Neural-Network-Based Transient Modeling of Nonlinear Electronic Circuits for High-Speed Applications

By

Yi Cao, M.A.Sc.

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

Ottawa-Carleton Institute for
Electrical and Computer Engineering

Department of Electronics
Carleton University
Ottawa, Ontario, Canada
April 2009

Copyright ©
2009 – Yi Cao

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-52062-8
Our file Notre référence
ISBN: 978-0-494-52062-8

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

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

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

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

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Artificial neural networks (ANN) have recently emerged as useful computational tools for the computer-aided design (CAD) of nonlinear radio frequency (RF)/microwave devices and circuits. The overall objective of this thesis is to develop efficient and robust neural network methodologies for transient modeling and simulation of nonlinear electronic circuits. The first contribution of the thesis is the development of a novel neural-network-based transient modeling technique. Specifically, a state-space dynamic neural network (SSDNN) technique is proposed for modeling the transient behavior of essential nonlinear blocks in high-speed systems, such as drivers and receivers. The proposed SSDNN expands the existing time-domain ANN structure into a general and flexible formulation for transient-oriented modeling. Under this generalized framework, a set of stability criteria is presented for verifying both local/global stabilities of a trained SSDNN model. Furthermore, a new training algorithm is developed that incorporates the proposed stability criteria into model training as a series of constraints. Using the proposed constrained training, the trained SSDNN model can maintain high accuracy while preserving global stabilities. As a further contribution, this thesis presents a new training approach for robust transient modeling of nonlinear circuits based on the recurrent neural network (RNN) formulation. One of the main challenges of nonlinear transient modeling using neural networks is that the training information depends on the

shapes of circuit waveforms and/or circuit terminations. To overcome this limitation, an internal space of the RNN is exploited to relate the information of training waveforms with that of test waveforms. Based on this concept, a new circuit block combining a generic load and a generic excitation is formulated as a general termination of the original circuit. By sweeping the coefficients of the proposed circuit block, a rich combination of training waveforms is obtained to effectively cover the region of interest in the internal space of the RNN. Through this systematic approach, the resulting RNN model maintains good accuracy for versatile test waveforms and test loads that are not known during training. The techniques presented in this thesis provide a further advancement beyond the existing methods, enabling accurate and robust transient modeling of nonlinear circuits for high-speed CAD.

To my parents

Shizhong Cao and Shurong Duan

Acknowledgements

First of all, I would like to express my deepest gratitude to my supervisor, Dr. Qi-jun Zhang, for his professional guidance, invaluable inspiration, continuous financial support and expert supervision throughout the research work and the preparation of this thesis. His expert leadership and vision for high-level research have made the completion of this thesis a challenging, rewarding and stimulating journey. His constant striving for rigorous research at the highest level will influence me for the future of my professional life.

Specially, I would like to express my appreciations to Dr. Mustapha C.E. Yagoub for his continuous support, guidance, and involvement during various stages of my Ph.D. study. Dr. Ihsan Erdin of Celestica Design Services, Ottawa, Canada is thanked for providing several I/O buffer devices that have been used for the validations of the algorithms presented in this thesis. Dr. Vijay K. Devabhaktuni and Dr. J. Xu provided me enormous help through frequent technical discussions during the initial stage of the thesis. Dr. Lei Zhang is thanked for active collaborations and discussions. Humayun Kabir and other students in my group are thanked for the help, friendship and productive discussions.

In addition, I greatly appreciate the valuable academic advices and assistance provided by Dr. Michel Nakhla and Dr. Ram Achar. I am also grateful to Nagui Mikhail, Scott Bruce, Jacques Lemieux, Blazenka Power, Peggy Piccolo, Sylvie Beekmans, and

all other staff for providing excellent lab facilities and an enjoyable environment in the department for study and research. Financial support from the Department of Electronics in the form of Teaching Assistantships, from Faculty of Graduate Study and Research in the form of an Indira-Gandi Fellowship, and from Ministry of Training, Colleges and University of Ontario in the form of Ontario Graduate Scholarships, is greatly acknowledged.

Finally, I would like to thank my parents, the two most influential persons in my life. Without their love as well as years of support and encouragement, the accomplishment of this thesis would not have been possible. They have given me tremendous strength to go through such difficult times in my journey of pursuing a Ph.D. degree. I am happy that I can now proudly say that I have lived up to their expectations. I dedicate this thesis to them.

