldsymbol{D}_{11} &= \sum_{k=1}^{n} \sum_{i=1}^{N} \left(\boldsymbol{A}^{-1} \boldsymbol{B} \boldsymbol{P}_E \right)^T \left(\boldsymbol{c}_k|_{p_i} \right) \left(\boldsymbol{c}_k|_{p_i} \right)^T \left(\boldsymbol{A}^{-1} \boldsymbol{B} \boldsymbol{P}_E \right) \\ &= \boldsymbol{D}_{11}^T \end{aligned} \tag{5.22a}$$

$$\boldsymbol{D}_{12} = \sum_{k=1}^{n} \sum_{i=1}^{N} \left(\boldsymbol{A}^{-1} \boldsymbol{B} \boldsymbol{P}_E \right)^T \left(\boldsymbol{c}_k|_{p_i} \right) \left(\boldsymbol{c}_k|_{p_i} \right)^T \left(\boldsymbol{A}^{-1} \boldsymbol{B} \boldsymbol{P}_L \right) \tag{5.22b}$$

$$\begin{aligned} \boldsymbol{D}_{21} &= \sum_{k=1}^{n} \sum_{i=1}^{N} \left(\boldsymbol{A}^{-1} \boldsymbol{B} \boldsymbol{P}_L \right)^T \left(\boldsymbol{c}_k|_{p_i} \right) \left(\boldsymbol{c}_k|_{p_i} \right)^T \left(\boldsymbol{A}^{-1} \boldsymbol{B} \boldsymbol{P}_E \right) \\ &= \boldsymbol{D}_{12}^T \end{aligned} \tag{5.22c}$$

$$\boldsymbol{D}_{22} = \sum_{k=1}^{n} \sum_{i=1}^{N} \left(\boldsymbol{A}^{-1} \boldsymbol{B} \boldsymbol{P}_L \right)^T \left(\boldsymbol{c}_k|_{p_i} \right) \left(\boldsymbol{c}_k|_{p_i} \right)^T \left(\boldsymbol{A}^{-1} \boldsymbol{B} \boldsymbol{P}_L \right) \tag{5.22d}$$

The index p_i in (5.22a) to (5.22d) means the ith corner point of the cell, and the index k means the index for model input variables. The matrices $\boldsymbol{D}_{11}$, $\boldsymbol{D}_{12}$, $\boldsymbol{D}_{21}$ and $\boldsymbol{D}_{22}$ calculated from (5.22a) to (5.22d) represent the intermediate 2nd order information for a particular cell. As discussed in Subsection 5.2.4, matrix $\boldsymbol{A}^{-1}$ is constant which depends only on dimensionality n and is independent of the modeling problem. For each n, there is a fixed $\boldsymbol{A}^{-1}$ which can be preprocessed and saved in pre-calculated data base. In (5.21), the vector $\boldsymbol{y}_L$ is constant containing known training data. Therefore, $e(\boldsymbol{y}_e)$ is the function of extrapolated output values in $\boldsymbol{y}_e$.

Different cells in the entire region have different extrapolation function f (i.e., different coefficients $\boldsymbol{\alpha}$ in the extrapolation function). Therefore, the values of the error function $e(\boldsymbol{y}_e)$ for different cells are different. Suppose $e_j(\boldsymbol{y}_e)$ is the value of the error function for the jth cell. The error function for the entire region is defined as

$$E(\boldsymbol{y}_e) = \sum_{j=1}^{N_c} e_j(\boldsymbol{y}_e) \tag{5.23}$$

In this way, the 2nd order derivatives with respect to all inputs at all grid points in all cells of the entire region are considered. The optimization problem can be described as

$$\min_{\boldsymbol{y}_e} E(\boldsymbol{y}_e) \tag{5.24}$$

To solve this optimization problem analytically, let

$$\frac{\partial E(\boldsymbol{y}_e)}{\partial \boldsymbol{y}_e} = 0 \tag{5.25}$$

From (5.21) and (5.25), we derive

$$\sum_{j=1}^{N_c}(\boldsymbol{D}_{11})_j\boldsymbol{y}_e + \sum_{j=1}^{N_c}(\boldsymbol{D}_{12})_j\boldsymbol{y}_L = 0 \tag{5.26}$$

where $(\boldsymbol{D}_{11})_j$ and $(\boldsymbol{D}_{12})_j$ represent the $\boldsymbol{D}_{11}$ in (5.22a) and the $\boldsymbol{D}_{12}$ in (5.22b) for the jth cell, $j = 1, 2, \cdots, N_c$. According to (5.22a) and (5.22b), $\boldsymbol{D}_{11}$ and $\boldsymbol{D}_{12}$ for all cells are known, and $\boldsymbol{y}_L$ contains known training data. Only $\boldsymbol{y}_e$ contains unknown extrapolated values. We define a vector $\boldsymbol{q}$ and a matrix $\boldsymbol{Q}$ by

$$\boldsymbol{q} = -\sum_{j=1}^{N_c}(\boldsymbol{D}_{12})_j\boldsymbol{y}_L \tag{5.27a}$$

and

$$\boldsymbol{Q} = \sum_{j=1}^{N_c}(\boldsymbol{D}_{11})_j \tag{5.27b}$$

Now we can derive the final equation for the proposed method by rewriting (5.26) as

$$\boldsymbol{Q}\boldsymbol{y}_e = \boldsymbol{q}. \tag{5.28}$$

Matrix $\boldsymbol{Q}$ is a symmetric and positive-definite matrix. Equation (5.28) is the main system of linear equations to be solved in the proposed method. $\boldsymbol{Q}$ contains the locations and relationships between the grid points and $\boldsymbol{q}$ contains the information of given training data. Both matrix $\boldsymbol{Q}$ and vector $\boldsymbol{q}$ are constant and known, therefore we obtain a linear system of equations with optimization variables $\boldsymbol{y}_e$ as unknowns. Equation (5.28) is the analytical solution of the optimization problem in (5.24), and the target values of the model at extrapolated points in $\boldsymbol{y}_e$ are thus obtained.

Even though the derivation from (5.10) to (5.28) is complex and detailed, the

final equation for solving $\boldsymbol{y}_e$ in (5.28) is surprisingly simple.

We use the Conjugate Gradient (CG) method to solve the system of linear equations in (5.28). The CG method is implemented as an iterative algorithm, applicable to large sparse systems where direct LU decomposition or other direct methods are not effective. After the calculation using the CG method, the target values at the extrapolated points in $\boldsymbol{y}_e$ are obtained and the information outside the range of original training data is thus completely available. We use $\boldsymbol{y}_L$ and $\boldsymbol{y}_e$ together as the training data to train the neural network, where $\boldsymbol{y}_L$ represents the original known data and $\boldsymbol{y}_e$ represents the new extrapolated data. Finally we achieve neural-based models with good accuracy inside the training range and smooth tendency outside the training range.

5.2.6 Discussion

The proposed extrapolation method addresses the specific issues in the iterative computational loops during EM optimization and HB simulation, where the iterative variables may go beyond the boundary of the model. Typical iterative variables are geometrical parameters in EM optimization, and voltage or current variables in the iterative process of HB simulation. Consequently, typical extrapolation variables in our proposed technique are geometrical variables in EM structures, or voltage and current variables in nonlinear device models. Since frequency is usually not an iterative variable in EM optimization or in the iterative process of HB simulation, our technique is not intended to extrapolate over frequency. For extrapolation of frequency variables, other techniques such as rational function [86], [114]-[117] may

be considered.

In existing extrapolation methods for neural models [8], [10]-[12], [15], [98], smooth and linearization formulas are used outside the model training range to maintain the tendency along each extrapolation direction as we move away from the boundaries of the training range. However, these fixed linearization formulas do not have freedom for smoothness across different extrapolation directions. Therefore, we propose to use grid formulation to discretize the extrapolation region in the $\boldsymbol{x}$-space into many grids/cells so that the unknown outputs of the model at all grid points in the extrapolation region have freedom to permit smoothness of the model across different extrapolation directions. Since frequency is not an extrapolation variable, we have selected piecewise polynomial as an intermediate step to relate the model outputs at the grid points with the derivatives of the model. The variables for the piecewise polynomial are the extrapolation variables such as geometrical variables. This is similar to the idea of using finite element methods in 3-D EM simulators [118] where the 3-D space is discretized into meshes, and the E-field solution over the mesh is represented by piecewise polynomial function with geometrical dimensional variables. We also propose a mechanism to force the smoothness of the model across different extrapolation directions. The mechanism is to minimize the 2nd order derivatives along every dimension for every grid point. In this way, the proposed extrapolation method can provide smoothness at every direction, i.e., along the extrapolation direction as we move away from the boundaries of the training region, and also across different extrapolation directions.

For a n-dimensional problem, the size of $\boldsymbol{A}$ in (5.8) is $4^n \times 4^n$. For example, in

a 3-D extrapolation example, the size of $\boldsymbol{A}$ is $4^3 \times 4^3 = 64 \times 64$. Specifically, "64 rows" of $\boldsymbol{A}$ correspond to the grid information relating to the function values and the mixed partial derivatives $\left[f,\ \frac{\partial f}{\partial x_1},\ \frac{\partial f}{\partial x_2},\ \frac{\partial f}{\partial x_3},\ \frac{\partial^2 f}{\partial x_1 \partial x_2},\ \frac{\partial^2 f}{\partial x_1 \partial x_3},\ \frac{\partial^2 f}{\partial x_2 \partial x_3},\ \frac{\partial^3 f}{\partial x_1 \partial x_2 \partial x_3}\right]$ at all corner points (i.e., 8 corner points) of a 3-D cell. "64 columns" of $\boldsymbol{A}$ correspond to the grid information relating to the 64 unknown coefficients in $\boldsymbol{\alpha}$ of the 3-D cubic polynomial formula.

According to (5.22a) and (5.27b), matrix $\boldsymbol{Q}$ is calculated from $\boldsymbol{A}^{-1}$, $\boldsymbol{B}$, $\boldsymbol{P}_E$ and $\boldsymbol{c}_k|_{p_i}$. It is noticed that all these matrices and vectors are sparse, subsequently $\boldsymbol{Q}$ is sparse. As a further explanation, because the elements of the vector $\boldsymbol{c}_k|_{p_i}$ in (5.17) are calculated at the corner points of cells, most of the elements in $\boldsymbol{c}_k|_{p_i}$ are 0. The incidence matrix $\boldsymbol{P}_E$ relates the indices of the extrapolated points in one cell to the indices of the extrapolated points in all cells, therefore $\boldsymbol{P}_E$ is a sparse matrix. The problem-independent constant matrix $\boldsymbol{A}^{-1}$ is also a sparse matrix, and the sparsity of matrix $\boldsymbol{A}^{-1}$ increases with the increase of the dimensionality of the extrapolation problem. For example, the sparsity of $\boldsymbol{A}^{-1}$ for a 5-D (i.e., $n = 5$) extrapolation problem is 90.46%, and for a 6-D (i.e., $n = 6$) extrapolation problem is 92.01%. In addition, matrix $\boldsymbol{B}$ is also a sparse matrix, as explained in Subsection 5.2.3. All these factors contribute to the sparsity of matrix $\boldsymbol{Q}$.

With the increase of dimensionality n of the modeling space and the increase of extrapolated grids for an extrapolation problem, the size of matrix $\boldsymbol{Q}$ in the proposed method increases exponentially. High dimensionality of the extrapolation problem and large number of extrapolated data will result in a large system of linear equations in (5.28). For example, for an 8-D extrapolation problem, if there are 7

extrapolation grids along each dimension, the number of extrapolated grid points will be about 5.7 million (i.e., $N_g = 5700000$) in total. In this case, the memory size for $\boldsymbol{Q}$ is about 19.7 GB. Therefore, the proposed extrapolation method will be limited by the memory size and CPU performance of computers. However, this is intermediate memory and computation expense during model development. Once the model is developed, the final model is a neural model requiring only 20 KB memory. Therefore, the final model is very compact and fast for users to use during microwave design. For $n \geqslant 5$, the $\boldsymbol{Q}$ matrix is usually very large. Therefore, we suggest to use the CG method to solve (5.28). However, if the model has fewer variables, e.g., $n \leqslant 3$, the sparse LU decomposition could be used when solving (5.28).

When it comes to the number of training data to start the proposed extrapolation, at least two training data with different values in each dimension of inputs are required to start the extrapolation mathematically. Practically, we suggest more training data. More data are needed for the region with highly nonlinear model behavior, while fewer data are needed for the region with less nonlinear model behavior.

For typical Learn-by-Example (LBE) methods, such as ANN techniques, support vector machine (SVM) regression [119], and Gaussian process (GP) approach [120], the models work well inside the training range. However, the direct extrapolations by original formulas of these models may be inconsistent with various kinds of tendencies in microwave modeling. For comparison purpose, we use a simple example of drain current I_d vs. gate voltage V_g of a GaAs MESFET [121] to compare the

extrapolation properties between different LBE methods, such as ANN technique, SVM regression and GP approach. The training range for V_g is from -0.8 V to 0.1 V with step = 0.1 V. ANN model, SVM model and GP model are trained with the same training data. After training, we use these models directly to extrapolate V_g from 0.1 V to 2.5 V. Fig. 5.4 (a) illustrates the extrapolation results using direct extrapolation by original ANN formulas, direct extrapolation by original SVM formulas and direct extrapolation by original GP formulas, where the original ANN formulas, original SVM formulas and original GP formulas are all determined by original training data. The extrapolation results using the proposed extrapolation method are also illustrated in Fig. 5.4 (a). From this figure, it can be seen that the direct extrapolation by original ANN, SVM and GP formulas all approach constant I_d values as V_g increases. Such an extrapolation is usually inconsistent with the model tendency at the boundary of the training region. It is also inconsistent with the ideal curve of $I_d = f(V_g)$ in which I_d should increase as V_g increases. Instead, the proposed extrapolation method can maintain the correct tendency at the boundary of the training region. In addition, we also compare the robustness of the ANN, SVM and GP models when they are used in circuit simulation. We illustrate this comparison by using the ANN, SVM and GP models to find the DC bias voltage V_g for the required value of I_d, i.e., $I_d = 0.03$ A. This process is illustrated in Fig. 5.4 (b)-(d). In the figure, we illustrate the iterative process of Newton's method to solve the nonlinear equation $f(V_g) = 0.03$ where $f(V_g)$ is represented by ANN or SVM or GP. With the starting point $V_g = 1$ V outside the training range, the Newton's iterations using the original ANN model and SVM model will diverge, as

shown in Fig. 5.4 (b). The Newton's iterations using the GP model converge to the wrong solution outside the training range and cannot find the correct solution inside the training range, as illustrated in Fig. 5.4(c). Instead, the Newton's iterations using the proposed extrapolation model can converge to the correct solution inside the training range, as illustrated in Fig. 5.4 (d). This is because that the proposed model retains correct tendency of the model outside training range whereas the wrong tendency of the original ANN, SVM and GP models outside training range misleads the Newton's iterations.

