Neural Networks for RF and Microwave Design: Toward Automatic Model Generation

By

Vijaya Kumar Devabhaktuni, B.E. (EE), M.Sc. (Physics)

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

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Electronics
Faculty of Engineering and Design
Carleton University
Ottawa, Ontario, K1S 5B6, Canada

January 22, 2003

© Vijaya Kumar Devabhaktuni

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-79453-9

Canada

The undersigned recommend to the
Faculty of Graduate Studies and Research
acceptance of the thesis

Neural Networks for RF and Microwave Design: Toward Automatic Model Generation

Submitted by Vijaya Kumar Devabhaktuni, B.E. (EE), M.Sc. (Physics),
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Dr. Q.J. Zhang (Thesis Supervisor)

Professor, Department of Electronics, Carleton University, Ottawa, Canada

Dr. Michel S. Nakhla

Professor and Chairman, Department of Electronics, Carleton University, Ottawa, Canada

Dr. Ke Wu (External Examiner)

Professor, Department of Electrical Engineering, Ecole Polytechnique, Montreal, Canada

Abstract

Over the last decade, rapid progress in telecommunications, VLSI and wireless industries resulted in ever-increasing circuit complexities and signal frequencies. This, coupled with tightened design tolerances and shortened time-to-market cycles, necessitated powerful computer aided design (CAD) methodologies. Artificial neural networks (ANN) have recently gained recognition as accurate and fast vehicle for RF and microwave CAD including modeling, simulation and optimization. Key objectives of the thesis are, (a) To develop robust neural modeling techniques that can address complicated high-frequency modeling challenges, and (b) To formulate efficient neural network modeling algorithms that can facilitate automatic generation of accurate RF and microwave neural models.

Major contributions of the thesis include, the iterative multi stage (IMS) algorithm, the automatic model generation (AMG) algorithm, and the knowledge-based automatic model generation (KAMG) algorithm. Through iterative decomposition and stage-wise training, the IMS allows neural modeling of otherwise complicated highly nonlinear and multi-dimensional RF/microwave behaviors even in the presence of gross errors or sharp variations in training data. Starting with no training data and proceeding with adaptive training and validation data augmentation during neural network training, the AMG automatically generates accurate neural models with reduced use of data and shortened model development time. Motivated by the space-mapping concept, the KAMG utilizes extensive coarse data but fewest fine data as compared to any other existing neural modeling techniques to generate neural network models that accurately match fine data, with minimal human intervention. By

virtues of automation, efficiency and robustness, the work presented in the thesis enhances today's high-frequency electronics design and strengthens the ground for computerization of neural network based end-to-end CAD methodologies in future.

Acknowledgements

I would like to thank my thesis advisor Dr. Q.J. Zhang for active discussions, continued encouragement, expert guidance and wonderful supervision. His leadership and vision for quality research and developmental activities has made the pursuit of this thesis a challenging, enjoyable, rewarding and stimulating experience.

Dr. Fang Wang provided me with direction, guidance and support during the initial stages of the thesis. Dr. Mustapha Yagoub contributed to my research through active involvement and invaluable collaboration. I wish to thank Dr. K.J. Vinoy for reading the manuscript and for many helpful suggestions to improve the thesis. I like to express my gratitude to Biswarup Chattaraj, Changgeng Xi and all other colleagues for their cooperation.

Dr. Michel Nakhla and Dr. Garry Tarr are specially thanked for giving me opportunities for undergraduate teaching. I enjoyed working with Tom Ray first as a teaching assistant and later as a colleague. He will remain a precious friend for helping me through difficult times. Dr. Ram Achar is my role model in various aspects of life and work. Nagui Mikhail and Scott Bruce are thanked for providing technical support. Refreshing conversations with Betty Zahalan, Lorena Duncan and Peggy Piccolo will be cherished. Faculty members Dr. Tadeusz Kwasniewski, Dr. Leonard MacEachern and Dr. Alan Steele are thanked for encouragement. Financial support from the Department of Electronics, MICRONET, NSERC, and the Ministry of Training, Colleges and Universities of Ontario, is gratefully acknowledged.

My wife Neelima Uppaluri took a good care of me both at home and school. Without her love and support, working on this thesis could have been extremely difficult. My parents Smt. Ratnamamba and Sri. Gangadhara Rao made sacrifices to give me higher education, and I am happy to have lived up to their expectations. Friends, relatives, teachers, well-wishers, and in particular grand parents, aunts, uncles, cousins, in-laws, Smt. G. Vasumathy, Sri. Y. Nageswara Rao, Smt. V. Kamala Kumari, Dr. L.K. Maheshwari, Murali, Hari, Prasad, and Abhishek are all remembered for being with me all the time.

Dedicated to my grandparents

Sri. Ganne Gangadhara Rao and Smt. Annapurnamma

for showering me with boundless affection and love

Table of Contents

Abstract	i
Acknowledgements	iii
 CHAPTER 1	
Introduction	1
1.1 Background	1
1.2 Motivation	3
1.3 Contributions of the Thesis	5
1.4 Organization	7
 CHAPTER 2	
Artificial Neural Networks for RF and Microwave Design: An Overview	9
2.1 Introduction	9
2.2 Neural Network Structures	13
2.2.1 Basic Components	13
2.2.2 Concept of a Neural Network Model	14
2.2.3 Neural Network versus Conventional Approaches	16
2.2.4 Multi-Layer Perceptrons Neural Networks	17
2.2.4.1 Structure and Notation	17
2.2.4.2 Anatomy of Neurons	18
2.2.4.3 Feedforward Computation	20

2.2.4.4 Important Features	20
2.2.4.5 Network Size and Layers	21
2.2.4.6 Discussions	21
2.3 Model Development Issues	22
2.3.1 Inputs and Outputs	22
2.3.2 Data Range and Sample Distribution	23
2.3.3 Data Generation	24
2.3.4 Data Organization	24
2.3.5 Data Preprocessing	25
2.3.6 Weight Parameters Initialization	27
2.4 Neural Network Training	28
2.4.1 Error Derivative Computation	29
2.4.2 Related Discussions	30
2.4.3 Summary of Training Process	31
2.4.4 Learning Phenomena	32
2.4.5 Quality Measures	33
2.4.6 Training Algorithms	34
2.4.6.1 Backpropagation	34
2.4.6.2 Gradient Based Optimization Techniques	36
2.4.6.3 Conjugate Gradient Algorithm	37
2.4.6.4 Quasi-Newton Algorithm	37
2.4.6.5 Guass-Newton and Levenberg-Marquardt Algorithms	38
2.4.6.6 Global Optimization	39

2.4.7 ANN Model Development: Manual versus Automation	40
2.5 Review Examples	42
2.5.1 Neural Models for Components and Circuits	42
2.5.1.1 High-Speed Interconnect Network	42
2.5.1.2 CPW Symmetric T-junction	44
2.5.1.3 MESFET	45
2.5.2 Nonlinear Modeling Using Neural Networks	47
2.5.2.1 DC Models	47
2.5.2.2 Small Signal S-Parameter Models	47
2.5.2.3 Large-Signal Models	49
2.5.2.4 Neural Modeling Through a Known Equivalent Circuit	50
2.5.2.5 Circuit Models	50
2.5.3 ANN Based Circuit Design	53
2.5.3.1 CPW-Folded Double-Stub Filter	53
2.5.3.2 Three-Stage MMIC Amplifier	54
2.6 Summary	56

CHAPTER 3

Iterative Multi Stage Algorithm	57
3.1 Introduction	57
3.2 IMS Algorithm: Macro Level	59
3.2.1 Mathematical Description	60
3.2.2 Steps in Macro Level Process	63

3.2.3 Discussions	64
3.3 IMS Algorithm: Stage Level	67
3.3.1 General Description	67
3.3.2 Sub Neural Network Structures	67
3.3.3 Sub Neural Network Training	68
3.3.4 Algorithms for Stage Level Training	69
3.3.5 Huber Quasi-Newton Algorithm	69
3.3.6 Adaptive Controller	72
3.4 Demonstration Examples	73
3.4.1 Heterojunction Bipolar Transistor	73
3.4.2 MESFET	73
3.4.3 Circuit Waveform Model	75
3.4.4 Microstrip Filter	80
3.4.5 Eight Conductor Stripline	83
3.5 Summary	86

CHAPTER 4

Automatic Model Generation Algorithm	87
4.1 Introduction	87
4.2 Automation Issues: An Insight	90
4.3 AMG Algorithm	92
4.3.1 Notation	93
4.3.2 Key Aspects of the Algorithm	94

4.3.2.1 Automatic Sampling and Data Generation	95
4.3.2.2 Automatic Neural Network Structure Adaptation	99
4.3.2.3 Automatic Handling of Large Errors in Data	99
4.3.2.4 Overall Automation	100
4.3.2.5 Implementation Details	101
4.4 Demonstration Examples	104
4.4.1 MESFET	104
4.4.2 Embedded Capacitor	105
4.4.3 Power Amplifier	110
4.5 Summary	116

CHAPTER 5

Knowledge Based Automatic Model Generation Algorithm	117
5.1 Introduction	117
5.2 Data Generator	119
5.3 KAMG Algorithm	120
5.3.1 KAMG Notation	120
5.3.2 KAMG Formulation	121
5.3.2.1 Phase I: Generation of Coarse Neural Network Model	121
5.3.2.2 Phase II: Initialization of Fine Neural Network Model	123
5.3.2.3 Phase III: Refined Training of Fine Neural Network	129
5.3.2.4 Steps in KAMG Algorithm	130
5.4 Demonstration Examples	138

5.4.1 MOSFET	138
5.4.2 Embedded Resistor	140
5.4.3 Embedded Capacitor	143
5.4.4 Spiral Inductor	146
5.5 Summary	150
 CHAPTER 6	
Conclusions and Future Research	151
6.1 Conclusions	151
6.2 Future Directions	155
 Bibliography	 159

List of Figures

Figure 2.1. A physics-based FET to be modeled using a neural network [1].	15
Figure 2.2. Multi-layer perceptrons neural network structure. Typically, an MLP network consists of an input layer, one or more hidden layers, and an output layer.	18
Figure 2.3. Input/output scaling and de-scaling during neural network model development and during the usage of the trained neural model [1]. Here, $\tilde{\mathbf{y}}$ represents scaled outputs of neural network model and $\tilde{\mathbf{y}} \approx \tilde{\mathbf{d}}$ represents the de-scaled outputs of the neural network model.	27
Figure 2.4. Flow-chart demonstrating neural network training, neural model testing, and the use of training, validation, and test data sets in ANN modeling approach.	32
Figure 2.5. Flow diagram of the MLPNN model development process [7].	41
Figure 2.6. Electrical equivalent circuit of the interconnect configuration showing the connection of a source IC pin to four receiver pins [106].	43
Figure 2.7. Three-layered MLP structure with 18 inputs and 4 outputs used for modeling the interconnect network [106].	44

Figure 2.8. Comparison of small-signal S-parameter predictions from the large-signal MESFET neural network model (o, +, x, *) with those from physics-based <i>Khatibzadeh</i> and <i>Trew</i> MESFET model (—) [49].	46
Figure 2.9. Comparison of neural model (MLP 3-20-8) prediction with measured data.	49
Figure 2.10. Incorporation of a large-signal MESFET neural network model into a harmonic balance circuit simulator [49]. The ANN model can be included as an additional nonlinear subnetwork.	51
Figure 2.11. Structure of a typical recurrent neural network [69].	52
Figure 2.12. A 3-stage MMIC amplifier. During yield optimization, various components in the amplifier including MESFETs were substituted by neural network models [49].	54
Figure 3.1. Development of accumulated neural model and computation of training data for each training stage in the macro-level process of IMS algorithm.	65
Figure 3.2. Comparison between neural model prediction and training data for a HBT. The training data has occasional large errors, mostly due to extreme frequency samples. (a) Neural model trained by standard l_2 -method is affected by large errors. (b) Neural model trained by the HQN technique is not affected by large errors.	74

Figure 3.3. Comparison between neural model prediction and training data for a MESFET. The training data has a few large/gross errors. (a) Neural model trained by standard l_2 method is affected by large errors. (b) Neural model trained by HQN technique is not affected by large errors.	76
Figure 3.4. The circuit for the waveform modeling example with 3 transmission lines representing high-speed VLSI interconnects.	77
Figure 3.5. Training data and neural model prediction for the circuit waveform modeling example. (a) First stage of training, (b) Second stage of training, (c) Third stage of training and (d) Overall neural model.	79
Figure 3.6. Error measure C^k versus training stages in the circuit waveform modeling example.	80
Figure 3.7. A comparison between the neural network model and EM training data for the microstrip filter modeled using the proposed IMS algorithm. The filter neural model accurately represents the high-frequency behaviors with respect to changes in the filter's geometrical parameters. The response shown here corresponds to a filter with normalized values of w_d , g_{12} and g_{23} as 1.0, 1.5 and 2.0 respectively.	82
Figure 3.8. Eight-conductor stripline to be modeled for subsequent use in the design of multi-layered printed-circuit boards.	84

Figure 4.1. (a) Training and validation data in a typical sub-region of 2-dimensional input space. (b) When the sub-region is identified as the worst region, it is further divided into smaller regions. Newly generated training and validation data are indicated.	98
Figure 4.2. Systematic framework of the proposed automatic model generation algorithm in the form of a flow-chart.	103
Figure 4.3. Inter-twined distribution of training and validation data by the proposed AMG algorithm for the MESFET example. (a) First stage, (b) Second stage, (c) Third stage, and (d) Fourth stage of training. The algorithm identifies non-linear sub-regions of the MESFET input-space and automatically generates more samples in such regions.	107
Figure 4.4. Embedded capacitor used in multi-layer PCB. S-parameter neural model of the capacitor is developed from 3D-EM data of <i>Ansoft-HFSS</i> using the AMG algorithm.	108
Figure 4.5. Comparison between neural model prediction and training data for the embedded capacitor. The effect of large errors introduced in S_{12} training data during automatic data generation of the proposed AMG algorithm are discarded during neural network training of AMG as can be seen from the resulting neural model responses.	111

Figure 4.6. Circuit diagram of power amplifier used in wireless communication systems to be modeled using the AMG algorithm.	112
Figure 4.7. A 3-layer MLP neural network structure used by the proposed AMG algorithm to model dynamic behaviors of the power amplifier circuit.	113
Figure 4.8. Comparison of neural model prediction of the amplifier power outputs at fundamental, second and third harmonics (—) with original <i>Agilent-ADS</i> data (various symbols) for different values of P_{in} with $V_{din} = 3V$ and $R_L = 80 \Omega$.	115
Figure 4.9. Quantitative relationship between neural model accuracy and number of training data for the power amplifier circuit as provided by the proposed AMG algorithm.	115
Figure 5.1. Overall framework of KAMG algorithm showing various phases of neural network model development simultaneously utilizing coarse and fine data generators.	133
Figure 5.2. Neural network model development using the KAMG formulation for the difference method based neural network architecture resulting in the KAMG-DM. The portion within the dotted boundary represents the fine (overall) neural model for use in RF and microwave CAD.	134

Figure 5.3. Neural network model development using the KAMG formulation for the knowledge based neural network architecture resulting in the KAMG-KBNN. Part A of the sub network represents region and boundary layers, and Part B represents the gating network. The portion within the dotted boundary represents the fine (overall) neural model for use in RF and microwave CAD.	135
Figure 5.4. Neural network model development using the KAMG formulation for the prior knowledge input neural network architecture resulting in the KAMG-PKI. The portion within the dotted boundary represents the fine (overall) neural model for use in RF and microwave CAD.	136
Figure 5.5. Neural network model development using the KAMG formulation for the space mapped neural network resulting in KAMG-SMNN. The portion within the dotted boundary represents the fine (overall) neural model for use in RF and microwave CAD.	137
Figure 5.6. Accuracy comparison of MOSFET neural models generated by KAMG and earlier AMG algorithms. All the KAMG neural models achieved better accuracies with fewer fine data as compared to MLP neural models from AMG.	139
Figure 5.7. Accuracy comparison of embedded resistor neural models generated by KAMG and AMG algorithms. When only a few fine data are available, all the KAMG neural models achieved better accuracies than the MLP neural models from AMG.	142

Figure 5.8. 3D-EM representation of the embedded capacitor to be modeled using KAMG algorithm.	144
Figure 5.9. Accuracy comparison of capacitor neural models generated by KAMG and AMG algorithms. All the KAMG neural models achieved better accuracies with fewer fine data as compared to MLP neural models from AMG.	145
Figure 5.10. 3D-EM representation of the spiral inductor to be modeled using KAMG algorithm.	146
Figure 5.11. Accuracy comparison of 2-loop spiral inductor neural models generated by KAMG and AMG algorithms. All the KAMG neural models achieved better accuracies with fewer fine data as compared to MLP neural models from AMG.	147
Figure 5.12. Accuracy comparison of 8-loop spiral inductor neural models generated by KAMG and AMG algorithms. All the KAMG neural models achieved better accuracies with fewer fine data as compared to MLP neural models from AMG.	149

List of Tables

Table 2.1. Average test errors of S-parameter neural models for different transistor sizes.	48
Table 2.2. Summary of CPU-time comparison between original physics-based and neural network based CAD operations in the 3-stage amplifier design [1][49].	55
Table 3.1. Selection of neural network structure and training algorithm for the training stages with different objectives in stage-level process of IMS algorithm.	69
Table 3.2. Different situations encountered in a training stage and appropriate actions taken by the adaptive-controller of stage-level process in the proposed IMS algorithm.	72
Table 3.3. Average and worst-case errors between neural model prediction and test data for HBT example. Data set A has small and large measurement errors. Data set B is same as set A, except that the large errors are identified and removed.	74
Table 3.4. Stage-wise training results of the circuit-waveform modeling example using the proposed IMS algorithm.	78
Table 3.5. Stage-wise training results for the microstrip filter example using the proposed IMS algorithm.	83

Table 3.6. Stage-wise training results of the eight-conductor stripline modeling example using the proposed IMS algorithm.	85
Table 4.1. Model accuracy comparison between the proposed AMG algorithm based on automatic sampling and the step-by-step approach based on conventional grid distribution for the MESFET. Model accuracy reported for the manual approach is an average validation error of three neural models with 9, 12 and 16 hidden neurons.	106
Table 4.2. Comparison of the time taken by the proposed AMG algorithm and manual step-by-step neural modeling approach for the embedded capacitor.	110
Table 5.1. Comparison of fine data needed by various neural modeling techniques to achieve MOSFET models with 0.50% validation error.	140
Table 5.2. Accuracy comparison between MOSFET neural models developed by various modeling techniques using 454 fine data.	141
Table 5.3. Accuracy comparison between resistor neural models developed by various modeling techniques using 18 fine data.	143
Table 5.4. Comparison of fine data and CPU-time needed by various neural modeling techniques to achieve capacitor models with 1% validation error.	145
Table 5.5. Accuracy comparison between 2-loop inductor neural models developed by various modeling techniques using 386 fine data.	148

Table 5.6. Comparison of fine data and CPU-time needed by various neural modeling techniques to achieve 8-loop inductor models with 1% validation error. 149

Glossary of Terms

AMG	Automatic model generation
ANN	Artificial neural networks
BP	Backpropagation
CAD	Computer aided design
CPU	Central processing unit
DM	Difference method
DOE	Design of experiments
EBP	Error backpropagation
EM	Electromagnetics
HB	Harmonic balance
HQN	Huber quasi-Newton
IMS	Iterative multi stage
KAMG	Knowledge based automatic model generation
KBNN	Knowledge based neural networks

LM	Levenberg-Marquardt
MLP	Multi-layer perceptrons
MLPNN	Multi-layer perceptrons neural network
PCB	Printed circuit board
PKI	Prior knowledge input
RBF	Radial basis function
RNN	Recurrent neural networks
SMNN	Space mapped neural networks
SOM	Self-organizing map
VLSI	Very large scale integration

CHAPTER 1

Introduction

1.1 Background

Rapid developments in telecommunication, VLSI and wireless industries over the last decade have led to dramatic increase in circuit complexities and signal frequencies. The drive of the hi-tech industry towards lowering manufacturing costs and shortening time-to-market cycles has put a heavy onus on researchers to engineer robust methodologies for the design of high-frequency electronics.

Historically, RF and microwave circuit design was carried out using traditional design procedure [1]. While there is no precise definition, the broad understanding of the phrase “traditional procedure” was to say there is no involvement of computer aided design (CAD) methods or software tools. In this procedure, a designer starts with a set of user-specifications and arrives at a design solution through a series of repetitive steps often based on experience, empirical knowledge and measurements. Steps involved in this approach include initial design, fabrication of a laboratory prototype, measurements, and adjustment of design parameters for fine-tuning the performance. This approach became extremely difficult and impractical with ever-increasing circuit complexities and tightened design tolerances.

In due course, CAD approach has emerged as a strong alternative to this expensive,

human-intensive and time-consuming traditional design procedure. This approach has progressed considerably in the recent years [2], thanks to significant research contributions in the form of simulation techniques and software tools, e.g. Electromagnetic (EM) and physics based simulation tools for passive components and active devices respectively. However, the time-consuming nature of theoretical EM/physics simulations makes high-level microwave-CAD operations (e.g. *Monte Carlo* analysis and yield optimization) prohibitively expensive in terms of CPU-time, leading to the following directions. (i) Development of accurate and fast RF and microwave-oriented simulation tools or representative models is essential for robust high-frequency CAD. (ii) Investigation of algorithms that can lead to efficient utilization of existing EM/physics simulation tools can significantly enhance RF and microwave CAD.

Modeling still remains a major bottleneck for the successful CAD of high-frequency components and circuits. A variety of modeling approaches are available for active devices, passive components and nonlinear circuits. These include detailed theoretical models, empirical and equivalent circuit models, polynomial models and table-lookup models [1][3]. Detailed models are accurate but can be slow as they involve numerical simulations based on extensive theory. Empirical and equivalent circuit models are fast, but have limited accuracies. Polynomial models exhibit limited degree of nonlinearity and hence not suited for RF/microwave modeling. Table-lookup models work fairly well for low-dimensional problems but data storage can be cumbersome in high-dimensional cases. As such, there is immense scope for the development of advanced high-frequency modeling approaches.

1.2 Motivation

Artificial neural networks (ANN) recently gained recognition as an unconventional alternative at various levels of RF and microwave CAD including modeling, simulation and optimization. A neural network is a mathematical model typically consisting of a number of smooth switch functions (e.g. sigmoid functions) and has the ability to learn and generalize arbitrary continuous multi-dimensional nonlinear input-output relationships. ANN can be trained to accurately learn RF/microwave characteristics of active devices, passive components and nonlinear circuits. A trained neural network, also referred to as neural network model, can be used in place of CPU-intensive detailed models during CAD, to provide instant answers to the task it has learned [4]. It is this unique combination of “accuracy” and “speed” that makes ANN an efficient and viable modeling approach.

ANN-based RF/microwave CAD is currently an active topic for both advanced research and industrial applications. In general, ANN-based CAD approach involves the following steps.

- Identification of components in a given circuit whose computational models are expensive in terms of CPU-time.
- Generation of input-output samples called training data for each of these components using detailed theoretical models or measurements.
- Training neural networks to accurately learn each of the component’s behaviors using the corresponding set of training data.
- Building the circuit schematic in a simulator environment using the simulator’s library components together with neural models of selected components.

- Finally, carrying out efficient circuit design using a mix of neural and non-neural models.

From the above discussions, it is clear that neural network modeling is the first and foremost step toward neural-based high-frequency CAD. As this emerging technology expands from university research into practical RF and microwave applications, new challenges are recognized. RF/microwave data from simulation and measurement has unavoidable errors [5]. While most of these errors could be small (e.g. due to equipment tolerances), there could be some accidental large errors as well. Training neural networks with erroneous training data results in unreliable neural models. RF/microwave behaviors of certain components exhibit sudden variations with changes in input parameter values, and capturing these behaviors from data could be difficult for neural network training. ANN modeling involving a large number of input and output parameters requires massive amounts of training data, requiring CPU-expensive data generation and neural network training.

Neural model development involves sub-tasks such as data generation, neural network selection, training and validation [6][7]. The ANN model development process requires human experience in terms of decisions regarding neural network size, number of training data, sample distribution in model input space, training algorithm selection etc. RF and microwave designer's wish to efficiently develop neural models overcoming these hurdles necessitates continued research and developmental activities in the strategic area of ANN-modeling and neural-based high-frequency CAD. In conclusion, all the above-mentioned challenges and practical aspects are strong motivations for this thesis work.

1.3 Contributions of the Thesis

The subject of the thesis is to investigate algorithms for advanced RF and microwave oriented neural network modeling. The main objectives are, (a) To develop robust neural network modeling techniques that can address practical challenges in RF and microwave modeling, e.g. presence of errors and sharp variations in training data, and (b) To formulate efficient neural network modeling algorithms that can facilitate automatic generation of RF and microwave neural network models of highest accuracies at lowest cost as compared to any other existing neural modeling techniques. Accomplishing these objectives can be significant for high-frequency CAD in many ways. Firstly, ANN-modeling can be cost-effective and practical for simple as well as complicated RF/microwave modeling problems. Secondly, ANN-based RF/microwave CAD technology can attain higher levels of maturity and can greatly assist the high-frequency designers as a viable alternative to existing CAD approaches through combined benefits of both accuracy and speed. Specifically, in view of the above-mentioned objectives, the contributions of this thesis are summarized as follows.

- An Iterative multi stage (IMS) algorithm [5][8] including a macro-level process and a stage-level process is developed to simplify highly nonlinear and multi-dimensional RF/microwave ANN-modeling problems. Through iterative decomposition and multi-stage training the proposed IMS algorithm maximally combines the merits of a variety of neural network structures (including both conventional and knowledge-based neural networks) and the strengths of different training algorithms to model a given complicated RF/microwave problem.

- An Automatic model generation (AMG) algorithm [9][10] is created for automating the neural-network-based RF/microwave model development process. Starting with zero amounts of training data and then proceeding with neural network training in a stage-wise manner, the AMG can automatically generate a neural model that exceeds user-specified accuracy. For the first time, the proposed AMG provides error-based adaptive sampling of training and validation data sets in an inter-twined fashion, and automatic neural network structure adaptation based on ANN learning phenomena.
- To accomplish dynamic data generation within the AMG algorithm, a simulator driver framework [11] has been established. The framework allows automatic driving of simulation tools, e.g. *OSA90* [12], *Ansoft-HFSS* [13], *Agilent-ADS* [14] etc.
- In order to deal with neural network training in the presence of gross errors in training data in various stages of the IMS and the AMG algorithms, a Huber quasi-Newton (HQN) training technique [15] is incorporated.
- A Knowledge-based automatic model generation (KAMG) algorithm [16][17] aimed at automatically generating RF/microwave neural models of highest accuracies using fewest data as compared to other ANN-modeling techniques is formulated. The algorithm is comprehensively derived to integrate powerful concepts, namely, automatic model generation, knowledge neural networks and space mapping. For the first time, the KAMG algorithm utilizes extensive coarse data together with minimal amounts of fine data to generate neural models that accurately match fine data.

- An object-oriented implementation of the AMG and the KAMG algorithms is accomplished in C++, incorporating various neural-network-modeling aspects including data generation, neural network training, neural network size adjustment etc. This computer program has been used in deriving the results in this thesis and incorporated into the trial version of *NeuroModeler* software [18].

1.4 Organization

The thesis is organized as follows.

In Chapter 2, an overview of ANN-based RF and microwave modeling including literature review is presented. Problem statement of neural-network-based RF/microwave modeling is defined. Various aspects involved in neural-network-modeling are described from RF/microwave perspective. Commonly used neural network structures, training algorithms, data sampling and scaling strategies, and various key issues in automation of ANN-modeling are described. Neural modeling of high-frequency components and neural-based circuit design are illustrated through review examples.

Practical challenges in RF/microwave modeling are summarized in Chapter 3. The proposed IMS algorithm including both macro-level decomposition and stage-level ANN-training is presented. Concepts of sub neural network and overall neural network are introduced. Huber-norm based quasi-Newton training algorithm is described along with the formulation of the training objective and its relevance to the IMS algorithm. Advantages of the IMS algorithm are demonstrated through practical RF/microwave

examples such as the HBT and a microstrip filter.

The overall framework of the proposed AMG algorithm is presented in Chapter 4. Neural network learning phenomenon such as over-learning and under-learning that form a basis for the AMG are described. Formulations for error-based sampling of training data during various stages of AMG are presented. Advantages of the AMG algorithm relative to conventional neural modeling approaches (e.g., manual grid-based neural modeling) are illustrated through examples of MOSFET and embedded passives.

In Chapter 5, the concept of coarse and fine data generators is introduced. A comprehensive framework of the proposed KAMG algorithm including various phases, namely, coarse neural model generation, fine (overall) neural model initialization, and fine (overall) neural model refinement is presented. Formulations of the KAMG for advanced knowledge neural network architectures including difference method (DM), knowledge-based neural networks (KBNN), prior knowledge input (PKI) method, and space mapped neural networks (SMNN) are presented. Advantages of the proposed KAMG algorithm over the existing neural modeling approaches are demonstrated through neural modeling examples of MOSFET, embedded resistor and capacitor, and spiral inductor.

Conclusions of the thesis are presented in Chapter 6 explaining how the thesis objectives have been successfully achieved. Recommendations for future research toward accomplishing end-to-end neural based RF and microwave CAD solutions are also made.

CHAPTER 2

Artificial Neural Networks for RF and Microwave Design: An Overview

2.1 Introduction

Artificial neural networks are information processing systems inspired by the ability of human-brain to learn from observations and to generalize by abstraction [1]. ANN can be trained with given input-output information through a learning process involving storage of such information in the form of synaptic weights of the network. The fact that neural networks can learn arbitrary, continuous, multi-dimensional, and nonlinear input-output relationships from corresponding data has resulted in their use in diverse areas of engineering such as bio-medical [19], control engineering [20], pattern recognition [21], and speech processing [22].

In recent years, a computer aided design approach based on ANN has been introduced for RF/microwave modeling, simulation and optimization [2]. Fast and accurate neural network models can be developed from measured or simulated RF/microwave data through a learning process called training. Resulting neural network models, also referred to as neural models, can be used in place of CPU-intensive EM/physics models of passive/active components to speed-up RF/microwave design [2]-[4]. Neural networks have been used to model a wide variety of RF/microwave

components and circuits such as transmission line components [23]-[33], bends [34][35], vias [6][36]-[39], CPW components [40][41], spiral inductors [7][42], FETs [3][43]-[52], HBTs [53][54], HEMTs [55][56], waveguides [57], laser diodes [58], embedded resistors [59], filters [60]-[65], amplifiers [66][67], mixers [68]-[70] and antennas [71]-[82].