Table of Contents

Abstract	i
Acknowledgements	iv
Table of Contents	vi
List of Tables	ix
List of Figures	xi
List of Symbols	xv
Nomenclature	xxiv
Chapter 1: Introduction	1
1.1 Background	1
1.2 Motivations	2
1.3 Contributions of the Thesis	5
1.4 Thesis Organization	8
Chapter 2: Literature Review	10
2.1 Overview of ANNs for RF and Microwave Applications	10
2.2 Neural Network Structures	13
2.2.1 Multilayer Perceptron Networks	14
2.2.2 Radial Basis Function Networks	20
2.2.3 Time Domain Neural Networks	21
2.3 Training Algorithms	29
2.3.1 Training Objective	29
2.3.2 Back Propagation Algorithms	31

2.3.3 Gradient-Based Optimization Methods	33
2.3.4 Global Optimization Methods.....	37
2.4 Existing Transient Behavioral Modeling Approaches for Nonlinear Circuits.....	38
2.4.1 Equivalent-Circuit-Based Approaches.....	39
2.4.2 ANN-Based Approaches.....	41
2.4.3 Nonlinear Model Order Reduction Techniques.....	44
2.5 Summary	47
Chapter 3: Nonlinear Transient Modeling and Stability Analysis for High-Speed Applications Exploiting State-Space Dynamic Neural Networks	49
3.1 Introduction.....	50
3.2 Proposed SSDNN Modeling Technique	53
3.3 Stability Analysis of SSDNN models	55
3.3.1 SSDNN Parameters for Stability Analysis	55
3.3.2 Stability Analysis via Lyapunov Method	57
3.4 SSDNN Sensitivity and Model Training	64
3.4.1 Adjoint Sensitivity for SSDNN	64
3.4.2 SSDNN Training.....	67
3.5 Demonstration Examples	70
3.5.1 CMOS Inverter.....	70
3.5.2 Receiver Buffer	74
3.5.3 Physics-Based Driver Buffer	77
3.5.4 8-Bit High-Speed Driver.....	81
3.6 Summary	86
Chapter 4: A Robust Training Approach for Transient Modeling of Nonlinear Circuits Based on Recurrent Neural Networks	88
4.1 Introduction.....	89
4.2 Overview of Nonlinear Transient Modeling using RNN formulations	91
4.3 Proposed Training Approach for RNN-Based Nonlinear Modeling	93
4.3.1 Problem Statement.....	93

4.3.2 Formulation to Explore the Internal Input-Neuron Space	96
4.3.3 Data Reduction Method	104
4.3.4 Discussion	107
4.3.5 Overall Training Algorithm	110
4.4 Demonstration Examples	113
4.4.1 Transient Behavioral Modeling of a High-Speed Driver.....	113
4.4.2 Large Signal Behavioral Modeling of an Amplifier.....	122
4.4.3 Transient Behavioral Modeling of a Four-Stage Driver from a Physics-Based Simulator.....	132
4.5 Summary	138
Chapter 5: Conclusions and Future Research	139
5.1 Conclusions.....	139
5.2 Further Directions	142
Appendix A: Derivation of Adjoint Sensitivity for SSDNN	148
Bibliography	151

List of Tables

Table 3.1: Comparisons of Matrix Eigenvalues using Different Stability Criteria for the CMOS Inverter Example. Model I is Obtained by Original Unconstrained Training Method and Model II is Obtained by Our Proposed Constrained Training Method.....	72
Table 3.2: Eigenvalues for Stability Verification of the SSDNN Model for the Receiver Example, and Comparison between Proposed Constrained Training versus Unconstrained Training.....	76
Table 3.3: Comparison of the Model Test Error between Discrete RNN, DNN and SSDNN for Transient Modeling of a Physics-Based Driver.....	79
Table 3.4: Eigenvalues for Stability Verification of the SSDNN Model for the Physics-Based Driver Example, and Comparison between Proposed Constrained Training versus Unconstrained Training.....	80
Table 3.5: Comparisons of CPU Time and Accuracy for Various Transient Models for HSPICE Simulation in the TI Driver Example.	86
Table 4.1: Comparison of the Test Accuracy of the RNN Model between the Existing and Proposed Methods for Various Test Loads in the High-Speed Driver Example	118
Table 4.2: CPU Comparison between the Detailed Transistor-Level Model and the RNN Model Trained with the Proposed Technique in the High-Speed Driver Example.....	121
Table 4.3: Comparison of the Coverage Rate of the Internal Input-Neuron Space (Q_{ie}) and the Model Test Accuracy for Different Test Loads in the Amplifier Example.....	126