Because of the inherent structure of the kernel functions in ANN, SVM and GP models, the direct extrapolations by original formulas of these models may be inconsistent with various kinds of tendencies in microwave modeling. For example, using popular Gaussian function as the kernel function, the SVM model will approach constant values as the variable moves far away from the training boundary. Such a tendency is usually inconsistent with the tendency in microwave behavior modeling. Similar limitations exist in ANN and GP models as we use them outside training range. In order to solve the extrapolation problem, new formulations of extrapolation, such as the proposed formulation to maintain the tendency and smoothness of the model, are needed regardless whether we use ANN, SVM or GP. The choice between different LBE methods (such as ANN, SVM and GP) may affect the accuracy of modeling (under the same formulation of extrapolation). Further information on the comparison of accuracies between ANN, SVM and GP in microwave modeling can be found in the literature such as [122] and [123]. SVM and GP have good generalization capability when training data is limited. ANN is well suited to the

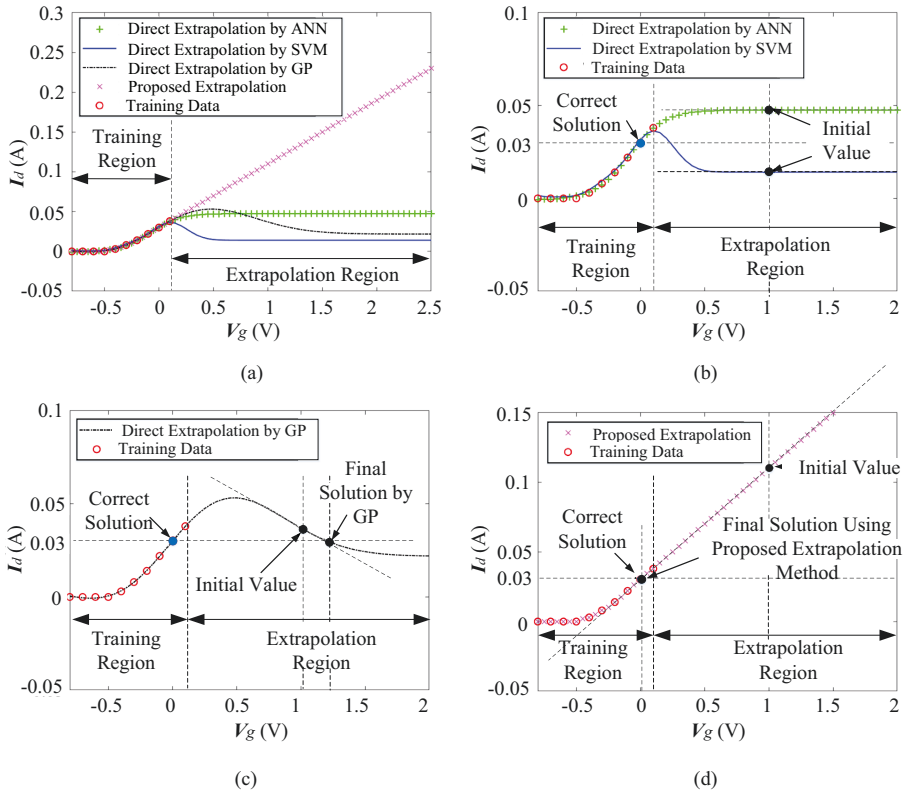


Figure 5.4: (a) The comparison of extrapolation results using different LBE models and the proposed extrapolation method for the simple example of drain current I_d vs. gate voltage V_g of a GaAs MESFET. The different LBE models include ANN, SVM and GP models determined by original training data. (b) The ANN model and SVM model are used to find the DC bias voltage V_g for the required value of I_d, i.e., $I_d = 0.03$ A. (c) The GP model is used to find the DC bias voltage V_g for the required value of I_d, i.e., $I_d = 0.03$ A. (d) The proposed extrapolation model is used to find the DC bias voltage V_g for the required value of I_d, i.e., $I_d = 0.03$ A.

case when the amount of training data is large. In our examples, the training data and extrapolated data are in a large data base. Therefore, ANN has been used in this chapter.

5.2.7 Stepwise Algorithm

The proposed multi-dimensional extrapolation method can guarantee good accuracy wherever training data are given inside the training range and guarantee smoothly extrapolated tendency outside the training range. The proposed extrapolation method can be used for neural-based microwave modeling and design.

The computational steps for implementing the proposed extrapolation technique can be summarized as follows.

Step 1) For the given n, load the problem-independent constant matrix $\boldsymbol{A}^{-1}$ from the pre-calculated data base.

Step 2) Formulate the nonuniform grids in the user-defined region based on the given training data. At some of the grid points, target values of the model are known, i.e., the given training data $\boldsymbol{y}_L$, while at the other grid points, target values of the model are unknown which need to be extrapolated through optimization. The unknown target values at extrapolated points are represented by vector $\boldsymbol{y}_e$, which are optimization variables to be determined by optimization. After formulating the nonuniform grids, the entire region is divided into many cells.

Step 3) Use given training data $\boldsymbol{y}_L$ and the grid information to formulate matrices $\boldsymbol{B}$, $\boldsymbol{P}_E$ and $\boldsymbol{P}_L$ for each cell in the entire region. Formulate vector $\boldsymbol{c}$ for each input at each corner point of each cell, where each input means each k for x_k, $k = 1, 2, \cdots, n$.

Step 4) Calculate $\boldsymbol{D}_{11}$ in (5.22a) for each cell using $\boldsymbol{A}^{-1}$, $\boldsymbol{B}$, $\boldsymbol{P}_E$ and $\boldsymbol{c}$. Calculate $\boldsymbol{D}_{12}$

in (5.22b) for each cell using $\boldsymbol{A}^{-1}$, $\boldsymbol{B}$, $\boldsymbol{P}_E$, $\boldsymbol{P}_L$ and $\boldsymbol{c}$.

Step 5) Formulate the system of linear equations $\boldsymbol{Q}\boldsymbol{y}_e = \boldsymbol{q}$ with optimization variables $\boldsymbol{y}_e$ as unknowns in (5.28). The vector $\boldsymbol{q}$ is calculated from $\boldsymbol{y}_L$ and $\boldsymbol{D}_{12}$ for all cells. The matrix $\boldsymbol{Q}$ is calculated from $\boldsymbol{D}_{11}$ for all cells. This is the main system of linear equations to be solved in the proposed method. Solving this system of linear equations will produce the final solution of the optimization problem in (5.24).

Step 6) Use the CG method to calculate the numerical solution of the sparse linear system in (5.28), and obtain the solutions of $\boldsymbol{y}_e$ which contains the unknown values at the extrapolated grid points.

Step 7) Use both the training data $\boldsymbol{y}_L$ and extrapolated data $\boldsymbol{y}_e$ to train neural networks and finally achieve neural-based models with good accuracy inside the training range and smooth tendency across all directions outside the training range.

Fig. 5.5 shows the flow diagram of the proposed extrapolation method.

5.3 Application Examples

5.3.1 Extrapolation for Neural-Based EM Optimization of a Bandstop Microstrip Filter

This example illustrates that the proposed extrapolation technique can be applied to improve the robustness of the EM-based neural model involved in EM optimization. The neural network models are reusable for design with different specifications [20].

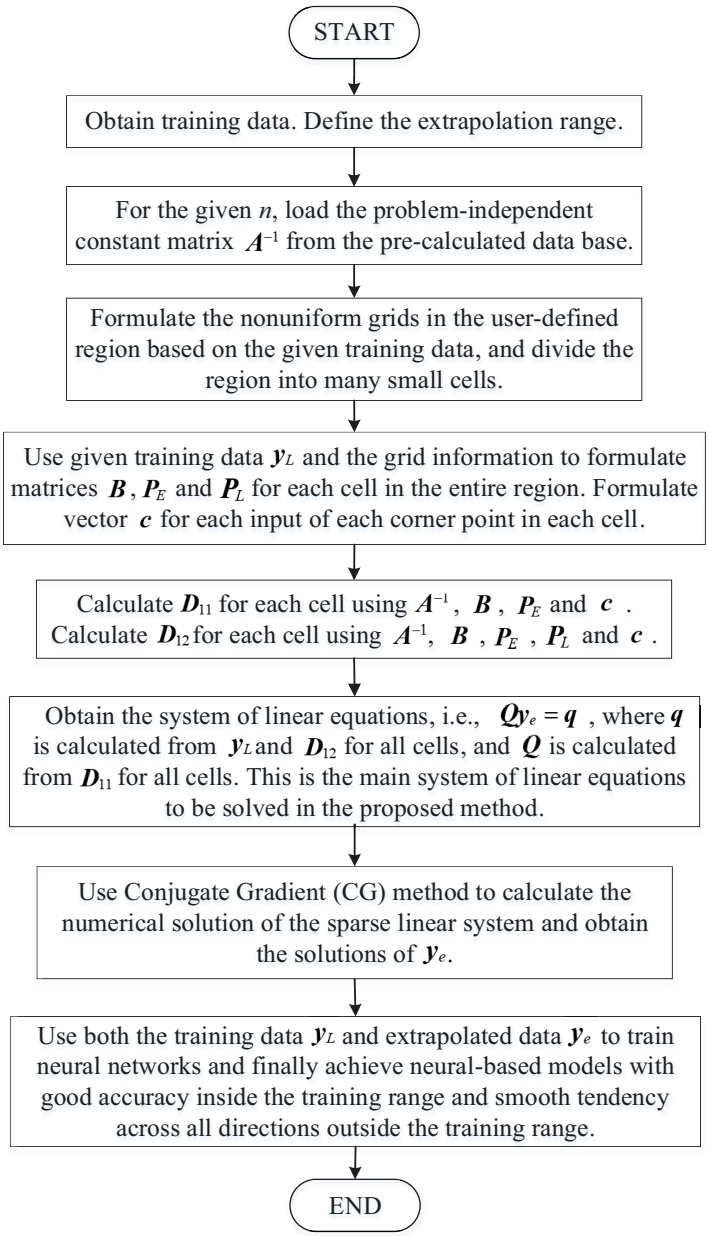


Figure 5.5: Flow diagram of the proposed multi-dimensional neural network extrapolation technique.

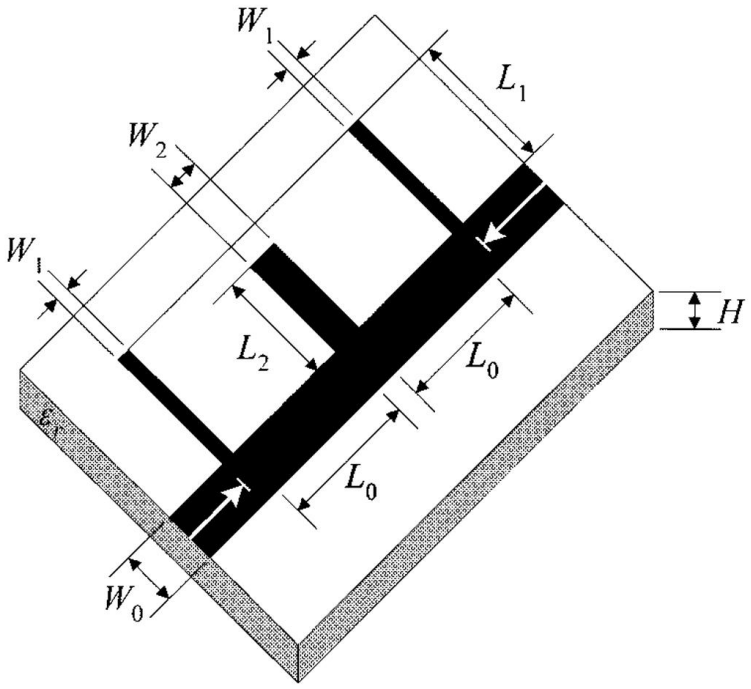


Figure 5.6: Bandstop microstrip filter example. A neural-based parametric model of $|S_{21}|$ versus $W_1,\ W_2,\ L_0,\ L_1$, and L_2 is developed and the model will be extrapolated outside the training range of $W_1,\ W_2,\ L_0,\ L_1$, and L_2.

The use of neural network models for EM optimization can achieve substantial savings in time, especially when numerous EM solutions are required and the neural network models are to be used over and over again in different EM designs [3]. Consider a bandstop microstrip filter with quarter-wave resonant open stubs [15], [40], [98], illustrated in Fig. 5.6. In the figure, L_1 and L_2 are the open stub lengths, and W_1 and W_2 are the open stub widths. L_0 is the interconnecting transmission line length between the two open stubs. W_0 is the width of a 50 Ω feeding line and $W_0 = 0.635$ mm. An alumina substrate with thickness of $H = 0.635$ mm is used and the dielectric constant of the substrate is 9.4.

This example contains two parts. The first part is to develop a neural network parametric model with magnitude of S_{21} as model output y and geometrical variables as model inputs $\boldsymbol{x}$, i.e., $\boldsymbol{x} = [W_1\ W_2\ L_0\ L_1\ L_2]^T$. The neural model is trained by EM data. The second part is to implement the trained parametric model into design optimization where the geometrical parameters can be repetitively adjusted during optimization.

Firstly, we use given training data to develop the parametric model. The geometrical parameters of 25 scattered training data are listed in Table 5.1. The training data are generated by *CST Microwave Studio* EM simulator at 101 frequency points between 5 GHz and 15 GHz. The proposed extrapolation technique is performed to extrapolate the given training data in Table 5.1 and obtain extrapolated model with smooth tendency outside the training region. From the training data and user requirements, the nonuniform grids for extrapolation are determined as listed in Table 5.2. In this example, $m_1 = 9$, $m_2 = 9$, $m_3 = 7$, $m_4 = 7$ and $m_5 = 7$. There are a total of 13824 cells in the entire region. For $n = 5$, the sizes of $\boldsymbol{A}^{-1}$ and $\boldsymbol{B}$ are both 1024×1024. The values of $\boldsymbol{A}^{-1}$ are constants and loaded from the pre-computed problem-independent constant data base. We perform the proposed extrapolation over the grids in the space of all geometrical parameters $W_1,\ W_2,\ L_0,\ L_1$, and L_2. For this example, there are 25 grid points where training data are available, i.e., $N_L = 25$. There are 27758 grid points where training data are not available, i.e., $N_E = 27758$. The vector $\boldsymbol{y}_L$ contains $|S_{21}|$ values at 25 grid points and the vector $\boldsymbol{y}_e$ contains unknown $|S_{21}|$ values at 27758 extrapolated grid points. The symmetric and sparse matrix $\boldsymbol{Q}$ is a 27758×27758 matrix which contains 229342 nonzero

Table 5.1: Training Data for the Bandstop Filter Example

W_1 (mm)	W_2 (mm)	L_0 (mm)	L_1 (mm)	L_2 (mm)
0.20	0.33	2.54	2.80	2.66
0.20	0.35	2.60	2.92	2.86
0.20	0.38	2.66	2.98	2.92
0.20	0.40	2.73	3.05	2.73
0.20	0.43	2.80	2.86	2.80
0.21	0.33	2.60	2.86	2.73
0.21	0.35	2.66	3.05	2.66
0.21	0.38	2.80	2.80	2.86
0.21	0.40	2.54	2.98	2.80
0.21	0.43	2.73	2.92	2.92
0.23	0.33	2.66	2.92	2.80
0.23	0.35	2.80	2.98	2.73
0.23	0.38	2.73	2.86	2.66
0.23	0.40	2.60	2.80	2.92
0.23	0.43	2.54	3.05	2.86
0.24	0.33	2.73	2.98	2.86
0.24	0.35	2.54	2.86	2.92
0.24	0.38	2.60	3.05	2.80
0.24	0.40	2.80	2.92	2.66
0.24	0.43	2.66	2.80	2.73
0.25	0.33	2.80	3.05	2.92
0.25	0.35	2.73	2.80	2.80
0.25	0.38	2.54	2.92	2.73
0.25	0.40	2.66	2.86	2.86
0.25	0.43	2.60	2.98	2.66

elements. The sparsity pattern of the matrix $\boldsymbol{Q}$ is shown in Fig. 5.7. The sparsity of $\boldsymbol{Q}$ is 97.02%, confirming that $\boldsymbol{Q}$ is a large sparse matrix. After solving (5.28), we obtain extrapolated data outside training range. Then we use the extrapolated data $\boldsymbol{y}_e$ together with the given training data $\boldsymbol{y}_L$ to train a neural network.

For comparison purpose, we use sparse LU decomposition and CG methods to

Table 5.2: Nonuniform Grids in Proposed Extrapolation Technique for the Bandstop Filter Example

	W_1 (mm)	W_2 (mm)	L_0 (mm)	L_1 (mm)	L_2 (mm)
Extrapolation Region	0.13				
	0.15	0.28			
	0.18	0.30	2.41	2.66	2.54
Training Region	0.20	0.33	2.54	2.80	2.66
	0.21	0.35	2.60	2.86	2.73
	0.23	0.38	2.66	2.92	2.80
	0.24	0.40	2.73	2.98	2.86
	0.25	0.43	2.80	3.05	2.92
Extrapolation Region	0.28	0.46	2.92	3.17	3.05
		0.48			

solve the large sparse system of linear equations in (5.28) for this example. While it takes about two minutes using the sparse LU decomposition, it takes only 10 seconds by using the CG method. Therefore, the CG method is faster than the sparse LU decomposition for such large sparse system of linear equations.

For another comparison purpose, we also use the same 25 scattered training data to develop a neural model without extrapolation, a neural model with the existing model-level extrapolation method [11] and a neural model with the existing neuron-level extrapolation method [12]. In total, we develop four neural models with or without extrapolation using the same training data, i.e., the 25 samples of given training data.