Neural networks have been used for microwave circuit design (e.g. filters [33][62][64], CPW circuits [40], stabilizing circuits [83] etc.), high-speed interconnect optimization [25][31], electronic packaging [27], synthesis [66][84], object recognition [85][86], wave propagation [87][88], impedance matching [89][90], inverse modeling [91]-[93] and yield prediction [47][52]. Neural models are much faster than detailed theoretical EM/physics models [1][49]; more accurate than polynomial, empirical, and equivalent circuit models [94]; allow more dimensions than table-lookup models [95]; and are easier to develop when a new device/technology is introduced [25].

A variety of neural network structures have been used for RF/microwave applications including feedforward neural networks, e.g. multi-layer perceptrons (MLP) neural networks [96], Radial Basis Function (RBF) networks [97], wavelet networks [63] etc., and neural networks with feedback, e.g. Recurrent Neural Networks (RNN) [69]. Research in the RF/microwave-ANN area recently led to the creation of state-of-the-art knowledge-neural-network structures, which advocate incorporation of existing engineering knowledge in the form of empirical equations/models or equivalent circuits into the neural network internal structure. Neural networks with knowledge including knowledge based neural networks [3], difference method [6], prior knowledge input

network [37], and space mapped neural networks [35] have been shown to outperform conventional neural network structures (without knowledge) in high-frequency modeling and design. In this chapter, structure of a MLP neural network is described in detail.

The process of neural model development involves various issues such as sampling and generation of training data [6], data pre-processing (e.g. scaling [65]), and ANN weight initialization [97]. The present trend is to manually deal with each of these issues separately, i.e., one issue at a time. Such an approach referred to as the step-by-step neural modeling approach requires considerable human involvement. In this chapter, a detailed description of the above-mentioned fundamental issues is presented from RF/microwave perspective. For example, sampling strategies used to determine the distribution of training data in neural-model input-parameter space, methods of input and output scaling, and various techniques for weight initialization are covered.

Neural network training is the most crucial step in the neural model development process. Training involves modification of neural network internal parameters or synaptic weights in an orderly fashion. Such modification is carried out until the input-output relationship from corresponding training data is satisfactorily learnt. Neural network training is carried out using various optimization techniques that can be classified into gradient-based and non-gradient-based algorithms. Commonly used gradient-based algorithms in the RF/microwave CAD area include backpropagation (BP) [98][99], conjugate-gradient [100], quasi-Newton [101] and Levenberg-Marquardt [102]. Examples of non-gradient-

based algorithms include simplex method [103], simulated annealing [104] and genetic algorithms [105]. Some of these training algorithms are reviewed in this chapter.

Computational approaches available in the form of software tools helped to advance RF/microwave circuit CAD from theory to application [2]. From the designer's standpoint, the efficiency of CAD tools depends largely on the accuracy and speed of component models. Consequently, component/device modeling is an important aspect of RF/microwave CAD and neural networks are flexible vehicles in this direction. Development of fast and accurate neural models has been demonstrated for active devices [3], high-speed interconnects [25] and passive components [6]. One category of ANN-based RF/microwave modeling is the use of feedforward networks (e.g., MLP and RBF) to learn static input-output relationships. Examples of this category reviewed here include ANN-modeling of CPW T-junction [40] and interconnect networks [106].

Another category of ANN-based modeling is the RF/microwave modeling of nonlinear devices/circuits and their dynamic behaviors. Firstly, consider the transistor-modeling problem. Traditional approaches for transistor (device) modeling are based on lumped equivalent circuits, e.g. [107][108]. The equivalent circuit approach involves determination of an equivalent circuit topology and formulation of the circuit elements. Such an approach could require experience and involves a tedious trial-and-error process. Neural network approaches have been used for modeling of transistor DC [32][46], small-signal [4][53] and large-signal [49][56] behaviors. Various ANN-based transistor-modeling approaches are reviewed in this chapter. Secondly, consider modeling of

dynamic characteristics of nonlinear circuits. A recent work [69] that advocates the use of RNN for dynamic modeling of circuits is also reviewed.

Neural models of RF/microwave components can subsequently be used in circuit design and optimization. In order to accomplish this, component neural models are first uploaded into the circuit simulators, e.g. *Agilent-ADS* [14], using corresponding plug-in software tools, e.g. Carleton University's *NeuroADS* [109]. Designers can then utilize these neural models just as any other non-neural models available in the simulator's library. An ANN model can be connected with other ANN models or with any other models in the simulator environment to construct a circuit schematic. Circuit optimization examples [40][49] utilizing fast and accurate neural models are reviewed in this chapter.

2.2 Neural Network Structures

2.2.1 Basic Components

A typical neural network structure has at least two basic components, namely, the processing elements and the interconnections between them [1]. The processing elements are called neurons and the connections between the neurons are known as links. The principal task of a neuron is to process information, and is therefore characterized by a mathematical function called neuron activation function. Every link has a weight parameter associated with it. Each neuron receives stimulus from other neurons connected to it, processes the information, and produces an output. Neurons that receive stimuli from outside the network are called input neurons while neurons whose outputs

are externally used are called output neurons. Neurons that receive stimuli from other neurons and whose outputs are stimuli for other neurons in the network are known as hidden neurons. Different neural network structures can be constructed by using different neurons (i.e., neuron activation functions) and by connecting them differently [97].

2.2.2 Concept of a Neural Network Model

Let x be an n -vector containing the parameters of a given device or circuit. Let y be an m -vector containing the RF/microwave responses of the device or circuit under consideration. As an example, consider the FET shown in Figure 2.1. Any change in the values of geometrical/physical parameters (inputs) of the FET affects its electrical responses (outputs) such as the S-parameters. Inputs and outputs of the FET are given by

$$x = [L \quad W \quad H \quad N_d \quad V_{gs} \quad V_{ds} \quad f]^T \quad (2.1)$$

and

$$y = [MS_{11} \quad PS_{11} \quad MS_{12} \quad PS_{12} \quad MS_{21} \quad PS_{21} \quad MS_{22} \quad PS_{22}]^T, \quad (2.2)$$

where MS_{ij} and PS_{ij} represent magnitude and phase of the S-parameter S_{ij} .

The original FET input-output relationship can be expressed as

$$y = f(x), \quad (2.3)$$

where f represents the detailed theoretical FET representation, e.g. physics-based semiconductor equations. While f accurately represents the FET's input-output characteristics, it can be expensive in terms of CPU-time when used repetitively during CAD.

Gate length:	L
Gate width:	W
Channel thickness:	H
Doping density:	N_d
Bias:	V_{gs}, V_{ds}
Frequency:	f

Figure 2.1. A physics-based FET to be modeled using a neural network [1].

An accurate and fast neural network model that can represent the FET's input-output relationship is to be developed. Consider a neural network $h(\mathbf{x}, \mathbf{w})$ with n input neurons and m output neurons, where $\mathbf{w}$ is an N_w -vector containing all the trainable or adjustable weight parameters in the neural network. Corresponding neural network input-output relationship or neural network model of the FET is given by

$$\tilde{\mathbf{y}} = h(\mathbf{x}, \mathbf{w}), \quad (2.4)$$

where $\tilde{\mathbf{y}}$ is an m -vector containing the neural network outputs. The definition of $\mathbf{w}$, and the manner in which $\tilde{\mathbf{y}}$ is computed from $\mathbf{x}$ and $\mathbf{w}$, determine the structure of the neural network [97].

The neural network model in (2.4) can represent the theoretical FET behavior in (2.3), only after learning the $\mathbf{x}$ - $\mathbf{y}$ relationship f through a process called training. As such,

several (x, y) samples called training data need to be generated either from FET's physics-based simulation software (*i.e.*, simulator) or from measurements, prior to training. The objective of neural network training is to adjust neural network weights w such that the neural model outputs satisfactorily match the training data outputs [97][98]. A trained neural network model can be used during online RF/microwave design to provide instant answers to the task it has learned. For example, the FET neural network model can be used for fast estimation of S-parameters corresponding to different geometrical/physical parameter values.

2.2.3 Neural Network versus Conventional Approaches

In this section, neural network approach to RF/microwave modeling is compared with conventional modeling approaches. The first approach is the detailed modeling approach, where in-depth theory/equations of a given problem can be used as a computational model. For example, EM simulation tools [13][110] can be used as detailed models of passive microwave components. Detailed models are highly accurate but can be computationally intensive [2]. The second approach is the approximate modeling approach, which uses either empirical equations or equivalent circuits as models. These models are developed using a mixture of simplified component theory, heuristic interpretation and representation, and/or fitting of experimental data. Examples of approximate models include empirical equations of transmission lines [111] and MESFETs [112]. Approximate models are much faster than the detailed models. However, they can be limited in terms of accuracy and the range of input parameter space over which they are reliable. Neural network approach possesses the advantages of both

detailed and approximate models, i.e., neural models can be as accurate as detailed models and as fast as approximate models.

2.2.4 Multi-Layer Perceptrons Neural Networks

2.2.4.1 Structure and Notation

Multi-layer perceptrons (MLP) neural network [93][96] is a popularly used feedforward ANN structure. Typically, an MLP neural network consists of an input layer, one or more hidden layers, and an output layer, as shown in Figure 2.2. For example, an MLP neural network with one hidden layer is referred to as 3-layered MLP or MLP3. Suppose the total number of layers is L . The 1st layer is the input layer, the L^{th} layer is the output layer, and layers 2 to $L - 1$ are hidden layers. Let the number of neurons in l^{th} layer be N_l . Let w_{ij}^l represent the weight of the link between j^{th} neuron of $l-1^{th}$ layer and i^{th} neuron of l^{th} layer. Let x_i represent the i^{th} external input to the MLP network and z_i^l be the output of i^{th} neuron of l^{th} layer. There is an additional weight parameter for each neuron, w_{i0}^l , representing the bias for i^{th} neuron of l^{th} layer. As such, $\mathbf{w}$ of MLP includes w_{ij}^l , $j=0,1, \dots, N_{l-1}$, $i=1,2, \dots, N_l$, $l=2,3, \dots, L$, i.e., $\mathbf{w} = [w_{10}^2 \ w_{11}^2 \ w_{12}^2 \ \dots \ w_{N_L N_{L-1}}^L]^T$. The weight parameters are real numbers, which are initialized before MLP training. During training, they are iteratively updated in a systematic manner [97][98]. Once training is completed, vector $\mathbf{w}$ remains fixed while neural network is being used as a neural model.

Figure 2.2. Multi-layer perceptrons neural network structure. Typically, an MLP network consists of an input layer, one or more hidden layers, and an output layer.

2.2.4.2 Anatomy of Neurons

In the MLP network, each neuron processes the stimuli (inputs) received from other neurons. The processed information is available at the output end of the neuron. For example, every neuron in the l^{th} layer receives stimuli from the neurons of $l-1^{th}$ layer, i.e., $z_1^{l-1}, z_2^{l-1}, \dots, z_{N_{l-1}}^{l-1}$. A typical i^{th} neuron in the l^{th} layer processes this information in two

steps [25]. Firstly, each of the inputs is multiplied by the corresponding weight parameter and the products are added to generate a weighted sum γ_i^l , i.e.,

$$\gamma_i^l = \sum_{j=0}^{N_{l-1}} w_{ij}^l z_j^{l-1}. \quad (2.5)$$

Assuming a fictitious neuron $z_0^{l-1} = 1$ in the $l-1^{th}$ layer, the effect of bias parameter (w_{i0}^l) can be included. Secondly, the weighted sum in (2.5) is processed by the neuron's activation function $\sigma(\cdot)$ to produce the neuron output $z_i^l = \sigma(\gamma_i^l)$. This output can, in turn, become the stimulus to neurons in the $l+1^{th}$ layer.

Commonly used hidden neuron activation function is the sigmoid function given by

$$\sigma(\gamma) = \frac{1}{(1+e^{-\gamma})}. \quad (2.6)$$

Other neuron activation functions such as arctangent and hyperbolic-tangent can also be used. These are smooth switch functions that are bounded, continuous, monotonic and continuously differentiable [65]. Input neurons use relay activation function and simply relay the external stimuli to the first hidden layer (or second layer) neurons, i.e., $z_i^1 = x_i$, $i = 1, 2, \dots, n$. In the case of RF/microwave modeling, where the purpose is to model continuous behaviors, a linear activation function can be used as the output neuron activation function, i.e.,

$$\sigma(\gamma_i^L) = \gamma_i^L = \sum_{j=0}^{N_{L-1}} w_{ij}^L z_j^{L-1} \quad (2.7)$$

2.2.4.3 Feedforward Computation

Given the input vector $\mathbf{x} = [x_1 \ x_2 \ \dots \ x_n]^T$ and the weight vector $\mathbf{w}$, neural network feedforward computation [7] is a process used to compute the output vector $\tilde{\mathbf{y}} = [\tilde{y}_1 \ \tilde{y}_2 \ \dots \ \tilde{y}_m]^T$. Feedforward computation is useful both during neural network training and while using the neural model. Throughout feedforward computation, weights $\mathbf{w}$ remain fixed. The computation involves a set of formulae [113] given by

$$z_i^1 = x_i, \quad i = 1, 2, \dots, N_1, \quad n = N_1, \quad (2.8)$$

$$z_i^l = \sigma \left(\sum_{j=0}^{N_{l-1}} w_{ij}^l z_j^{l-1} \right), \quad i = 1, 2, \dots, N_l, \quad l = 2, 3, \dots, L, \quad (2.9)$$

and

$$\tilde{y}_i = z_i^L, \quad i = 1, 2, \dots, N_L, \quad m = N_L. \quad (2.10)$$

2.2.4.4 Important Features

The universal approximation theorem [114] states that there always exists a 3-layer MLP neural network that can approximate an arbitrary, continuous, multi-dimensional, nonlinear function to any desired accuracy. This forms a theoretical basis for employing MLP neural networks to approximate RF/microwave behaviors that can be functions of bias/geometrical/physical parameters. MLP neural networks are distributed models, i.e., no single neuron can produce the overall $\mathbf{x}$ - $\mathbf{y}$ relationship. For a given $\mathbf{x}$, some neurons are switched on, some are off, and others are in transition. It is this combination of neuron switching states that enables the MLP to represent a given input-output mapping. During training, MLP's weight parameters capture or encode the problem information from the corresponding $(\mathbf{x}, \mathbf{y})$ training data.

2.2.4.5 Network Size and Layers

The universal approximation theorem does not specify as to what should be the size of the MLP network. The precise number of hidden neurons required for a given microwave-modeling problem remains an open question [115]. There is no clear-cut answer, however, the number of hidden neurons depends upon the degree of non-linearity of f , and the dimensionality of x and y (i.e., values of n and m). Highly nonlinear problems need more hidden neurons and smoother problems need fewer neurons [1]. Traditionally, either experience or trial-and-error has been used to settle on a reasonable number of hidden neurons. There has been significant research in the neural network area to determine proper network size, e.g. constructive algorithm [116], network pruning [117], regularization [118], etc. MLP networks with one or two hidden layers (i.e., 3-layered or 4-layered MLP networks) are commonly used for RF/microwave applications. If generalization capability of the neural network model is a major concern, MLP3 is preferred over MLP4 [119].

2.2.4.6 Discussions

Selection of neural network structure for a given application depends upon the nature of the x - y relationship. Non-dynamic modeling problems or problems converted from dynamic to non-dynamic form using techniques like harmonic balance [120], can be solved using feedforward networks. Most popular choice is the MLP, since its structure and training are well established. RBF networks are used when the problem exhibits highly nonlinear and localized phenomena, e.g. sharp variations [8]. Time-domain dynamic responses such as those in nonlinear modeling can be represented using RNN

[69]. Knowledge neural networks (e.g. [3][6]) that utilize existing engineering knowledge for efficient ANN-modeling can be used when such knowledge is available.

Feedforward networks MLP and RBF behave differently owing to the different nature of their hidden neuron activation functions. MLP networks construct global approximations to nonlinear x - y mapping. Consequently, they are capable of generalizing in those regions of the input space, where a little or no training data is available [97]. On the contrary, RBF networks use exponentially decaying localized nonlinearities to construct local approximations to nonlinear x - y mapping. As a result, RBF networks are capable of faster learning and exhibit reduced sensitivity to the order of presentation of training data [98]. It is stated in [121] that RBF networks are suited for problems with fewer input parameters.

2.3 Model Development Issues

2.3.1 Inputs and Outputs

The first step towards developing a neural network model is the identification of inputs (x) and outputs (y). The output parameters are determined based on the purpose of the neural network model. For example, real and imaginary parts of S-parameters are selected for linear circuit component models, currents and charges are used for large-signal device models, and cross-sectional RLCG parameters are chosen for interconnect models [113]. Neural model inputs are those device/circuit parameters (e.g. bias, geometrical and physical parameters) that affect the output parameters of interest. Geometrical parameters and frequency are selected for EM-based passive component

modeling, and geometrical and electrical parameters (*e.g.*, bias) are chosen for active device modeling [65].

2.3.2 Data Range and Sample Distribution

The next step is to define the range of training and test data to be used in ANN model development and the distribution of the data samples within that range. Consider that the range of the input space in which the neural network model would be used after training is $[x_{\min}, x_{\max}]$. Training data is sampled slightly beyond the neural model utilization range, *i.e.*, $[x_{\min} - \Delta, x_{\max} + \Delta]$, in order to ensure reliability of the neural model at the boundaries of that range. Test data is generated in the range $[x_{\min}, x_{\max}]$.

Once the range of model input parameters is finalized, a sampling distribution needs to be chosen. Commonly used sample distributions include uniform and non-uniform grid distributions [1], star distribution [35], and central-composite distribution [6]. In uniform grid distribution, each input parameter x_i is sampled at equal intervals. In non-uniform grid distribution, on the other hand, each input parameter is sampled at unequal intervals. The latter is useful when the problem behavior is highly nonlinear in certain sub-regions of the x -space, thus requiring dense sampling in such sub-regions. Modeling of DC characteristics (*i.e.*, I - V curves) of an FET is a typical example for non-uniform grid distribution.

In the case of star distribution, a mid-point of the input space x_{mid} is first determined. Samples are then obtained by perturbing one element of x_{mid} at a time, either towards its

minimum value or towards its maximum value. Total number of samples in star distribution is $2n+1$ [35]. In central-composite distribution based on design of experiments (DOE) methodology, 2^n corner points are generated in addition to those of the star distribution. Star and central-composite distributions are used in situations where data generation is expensive [1].

2.3.3 Data Generation

The $(\mathbf{x}, \mathbf{y})$ sample pairs can be generated using either a simulation-software (e.g. 3D-EM simulations using *HFSS*) or a measurement setup (e.g., S-parameter measurements from a network analyzer). The generated data can be used for training the neural network and for testing the resulting neural model. In practice, both simulations and measurements can have small errors. While errors in simulation can be due to truncation/round-off or non-convergence, errors in measurement can be due to equipment limitations or tolerances [5]. Considering these, a vector $\mathbf{d}$ is introduced to represent the outputs from simulation or measurement corresponding to an input $\mathbf{x}$. Data generation is carried out for obtaining sample pairs $(\mathbf{x}_p, \mathbf{d}_p), p = 1, 2, \dots, N_p$, where N_p denotes the total number of samples.

2.3.4 Data Organization

The generated $(\mathbf{x}, \mathbf{d})$ sample pairs could be divided into three sets, namely, training data, validation data, and test data. Let T , V , T_e and D represent index sets of training data, validation data, test data and overall (or available) data respectively. Training data is utilized to guide the training process i.e., to update the neural network weight parameters

during training [98]. Validation data is used to monitor the quality of the neural network model during training and to help decide when to stop the training process. Once training is completed, test data can be used to independently evaluate the quality of the neural model. Ideally, each of the data sets T , V and T_e must adequately represent the given RF/microwave modeling problem $y = f(x)$. In practice, splitting of the available data depends upon the size of D . When D is sufficiently large, it can be split into three mutually disjoint sets. If D is limited due to expensive simulations or measurements, it can be split into just two sets. One of the sets is used for training and validation ($T = V$) and the other for testing (T_e). Alternatively, one of the sets is used for training (T) and the other for validation and testing ($V = T_e$) as stated in [1].

2.3.5 Data Preprocessing

Contrary to binary data (0's and 1's) in pattern recognition applications [21], the orders of magnitude of various input (x) and output (d) parameter values in RF/microwave applications can be very different from one another. As such, a systematic preprocessing of training data called scaling is desirable for carrying out efficient neural network training by preventing neuron saturation [7][47]. Let x , $x_{\min}$ and $x_{\max}$ represent a generic input element in the input vectors x , $x_{\min}$ and $x_{\max}$ of original (generated) data respectively. Similarly let $\tilde{x}$, $\tilde{x}_{\min}$ and $\tilde{x}_{\max}$ represent a generic element in the input vectors $\tilde{x}$, $\tilde{x}_{\min}$ and $\tilde{x}_{\max}$ of scaled data, where $[\tilde{x}_{\min}, \tilde{x}_{\max}]$ is the input parameter range after scaling. Linear scaling is given by

$$\tilde{x} = \tilde{x}_{min} + \frac{x - x_{min}}{x_{max} - x_{min}} (\tilde{x}_{max} - \tilde{x}_{min}), \quad (2.11)$$

and the corresponding de-scaling is given by

$$x = x_{min} + \frac{\tilde{x} - \tilde{x}_{min}}{\tilde{x}_{max} - \tilde{x}_{min}} (x_{max} - x_{min}). \quad (2.12)$$

Elements in output vector $\mathbf{d}$ can also be scaled similarly. Linear scaling of data inputs and outputs can balance different parameter values [113]. Input/output scaling and de-scaling during neural network model development and use are shown in Figure 2.3.

Logarithmic scaling [65] can be applied to outputs with large variations in order to provide a balance between small and large magnitudes of the same output in different sub-regions of the x -space. Logarithmic scaling is given by

$$\tilde{x} = \ln(x - x_{min} + 1), \quad (2.13)$$

and the corresponding de-scaling is given by

$$x = e^{\tilde{x}} + x_{min} - 1. \quad (2.14)$$

In the training of knowledge-based neural networks [3][4], where knowledge neuron functions (e.g. Ohm's law, Faraday's law) require preservation of physical meaning of the input parameters, training data is not scaled [1], i.e., $x = \tilde{x}$. At the end of this step, training, validation and test data are ready to be used.

Figure 2.3. Input/output scaling and de-scaling during neural network model development and during the usage of the trained neural model [1]. Here, $\tilde{\tilde{y}}$ represents scaled outputs of neural network model and $\tilde{y} \approx \tilde{d}$ represents the de-scaled outputs of the neural network model.

Assuming (in order to simplify the notation) that scaling has already been done, notations x and d are used in place of $\tilde{x}$ and $\tilde{d}$ to denote scaled neural model inputs and scaled neural model outputs respectively in the following sections.

2.3.6 Weight Parameters Initialization

In this step, neural network weights (w) are initialized so as to provide a good starting point for the training process. Random-weight method is the widely used strategy for

MLP weight initialization [1], in which the weights are initialized with small random values (e.g. in the range $[-0.5, 0.5]$). To improve the convergence of training, one can use a variety of distributions (e.g. Gaussian distribution), or different ranges/variances for the random number generators used in initializing the ANN weights [122]. The strategy for weight initialization depends upon various factors including the order and topology of the network, and the neuron activation functions used. Once weight parameters are initialized, neural network is ready to be trained.

2.4 Neural Network Training

The most important step in neural model development is neural network training. The training data consists of sample pairs, $\{(\mathbf{x}_p, \mathbf{d}_p), p \in T\}$, where $\mathbf{x}_p$ and $\mathbf{d}_p$ are n - and m -vectors representing inputs and desired outputs of the neural network. Neural network training error [25] is defined as

$$E_T(\mathbf{w}) = \frac{1}{2} \sum_{p \in T} \sum_{j=1}^m \left| \tilde{y}_j(\mathbf{x}_p, \mathbf{w}) - d_{jp} \right|^2, \quad (2.15)$$

where d_{jp} is the j^{th} element of $\mathbf{d}_p$, and $\tilde{y}_j(\mathbf{x}_p, \mathbf{w})$ is the j^{th} neural network output for a given input $\mathbf{x}_p$. The purpose of neural network training is to adjust $\mathbf{w}$ such that the error function $E_T(\mathbf{w})$ is minimized [49]. Error $E_T(\mathbf{w})$ is a nonlinear function of $\mathbf{w}$ and iterative algorithms are often used to explore the $\mathbf{w}$ -space. Gradient-based iterative training techniques update $\mathbf{w}$ based on error $E_T(\mathbf{w})$ and error derivative information

$\frac{\partial E_T(\mathbf{w})}{\partial \mathbf{w}}$. Subsequent point in $\mathbf{w}$ -space denoted as $\mathbf{w}_{\text{next}}$ is determined by updating the current point $\mathbf{w}_{\text{now}}$ along a direction vector $\mathbf{u}$ as

$$\mathbf{w}_{\text{next}} = \mathbf{w}_{\text{now}} + \eta \mathbf{u} . \quad (2.16)$$

Here, $\Delta \mathbf{w} = \eta \mathbf{u}$ is called the weight update and η is a positive step size called learning rate [97]. As an example, Backpropagation training algorithm updates $\mathbf{w}$ along the negative direction of the derivative (or gradient) of training error as

$$\mathbf{w} = \mathbf{w} - \eta \frac{\partial E_T(\mathbf{w})}{\partial \mathbf{w}} . \quad (2.17)$$

2.4.1 Error Derivative Computation

As mentioned earlier, gradient-based training techniques need error derivative information $\frac{\partial E_T(\mathbf{w})}{\partial \mathbf{w}}$. For the MLP neural network, the derivatives are computed using a standard approach, often referred to as error backpropagation (EBP) [1], which is described in this section. A per-sample error function e_p is given by

$$e_p(\mathbf{w}) = \frac{1}{2} \sum_{j=1}^m (\tilde{y}_j(\mathbf{x}_p, \mathbf{w}) - d_{jp})^2, \quad (2.18)$$

for p^{th} data sample, $p \in T$. Let δ_i^L represent the error between i^{th} neural network output and i^{th} training data output, i.e.,

$$\delta_i^L = \tilde{y}_i(\mathbf{x}_p, \mathbf{w}) - d_{ip} . \quad (2.19)$$

This error at the output layer can be backpropagated to hidden layers as

$$\delta_i^l = \left(\sum_{j=1}^{N_{l+1}} \delta_j^{l+1} w_{ji}^{l+1} \right) z_i^l (1 - z_i^l) \quad l = L-1, L-2, \dots, 3, 2, \quad (2.20)$$

where δ_i^l represents local error at the i^{th} neuron of l^{th} layer. The derivative of the per-sample error in (2.18) with respect to a given weight parameter w_{ij}^l is given by

$$\frac{\partial e_p(\mathbf{w})}{\partial w_{ij}^l} = \delta_i^l z_j^{l-1} \quad l = L, L-1, \dots, 3, 2. \quad (2.21)$$

Finally, derivative of the training error in (2.15) with respect to w_{ij}^l can be computed as

$$\frac{\partial E_T(\mathbf{w})}{\partial w_{ij}^l} = \sum_{p \in T} \frac{\partial e_p(\mathbf{w})}{\partial w_{ij}^l}. \text{ Using the EBP approach, } \frac{\partial E_T(\mathbf{w})}{\partial \mathbf{w}} \text{ can be systematically}$$

evaluated for any MLP neural network structure and can be provided to gradient-based training algorithms for the determination of the weight update $\Delta \mathbf{w}$ during training [113].

2.4.2 Related Discussions

Validation error E_V and test error E_{T_e} can be defined in a manner similar to (2.15) using validation and test data sets V and T_e . During ANN training, validation error is periodically evaluated and training is terminated once a satisfactory E_V is reached [1]. After training, the quality of the neural model can be independently assessed by evaluating E_{T_e} [6]. Neural network training can be categorized into sample-by-sample training and batch-mode training. In sample-by-sample training (also called online training) $\mathbf{w}$ is updated each time a training sample $(\mathbf{x}_p, \mathbf{d}_p)$ is presented to the network. In batch-mode training (also known as offline training) $\mathbf{w}$ is updated after each epoch [49]. An epoch is defined as a stage of training, which involves presentation of all the training samples to the neural network once. For microwave modeling, batch-mode training is reported to be more effective [113].

2.4.3 Summary of Training Process

Let E_d and max_epoch represent desired neural model accuracy (i.e., validation error) and maximum allowable number of epochs respectively, both specified by the user. Batch-mode neural network training using gradient-based training algorithms is summarized here.

Step 1: Set $epoch_number = 0$ and initialize neural network weights $\mathbf{w} = \mathbf{w}_{initial}$.

Step 2: Perform feedforward computation of the neural network for all the samples in validation data set V and evaluate validation error E_v .

Step 3: If ($E_v \leq E_d$) or ($epoch_number \geq max_epoch$) stop training and go to step 6.

Step 4: Compute E_T and $\frac{\partial E_T}{\partial \mathbf{w}}$ using all samples in T simultaneously employing neural network feedforward computation and EBP.

Step 5: Find weight update $\Delta \mathbf{w}$ using a gradient-based training algorithm and update the weights as $\mathbf{w} = \mathbf{w} + \Delta \mathbf{w}$. Set $epoch_number = epoch_number + 1$ and go to step 2.

Step 6: Perform feedforward computation of the neural network for all samples in T_e . Evaluate E_{T_e} and independently assess the quality of the trained neural model.

A simplified flow-chart illustrating neural network training and neural model testing is shown in Figure 2.4.

Figure 2.4. Flow-chart demonstrating neural network training, neural model testing, and the use of training, validation, and test data sets in ANN modeling approach.