Table 4.4: Convergence Rate of the RNN Model in HB Simulation for Various Input Power Levels and Resistor Loads in the Amplifier Example.....	128
Table 4.5: Comparison of the Coverage Rate Q_{te} between the Existing Methods and the Proposed Technique for Various Test Loads in the Four-Stage Driver Example	136
Table 4.6: Comparison of the Test Accuracy of the RNN Model between the Existing Methods and the Proposed Technique for Different Test Loads in the Four-Stage Driver Example	137

List of Figures

Figure 2.1: MLP neural network structure. Typically, an MLP network consists of one input layer, one or more hidden layers, and one output layer.....	15
Figure 2.2: RBF neural network structure.	22
Figure 2.3: The discrete-time RNN based macromodel structure for nonlinear dynamic modeling [5]. τ represents the delay step.	24
Figure 2.4: A general DNN structure and its training process in time-domain using spectrum data, where $B^{(i)}$ is the time derivative operator [6].	30
Figure 2.5: The basis structure of the IBIS model for an output buffer.....	41
Figure 3.1: A three-layer MLP neural network used in the SSDNN. The MLP connection weights are grouped into three parts, as defined by W_u , W_s and W_o . The MLP input neurons include SSDNN overall input signals $u_i(t)$ and state variables $x_i(t)$. The MLP output neurons are f_{ANN} in state-space equations.	57
Figure 3.2: Flowchart for the proposed SSDNN constrained training algorithm for enforcing the GAS of a trained SSDNN model. The training error sensitivities are efficiently computed by exploiting the adjoint SSDNN model.	69
Figure 3.3: Circuit representation of the SSDNN nonlinear model. All the resistor and capacitor values are unity.....	70
Figure 3.4: The state trajectories of the SSDNN for modeling a CMOS inverter under 20 random initial conditions: (a) Model I without GAS property where state variables converge to different equilibrium points, meaning that none of the equilibrium points are globally asymptotically stable. (b) Model II with GAS property where state variables converge to a unique equilibrium point.	73

Figure 3.5: The schematic of a TI 8-bit high-speed driver to be modeled by SSDNN. Training waveforms of $v_{in}(t)$, $v_{out}(t)$, and $i_{out}(t)$ are generated by HSPICE simulations using detailed transistor-level models for developing the SSDNN model.....	83
Figure 3.6: Sample comparisons of the output voltage responses using detailed transistor-level model (—), IBIS model (---), and proposed SSDNN model (o) for the TI driver example. The comparisons shown here are the cases for (a) $T_r = 1.25\text{ns}$ and $d = 1.25\text{cm}$, and (b) $T_r = 1.75\text{ns}$ and $d = 2.75\text{cm}$. Good agreement is achieved between the responses of the proposed SSDNN models and those of detailed transistor-level models even though such test waveforms are never used in training.....	84
Figure 3.7: The state-space trajectories of the SSDNN model in the TI driver example. These trajectories are generated by transient simulation of the model with randomly selected initial conditions, as $t \rightarrow \infty$. The SSDNN model is trained to be globally asymptotically stable through our proposed constrained training algorithm. The model state variables x_1 and x_2 all converge to a unique equilibrium point as observed from the figure.....	85
Figure 4.1: Illustration of the external waveform space of an RNN and the internal input-neuron space. The entire diagram represents the RNN and the box with a dashed border represents the feedforward neural network.	94
Figure 4.2: The proposed circuit block with a generic load and excitation to explore the internal input-neuron space for training RNN models.....	98
Figure 4.3: Illustration of the hyper-plane realized by the proposed formulation in the internal input-neuron space. (a) $\hat{a}_0 = 0.02$, $\hat{a}_1 = 0.01$, $\hat{b}_1 = 0.1$, and $v_E(t) = 2$. (b) $\hat{a}_0 = 0.02$, $\hat{a}_1 = 0.07$, $\hat{b}_1 = 0.6$, and $v_E(t) = -2$. By sweeping the coefficients in the proposed formulation, the hyper-plane of (4.7) is allowed to have different angles and positions in the internal input-neuron space...	102
Figure 4.4: A flowchart illustrating the overall algorithm of the proposed training approach for developing RNN models of nonlinear circuits.	112