After developing the parametric models, we use these neural models to do design optimization to find corresponding reasonable W_1, W_2, L_0, L_1 and L_2 for given

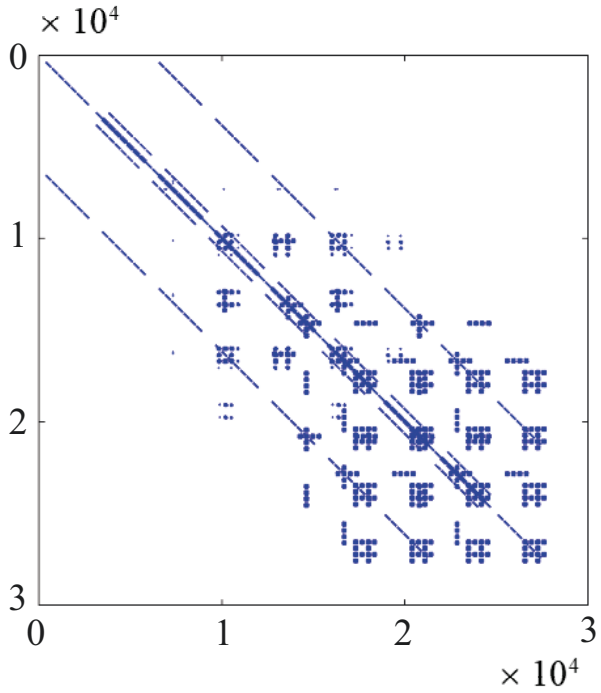


Figure 5.7: Sparsity pattern of the symmetric matrix $\boldsymbol{D}_{11}$ for the bandstop filter example. $\boldsymbol{D}_{11}$ is a 27758×27758 matrix with 229342 nonzero elements. The sparsity of this matrix $\boldsymbol{D}_{11}$ is 97.02%, confirming that $\boldsymbol{D}_{11}$ is a large sparse matrix.

design specifications of the bandstop filter defined as follows [89]

$$
\begin{aligned}
&|S_{21}| \geqslant 0.9, && \text{for 5 GHz} \leqslant \omega \leqslant 8 \text{ GHz} \\
&|S_{21}| \leqslant 0.05, && \text{for 9.35 GHz} \leqslant \omega \leqslant 10.75 \text{ GHz} \\
&|S_{21}| \geqslant 0.9, && \text{for 12 GHz} \leqslant \omega \leqslant 15 \text{ GHz}
\end{aligned}
$$

where ω represents frequency. During optimization process, new values of design variables W_1, W_2, L_0, L_1 and L_2 are produced in each iteration of optimization. As part of the optimization process, the neural models will be evaluated at these new

values of W_1, W_2, L_0, L_1 and L_2, and the response $|S_{21}|$ is obtained to see whether these design values satisfy the design specifications or not. The optimization will determine the updates to W_1, W_2, L_0, L_1 and L_2 for next iteration until the design specifications are satisfied.

We repeat this optimization by using four different neural network models inside optimization iteration. The movement (i.e., change of values) of W_1, W_2, L_0, L_1 and L_2 from iteration to iteration during optimization process forms a trajectory in the geometrical parameter space. With the same initial point, the comparison of the optimization trajectory of design variables W_1 and W_2 using the neural model without extrapolation and the neural models with different extrapolation methods are shown in Fig. 5.8. Although the initial values and the optimal solution are both inside training range, the movement of W_1, W_2, L_0, L_1 and L_2 during optimization often goes beyond the training range. Therefore the quality of the neural model outside the training range is still important and extrapolation methods are needed to help the model find the correct direction for the next change of W_1, W_2, L_0, L_1 and L_2. The neural model without extrapolation is not reliable outside the training range, therefore the optimization is misled to a wrong solution. The optimization results using existing extrapolation techniques can be slightly improved but are still not enough to get the correct solution for this example. Using the proposed extrapolation method, the neural model has smoother tendency outside the training range and the reliable updates for the optimization outside the training range can be found. In this way, the optimization can finally reach the correct solution.

The comparison of the filter responses computed by the EM simulator at the

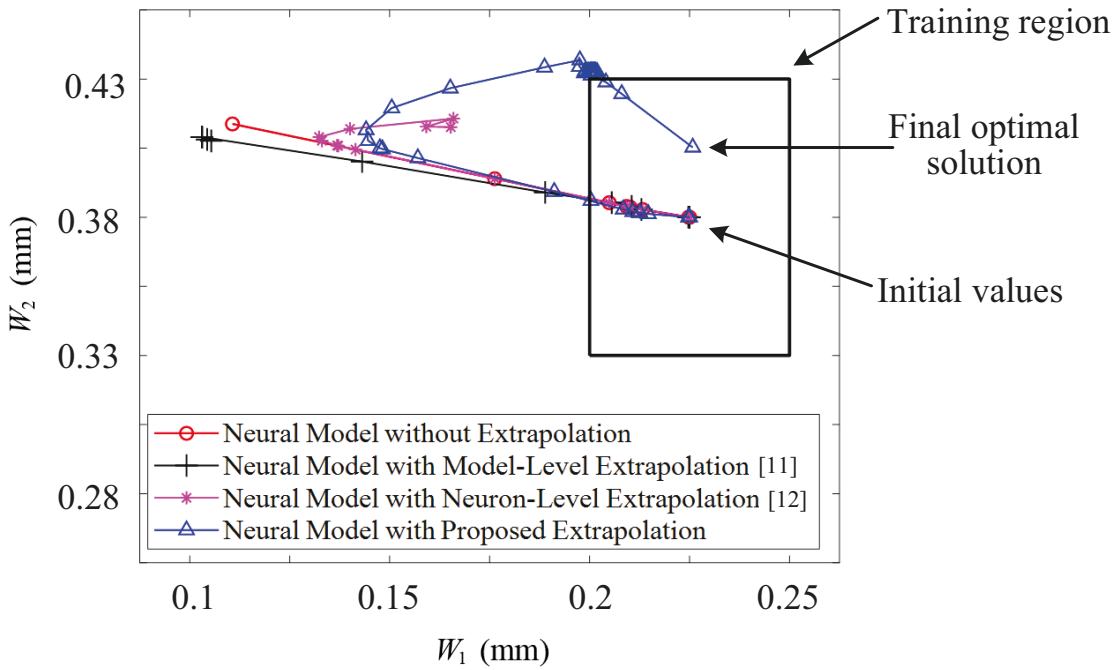


Figure 5.8: Comparison of the optimization trajectory of design variables W_1 and W_2 in the bandstop filter example using the neural model without extrapolation, the neural model with model-level extrapolation, the neural model with neuron-level extrapolation and the neural model with proposed extrapolation with the same initial values.

final solutions of the optimization using the neural model without extrapolation, the neural model with model-level extrapolation, the neural model with neuron-level extrapolation and the neural model with proposed extrapolation are shown in Fig. 5.9. Only the optimization solution using the model with proposed extrapolation can satisfy the design specifications. This example demonstrates that the proposed extrapolation method allows the trained neural models to be used more robustly than existing methods in microwave device optimization.

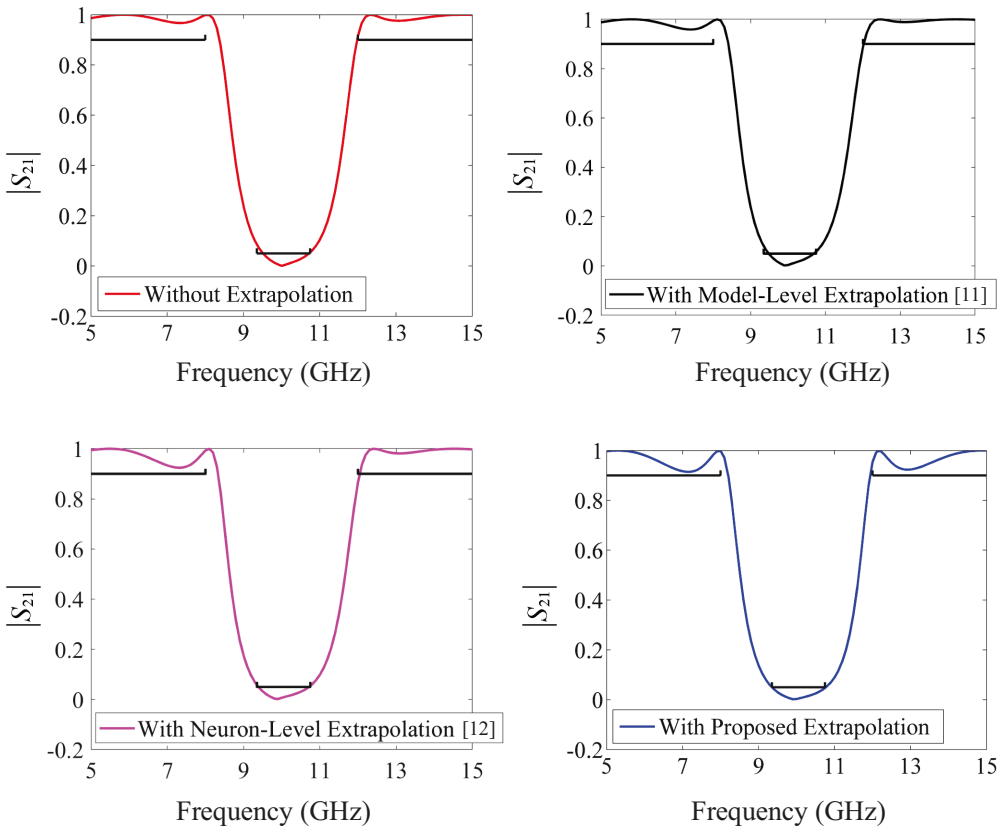


Figure 5.9: Comparison of filter responses computed by *CST* at the final solutions of the optimization using the model without extrapolation, the model with model-level extrapolation, the model with neuron-level extrapolation and the model with proposed extrapolation.

For further comparison, we also use the same training data listed in Table 5.1 to develop models using SVM and GP methods. We compare the extrapolation capability of SVM and GP models with that of the proposed extrapolation method by using these models to do EM optimization again. The optimization results using direct extrapolations by original ANN formulas, direct extrapolation by original SVM formulas, direct extrapolation by original GP formulas and the ANN model

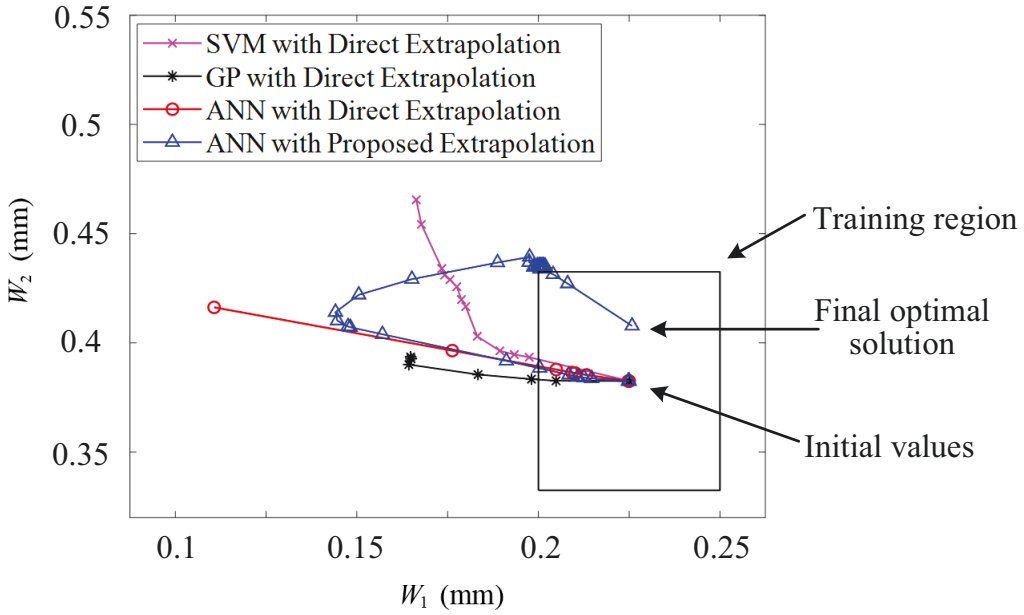


Figure 5.10: Comparison of the optimization trajectory of design variables W_1 and W_2 in the bandstop filter example using different LBE methods and the proposed extrapolation method. The different LBE methods in this comparison include ANN, SVM and GP methods.

with proposed extrapolation method are shown in Fig. 5.10. With direct extrapolation, the optimizations using original ANN, SVM and GP formulas are all misled to wrong solutions. On the other hand, the neural model using the proposed extrapolation method successfully guides the EM optimization to the correct solution as shown in Fig. 5.10.

5.3.2 Extrapolation of Neural-Based Large-Signal Models for HB Simulation of a GaAs MESFET

This example illustrates that the proposed extrapolation technique with TDNN structure [60] can improve the robustness of the neural models involved in HB simulation with nonlinear microwave devices. The TDNN model can achieve good accuracy for dynamic modeling of nonlinear devices which have higher order dynamic effects (e.g., which requires more capacitors) [60]. However, the TDNN model without extrapolation sometimes is not robust enough during HB simulation. To address this problem, new structures of neural model, such as the Wiener-type neural model structure [97], have been presented to improve the convergence properties during HB simulation. In this subsection, we use the proposed extrapolation method to make the original TDNN model more robust and have enhanced convergence properties during HB simulation.

This example has two parts. The first part is to develop TDNN models for the GaAs MESFET [121], and the second part is to use these TDNN models to do HB simulation using *Keysight Advanced Design System (ADS)* simulator.

In the first part, the structures of TDNN models are

$$i_g(t) = f_{ANN1}\Big(v_g(t), v_g(t-\tau), v_g(t-2\tau), v_d(t), v_d(t-\tau), v_d(t-2\tau)\Big) \tag{5.29a}$$

$$i_d(t) = f_{ANN2}\Big(v_g(t), v_g(t-\tau), v_g(t-2\tau), v_d(t), v_d(t-\tau), v_d(t-2\tau)\Big) \tag{5.29b}$$

where $i_g(t)$, $i_d(t)$, $v_g(t)$ and $v_d(t)$ represent the time-domain gate current, drain current, gate voltage and drain voltage, respectively. The time delay parameter τ is

0.0045 ns. The large-signal training data are generated in *ADS* simulator by exciting the GaAs MESFET with a set of fundamental frequencies (from 1 GHz to 5 GHz, step-size 1 GHz) and input power levels (from −5 dBm to 10 dBm, step-size 1 dBm) at one bias point ($V_g = -0.2$ V, $V_d = 3$ V). Even though the frequencies and power levels are grid distributed, they are not directly the TDNN input variables. The actual TDNN inputs are $\boldsymbol{x} = [v_g(t), v_g(t-\tau), v_g(t-2\tau), v_d(t), v_d(t-\tau), v_d(t-2\tau)]^T$, resulting in nongrid scattered data with irregular boundaries of the training region as shown in Fig. 5.11.

For comparison purpose, we develop TDNN models using different extrapolation methods, i.e., the standard TDNN without extrapolation, the TDNN with model-level extrapolation [11], the TDNN with neuron-level extrapolation [12] and the TDNN with proposed extrapolation. Fig. 5.11 and Table 5.3 show the given time-domain training data and nonuniform grids following the proposed extrapolation method. For this example, $m_1 = m_2 = m_3 = m_4 = m_5 = m_6 = 6$. There are a total of 15625 cells in the entire region. For $n = 6$, the sizes of $\boldsymbol{A}^{-1}$ and $\boldsymbol{B}$ are both 4096×4096. The values of $\boldsymbol{A}^{-1}$ are constants and loaded from the pre-computed problem-independent constant data base. In this example, linear transformation is used to preprocess the training data such that the grid orientations in the training region are better aligned with that of the training data.