2.4.4 Learning Phenomena

Ability of a neural network to estimate output $\tilde{y}_p$ accurately when presented with input $\mathbf{x}_p$ never seen during training ($p \notin T$) is called generalization ability. Normalized training error [10] is defined as

$$\hat{E}_T(\mathbf{w}) = \left[\frac{1}{N_T m} \sum_{p \in T} \sum_{j=1}^m \left| \frac{\tilde{y}_j(\mathbf{x}_p, \mathbf{w}) - d_{jp}}{d_{\max,j} - d_{\min,j}} \right|^2 \right]^{1/2}, \quad (2.22)$$

where N_T represents number of samples in T , and $d_{\min,j}$ and $d_{\max,j}$ are the minimum and maximum values of j^{th} output data in $\mathbf{d}_p$, $p \in D$. A normalized validation

error $\hat{E}_V$ can be similarly defined. *Good learning* of a neural network is achieved when both $\hat{E}_T$ and $\hat{E}_V$ have small values (e.g. 0.50%). ANN exhibits *over-learning* [1], when it memorizes the training data but cannot generalize well (i.e., $\hat{E}_T$ is small but $\hat{E}_V \gg \hat{E}_T$). Remedies for over-learning are, deleting a certain number of hidden neurons or adding more samples to the training data. ANN exhibits *under-learning*, when it has difficulties learning the training data itself (i.e., $\hat{E}_T \gg 0$). Possible remedies are, adding more hidden neurons, or perturbing the current solution w to escape from a local minimum of $E_T(w)$ and then continuing the training [115].

2.4.5 Quality Measures

Quality of a trained neural network model is evaluated with an independent set of data called test data T_e [6]. A relative error δ_{jp} for j^{th} neural model output and p^{th} test sample is defined as

$$\delta_{jp} = \frac{\tilde{y}_j(x_p, w) - d_{jp}}{d_{\max, j} - d_{\min, j}}, \quad j = 1, \dots, m, \quad p \in T_e. \quad (2.23)$$

A quality measure based on q^{th} -norm is then defined as

$$M_q = \left[\sum_{p \in T_e} \sum_{j=1}^m |\delta_{jp}|^q \right]^{\frac{1}{q}}. \quad (2.24)$$

Average test error [1] can be calculated using M_1 as

$$\hat{E}_{avg} = \frac{M_1}{N_{T_e} m}, \quad (2.25)$$

where N_{T_e} represents number of samples in T_e . Worst-case error [3] among all test samples and all neural model outputs can be calculated as

$$M_{\infty} = \max_{k \in T_e} \left(\max_{j=1}^m |\delta_{jp}| \right). \quad (2.26)$$

Other statistical measures such as correlation coefficient [7] and standard deviation [37] can also be used.

2.4.6 Training Algorithms

Each training algorithm has a scheme for updating the weights of the neural network such that the neural network converges to an acceptable solution, i.e., neural model predictions match corresponding target values. Some of the neural network training algorithms commonly used for RF and microwave modeling are reviewed in this section.

2.4.6.1 Backpropagation

Backpropagation [98] is the most popular algorithm for neural network training. BP is a stochastic algorithm based on the steepest descent principle [123], in which, weights are updated along the negative gradient direction as

$$\Delta w_{now} = w_{next} - w_{now} = -\eta \left. \frac{\partial e_p(w)}{\partial w} \right|_{w=w_{now}} \quad (2.27)$$

or

$$\Delta w_{now} = w_{next} - w_{now} = -\eta \left. \frac{\partial E_T(w)}{\partial w} \right|_{w=w_{now}}. \quad (2.28)$$

Here, (2.27) is called sample-by-sample update and (2.28) is called batch-mode update. The basic BP suffers from slower convergence and possible weight oscillation. Addition of a momentum term to (2.27) and (2.28) as

$$\Delta w_{now} = -\eta \left. \frac{\partial e_p(w)}{\partial w} \right|_{w=w_{now}} + \alpha \Delta w_{old} = -\eta \left. \frac{\partial e_p(w)}{\partial w} \right|_{w=w_{now}} + \alpha(w_{now} - w_{old}) \quad (2.29)$$

and

$$\Delta w_{now} = -\eta \left. \frac{\partial E_T(w)}{\partial w} \right|_{w=w_{now}} + \alpha \Delta w_{old} = -\eta \left. \frac{\partial E_T(w)}{\partial w} \right|_{w=w_{now}} + \alpha(w_{now} - w_{old}) \quad (2.30)$$

reduces the weight oscillation [99], where α is called the momentum factor.

Several other techniques [124][125] have also been proposed to reduce weight oscillation. Engineering applications involving large-scale neural networks resulted in a need to further improve the efficiency of BP [126]-[128]. One way to enhance BP is to allow the learning rate and the momentum factor to be adaptive during learning [129][130]. Sparsity of hidden neuron activation pattern has also been used to reduce computations of BP [128][131]. It is suggested in [132]-[135] that first- and second-order derivatives of the objective function $E_T(w)$ with respect to the learning rate be calculated from the information gathered during forward and backward propagation. Basic BP uses the learning rate to adjust the step size. A different way to determine the step size is to use line search methods [123] such that the training error is optimized along the given update direction [136][137]. Yet another way to improve training efficiency of BP is the gradient reuse algorithm [138].

2.4.6.2 Gradient Based Optimization Techniques

Although BP algorithm is easy to implement, it has a slower rate of convergence. Gradient-based optimization techniques, which determine the update direction using derivative information of $E_T(\mathbf{w})$ can help to improve the rate of convergence [139]-[141]. Let $\mathbf{u}$ be the update direction, η be the learning rate, and $\mathbf{w}_{now}$ be the current value of $\mathbf{w}$. As mentioned in [97], gradient-based optimization algorithms tend to update $\mathbf{w}$ such that

$$E_T(\mathbf{w}_{next}) = E_T(\mathbf{w}_{now} + \eta \mathbf{u}) < E_T(\mathbf{w}_{now}). \quad (2.31)$$

The principal difference between various gradient-based algorithms lies in the procedure to determine successive update directions $\mathbf{u}$ [142]. Once $\mathbf{u}$ is determined, optimal step size can be found using line search, i.e.,

$$\eta^* = \min_{\eta > 0} \phi(\eta), \quad (2.32)$$

where

$$\phi(\eta) = E_T(\mathbf{w}_{next}) = E_T(\mathbf{w}_{now} + \eta \mathbf{u}). \quad (2.33)$$

Gradient of E_T is needed for determining update direction $\mathbf{u}$. For MLP, gradients are evaluated using derivative chain rule [97] starting from the output layer, i.e.,

$$\frac{\partial E_T}{\partial \mathbf{w}^l} = \frac{\partial E_T}{\partial \tilde{\mathbf{y}}} \cdot \frac{\partial \tilde{\mathbf{y}}}{\partial \mathbf{z}^l} \cdot \frac{\partial \mathbf{z}^l}{\partial \mathbf{w}^l} \quad l = L, \quad (2.34)$$

and then proceeding through various hidden layers down to the input layer as

$$\frac{\partial E_T}{\partial \mathbf{w}^l} = \frac{\partial E_T}{\partial \tilde{\mathbf{y}}} \cdot \frac{\partial \tilde{\mathbf{y}}}{\partial \mathbf{z}^L} \cdots \frac{\partial \mathbf{z}^{l+1}}{\partial \mathbf{z}^l} \cdot \frac{\partial \mathbf{z}^l}{\partial \mathbf{w}^l} \quad l = L-1, L-2, \dots, 1. \quad (2.35)$$

Here, $\tilde{y}$ represents neural network outputs, z^l represents l^{th} hidden layer outputs, and w^l represents the trainable weight parameters in the l^{th} layer of MLP.

2.4.6.3 Conjugate Gradient Algorithm

Conjugate gradient methods are originally derived from quadratic minimization where the minimum of the objective function is found within N_w iterations [98]. In these methods [100], neural network weights are updated in the current conjugate direction $\mathbf{u}_{now}$, with η of (2.16) determined through line minimization along $\mathbf{u}_{now}$. For the next iteration, conjugate direction $\mathbf{u}_{next}$ at $\mathbf{w}_{next}$ can be computed using

$$\mathbf{u}_{next} = -\mathbf{G}_{next} + \lambda_{now} \mathbf{u}_{now}, \quad (2.36)$$

where

$$\lambda_{now} = \frac{\mathbf{G}_{next}^T \mathbf{G}_{next}}{\mathbf{G}_{now}^T \mathbf{G}_{now}} \quad (2.37)$$

and $\mathbf{G} = \frac{\partial E_T(\mathbf{w})}{\partial \mathbf{w}}$ is the gradient of the objective function. Here, (2.37) is called the Fletcher-Reeves formula. The convergence of conjugate gradient algorithm is better than that of BP. Because of pure vector computations, conjugate gradient algorithms can be used to train both small and large neural networks. Other modified approaches (e.g. [132], [143]) have also been proposed to further reduce computational complexities.

2.4.6.4 Quasi-Newton Algorithm

Quasi-Newton methods [98] use second-order derivative information, i.e., inverse Hessian matrix $\mathbf{B}$ to update neural network weights as

$$\mathbf{w}_{next} = \mathbf{w}_{now} - \eta \mathbf{B}_{now} \mathbf{G}_{now}. \quad (2.38)$$

Here, matrix $\mathbf{B}$ is not computed. It is successively approximated using Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula [101] as

$$\mathbf{B}_{now} = \mathbf{B}_{old} + (1 + \frac{\Delta \mathbf{G}^T \mathbf{B}_{old} \Delta \mathbf{G}}{\mathbf{u}^T \Delta \mathbf{G}}) \frac{\mathbf{u} \mathbf{u}^T}{\mathbf{u}^T \Delta \mathbf{G}} - \frac{\mathbf{u} \Delta \mathbf{G}^T \mathbf{B}_{old} + \mathbf{B}_{old} \Delta \mathbf{G} \mathbf{u}^T}{\mathbf{u}^T \Delta \mathbf{G}}, \quad (2.39)$$

where

$$\mathbf{u} = \mathbf{w}_{now} - \mathbf{w}_{old} \quad (2.40)$$

and

$$\Delta \mathbf{G} = \mathbf{G}_{now} - \mathbf{G}_{old}. \quad (2.41)$$

Quasi-Newton algorithm exhibits faster convergence than conjugate gradient method. However, since the matrix $\mathbf{B}$ needs to be stored, this method is not effective for large neural networks. Limited-memory or one-step BFGS is a simplification of the full-memory algorithm, in which the inverse Hessian approximation is reset to the identity matrix after every iteration, thus avoiding the need to store matrices [144][145].

2.4.6.5 Gauss-Newton and Levenberg-Marquardt Algorithms

Let $\mathbf{r}$ be a vector containing the individual error terms in (2.15) for different samples and $\mathbf{J}$ be a Jacobian matrix containing the derivatives of $\mathbf{r}$ with respect to $\mathbf{w}$. Neural network weights are updated as

$$\mathbf{w}_{next} = \mathbf{w}_{now} - (\mathbf{J}_{now}^T \mathbf{J}_{now} + \mu \mathbf{I})^{-1} \mathbf{J}_{now}^T \mathbf{r}_{now}, \quad (2.42)$$

where μ is a positive number for Levenberg-Marquardt (LM) algorithm and zero for Gauss-Newton algorithm [102]. These methods are very effective for small residue problems, i.e., neural network training where $\mathbf{r}$ can be trained to small values. However,

due to the need for storing and solving linear equations, these methods are not effective for larger neural networks. A modified LM algorithm [146] using a diagonal matrix instead of the identity matrix I in (2.42) has been used for efficient training of MLP networks. To reduce the size of the Jacobian matrix, training samples are divided into several groups called local batches in [147]. Deficient Jacobian matrix [148] can also be utilized to reduce computational requirement and memory complexity of LM method.

2.4.6.6 Global Optimization

Global training algorithms use random optimization techniques, which allow these algorithms to escape from local minima and converge to the global minimum. Examples include simulated annealing [104] which allows the optimization process to jump out of a local minimum through an annealing process controlled by a temperature parameter; genetic algorithms [105] that evolve the structure and weights of the neural network through generations in a manner similar to biological evolution; Langevin updating rule for MLP networks [149] in which noise is added to the weights during training; and a stochastic minimization algorithm for training neural networks [150], which is basically a random optimization method with no gradient information needed. Since the convergence of pure random search techniques tends to be very slow, a more viable method is the hybrid method, which combines both conventional gradient-based training algorithms and random optimization concepts, e.g. [136].

2.4.7 ANN Model Development: Manual versus Automation

The process of developing neural models involves several key issues [115], e.g. sampling and generation of training data in the model input-space, selection of an appropriate neural network structure etc. Conventional ANN-modeling approach, also referred to as manual approach, requires human involvement in making these key decisions and then carrying out necessary steps manually. For example, given an RF/microwave modeling problem, the following questions arise.

- How many training and validation samples are needed?
- Where in the model input-space these samples are to be generated?
- What is the appropriate neural network size (i.e., number of hidden neurons)?
- Which of the available training algorithms is best suited?

The manual approach entails human understanding of neural-network-modeling issues and ANN-learning phenomena (e.g. over-learning) in order to be able to conveniently develop neural models [9]. Automation of neural model development process could be important both for minimizing the human-effort involved and for facilitating end-to-end ANN-based CAD. In [7], fast and accurate MLP neural network model of a spiral inductor was developed using a model development process termed as the MLPNN process. MLPNN process is an algorithm that requires human interaction in several steps. However, the flow diagram of the MLPNN process in Figure 2.5 is noteworthy. It provides an insight to the subject of automation and indicates scope for automation of actions such as selection of number of hidden layer neurons and addition of samples to the training set.

Figure 2.5. Flow diagram of the MLPNN model development process [7].

2.5 Review Examples

Component modeling and circuit design are important areas of neural-based RF/microwave CAD. In this section, ANN-modeling review examples for passive and active components are presented [115]. Examples illustrating use of component neural models in circuit design and optimization are also reviewed [113].

2.5.1 Neural Models for Components and Circuits

The first category of neural network applications is their use in modeling non-dynamic nonlinear input-output relationships of RF/microwave components, which can be accomplished using feedforward networks such as the MLP. Examples reviewed include neural modeling of interconnect networks [106], CPW symmetric T-junction [40] and MESFET [3].

2.5.1.1 High-Speed Interconnect Network

In [106], a neural network was trained to model signal propagation delays of an interconnect network used in printed circuit boards (PCB). The electrical equivalent circuit showing the interconnection of source IC pin to the receiver pins is shown in Figure 2.6. During PCB design, each individual interconnect network needs to be varied in terms of its interconnect lengths, receiver-pin load characteristics, source characteristics and network topology. Fast and accurate neural network model of the interconnect configuration is highly useful for this purpose. Variable inputs to the neural network model include $x = [L_i R_i C_i R_s V_p T_r e_j]^T$, $i = 1, 2, 3, 4$, and $j = 1, 2, 3$. Here, L_i is length of the i^{th} interconnect, R_i and C_i are terminations of the i^{th} interconnect, R_s is the

source impedance, and V_p and T_r are peak-value and rise-time of the source. The parameter e_j identifies the interconnect network topology [1][106]. Outputs of the neural model are the propagation delays at the four terminations, i.e., $y = [\tau_1 \ \tau_2 \ \tau_3 \ \tau_4]^T$. A 3-layer MLP shown in Figure 2.7, with 18 inputs and 4 outputs was used. Due to large dimensionality of the problem ($n = 18$), a relatively large number of training and test data (4500 training data and 4500 test data), and 40 hidden layer neurons were used. Data was generated using a circuit-based interconnect simulator named NILT [151]. Neural network training was carried out using a classical BP algorithm. Average test error $\hat{E}_{avg}$ of the resulting neural model was observed to be 0.04%. This fast and accurate neural model was used in a CAD environment, where 20,000 interconnect trees (with different interconnect lengths, terminations and topologies) had to be repetitively analyzed, and ANN-based design was found to be 300 times faster than the conventional techniques.

Figure 2.6. Electrical equivalent circuit of the interconnect configuration showing the connection of a source IC pin to four receiver pins [106].

Figure 2.7. Three-layered MLP structure with 18 inputs and 4 outputs used for modeling the interconnect network [106].

2.5.1.2 CPW Symmetric T-junction

Coplanar wave guide (CPW) circuits offer several advantages such as the ease of shunt and series connections, low radiation, low dispersion, and avoidance of the need for thin fragile substrates [40]. Currently, CAD tools available for CPW circuits are inadequate because of the non-availability of fast and accurate models for CPW discontinuities such as T-junctions, bends etc. Much effort has been expended in developing efficient methods for EM simulation of CPW discontinuities. However, the time-consuming nature of EM simulations limits the use of these tools for interactive CAD, where the geometry

of the component needs to be repetitively changed, thereby necessitating massive EM simulations.

Neural network based modeling and design addresses this challenge. In [40], neural network model of a symmetric T-junction was developed. Variable input parameters of the neural model are the physical dimensions W_{in} , G_{in} , W_{out} , G_{out} , and the signal frequency f , i.e., $\mathbf{x} = [W_{in} \ G_{in} \ W_{out} \ G_{out} \ f]^T$. Outputs of the ANN model are magnitudes and phases of S-parameters, i.e., $\mathbf{y} = [MS_{11} \ PS_{11} \ MS_{13} \ PS_{13} \ MS_{23} \ PS_{23} \ MS_{33} \ PS_{33}]^T$. EM simulations were performed for 25 physical configurations (sizes) of the component over a frequency range of 1 to 50 GHz generating 155 training and 131 test samples. A 3-layer MLP neural network with 15 hidden layer neurons was trained using the BP algorithm. Accurately trained neural models were used in place of CPU-intensive EM computations resulting in efficient CAD of CPW circuits.

2.5.1.3 MESFET

In [49], a neural model representing large-signal behaviors of a MESFET was developed. The neural model has 6 inputs, i.e., gate length (L), gate width (W), channel thickness (a), doping density (N_d), gate-source voltage (V_{gs}) and drain-source voltage (V_{ds}). The model has 4 outputs including the drain current and the electrode charges, i.e., $\mathbf{y} = [i_{dc} \ q_g \ q_d \ q_s]^T$. A 3-layer MLP neural network with 100 hidden layer neurons was used. A total of 1000 training and test data were generated from *OSA90* [12] simulations using physics-based *Khatibzadeh* and *Trew* MESFET model [152].

The neural network was trained using a modified BP algorithm including momentum adaptation to improve the speed of convergence. The trained neural model accurately predicted DC characteristics of the MESFET. The large-signal ANN model was then incorporated into the circuit simulation environment of *OSA90* and was used for both small- and large-signal simulations. A comparison of S-parameter predictions of the neural network model with those from the original *Khatibzadeh* and *Trew* MESFET model is shown in Figure 2.8.

Figure 2.8. Comparison of small-signal S-parameter predictions from the large-signal MESFET neural network model (o, +, x, *) with those from physics-based *Khatibzadeh* and *Trew* MESFET model (—) [49].

2.5.2 Nonlinear Modeling Using Neural Networks

Neural modeling of nonlinear device characteristics and dynamic responses of RF/microwave circuits is currently a key area of research. In this section, ANN approaches to modeling of transistor's DC, small-signal and large-signal behaviors are reviewed [115]. Neural modeling of dynamic circuit behaviors using recurrent neural networks is also illustrated.

2.5.2.1 DC Models

DC characteristics of transistors can be obtained either from simulation or measurement. Neural networks can be trained using such data to produce fast and accurate neural models [32]. In general, inputs to the DC neural network include bias parameters (e.g. gate and drain voltages), geometrical parameters (e.g. gate-length, channel thickness), and physical parameters (e.g. doping density, substrate permittivity). The output of the DC neural models is drain-current. In [3], empirical formulae [112] were utilized to develop DC models based on the KBNN concept. The empirical functions were used as hidden neuron activation functions in place of conventional sigmoid functions. The KBNN model development needed fewer samples and lesser training time, but yielded neural models with better accuracies as compared to MLP neural models.

2.5.2.2 Small Signal S-Parameter Models

Neural networks can learn measured/simulated small-signal data of transistors accurately. Inputs to the neural network include bias/geometrical/physical parameters and signal frequency. The outputs of the neural models are either real/imaginary parts or

magnitude/phase values of two-port S- or Y-parameters. MLP neural networks were used to develop S-parameter transistor models, e.g. MESFET [4], HBT [53]. In [4], a library of MESFET S-parameters was developed using conventional MLP approach as well as a hierarchical neural network approach. It was shown that the hierarchical approach [4][97] required fewer data and shorter training time as compared to the MLP approach. In [53], small-signal HBT neural models were developed. S-Parameter data for the HBT was measured for various input combinations of bias current, bias voltage and frequency, for five different transistor sizes. Three-layer MLP networks were trained for each transistor size. MLP3 networks take three inputs (collector bias current, bias voltage, frequency) and give eight outputs (magnitudes and phases of S_{11} , S_{12} , S_{21} and S_{22}). Average test errors of the resulting neural models were observed to be less than 2% as shown in Table 2.1. The MLP3 with 20 hidden neurons (MLP 3-20-8) was found to be the most accurate neural network for all five transistor sizes. A good agreement of the neural model estimation of S_{21} with the measured data is shown in Figure 2.9.

Table 2.1. Average test errors of S-parameter neural models for different transistor sizes.

HBT Size	MLP (3-12-8)	MLP (3-15-8)	MLP (3-20-8)	MLP (3-25-8)
Size 1	1.98%	1.95%	1.82%	1.86%
Size 2	1.40%	1.34%	1.16%	1.23%
Size 3	1.30%	1.18%	1.08%	1.15%
Size 4	1.29%	1.12%	1.06%	1.11%
Size 5	1.23%	1.07%	1.04%	1.07%

Figure 2.9. Comparison of neural model (MLP 3-20-8) prediction with measured data.

2.5.2.3 Large-Signal Models

Neural networks can be trained to learn the large-signal behavior of transistors. In [49], terminal currents and charges for different configurations of MESFET were simulated at a number of bias points using *OSA90* [12] with *Khatibzadeh* and *Trew* model [152]. The neural network model has six inputs namely gate-length, gate-width, channel thickness, doping density, gate voltage and drain voltage. The terminal currents and charges at drain, gate and source electrodes are the model outputs, leading to a total of six output parameters. Since the neural model directly describes terminal currents and charges as nonlinear functions of device parameters, it can be conveniently used in Harmonic Balance (HB) environment. The trained large-signal neural models were plugged into a circuit simulator as shown in Figure 2.10. The large-signal MESFET neural model was then used to satisfactorily perform DC, small-signal and HB simulations.

2.5.2.4 Neural Modeling Through a Known Equivalent Circuit

Lumped equivalent circuit approach is a traditional approach to transistor modeling [107]. Developing such models requires experience and involves a trial-and-error process to determine a matching topology [108]. Moreover, equivalent circuit parameters may not be related to the physical/geometrical parameters of the device under consideration. Empirical formulae for such relations exist and neural networks can easily learn these relationships. A hybrid approach that utilizes existing knowledge in the form of known equivalent circuit and empirical formulae, together with the powerful learning and generalization abilities of neural networks was demonstrated for MESFET [51] and HEMT [55]. In [51], MLP3 neural networks were trained to learn the bias-dependence of lumped elements of the MESFET. The neural models were then combined with the FDTD marching time algorithm leading to first-order approximation of the device-wave interaction and large-signal analysis. In [55], five-layer MLP neural networks were trained to learn the bias-dependence of small-signal equivalent circuit elements.

2.5.2.5 Circuit Models

Feedforward neural networks are well known for their ability to map static input-output relationships accurately. To model nonlinear circuit responses in time-domain, a neural network that can include temporal information is necessary. Recurrent neural networks (RNN) [69] were found to be suitable for this purpose. The structure of a typical RNN is shown in Figure 2.11. The input layer includes time-varying inputs u and time-invariant inputs p . The outputs of the neural network are denoted by vector y . The first hidden layer of RNN denoted by x contains buffered (time-delayed) history of y fed back from

the output layer, and buffered history of u and p . The second hidden layer z contains sigmoid neurons.

Figure 2.10. Incorporation of a large-signal MESFET neural network model into a harmonic balance circuit simulator [49]. The ANN model can be included as an additional nonlinear subnetwork.

Figure 2.11. Structure of a typical recurrent neural network [69].

The RNN can be trained to learn the dynamic characteristics of a nonlinear RF/microwave circuit. For such use, the training data can be a set of input and output waveforms of the nonlinear circuit under consideration. Since the present outputs of the neural model not only depend upon the present inputs, but also on the previous inputs and outputs, a novel BP training scheme called backpropagation-through-time needs to be used [1]. An RFIC power amplifier was modeled using the RNN approach [69]. The amplifier contains eight NPN BJT's modeled by two internal Agilent-ADS [14] nonlinear models Q34 and Q37. Inputs to RNN include input voltage waveform of the amplifier

and sampling cycle. Output of RNN is the amplifier's output voltage waveform. Neural based simulations of the power amplifier circuit were observed to be significantly faster than the original simulations.

2.5.3 ANN Based Circuit Design

ANN models for RF/microwave components can be used in the design and optimization of circuits. To achieve this, the neural models are first linked to the circuit simulator (e.g. *Agilent-ADS* [14]). Once integrated into the circuit simulator environment, the neural models can be used just as any other models available in the simulator's library. An ANN model can be connected with other ANN models or with any other non-neural models in the simulator for carrying out circuit-level design and optimization. In this section, circuit optimization examples utilizing fast and accurate neural models are reviewed [113].

2.5.3.1 CPW Folded Double-Stub Filter

In [40], a CPW folded double-stub filter was designed. For this design example, parameters $W = 70 \mu\text{m}$ and $G = 60 \mu\text{m}$ were fixed, yielding $Z_o \approx 50 \Omega$. Fast neural network models of CPW components, namely, CPW transmission line, 90° compensated bend, short-circuit stubs, and symmetric T-junctions (such as the one illustrated in section 2.5.1.2) were trained using accurate training data from full-wave EM simulations. Design and optimization of the filter were accomplished using *Agilent-MDS* [153] network simulator with the EM-ANN models of various components. The filter was designed for a center frequency of 26 GHz. Ideally, the length of each short-circuit stub and the section of line between the stubs should have a length of $\lambda/4$ at 26 GHz. However, due to

the presence of discontinuities, these lengths need to be adjusted. Parameters to be optimized are l_{stub} and l_{mid} . Initial values for these lengths were estimated, and the corresponding structure showed a less than ideal response when simulated. The circuit was then optimized using ANN-based CAD approach. The effect of optimization was a reduction in line lengths. A good agreement was obtained between EM-ANN and full-wave EM simulations of the optimized circuit over frequency range 1 - 50 GHz [113].

2.5.3.2 Three-Stage MMIC Amplifier

In [49], component neural models were used for efficient design and yield optimization of a 3-stage amplifier. A formulation for incorporating neural models into the circuit CAD framework was accomplished in the case of two circuit simulators *OSA90* [12] and *Agilent-ADS* [14]. Numerical results presented here were from the *OSA90* implementation. A 3-stage small-signal X-band cascable MMIC amplifier shown in Figure 2.12 with three MESFETs was designed.

Figure 2.12. A 3-stage MMIC amplifier. During yield optimization, various components in the amplifier including MESFETs were substituted by neural network models [49].

Specifications for the amplifier were:

- Passband (8 - 12 GHz): $12.4 \text{ dB} \leq \text{Gain} \leq 15.6 \text{ dB}$, Input VSWR ≤ 2.8
- Stopband (less than 6 GHz and greater than 15 GHz): Gain $\leq 2 \text{ dB}$

Physics-based ANN models of MESFETs (developed in section 2.5.1.3) and EM-based ANN models of passive components were used in this amplifier design. There are 14 design variables, i.e., metal-plate areas (S_{C_i}) of the MIM capacitors (C_i , $i = 1, 2, 3, 4$) and number of turns (n_{L_j}) of the spiral inductors (L_j , $j = 1$ to 10). A total of 37 statistical variables including gate length, gate width, channel thickness and doping density of MESFET models; metal-plate area and thickness of capacitor models; and conductor width and spacing of spiral inductor models were considered. The initial yield (before optimization) of the amplifier using the minimax nominal design was 26% with fast ANN based simulations and 32% with slow simulations using physics-based MESFET models. After neural-based yield optimization, the amplifier yield improved from 32% to 58%. The speed-up in the CAD of the amplifier achieved by the ANN-based approach is summarized in Table 2.2.

Table 2.2. Summary of CPU-time comparison between original physics-based and neural network based CAD operations in the 3-stage amplifier design [1][49].

CAD Operation	Design Using Khatibzadeh and Trew MESFET Model	Design Using Neural Network MESFET Model	Speed-up Factor
Nominal Optimization	7 min 8 sec	1 min 6 sec	6
Monte Carlo Analysis	40 hours 34 min	1 hour 30 min	30
Yield Optimization	4 hours 14 min	50 min	5

2.6 Summary

In this chapter, a comprehensive overview of RF/microwave computer-aided design approach based on artificial neural networks has been presented. Fundamental concepts such as problem statement of ANN-modeling, neural network structures, and training algorithms, have been systematically described. A detailed explanation of key neural model development issues including sample distributions, scaling methods, and quality measures, has been presented. Two categories of practical examples illustrating the use of neural networks in component modeling and circuit design have been reviewed. The chapter gives basic foundation of state-of-the-art in this emerging area and helps better understanding of the thesis contributions presented in the following chapters.

CHAPTER 3

Iterative Multi Stage Algorithm

In this chapter, one of the major contributions of this thesis work, namely, the Iterative Multi Stage (IMS) algorithm [5][8] is presented. In brief, the proposed IMS algorithm addresses several practical challenges in RF and microwave modeling, e.g. highly nonlinear behaviors, high dimensionality of the model, and presence of gross errors in training data. For the first time, the framework of the IMS algorithm introduces new concepts of iterative decomposition, multi-stage neural network training, sub neural networks and accumulated neural networks, which are described at length in the sections that follow.

3.1 Introduction

Neural network technology has recently transitioned from university research into high-frequency CAD arena. As neural-based techniques are being applied to practical and complicated RF/microwave problems, new challenges are brought to the forefront. RF/microwave data from measurements or simulations can have unavoidable errors [5]. Small errors in measurements are a result of calibration errors or equipment tolerances, and there could even be large errors due to equipment limitations, e.g. measurements at extreme frequencies. Data from simulations could have small errors resulting from accuracy limitations, e.g. round off or truncation, and large errors due to non-convergence, e.g. simulations with input parameter values in extreme locations. Presence of such errors in RF/microwave data can mislead ANN-training, resulting in non-reliable neural models.