Figure 4.5: Sample distribution of the training data generated with the proposed technique (·) and the test data (—) in the high-speed driver example. Good coverage of the internal input-neuron space allows this model to respond accurately to test signals of unknown shapes.....	116
Figure 4.6: Output voltages of the high-speed driver for different nonlinear loads. (a) The nonlinear load contains a transmission line with length of 2000 mil. (b) The nonlinear load contains a transmission line with length of 4000 mil. The proposed method leads to accurate RNN models for a variety of test loads that are never applied in training.....	119
Figure 4.7: Crosstalk signals at the output port of the quiet driver in the high-speed driver example. The proposed technique enables the RNN model to produce accurate results of the crosstalk signals that are never seen during training.	120
Figure 4.8: The RF amplifier circuit to be modeled by RNNs.	123
Figure 4.9: Sample distribution of training data using the proposed technique (·) and the test data (—) in the internal input-neuron space for the amplifier example. Through the proposed technique, the training data can sufficiently cover the region of the test waveforms whose shapes are not known during training.	125
Figure 4.10: The amplifier output power versus input power at different loads and fundamental frequencies. (a) $Z_L = 40 - j$ and $\omega_E = 950$ MHz. (b) $Z_L = 70 + 0.5j$ and $\omega_E = 1050$ MHz. Good agreements are achieved between the RNN model trained with the proposed technique and the original amplifier even though such test loads are never considered during training.	127
Figure 4.11: Output IMD3 power of the amplifier circuit for different test loads: (a) Lower side IMD3. (b) Upper side IMD3. The frequency spacing of input signals is 50 kHz. Accurate output IMD3 results are obtained using the RNN	

model trained with the proposed technique, even though the test loads are never seen in training.	129
Figure 4.12: The output power spectrum of the amplifier circuit under modulated test signals. Different test loads are used for this validation: (a) $Z_L = 45 + j$. (b) $Z_L = 65 + j$. By using the proposed technique, accurate spectrum regrowth predictions of the RNN model are achieved for different amplifier loads that are never used in training.	131
Figure 4.13: The four-stage CMOS driver circuit to be modeled by an RNN. The data generation is carried out by driving a physics-based simulator MINIMOS-NT.	133
Figure 4.14: Sample distribution of the training data with the proposed technique (o) and the test data (—) in the internal input-neuron space for the four-stage driver example. The proposed technique leads to good coverage of the region of the test data whose shapes are not known during training.	135
Figure 4.15: Output voltage of the four-stage driver for the transmission line loads with different lengths. Such test loads have not been used for training the RNN model.	137

List of Symbols

$\mathbf{a}$	A vector containing the numerator coefficients of the rational function $D(s)$
$\mathbf{A}_{local}$	The test matrix for local stability of an SSDNN model
$\mathbf{A}_{global}$	The test matrix for global stability of an SSDNN model
$\mathbf{b}$	A vector containing the denominator coefficients of the rational function $D(s)$
$\mathbf{B}(\omega, t)$	The coefficients of Inverse Fourier Transform for DNN training
$\tilde{\mathbf{c}}_{ij}$	The centers of radial basis activation functions in the RBF network
$\mathbf{c}_{nj}, n=1,2$	A vector containing the positions (centers) of the Gaussian basis functions in the regressor space $\mathbf{x}_n$ of the piece-wise RBF formulation
$\mathbf{C}$	The output measurement matrix in SSDNN
$\mathbf{d}_k$	A vector containing the desired outputs of the neural model for inputs $\tilde{\mathbf{u}}_k$

$D(s)$	The generic load in the form of a rational function representing a complex admittance
$E(\boldsymbol{w})$	The cost (error) function to be minimized during neural network training
E_d	The objective function (overall training error) for SSDNN training
E_d^m	The per-waveform training error for the m th waveform for SSDNN training
E_k	A per-sample error function when one training sample is used at a time to update the $\boldsymbol{w}$ in a neural network
E_R	The l_2 error of the RNN model over all the time samples per waveform as well as over all the waveforms
$E_R^j(k)$	The l_2 error of the RNN model for the k th training sample in the j th waveform
$\boldsymbol{f}_{ANN}$	A vector function representing an MLP neural network with multiple outputs
$\boldsymbol{f}_{FFNN}$	A static mapping function to be represented by a standard feedforward neural network
$F_L(.)$	A candidate of the Lyapunov function
$\boldsymbol{g}$	Gradient vector used during gradient-based training methods
$\boldsymbol{g}_{initial}$, $\boldsymbol{g}_{now}$, and $\boldsymbol{g}_{next}$	The gradient vector at the initial epoch, present epoch, and next epoch of training, respectively