After developing TDNNs without extrapolation and TDNNs with different kinds of extrapolation methods, we perform HB simulation using *ADS* simulator at 15 input power levels (from −4.5 dBm to 9.5 dBm, step-size 1 dBm) and four fundamental frequencies (from 1.5 GHz to 4.5 GHz, step-size 1 GHz), resulting in 60

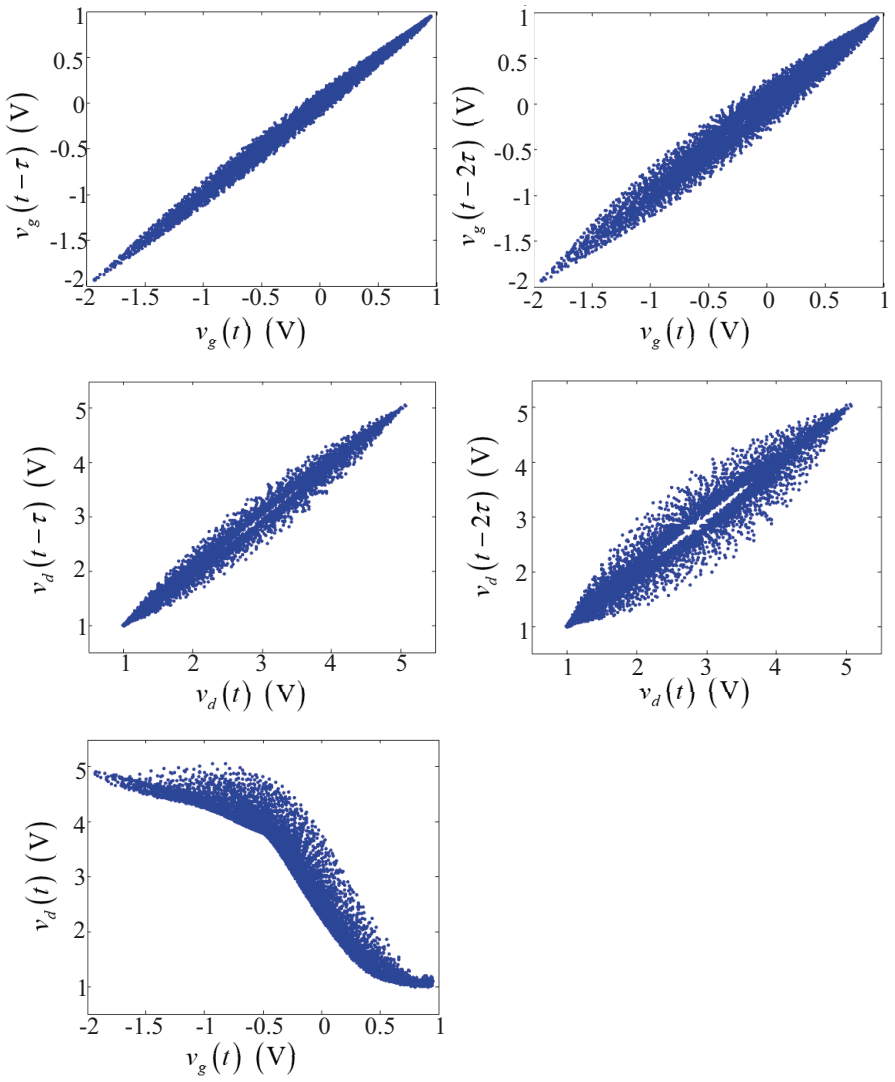


Figure 5.11: Training range with irregular boundaries in the input space of the TDNNs of the MESFET example.

combinations of different input power levels and fundamental frequencies. These 60 combinations are different from those in training data generation. For conve-

Table 5.3: Nonuniform Grids in Proposed Extrapolation Technique for the MESFET Example

$v_g(t)$ (V)	$v_g(t-\tau)$ (V)	$v_g(t-2\tau)$ (V)	$v_d(t)$ (V)	$v_d(t-\tau)$ (V)	$v_d(t-2\tau)$ (V)
-3	-3	-3	0	0	0
-1.937	-1.938	-1.936	1.097	1.080	1.063
-0.707	-0.686	-0.654	2.994	2.553	2.145
-0.145	-0.028	0.082	4.000	4.010	4.013
0.947	0.944	0.936	4.903	4.896	4.887
2	2	2	6	6	6

nience of description, we call these 60 combinations as 60 points of HB simulations. The HB simulation is an iterative process to solve nonlinear circuit equations with node voltages as variables. Examples of such node voltages are gate voltages and drain voltages of the MESFET. During HB simulation process, new values of $v_g(t), v_g(t-\tau), v_g(t-2\tau), v_d(t), v_d(t-\tau)$ and $v_d(t-2\tau)$ are updated in each iteration and the TDNN models are evaluated at these new values. This iterative process continues from iteration to iteration until HB simulation converges. In this process, it is often possible that the values of $v_g(t), v_g(t-\tau), v_g(t-2\tau), v_d(t), v_d(t-\tau)$ and $v_d(t-2\tau)$ go beyond the TDNN training range. Therefore, extrapolation is necessary to be incorporated for the HB simulation.

The convergence properties of HB simulation using the TDNN models without extrapolation or with different kinds of extrapolation in HB simulation are shown in Fig. 5.12. Three available solvers in *ADS* for HB simulation, i.e., "Auto", "Direct" and "Krylov" were tried. We observed that the best convergence results for this example were obtained by the "Krylov" solver in *ADS*. The HB simulation

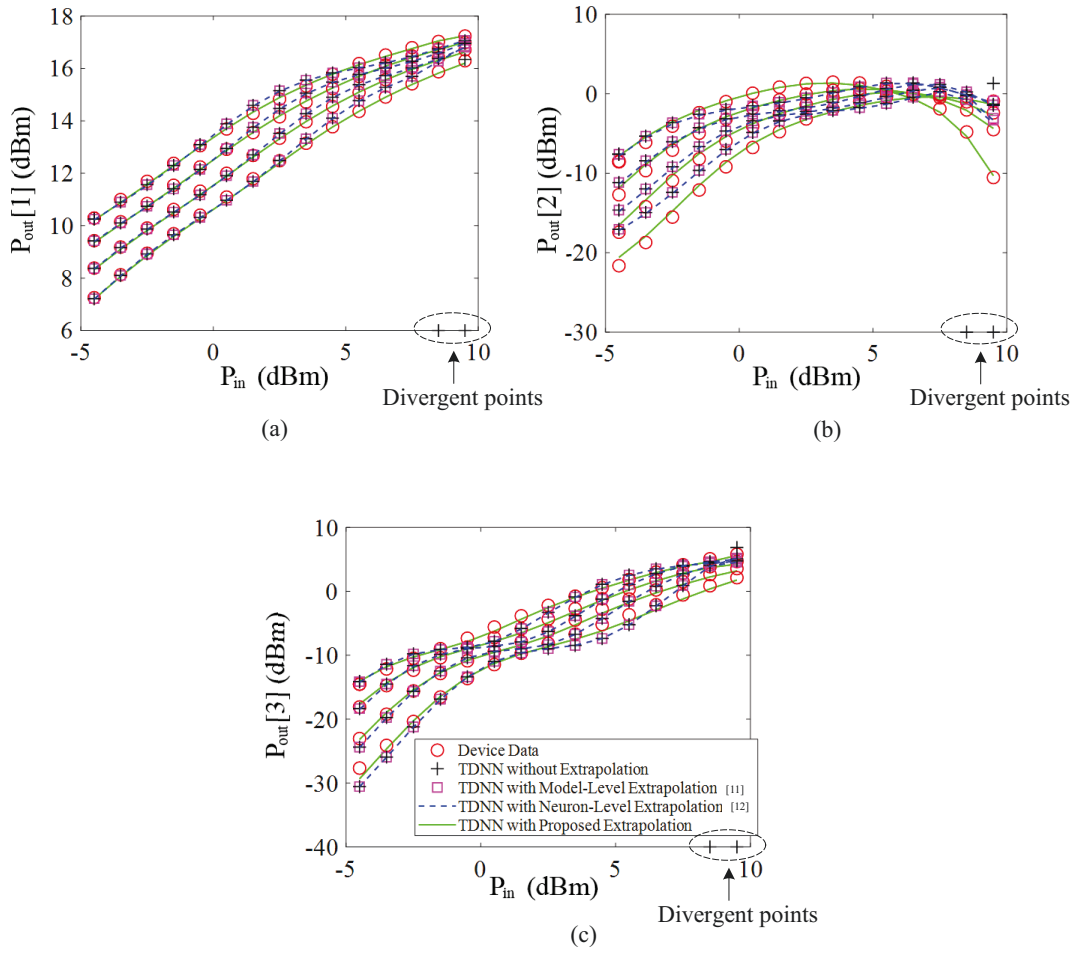


Figure 5.12: Comparisons between *ADS* HB responses of the TDNN without extrapolation, the TDNN with model-level extrapolation, the TDNN with neuron-level extrapolation, the TDNN with proposed extrapolation and the device data (*ADS* solution) at 15 input power levels and 4 fundamental frequency points in the MESFET example. (a) The output power at the fundamental frequency. (b) The 2nd harmonic component of the output power. (c) The 3rd harmonic component of the output power.

Table 5.4: Convergence Comparison of Different Extrapolation Approaches for HB Simulation of the MESFET Example

Extrapolation approach	Total no. of HB simulation points	No. of points converged to correct solutions	No. of points converged to wrong solutions	No. of points non-converged
Without extrapolation	60	42	16	2
Model-level extrapolation [11]	60	42	18	0
Neuron-level extrapolation [12]	60	42	18	0
Proposed extrapolation	60	60	0	0

results reported in Fig. 5.12 are obtained by the "Krylov" solver in *ADS* for HB simulation. When the input power level is low, HB simulations using TDNNs with or without extrapolation can converge and converge to the correct solutions. As the input power becomes higher, using the standard TDNN without extrapolation, some points of HB simulation converge while the remaining points of HB simulation diverge. The TDNN with model-level extrapolation and the TDNN with neuron-level extrapolation have improved convergence properties however the converged solutions are sometimes wrong. The number of HB simulation points that converge to correct solutions, the number of HB simulation points that converge to wrong solutions and the number of HB simulation points that do not converge are listed in Table 5.4 using different extrapolation approaches. Out of the 60 points of HB simulation using the TDNN without extrapolation, 42 points converge to correct

solutions, 16 points converge to wrong solutions, and two points diverge during the HB simulation. For the TDNN with model-level extrapolation and the TDNN with neuron-level extrapolation, all 60 points can converge, 42 of which converge to correct solutions and 18 of which converge to wrong solutions. For the TDNN with proposed extrapolation, the HB simulation not only converge but also converge to correct solutions at all 60 points. We can see that our proposed extrapolation method is more robust, and the TDNN with proposed extrapolation has better convergence property than the TDNN without extrapolation and the TDNN with existing extrapolation techniques during HB simulation.

Fig. 5.13 shows more details about the convergence accuracy during the HB simulation and further demonstrates the robustness of the proposed extrapolation technique. The inputs for TDNN model are gate and drain voltages at several time samples. We compare the curves of time-domain voltages of four TDNNs with or without extrapolation during HB simulation when input power level is 9.5 dBm and fundamental frequency is 3.5 GHz. We see that only the HB simulation using the TDNN with proposed extrapolation converges to the correct solution, where the curves of the model match those of data. These results demonstrate that compared to the existing extrapolation methods, the proposed extrapolation technique is more robust and has better performance when the TDNN is used outside the training region. We also compares the efficiency between different extrapolation methods for the 60 points of HB simulations, as shown in Table 5.5. The HB simulation using the TDNN with proposed extrapolation is faster than that using the TDNN with existing extrapolation methods. This is because the analytical formulations of

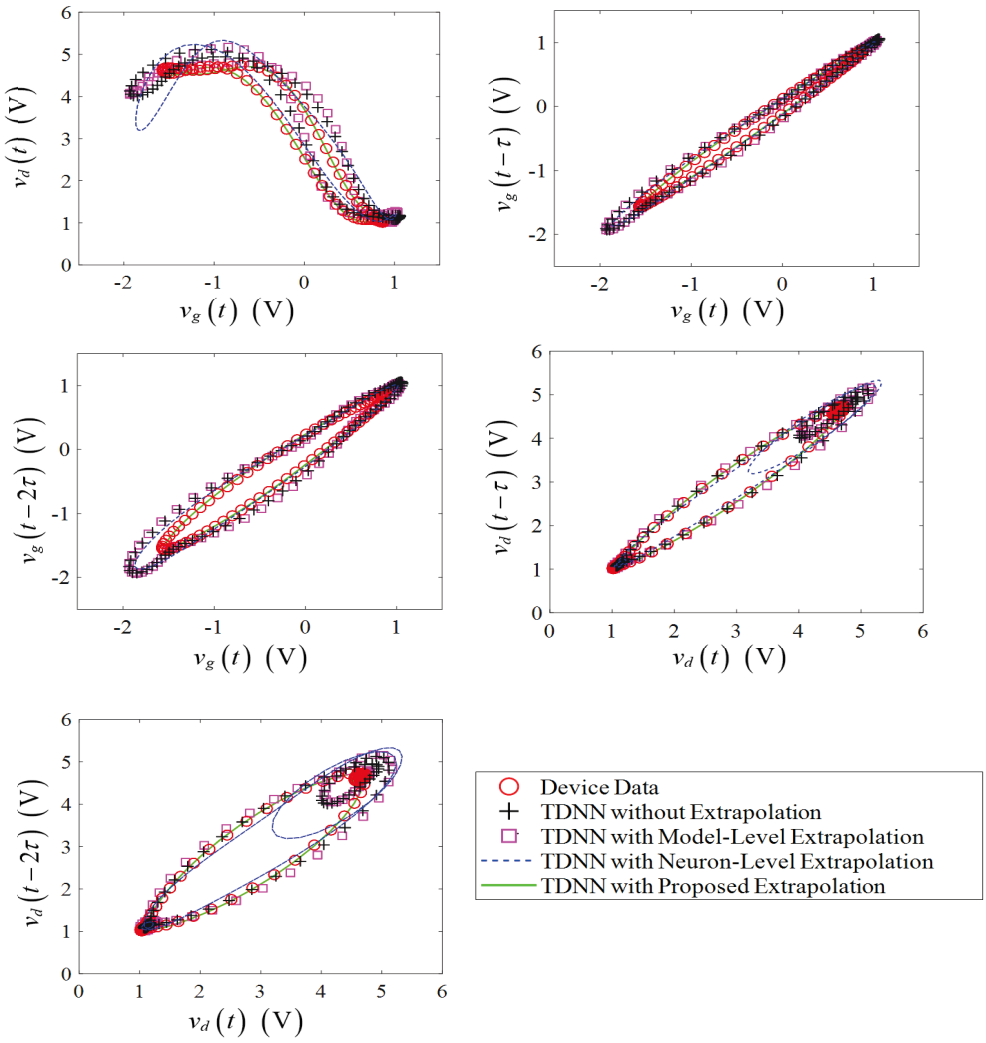


Figure 5.13: Comparisons between the curves of time-domain inputs of various TDNN models during HB simulation. The various TDNN models used in this comparison include the TDNN without extrapolation, the TDNN with model-level extrapolation, the TDNN with neuron-level extrapolation, the TDNN with proposed extrapolation and the device data (*ADS* solution) when input power level is 9.5 dBm and fundamental frequency is 3.5 GHz.

Table 5.5: Comparison of Speed between Different Extrapolation Approaches for HB Simulation of the MESFET Example

Extrapolation approach	CPU time for 60 points of HB simulations
Without extrapolation	6 s
Model-level extrapolation [11]	181 s
Neuron-level extrapolation [12]	34 s
Proposed extrapolation	2 s

the proposed extrapolation method avoid the required derivative evaluation by the model-level extrapolation and the smoother tendency outside the training range of the TDNN model with proposed extrapolation makes the HB simulation converge faster and more robustly. This example demonstrates that the TDNN with proposed extrapolation maintains smoother and better tendency than the standard TDNN without extrapolation, the TDNN with model-level extrapolation and the TDNN with neuron-level extrapolation when used beyond training region. The HB simulation results reported in Table 5.4, Table 5.5 and Fig. 5.13 are obtained by the "Krylov" solver in *ADS* for HB simulation.

5.3.3 Extrapolation of Neural-Based Large-Signal Models for Harmonic Balance (HB) Simulation of a GaAs HEMT

In this example, we apply the proposed extrapolation technique to TDNNs involved in Harmonic Balance (HB) simulation for a GaAs HEMT. We first develop TDNN models, then use these TDNNs to do HB simulation in *ADS* simulator.

For this example, the large-signal training data are generated from a five-layer

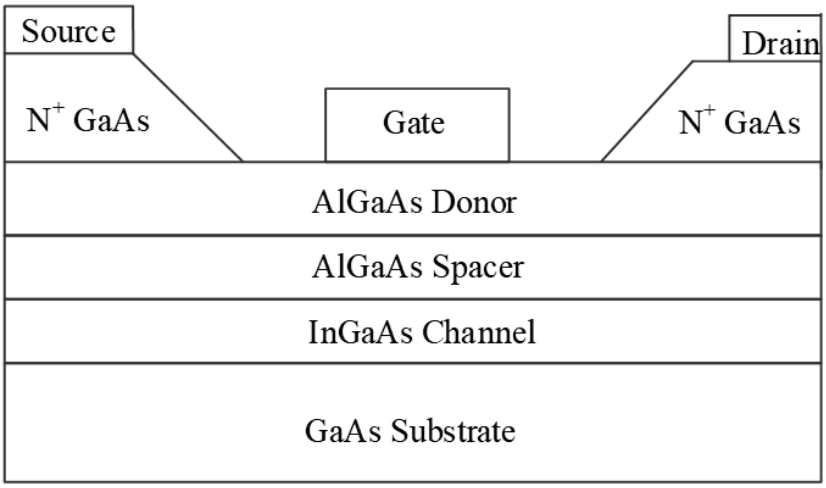


Figure 5.14: HEMT structure in Medici simulator used for data generation.

GaAs-AlGaAs-InGaAs HEMT example given in a physics-based device simulator *Medici*. The HEMT [37], [124] structure used in setting up the physical-based simulator is shown in Fig. 5.14. The parameters of HEMT are shown in Table 5.6. The HB data used for large-signal training are generated at different input power levels (from -20 dBm to -5 dBm, step-size 5 dBm; from -3 dBm to 9 dBm, step-size 2 dBm; 10 dBm) and different fundamental frequencies (from 2 GHz to 5 GHz, step-size 1 GHz) at the static bias point ($V_g = 0.2$ V, $V_d = 5$ V). The structures of TDNNs with the time delay parameter $\tau = 0.0045$ ns are same as (5.29a) and (5.29b).