Further, RF/microwave behaviors can be highly nonlinear with both smooth and non-smooth (large/sudden) variations, e.g. microstrip filters [64]. Simultaneously capturing these highly varying and contradicting portions of the problem behavior is difficult for conventional training (optimization) techniques because, over-emphasis on large variations may lead to poor fit in other regions of the model [15]. Yet another difficulty is posed by RF/microwave problems with a large set of design parameters, which lead to high-dimensional modeling. Neural modeling of such problems requires extremely large amounts of data and neural-network-training becomes cumbersome. In a nutshell, ANN-modeling of complicated RF/microwave input-output characteristics poses unique challenges, and these issues need to be addressed.

To simplify ANN-model development process, some techniques were proposed in the EM-ANN community. KBNN [3], difference method [6] and PKI networks [37] exploit the existing RF/microwave knowledge (e.g. empirical models) by incorporating it into the internal structure of ANN. These approaches are originally aimed at improving model accuracy and generalization capability of ANN models with fewer training data. However, they could indirectly help simplify the task of modeling highly nonlinear behaviors, if the embedded knowledge is correspondingly nonlinear. Research in the neural network area also led to some techniques including those based on robust statistical method [154] and higher-order cumulants [155] to deal with large errors in data; constructive algorithms, e.g. cascade correlation training architecture [156] that captures higher order features in a problem; clustering algorithms such as self-organizing map [157] and random mapping technique [158] that reduce neural network input dimension. However, each of these techniques has

addressed a specific challenge in neural network modeling at a time, and from an RF/microwave perspective a comprehensive approach is now needed to handle the above-mentioned challenges simultaneously. As a part of this thesis work, investigations have been carried out toward developing an RF/microwave-oriented framework that can handle highly nonlinear and high-dimensional ANN-modeling even in the presence of gross errors in training data. An iterative multi-stage algorithm that contains a macro-level process and a stage-level process is proposed. At the macro-level, the IMS decomposes the complicated original task into several simpler sub-tasks or stages. At the stage-level, suitable neural network structures and training algorithms are used to model a particular portion of the overall problem behavior. The IMS iteratively captures the finer and finer behaviors in the original problem by transforming the training process into a sequence of new learning problems, wherein a secondary behavior missed in a previous stage can be learned as the primary behavior in the current stage. To help the IMS algorithm in one of the training stages where only the smooth portion of the problem behavior needs to be modeled ignoring large variations, a novel Huber quasi-Newton [15] training method is integrated into the IMS framework. HQN also has inherent capability of handling RF/microwave ANN-modeling problems with gross errors in training data [159].

3.2 IMS Algorithm: Macro Level

The IMS algorithm contains a macro-level process and a stage-level process. At the macro-level, the IMS decomposes a complicated problem into relatively simpler sub-tasks or stages and transforms the problem into higher-order learning spaces. In the transformed learning space, an otherwise over-shadowed portion of the problem behavior becomes more

prominent. At the stage-level, i.e., in each training stage, depending upon the objective of the sub-task, an appropriate neural network structure is trained by suitable training algorithms. The sub neural models obtained after each stage are iteratively added to produce the overall neural model.

3.2.1 Mathematical Description

In general, RF/microwave component and circuit behaviors are nonlinear and multi-dimensional, and may be composed of a variety of patterns such as smooth, non-smooth, low-frequency, high-frequency etc. Assume these patterns are ranked on the basis of their contribution to the overall behavior. Qualitatively, the most dominant pattern is defined as the first-order behavior, the next dominant pattern as the second-order behavior and so on. The objective is to conveniently develop a neural network model $\tilde{y} = \mathbf{h}(\mathbf{x}, \mathbf{w})$ that accurately represents the complicated RF/microwave input-output relationship $y = f(\mathbf{x})$ of interest.

The IMS algorithm begins with a sub neural model

$$\tilde{Y}^1 = \mathbf{H}^1(\mathbf{x}, \mathbf{v}^1), \quad (3.1)$$

which approximates the first order behavior of $y = f(\mathbf{x})$. This is achieved by training a sub neural network $\mathbf{H}^1(\mathbf{x}, \mathbf{v}^1)$ to learn available measured/simulated $f(\mathbf{x})$ training data through optimization in $\mathbf{v}^1$ space, where $\mathbf{v}^1$ denotes the weights of the sub neural network $\mathbf{H}^1$. Training of $\mathbf{H}^1(\mathbf{x}, \mathbf{v}^1)$ should be easier than the original RF/microwave modeling problem since it does not need to bear the burden of completely learning the original problem, but needs to learn only the most dominant pattern of the original problem behavior. This can be

achieved by using a conservative estimate of number of hidden neurons, i.e., relatively fewer hidden neurons. The original RF/microwave problem $y = f(x)$ can then be represented as

$$y = \tilde{Y}^1 + R^1(x) = \tilde{y}^1 + R^1(x), \quad (3.2)$$

where

$$\tilde{y}^1 = h^1(x, w^1) \quad (3.3)$$

and

$$h^1(x, w^1) = H^1(x, v^1). \quad (3.4)$$

Here $\tilde{y}^1$ is the accumulated neural model approximating the original problem upto first order behavior, $h^1(x, w^1)$ and $w^1 = v^1$ are the corresponding accumulated neural network and accumulated weight vector respectively, and $R^1(x)$ is the remainder function.

If $R^1(x)$ is larger than desired, another sub neural model

$$\tilde{Y}^2 = H^2(x, v^2) \quad (3.5)$$

is used to approximate $R^1(x)$. This is achieved by training another sub neural network $H^2(x, v^2)$ to learn $R^1(x)$ through optimization in v^2 space, where v^2 denotes the weights of the sub neural network H^2 . Again this training should be easier since $H^2(x, v^2)$ is not required to learn $R^1(x)$ completely, but is only required to capture its primary pattern. Since the primary or the most dominant pattern of the original behavior is already captured by $H^1(x, v^1)$, the primary pattern of $R^1(x)$ to be captured by $H^2(x, v^2)$ is actually the secondary pattern of $y = f(x)$. Remainder $R^1(x)$ can then be represented as

$$R^1(x) = \tilde{Y}^2 + R^2(x), \quad (3.6)$$

where $R^2(x)$ is the new remainder function. Substituting (3.6) in (3.2) we obtain

$$\begin{aligned} y &= \tilde{Y}^1 + \tilde{Y}^2 + R^2(x) \\ &= \tilde{y}^2 + R^2(x) \end{aligned} \quad (3.7)$$

where

$$\tilde{y}^2 = h^2(x, w^2) \quad (3.8)$$

and

$$h^2(x, w^2) = H^1(x, v^1) + H^2(x, v^2). \quad (3.9)$$

Here $\tilde{y}^2$ is the accumulated neural model approximating the original problem upto second order behavior, and $h^2(x, w^2)$ and w^2 are the corresponding accumulated neural network and accumulated weight vector respectively. In the IMS algorithm, w^2 is obtained by suitably concatenating v^1 and v^2 .

Proceeding this way, the accumulated neural model approximating the original problem $y = f(x)$ upto k^{th} order is given by

$$\tilde{y}^k = h^k(x, w^k), \quad (3.10)$$

where corresponding accumulated neural network can be obtained by summing up the sub neural networks as

$$h^k(x, w^k) = \sum_{i=1}^k H^i(x, v^i). \quad (3.11)$$

The accumulated weight vector w^k is obtained by concatenation of all v^i , $i=1,2,...,k$. At this stage, the remainder function $R^k(x)$ can be expressed as

$$R^k(x) = y - \tilde{y}^k. \quad (3.12)$$

If $R^k(x)$ is small enough, the objective is met and the final neural model would be

$$\tilde{y} = h^k(x, w^k). \quad (3.13)$$

Otherwise, the algorithm proceeds to develop a new sub neural network model

$$\tilde{Y}^{k+1} = H^{k+1}(x, v^{k+1}), \quad (3.14)$$

to approximate the most dominant pattern of the remainder function $R^k(x)$, by training a new neural network $H^{k+1}(x, v^{k+1})$. Consequently, an even more accurate neural model of the problem $y = f(x)$ can be obtained by including higher-order behaviors.

3.2.2 Steps in Macro Level Process

The steps involved in the macro-level process are summarized here. It may be noted that the inputs to all sub neural networks remain the same, but the desired outputs keep changing.

Step 1: Start with the original microwave problem $y = f(x)$. Training data is generated either from measurements or simulations. Begin the first stage and set stage counter $k=1$. The training data for the first stage of training is given by,

$$d_p^1 = d_p = f(x_p), \quad p \in T. \quad (3.15)$$

Step 2: Train a sub neural network $H^1(x, v^1)$ to capture the most dominant pattern in training data. Tasks involved in sub neural model ($\tilde{Y}^1$) development such as selection of neural network structure and training algorithms and neural network training, are all achieved through stage-level process of IMS described in section 3.3.

Step 3: Obtain accumulated neural model $\tilde{y}^1 = \tilde{Y}^1$ that represents the original problem upto first order behavior, by enforcing (3.4).

Step 4: If the sum sampled-norm of $R^k(x)$, i.e., $\sum_{p \in V} \|R^k(x_p)\| = \sum_{p \in V} \|d_p - h^k(x_p, w^k)\|$, is small enough then stop. Otherwise, set $k = k+1$. Here, V is the index set of validation data.

Step 5: The training data for k^{th} training stage are samples of the remainder function $R^{k-1}(x)$ from $k-1^{th}$ stage and is obtained as

$$\begin{aligned} d_p^k &= d_p - h^{k-1}(x_p, w^{k-1}), \quad k = 2, 3, \dots, \quad p \in T \\ &= d_p^{k-1} - H^{k-1}(x_p, v^{k-1}) \end{aligned} \quad (3.16)$$

Step 6: Train a sub neural network $H^k(x, v^k)$ with sample pairs (x_p, d_p^k) , $p \in T$, so as to capture the k^{th} order behavior using the stage-level process of the IMS algorithm.

Step 7: Obtain the accumulated neural model $\tilde{y}^k = h^k(x, w^k)$ using

$$h^k(x, w^k) = h^{k-1}(x, w^{k-1}) + H^k(x, v^k) \quad (3.17)$$

as shown in Figure 3.1. Go to step 4.

3.2.3 Discussions

Conventional approach to RF/microwave neural network modeling is a one-shot training in which the entire problem behavior including dominant and finer patterns is treated as a single behavior and a single neural network is forced to capture this behavior. This approach is a special case of the IMS algorithm where the number of stages is equal to 1 and the entire problem behavior is considered as the first order behavior of the problem. By virtues of

iterative decomposition and multi-stage training, the IMS allows extra freedom of utilizing a variety of neural network structures and training algorithms suitably in different stages.

Figure 3.1. Development of accumulated neural model and computation of training data for each training stage in the macro-level process of IMS algorithm.

The IMS algorithm has an inherent capability to grow model structure beyond existing methods in neural networks, e.g. [156] in which neurons are added to the network to further learn the same set of data. In the IMS algorithm, each newly added sub neural model is trained by different data in a higher-order learning space, where the finer and less prominent patterns of the original problem become dominant. Further, neural network growth in the IMS algorithm exploits the merits of a variety of ANN structures.

In IMS algorithm, the remainder function $\mathbf{R}^{k-1}(\mathbf{x})$ represents the difference between the accumulated neural model at the end of $k-1^{th}$ stage and the original problem. To perform a heuristic error analysis, an error measure C^k is defined as

$$C^k = \max_{j=1,2,\dots,m} C_j^k, \quad k=1,2,3,\dots, \quad (3.18)$$

where

$$C_j^k = d_{\max,j}^k - d_{\min,j}^k. \quad (3.19)$$

In (3.19), $d_{\max,j}^k = \max_{p \in T} d_{jp}^k$ where d_{jp}^k is the j^{th} element/output of training data $\mathbf{d}_p^k$, and $d_{\min,j}^k$

is similarly defined. The error measure C^k thus represents the maximum swing of $\mathbf{R}^{k-1}(\mathbf{x})$ for all training samples among all outputs and this swing is more likely to be a part of its dominant pattern. Once the dominant pattern of $\mathbf{R}^{k-1}(\mathbf{x})$ is learned in the k^{th} stage, C^{k+1} can be smaller than C^k . In other words, the overall neural network model improves over the previous stage as can be seen from the demonstration examples in section 3.4.

It has been mentioned earlier that, qualitatively, the most dominant pattern is defined as the first-order behavior, the next dominant pattern as the second-order behavior and so on. In practice, this ranking is not performed by a pre-analysis. It is automatically carried out by the macro-level process. In each stage, a conservative number of hidden neurons are provided, leading to limited nonlinearity or freedom of the neural network. As such, the neural network captures only a portion of the overall behavior (say k^{th} order behavior).

3.3 IMS Algorithm: Stage Level

3.3.1 General Description

Development of sub neural models defined in the macro-level process of IMS is handled by the stage-level process. In a particular stage, depending upon the dominant pattern in the training data of that stage, a new neural network structure is trained by suitable training algorithms. Since the dominant patterns in different stages are likely to be different, each stage is allowed to use different structures and different algorithms. As such, IMS algorithm gives an opportunity to maximally combine the merits of a variety of neural network structures and the strengths of different training algorithms to model a single microwave problem. The stage level process mainly includes selection of a neural network structure, selection of training algorithms and an adaptive-controller mechanism that provides tips for decision-making during training.

3.3.2 Sub Neural Network Structures

In k^{th} training stage, the k^{th} order behavior of the original problem is modeled by a sub neural network. The default choice for the sub neural network structure is 3-layer perceptrons or

MLP3. However, other structures such as 4-layer perceptrons or MLP4, (RBF) networks, and knowledge based neural networks can also be used. For example, if the problem behavior exhibits local and large variations an RBF network is used; and if approximate problem knowledge is available knowledge-based networks can be used. Table 3.1 summarizes the selection criteria for neural network structures based on different training objectives in individual stages.

3.3.3 Sub Neural Network Training

The training objective for k^{th} training stage is to minimize the error between training data and predictions from the sub neural model of that stage. In other words, the objective of training is to adjust $\mathbf{v}^k$ such that training error function $E_T^k(\mathbf{v}^k)$ is minimized. As an example, for the standard l_2 -based methods, error function $E_T^k(\mathbf{v}^k)$ is defined following (2.15) as

$$E_T^k(\mathbf{v}^k) = \frac{1}{2} \sum_{p \in T} \sum_{j=1}^m (\tilde{y}_{jp}^k - d_{jp}^k)^2. \quad (3.20)$$

Here $\tilde{y}_{jp}^k$ is the j^{th} output of neural network $\mathbf{H}^k(\mathbf{x}, \mathbf{v}^k)$, when input $\mathbf{x}_p$ is presented. To simplify the notation, $\mathbf{v}$ and $E_T(\mathbf{v})$ can be used to denote the ANN weights $\mathbf{v}^k$ and the error function $E_T^k(\mathbf{v}^k)$ in k^{th} stage. Starting with an initial guess $\mathbf{v}_0$, optimization-based training techniques iteratively march towards $\mathbf{v}^*$ that minimizes $E_T(\mathbf{v})$. Different training algorithms use different schemes for determining the update direction [97] as described in section 2.4.

3.3.4 Algorithms for Stage Level Training

For different training objectives in individual stages, the stage-level process utilizes different training techniques. These techniques include l_2 -based training methods such as back-propagation [99], conjugate-gradient [100], quasi-Newton [101], Levenberg-Marquardt [102], and a novel Huber quasi-Newton technique integrated into the IMS framework. Guidelines for selection of appropriate training algorithms to meet different objectives in individual stages are presented in Table 3.1.

Table 3.1. Selection of neural network structure and training algorithm for the training stages with different objectives in stage-level process of IMS algorithm.

Training Objective in k^{th} Stage	Neural Network Structure	Training Algorithm
To ignore large/gross errors in training data	MLP	HQN technique
To model smooth dominant pattern in training data ignoring higher-order effects	MLP with fewer hidden neurons than required (i.e., limited or moderate freedom)	Standard l_2 -methods
To model training data with local and large variations	RBF	Standard l_2 -methods
To ignore large/sudden variations in training data	MLP	HQN technique
To model sudden variations in training data which are not errors	Knowledge based neural network structures	Standard l_2 -methods

3.3.5 Huber Quasi-Newton Algorithm

One of the frequently encountered challenges in developing RF/microwave neural models is the presence of large/sudden variations in training data. These variations need to be ignored,

if they are gross errors from measurement/simulation. Even if the large/sudden variations are due to the nature of the problem itself, the IMS algorithm may choose to ignore them in one of the stages where only the smooth patterns are modeled. A training technique that can model smooth portion of the data, neglecting large/sudden variations is therefore needed. The objective functions of conventional ANN training algorithms are formulated in l_2 -sense, e.g. (3.20). Although l_2 -norm based training can handle small/smooth variations in training data, it can be misled by large/sudden variations resulting in non-reliable neural models. On the other hand, l_1 -norm based training is robust against large/sudden variations, but is not very effective in dealing with small/smooth variations [160]. In summary, neither l_1 nor l_2 -norm based trainings can discriminate and treat small/smooth and large/sudden variations differently. The HQN training technique addresses this aspect of ANN-training.

Huber function [161][162] is a smooth combination of l_1 and l_2 -norms and was applied to device modeling and circuit optimization [163][164]. In the proposed IMS, it is utilized for neural network training. Consider the per-sample error function $e_p(\mathbf{v})$ as defined in (2.18).

The overall objective function $E_T(\mathbf{v})$ is then defined based on the Huber function as

$$E_T(\mathbf{v}) = \sum_{p \in T} \rho[e_p(\mathbf{v})], \quad (3.21)$$

where $\rho(\cdot)$ is the Huber function. The Huber-norm of $e_p(\mathbf{v})$ is given by

$$\rho[e_p(\mathbf{v})] = \begin{cases} [e_p(\mathbf{v})]^2 / 2 & , \text{ if } |e_p(\mathbf{v})| \leq K \\ K|e_p(\mathbf{v})| - K^2 / 2 & , \text{ if } |e_p(\mathbf{v})| > K \end{cases}, \quad (3.22)$$

where K is the Huber constant. By varying K , the proportion of neural network error functions to be treated in l_1 - or l_2 -sense can be controlled. From the implementation

point of view, the magnitude of error (i.e., small or large) in any data sample is with respect to the tolerance/accuracy of measurement-setup/simulator being used. As an example, if the measurement equipment has a specified tolerance of say 5%, any data error in excess of 5% is considered as a large error, and K is adjusted accordingly.

Large values of the per-sample error ($> K$) indicate data with large/sudden variations. Such samples or error functions are treated in l_1 -sense, and hence the training ignores these variations. On the other hand, small values of the per-sample error ($\leq K$) indicate data with small/smooth variations. Such samples or error functions are treated in l_2 -sense, and therefore the training efficiently captures these variations. Consequently, Huber-norm based training objective can be robust against large/sudden variations in training data. This feature is utilized by the IMS algorithm to extract only the smooth behaviors eliminating the large/sudden variations in one of the training stages.

The gradient of the Huber-norm based training objective function $E_T(\mathbf{v})$ in (3.21) with respect to the neural network weights $\mathbf{v}$ is obtained as

$$\begin{aligned} G &= \sum_{p \in T} \frac{\partial \rho[e_p(\mathbf{v})]}{\partial \mathbf{v}} \\ &= \sum_{p \in T_1} e_p(\mathbf{v}) \frac{\partial e_p(\mathbf{v})}{\partial \mathbf{v}} + \sum_{p \in T_2} (\pm K) \frac{\partial e_p(\mathbf{v})}{\partial \mathbf{v}} \end{aligned} \quad (3.23)$$

where $T_1 = \{p \mid e_p(\mathbf{v}) \leq K, p \in T\}$ and $T_2 = \{p \mid e_p(\mathbf{v}) > K, p \in T\}$. In (3.23), $\frac{\partial e_p(\mathbf{v})}{\partial \mathbf{v}}$ is computed in a manner similar to BP. The computed gradient is supplied to equation (2.38) of quasi-Newton equation resulting in HQN training technique. The quasi-Newton algorithm

uses this gradient information together with second-order derivatives (i.e., approximation of the inverse Hessian matrix $\mathbf{B}$) to update the sub neural network weights as

$$\mathbf{v}_{next} = \mathbf{v}_{now} - \eta \mathbf{B}_{now} \mathbf{G}_{now}, \quad (3.24)$$

where $\mathbf{G}_{now}$ is the gradient evaluated at $\mathbf{v}_{now}$. Matrix $\mathbf{B}$ is successively approximated using BFGS formulas (2.39) to (2.41) of chapter 2.

3.3.6 Adaptive Controller

As part of the overall IMS framework, an adaptive-controller has been implemented for assisting the stage-level training. Guidelines for decision-making during ANN training in various stages of the IMS presented in Table 3.1 and Table 3.2 are the basis for this adaptive controller. Directives from the adaptive-controller include switching between different training algorithms and adding more neurons or training data to improve neural model accuracies. The controller randomly perturbs neural network weights (e.g. up to 20%) to escape from traps of local minima, and helps in faster convergence when closer to solution.

Table 3.2. Different situations encountered in a training stage and appropriate actions taken by the adaptive-controller of stage-level process in the proposed IMS algorithm.

Training Status	Actions
Over-learning is detected	Delete neurons, or add training data and continue training
Under-learning is detected	Add neurons and continue training
Close to solution	Switch to quasi-Newton and continue training, unless HQN is already being used
Stuck in a local minimum	Perturb current weights of the neural network and continue training
Desired accuracy or maximum number of epochs is reached	Stop training

3.4 Demonstration Examples

3.4.1 Heterojunction Bipolar Transistor

This is a one-stage example to verify the HQN technique. The heterojunction bipolar transistor (HBT) model has three inputs namely, frequency (f), bias voltage (V_B) and bias current (I_B). There are eight outputs, i.e., magnitudes and phases of two-port S-parameters S_{11} , S_{12} , S_{21} and S_{22} . Data is generated from measurements using a network analyzer. The measured data has small errors in most of the samples and occasionally there are some large errors mostly due to extreme frequency samples. Two sets of data are used to train and test the neural model. Set A is the original data with small and large errors. Set B is obtained by preprocessing set A, i.e., samples with large errors are removed. A 3-layer MLP neural network with 12 hidden neurons is used. For the purpose of comparison, neural models are developed using both conventional l_2 -based training method and the HQN technique. It can be seen from Figure 3.2 and Table 3.3 that the HQN technique yields accurate neural models as compared to standard l_2 -method, when the training data has large/gross errors. The term, sample number, in Figure 3.2 represents frequency sweeps for 4 different bias conditions. Neural models in this example and all other subsequent examples are developed using *NeuroModeler* [18].

3.4.2 MESFET

This is another one-stage example to verify the HQN technique for neural model development based on simulation data. The MESFET model has four inputs namely, frequency (f), drain voltage (V_D), gate voltage (V_G) and channel thickness (a). There are eight outputs, i.e., real and imaginary parts of two-port S-parameters S_{11} , S_{12} , S_{21} and S_{22} .

Figure 3.2. Comparison between neural model prediction and training data for a HBT. The training data has occasional large errors, mostly due to extreme frequency samples. (a) Neural model trained by standard l_2 -method is affected by large errors. (b) Neural model trained by the HQN technique is not affected by large errors.

Table 3.3. Average and worst-case errors between neural model prediction and test data for HBT example. Data set A has small and large measurement errors. Data set B is same as set A, except that the large errors are identified and removed.

Training Method	Training Data Set	Test using Data Set A		Test using Data set B	
		Average Error	Worst-Case Error	Average Error	Worst-Case Error
Standard l_2 method	A	1.21%	15%	1.17%	15%
Proposed HQN	A	0.97%	100%	0.80%	9%
Standard l_2 method	B	-	-	0.82%	17%
Proposed HQN	B	-	-	0.73%	11%

Data is generated from simulations using *OS490* [12]. There are a few obvious large errors in data that can be attributed to non-convergence of simulations due to input samples in extreme locations. A 3-layer MLP neural network with 15 hidden neurons is used. Neural models developed using l_2 -based training method and the HQN technique are compared in Figure 3.3. HQN yields accurate neural models as compared to standard l_2 method in the presence of large gross errors.

3.4.3 Circuit Waveform Model

This example is for illustrating the macro-level iterative decomposition concept of IMS training algorithm. The output waveform of a simple VLSI interconnect circuit with three transmission lines shown in Figure 3.4 is to be modeled. The neural model has one input, namely, time (t). The output of the model is the terminal voltage at node 4 (V_4). Waveform data is obtained through HSPICE [165] simulation. The voltage waveform is made up of different patterns namely, a step, a dominant sinusoid and higher-order harmonics with small amplitude. The macro-level process of IMS ranks the step as the first order behavior, the sinusoid as the second order behavior and the higher-order harmonics as higher-order behaviors. The stage-level process of IMS then captures each pattern in a different stage.

The training data for the first stage is the original data itself. A 3-layer MLP neural network with 1 hidden neuron is deliberately chosen so as to learn only the step component through limited freedom of the neural network. As shown in Figure 3.5(a), the step component of the waveform is captured by the sub neural model trained in the first stage.

Figure 3.3. Comparison between neural model prediction and training data for a MESFET. The training data has a few large/gross errors. (a) Neural model trained by standard l_2 method is affected by large errors. (b) Neural model trained by HQN technique is not affected by large errors.

Figure 3.4. The circuit for the waveform modeling example with 3 transmission lines representing high-speed VLSI interconnects.

For the second stage, the training data is obtained by subtracting the prediction of first stage sub neural model from the original data. Clearly, the training data is left with sinusoid and higher-order harmonics as shown in Figure 3.5(b). A 3-layer MLP with 15 hidden neurons is chosen for the second stage in order to learn the sinusoid through moderate freedom of the neural network. Figure 3.5(b) shows that the sinusoid component is captured by the trained sub neural model of second stage. At this point, the sub neural models of first and second stages are added to produce the accumulated model.

For the third stage, the training data is obtained by subtracting the prediction of accumulated model from original data. As shown in Figure 3.5(c), the over-shadowed higher-order

harmonics in the original waveform become dominant pattern in the third stage. These higher-order harmonics exhibit local and large variations. A RBF network with 80 hidden neurons is used and performance of the trained sub neural model is shown in Figure 3.5(c). The sub neural model of third stage is added to the previous accumulated model to yield the overall model. It can be seen from Figure 3.5(d) that the overall model predicts the original voltage waveform accurately. Stage-wise training results are summarized in Table 3.4 and the tendency of the error measure C^k versus number of stages is shown in Figure 3.6.

Table 3.4. Stage-wise training results of the circuit-waveform modeling example using the proposed IMS algorithm.

Stage Number	Average Test Error of Accumulated Model	Average Test Error of Sub Model	Neural Network Structure	Error Measure
k	$(\tilde{y}^k)$	$(\tilde{Y}^k)$	(H^k)	(C^k)
1	8.78%	8.78%	MLP3	16.87
2	0.78%	1.14%	MLP3	11.57
3	0.08%	1.29%	RBF	1.00
4	-	-	-	0.12

(a)

(b)

(c)

(d)

Figure 3.5. Training data and neural model prediction for the circuit waveform modeling example. (a) First stage of training, (b) Second stage of training, (c) Third stage of training and (d) Overall neural model.

Figure 3.6. Error measure C^k versus training stages in the circuit waveform modeling example.

3.4.4 Microstrip Filter

In this example, neural network model of a microstrip filter is developed. The model has four inputs namely, frequency (f), width of sections (w_d) and gaps between sections (g_{12} and g_{23}). There are four outputs namely, real and imaginary parts of the two-port s-parameters S_{11} and S_{21} . Data is generated from simulation using *Serenade* [166]. Thirty-six filters with different widths and gaps are considered. For each filter, real and imaginary parts of the s-parameters are simulated at 201 frequency points, resulting in a total of 7236 data samples. The outputs are smooth over most of the frequency range except at a couple of frequencies where each filter exhibits sharp/large variations.

Conventional one-shot training by standard l_2 methods, to capture both smooth and sudden variations, in just one stage has been tried but it did not yield satisfactory models, because over-emphasis on large variations at resonant frequencies led to a poor fit at other frequencies. The macro-level process of the IMS algorithm ranks the smooth portion as the first order behavior, the large/sudden variations as the second order behavior and the remaining tiny patterns as higher-order behaviors. The stage-level process of IMS then captures each pattern in a different stage.

In the first stage, a 3-layer MLP structure is trained by Huber quasi-Newton technique to capture the smooth portion of the filter response, ignoring large/sudden variations in training data. In the second stage, the large/sudden variations are modeled using a suitable knowledge based neural network. The knowledge neuron activation function has a behavior similar to the large/sudden variations of the filter response and is given by,

$$\psi(f, f_0, h, c) = \frac{c h}{(f - f_0)^2 + c}, \quad (3.25)$$

where f represents frequency, f_0 represents resonant frequency, h represents the height of sudden variation or dip, and c regulates dip-width. The training data for the third stage is the remaining tiny patterns that are highly nonlinear, and so a 4-layer MLP neural network is chosen. The sub neural models developed in the above three stages are combined to produce the overall model. The overall neural model is observed to predict the s-parameters with respectable accuracies, e.g. reflection coefficient shown in Figure 3.7. The stage-wise training results are presented in Table 3.5.

Figure 3.7. A comparison between the neural network model and EM training data for the microstrip filter modeled using the proposed IMS algorithm. The filter neural model accurately represents the high-frequency behaviors with respect to changes in the filter's geometrical parameters. The response shown here corresponds to a filter with normalized values of w_d , g_{12} and g_{23} as 1.0, 1.5 and 2.0 respectively.

Table 3.5. Stage-wise training results for the microstrip filter example using the proposed IMS algorithm.

Stage Number	Average Test Error of Accumulated Model	Average Test Error of Sub Model	Neural Network Structure	Error Measure
k	$(\tilde{y}^k)$	$(\tilde{Y}^k)$	(H^k)	(C^k)
1	2.16%	2.16%	MLP3	2.0000
2	1.54%	2.29%	Knowledge based network	1.4089
3	0.34%	0.52%	MLP4	1.0323
4	-	-	-	0.3637

3.4.5 Eight Conductor Stripline

Transmission-line models are very important for delay and cross-talk analysis in high-speed VLSI interconnect design. Eight-conductor transmission-line in Figure 3.8 typically represents a data/address bus on a printed-circuit board. The neural model has 16 inputs namely, width of each conductor ($w_i, i=1, 2, \dots, 8$), separation between conductors ($s_i, i=1, 2, \dots, 7$), and height of substrate (h). The model has 36 outputs, i.e., self-inductance of each conductor ($L_{ii}, i=1, 2, \dots, 8$) and mutual inductance between any two conductors ($L_{11}, L_{12}, \dots, L_{78}$). Data is generated from EM simulation using LINPAR [167]. To improve the performance of the neural model at the boundaries of the input space of interest (also referred to as model utilization range), two sets of training data are used. Training set A with 2000 samples is generated within the model utilization range. Training set B with 3000

samples is generated within and slightly beyond the boundaries. A total of 5000 test samples are generated again within the model utilization range following discussions in 2.3.2.