For comparison purpose, we develop the standard TDNN without extrapolation, the TDNN with model-level extrapolation, the TDNN with neuron-level extrapola-

Table 5.6: Values of Geometrical/Physical Parameters for the HEMT Device

Parameter Name		Value
Gate Length (um)		0.2
Gate Width (um)		100
Thickness (um)	AlGaAs Donor Layer	0.025
	AlGaAs Spacer Layer	0.01
	InGaAs Channel Layer	0.01
	GaAs Substrate	0.045
Doping Density ($1/cm^3$)	AlGaAs Donor Layer	1e18
	InGaAs Channel Layer	1e2
	Source N^+	2e20
	Drain N^+	2e20

Table 5.7: Nonuniform Grids in Proposed Extrapolation Technique for the HEMT Example

$v_g(t)$ (V)	$v_g(t-\tau)$ (V)	$v_g(t-2\tau)$ (V)	$v_d(t)$ (V)	$v_d(t-\tau)$ (V)	$v_d(t-2\tau)$ (V)
-2.5	-2.5	-2.5	1	1	1
-1.589	-1.581	-1.564	1.850	1.886	1.960
-0.500	-0.498	-0.490	4.597	5.046	5.461
0.495	0.423	0.349	6.653	6.651	6.652
1.535	1.534	1.532	7.563	7.569	7.568
2.5	2.5	2.5	9	9	9

tion and the TDNN with proposed extrapolation to learn the large-signal behavior of the the HEMT. Fig. 5.15 shows the given time-domain training data and Table 5.7 indicates the nonuniform grids following the proposed extrapolation method.

For this example, $m_1 = m_2 = m_3 = m_4 = m_5 = m_6 = 6$. There are a total of 15625 cells in the entire region. For $n = 6$, the sizes of $\boldsymbol{A}^{-1}$ and $\boldsymbol{B}$ are both 4096×4096. The values of $\boldsymbol{A}^{-1}$ are constants and loaded from the pre-calculated problem-independent constant data base. In this example, linear transformation is

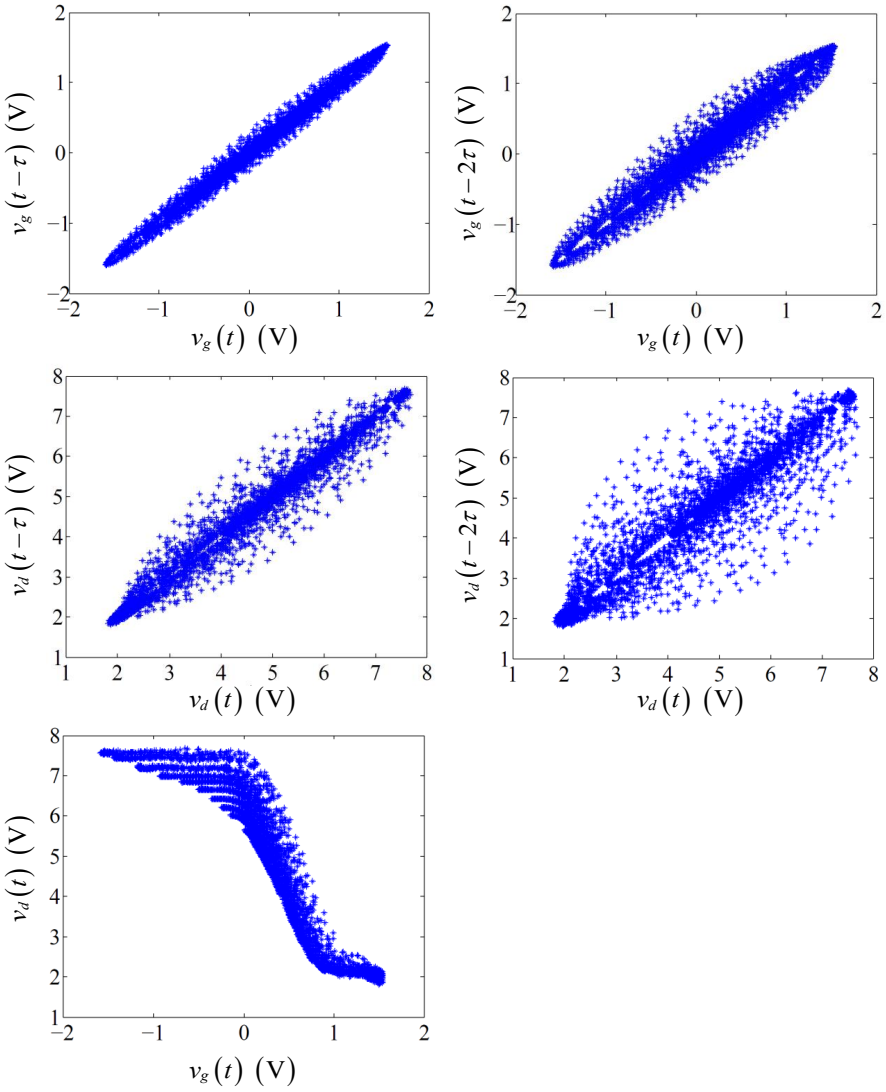


Figure 5.15: Training range with irregular boundaries in the input space of the TDNNs of the HEMT example.

used to preprocess the training data such that the grid orientations in the training region are better aligned with that of the training data.

After developing TDNNs with or without extrapolation, we compare the convergence properties between the standard TDNN without extrapolation, the TDNN with model-level extrapolation [11], the TDNN with neuron-level extrapolation [12] and the TDNN with proposed extrapolation. We perform HB simulation using *ADS* simulator at 12 input power levels from −20 dBm to 10 dBm and three fundamental frequencies from 2.5 GHz to 4.5 GHz, resulting in 36 combinations of different input power levels and fundamental frequencies. For convenience of description, we call these 36 combinations as 36 points of HB simulations. The comparison results are shown in Fig. 5.16. For the standard TDNN without extrapolation, some points of HB simulation converge, while the remaining points of HB simulation diverge when the input power is high. For the TDNN with model-level extrapolation, the TDNN with neuron-level extrapolation and the TDNN with proposed extrapolation, the HB simulations converge with all input power levels at all fundamental frequencies. However, the HB solutions may not be necessarily correct, even if the HB simulation converges. The number of HB simulation points that converge to correct solutions, the number of HB simulation points that converge to wrong solutions and the number of HB simulation points that do not converge are listed in Table 5.8 using different extrapolation approaches. For the TDNN without extrapolation, 30 points converge to correct solutions, three points converge to wrong solutions, and three points diverge during the HB simulation. For the TDNN with model-level or neuron-level extrapolation, all 36 points can converge, 30 of which converge to correct solutions and six of which converge to wrong solutions. For the TDNN with proposed extrapolation, the HB simulation not only converge but also converge to

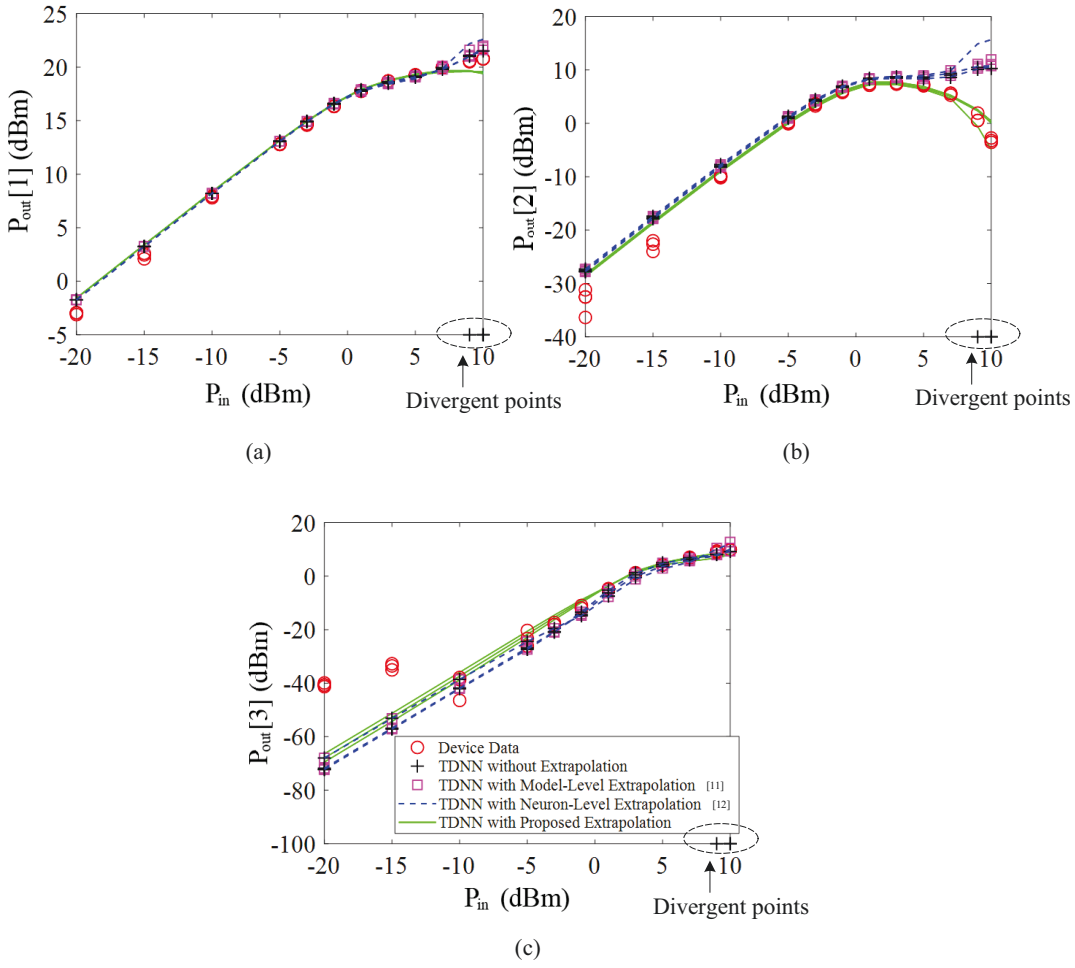


Figure 5.16: Comparisons between *ADS* HB responses of the TDNN without extrapolation, the TDNN with model-level extrapolation, the TDNN with neuron-level extrapolation, the TDNN with proposed extrapolation and the device data (*ADS* solution) at 12 input power levels and 3 fundamental frequency points in the HEMT example. (a) The output power at the fundamental frequency. (b) The 2nd harmonic component of the output power. (c) The 3rd harmonic component of the output power.

Table 5.8: Convergence Comparison of Different Extrapolation Approaches for HB Simulation of the HEMT Example

Extrapolation approach	Total no. of HB simulation points	No. of points converged to correct solutions	No. of points converged to wrong solutions	No. of points non-converged
Without extrapolation	36	30	3	3
Model-level extrapolation [11]	36	30	6	0
Neuron-level extrapolation [12]	36	30	6	0
Proposed extrapolation	36	36	0	0

Table 5.9: Comparison of Speed between Different Extrapolation Approaches for HB Simulation of the HEMT Example

Extrapolation approach	CPU time for 36 points of HB simulations
Without extrapolation	18 s
Model-level extrapolation [11]	389 s
Neuron-level extrapolation [12]	86 s
Proposed extrapolation	2 s

correct solutions at all 36 points. We also compares the efficiency between different extrapolation methods for the 36 points of HB simulations, as shown in Table 5.9. Compared to TDNN models with existing extrapolation methods, the TDNN with proposed extrapolation has smoother tendency outside the training range. The 36 points of HB simulations using the TDNN with proposed extrapolation are faster and more efficient than those using TDNN models with the existing extrapolation

methods. The HB simulation results reported in Fig. 5.16, Table 5.8 and Table 5.9 are obtained by the "Krylov" solver in *ADS* for HB simulation.

These comparisons demonstrate that the TDNN with proposed extrapolation technique has smoother tendency outside the training range which leads to better convergence properties and faster convergence speed in HB simulation than the TDNN without extrapolation and the TDNN with existing extrapolation methods. The proposed extrapolation technique outperforms the existing techniques with improved reliability and faster convergence in HB simulation.

5.4 Conclusion

In this chapter, an advanced multi-dimensional extrapolation technique for neural-based microwave modeling and design has been proposed to address the modeling challenges in microwave simulation and optimization, such as EM optimization and large-signal HB simulation. Grid formulation is used to discritize the input space into many grids so that the unknown outputs of the model at all grid points in the extrapolation region have freedom to permit smoothness of the model across different extrapolation directions. The proposed method preserves 1st order continuity of model outputs versus model inputs across the boundary of the training region and minimizes 2nd order derivatives everywhere in the extrapolation region. The formulation of the proposed method guarantees the best model accuracy where training data are given and the maximum smoothness where training data are not given. The proposed extrapolation technique makes neural models for both passive and active components more robust, resulting in faster convergence in microwave

simulation and optimization involving neural model inputs as iterative variables. Compared to existing extrapolation methods for neural networks, the proposed extrapolation technique has better performance when a neural model is used outside the training range.

Chapter 6

Conclusions and Future Research

6.1 Conclusions

This thesis has presented systematic research about automated knowledge-based neural model generation and extrapolation techniques for microwave applications. An advanced algorithm to automate the process of developing knowledge-based models with a new adaptive sampling technique has been proposed. A unified automated model structure adaptation algorithm for knowledge-based modeling using l_1 optimization has been introduced. An advanced multi-dimensional extrapolation technique for neural-based microwave modeling and design has also been proposed.

In the thesis, we have firstly proposed automated knowledge-based model generation algorithm with a new adaptive sampling technique for microwave applications. It is the first time that the subtasks in generating a knowledge-based model like initial optimization, unit mapping, data generation, determination of the number of hidden neurons, training and testing are integrated into one unified framework. In the proposed method, a new adaptive sampling technique has been applied in the

knowledge-based model process to make the modeling much faster. The proposed method uses relatively fewer training samples than traditional manual knowledge-based modeling techniques to achieve similar model accuracy, and significantly reduces the human time.

A unified automated model structure adaptation algorithm for knowledge-based neural network using l_1 optimization has been proposed to automatically determine the type and topology of the mapping structure in a knowledge-based model. A unified model structure encompassing various types of mappings has been proposed. New formulations using l_1 optimization has also been proposed to automatically determine whether input mapping, frequency mapping and output mapping in a knowledge-based model is needed or not. Compared to existing knowledge-based modeling method, the proposed method can automatically adjust the mapping structure to achieve a model with the good accuracy as well as the most compact structure. The proposed algorithm is a systematic technique which can increase the knowledge-based modeling efficiency.

This thesis has also proposed an advanced extrapolation technique for neural-based microwave modeling and design. The proposed technique aims to address the issue that a neural model is accurate only within the training range and is unreliable outside the training range. In the proposed technique, grid formulation has been used to discritize the input space into many grids and the extrapolation has been performed over these grids. The unknown target values of the model at these grid points are formulated as optimization variables and are determined by optimization. In this way, the proposed extrapolation method can guarantee good

model accuracy inside the training range and maximally smooth across all directions everywhere outside the training range. Compared to existing extrapolation methods for neural models, The proposed extrapolation method makes neural models have better performance when they are used outside the training range.

6.2 Future Research

Some of the future directions based on the expansion of the proposed unified automated model generation method and the proposed extrapolation technique have been identified in this thesis as follows

- One of the future directions is to incorporate the proposed extrapolation technique into automated knowledge-based modeling process. Extrapolation can be applied to every space mapping neural network of a knowledge-based model to make the final knowledge-based model more reliable when used outside the training range. Currently, the proposed extrapolation is performed based on the measurement/simulation data of the device under consideration (i.e., the data for the input-output values of the entire model). Since in reality the mapping data (i.e., the data for the input-output values of each mapping neural network) are unknown, how to obtain these mapping data and extrapolate the mapping neural network in a knowledge-based model is not clear and straightforward. New extrapolation formulations and methodology will be required to address this issue. The entire knowledge-based modeling process with mapping extrapolation will still be an automated process.

- Another future direction is to add parallel computation into the proposed extrapolation technique. Currently, the proposed extrapolation technique is based on sequential computation mechanism, i.e., we extrapolate every output of the model one-by-one. With the increase of the number of model outputs, the efficiency of the extrapolation decreases. To address this issue, we can improve the extrapolation technique by expanding the extrapolation formula from one output variable to multiple output variables. We can add parallel computation during the extrapolation process to extrapolate multiple output variables simultaneously. In this way, we can speed up the extrapolation process when there are multiple output variables needed to be extrapolated.

- As a further research, the proposed automated model generation method can be applied to deep neural network modeling for microwave applications. With the challenges of increasing complexity in microwave modeling and design, deep neural network is becoming an important vehicle in addressing modeling problem with high-dimension and large range. Problems that require more than two hidden layers were rare prior to deep learning. How to determine the number of hidden layers in a deep neural network becomes an important aspect during the modeling process. We can apply the proposed automated model generation method into deep neural network modeling process. One hidden layer is initially used, then the algorithm adjusts the neural network size (i.e., adding or reducing hidden layers) whenever it detects under-learning or over-learning. During this process, we fix the number of hidden neurons in each hidden layer. All the subtasks involved in neural modeling are still integrated

as one framework. In this way, the intensive human effort demanded by the conventional step-by-step process in deep neural network modeling process can be significantly reduced.

- Automated model generation algorithm for dynamic neural network (DNN) and time-delay neural network (TDNN) for microwave modeling can also be envisioned. Automated dynamic order selection and automated time-delay selection will be incorporated into the proposed automated modeling method to automatically choose the most suitable orders for a DNN model and the appropriate number of time-delays for a TDNN model, respectively. The proposed adaptive sampling technique will be also utilized during the automated DNN and TDNN development process. The resulting DNN and TDNN models can be used for behavior modeling for nonlinear device and circuits.

Appendix A

In this appendix, we provide detailed information of the matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$ defined in (5.8) following the existing literature [112]. For n-dimensional extrapolation problems, the 4^n unknown coefficients in $\boldsymbol{\alpha}$ for a given cell with the origin $(i_1, i_2, \ldots, i_k, \ldots, i_n)$ in (5.4) will be determined in such a way that the values of f and the 1st order to nth order mixed partial derivatives of f take prescribed values at N corner points of the cell [112]. The values of f and the 1st order to nth order partial derivatives of f at every corner point of the cell whose origin is $(i_1, i_2, \ldots, i_k, \ldots, i_n)$ can be directly derived from (5.4). More specifically, for the given cell whose origin is

$$(i_1, i_2, \ldots, i_k, \ldots, i_n)\,, \tag{6.1}$$

a $4^n \times 4^n$ matrix $\boldsymbol{A}$ is defined from the grid information relating to the values of f and the 1st order to nth order partial derivatives of f at every corner point of the cell, i.e.,

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \vdots \\ \boldsymbol{A}_N \end{bmatrix} \tag{6.2}$$

where the submatrices $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\cdots$, $\boldsymbol{A}_N$ are defined following [112]. In order to provide a summary of definitions for $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\cdots$, $\boldsymbol{A}_N$, we first define two parameters β_k and l_m. We define

$$\beta_k = \frac{\left(x_k - x_k^{(i_k)}\right)}{x_k^{(i_k+1)} - x_k^{(i_k)}}, \quad k = 1, 2, \cdots, n \tag{6.3}$$

where β_k is a coefficient evaluated according to the location of a given corner point of the cell, i.e., according to whether $x_k = x_k^{(i_k)}$ or $x_k = x_k^{(i_k+1)}$, where x_k is the kth coordinate value of the given corner point. Specifically, β_k is evaluated as

$$\beta_k = \begin{cases} 1 & \text{if } x_k = x_k^{(i_k+1)} \\ 0 & \text{if } x_k = x_k^{(i_k)} \end{cases} \tag{6.4}$$

We also define index parameters l_1 and l_2 as

$$l_1 = u_1, \tag{6.5}$$

$$l_2 = \sum_{r_1=1}^{u_1} (n - r_1) - (n - u_2) \tag{6.6}$$

and $l_m,\ m = 3, \cdots, n$, as

$$\begin{aligned} l_m = & \sum_{r_1=1}^{u_1} \sum_{r_2=r_1+1}^{n-m+2} \sum_{r_3=r_2+1}^{n-m+3} \cdots \sum_{r_{m-1}=r_{m-2}+1}^{n-m+(m-1)} (n - r_{m-1}) \\ & - \sum_{r_2=u_2+1}^{n-m+2} \sum_{r_3=r_2+1}^{n-m+3} \cdots \sum_{r_{m-1}=r_{m-2}+1}^{n-m+(m-1)} (n - r_{m-1}) \\ & - \sum_{r_3=u_3+1}^{n-m+3} \sum_{r_4=r_3+1}^{n-m+4} \cdots \sum_{r_{m-1}=r_{m-2}+1}^{n-m+(m-1)} (n - r_{m-1}) \\ & - \cdots - \sum_{r_{m-1}=u_{m-1}+1}^{n-m+(m-1)} (n - r_{m-1}) \\ & - (n - u_m) \end{aligned} \tag{6.7}$$

where u_1, u_2, $\cdots$, u_m are index parameters selected from

$$u_1 \in \{1, 2, \cdots, n - m + 1\}, \tag{6.8a}$$

$$u_2 \in \{u_1 + 1, u_1 + 2, \cdots, n - m + 2\}, \tag{6.8b}$$

$$\vdots$$

$$u_m \in \{u_{m-1} + 1, u_{m-1} + 2, \cdots, n\} \tag{6.8c}$$

For example, for $m = 4$, (6.7) becomes

$$\begin{aligned} l_4 = & \sum_{r_1=1}^{u_1} \sum_{r_2=r_1+1}^{n-2} \sum_{r_3=r_2+1}^{n-1} (n - r_3) - \sum_{r_2=u_2+1}^{n-2} \sum_{r_3=r_2+1}^{n-1} (n - r_3) \\ & - \sum_{r_3=u_3+1}^{n-1} (n - r_3) - (n - u_4) \end{aligned} \tag{6.9}$$

Next we define the contents for submatrices $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\cdots$, $\boldsymbol{A}_N$. Let $a_{it}^{(l)}$ represent the element in the ith row and tth column of submatrix $\boldsymbol{A}_l$ for $l = 1, 2, \cdots, N$,

where $i = 1, 2, \cdots, N$ and t is defined in (5.5). The submatrix $\boldsymbol{A}_l$ and the element $a_{it}^{(l)}$ in $\boldsymbol{A}_l$ are then defined as follows.

For $l = \mathrm{C}_n^0 = 1$,

$$\boldsymbol{A}_l\boldsymbol{\alpha} = \begin{bmatrix} f|_{p_1} \\ f|_{p_2} \\ \vdots \\ f|_{p_N} \end{bmatrix} \tag{6.10}$$

and the element $a_{it}^{(l)}$ in $\boldsymbol{A}_l$ is defined as

$$a_{it}^{(l)} = \prod_{k=1}^{n} (\beta_k)^{j_k} \Big|_{p_i} \tag{6.11}$$

where $j_k \in \{0, 1, 2, 3\}$ for $k = 1, 2, \cdots, n$. Different values of $(j_1, j_2, \cdots, j_n)$ lead to different values of t, as defined in (5.5). Notice that we use the symbol C_n^k to represent the number of k-combinations from a set of n elements.

For

$$l = \mathrm{C}_n^0 + \mathrm{C}_n^1 + \cdots + \mathrm{C}_n^{m-1} + l_m, \tag{6.12}$$

where $m = 1, 2, \cdots, n$, the submatrix $\boldsymbol{A}_l$ is related to the mth order derivative of f w.r.t m variables x_{u_1}, x_{u_2}, $\cdots$, x_{u_m}, defined as

$$\boldsymbol{A}_l\boldsymbol{\alpha} = \begin{bmatrix} \partial^m f / (\partial x_{u_1} \partial x_{u_2} \cdots \partial x_{u_m}) |_{p_1} \\ \partial^m f / (\partial x_{u_1} \partial x_{u_2} \cdots \partial x_{u_m}) |_{p_2} \\ \vdots \\ \partial^m f / (\partial x_{u_1} \partial x_{u_2} \cdots \partial x_{u_m}) |_{p_N} \end{bmatrix} \tag{6.13}$$

where u_1, u_2, $\cdots$, u_m are the subscripts of the derivative variables x_{u_1}, x_{u_2}, $\cdots$,

x_{u_m}, respectively, and their values should satisfy (6.8a) to (6.8c). The parameter l_m is determined from u_1, u_2, $\cdots$, u_m as defined in (6.5) to (6.7). The element $a_{it}^{(l)}$ in $\boldsymbol{A}_l$ is defined as

$$a_{it}^{(l)} = \prod_{\eta=1}^{m} \left(\frac{j_u}{s_u^{(i_u)}}\bigg|_{u=u_\eta} \right) \prod_{\substack{k=1 \\ k\notin U}}^{n} (\beta_k)^{j_k} \Bigg|_{p_i} \tag{6.14}$$

where U is an index set $U = \{u_1, u_2, \cdots, u_m\}$, and l is defined in (6.12).

The submatrices $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\cdots$, $\boldsymbol{A}_N$ contain the grid information of the cell with the origin $(i_1, i_2, \ldots, i_n)$ and are calculated at all corner points of the cell. More specifically, these submatrices correlate the unknown coefficients $\boldsymbol{\alpha}$ to the values of f and the 1st order to nth order partial derivatives of f at every corner points of the cell with the origin $(i_1, i_2, \ldots, i_n)$. For example, $\boldsymbol{A}_1$ correlates $\boldsymbol{\alpha}$ to the values of f at all corner points of the cell; $\boldsymbol{A}_2$ correlates $\boldsymbol{\alpha}$ to the 1st order derivatives of f with respect to x_1 at all corner points of the cell; $\boldsymbol{A}_{n+2}$ correlates $\boldsymbol{\alpha}$ to the 2nd order derivatives of f with respect to x_1 and x_2 at all corner points of the cell; $\boldsymbol{A}_N$ correlates $\boldsymbol{\alpha}$ to the nth order partial derivatives of f with respect to $x_1, x_2, \cdots, x_n$ at all corner points of the cell. Because the elements in these submatrices $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\cdots$, $\boldsymbol{A}_N$ are calculated at the corner points of the cell, the values of β_k in the submatrices are either 0 or 1. The size of each of these submatrices is $N \times 4^n$. These submatrices are sparse and constant matrices. Therefore matrix $\boldsymbol{A}$ is a sparse and constant matrix.

For a n-dimensional problem, the size of $\boldsymbol{A}$ is $4^n \times 4^n$. For example, in a 3-D extrapolation example, the size of $\boldsymbol{A}$ is $4^3 \times 4^3 = 64 \times 64$. As described in Subsection 5.2.6, "64 rows" of $\boldsymbol{A}$ correspond to the grid information relating to the function val-

ues and the mixed partial derivatives $\left[f,\ \frac{\partial f}{\partial x_1},\ \frac{\partial f}{\partial x_2},\ \frac{\partial f}{\partial x_3},\ \frac{\partial^2 f}{\partial x_1 \partial x_2},\ \frac{\partial^2 f}{\partial x_1 \partial x_3},\ \frac{\partial^2 f}{\partial x_2 \partial x_3},\ \frac{\partial^3 f}{\partial x_1 \partial x_2 \partial x_3}\right]$ at all corner points (i.e., eight corner points) of a 3-D cell. "64 columns" of $\boldsymbol{A}$ correspond to the grid information relating to 64 unknown coefficients in $\boldsymbol{\alpha}$ of the 3-D cubic polynomial formula. It is worth noting that the information of the 2nd or higher order partial derivatives with respect to the same variable, such as $\frac{\partial^2 f}{\partial {x_1}^2}$, $\frac{\partial^2 f}{\partial {x_2}^2}$, $\frac{\partial^2 f}{\partial {x_3}^2}$, $\frac{\partial^3 f}{\partial {x_1}^2 \partial x_2}$, $\frac{\partial^3 f}{\partial {x_1}^2 \partial x_3}$, $\frac{\partial^3 f}{\partial x_1 \partial {x_2}^2}$, $\frac{\partial^3 f}{\partial {x_2}^2 \partial x_3}$, $\frac{\partial^3 f}{\partial x_1 \partial {x_3}^2}$ and $\frac{\partial^3 f}{\partial x_2 \partial {x_3}^2}$, are not included in $\boldsymbol{A}$. This is because that the current information in $\left[f,\ \frac{\partial f}{\partial x_1},\ \frac{\partial f}{\partial x_2},\ \frac{\partial f}{\partial x_3},\ \frac{\partial^2 f}{\partial x_1 \partial x_2},\ \frac{\partial^2 f}{\partial x_1 \partial x_3},\ \frac{\partial^2 f}{\partial x_2 \partial x_3},\ \frac{\partial^3 f}{\partial x_1 \partial x_2 \partial x_3}\right]$ at all corner points of a cell is sufficient to solve the 4^n unknown variables $\boldsymbol{\alpha}$ by $\boldsymbol{A\alpha} = \boldsymbol{b}$.

In (5.8), $\boldsymbol{b}$ is defined as a vector containing the values of f and the 1st order to nth order mixed partial derivatives of f at every corner point of a cell, i.e.,

$$\boldsymbol{b} = \left[f|_{p_1} \quad f|_{p_2} \quad \cdots \quad f|_{p_N} \quad \left.\frac{\partial f}{\partial x_1}\right|_{p_1} \quad \left.\frac{\partial f}{\partial x_1}\right|_{p_2} \quad \cdots \quad \left.\frac{\partial f}{\partial x_1}\right|_{p_N} \quad \cdots \quad \left.\frac{\partial^n f}{\partial x_1 \partial x_2 \cdots \partial x_n}\right|_{p_1} \right.$$
$$\left. \left.\frac{\partial^n f}{\partial x_1 \partial x_2 \cdots \partial x_n}\right|_{p_2} \quad \cdots \quad \left.\frac{\partial^n f}{\partial x_1 \partial x_2 \cdots \partial x_n}\right|_{p_N} \right]^T \tag{6.15}$$

The size of vector $\boldsymbol{b}$ is 4^n. For example, in a 3-D extrapolation example, there are 64 elements in $\boldsymbol{b}$.

Appendix B

In this appendix, we provide a complete and detailed definition of the proposed submatrices $\boldsymbol{B}_1$, $\boldsymbol{B}_2$, $\cdots$, $\boldsymbol{B}_N$ introduced in (5.11a) and (5.11b). Let $b_{it}^{(l)}$ represent the element in the ith row and tth column of submatrix $\boldsymbol{B}_l$ for $l = 1, 2, \cdots, N$, where $i = 1, 2, \cdots, N$. These submatrices are defined for a given cell whose origin is expressed in (6.1).

For $l = \mathrm{C}_n^0 = 1$, the submatrix $\boldsymbol{B}_l$ relates the model output at the grid points of the given cell and its neighboring cells to the values of f at the corner points of the current cell, defined as in (5.11a). The element $b_{it}^{(l)}$ in $\boldsymbol{B}_l$ is defined as

$$b_{it}^{(l)} = \begin{cases} 1, & \text{if Condition \#1 and Condition \#2} \\ 0, & \text{otherwise} \end{cases} \tag{6.16}$$

where

Condition #1: The grid point $(j_1, j_2, \cdots, j_k, \cdots, j_n)$ coincides with the corner point p_i, where j_k is the grid coordinate along the kth dimension within the local area of the current cell and its neighboring cells, $j_k \in \{0, 1, 2, 3\}$ and $k = 1, 2, \cdots, n$.

Condition #2: t is equal to $1 + \sum_{k=1}^{n} \left(4^{n-k} \cdot j_k\right)$.

For l defined in (6.12), where $m = 1, 2, \cdots, n$, the submatrix $\boldsymbol{B}_l$ is related to the mth order derivative of f w.r.t m variables x_{u_1}, x_{u_2}, $\cdots$, x_{u_m}, defined as in (5.11b). In (5.11b), the index parameters u_1, u_2, $\cdots$, u_m are the subscripts of the derivative variables x_{u_1}, x_{u_2}, $\cdots$, x_{u_m}, respectively, and the values of u_1, u_2, $\cdots$, u_m are defined in (6.8a) to (6.8c). The parameter l_m in (6.12) is determined from u_1, u_2, $\cdots$, u_m as defined in (6.5) to (6.7). The element $b_{it}^{(l)}$ in $\boldsymbol{B}_l$ is defined as

$$b_{it}^{(l)} = \begin{cases} \prod_{\eta=1}^{m} \left(\dfrac{\epsilon_u}{s_u^{(i_u+j_u-1)} + s_u^{(i_u+j_u-2)}} \Big|_{u=u_\eta} \right), \\ \qquad\qquad \text{if Condition \#1 and Condition \#3} \\ 0, \qquad\qquad \text{otherwise} \end{cases} \tag{6.17}$$

where i_u is defined in (6.1), $m = 1, 2, \cdots, n$, and

Condition #3: t is equal to

$$1 + \sum_{k=1}^{n} \left(4^{n-k} \cdot j_k\right) + \sum_{\eta=1}^{m} \left(4^{n-u} \cdot \epsilon_u\right)\Big|_{u=u_\eta},$$

for $\epsilon_u = -1,\ 1$.

Appendix C

In the computer programming implementation, we can further speed up the computation of $(\boldsymbol{A}^{-1}\boldsymbol{B})$ involved in Equations (5.15), (5.16), (5.19), (5.22a) to (5.22d) by taking the step size information out of matrix $\boldsymbol{A}$ and putting it into matrix $\boldsymbol{B}$. In this way, matrix $\boldsymbol{A}$ and matrix $\boldsymbol{A}^{-1}$ become constant matrices and are independent of the modeling problem. For each n, there is a fixed modified $\boldsymbol{A}^{-1}$ which can be preprocessed and saved in pre-calculated data base. Suppose we use $\boldsymbol{A}_{new}$ and $\boldsymbol{B}_{new}$ to represent the modified version of $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively. In this appendix, we show how to modify $\boldsymbol{A}$ and $\boldsymbol{B}$, and provide a mathematical proof that $\boldsymbol{A}_{new}^{-1}\boldsymbol{B}_{new} = \boldsymbol{A}^{-1}\boldsymbol{B}$.