Figure 3.8. Eight-conductor stripline to be modeled for subsequent use in the design of multi-layered printed-circuit boards.

The challenges in this example are the large size of the neural network and the huge amount of training data, because of high-dimension of design parameter space. The iterative decomposition idea of the IMS algorithm reduces this huge problem into several sub-tasks, i.e., different training stages, and training in each stage can be relatively easier. In the first and second stages, training sets A and B are both used for ANN-training. In the first stage, a 3-layer MLP with 36 hidden neurons is used. The sub neural model of this stage captured most dominant tendencies ignoring higher-order effects. In the second stage, a 3-layer MLP with 50 hidden neurons is used to capture the second order behavior.

Starting from the third stage, the focus is exclusively on the model utilization range and data set A alone is used for training. In the third stage, MLP3 with 60 hidden neurons is used. In fourth and fifth stages, 4-layer MLP (16-40-40-36) is used because training data representing higher-order behaviors in these stages is observed to be highly nonlinear. After five stages, the accumulated neural model predicts the self and mutual inductances with an average test error of 1.63%. The stage-wise training results are presented in Table 3.6.

Table 3.6. Stage-wise training results of the eight-conductor stripline modeling example using the proposed IMS algorithm.

Stage Number	Average Test Error of Accumulated Model	Average Test Error of Sub Model	Neural Network Structure	Error Measure
k	$(\tilde{y}^k)$	$(\tilde{Y}^k)$	(H^k)	(C^k)
1	10.68%	10.68%	MLP3	1.20E-6
2	3.66%	2.84%	MLP3	1.03E-6
3	2.50%	5.07%	MLP3	5.67E-7
4	1.80%	4.50%	MLP4	4.65E-7
5	1.63%	5.41%	MLP4	3.85E-7
6	-	-	-	2.64E-7

Conventional one-stage approach to model this problem requires a large neural network to be trained by huge amount of training data. Training algorithms with faster convergence such as quasi-Newton and LM are not suitable due to extremely large memory requirements. For comparison with IMS algorithm, a MLP3 with 146 hidden neurons (equivalent to hidden neurons used upto the first three stages of IMS) has been trained using conjugate-gradient algorithm. After a long training, the one-stage model reached an accuracy of 4.0% (average test error). Continued training for a much longer time resulted in a final model accuracy of 3.8% with no further improvement. The IMS algorithm required half the training time to break this barrier and reached an accuracy of 2.5% at the end of third stage. The model accuracy further improved to 1.63% at the end of five stages.

3.5 Summary

In this chapter, a new iterative multi-stage training algorithm has been presented. The IMS algorithm has the potential to simplify highly nonlinear and high-dimensional RF/microwave neural network modeling problems, through iterative decomposition and multi-stage training. For the first time, the IMS algorithm facilitates the use of multiple neural network structures and training algorithms to model an RF/microwave modeling task. The integration of a novel HQN training technique into the IMS framework allows reliable ANN-modeling even in the presence of large/sudden variations or gross errors in training data. The effectiveness of the proposed algorithm has been demonstrated through active device, passive component and circuit waveform examples.

CHAPTER 4

Automatic Model Generation Algorithm

In this chapter, a major contribution of this thesis, namely automatic model generation (AMG) algorithm [9][10] is presented. In brief, the proposed AMG algorithm opens the door for an exciting arena of neural-network-based RF/microwave modeling, i.e., “*automation*”. For the first time, a broad framework for automatic generation of neural models, including new concepts of neural network error-based adaptive data sampling, simultaneous training and validation data augmentation in an inter-twined fashion dynamically during neural network training, and systematic neural-network-structure adaptation, is developed. A step-by-step description of the AMG framework is presented in the sections that follow.

4.1 Introduction

Neural network based component or device modeling, which is generally followed by neural-model-based circuit design, has been successfully applied to numerous high-frequency CAD problems. The progress of ANN-based RF/microwave CAD depends on innovative research activities that can further strengthen it in terms of accuracy and speed, and make it even more attractive for practical applications through cost-effective, easy-to-use neural-network-modeling approaches. Research work presented in this chapter touches upon these key aspects through “*automation*” of the overall neural model development process.

Automation has a significant role in ANN-modeling. As elaborated in chapter 2, neural model development process involves several sub-tasks such as data generation and scaling, structure and algorithm selection, neural network training, and neural model validation [7]. Conventionally, these sub-tasks are manually carried out in a sequential manner independent of one another. Such an approach, referred to as the step-by-step manual neural modeling, requires human experience, effort and involvement [9]. Modeling is just one aspect of RF and microwave CAD and the designers wish to quickly develop and use neural network component models in their circuit design, even though they do not have enough exposure to ANN model development issues. As such, there is a definite need for automation of neural model development process.

It is to be noted that a successful automation technique needs to address multiple, complicated, and inter-dependent practical high-frequency modeling challenges. In addition it needs to compensate for lack of manual intervention or navigation. RF/microwave modeling problems often involve a large number of model parameters and the number of training/validation data samples needed for developing a satisfactory neural model remains a question. Highly nonlinear nature of RF/microwave input-output characteristics, on the other hand, makes data sampling (i.e. distribution of data samples in the model input parameter space) a crucial issue for the automation technique to consider. Neural network size, which used to be handled earlier by manual trial-and-error approach (e.g. Table 2.1), now needs an automatic method. In addition, a mechanism for dealing with accidental large errors [5] is desired in the automation framework.

The subject of automated neural-network-modeling has not been dealt with in the existing EM-ANN literature. However, research in the neural-based RF/microwave CAD area toward meeting certain other goals resulted in some notable techniques. Neural networks with embedded knowledge [3][6] have been shown to facilitate improved neural model accuracy when limited data is available. A variety of sample distributions, e.g. central-composite distribution [6], star distribution [35] etc., have been suggested for exploring the multi-dimensional neural model input-space. The MLPNN process [7] advocated a need for neural network size adjustment based on problem complexity. Research in the neural network area also led to some techniques, e.g. CasPer [168] that begins with a minimal neural network and add neurons during training. While some of the above-listed techniques are derived for signal-processing applications, others merely address a particular sub-task of neural network model development process.

A novel contribution of the thesis namely, automatic model generation algorithm that integrates all the sub-tasks involved in neural modeling into a unified framework facilitating an automated RF/microwave neural model development process, is presented in this chapter. Starting with zero amounts of training data and then proceeding with neural network training in a stage-wise manner, the AMG can automatically produce a neural model that exceeds user-specified accuracy. In each stage, the algorithm utilizes neural network error criteria to determine additional training samples required and their location in model input space. Initially, fewer hidden neurons are used, and the AMG has a built-in mechanism for ANN-size adjustment. The AMG algorithm significantly reduces the intensive human effort demanded by step-by-step neural modeling approach.

4.2 Automation Issues: An Insight

Having recognized the significance of automation of neural model development process, the next step is to devise a strategy for realizing an automation algorithm. In this section, each of the issues critical to automation is treated individually and a hypothesis is formed. Firstly, unlike the manual step-by-step neural modeling approach that is supported by human experience and/or problem-specific knowledge, a fully automated neural modeling approach receives no external help in terms of decision-making involved. In this thesis, an algorithmic approach is therefore advocated for evolving an automated approach to ANN-modeling.

Secondly, neural network training and neural model validation in the manual approach are sequential and hence training and validation sets can be completely independent of one another. In an algorithmic automation approach, training and validation may have to proceed hand-in-hand. Toward this end, the thesis promotes systematic augmentation of training and validation data sets in an inter-twined fashion. In other words, some of the validation data in the current stage of neural network training can be transferred to the training set in the subsequent stage. In addition, algorithmic approach to neural model development is expected to involve (dynamic) data generation during ANN-training. This novel idea necessitated implementation of simulator drivers [11] for realizing the automation scheme.

Another issue for consideration is a built-in mechanism for controlling sampling issues (e.g. number of samples and their distribution in model input-space), in the

absence of manual guidance. General guidelines that are noteworthy include, (i) too few samples lead to over-learning of the neural network [1][9], (ii) highly nonlinear input-output RF/microwave behaviors need more samples, and (iii) the more the number of samples the better it is for ANN-training. Each of these guidelines is taken into account in formulating the automated approach to neural-modeling.

Neural network size, i.e., number of hidden layer neurons is yet another key issue. A systematic methodology for adapting neural network size is to be developed. While too many hidden layer neurons need more CPU-time for ANN-training and too few hidden neurons can result in under-learning of the neural network. The issue of accidental large errors in training data is also critical. In the step-by-step approach, it may be possible (with great difficulty) to manually check and remove erroneous data samples from the training set. In a successful automation approach, a special check for detection of such errors is mandatory. In summary, a lot of manual effort and periodic human decisions are involved in the step-by-step approach, that needs to be avoided in an automated approach.

During this thesis research, the above-mentioned ideas have all been tried and tested, eventually leading to a new AMG algorithm, which is the first of its kind in the existing EM-ANN literature. Given an RF/microwave ANN-modeling problem $y=f(x)$, user provides the AMG algorithm with corresponding high frequency simulation software and specifies the desired neural model accuracy relative to the training/validation data from the simulator (among other specifications). The AMG

algorithm automatically generates a satisfactory neural network model $\tilde{y}=h(x,w)$ representing the detailed theoretical relationship f .

4.3 AMG Algorithm

Contrary to the step-by-step approach, the automatic model generation (AMG) algorithm presented here focuses on the mutual dependence of various sub-tasks like data generation, neural network size adjustment, neural network training and validation. The AMG algorithm links these sub-tasks through neural network learning phenomena (e.g. over-learning), and integrates them into one unified process. The integrated process is computerized and is carried out automatically in a stage-wise fashion. Within a stage, the algorithm facilitates periodic communication between various sub-tasks, thus enabling adjustment or enhancement in the execution of a sub-task based on the feedback from other sub-tasks. As a result, each stage could involve dynamic incremental data generation, neural network size adjustment, neural network training with training data, and neural model testing with validation data. The algorithm has built-in simulation drivers (e.g. *HFSS*-driver, *ADS*-driver) for facilitating automatic data generation during ANN training. Accidental large errors that could occur in training data during dynamic data generation are detected and the effect of these errors is neglected by automatic switching to HQN training algorithm. The technique establishes a quantitative link between neural model accuracy, the number and distribution of training/validation data, and neural network size. Automation of the model development process shifts the workload from human to computer thus making model development more efficient and less prone to errors.

4.3.1 Notation

Two disjoint sets of data namely, the training data and the validation data, are used. Training data is used to update neural network weights during training and validation data is used to monitor the quality of neural model during training. T^k and V^k are defined as index sets of training (learning) and validation data during k^{th} stage, i.e., $T^k = \{i \mid (\mathbf{x}_i, \mathbf{d}_i) \text{ is a training sample}\}$ and $V^k = \{j \mid (\mathbf{x}_j, \mathbf{d}_j) \text{ is a validation sample}\}$. For the first time, the proposed AMG treats T^k and V^k as variables during neural network model generation. Let $\mathbf{h}^k(\mathbf{x}, \mathbf{w}^k)$ represent neural network structure in k^{th} stage with outputs $\tilde{\mathbf{y}}^k = \mathbf{h}^k(\mathbf{x}, \mathbf{w}^k)$, where $\mathbf{w}^k$ is the corresponding weight vector. The range of input parameter space in which the neural model would be used during design is referred to as input space of interest and is denoted by R_0 .

Following equation (2.22), normalized training error of neural network $\mathbf{h}^k(\mathbf{x}, \mathbf{w}^k)$ at the end of k^{th} stage is re-defined as

$$\hat{E}_T^k = \frac{1}{N_T^k} \sum_{p \in T^k} (e_p(\mathbf{w}^k))^q, \quad (4.1)$$

where q represents the q^{th} norm, N_T^k is the number of samples in T^k , and $e_p(\mathbf{w}^k)$ is the error due to p^{th} sample in T^k given by

$$e_p(\mathbf{w}^k) = e(\mathbf{x}_p, \mathbf{w}^k) = \frac{1}{m} \left[\frac{1}{q} \sum_{j=1}^m \left| \frac{\tilde{y}_{jp}^k - d_{jp}}{d_{\max,j} - d_{\min,j}} \right|^q \right]^{\frac{1}{q}}. \quad (4.2)$$

Here $\tilde{y}_{jp}^k$ is the j^{th} output of neural network $\mathbf{h}^k(\mathbf{x}, \mathbf{w}^k)$ at the end of k^{th} stage when input

x_p is presented, and d_{jp} is the j^{th} element of $\mathbf{d}_p$. In (4.2), $d_{\min,j}$ and $d_{\max,j}$ are the minimum and maximum values of j^{th} output data in $\mathbf{d}_k$, within the input space of interest.

The objective of neural network training in k^{th} stage is to find $\mathbf{w}^k$ such that $\hat{E}_T^k$ is minimized. Similar to the normalized training error, a normalized validation error of neural network $\mathbf{h}^k(\mathbf{x}, \mathbf{w}^k)$ at the end of k^{th} stage training can be defined as

$$\hat{E}_V^k = \frac{1}{N_V^k} \sum_{p \in V^k} (e_p(\mathbf{w}^k))^q, \quad (4.3)$$

where N_V^k is the number of samples in V^k , and $e_p(\mathbf{w}^k)$ is the error due to k^{th} sample in V^k computed following (4.2). Let E_d represent user-desired neural model accuracy (validation error). The objective of the proposed AMG algorithm is to automatically carry out stage-wise neural model development process until $\hat{E}_V^k \leq E_d$.

4.3.2 Key Aspects of the Algorithm

To begin with, the algorithm considers the original bounded n -dimensional input space of interest R_0 as one region. For the first stage, training data (T^1) and validation data (V^1) are systematically generated in R_0 in a pre-defined way (e.g. star distribution). In this work, training and validation data for a given region are generated as shown in Figure 4.1(a). A first stage neural network $\mathbf{h}^1(\mathbf{x}, \mathbf{w}^1)$ with relatively fewer hidden layer neurons is trained with data samples in T^1 , i.e., $(\mathbf{x}_i, \mathbf{d}_i)$, $i \in T^1$. The resulting neural model is validated with data samples in V^1 , i.e., $(\mathbf{x}_j, \mathbf{d}_j)$, $j \in V^1$. The algorithm stops if $\hat{E}_V^1 \leq E_d$.

Otherwise, based on the neural network error criteria, the algorithm automatically takes suitable actions and proceeds to the next stage of model development.

4.3.2.1 Automatic Sampling and Data Generation

Over-learning of the neural network may be detected at the end of k^{th} stage, using error information $\hat{E}_T^k$ and $\hat{E}_V^k$. Over-learning is a phenomenon in which the neural network memorizes the training data accurately but cannot generalize well, i.e., $\hat{E}_T^k$ is small enough (compared to E_d) but $\hat{E}_V^k \gg \hat{E}_T^k$. When the algorithm detects over-learning, it dynamically adds more data samples to the training and validation sets. Motivated by the concepts of sampling based on multinomials [169], rational interpolation [170], and error-based data exploration [171], a unique neural network oriented technique featuring the growth of training and validation sets in an intertwined way, is developed. Utilizing this technique, the AMG algorithm handles the issues of number of additional samples and their distribution in model input space.

The validation sample index $p^* \in V^k$ with maximum (worst case) error is determined by

$$p^* = \arg \max_{p \in V^k} \|\mathbf{h}^k(\mathbf{x}_p, \mathbf{w}^k) - \mathbf{d}_p\|, \quad (4.4)$$

and corresponding worst-case validation data sample $(\mathbf{x}_{p^*}, \mathbf{d}_{p^*})$ is identified. A simple notation $(\mathbf{x}^*, \mathbf{d}^*)$ is used to represent the worst-case validation sample. The worst region R^* to which $\mathbf{x}^*$ belongs is further divided (split) into 2^n new sub-regions. Considering these new regions, T^k and V^k are updated, i.e.,

$$T^{k+1} = T^k \cup T^{new}, \quad (4.5)$$

$$V^{k+1} = V^k \cup V^{new}, \quad (4.6)$$

where T^{new} and V^{new} represent the index sets of new data samples to be generated. Each new training input point $\mathbf{x}_{T,new}$ is obtained as

$$\mathbf{x}_{T,new} = \mathbf{x}^* + \mathbf{P}_i \frac{\mathbf{x}_{\max} - \mathbf{x}_{\min}}{2^{r+1}}, \quad i = 1, 2, \dots, N_T^{new}, \quad (4.7)$$

where $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$ are the extreme boundaries of R^* , r is the number of splits after which R^* was formed, N_T^{new} is the number of newly added training points, and $\mathbf{P}_i$ is $n \times n$ diagonal matrix.

Each diagonal element of $\mathbf{P}_i$ can take one of the values 0, +1, or -1, depending upon the pre-defined sample distribution in each sub-region. Each new validation input point $\mathbf{x}_{V,new}$ is obtained as

$$\mathbf{x}_{V,new} = \mathbf{x}^* + \mathbf{Q}_j \frac{\mathbf{x}_{\max} - \mathbf{x}_{\min}}{2^{r+2}}, \quad j = 1, 2, \dots, N_V^{new}, \quad (4.8)$$

where N_V^{new} is the number of newly added validation points and $\mathbf{Q}_j$ is a diagonal matrix just like $\mathbf{P}_i$. Incremental training and validation data outputs are generated corresponding to the newly added input points by dynamic driving of simulators (e.g. 3D-EM).

For example, suppose the original 2-dimensional input space (region) shown in Figure 4.1(a) needs to be split because the first stage of training did not yield a satisfactory neural model. We then have $\mathbf{x}^* = [2 \ 2]^T$, $\mathbf{x}_{\min} = [0 \ 0]^T$, $\mathbf{x}_{\max} = [4 \ 4]^T$, and $r = 0$.

For the second stage, five new training samples are generated with

$$\mathbf{P}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad (4.9)$$

$$\mathbf{P}_2 = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}, \quad (4.10)$$

$$\mathbf{P}_3 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad (4.11)$$

$$\mathbf{P}_4 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad (4.12)$$

and

$$\mathbf{P}_5 = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}. \quad (4.13)$$

Four new validation samples are generated with

$$\mathbf{Q}_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad (4.14)$$

$$\mathbf{Q}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad (4.15)$$

$$\mathbf{Q}_3 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad (4.16)$$

and

$$\mathbf{Q}_4 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \quad (4.17)$$

as shown in Figure 4.1(b).

Figure 4.1. (a) Training and validation data in a typical sub-region of 2-dimensional input space. (b) When the sub-region is identified as the worst region, it is further divided into smaller regions. Newly generated training and validation data are indicated.

4.3.2.2 Automatic Neural Network Structure Adaptation

Under-learning of the neural network may be detected after k^{th} stage, using training error $\hat{E}_T^k$ and its gradient $\Delta\hat{E}_T^k = \hat{E}_T^{k-1} - \hat{E}_T^k$. Under-learning is a phenomenon in which the neural network has difficulties in learning the training data itself, i.e., $\hat{E}_T^k$ is large and $\Delta\hat{E}_T^k$ is small. When the AMG algorithm detects under-learning, it dynamically adds more hidden layer neurons. A larger neural network would then provide increased freedom to better learn the nonlinearities in training data. If the neural network structure $\mathbf{h}^k(\mathbf{x}, \mathbf{w}^k)$ has N_h^k hidden neurons, $\mathbf{h}^{k+1}(\mathbf{x}, \mathbf{w}^{k+1})$ with $N_h^{k+1} = N_h^k + \delta$ hidden neurons is used by the algorithm in the $k+1^{\text{th}}$ stage. Suggested range for the newly added hidden neurons δ is 10-20 % of N_h^k .

4.3.2.3 Automatic Handling of Large Errors in Data

As mentioned earlier, it is possible that a few accidental large errors occur in training data during dynamic RF/microwave data generation of the AMG algorithm. The algorithm must be able to detect these large errors and discard them during neural network training, as there is no place for human intervention in an automated approach. The large errors are detected as a special case of under-learning, i.e., when $\hat{E}_T^k$ continues to be large and $\Delta\hat{E}_T^k$ remains small for several consecutive stages even after repeatedly adding hidden neurons.

Once large errors are detected, the neural network training in AMG switches from conventional neural network training algorithms (e.g. quasi-Newton) to Huber quasi-

Newton technique described in section 3.3. Assume that the AMG algorithm detects large errors in training data in the k^{th} -stage. Following equation (3.21), a normalized $k+1^{\text{th}}$ stage training-objective $\hat{E}_T^{k+1}(\mathbf{w}^k)$ is defined utilizing Huber function $\rho(\cdot)$ as

$$\hat{E}_T^{k+1}(\mathbf{w}^{k+1}) = \frac{1}{N_T^{k+1}} \sum_{p \in T^{k+1}} \rho[e_p(\mathbf{w}^{k+1})] . \quad (4.18)$$

Training of neural network $\mathbf{h}^{k+1}(\mathbf{x}, \mathbf{w}^{k+1})$ in the $k+1^{\text{th}}$ stage using the HQN algorithm aims at minimizing this new training objective, for efficient learning of the original problem behavior neglecting the large errors in data.

4.3.2.4 Overall Automation

At the end of each stage, the AMG checks for various possible ANN-training situations and takes relevant actions, e.g. update data, adjust network size etc. In the subsequent stage, neural network $\mathbf{h}^{k+1}(\mathbf{x}, \mathbf{w}^{k+1})$ is trained with samples in T^{k+1} and the neural model is tested with samples in V^{k+1} . A few conservative options are also incorporated to make the algorithm more effective. (i) Periodically, after a fixed number of stages, incremental training/validation samples are generated in a randomly chosen sub-region instead of a worst sub-region. (ii) The algorithm terminates only if $\hat{E}_V^k \leq E_d$ consecutively for a given number of stages. A framework of the proposed AMG algorithm is shown in Figure 4.2.

4.3.2.5 Implementation Details

Let $\hat{E}_T^k$, $\hat{E}_V^k$ and $\Delta\hat{E}_T^k$ represent training error, validation error and gradient of the training error of the neural network $h^k(x, w^k)$ at the end of k^{th} stage. Let $\mathfrak{R}$ represent a set of regions in the neural model input space. The main user inputs to the AMG are, user-desired neural model accuracy E_d and maximum allowable number of stages k_{max} . There are other user-inputs $k_c, \beta, \xi, k_r, \psi, \Delta E, k_u$ and k_e . For $E_d = 1\%$, suggested ranges of these constants are $1 \leq k_c \leq 5$, $0 \leq \beta \leq 1$, $\xi \geq 1$, $4 \leq k_r \leq 6$, $2 \leq \psi \leq 4$, $0.01 \leq \Delta E \leq 0.1$ (in % $\hat{E}_T^k$), $1 \leq k_u \leq 6$, and $4 < k_e \leq 6$. Pseudocode of the algorithm is presented here.

Initialization: $k = 1$, $N_h^1 = N_0$, $T^1 = \{i \mid x_i \text{ is a vertex of } R_0\}$, $V^1 = \{j \mid x_j \text{ is the center of } R_0\}$, and $flag = 0$. Automatically generate d_i and d_j for all x_i and x_j . There is only one region, $R_0 \in \mathfrak{R}$. Train neural network $h^1(x, w^1)$ using samples (x_i, d_i) , $i \in T^1$, and test the neural model with (x_j, d_j) , $j \in V^1$. *Activate Data Generation*(R_0).

Activate Data Generation(R): Split R into 2^n new regions. Delete R from $\mathfrak{R}$ and add the new regions to $\mathfrak{R}$. Update T^k and V^k , and generate training and validation data for the new sub-regions by automatically driving the simulator. Go to *Automatic Training*.

Automatic Training: $k = k+1$. Train $h^k(x, w^k)$ using data samples in T^k . Test the neural model with samples in T^k and V^k to obtain $\hat{E}_T^k$ and $\hat{E}_V^k$ respectively.

if $(k > k_{max})$ or $(\hat{E}_v^k \leq E_d \text{ for } k_c \text{ consecutive stages})$ *Stop Training*.

else if $(\hat{E}_T^k < \beta E_d)$ and $(\hat{E}_v^k > \xi E_d)$, over-learning is detected. Set $flag = 0$.

if $((k \% k_r) \neq 0)$ choose worst sample index $p^* \in V^k$ and identify the corresponding sub-region R^* . Set $N_h^{k+1} = N_h^k$ and *Activate Data Generation*(R^*).

else randomly choose a region $R^* \in \mathfrak{R}$. Set $N_h^{k+1} = N_h^k$. *Activate Data Generation*(R^*).

else if $(\hat{E}_T^k > \psi E_d)$ and $(\Delta \hat{E}_T^k < \Delta E \text{ for } k_u \text{ consecutive stages})$, under-learning is detected.

if $((flag \% k_e) = 0)$, training data has accidental large errors. Set $N_h^{k+1} = N_h^k$, $T^{k+1} = T^k$, $V^{k+1} = V^k$. *Switch Training Algorithm* from *quasi-Newton* to *HQN*.
Go to *Automatic Training*.

else *Add Hidden Neurons*, i.e., $N_h^{k+1} = N_h^k + \delta$. Set $T^{k+1} = T^k$, $V^{k+1} = V^k$.

$flag = flag + 1$. Go to *Automatic Training*.

else possible local minimum in neural network training. Set $N_h^{k+1} = N_h^k$, $T^{k+1} = T^k$, $V^{k+1} = V^k$. Set $flag = 0$. Randomly perturb network weights w^k . Go to *Automatic Training*.

Figure 4.2. Systematic framework of the proposed automatic model generation algorithm in the form of a flow-chart.

4.4 Demonstration Examples

4.4.1 MESFET

The proposed AMG algorithm is incorporated into *NeuroModeler* software [18]. Input space $\mathbf{x}$ of the neural network contains gate-length (l), channel thickness (a), gate-source voltage (v_g) and drain-source voltage (v_d). Drain current (i_d) is the neural network output y . For a user-desired neural model accuracy E_d , the number of training and validation samples needed and their distribution in (l, a, v_g, v_d) space, and the neural network size are not known. The proposed algorithm is used to develop a neural model to represent the physics-based *Khatibzadeh* and *Trew* MESFET model of *OSA90* [12].

The algorithm starts with an initial neural network (3-layer MLP) with $N_h^1 = 9$ hidden neurons. In the first stage, the algorithm dynamically generates $N_T^1 = 16$ samples to train the neural network and $N_V^1 = 1$ validation sample to test it. During subsequent stages, the algorithm automatically decides the number and distribution of additional training and validation samples needed and dynamically drives the *OSA90* simulator using in-house *OSA90*-driver. Extra hidden layer neurons are also automatically added as needed. For $E_d = 0.5\%$, the user-inputs of the algorithm are set to $k_c = 1$, $\beta = 1$, $\xi = 1$, $k_r = 5$, $\psi = 2$, $\Delta E = 0.05\%$, $k_u = 1$, and $k_e = 6$. The AMG algorithm produced a neural model with a validation error $\hat{E}_V^8 = 0.48\%$ after 8 stages of model development. A total of $N_T^8 = 318$ training samples and $N_V^8 = 90$ validation samples are used and the final neural model has $N_h^8 = 16$ hidden neurons.

To confirm the reliability of the generated MESFET neural model, it is further tested with a large set of independent test data never seen during AMG's model development process. Average test error is observed to be 0.49%, which satisfies the $E_d = 0.5\%$ specification. Using the manual step-by-step neural modeling approach, MLP networks with 9, 12 and 16 hidden neurons are trained using uniform-grid samples and the average test errors are reported in Table 4.1. It can be seen that the proposed AMG gives more accurate neural models with similar amounts of training data as compared to step-by-step approach. This is because the AMG algorithm uses efficient sample distribution, i.e., more (less) data are generated in nonlinear (smooth) regions as shown in Figure 4.3.

4.4.2 Embedded Capacitor

Accurately modeling 3D-EM behaviors of embedded components [172] is important for efficient CAD of high-speed multi-layer printed circuit boards (PCB). In this example, neural model of an embedded capacitor shown in Figure 4.4 is developed. The $f(x)$ relationship of the capacitor is available from CPU-intensive EM simulations of *Ansoft-HFSS* simulator [13]. The step-by-step neural modeling approach based on manual data generation is prohibitive because it involves the following tasks, (i) Re-drawing the capacitor with new physical parameter values graphically (or editing simulator input file for every new input vector x), (ii) Submitting and waiting for the CPU-intensive data computation to finish, (iii) Viewing simulator output file for the outputs of interest d , and (iv) Appending the new (x, d) sample pair to the training or validation data file. These exhaustive tasks make manual data generation highly time-consuming, human intensive and error-prone. The proposed AMG algorithm is used.

Table 4.1. Model accuracy comparison between the proposed AMG algorithm based on automatic sampling and the step-by-step approach based on conventional grid distribution for the MESFET. Model accuracy reported for the manual approach is an average validation error of three neural models with 9, 12 and 16 hidden neurons.

Proposed Automatic Model					Manual Step-by-Step Neural	
Stage No. (k)	No. of Training Samples (N_t^k)	No. of Hidden Neurons (N_h^k)	Action taken at the end of Stage	Model Accuracy ($\hat{E}_v^k$ in %)	Modeling Approach	
					No. of Training Samples used	Model Accuracy ($\hat{E}_v$ in %)
1	16	9	Add Samples	23.38	16	25.74
2	81	9	Add Samples	6.64	81	8.05
3	146	9	Add Neurons	4.32	144	6.42
4	146	12	Add Samples	3.54		
5	211	12	Add Samples	1.82	225	3.20
6	256	12	Add Samples	0.90	256	1.28
7	318	12	Add Neurons	0.75	320	0.75
8	318	16	Stop	0.48

Figure 4.3. Inter-twined distribution of training and validation data by the proposed AMG algorithm for the MESFET example. (a) First stage, (b) Second stage, (c) Third stage, and (d) Fourth stage of training. The algorithm identifies non-linear sub-regions of the MESFET input-space and automatically generates more samples in such regions.

Input $\mathbf{x}$ of the capacitor neural network includes length (l), thickness (t), dielectric constant (ϵ_r), and frequency ($f = 0.1 - 20 \text{ GHz}$). Real and imaginary parts of two-port S-parameters are the neural model outputs $\mathbf{y}$. User-specified neural model accuracy is $E_d = 1.25\%$. In the first stage, a neural network $\mathbf{h}^1(\mathbf{x}, \mathbf{w}^1)$ with $N_h^1 = 12$ hidden neurons is used. A total of 16 training and one validation samples are used. Whenever the AMG decides to add more data, it dynamically drives the *Ansoft-HFSS* simulator using an in-house *HFSS*-driver. After 10 stages, the final neural model $\mathbf{h}^{10}(\mathbf{x}, \mathbf{w}^{10})$ with $N_h^{10} = 20$, 554 training and 136 validation samples, achieved an accuracy of $\hat{E}_V^{10} = 0.92\%$.

Figure 4.4. Embedded capacitor used in multi-layer PCB. S-parameter neural model of the capacitor is developed from 3D-EM data of *Ansoft-HFSS* using the AMG algorithm.

On the other hand, a MLP neural network with 20 hidden neurons trained using the manual step-by-step neural modeling approach with 554 uniform grid samples achieved a validation error of 6.8%. A comparison of training data shows that the automatic algorithm used 176 training samples in the sub-region $f \in (0.1 - 3GHz)$ where S-parameters exhibit a relatively nonlinear behavior, while the manual uniform-grid sampling used only 128 samples. The manual approach required 768 training samples to achieve a model accuracy of 1%. Neural model developed by the AMG algorithm is subjected to an independent test with a large set of data (8200 samples) never seen during training and the test error is observed to be 1.04%, further confirming the reliability of the AMG algorithm. A time comparison between the proposed algorithm and the step-by-step manual neural modeling approach is shown in Table 4.2. It can be seen that the human time required in the case of the proposed AMG is very small as compared to the manual approach. The reason is that, data generation in the AMG is automatic as opposed to manual data generation in step-by-step approach. The CPU time required by the AMG approach is also relatively smaller. This is because the manual approach requires relatively larger number of training samples to achieve similar model accuracy.

During automatic data generation using *HFSS*, a few large accidental errors are detected in S_{12} . These errors can be attributed to simulations in extreme locations of the model input space. The algorithm successfully detected these non-convergence simulation errors and automatically switched the neural network training process from quasi-Newton algorithm to HQN. The resulting neural model neglected these large errors and modeled only the non-erroneous data samples as can be seen in Figure 4.5.

Table 4.2. Comparison of the time taken by the proposed AMG algorithm and manual step-by-step neural modeling approach for the embedded capacitor.

	Proposed Automatic Model Generation Algorithm		Manual Step-By-Step Neural Modeling Approach	
	Human Time	CPU Time	Human Time	CPU Time
	5 min	1158 min	498 min	1625 min
Total Time	1163 min		2123 min	

4.4.3 Power Amplifier

This example is for the purpose of illustrating the applicability of the AMG algorithm to circuit modeling problems. The algorithm is used to develop a neural model to represent dynamic behaviors of a power amplifier circuit used in wireless communication systems. The amplifier shown in Figure 4.6 has eight NPN BJT transistors represented by two internal nonlinear models Q34 and Q37 of *Agilent-ADS* [14]. Input parameters for the power amplifier neural network are power input (P_{in}), input bias voltage (V_{din}), resistive load (R_L) and input frequency (f). Outputs of the neural model include power outputs at fundamental frequency, second harmonic and third harmonic denoted by P_f , P_{2f} and P_{3f} respectively. The $f(x)$ relationship of the amplifier is originally produced by *Agilent-ADS* simulator. A 3-layered MLP network shown in Figure 4.7 is used by the proposed AMG algorithm to model high-frequency characteristics of the power amplifier circuit.

Figure 4.5. Comparison between neural model prediction and training data for the embedded capacitor. The effect of large errors introduced in S_{12} training data during automatic data generation of the proposed AMG algorithm are discarded during neural network training of AMG as can be seen from the resulting neural model responses.

Figure 4.6. Circuit diagram of power amplifier used in wireless communication systems to be modeled using the AMG algorithm.

Figure 4.7. A 3-layer MLP neural network structure used by the proposed AMG algorithm to model dynamic behaviors of the power amplifier circuit.

The input parameter space of interest R_0 is bounded by $P_{in} = -20$ to $+20$ dBm, $V_{din} = 2$ to 4 Volts, $R_L = 20$ to 140Ω and $f = 0.7$ to 4.2 GHz. To facilitate dynamic data generation during neural network training, an *ADS*-driver is implemented to automatically drive *Agilent-ADS*. After 28 stages, MLP neural network with 60 hidden neurons achieved an accuracy of 1.25%. A total of 1758 training and 436 validation samples are used. A comparison of neural model prediction of the amplifier outputs with original data from *ADS* is shown in Figure 4.8. Quantitative relationship between number of training data and neural model accuracy as provided by the AMG algorithm is shown in Figure 4.9.

Figure 4.8. Comparison of neural model prediction of the amplifier power outputs at fundamental, second and third harmonics (—) with original *Agilent-ADS* data (various symbols) for different values of P_{in} with $V_{din} = 3V$ and $R_L = 80 \Omega$.

Figure 4.9. Quantitative relationship between neural model accuracy and number of training data for the power amplifier circuit as provided by the proposed AMG algorithm.

Discussion:

A notable aspect for a user is that as AMG is continued for more and more stages, the neural model accuracy keeps improving. In the process, more data needs to be automatically generated and used for training and validation, and neural model development time increases. However, the rate of improvement slows down as the user-specified accuracy criterion becomes tighter and tighter. This trend is observed in RF/microwave neural modeling problems and Figure 4.9 is one such example.

4.5 Summary

A robust AMG algorithm for automating the neural network based RF/microwave model development process has been proposed. Starting with no training data and then proceeding with neural network training in a stage-wise manner, the algorithm can automatically produce neural models of desired accuracies. In each stage, the algorithm utilizes neural network error criteria to determine additional training samples required and their location in the model input space. The new data samples are dynamically generated during training, by automatic driving of simulation tools (e.g., *OSA90*, *Ansoft-HFSS*, *Agilent-ADS*). Initially, fewer hidden neurons are used, and the algorithm adjusts the neural network size whenever it detects under-learning. The AMG algorithm significantly reduces intensive human effort demanded by the conventional step-by-step neural modeling approach. In microwave-modeling problems (e.g. 3D-EM) where data generation is extremely expensive, there is a noticeable reduction in the CPU-time for neural model development, as AMG uses fewer training samples. The algorithm has been demonstrated through practical RF/microwave device and circuit examples.

CHAPTER 5

Knowledge Based Automatic Model Generation

Algorithm

In this chapter, a novel contribution of this thesis namely, knowledge based automatic model generation (KAMG) algorithm [16][17] is presented. In brief, motivated by the space-mapping concept, the KAMG utilizes extensive coarse data but fewest fine data as compared to any other existing neural modeling techniques to generate neural models that accurately match fine data. The KAMG framework introduces fresh concepts of coarse and fine data generators; automatic generation of coarse and fine neural models; and reinforced ANN-learning from multiple types of data exploiting knowledge based neural networks. “*Automation*” is another key aspect of the KAMG algorithm that makes it even more attractive for practical use.

5.1 Introduction

Neural-based RF/microwave CAD approach has the combined merits of accuracy and speed [1][3]. It may however be noted that reliable ANN-based CAD solutions need accurate neural models, which in turn require lots of accurate training data. For example, extensive accurate training data from a detailed 3D-EM simulator is needed for developing a reliable embedded capacitor neural model with 3D-EM accuracy [9][10]. Data generation is expensive as it involves both CPU-time (for detailed model

computations) and human-time (for repetitive geometry changes), and this can slow down model development process. There has been continuous activity in the EM-ANN area aimed at accelerating neural model development through reduced use of data, while maintaining neural model accuracies. One of the recent advances is neural networks with knowledge [3][6][35][37] that utilize existing knowledge, thereby reducing the need for expensive training data. The AMG algorithm [10] is another breakthrough and MLP neural model generation using AMG has been shown to judiciously use fewer data based on adaptive sampling, and lesser training time through automation.

The motivation for this work is to investigate a possible way for simultaneously utilizing various types of available RF/microwave data via knowledge based neural networks, and further strengthen such an approach through automation of AMG algorithm. What are the available sources of RF and microwave training data, in general? The answer is, (i) Detailed theoretical models (e.g., 3D-EM models of passive components), and (ii) Approximate models (e.g., empirical equations). Detailed data leads to accurate neural models, but data generation and hence neural model development are both CPU-expensive. On the other hand, approximate data results in faster neural model development process, but in this case neural model accuracy is sacrificed. This dilemma between choices of neural model accuracy and speed of neural model development is critical and hence a strong motivation for this work.

In this chapter, a robust knowledge based automatic model generation algorithm that simultaneously utilizes both accurate and approximate data is presented. Motivated by

space-mapping concept [173][174], the KAMG uses extensive amounts of approximate data (also known as coarse data) together with fewest accurate data (also referred to as fine data) to generate neural network models of highest fine accuracies as compared to all other existing neural modeling techniques. A variety of knowledge neural network architectures are exploited to enable both coarse and fine data to contribute toward reinforced neural network learning of detailed or fine RF/microwave behaviors. The KAMG framework also incorporates stage-wise training and adaptive training-data sampling to facilitate automatic generation of neural models. As a result, the KAMG algorithm leads to faster neural model development process, while retaining highest neural model accuracies.

5.2 Data Generator

In general, f defined in (2.3) such that $y = f(x)$, represents a detailed theoretical RF/microwave input-output relationship to be modeled using a neural network. For example, f can be an EM relationship in the case of passive components or a semiconductor physics relationship in the case of active devices. The detailed relationship f can be accessible to RF/microwave designers in the form of simulation software $d = g(x)$. Here, a new terminology is introduced and a simulator or any other source of training data “ g ” is referred to as a “*data generator*” for better understanding. For a given input x , data generator $g(x)$ can be used to compute the outputs d . Data generation therefore involves repetitive use of g to obtain sample pairs (x_p, d_p) , where p is the sample index.

5.3 KAMG Algorithm

5.3.1 KAMG Notation

In practice, a given RF/microwave input-output relationship to be modeled can be accessible in the form of different simulation tools (i.e., different g 's). Some g 's are implemented using detailed theories such as the full-wave EM equations, and are accurate but can be slow. On the other hand, some g 's are based on simplified theories, and are therefore approximate and fast. Here, g_c and g_f are defined as coarse data generator and fine data generator respectively. For example, $\mathbf{x}$ can include length of an embedded capacitor and signal frequency, and $\mathbf{d}_c = g_c(\mathbf{x})$ and $\mathbf{d}_f = g_f(\mathbf{x})$ can represent corresponding S-parameters computed from the coarse (e.g., 2.5D-EM) and the fine (e.g., 3D-EM) data generators (simulators) respectively. Let

$$\tilde{\mathbf{y}}_c = \mathbf{h}_c(\mathbf{x}, \mathbf{w}_c), \quad (5.1)$$

$$\tilde{\mathbf{y}}_s = \mathbf{h}_s(\mathbf{x}, \mathbf{w}_s), \quad (5.2)$$

and

$$\tilde{\mathbf{y}}_f = \mathbf{h}_f(\mathbf{x}, \mathbf{w}_f), \quad (5.3)$$

be coarse, sub, and fine (overall) neural network models respectively. Here $\mathbf{h}_c$, $\mathbf{h}_s$, and $\mathbf{h}_f$ are coarse, sub, and fine (overall) neural networks, and $\mathbf{w}_c$, $\mathbf{w}_s$, and $\mathbf{w}_f$ are corresponding weight vectors. Let $\mathbf{d}_{cp} = g_c(\mathbf{x}_p)$ and $\tilde{\mathbf{y}}_{cp} = \mathbf{h}_c(\mathbf{x}_p, \mathbf{w}_c)$ be coarse training data output and coarse neural network model output respectively corresponding to an input $\mathbf{x}_p$, $p \in T_c$, where T_c is the index set of coarse training data. Similarly, we define $\mathbf{d}_{fp} = g_f(\mathbf{x}_p)$ and $\tilde{\mathbf{y}}_{fp} = \mathbf{h}_f(\mathbf{x}_p, \mathbf{w}_f)$ as fine training data output and fine neural network

model output respectively corresponding to an input $\mathbf{x}_p$, $p \in T_f$, where T_f is the index set of fine training data.

5.3.2 KAMG Formulation

The main objective of KAMG algorithm is to automatically generate a neural model $\tilde{\mathbf{y}}_f = \mathbf{h}_f(\mathbf{x}, \mathbf{w}_f)$ that accurately matches fine data from $\mathbf{g}_f$ over the entire $\mathbf{x}$ -space (input-space) of interest over which the generated neural model would be used during CAD. This is accomplished through extensive use of coarse training data from $\mathbf{g}_c$ and minimized use of expensive fine training data from $\mathbf{g}_f$. Automatic neural model generation using KAMG algorithm is accomplished through three major phases described in the following sections.

5.3.2.1 Phase I: Generation of Coarse Neural Network Model

In the first phase, the objective of KAMG is to generate a coarse neural network model $\tilde{\mathbf{y}}_c = \mathbf{h}_c(\mathbf{x}, \mathbf{w}_c)$. A neural network $\mathbf{h}_c$ is trained to learn the coarse RF/microwave relationship using coarse training data from $\mathbf{g}_c$. Training objective here is to minimize the difference between coarse neural model outputs and coarse data generator outputs, i.e.,

$$\min_{\mathbf{w}_c} \sum_{p \in T_c} \|\mathbf{h}_c(\mathbf{x}_p, \mathbf{w}_c) - \mathbf{d}_{cp}\|, \quad (5.4)$$

by adjusting weight vector $\mathbf{w}_c$ using neural network training (optimization) algorithms, e.g., quasi-Newton [97]. For any input $\mathbf{x}_p$ in coarse training set, vector $\mathbf{d}_{cp}$ represents corresponding output parameters. In this phase, coarse training data from $\mathbf{g}_c$ is used extensively because $\mathbf{g}_c$ is fast and inexpensive. The training ensures that $\mathbf{h}_c$ captures the

coarse $\mathbf{x}$ - $\mathbf{y}$ relationship available in the form of $\mathbf{g}_c$, i.e., neural model $\tilde{\mathbf{y}}_c = \mathbf{h}_c(\mathbf{x}, \mathbf{w}_c)$ accurately matches coarse data from $\mathbf{g}_c$ over the entire $\mathbf{x}$ -space of interest.

The process of coarse neural model development including data generation using $\mathbf{g}_c$ and training of neural network $\mathbf{h}_c$ is achieved automatically through adaptive sampling and stage-wise training [9][10]. Data sets T_c and V_c are empty initially and are updated in every training-stage by automatic driving of $\mathbf{g}_c$ using a suitable simulator-driver [10][16]. Initially, the n -dimensional $\mathbf{x}$ -space of interest (i.e., neural model utilization range) is considered as a set of sub-regions. For the first stage of multi-stage training, coarse training data (T_c) and coarse validation data (V_c) are systematically generated covering all the sub-regions in a pre-defined way (e.g., star distribution [35]). Neural network $\mathbf{h}_c$ is trained with data samples in T_c , i.e., $(\mathbf{x}_p, \mathbf{d}_{cp})$, $p \in T_c$, and validated with samples in V_c , i.e., $(\mathbf{x}_p, \mathbf{d}_p)$, $p \in V_c$. Worst validation sample $p^* \in V_c$, with maximum error between coarse neural model and coarse data is identified following (4.4) as

$$p^* = \arg \max_{p \in V_c} \left\| \mathbf{h}_c(\mathbf{x}_p, \mathbf{w}_c) - \mathbf{d}_{cp} \right\| \quad (5.5)$$

The sub-region to which this worst sample belongs is further divided into smaller sub-regions. Additional training and validation data are generated in the new regions again by automatic driving of $\mathbf{g}_c$. The coarse training and validation index sets are augmented with new coarse data samples as

$$T_c^{next} = T_c \cup T_c^{new}, \quad (5.6)$$

$$V_c^{next} = V_c \cup V_c^{new}, \quad (5.7)$$

where T_c^{new} and V_c^{new} are index sets of newly generated coarse training and validation data, and T_c^{next} and V_c^{next} are used for subsequent stage of coarse neural network (h_c) training. The automatic coarse neural model development process including incremental data generation and stage-wise neural network training is continued until $\tilde{y}_c = h_c(x, w_c)$ exceeds user-specified coarse validation error [9][16], i.e., h_c satisfactorily captures the coarse relationship g_c .

5.3.2.2 Phase II: Initialization of Fine Neural Network Model

The key idea behind KAMG algorithm presented here is to systematically establish a framework that can enable simultaneous utilization of g_c and g_f toward efficient neural network learning of fine behaviors [16][17]. After the first phase, a satisfactorily close representation of g_c is available in the form of a coarse neural model $\tilde{y}_c = h_c(x, w_c)$. What now needs to be created is a framework that facilitates reinforced neural network learning from coarse information $\tilde{y}_c$ together with fine data from g_f .

Toward this end, a variety of knowledge-neural-network architectures including DM, KBNN, PKI and SMNN are exploited, and a set of novel formulations for combining coarse and fine information are developed. The objective of the second phase is to construct an initialized fine (overall) neural model, combining $\tilde{y}_c = h_c(x, w_c)$ with a suitable sub neural model, before using any fine data. To accomplish this, a sub neural model $\tilde{y}_s = h_s(x, w_s)$ is generated through a simple training of h_s re-using T_c and V_c

from the first phase. The coarse and sub neural networks are then combined to construct a fine (overall) neural network as

$$\tilde{y}_f = h_f(x, w_f) = h_f(x, w_c, w_s), \quad (5.8)$$

where w_f includes both w_c and w_s .

The manner, in which, the fine (overall) neural network is constructed from h_c and h_s , depends upon the knowledge-neural-network architecture being used. However, since fine data from g_f has not been used so far, $\tilde{y}_f$ in (5.8) is merely an initialized version of the fine (overall) neural network model. This initial $\tilde{y}_f$ closely matches g_c , and the KAMG can improve $\tilde{y}_f$ to accurately match g_f by further training of h_f using fewer fine data from g_f . In this section, fine (overall) neural model initialization through a simple sub-neural-network training is described for various knowledge ANN architectures.

Case 1: Initialization of KAMG-DM

Coarse data generator g_c can miss finer details in the original x - y relationship to be modeled. Consequently, g_c and g_f differ over the entire x -space of interest, and the difference can vary with x . Here, the focus is on this difference in an attempt to use coarse and fine information simultaneously toward fine learning. The difference method based neural network architecture [6] is exploited. Coarse neural network model $\tilde{y}_c = h_c(x, w_c)$ closely representing g_c has already been generated in the first phase. Considering this, the sub neural network h_s is expected to learn the difference between coarse and fine data.

Neural network h_s is initialized through a training process whose objective is to

$$\min_{w_s} \sum_{p \in T_c} \|h_s(x_p, w_s) - 0\| \quad (5.9)$$

by adjusting w_s . In other words, h_s is forced to learn a 0 difference, which is a reasonable initialization for the difference, in the absence of any fine data. An initial fine (overall) neural network model is then constructed utilizing the coarse and the sub neural models as

$$\tilde{y}_f = h_f(x, w_c, w_s) = h_c(x, w_c) + h_s(x, w_s), \quad (5.10)$$

where outputs of h_f are computed through a vector summation of outputs from h_c and h_s . In this way, the initial fine (overall) network h_f would be as good as h_c (i.e., g_c) in representing the given x - y relationship, even before any expensive fine data is used.

Case 2: Initialization of KAMG-KBNN

In general, coarse data generator g_c (e.g., empirical equations or models) can at best represent the fine RF/microwave behaviors g_f accurately in a sub-region of the x -space of interest. In order to develop a fine (overall) neural network model that can accurately represent g_f over the entire x -space of interest, use of multiple coarse neural network models is proposed. The KBNN architecture [3] is exploited, and coarse neural model h_c generated in the first phase is incorporated into several knowledge layer neurons. The idea here is that the KBNN's weight parameters associated with several instances of h_c , can be tweaked to build an accurate fine (overall) neural model.

Neural network h_s is initialized through a training process whose objective is to

$$\min_{w_s} \sum_{p \in T_c} \|h_s(x_p, h_{c1}(x_p, w_c), h_{c2}(x_p, w_c), \dots, h_{cN}(x_p, w_c), w_s) - \tilde{y}_{cp}\|, \quad (5.11)$$

by adjusting w_s and keeping w_c fixed. Here, N is the number of knowledge layer neurons (i.e., number of times coarse neural model is repeated in the KBNN sub network), and $\tilde{y}_{cp} = h_c(x_p, w_c)$ is the coarse neural model output corresponding to an input x_p . The sub network h_s is used together with a region network h_r and a gating operator $\otimes$ [3] to construct the initial version of fine (overall) neural network model as

$$\tilde{y}_f = h_f(x, w_c, w_s) = h_s(x, h_{c1}(x, w_c), \dots, h_{cN}(x, w_c), w_s) \otimes h_r(x, w_r). \quad (5.12)$$

Weight parameters w_r in h_r associated with each of the h_c 's used in the fine (overall) network are initialized with equal values to ensure equal contribution of each h_c toward fine outputs $\tilde{y}_f$ throughout the x -space. The initial fine (overall) neural model $\tilde{y}_f$ is as good as coarse neural model $\tilde{y}_c$ in representing the given x - y relationship, prior to using any fine data. It may be noted that conceptually region network weights w_r are part of the sub neural network weights w_s .

Case 3: Initialization of KAMG-PKI

In this case, the KAMG focuses on the relationship between coarse data generator outputs d_c and fine data generator outputs d_f . The d_c versus d_f relationship can be influenced by the input vector x as well. The PKI architecture [37] is exploited and the purpose is to

train a sub neural network h_s to learn the relationship $(\mathbf{x}, \mathbf{d}_c)$ versus $\mathbf{d}_f$. Since the fine data generator g_f has not been used so far, h_s is made to learn the relationship $(\mathbf{x}, \mathbf{d}_c)$ versus $\mathbf{d}_c$, which is a reasonable initialization. Data $\mathbf{d}_c$ is available in the form of coarse neural model $\tilde{\mathbf{y}}_c = \mathbf{h}_c(\mathbf{x}, \mathbf{w}_c)$ generated in the first phase. Neural network h_s is initialized through a training process whose objective is to

$$\min_{\mathbf{w}_s} \sum_{p \in T_c} \left\| \mathbf{h}_s(\mathbf{x}_p, \tilde{\mathbf{y}}_{cp}, \mathbf{w}_s) - \tilde{\mathbf{y}}_{cp} \right\|, \quad (5.13)$$

by adjusting $\mathbf{w}_s$.

Following this, an initial version of fine (overall) neural network is constructed. Coarse neural network h_c trained in the first phase is used at the lower level of the hierarchical PKI structure. Inputs $\mathbf{x}$ together with outputs $\tilde{\mathbf{y}}_c$ from h_c are then supplied to the sub network h_s at the higher level of the hierarchy. The initial fine (overall) neural network model given by

$$\tilde{\mathbf{y}}_f = \mathbf{h}_f(\mathbf{x}, \mathbf{w}_c, \mathbf{w}_s) = \mathbf{h}_s(\mathbf{x}, \mathbf{h}_c(\mathbf{x}, \mathbf{w}_c), \mathbf{w}_s) \quad (5.14)$$

is as good as h_c (i.e., g_c) in representing the $\mathbf{x}$ - $\mathbf{y}$ relationship before using any fine data.

Case 4: Initialization of KAMG-SMNN

To better formulate the KAMG-SMNN problem, $\mathbf{x}_c$ and $\mathbf{x}_f$ are defined as coarse and fine input vectors respectively (where $\mathbf{x}_f$ is the original input $\mathbf{x}$ itself). When a fine input $\mathbf{x}_f = \mathbf{x}$ is presented to coarse data generator g_c , an approximate or coarse output is

computed. If a coarse input $\mathbf{x}_c$ corresponding to the fine input $\mathbf{x}_f$ can be found, such that $\mathbf{g}_f(\mathbf{x}_f) \equiv \mathbf{g}_c(\mathbf{x}_c)$, then fast $\mathbf{g}_c$ can be used in place of CPU-intensive $\mathbf{g}_f$. The challenge here is to find a space mapping [173][174] between $\mathbf{x}_f$ and $\mathbf{x}_c$. The SMNN architecture [35] is exploited and the sub neural network $\mathbf{h}_s$ is made to learn the mapping between above-mentioned input-spaces. In the absence of any fine data, a logical initialization for the space mapping can be $\mathbf{x}_c = \mathbf{x}_f$. As such, neural network $\mathbf{h}_s$ is initialized through a training process whose objective is to

$$\min_{\mathbf{w}_s} \sum_{p \in T_c} \left\| \mathbf{h}_s(\mathbf{x}_p, \mathbf{w}_s) - \mathbf{x}_p \right\|, \quad (5.15)$$

by adjusting $\mathbf{w}_s$.

Utilizing $\mathbf{h}_c$ and $\mathbf{h}_s$, an initial version of the fine (overall) neural network can now be constructed. The original or fine input $\mathbf{x}_f$ is supplied to $\mathbf{h}_s$ and corresponding $\mathbf{x}_c$ is computed. The coarse neural network model $\mathbf{h}_c$ developed in the first phase is then used to compute the output vector corresponding to $\mathbf{x}_c$. The initial fine (overall) neural network model is given by

$$\tilde{\mathbf{y}}_f = \mathbf{h}_f(\mathbf{x}_f, \mathbf{w}_c, \mathbf{w}_s) = \mathbf{h}_c(\mathbf{h}_s(\mathbf{x}_f, \mathbf{w}_s), \mathbf{w}_c). \quad (5.16)$$

Training of $\mathbf{h}_s$ in (5.15) ensures that the initial fine (overall) neural model $\tilde{\mathbf{y}}_f$ in equation (5.16) can be meaningful even without using any fine data, and is as good as coarse model $\tilde{\mathbf{y}}_c$ (i.e., $\mathbf{g}_c$) in representing the given RF/microwave input-output relationship.

Discussion:

Through systematic training and construction of initial fine (overall) neural network in the second phase, the KAMG algorithm ensures that

$$\mathbf{h}_f(\mathbf{x}, \mathbf{w}_c, \mathbf{w}_s) \cong \mathbf{h}_c(\mathbf{x}, \mathbf{w}_c), \quad (5.17)$$

when sub neural network weight vector $\mathbf{w}_s$ is at its current or initialized value. In other words, the initialized $\tilde{\mathbf{y}}_f = \mathbf{h}_f(\mathbf{x}, \mathbf{w}_c, \mathbf{w}_s)$ can be as good as the available coarse data $\mathbf{d}_c = \mathbf{g}_c(\mathbf{x})$ in representing the given $\mathbf{x}$ - $\mathbf{y}$ relationship, prior to using any fine data from $\mathbf{g}_f$.

5.3.2.3 Phase III: Refined Training of Fine Neural Network

In the third phase, the KAMG algorithm emphasizes on capturing the finer RF/microwave behaviors of the given $\mathbf{x}$ - $\mathbf{y}$ relationship missed in the coarse neural network model $\mathbf{h}_c$ (i.e., those missed by $\mathbf{g}_c$). The initial version of the fine (overall) neural network $\mathbf{h}_f$ is further trained using fewest fine data from $\mathbf{g}_f$, to achieve highest possible fine accuracy. This is required in order to improve the neural model performance $\tilde{\mathbf{y}}_f$ currently close to $\mathbf{g}_c$, to an accuracy closest to that of $\mathbf{g}_f$. The training objective here is to minimize the difference between fine (overall) neural network model outputs and fine data outputs, i.e.,

$$\min_{\mathbf{w}_s} \sum_{p \in T_f} \|\mathbf{h}_f(\mathbf{x}_p, \mathbf{w}_c, \mathbf{w}_s) - \mathbf{d}_{fp}\|, \quad (5.18)$$

by adjusting $\mathbf{w}_s$, keeping $\mathbf{w}_c$ fixed. In (5.18), $\mathbf{d}_{fp}$ represents the fine training data output corresponding to any given input $\mathbf{x}_p$, $p \in T_f$, as defined earlier.

Similar to training in the first phase, the $\mathbf{x}$ -space of interest is considered as a set of sub-regions. For the first stage, fine training data (T_f) and fine validation data (V_f) are systematically generated covering all the sub-regions. The initial fine (overall) neural network h_f is trained with data samples in T_f and validated with those in V_f . For subsequent stages of training, new data sample locations are selected by error-based sampling following (5.5), and T_f and V_f are augmented as in (5.6) and (5.7) by automatically driving g_f using a suitable simulator-driver [10]. The overall framework of KAMG algorithm showing neural model development through simultaneous utilization of coarse and fine data generators is shown in Figure 5.1. Novel KAMG formulations specific to knowledge-neural-network architectures DM, KBNN, PKI and SMNN are illustrated in Figure 5.2, Figure 5.3, Figure 5.4 and Figure 5.5 respectively.

5.3.2.4 Steps in KAMG Algorithm

In this section, various steps involved in KAMG are summarized into a computational algorithm.

Step 1: For a given RF/microwave-modeling problem, select coarse and fine data generators g_c and g_f . For example, 2.5D-EM and 3D-EM simulators can be used as g_c and g_f respectively for neural network modeling of EM components such as the embedded passives (e.g. capacitors and spiral inductors).

Step 2: Select an MLP neural network $h_c(\mathbf{x}, \mathbf{w}_c)$. Perform error-based adaptive sampling and incremental coarse data generation together with stage-wise training [10][16], with a training objective $\min_{\mathbf{w}_c} \sum_{p \in T_c} \|h_c(\mathbf{x}_p, \mathbf{w}_c) - \mathbf{d}_{cp}\|$ to automatically generate a coarse neural network model $\tilde{\mathbf{y}}_c = h_c(\mathbf{x}, \mathbf{w}_c)$. Neural model development process is terminated when neural model responses from h_c are sufficiently close to coarse data outputs from $\mathbf{g}_c$, i.e., $\tilde{\mathbf{y}}_c = h_c(\mathbf{x}, \mathbf{w}_c)$ and $\mathbf{d}_c = \mathbf{g}_c(\mathbf{x})$ closely match in the $\mathbf{x}$ -space of interest.