In order to move the step size information from $\boldsymbol{A}$ to $\boldsymbol{B}$, we define a diagonal matrix $\boldsymbol{S}$. Let γ_{jj} represent the element at the jth row and jth column of $\boldsymbol{S}$. The value of γ_{jj} is defined as

$$\gamma_{jj} = \begin{cases} 1, & \text{if } 0 < j \leqslant N \\ \prod_{\eta=1}^{m} s_u^{(i_u)}\Big|_{u=u_\eta}, & \text{if } (l-1)\cdot N < j \leqslant l \cdot N \end{cases} \tag{6.18}$$

where l is computed from u_1, u_2, $\cdots$, u_m using (6.12), (6.5) to (6.7), and $m = 1, 2, \cdots, n$.

The step size information in $\boldsymbol{A}$ is moved to $\boldsymbol{B}$ as follows

$$\boldsymbol{A}_{new} = \boldsymbol{S}\boldsymbol{A} \tag{6.19a}$$

and

$$\boldsymbol{B}_{new} = \boldsymbol{S}\boldsymbol{B} \tag{6.19b}$$

By multiplying $\boldsymbol{S}$ to $\boldsymbol{A}$, the step size parameters in $\boldsymbol{A}$ are completely canceled. The remaining elements in $\boldsymbol{A}$ (i.e., the elements in $\boldsymbol{A}_{new}$) are either nonzeros or zeros. The values of the nonzeros are constants and problem-independent under the given dimensionality parameter n. The position of the nonzeros and zeros are also fixed, because the relative position (except step size parameters) of a n-dimensional cell and its neighboring cells are fixed. Further, the relative position of corner points of a cell (except step size parameters) are also fixed. Therefore, after multiplying $\boldsymbol{S}$ to cancel the step size factor in $\boldsymbol{A}$, the remaining elements in $\boldsymbol{A}$ (i.e., the elements in $\boldsymbol{A}_{new}$) are constants and independent to the application examples. In other words, matrix $\boldsymbol{A}_{new}$ and matrix $\boldsymbol{A}_{new}^{-1}$ become constant matrices and are independent of the modeling problem. For each n, the fixed $\boldsymbol{A}_{new}^{-1}$ can be preprocessed and saved in pre-calculated data case.

It can be proven that $\boldsymbol{A}_{new}^{-1}\boldsymbol{B}_{new} = \boldsymbol{A}^{-1}\boldsymbol{B}$ as follows,

$$\begin{aligned} \boldsymbol{A}_{new}^{-1}\boldsymbol{B}_{new} &= \boldsymbol{A}^{-1}\boldsymbol{S}^{-1}\boldsymbol{S}\boldsymbol{B} \\ &= \boldsymbol{A}^{-1}\boldsymbol{I}\boldsymbol{B} \\ &= \boldsymbol{A}^{-1}\boldsymbol{B} \end{aligned} \tag{6.20}$$

where matrix $\boldsymbol{I}$ represents the identity matrix. In this way, $\boldsymbol{A}_{new}^{-1}\boldsymbol{B}_{new} = \boldsymbol{A}^{-1}\boldsymbol{B}$ is

proved and the modified matrices $\boldsymbol{A}_{new}^{-1}$ and $\boldsymbol{B}_{new}$ are used for the implementation of (5.22a) and (5.22b) in computer programming.

References

[1] Q. J. Zhang and K. C. Gupta, *Neural Networks for RF and Microwave Design,* Norwood, MA: Artech House, 2000.

[2] M. B. Steer, J. W. Bandler, and C. M. Snowden, "Computer-aided design of RF and microwave circuits and systems," *IEEE Trans. Microw. Theory Techn.*, vol. 50, no. 3, pp. 996-1005, Mar. 2002.

[3] J. E. Rayas-Sanchez, "EM-based optimization of microwave circuits using artificial neural networks: The state-of-the-art," *IEEE Trans. Microw. Theory Techn.*, vol. 52, no. 1, pp. 420-435, Jan. 2004.

[4] V. K. Devabhaktruni, M. Yagoub, and Q. J. Zhang, "A robust algorithm for automatic development of neural-network models for microwave applications," *IEEE Trans. Microw. Theory Techn.*, vol. 49, no. 12, pp. 2282–2291, Dec. 2001.

[5] V.K. Devabhaktuni, C. Xi, F. Wang, and Q. J. Zhang, "Robust training of microwave neural models," *Int. J. RF Microwave CAE*, vol. 12, no. 1, pp. 109-124, Jan. 2002.

[6] F. Wang, V. K. Devabhaktuni, C. Xi, and Q. J. Zhang, "Neural network structures and training algorithms for RF and microwave applications," *Int. J. RF Microwave CAE*, vol. 9, pp. 216-240, 1999.

[7] 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 Techn.,* vol. 45, no. 5, pp. 794-802, May 1997.

[8] F. Wang and Q. J. Zhang, "Knowledge based neural models for microwave design," *IEEE Trans. Microw. Theory Techn.,* vol. 45, no. 12, pp. 2333-2343, Dec. 1997.

[9] C. Brezinski and M. R. Zagliz, *Extrapolation Methods: Theory and Pratice.* New York: North-Holland, 1991.

[10] L. C. Kiong, M. Rajeswari, and M. V. C. Rao, "Extrapolation detection and novelty-based node insertion for wequential growing multi-experts network," *Appl. Soft Comput.,* vol. 3, no. 2, pp. 159-175, Sep. 2003.

[11] J. Xu, M. C. E. Yagoub, R. Ding, and Q. J. Zhang, "Robust neural based microwave modeling and design using advanced model extrapolation," in *IEEE MTT-S Int. Microw. Symp. Dig.,* Fort Worth, TX, USA, Jun. 2004, pp. 1549-1552.

[12] L. Zhang and Q. J. Zhang, "Simple and effective extrapolation technique for neural-based microwave modeling," *IEEE Microw. Wireless Compon. Lett.,* vol. 20, no. 6, pp. 301-303, Jun. 2010.

[13] W. Na and Q. J. Zhang, "Automated knowledge-based neural network modeling for microwave applications," *IEEE Microw. Wireless Compon. Lett.,* vol. 24, no. 7, pp. 499–501, Jul. 2014.

[14] W. Na and Q. J. Zhang, "Automated parametric modeling of microwave components using combined neural network and interpolation techniques," in *IEEE MTT-S Int. Dig.,* Seattle, WA, USA, Jun. 2013.

[15] W. Na, F. Feng, C. Zhang, and Q. J. Zhang, "A unified automated parametric modeling algorithm using knowledge-based neural network and l1 optimization," *IEEE Trans. Microw. Theory Techn.,* vol. 65, no. 3, pp. 729-745, Mar. 2017.

[16] W. Na and Q. J. Zhang, "Unified automated knowledge-based neural network modeling for microwave devices," in *IEEE MTT-S Int. Conf. Numerical Electromagnetic and Multiphysics Modeling and Optim.*, Ottawa, ON, Canada, Aug. 2015, pp. 1–3.

[17] W. Na, W. Liu, L. Zhu, F. Feng, J. Ma and Q. J. Zhang, "Advanced extrapolation technique for neural-based microwave modeling and design," *IEEE Trans. Microw. Theory Techn.,* in press, Jun. 2018.

[18] Q. J. Zhang, K. C. Gupta, and V. K. Devabhaktuni, "Artificial neural networks for RF and microwave design: from theory to practice," *IEEE Trans. Microw. Theory Techn.,* vol. 51, no. 4, pp. 1339-1350, Apr. 2003.

[19] A. Viveros-Wacher and J. E. Rayas-Sanchez, "Analog fault identification in RF circuits using artificial neural networks and constrained parameter extraction,"

in *IEEE MTT-S Int. Conf. Num. EM Multiphysics Modeling Opt. (NEMO-2018)*, Reykjavik, Iceland, Aug. 2018, pp. 1-3.

[20] H. Kabir, L. Zhang, M. Yu, P. H. Aaen, J. Wood, and Q. J. Zhang, "Smart modeling of microwave devices," *IEEE Microw. Mag.*, vol. 11, no. 3, pp. 105-118, May 2010.

[21] F. E. Rangel-Patino, J. E. Rayas-Sanchez, A. Viveros-Wacher, J. L. Chavez-Hurtado, E. A. Vega-Ochoa, and N. Hakim, "Post-silicon receiver equalization metamodeling by artificial neural networks," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, May 2018.

[22] S. K. Mandal, S. Sural, and A. Patra, "ANN- and PSO-based synthesis of on-chip spiral inductors for RF ICs," *IEEE Trans. Comp.-Aided Design*, vol. 27, no. 1, pp. 188-192, Jan. 2008.

[23] J. E. Rayas-Sanchez, "Power in simplicity with ASM: tracing the aggressive space mapping algorithm over two decades of development and engineering applications," *IEEE Microwave Magazine*, vol. 17, no. 4, pp. 64-76, Apr. 2016.

[24] W. Liu, W. Na, L. Zhu, and Q. J. Zhang, "A review of neural network based techniques for nonlinear microwave device modeling," in *IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optim.,*, Beijing, China, Jul. 2016.

[25] A. H. Zaabab, Q. J. Zhang, and M. S. Nakhla, "A neural network modeling approach to circuit optimization and statistical design," *IEEE Trans. Microw. Theory Techn.,* vol. 43, no. 6, pp. 1349-1358, Jun. 1995.

[26] X. Ding, J. Xu, M. C. E. Yagoub, and Q. J. Zhang, "A combined state space formulation/equivalent circuit and neural network technique for modeling of embedded passives in multilayer printed circuits," *J. of the Applied Computational Electromagnetics Society*, vol. 18, 2003.

[27] X. Ding, V. K. Devabhaktuni, B. Chattaraj, M. C. E. Yagoub, M. Doe, J. J. Xu, and Q. J. Zhang, "Neural-network approaches to electromagnetic-based modeling of passive components and their applications to high-frequency and high-speed nonlinear circuit optimization," *IEEE Trans. Microw. Theory Tech.*, vol. 52, no. 1, pp. 436-449, Jan. 2004.

[28] R. Biernacki, J. W. Bandler, J. Song, and Q. J. Zhang, "Efficient quadratic approximation for statistical design," *IEEE Trans. Circuit Syst.*, vol. 36, no. 11, pp. 1449-1454, Nov. 1989.

[29] P. Meijer, "Fast and smooth highly nonlinear multidimensional table models for device modeling," *IEEE Trans. Circuit Syst.*, vol. 37, no. 3, pp. 335-346, Mar. 1990.

[30] Q. J. Zhang, F. Wang, and M. S. Nakhla, "Optimization of high-speed VLSI interconnects: A review," *Int. J. of Microwave and Millimeter-Wave CAD*, vol. 7, pp. 83-107, 1997.

[31] Y. Cao, L. Simonovich, and Q. J. Zhang, "A broadband and parametric model of differential via holes using space-mapping neural network," *IEEE Microw. Wireless Compon. Lett.*, vol. 19, no. 9, pp. 533-535, Sep. 2009.

[32] J. W. Bandler, M. A. Ismail, J. E. Rayas-Sanchez, and Q. J. Zhang, "Neuromodeling of microwave circuits exploiting space-mapping technology," *IEEE Trans. Microw. Theory Techn.*, vol. 47, no. 12, pp. 2417-2427, Dec. 1999.

[33] P. M. Watson and K. C. Gupta, "Design and optimization of CPW circuits using EM-ANN models for CPW components," *IEEE Trans. Microw. Theory Techn.,* vol. 45, no. 12, pp. 2515-2523, Dec. 1997.

[34] M. Vai and S. Prasad, "Qualitative modeling heterojunction bipolar transistors for optimization: A neural network approach," *Proc. IEEE/Cornell Conf. Adv. Concepts in High Speed Semiconductor Dev. and Circuits,* pp. 219-227, Aug. 1993.

[35] V. K. Devabhaktuni, C. Xi, and Q. J. Zhang, "A neural network approach to the modeling of heterojunction bipolar transistors from S-parameter data," *Proc. 28th European Microw. Conf.*, Amsterdam, Netherlands, Oct. 1998, pp. 306-311.

[36] A. Huang, Z. Zhong, W. Wu, and Y. Guo, "Artificial neural network-based electrothermal model for GaN HEMTs with dynamic trapping effects consideration," *IEEE Trans. Microw. Theory Tech.*, vol. 64, no. 8, pp. 2519-2528, Aug. 2016.

[37] L. Zhu, Q. J. Zhang, K. Liu, Y. Ma, B. Peng, and S. Yan, "A novel dynamic neuro-space mapping approach for nonlinear microwave device modeling," *IEEE Microw. Wireless Compon. Lett.*, vol. 26, no. 2, pp. 131-133, Feb. 2016.

[38] A. Garcia-Lamperez and M. Salazar-Palma, "Multilevel aggressive space mapping applied to coupled-resonator filters," in *IEEE MTT-S Int. Microw. Symp. Dig.*, San Francisco, CA, May 2016, pp. 1-4.

[39] M. Sans, J. Selga, P. Velez, A. Rodriguez, J. Bonache, V. E. Boria, and F. Martin, "Automated design of common-mode suppressed balanced wideband bandpass filters by means of aggressive space mapping," *IEEE Trans. Microw. Theory Tech.*, vol. 63, no. 12, pp. 3896-3908, Dec. 2015.

[40] F. Feng, C. Zhang, V.-M.-R. Gongal-Reddy, Q. J. Zhang, and J. Ma, "Parallel space-mapping approach to EM optimization," *IEEE Trans. Microw. Theory Techn.*, vol. 62, no. 5, pp. 1135-1148, May 2014.

[41] F. Feng, V.-M.-R. Gongal-Reddy, C. Zhang, J. G. Ma, and Q. J. Zhang, "Parametric modeling of microwave components using adjoint neural networks and pole-residue-based transfer functions with EM sensitivity analysis," *IEEE Trans. Microw. Theory Techn.*, vol. 65, no. 6, pp. 1955-1975, Feb. 2017.

[42] M. Isaksson, D. Wisell, and D. Ronnow, "Wide-band dynamic modeling of power amplifiers using radial-basis function neural networks," *IEEE Trans. Microw. Theory Techn.,* vol. 53, no. 11, pp. 3422–3428, Nov. 2005.

[43] F. Mkadem and S. Boumaiza, "Physically inspired neural network model for RF power amplifier behavioral modeling and digital predistortion," *IEEE Trans. Microw. Theory Techn.,* vol. 59, no. 4, pp. 913–923, Apr. 2011.

[44] D. Schreurs, M. O'Droma, A. A. Goacher, and M. Gadringer, *RF Power Amplifier Behavioral Modeling.* Cambridge, U.K.: Cambridge Univ. Press, 2008.

[45] N. Knudde, I. Couckuyt, D. Spina, K. Lukasik, P. Barmuta, D. Schreurs, and T. Dhaene,"Data-efficient Bayesian optimization with constraints for power amplifier design," in *IEEE MTT-S Int. Conf. Num. EM Multiphysics Modeling Opt. (NEMO-2018)*, Reykjavik, Iceland, Aug. 2018, pp. 1-3.

[46] Y. H. Fang, M. C. E. Yagoub, F. Wang and Q. J. Zhang, "A new macromodeling approach for nonlinear microwave circuits based on recurrent neural networks," *IEEE Trans. Microw. Theory Tech.*, vol. 48, no. 12, pp. 2335-2344, Dec. 2000.

[47] G. Gosal, E. Almajali, D. McNamara and M. Yagoub, "Transmitarray antenna design using forward and inverse neural network modeling," *IEEE Ant. and Wireless Prop. Lett.*, vol. 15, no. 1, pp. 1483-1486.

[48] H. M. Torun, M. Larbi, and M. Swaminathan "A Bayesian framework for optimizing interconnects in high-speed channels," in *IEEE MTT-S Int. Conf. Num. EM Multiphysics Modeling Opt. (NEMO-2018)*, Reykjavik, Iceland, Aug. 2018, pp. 1-4.

[49] H. Yu, M. Swaminathan, C. Ji, and D. White, "A nonlinear behavioral modeling approach for voltage-controlled oscillators using augmented neural net-

works," *IEEE MTT-S Int. Microw. Symp. Dig.*, Philadelphia, PA, Jun. 2018, pp. 1-4.

[50] Q. J. Zhang and M. S. Nakhla, "Signal integrity analysis and optimization of VLSI interconnects using neural network models," *IEEE Int. Circuits Syst. Symp.*, London, England, pp. 459-462, May 1994.