Step 3: Prepare the data index sets T_c and V_c for re-use in training and validation of sub neural network model. Data format for sub neural network training is different for different knowledge-neural-network architectures. Coarse training and validation data are reformatted into ordered pairs $(\mathbf{x}_p, \mathbf{0})$ for KAMG-DM architecture (ignoring outputs $\mathbf{d}_{cp}$), into $(\mathbf{x}_p, \mathbf{d}_{cp})$ for KAMG-KBNN architecture, into $((\mathbf{x}_p, \mathbf{d}_{cp}), \mathbf{d}_{cp})$ for KAMG-PKI architecture, and into ordered pairs $(\mathbf{x}_p, \mathbf{x}_p)$ for KAMG-SMNN architecture.

Step 4: A sub neural network $h_s(\mathbf{x}, \mathbf{w}_s)$ is trained and validated using the corresponding re-formatted data sets from step 3. Training objective for sub neural network training is different for different knowledge-neural-network architectures as already defined in equations (5.9), (5.11), (5.13) and (5.15).

Step 5: Utilizing the coarse and the sub neural network models $h_c(\mathbf{x}, \mathbf{w}_c)$ and $h_s(\mathbf{x}, \mathbf{w}_s)$ from step 2 and step 4 respectively, an initial version of the fine (overall)

neural model $h_f(\mathbf{x}, \mathbf{w}_c, \mathbf{w}_s)$ is constructed. The formulation ensures that the neural model responses $\tilde{\mathbf{y}}_f$ from h_f are as good as $\tilde{\mathbf{y}}_c$ from h_c in the entire $\mathbf{x}$ -space of interest, prior to using any fine data.

REPEAT steps 3 through 5 for various knowledge neural network formulations including KAMG-DM, KAMG-KBNN, KAMG-PKI and KAMG-SMNN.

Step 6: Starting with the available version of the fine (overall) neural network h_f , perform error-based adaptive sampling, incremental fine data generation, and neural network training for a single stage. The training objective is $\min_{\mathbf{w}_s} \sum_{p \in T_f} \|h_f(\mathbf{x}_p, \mathbf{w}_c, \mathbf{w}_s) - \mathbf{d}_{fp}\|$, and fine training and validation data are generated by automatically driving g_f . At the end of this step, a refined version of the fine (overall) neural network model is available corresponding to each of the four knowledge architectures formulated within the proposed KAMG framework.

Step 7: Compute the validation error of the available fine (overall) neural model corresponding to each of the four KAMG-based knowledge architectures.

Step 8: If validation error of one or more of the fine models meets user-specification, select the fine (overall) neural network model with least validation error and exit the algorithm. Provide this best neural network model to the RF/microwave designer for further use in circuit or system CAD. Otherwise, **GOTO** step 6 for an additional stage of refined training using fine data.

Figure 5.1. Overall framework of KAMG algorithm showing various phases of neural network model development simultaneously utilizing coarse and fine data generators.

Figure 5.2. Neural network model development using the KAMG formulation for the difference method based neural network architecture resulting in the KAMG-DM. The portion within the dotted boundary represents the fine (overall) neural model for use in RF and microwave CAD.

Figure 5.3. Neural network model development using the KAMG formulation for the knowledge based neural network architecture resulting in the KAMG-KBNN. Part A of the sub network represents region and boundary layers, and Part B represents the gating network. The portion within the dotted boundary represents the fine (overall) neural model for use in RF and microwave CAD.

Figure 5.4. Neural network model development using the KAMG formulation for the prior knowledge input neural network architecture resulting in the KAMG-PKI. The portion within the dotted boundary represents the fine (overall) neural model for use in RF and microwave CAD.

Figure 5.5. Neural network model development using the KAMG formulation for the space mapped neural network resulting in KAMG-SMNN. The portion within the dotted boundary represents the fine (overall) neural model for use in RF and microwave CAD.

Here, various cases of the proposed KAMG algorithm are discussed. In a case where coarse data equals fine data, the KAMG terminates with phase I alone. In cases where coarse data differs from fine data, model refinement (i.e., phase III) using fine data is essential. There can be two possibilities: (a) If the difference is slender, very few fine data are used and phase III ends quickly, (b) Otherwise, phase III of the KAMG takes relatively longer time and the number of fine data used depends upon the difference.

5.4 Demonstration Examples

5.4.1 MOSFET

This example illustrates development of MOSFET neural models with physics-level accuracy, but without using too much physics-based (expensive) fine training data. Neural network input $\mathbf{x}$ contains five parameters namely, channel length (L), channel width (W), oxide thickness (T_{ox}), drain voltage (V_d) and gate voltage (V_g). Drain current (I_d) is the parameter in neural model output y . A fast and approximate semi-analytical equivalent circuit model in *HSPICE* [165] is used as coarse data generator g_c , while a CPU-intensive (slow) but accurate physics-based *MINIMOS* simulator [175] is used as fine data generator g_f .

KAMG algorithm is used for automatic generation of MOSFET neural models. In the first phase, a coarse neural model $\tilde{y}_c$ is generated by extensive use of coarse training data from g_c (3265 coarse samples). Neural model $\tilde{y}_c$ predicts drain current with an average validation error of 4.50%. In the second phase, a sub neural model $\tilde{y}_s$ is developed and an initial fine (overall) neural network is constructed, corresponding to each of the four knowledge neural network formulations namely, KAMG-DM, KAMG-KBNN, KAMG-PKI and KAMG-SMNN. In each case, it is ensured that the initial fine (overall) neural model is as good as $\tilde{y}_c$ in representing MOSFET's $\mathbf{x}$ - y characteristics, i.e., average validation error of each of the initial (fine) neural models is 4.50%, prior to using any fine data.

These initial fine (overall) neural networks are further trained using fewer fine training data from g_f through automatic sampling and stage-wise training as described in steps 6 through 8 of the KAMG. As can be seen in Figure 5.6, all the KAMG neural models achieved better accuracies with fewer fine data as compared to MLP neural models from the AMG algorithm.

Figure 5.6. Accuracy comparison of MOSFET neural models generated by KAMG and earlier AMG algorithms. All the KAMG neural models achieved better accuracies with fewer fine data as compared to MLP neural models from AMG.

For a user-specified neural model accuracy of 0.50%, the KAMG algorithm requires fewer fine data than conventional training (manual neural modeling approach) and

AMG as shown in Table 5.1. In this example, the KAMG algorithm terminated with the KAMG-SMNN neural model achieving the specified 0.50% accuracy with only 454 fine data. Table 5.2 shows that the KAMG models outperform corresponding conventional knowledge models, when only a few fine data are used.

Table 5.1. Comparison of fine data needed by various neural modeling techniques to achieve MOSFET models with 0.50% validation error.

Neural Modeling Technique	Number of Fine Data Used
Conventional training	1495
AMG	1190
KAMG-DM	724
KAMG-KBNN	562
KAMG-PKI	645
KAMG-SMNN	454

5.4.2 Embedded Resistor

Fast and accurate models representing 3D-EM behaviors of embedded passives are important for multi-layer PCB design. In this example, neural models of an embedded resistor are developed. Input vector $\mathbf{x}$ contains resistor length (L) and signal frequency (f), and output vector $\mathbf{y}$ contains real and imaginary parts of two-port S-parameters RS_{11} , IS_{11} , RS_{21} , and IS_{21} . A fast 2.5D-EM simulator [110] is used as coarse data generator $\mathbf{g}_c$, and a CPU-intensive 3D-EM simulator [13] including finer EM effects is used as fine

data generator g_f . Embedded resistor neural models are developed using KAMG algorithm, and other existing techniques such as the conventional training technique.

Table 5.2. Accuracy comparison between MOSFET neural models developed by various modeling techniques using 454 fine data.

Neural Modeling Technique	Validation Error
Conventional training	3.24%
AMG	2.52%
Conventional DM	1.48%
KAMG-DM	0.95%
Conventional KBNN	0.98%
KAMG-KBNN	0.65%
Conventional PKI	0.92%
KAMG-PKI	0.60%
Conventional SMNN	0.84%
KAMG-SMNN	0.48%

The first phase of the KAMG algorithm using extensive coarse training data (220 samples) resulted in a coarse neural model with 7.02% average validation error. Training and initialization in the second phase re-using the coarse data yielded initial fine (overall) embedded resistor neural models with accuracies close to that of the coarse neural model.

These initial fine neural models are further trained using 3D-EM fine data in the third phase. Figure 5.7 and Table 5.3 show that KAMG algorithm yields relatively accurate neural models as compared to the other techniques, when fewer fine data are available. In this particular example, the KAMG algorithm exited with the KAMG-PKI neural model exceeding user-specified accuracy of 0.50% with only 18 fine data.

Figure 5.7. Accuracy comparison of embedded resistor neural models generated by KAMG and AMG algorithms. When only a few fine data are available, all the KAMG neural models achieved better accuracies than the MLP neural models from AMG.

Table 5.3. Accuracy comparison between resistor neural models developed by various modeling techniques using 18 fine data.

Neural Modeling Technique	Validation Error
Conventional training	2.85%
AMG	1.65%
KAMG-DM	1.02%
KAMG-KBNN	0.65%
KAMG-PKI	0.42%
KAMG-SMNN	0.95%

5.4.3 Embedded Capacitor

In this example, neural network models of an embedded capacitor shown in Figure 5.8 are developed. The input x includes capacitor length (L) and signal frequency (f). Real and imaginary parts of two-port S-parameters RS_{11} , IS_{11} , RS_{21} , and IS_{21} are the model outputs y . A 2.5D-EM simulator [110] is used as coarse data generator and a 3D-EM simulator [13] is used as fine data generator. Attempts to develop capacitor neural models with 3D-EM accuracy using fine data from 3D-EM simulator alone have proved to be computationally prohibitive, as each 3D-EM simulation needs a lot of CPU-time. In addition, lot of human effort was also required. This led to a need for examining possibilities of using inexpensive coarse data together with fine data for efficient neural model development.

Figure 5.8. 3D-EM representation of the embedded capacitor to be modeled using KAMG algorithm.

The first and the second phases of training using 180 coarse data resulted in initial fine (overall) neural models with an average validation error of 3.02%. These initial fine neural networks are further refined in the third phase using 3D-EM fine data. In order to generate a neural model with a given accuracy, it can be seen from Figure 5.9 that the proposed KAMG algorithm requires fewer fine data as compared to the AMG algorithm. Consequently, CPU-time for data generation is significantly reduced as can be seen in Table 5.4. Since data generation time is the major constituent of the total neural model development time, it can be concluded that neural model development using the proposed KAMG algorithm is faster than the existing neural modeling approaches. For a user-specified model accuracy of 1%, the KAMG algorithm stopped with the KAMG-KBNN model exceeding the specification with only 14 fine data.

Figure 5.9. Accuracy comparison of capacitor neural models generated by KAMG and AMG algorithms. All the KAMG neural models achieved better accuracies with fewer fine data as compared to MLP neural models from AMG.

Table 5.4. Comparison of fine data and CPU-time needed by various neural modeling techniques to achieve capacitor models with 1% validation error.

Neural Modeling Technique	Number of Fine Data	CPU-Time for Data Generation (in min)
Conventional training	125	625
AMG	96	480
KAMG-DM	48	240
KAMG-KBNN	14	70
KAMG-PKI	24	120
KAMG-SMNN	23	115

5.4.4 Spiral Inductor

Passives such as spiral inductors [172] on a chip are critical for a successful integrated RF/microwave system design. Although 3D-EM simulators can be used to analyze spiral inductors, they are computationally prohibitive especially if the inductor's geometrical and physical parameters need to be repetitively changed during CAD. For the spiral inductor shown in Figure 5.10, automatic generation of neural network models with EM-level accuracy, but without using too much 3D-EM based fine training data, is illustrated. A bridge drawn 0.1mil above the spiral pattern is used as the inductor's connector pass. A 2.5D-EM simulator [110] is used as a coarse data generator and a 3D-EM simulator [13] is used as a fine data generator.

Figure 5.10. 3D-EM representation of the spiral inductor to be modeled using KAMG algorithm.

The KAMG algorithm is used to automatically generate 2-loop inductor neural models. Input x contains width (W), space (S), dielectric constant (ϵ_r), and frequency (f). Real and imaginary parts of S-parameters are the model outputs y . Extensive coarse data (540 samples) is used in the first and the second phases of the KAMG algorithm. At the end of third phase using fine data, all the 2-loop inductor neural models from KAMG achieved better accuracies as compared to MLP neural models from AMG, as can be seen in Figure 5.11. For a given model accuracy, the KAMG algorithm requires fewest fine data than all other existing neural modeling approaches as shown in Table 5.5.

Figure 5.11. Accuracy comparison of 2-loop spiral inductor neural models generated by KAMG and AMG algorithms. All the KAMG neural models achieved better accuracies with fewer fine data as compared to MLP neural models from AMG.

Table 5.5. Accuracy comparison between 2-loop inductor neural models developed by various modeling techniques using 386 fine data.

Neural Modeling Technique	Validation Error
Conventional training	6.25%
AMG	2.34%
KAMG-DM	0.96%
KAMG-KBNN	1.12%
KAMG-PKI	0.85%
KAMG-SMNN	0.99%

Neural models of an 8-loop inductor are also generated using KAMG algorithm. Input x includes space (S) and frequency (f), and output y includes real and imaginary parts of S-parameters. The algorithm uses 184 coarse data for initial training and only a few fine data for model refinement. The KAMG algorithm terminated with the KAMG-PKI model achieving the user-specified 1% accuracy with 23 fine data. Figure 5.12 demonstrates that KAMG yields relatively accurate models using fewer fine data compared to AMG. Table 5.6 shows that the proposed KAMG significantly reduces CPU-time for data generation. Consequently, the overall neural model generation time is considerably reduced as well.

Figure 5.12. Accuracy comparison of 8-loop spiral inductor neural models generated by KAMG and AMG algorithms. All the KAMG neural models achieved better accuracies with fewer fine data as compared to MLP neural models from AMG.

Table 5.6. Comparison of fine data and CPU-time needed by various neural modeling techniques to achieve 8-loop inductor models with 1% validation error.

Neural Modeling Technique	Number of Fine Data	CPU-Time for Data Generation (in min)
Conventional training	98	1764
AMG	52	936
Proposed KAMG-DM	27	486
Proposed KAMG-KBNN	32	576
Proposed KAMG-PKI	23	414
Proposed KAMG-SMNN	27	486

5.5 Summary

A robust KAMG algorithm has been proposed for automatic generation of RF/microwave neural models of highest accuracies using fewest expensive data. The KAMG integrates advanced concepts such as AMG, knowledge neural networks and space mapping, into an even more powerful framework. Motivated by the space-mapping concept, the KAMG simultaneously exploits coarse and fine data generators for efficient neural model development. The unified approach allows a variety of knowledge neural network architectures to be exploited for accomplishing reinforced neural network training from both coarse and fine data. The KAMG is further strengthened by automation through error-based adaptive sampling. For a given model accuracy, the KAMG algorithm uses fewest fine data compared to all other existing techniques including step-by-step approach, conventional knowledge methodologies, and the AMG algorithm for MLP networks. Fewer fine data translates into significantly reduced CPU-time for data generation resulting in faster model development process.

CHAPTER 6

Conclusions and Future Research

6.1 Conclusions

Rapid progress in the RF and microwave electronics industry over the last decade has led to dramatic increase in circuit complexity and size. Conventional design procedures [1] became increasingly difficult with ever-increasing circuit complexities coupled with tightened design tolerances. Computer-aided high-frequency design has emerged as a strong alternative to cumbersome and expensive conventional design procedure. Modeling is a major bottleneck for efficient computer-aided design and optimization of RF/microwave components and circuits. Recently, neural network based CAD that advocates the use of accurate and fast neural models in place of computationally prohibitive theoretical models has gained recognition [2].

The thesis has presented state-of-the-art research in the neural-network-based RF/microwave CAD area aimed at enhancing this emerging technology to higher levels of maturity. The central objectives of the thesis have been:

- Development of robust neural network modeling techniques that can address practical challenges in RF and microwave modeling.
- Formulation of efficient neural modeling algorithms that can facilitate automatic generation of RF/microwave neural models of highest possible accuracies at lowest cost as compared to any other existing neural modeling techniques.

It is envisioned that meeting such key objectives through focused and methodical research in this area could enable its success in terms of accuracy, cost-effectiveness, speed and viability when employed in practical CAD applications. Specifically, in view of the above-mentioned objectives and vision, the following contributions have been made through this thesis work.

First, neural-network-based modeling of highly nonlinear and multi-dimensional RF/microwave behaviors even in the presence of measurement- or simulation-induced training data errors has been addressed by a new Iterative Multi Stage algorithm [5][8]. In the case of complicated high-frequency modeling problems, where conventional or manual neural modeling approaches balk, the proposed IMS algorithm shines by first (i) Decomposing the complicated original problem into simple sub-tasks, and subsequently (ii) Capturing each of these simpler sub-tasks through a stage-wise training that uses a suitable structure-algorithm pair. Such an approach allows the strengths of various ANN-structures and training algorithms to be simultaneously utilized in conquering a given problem. The framework of the IMS algorithm also includes a novel Huber-norm based HQN training technique that allows the IMS to model only smooth portion of the problem behavior in one of the training stages, ignoring either gross errors or sudden variations in training data. The proposed IMS algorithm has been illustrated through practical RF/microwave examples including those with training data errors (e.g. HBT, MESFET), with sharp variations (e.g. microstrip filter), and with large number of model input-output parameters (e.g. eight-conductor microstripline).

Second, automation of RF/microwave neural model development process has been accomplished for the first time in the EM-ANN literature, through a novel Automatic Model Generation algorithm [9][10]. Starting with zero amounts of training and validation data and then proceeding with neural network training in a stage-wise manner, the AMG algorithm can automatically produce a neural model that exceeds user-specified accuracy criterion. Each stage can involve some or all of the sub-tasks like error-based adaptive sampling and incremental data generation, neural network size adjustment, neural network training and neural model validation. To facilitate dynamic data generation by the AMG, a simulator driver framework has been established and several innovative in-house simulator drivers have been implemented (e.g. *ADS-Driver*, *HFSS-Driver*, *MINIMOS-Driver*). The framework of the proposed AMG algorithm allows judicious use of training data, i.e., more samples are used in nonlinear sub-regions of the model input space and relatively fewer samples are used in smooth sub-regions. Through selected demonstration examples, the AMG algorithm has been shown to outperform manual step-by-step neural modeling approach in terms of both cost and speed of neural model development process.

Third, an efficient Knowledge Based Automatic Model Generation algorithm [16][17] aimed at generating RF/microwave neural models of highest possible accuracies using fewest accurate (expensive) training data as compared to all other existing EM-ANN approaches has been proposed. The technique is comprehensively derived to integrate three distinct powerful concepts, namely, automatic model generation, knowledge neural networks and space mapping. For the first time, KAMG utilizes multiple sources

of data including, approximate and fast coarse data generators, and accurate and slow fine data generators in dealing with a given RF/microwave modeling problem. In order to facilitate reinforced neural network learning from both coarse and fine data, a variety of most up-to-date ANN structures with knowledge including DM, KBNN, PKI and SMNN have been incorporated into the KAMG formulation. Striking advantages of the KAMG algorithm have been illustrated through two categories of examples namely, active devices and embedded passives used in multi-layer PCB's. The KAMG algorithm represents cutting-edge research innovation in the area that allows cheaper and faster generation of neural models, and the neural model development process is automated thereby minimizing the need for human-effort and time. It is to be noted that when coarse data generator represents basic problem behavior of corresponding fine data generator, the KAMG is useful. In an extreme case where there is a contradiction between coarse and fine data generators, use of AMG with fine data alone is suggested.

In order to accelerate the practical use of the algorithms proposed in this thesis, a generalized implementation is necessary. An object-oriented computer program embedding AMG and KAMG algorithms has been developed in C++, incorporating various aspects such as adaptive controller sub-routine, dynamic data generation through simulator drivers, automatic neural network training, neural network size adjustment etc. The program has been used in deriving the results in this thesis and incorporated into the trial version of *NeuroModeler* software [18]. The work presented in the thesis has made neural network based CAD approach much more accessible. In other words, using the results of the thesis in the form of efficient algorithms,

RF/microwave designers who are non-experts in ANN issues can automatically build neural models with similar accuracies as those attained by ANN experts.

6.2 Future Directions

Artificial neural networks represent one of the most recent trends in RF and microwave computer aided design, and neural-network-based CAD approaches including those presented in this thesis aim to achieve highest levels of speed, precision and automation, in an attempt to meet the challenges posed by the next generation of high-frequency design. This area combines artificial intelligence concepts with state-of-the-art CAD technologies creating numerous opportunities for technical discovery and industrial applications at various stages of high frequency CAD including modeling, simulation and design.

Neural networks certainly possess some unique qualities, which make them an attractive alternative to conventional methods. For instance, neural network approaches are “*generic*” in the sense that they can be applied to active (semi-conductor) or passive (EM) domains, component- or circuit-level CAD applications, and can aid designers in breaking the barriers faced in information-technology-driven concurrent design and optimization tasks that are constrained by multi-disciplinary factors, e.g. electro-thermal packaging, which involves electrical, mechanical and thermal effects. Future work is anticipated in (but not limited to) the following thrust areas of RF, microwave and electronics engineering in general.

Combining the learning and the generalization capabilities of artificial neural networks with existing RF and microwave engineering knowledge continues to be a strategic area of EM-

ANN research. The objective is to utilize available knowledge maximally to develop hybrid circuit-ANN model architectures that can attain highest levels of model accuracies especially when limited training data is available. Investigation for different methods of incorporating knowledge into the neural networks, e.g. empirical equations and equivalent circuits, frequency- or time-domain approximations etc., can lead to even more advanced knowledge ANN structures. Higher forms of engineering knowledge being considered in today's research include circuit simulators and state-space formulations for embedded passive component modeling and design including high-frequency effects, passive reduced-order models for optimization of high-speed interconnects with 3D-EM accuracies etc. Such research activity can be highly significant, e.g. the resulting advanced knowledge based ANN architectures can further broaden the framework of KAMG algorithm.

Modeling large-signal behaviors of transistors and dynamic characteristics of nonlinear circuits is another strategic topic for research, because of several reasons. Every time a new semiconductor technology evolves, creation of fresh transistor models becomes mandatory in order to be able to use the new generation of devices in VLSI circuit design. Often since technologies evolve rapidly and detailed models are not readily available, ability of artificial neural networks to model even in the absence of any equations makes them the unique choice. Nonlinear circuit modeling, on the other hand, is a key to system-level CAD, e.g. simulation of a transmitter- receiver module. From the neural network structural point of view, nonlinear device and circuit behaviors are best described in time-domain, and consequently dynamic or recurrent neural networks are best suited. This places a high demand for developing advanced ANN structures (e.g. dynamic neural networks with adjoint

capabilities) and training algorithms (e.g. back-propagation-through-time). The AMG and the IMS algorithms presented in the thesis can further compliment this research subject by allowing automated and iterative approaches to nonlinear device and circuit modeling.

“Automation” is a popular 21st century buzzword in every discipline of engineering and information technology including RF/microwave CAD. The core objective of an automation technique is to convert the conventional human-tasks into computer-based tasks. As a result, routine workload (e.g., trial-and-error) can be translated from humans to CPU. The portion of the time saved can be expended by humans in creative and innovative rather than routine tasks, thereby resulting in a performance- and productivity-enhanced design environment. Automation allows massive and repetitive design tasks to be performed more efficiently, facilitates 24-hour nonstop work cycles even in the absence of humans, and most importantly removes the element of risk or possibility of human-induced errors. In essence, automated approaches can help to speedup CAD activities through faster design cycles enabling RF/microwave designers to meet greater demands of today’s electronics industry, e.g. shortened time-to-market. In today’s hi-tech world, there is a constant need for developing algorithms that can lead to higher and even higher levels of automation. In the case of ANN-based CAD, the concept of automation of neural model development process, introduced in the thesis has succeeded in opening the doors in this direction for the first time and is expected to remain a thrust area for future research. The developed AMG and the KAMG algorithms have addressed the aspect of modeling automation. The next step is to investigate methodologies for automation of ANN-based RF/microwave CAD at higher-levels including design and optimization. One way to expedite this research is to examine research activities

in theoretical-ANN area and quickly translate interesting results that can enhance automation into an RF/microwave context. In this thesis, efficient generation of neural models by automatic driving of simulators has been demonstrated. Another interesting aspect is to facilitate neural model generation through automatic driving of computer-controlled measurement equipment, which can be of greater practical relevance in future.

Last but not the least, it is important that we continue to incorporate advanced RF/microwave-oriented neural network modeling algorithms and techniques as they evolve into readily usable ANN-software tools. These tools enable RF/microwave designers to quickly build neural models for their high-frequency components. Most of the commercially available RF/microwave network simulators do have provision for linking externally developed component models including neural models. A key requirement is to develop plug-in software tools that can allow effortless insertion of neural models into commercial simulator environments for carrying out circuit- and system-level CAD. These activities have tremendous potential for propagating neural based design and optimization into newer arenas of electronics CAD including CPW circuit design, integrated circuit-antenna module design, and incorporation of high-frequency considerations in high-speed digital circuits.

In conclusion, artificial neural networks with their unparalleled qualities of accuracy, flexibility and speed, continue to be one of the most attractive and powerful vehicles at the forefront of RF and microwave computer aided design today and tomorrow. The concept of automation, introduced for the first time in this vibrant area, will take the lead in bringing the dream of computerized end-to-end neural-based CAD methodologies to reality.

Bibliography

- [1] Q.J. Zhang and K.C. Gupta, *Neural Networks for RF and Microwave Design*, Norwood, MA: Artech House, 2000.
- [2] K.C. Gupta, "Emerging trends in millimeter-wave CAD," *IEEE Trans. Microwave Theory Tech.*, vol. 46, pp. 747-755, 1998.
- [3] F. Wang and Q.J. Zhang, "Knowledge-based neural models for microwave design," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 2333-2343, 1997.
- [4] F. Wang, V.K. Devabhaktuni, and Q.J. Zhang, "A hierarchical neural network approach to the development of a library of neural models for microwave design," *IEEE Trans. Microwave Theory Tech.*, vol. 46, pp. 2391-2403, 1998.
- [5] V.K. Devabhaktuni, C. Xi, F. Wang, and Q.J. Zhang, "Robust training of microwave neural models," *IEEE MTT-S Int. Microwave Symp. Dig.*, Anaheim, CA, 1999, pp. 145-148.
- [6] P.M. Watson and K.C. Gupta, "EM-ANN models for microstrip vias and interconnects in dataset circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 44, pp. 2495-2503, 1996.
- [7] G.L. Creech, B.J. Paul, C.D. Lesniak, T.J. Jenkins, and M.C. Calcaterra, "Artificial neural networks for fast and accurate EM-CAD of microwave circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 794-802, 1997.
- [8] V.K. Devabhaktuni, C. Xi, F. Wang, and Q.J. Zhang, "Robust training of microwave neural models," *Int. J. RF Microwave CAE*, vol. 12, pp. 109-124, 2002.

- [9] V.K. Devabhaktuni, M.C.E. Yagoub, and Q.J. Zhang, "A robust algorithm for automatic development of neural network models for microwave applications," *IEEE MTT-S Int. Microwave Symp. Dig.*, Phoenix, AZ, 2001, pp. 2087-2090.
- [10] V.K. Devabhaktuni, M.C.E. Yagoub, and Q.J. Zhang, "A robust algorithm for automatic development of neural network models for microwave applications," *IEEE Trans. Microwave Theory Tech.*, vol. 49, pp. 2282-2291, 2001.
- [11] V.K. Devabhaktuni and Q.J. Zhang, "Neural network training-driven adaptive sampling algorithm for microwave modeling," *Proc. European Microwave Conf.*, Paris, France, 2000, pp. 222-225.
- [12] *OSA90 v.3.0*, Optimization Systems Associates, P.O. Box 8083, Dundas L9H 5E7, Ontario, Canada, now Agilent EEsof, 1400 Fountaingrove Parkway, Santa Rosa, CA 95403.
- [13] *HFSS v.8.0*, Ansoft Corporation, Four Station Square, Suite 200, Pittsburgh, PA 15219-1119.
- [14] *ADS v.1.5*, Agilent Technologies, 395 Page Mill Road, P.O. Box 10395, Palo Alto, CA 94303.
- [15] C. Xi, F. Wang, V.K. Devabhaktuni, and Q.J. Zhang, "Huber optimization of neural networks: A robust training method," *Int. Joint Conf. Neural Networks*, Washington, DC, 1999, pp. 1639-1642.
- [16] V.K. Devabhaktuni, B. Chattaraj, M.C.E. Yagoub, and Q.J. Zhang, "Advanced microwave modeling framework exploiting automatic model generation, knowledge neural networks, and space mapping," *IEEE MTT-S Int. Microwave Symp. Dig.*, Seattle, WA, 2002, pp. 1097-1100.

- [17] V.K. Devabhaktuni, B. Chattaraj, M.C.E. Yagoub, and Q. J. Zhang, "Advanced microwave modeling framework exploiting automatic model generation, knowledge neural networks, and space mapping," (Accepted) *IEEE Trans. Microwave Theory Tech.*
- [18] *NeuroModeler* v.1.2, Dr. Q.J. Zhang, Department of Electronics, Carleton University, 1125 Colonel By Drive, Ottawa K1S 5B6, Ontario, Canada.
- [19] G. Cheron, J.P. Draye, M. Bourgeois, and G. Libert, "A dynamic neural network identification of electromyography and arm trajectory relationship during complex movements," *IEEE Trans. Biomed. Eng.*, vol. 43, pp. 552-558, 1996.
- [20] J.R. Noriega and H. Wang, "A direct adaptive neural-network control for unknown nonlinear systems and its application," *IEEE Trans. Neural Networks*, vol. 9, pp. 27-34, 1998.
- [21] B. Hussain and M.R. Kabuka, "A novel feature recognition neural network and its application to character recognition," *IEEE Trans. Pattern Anal. Machine Intelligence*, vol. 16, pp. 98-106, 1994.
- [22] A. Waibel, T. Hanazawa, G Hinton, K. Shikano, and K.J Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoustics Speech Signal Processing*, vol. 37, pp. 328-339, 1989.
- [23] A. Patnaik, R.K. Mishra, G.K. Patra, and S.K. Dash, "An artificial neural network model for effective dielectric constant of microstripline," *IEEE Trans. Antennas Propagat.*, vol. 45, p. 1697, 1997.