[51] T. Hong, C. Wang, and N.G. Alexopoulos, "Microstrip circuit design using neural networks," *IEEE MTT-S Int. Microw. Symp. Dig.*, Atlanta, GA, Jun. 1993, pp. 413-416.

[52] M. D. Baker, CD. Himmel, and G.S. May, "In-situ prediction of reactive ion etch endpoint using neural networks," *IEEE Trans. Components, Packaging, and manufacturing Tech. Part A*, vol.18, no. 3, pp. 478-483, Sep. 1995.

[53] M. Vai and S. Prasad, "Microwave circuit analysis and design by a massively distributed computing network," *IEEE Trans. Microwave Theory Tech.,* vol. 43, no. 5, pp. 1087-1094, May 1995.

[54] M. Vai, S. Wu, B. Li, and S. Prasad, "Reverse modeling of microwave circuits with bidirectional neural network models," *IEEE Trans. Microwave Theory Tech.,* vol. 46, no. 10, pp. 1492-1494, Oct. 1998.

[55] J. A. Jargon, K. C. Gupta, and D. C. DeGroot, "Applications of artificial neural networks to RF and microwave measurements," *Int. J. RF Microwave CAE,* vol. 12, no. 1, pp. 3-24, Jan. 2002.

[56] P. M. Watson, C. Cho, and K. C. Gupta, "Electromagnetic artificial neural network model for synthesis of physical dimensions for multilayer asymmetric coupled transmission structures," *Int. J. RF Microwave CAE,* vol. 9, no. 3, pp. 175-186, May 1999.

[57] 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 Microw. Computer-Aided Eng.*, vol. 12, no. 1, pp. 51-70, Jan. 2002.

[58] I. S. Stievano, I. A. Maio, and F. G. Canavero, "Parametric macromodels of digital I/O ports," *IEEE Trans. Adv. Packag.*, vol. 25, no. 5, pp. 255-264, May 2002.

[59] B. O'Brien, J. Dooley, and T. J. Brazil, "RF power amplifier behavioral modeling using a globally recurrent neural network," *in IEEE MTT-S Int. Microw. Symp. Dig.*, San Francisco, CA, Jun. 2006, pp. 1089-1092.

[60] T. Liu, S. Boumaiza, and F. M. Ghannouchi, "Dynamic behavioral modeling of 3G power amplifiers using real-valued time-delay neural networks," *IEEE Trans. Microw. Theory Techn.*, vol. 52, no. 3, pp. 1025-1033, Mar. 2004.

[61] J. Xu, M. C. E. Yagoub, R. Ding, and Q. J. Zhang, "Neural based dynamic modeling of nonlinear microwave circuits," *IEEE Trans. Microw. Theory Techn.*, vol. 50, no. 12, pp. 2769-2780, Dec. 2002.

[62] Y. Cao, J. J. Xu, V. K. Devabhaktuni, R. T. Ding, and Q. J. Zhang, "An adjoint dynamic neural network technique for exact sensitivities in nonlinear transient modeling and high-speed interconnect design," *in IEEE MTT-S Int. Microw. Symp. Dig.*, PA, Philadelphia, Jun. 2003, pp. 165-168.

[63] Y. Cao, R. T. Ding, and Q. J. Zhang, "State-space dynamic neural network technique for high-speed IC applications: Modeling and stability analysis," *IEEE Trans. Microw. Theory Techn.*, vol. 54, no. 6, pp. 2398-2409, Jun. 2006.

[64] Y. Cao, R. T. Ding, and Q. J. Zhang, "A new nonlinear transient modeling technique for high-speed integrated circuit applications based on state-space dynamic neural network," *IEEE MTT-S Int. Microw. Symp. Dig.*, Fort Worth, TX, Jun. 2004, pp. 1553-1556.

[65] J. M. Zamarreno, P. Vega, L. D. Garcia, and M. Francisco, "State-space neural network for modeling, prediction and control," *Contr. Eng. Practice*, vol. 8, no. 9, pp. 1063-1075, Sep. 2000.

[66] P. Gil, A. Dourado, and J. O. Henriques, "State space neural networks and the unscented Kalman filter in online nonlinear system identification," *IASTED Int. Conf. Intell. Syst. Contr.*, Tampa, FL, Nov. 2001, pp. 337-342.

[67] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Systems*, vol. 2, pp. 303-314, 1989.

[68] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359-366, 1989.

[69] T. Y. Kwok, and D. Y. Yeung, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," *IEEE Trans. Neural Networks*, vol. 8, no. 3, pp. 630-645, May 1997.

[70] R. Reed, "Pruning algorithms - a Survey," *IEEE Trans. Neural Networks*, vol. 4, no. 5, pp. 740-747, Sep. 1993.

[71] A. Krzyzak, and T. Linder, "Radial basis function networks and complexity regularization in function learning," *IEEE Trans. Neural Networks*, vol. 9, no. 2, pp. 247-256, Mar. 1998.

[72] J. de Villiers, and E. Barnard, "Backpropagation neural nets with one and two hidden layers," *IEEE Trans. Neural Networks*, vol. 4, no. 1, pp. 136-141, Jan. 1993.

[73] S. Tamura, and M. Tateishi, "Capabilities of a four-layered feedforward neural network: four layer versus three," *IEEE Trans. Neural Networks*, vol. 8, no. 2, pp. 251-255, Mar. 1997.

[74] J. A. Garcia, et al., "Modeling MESFET's and HEMT's intermodulation distortion behavior using a generalized radial basis function network," *Int. Journal of RF and Microwave CAE*, Special Issue on Applications of ANN to RF and Microwave Design, vol. 9, pp. 261-276, 1999.

[75] J. Park, and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, pp. 246-257, 1991.

[76] J. Aweya, Q. J. Zhang, and D. Montuno, "A direct adaptive neural controller for flow control in computer networks," *IEEE Int. Conf. Neural Networks*, Anchorage, Alaska, pp. 140-145, May 1998.

[77] J. Aweya, Q. J. Zhang, and D. Montuno, "Modelling and control of dynamic queues in computer networks using neural networks," *IASTED Int. Conf. Intelligent Syst. Control*, Halifax, Canada, pp. 144-151, Jun. 1998.

[78] L. H. Tsoukalas, and R. E. Uhrig, *Fuzzy and Neural Approaches in Engineering*, NY: Wiley-Interscience, 1997.

[79] 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, Editors, Cambridge, MA:MIT Press, pp. 318-362, 1986.

[80] M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, pp. 525-533, 1993.

[81] T. R. Cuthbert, "Quasi-newton methods and constraints," In *Optimization using Personal Computers*, NY: John Wiley & Sons, pp. 233-314, 1987.

[82] A. J. Shepherd, "Second-order optimization methods," In *Second-Order Methods for Neural Networks*, London: Springer-Verlag, pp. 43-72, 1997.

[83] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA: Addison-Wesley, 1989.

[84] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671-680, 1983.

[85] J. W. Bandler, R. M. Biernacki, S. H. Chen, P. A. Grobelny, and R. H. Hemmers, "Space mapping technique for electromagnetic optimization," *IEEE Trans. Microw. Theory Tech.*, vol. 42, no. 12, pp. 2536-2544, Dec. 1994.

[86] F. Feng, C. Zhang, J. G. Ma, and Q. J. Zhang, "Parametric modeling of EM behavior of microwave components using combined neural networks and pole-residue-based transfer functions," *IEEE Trans. Microw. Theory Techn.*, vol. 64, no. 1, pp. 60-77, Jan. 2016.

[87] J. W. Bandler, Q. S. Cheng, S. A. Dakroury, A. S. Mohamed, M. H. Bakr, K. Madsen, and J. Sondergaard, "Space mapping: The state of the art," *IEEE Trans. Microw. Theory and Techn.*, vol. 52, no. 1, pp. 337-361, Jan. 2004.

[88] V. Gutiérrez-Ayala and J. E. Rayas-Sanchez, "Neural input space mapping optimization based on nonlinear two-layer perceptrons with optimized nonlinearity," *Int. J. RF Microw. Comput.-Aided Eng.*, vol. 20, no. 5, pp. 512-526, Sep. 2010.

[89] M. H. Bakr, J. W. Bandler, M. A. Ismail, J. E. Rayas-Sanchez, and Q. J. Zhang, "Neural space-mapping optimization for EM-based design," *IEEE Trans. Microw. Theory Techn.*, vol. 48, no. 12, pp. 2307-2315, Dec. 2000.

[90] S. Koziel, J. W. Bandler, and K. Madsen, "Space mapping with adaptive response correction for microwave design optimization," *IEEE Trans. Microw. Theory Techn.*, vol. 57, no. 2, pp. 478-486, Feb. 2009.

[91] J. E. Rayas-Sanchez and V. Gutiérrez-Ayala, "EM-based Monte Carlo analysis and yield prediction of microwave circuits using linear-input neural-output space mapping," *IEEE Trans. Microw. Theory Techn.*, vol. 54, no. 12, pp. 4528-4537, Dec. 2006.

[92] R. B. Ayed, J. Gong, S. Brisset, F. Gillon, and P. Brochet, "Three-level output space mapping strategy for electromagnetic design optimization," *IEEE Trans. Magn.*, vol. 48, no. 2, pp. 671-674, Feb. 2012.

[93] J. W. Bandler, Q. S. Cheng, D. H. Gebre-Mariam, K. Madsen, F. Pedersen, and J. Søndergaard, "EM-based surrogate modeling and design exploiting implicit, frequency and output space mappings," in *IEEE MTT-S Int. Microw. Symp. Dig.*, Philadelphia, PA, Jun. 2003, pp. 1003-1006.

[94] R. M. Biernacki, J.W. Bander, J. Song, and Q. J. Zhang, "Efficient quadratic approximation for statistical design," *IEEE Trans. on Circuits Syst.*, vol. 36, pp. 1449-1454, 1989.

[95] J. C. Nash, *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation*, Bristol, England: Adam Hilger, 1990.

[96] J. Xu, M. C. E. Yagoub, R. T. Ding, and Q. J. Zhang, "Exact adjoint sensitivity analysis for neural based microwave modeling and design," *IEEE Trans. Microw. Theory Techn.*, vol. 51, pp. 226-237, 2003.

[97] W. Liu, W. Na, L. Zhu, J. Ma, and Q. J. Zhang, "A Wiener-type dynamic neural network approach to the modeling of nonlinear microwave devices," *IEEE Trans. Microw. Theory Techn.*, vol. 65, no. 6, pp. 2043-2062, Feb. 2017.

[98] S. Koziel, Q. S. Cheng, and J. W. Bandler, "Space mapping," *IEEE Microw. Mag.*, vol. 9, no. 6, pp. 105-122, Dec. 2008.

[99] S. Koziel, J. W. Bandler, and K. Madsen, "A space-mapping framework for engineering optimization-Theory and implementation," *IEEE Trans. Microw. Theory Techn.*, vol. 54, no. 10, pp. 3721-3730, Oct. 2006.

[100] L. Zhang, J. Xu, M. C. M. Yagoub, R. Ding, and Q. J. Zhang, "Efficient analytical formulation and sensitivity analysis of neuro-space mapping for nonlinear microwave device modeling," *IEEE Trans. Microw. Theory Techn.*, vol. 53, no. 9, pp. 2752-2767, Sep. 2005.

[101] D. Gorissen, L. Zhang, Q. J. Zhang, and T. Dhaene, "Evolutionary neuro-space mapping technique for modeling of nonlinear microwave devices," *IEEE Trans. Microw. Theory Techn.*, vol. 59, no. 2, pp. 213-229, Feb. 2010.

[102] J. W. Bandler, S. H. Chen, and S. Daijavad, "Microwave device modeling using efficient l_1 optimization: A novel approach," *IEEE Trans. Microw. Theory Techn.*, vol. 34, no. 12, pp. 1282-1293, Dec. 1986.

[103] J. W. Bandler, W. Kellermann, and K. Madsen, "A nonlinear l_1 optimization algorithm for design, modeling, and diagnosis of networks," *IEEE Trans. Circuits Syst.*, vol. 34, no. 2, pp. 174-181, Feb. 1987.

[104] J. Hald and K. Madsen, "Combined LP and quasi-Newton methods for nonlinear l_1 optimization," *J. Numer. Anal.*, vol. 22, no. 1, pp. 68-80, 1985.

[105] S. R. Schmidt and R. G. Launsby, *Understanding Industrial Designed Experiments.* Colorado Springs, CO, USA: Air Force Academy, 1992.

[106] M. L. Strydom and J. P. Eberhard, "Reliability and efficiency in computational electromagnetics: How CEM software meets the future needs of engineers," in *Proc. IEEE Antennas Propag. Soc. Int. Symp.*, Honolulu, HI, USA, Jun. 2007, pp. 5451-5454.

[107] A. Vasylchenko, Y. Schols, W. D. Raedt, and G. A. E. Vandenbosch, "Quality assessment of computational techniques and software tools for planar-antenna analysis," *IEEE Antennas Propag. Mag.*, vol. 51, no. 1, pp. 23-38, Feb. 2009.

[108] W. H. Tu and K. Chang, "Microstrip eeliptic-function low-pass filters using distributed elements or slotted ground structure," *IEEE Trans. Microw. Theory Techn.*, vo. 54, no. 10, pp. 3786-3792, Oct. 2006.

[109] R. Esfandiari, D. Maku, and M. Siracusa, "Design of interdigitated capacitors and their application to gallium arsenide monolithic filters," *IEEE Trans. Microw. Thoery Techn.*, vol. 31, no. 1, pp. 57-64, Jan. 1983.

[110] E. Hammerstad and O. Jensen, "Accurate models for microstrip computer-aided design," in *IEEE MTT-S Int. Microw. Symp. Dig.*, Washington, DC, USA, May 1980, pp. 407-409.

[111] M. Kirschning and R. H. Jansen, "Accurate wide-range design equations for the frequency-dependent characteristic of parallel coupled microstrip lines," *IEEE Trans. Microw. Theory Techn.*, vol. 32, no. 1, pp. 83-90, Jan. 1984.

[112] F. Lekien and J. Marsden, "Tricubic interpolation in three dimensions," *Journal of Numerical Methods and Engineering*, vol. 63, no. 3, pp. 455-471, Mar. 2005.

[113] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409-436, Dec. 1952.

[114] P. Triverio, S. Grivet-Talocia, M. Bandinu, and F. G. Canavero, "Geometrically parameterized circuit models of printed circuit board traces inclusive of antenna coupling," *IEEE Trans. Electromagn. Compat.*, vol. 52, no. 2, pp. 471-478, May 2010.

[115] P. Triverio, S. Grivet-Talocia, and M. Nahkla, "An improved fitting algorithm for parametric macromodeling from tabulated data," *in Proc. IEEE Conf. Signal Propagat. Interconn.*, May 2008, p. 4.

[116] F. Ferranti, T. Dhaene, and L. Knockaert, "Compact and passivity parametric macromodeling using reference macromodels and positive interpolation opera-

tors," *IEEE Trans. Components, Packaging and Manufacturing Tech.*, vol. 2, no. 12, pp. 2080-2088, Dec. 2012.

[117] Y. Cao, G. Wang, and Q. J. Zhang, "A new training approach for parametric modeling of microwave passive components using combined neural networks and transfer functions," *IEEE Trans. Microw. Theory Techn.*, vol. 57, no. 11, pp. 2727-2742, Nov. 2009.

[118] J. Jin, *The Finite-Element Method in Electromagnetics, 3nd ed.*, Somerset, NJ, USA: Wiley-IEEE Press, 2015.

[119] P. H. Chen, R. E. Fan, and C. J. Lin, "A study on SMO-type decomposition methods for support vector machines," *IEEE Trans. Neural Networks*, vol. 17, no. 4, pp. 893-908, Jul. 2006.

[120] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, Jul. 2006.

[121] H. Statz, P. Newman, I. W. Smith, R. A. Pucel, and H. A. Haus, "GaAs FET device and circuit simulation in SPICE," *IEEE Trans. Electron Devices*, vol. 34, no. 2, pp. 160-169, Feb. 1987.

[122] D. Gorissen, L. De Tommasi, K. Crombecq, and T. Dhaene, "Sequential modeling of a low noise amplifier with neural networks and active learning," *Neural Comput. Applic.*. vol. 18, no. 5, pp. 485-494, Jun. 2009.

[123] J. L. Chavez-Hurtado and J. E. Rayas-Snchez, "Polynomial-based surrogate modeling of RF and microwave circuits in frequency domain exploiting the multi-

nomial theorem," *IEEE Trans. Microwave Theory Techn.*, vol. 64, no. 12, pp. 4371-4381, Dec. 2016.

[124] C. Y. Chang and F. Kai, *GaAs High-Speed Devices: Physics, Technology, and Circuit Applications*. New York, NY, USA: Wiley, 1994.