- [24] G. Antonini and A. Orlandi, "Gradient evaluation for neural-networks-based electromagnetic optimization procedures," *IEEE Trans. Microwave Theory Tech.*, vol. 48, pp. 874-876, 2000.
- [25] Q.J. Zhang, F. Wang, and M.S. Nakhla, "Optimization of high-speed VLSI interconnects: A review," *Int. J. Microwave Millimeter-Wave CAE*, vol. 7, pp. 83-107, 1997.
- [26] T. Horng, C. Wang, and N.G. Alexopoulos, "Microstrip circuit design using neural networks," *IEEE MTT-S Int. Microwave Symp. Dig.*, Atlanta, GA, 1993, pp. 413-416.
- [27] R.L. Mahajan, "Design and optimization through physical/neural network models," *IEEE MTT-S Int. Microwave Symp. Workshop on Applications of ANN to Microwave Design*, Denver, CO, 1997, pp. 1-16.
- [28] Q.J. Zhang, "A neural network paradigm for microwave modeling, simulation and optimization," *IEEE MTT-S Int. Microwave Symp. Workshop on Applications of ANN to Microwave Design*, Denver, CO, 1997, pp. 87-107.
- [29] P.M. Watson, M.Y. Mah, and L.L. Liou, "Input variable space reduction using dimensional analysis for artificial neural network modeling," *IEEE MTT-S Int. Microwave Symp. Dig.*, Anaheim, CA, 1999, pp. 269-272.
- [30] F. Wang and Q.J. Zhang, "Incorporating functional knowledge into neural networks," *Proc. Int. Conf. Neural Networks*, Houston, TX, 1997, pp. 266-269.
- [31] M. Nakhla, "Application of neural networks to simulation and optimization of high-speed interconnects," *IEEE MTT-S Int. Microwave Symp. Workshop on Applications of ANN to Microwave Design*, Denver, CO, 1997, pp. 49-66.

- [32] A.H. Zaabab, Q.J. Zhang, and M.S. Nakhla, "Device and circuit level modeling using neural networks with faster training based on network sparsity," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 1696-1704, 1997.
- [33] C. Cho and K. Gupta, "EM-ANN modeling of overlapping open-ends in multilayer microstrip lines for design of bandpass filters," *IEEE AP-S Int. Symp. Dig.*, Orlando, FL, 1999, pp. 2592-2595.
- [34] J. Bandler, M. Ismail, J. Rayas-Sanchez, and Q. Zhang, "New directions in model development for RF/microwave components utilizing artificial neural networks and space mapping," *IEEE AP-S Int. Symp. Dig.*, Orlando, FL, 1999, pp. 2572-2575.
- [35] J.W. Bandler, M.A. Ismail, J.E. Rayas-Sanchez, and Q.J. Zhang, "Neuromodeling of microwave circuits exploiting space-mapping technology," *IEEE Trans. Microwave Theory Tech.*, vol. 47, pp. 2417-2427, 1999.
- [36] P.M. Watson, K.C. Gupta, and R.L. Mahajan, "Development of knowledge based artificial neural network models for microwave components," *IEEE MTT-S Int. Microwave Symp. Dig.*, Baltimore, MD, 1998, pp. 9-12.
- [37] P.M. Watson, K.C. Gupta, and R.L. Mahajan, "Applications of knowledge-based artificial neural network modeling to microwave components," *Int. J. RF Microwave CAE*, vol. 9, pp. 254-260, 1999.
- [38] K.C. Gupta, "EM-ANN models for microwave and millimeter-wave components," *IEEE MTT-S Int. Microwave Symp. Workshop on Applications of ANN to Microwave Design*, Denver, CO, 1997, pp. 17-47.

- [39] P. Watson and K.C. Gupta, "EM-ANN models for via interconnects in microstrip circuits," *IEEE MTT-S Int. Microwave Symp. Dig.*, San Francisco, CA, 1996, pp. 1819-1822.
- [40] P.M. Watson and K.C. Gupta, "Design and optimization of CPW circuits using EM-ANN models for CPW components," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 2515-2523, 1997.
- [41] P. Watson, G. Creech, and K. Gupta, "Knowledge based EM-ANN models for the design of wide bandwidth CPW patch/slot antennas," *IEEE AP-S Int. Symp. Dig.*, Orlando, FL, 1999, pp. 2588-2591.
- [42] G.L. Creech, B. Paul, C. Lesniak, T. Jenkins, R. Lee, and M. Calcaterra, "Artificial neural networks for accurate microwave CAD applications," *IEEE MTT-S Int. Microwave Symp. Dig.*, San Francisco, CA, 1996, pp. 733-736.
- [43] Y. Harkouss, J. Rousset, H. Chehade, E. Ngoya, D. Barataud, and J.P. Teyssier, "Modeling microwave devices and circuits for telecommunications system design," *Proc. Int. Conf. Neural Networks*, Anchorage, Alaska, 1998, pp. 128-133.
- [44] A.H. Zaabab, Q.J. Zhang, and M. Nakhla, "Analysis and optimization of microwave circuits & devices using neural network models," *IEEE MTT-S Int. Microwave Symp. Dig.*, San Diego, CA, 1994, pp. 393-396.
- [45] G. Kothapalli, "Artificial neural networks as aids in circuit design," *Microelectronics J.*, vol. 26, pp. 569-578, 1995.

- [46] V.B. Litovski, J.I. Radjenovic, Z.M. Mrcarica, and S.L. Milenkovic, "MOS transistor modeling using neural network," *Elect. Lett.*, vol. 28, pp. 1766-1768, 1992.
- [47] G.L. Creech and J.M. Zurada, "Neural network modeling of GaAs IC material and MESFET device characteristics," *Int. J. RF Microwave CAE*, vol. 9, pp. 241-253, 1999.
- [48] J. Rousset, Y. Harkouss, J.M. Collantes, and M. Campovecchio, "An accurate neural network model of FET intermodulation and power analysis," *Proc. European Microwave Conf.*, Prague, Czech Republic, 1996, pp. 16-19.
- [49] A.H. Zaabab, Q.J. Zhang, and M. Nakhla, "A neural network modeling approach to circuit optimization and statistical design," *IEEE Trans. Microwave Theory Tech.*, vol. 43, pp. 1349-1358, 1995.
- [50] J.A. Garcia, A.T. Puente, A.M. Sanchez, I. Santamaria, M. Lazaro, C.J. Pantaleon, and J.C. Pedro, "Modeling MESFETs and HEMTs intermodulation distortion behavior using a generalized radial basis function network," *Int. J. RF Microwave CAE*, vol. 9, pp. 261-276, 1999.
- [51] S. Goasguen, S.M. Hammadi, and S.M. El-Ghazaly, "A global modeling approach using artificial neural network," *IEEE MTT-S Int. Microwave Symp. Dig.*, Anaheim, CA, 1999, pp. 153-156.
- [52] G.L. Creech, "Neural networks for the design and fabrication of integrated circuits," *IEEE MTT-S Int. Microwave Symp. Workshop on Applications of ANN to Microwave Design*, Denver, CO, 1997, pp. 67-86.

- [53] 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. European Microwave Conf.*, Amsterdam, Netherlands, 1998, pp. 306-311.
- [54] M. Vai and S. Prasad, "Qualitatively modeling heterojunction bipolar transistors for optimization: A neural network approach," *Proc. IEEE/Cornell Conf. Adv. Concepts High Speed Semiconductor Dev. Circuits*, 1993, pp. 219-227.
- [55] K. Shirakawa, M. Shimiz, N. Okubo, and Y. Daido, "A large-signal characterization of an HEMT using a multilayered network," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 1630-1633, 1997.
- [56] K. Shirakawa, M. Shimizu, N. Okubo, and Y. Daido, "Structural determination of multilayered large-signal neural-network HEMT model," *IEEE Trans. Microwave Theory Tech.*, vol. 46, pp. 1367-1375, 1998.
- [57] G. Fedi, S. Manetti, G. Pelosi, and S. Selleri, "Design of cylindrical posts in rectangular waveguide by neural network approach," *IEEE AP-S Int. Symp. Dig.*, Salt Lake City, UT, 2000, pp. 1054-1057.
- [58] Q.J. Zhang, G. Wilson, R. Venkatachalam, A. Sarangan, J. Williamson, and F. Wang, "Ultra fast neural models for analysis of electro/optical interconnects," *Proc. IEEE Elec. Component Tech. Conf.*, San Jose, CA, 1997, pp. 1134-1137.
- [59] S. Wang, F. Wang, V.K. Devabhaktuni, and Q.J. Zhang, "A hybrid neural and circuit-based model structure for microwave modeling," *Proc. European Microwave Conf.*, Munich, Germany, 1999, pp. 174-177.

- [60] M.H. Bakr, J.W. Bandler, M.A. Ismail, J.E. Rayas-Sanchez, and Q.J. Zhang, "Neural space mapping EM optimization of microwave structures" *IEEE MTT-S Int. Microwave Symp. Dig.*, Boston, MA, 2000, pp. 879-882.
- [61] P. Burrascano, M. Dionigi, C. Fancelli, and M. Mongiardo, "A neural network model for CAD and optimization of microwave filters," *IEEE MTT-S Int. Microwave Symp. Dig.*, Baltimore, MD, 1998, pp. 13-16.
- [62] G. Fedi, A. Gaggelli, S. Manetti, and G. Pelosi, "Direct-coupled cavity filters design using a hybrid feedforward neural network - finite elements procedure," *Int. J. RF Microwave CAE*, vol. 9, pp. 287-296, 1999.
- [63] S. Bila, Y. Harkouss, M. Ibrahim, J. Rousset, E. Ngoya, D. Billargeat, S. Verdeyme, M. Auborg, and P. Guillon, "An accurate wavelet neural-network-based model for electromagnetic optimization of microwave circuits," *Int. J. RF Microwave CAE*, vol. 9, pp. 297-306, 1999.
- [64] S. Verdeyme, D. Billargeat, S. Bila, S. Moraud, H. Blondeaux, M. Aubourg, and P. Guillon, "Finite element CAD for microwave filters," *European Microwave Conf. Workshop*, Amsterdam, Netherlands, 1998, pp. 12-22.
- [65] P. Burrascano, S. Fiori, and M. Mongiardo, "A review of artificial neural networks applications in microwave computer-aided design," *Int. J. RF Microwave CAE*, vol. 9, pp. 158-174, 1999.
- [66] M. Vai, S. Wu, B. Li, and S. Prasad, "Creating neural network based microwave circuit models for analysis and synthesis," *Proc. Asia Pacific Microwave Conf.*, Hong Kong, 1997, pp. 853-856.

- [67] Y. Harkouss, J. Rousset, H. Chehade, E. Ngoya, D. Barataud, and J.P. Teyssier, "The use of artificial neural networks in nonlinear microwave devices and circuits modeling: An application to telecommunication system design," *Int. J. RF Microwave CAE*, vol. 9, pp. 198-215, 1999.
- [68] J. Xu, M.C.E. Yagoub, and Q.J. Zhang, "Exact adjoint sensitivity analysis for neural based microwave modeling and design," (Accepted) *IEEE Trans. Microwave Theory Tech.*
- [69] Y. 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. Microwave Theory Tech.*, vol. 48, pp. 2335-2344, 2000.
- [70] J. Xu, M.C.E. Yagoub, R. Ding, and Q.J. Zhang, "Neural based dynamic modeling of nonlinear microwave circuits," (Accepted) *IEEE Trans. Microwave Theory Tech.*
- [71] G. Fedi, R. Giomi, M. Giustini, R. Nesti, G. Pelosi, and S. Selleri, "An artificial neural network for the analysis of profiled corrugated circular horns," *IEEE AP-S Int. Symp. Dig.*, Orlando, FL, 1999, pp. 2604-2607.
- [72] C. Christodoulou, A.E. Zooghby, and M. Georgiopoulos, "Neural network processing for adaptive array antennas," *IEEE AP-S Int. Symp. Dig.*, Orlando, FL, 1999, pp. 2584-2587.
- [73] R. Mishra and A. Patnaik, "Neurospectral analysis of coaxial fed rectangular patch antenna," *IEEE AP-S Int. Symp. Dig.*, Salt Lake City, UT, 2000, pp. 1062-1065.

- [74] G. Washington, "Aperture antenna shape prediction by feedforward neural networks," *IEEE Trans. Antennas Propagat.*, vol. 45, pp. 683-688, 1997.
- [75] R.K. Mishra and A. Patnaik, "Neural network based CAD model for the design of square-patch antennas," *IEEE Trans. Antennas Propagat.*, vol. 46, pp. 1890-1891, 1998.
- [76] A.H.E. Zooghby, C.G. Christodoulou, and M. Georgiopoulos, "Neural network-based adaptive beamforming for one- and two-dimensional antenna arrays," *IEEE Trans. Antennas Propagat.*, vol. 46, pp. 1891-1893, 1998.
- [77] E. Charpentier and J.J. Laurin, "An implementation of a direction-finding antenna for mobile communications using a neural network," *IEEE Trans. Antennas Propagat.*, vol. 47, pp. 1152-1159, 1999.
- [78] S. Sagiroglu, K. Guney, and M. Erler, "Calculation of bandwidth for electrically thin and thick rectangular microstrip antennas with the use of multilayered perceptrons," *Int. J. RF Microwave CAE*, vol. 9, pp. 277-286, 1999.
- [79] S.E. Khamy, M.A. Dahab, and K. Hijjah, "Sidelobes reduction and steerable nulling of antenna arrays using neural networks," *IEEE AP-S Int. Symp. Dig.*, Orlando, FL, 1999, pp. 2600-2603.
- [80] G. Castaldi, V. Pierro, and I.M. Pinto, "Neural net aided fault diagnostics of large antenna arrays," *IEEE AP-S Int. Symp. Dig.*, Orlando, FL, 1999, pp. 2608-2611.
- [81] P.R. Chang, W.H. Yang, and K.K. Chan, "A neural network approach to MVDR beamforming problem," *IEEE Trans. Antennas Propagat.*, vol. 40, pp. 313-322, 1992.

- [82] H.L. Southall, J.A. Simmers, and T.H. O'Donnell, "Direction finding in phased arrays with a neural network beamformer," *IEEE Trans. Antennas Propagat.*, vol. 43, pp. 1369-1374, 1995.
- [83] M. Vai and S. Prasad, "Microwave circuit analysis and design by a massively distributed computing network," *IEEE Trans. Microwave Theory Tech.*, vol. 43, pp. 1087-1094, 1995.
- [84] 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, pp. 175-186, 1999.
- [85] K. Watabe, K. Shimizu, K. Mizuno, and M. Yoneyama, "Millimeter-wave imaging using neural networks for object recognition," *IEEE MTT-S Int. Microwave Symp. Dig.*, San Francisco, CA, 1996, pp. 1135-1138.
- [86] K. Yoshitomi, A. Ishimaru, J.N. Hwang, and J.S. Chen, "Surface roughness determination using spectral correlations of scattered intensities and an artificial neural network technique," *IEEE Trans. Antennas Propagat.*, vol. 41, pp. 498-502, 1993.
- [87] H. Yang, C. He, W. Song, and H. Zhu, "Using artificial neural network approach to predict rain attenuation on earth-space path," *IEEE AP-S Int. Symp. Dig.*, Salt Lake City, UT, 2000, pp. 1058-1061.
- [88] J. Lee and A.Lai, "Function-based and physics-based hybrid modular neural network for radio wave propagation modeling," *IEEE AP-S Int. Symp. Dig.*, Salt Lake City, UT, 2000, pp. 446-449.

- [89] M. Vai and S. Prasad, "Automatic impedance matching with a neural network," *IEEE Microwave Guided Wave Lett.*, vol. 3, pp. 353-354, 1993.
- [90] M. Vai and S. Prasad, "Applications of neural networks optimized by the genetic algorithm to microwave systems," *IEEE AP-S Int. Symp. Dig.*, Orlando, FL, 1999, pp. 2580-2583.
- [91] E. Bermani, S. Caorsi, and M. Raffetto, "Time domain inverse scattering by means of neural networks," *IEEE AP-S Int. Symp. Dig.*, Salt Lake City, UT, 2000, pp. 1760-1763.
- [92] M.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, pp. 1492-1494, 1998.
- [93] M. Vai and S. Prasad, "Neural networks in microwave circuit design - Beyond black box models," *Int. J. RF Microwave CAE*, vol. 9, pp. 187-197, 1999.
- [94] R. Biernacki, J.W. Bandler, J. Song, and Q.J. Zhang, "Efficient quadratic approximation for statistical design," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1449-1454, 1989.
- [95] P.B.L. Meijer, "Fast and smooth highly nonlinear multidimensional table models for device modeling," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 335-346, 1990.
- [96] Q. Zhang, F. Wang, and V. Devabhaktuni, "Neural network structures for EM/microwave modeling," *IEEE AP-S Int. Symp. Dig.*, Orlando, FL, 1999, pp. 2576-2579.

- [97] 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.
- [98] S. Haykin, *Neural Networks: A Comprehensive Foundation*, New York, NY: IEEE Press, 1994.
- [99] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, vol. I, D.E. Rumelhart and J.L. McClelland, Eds., Cambridge, MA: MIT Press, 1986.
- [100] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Cambridge, MA: Cambridge University Press, 1986.
- [101] T.R. Cuthbert Jr., "Quasi-Newton methods and constraints," in *Optimization Using Personal Computers*, New York, NY: John Wiley & Sons, 1987.
- [102] A.J. Shepherd, "Second-order optimization methods," in *Second-Order Methods for Neural Networks*, Berlin, NY: Springer-Verlag, 1997.
- [103] J.A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, pp. 308-313, 1965.
- [104] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671-680, 1983.
- [105] J.C.F. Pujol and R. Poli, "Evolving neural networks using a dual representation with a combined crossover operator," *Proc. Int. Conf. Evol. Comp.*, Anchorage, Alaska, 1998, pp. 416-421.

- [106] A. Veluswami, M.S. Nakhla, and Q.J. Zhang, "The application of neural networks to EM-based simulation and optimization of interconnects in high-speed VLSI circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 712-723, 1997.
- [107] S.M. Sze, *Physics of Semiconductor Devices*, New York, NY: Wiley, 1981.
- [108] W.R. Curtice, "GaAs MESFET modeling and nonlinear CAD," *IEEE Trans. Microwave Theory Tech.*, vol. 36, pp. 220-230, 1988.
- [109] *NeuroADS*, Dr. Q.J. Zhang, Department of Electronics, Carleton University, 1125 Colonel By Drive, Ottawa K1S 5B6, Ontario, Canada.
- [110] *SONNET-LITE v.7.0*, Sonnet Software Inc., 1020 Seventh North St., Suite 210, Liverpool, NY 13088.
- [111] C.S. Walker, *Capacitance, Inductance and Crosstalk Analysis*, Boston, MA: Artech House, 1990.
- [112] P.H. Ladbrooke, *MMIC Design: GaAs FET's and HEMT's*, Boston, MA: Artech House, 1989.
- [113] Q.J. Zhang, K.C. Gupta, and V.K. Devabhaktuni, "Artificial neural networks for RF and microwave design: From theory to practice," (Accepted), *IEEE Trans. Microwave Theory Tech.*
- [114] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359-366, 1989.
- [115] V.K. Devabhaktuni, M.C.E. Yagoub, Y. Fang, J. Xu, and Q.J. Zhang, "Neural networks for microwave modeling: Model development issues and nonlinear modeling techniques," *Int. J. RF Microwave CAE*, vol. 11, pp. 4-21, 2001.

- [116] 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, pp. 630-645, 1997.
- [117] R. Reed, "Pruning algorithms - A survey," *IEEE Trans. Neural Networks*, vol. 4, pp. 740-747, 1993.
- [118] A. Krzyzak and T. Linder, "Radial basis function networks and complexity regularization in function learning," *IEEE Trans. Neural Networks*, vol. 9, pp. 247-256, 1998.
- [119] J. de Villiers and E. Barnard, "Backpropagation neural nets with one and two hidden layers," *IEEE Trans. Neural Networks*, vol. 4, pp. 136-141, 1992.
- [120] M.S. Nakhla and J. Vlach, "A piecewise harmonic-balance technique for determination of periodic response of nonlinear systems," *IEEE Trans. Circuits Syst.*, vol. 23, pp. 85-91, 1976.
- [121] J. Moody and C.J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 281-294, 1989.
- [122] G. Thimm and E. Fiesler, "High-order and multilayer perceptron initialization," *IEEE Trans. Neural Networks*, vol. 8, pp. 349-359, 1997.
- [123] D.G. Luenberger, *Linear and Nonlinear Programming*, Reading, MA: Addison-Wesley, 1989.
- [124] K. Ochiai, N. Toda, and S. Usui, "Kick-out learning algorithm to reduce the oscillation of weights," *Neural Networks*, vol. 7, pp. 797-807, 1994.
- [125] S.J. Perantonis and D.A. Karras, "An efficient constrained learning algorithm with momentum acceleration," *Neural Networks*, vol. 8, pp. 237-249, 1995.

- [126] N.B. Karayiannis and A.N. Venetsanopoulos, *Artificial Neural Networks: Learning Algorithms, Performance Evaluation, and Applications*, Boston, MA: Kluwer Academic Publishers, 1993.
- [127] N. Morgan, "Big dumb neural nets: a working brute force approach to speech recognition," *Proc. Int. Conf. Neural Networks*, Orlando, Florida, 1994, pp. 4462-4465.
- [128] F.Wang and Q.J. Zhang, "A sparse matrix approach to neural network training," *Proc. Int. Conf. Neural Networks*, Perth, Australia, 1995, pp. 2743-2747.
- [129] *Neural Network Toolbox: For Use with Matlab*, Natick, MA: The MathWorks Inc., 1993.
- [130] R.A. Jacobs, "Increased rate of convergence through learning rate adaptation," *Neural Networks*, vol. 1, pp. 295-307, 1988.
- [131] F. Wang and Q.J. Zhang, "An adaptive and fully sparse training approach for multilayer perceptrons," *Proc. Int. Conf. Neural Networks*, Washington, DC, 1996, pp. 102-107.
- [132] R. Battiti, "Accelerated backpropagation learning: two optimization methods," *Complex Systems*, vol. 3, pp. 331-342, 1989.
- [133] M. Arisawa and J. Watada, "Enhanced back-propagation learning and its application to business evaluation," *Proc. Int. Conf. Neural Networks*, Orlando, Florida, 1994, pp. 155-160.
- [134] A.G. Parlos, B. Fernandez, A.F. Atiya, J. Muthusami, and W.K. Tsai, "An accelerated learning algorithm for multilayer perceptron networks," *IEEE Trans. Neural Networks*, vol. 5, pp. 493-497, 1994.

- [135] X.H. Yu, G.A. Chen, and S.X. Cheng, "Dynamic learning rate optimization of the backpropagation algorithm," *IEEE Trans. Neural Networks*, vol. 6, pp. 669-677, 1995.
- [136] N. Baba, Y. Mogami, M. Kohzaki, Y. Shiraishi, and Y. Yoshida, "A hybrid algorithm for finding the global minimum of error function of neural networks and its applications," *Neural Networks*, vol. 7, pp. 1253-1265, 1994.
- [137] P.P. Van-Der-Smagt, "Minimisation methods for training feedforward neural networks," *Neural Networks*, vol. 7, pp. 1-11, 1994.
- [138] D.R. Hush and J.M. Salas, "Improving the learning rate of back propagation with the gradient reuse algorithm," *Proc. Int. Conf. Neural Networks*, San Diego, CA, 1988, pp. 441-447.
- [139] D.B. Parker, "Optimal algorithms for adaptive network: second order back propagation, second order direct propagation and second order hebbian learning," *Proc. Int. Conf. Neural Networks*, San Diego, CA, 1987, pp. 593-600.
- [140] R.L. Watrous, "Learning algorithms for connectionist networks: applied gradient methods of nonlinear optimization," *Proc. Int. Conf. Neural Networks*, San Diego, CA, 1987, pp. 619-627.
- [141] R. Battiti, "First and second-order methods for learning: between steepest descent and Newton's method," *Neural Computation*, vol. 4, pp. 141-166, 1992.
- [142] J.S.R. Jang, C.T. Sun, and E. Mizutani, "Derivative-based optimization," in *Neuro-Fuzzy and Soft Computing, A Computational Approach to Learning and Machine Intelligence*, Upper Saddle River, NJ: Prentice Hall, 1997.

- [143] M.F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, pp. 525-533, 1993.
- [144] K.R. Nakano, "Partial BFGS update and efficient step-length calculation for three-layer neural network," *Neural Computation*, vol. 9, pp. 123-141, 1997.
- [145] S. McLoone and G.W. Irwin, "Fast parallel off-line training of multilayer perceptrons," *IEEE Trans. Neural Networks*, vol. 8, pp. 646-653, 1997.
- [146] S. Kollias and D. Anastassiou, "An adaptive least squares algorithm for the efficient training of artificial neural networks," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1092-1101, 1989.
- [147] M.G. Bello, "Enhanced training algorithms, and integrated training/architecture selection for multilayer perceptron networks," *IEEE Trans. Neural Networks*, vol. 3, pp. 864-875, 1992.
- [148] G. Zhou and J. Si, "Advanced neural-network training algorithm with reduced complexity based on jacobian deficiency," *IEEE Trans. Neural Networks*, vol. 9, pp. 448-453, 1998.
- [149] T. Rögnavaldsson, "On Langevin updating in multilayer perceptron," *Neural Computation*, vol. 6, pp. 916-926, 1994.
- [150] R. Brunelli, "Training neural nets through stochastic minimization," *Neural Networks*, vol. 7, pp. 1405-1412, 1994.
- [151] R. Griffith and M.S. Nakhla, "Time-Domain analysis of lossy coupled transmission lines," *IEEE Trans. Microwave Theory Tech.*, vol. 38, pp 1480-1487, 1990.

- [152] M.A. Khatibzadeh and R.J. Trew, "A large-signal, analytical model for the GaAs MESFET," *IEEE Trans. Microwave Theory Tech.*, vol. 36, pp. 231-238, 1988.
- [153] *MDS v.6.02.00*, Agilent Technologies, 395 Page Mill Road, P.O. Box 10395, Palo Alto, CA 94303.
- [154] D.S. Chen and R.C. Jain, "A robust back propagation learning algorithm for function approximation," *IEEE Trans. Neural Networks*, vol. 5, pp. 467-479, 1994.
- [155] C.T. Leung and T.W.S. Chow, "Noise robustness enhancement using fourth-order cumulants cost function," *Proc. Int. Conf. Neural Networks*, Washington, DC, 1996, pp. 1918-1923.
- [156] S.E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing*, San Mateo, CA: Morgan Kauffman, 1990.
- [157] S.L. Beux, G. Cazuguel, B. Solaiman, and C. Roux, "Automatic feature determination using unsupervised neural networks," *Proc. Int. Conf. Neural Networks*, Washington, DC, 1996, pp. 1406-1409.
- [158] S. Kaski, "Dimensionality reduction by random mapping: Fast similarity computation for clustering," *Proc. Int. Conf. Neural Networks*, Anchorage, Alaska, 1998, pp. 413-418.
- [159] V.K. Devabhaktuni, C. Xi, F. Wang, and Q.J. Zhang, "An iterative multi-stage algorithm for robust training of RF/microwave neural models," *Proc. Asia Pacific Conf. Circuits Syst.*, Tianjin, China, 2000, pp. 327-330.

- [160] 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, pp. 174-181, 1987.
- [161] P. Huber, *Robust Statistics*, New York, NY: Wiley, 1981.
- [162] G. Li and K. Madsen, "Robust nonlinear data fitting," in *Numerical Analysis*, D.F. Griffiths and G.A. Watson, Eds., *Pitman Research Notes in Mathematics Series* 170, 1988.
- [163] J.W. Bandler, S.H. Chen, R.M. Biernacki, L. Gao, K. Madsen, and H. Yu, "Huber optimization of circuits: A robust approach," *IEEE Trans. Microwave Theory Tech.*, vol. 41, pp. 2279-2287, 1993.
- [164] J.W. Bandler, S.H. Chen, R.M. Biernacki, and K. Madsen, "The Huber concept in device modeling, circuit diagnosis and design centering," *Int. Symp. Circuits Syst. Dig.*, London, England, 1994, pp. 129-132.
- [165] *HSPICE v.2001.2*, Synopsys Inc., 700 East Middlefield Road, Mountain View, CA 94043.
- [166] *Serenade*, Ansoft Corporation, 669 River Drive, Suite 200, Elmwood Park, NJ 07407-1361.
- [167] Djordjevic, R.F. Harrington, T. Sarkar, and M. Bazdar, *Matrix Parameters for Multiconductor Transmission Lines: Software and User's Manual*, Boston, MA: Artech House, 1989.
- [168] N.K. Treadgold and T.D. Gedeon, "Extending CasPer: a regression survey," *Proc. Int. Conf. Neural Information Processing*, Dunedin, New Zealand, 1997, pp. 310-313.

- [169] J.D. Geest, T. Dhaene, N. Fache, and D.D. Zutter, "Adaptive sampling algorithm for accurate modeling of general interconnection structures," *Proc. European Microwave Conf.*, Munich, Germany, 1999, pp. 178-181.
- [170] R. Lehmensiek and P. Meyer, "An efficient adaptive frequency sampling algorithm for model-based parameter estimation as applied to aggressive space mapping," *Microwave Optical Tech. Lett.*, vol. 24, pp. 71-78, 2000.
- [171] U. Beyer and F. Smieja, "Data exploration with reflective adaptive models," *Comp. Statistics and Data Anal.*, vol. 22, pp. 193-211, 1996.
- [172] T. Lenihan, L. Schaper, Y. Shi, G. Morcan, and J. Parkerson, "Embedded thin film resistors, capacitors and inductors in flexible polyimide films," *Proc. Electronic Components Tech. Conf.*, Orlando, FL, 1996, pp. 119-124.
- [173] 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. Microwave Theory Tech.*, vol. 48, pp. 2307-2315, 2000.
- [174] J.W. Bandler, R.M. Biernacki, S.H. Chen, P.A. Grobelny, and R.H. Hemmers, "Space mapping technique for electromagnetic optimization," *IEEE Trans. Microwave Theory Tech.*, vol. 42, pp. 2536-2544, 1994.
- [175] *MINIMOS v.6.1*, Institute for Microelectronics, Technical University Vienna, Vienna, Austria.