$G(t)$	A vector containing the derivatives of the hidden neuron activation functions in SSDNN
h	The sampling time interval for RNN modeling
H	The Hessian matrix of the cost function $E(\boldsymbol{w})$
$i_d(t)$	Driver output port current
$i_o(t)$ or $i_{out}(t)$	The output port current of a nonlinear circuit
I_s	An index set of training samples
I_T and I_V	The index set for the group of sub-regions occupied by training and test samples, respectively
I_l	An identity matrix used in constructing A_{local}
I_g	An identity matrix used in constructing A_{global}
J^L	A vector representing the lower bounds of the region of interest in internal input-neuron space of RNN
J^U	A vector representing the upper bounds of the region of interest in internal input-neuron space of RNN
k_i	The Lipschitz constant for σ_i
K	The number of dynamic outputs for a nonlinear circuit to be modeled by a time-domain neural network
L	The total number of layers in an MLP network
m	The number of delay steps for the input signals $\boldsymbol{u}(t)$ in the RNN formulation

M	The number of dynamic inputs for a nonlinear circuit to be modeled by a time-domain neural network
n	The number of delay steps for the output signals $y(t)$ in the RNN formulation
n_D	The order of DNN representing the reduced order for the original nonlinear circuit
N	The dimensions of the state-space in SSDNN, or the order of the SSDNN model
$\bar{N}$	The total number of neurons in a Hopfield network
N_f	The number of harmonics to be considered in the generic excitation
N_h	The number of hidden neurons in a feed-forward neural network (MLP or RBF)
$\bar{N}_p$	The number of time-invariant (static) inputs to an RNN network
N_r	The total number of sub-regions in the region of interest for RNN modeling
N_s	The present number of training samples in the sub-region s
N_t	The number of time samples in $[T_1, T_2]$
N_W	The total number of waveforms that is used for training neural models

N_{wp}	The number of weight parameters within a neural network structure
$\tilde{N}_u$	The number of input neurons of a neural network
$\tilde{N}_y$	The number of output neurons of a neural network
$\mathbf{p}$	A vector containing all the parameters in the proposed generic load and excitation
$\bar{\mathbf{p}}$	A vector containing time-invariant (static) inputs to an RNN network
$\mathbf{q}$	A vector representing the number of sub-intervals for splitting each input dimension of a feedforward neural network
Q_{all} and Q_{te}	The coverage rate of the internal input-neuron space during and after data generation, respectively
$\mathbf{r}$	a direction vector used in iterative training algorithm
$\mathbf{r}_{initial}$, $\mathbf{r}_{now}$, and $\mathbf{r}_{next}$	The direction vector at the initial epoch, present epoch, and next epoch of training, respectively
R_0 and R_i	The voltage magnitude of the generic excitation at dc and the i^{th} harmonic frequency, respectively
T_r	Rise-time of a pulse signal
$[T_1, T_2]$	The time interval of input and output waveforms for training RNN models
$\tilde{\mathbf{u}}$	A vector containing the inputs to a neural network
$\mathbf{u}(t)$	A vector of time-domain input signals of a nonlinear circuit

$\mathbf{u}_0$	A vector of constant signals for $\mathbf{u}(t)$, $t > 0$
$\mathbf{u}_d(t)$	The input waveform training data from a nonlinear circuit
$\overline{\mathbf{U}}(\omega)$	The input spectrums for DNN training
$v_d(t)$	Driver output port voltage
$v_E(t)$	The generic excitation as part of the formulation to explore the internal input-neuron space
$v_i(t)$ or $v_{in}(t)$	The input port voltage waveforms of nonlinear circuits
$v_o(t)$ or $v_{out}(t)$	The output port voltage waveforms of nonlinear circuits
V_{dd} and V_{ss}	The high and low power supply voltages of the original circuit, respectively
V_i	The signal amplitude of the generic excitation at the i th time point
$w_1(k)$ and $w_2(k)$	Time varying weight coefficients for the submodel f_1 and f_2 in the piece-wise RBF formulation, respectively
w_{ij}^l	The weight of the link between j^{th} neuron of $(l-1)^{th}$ hidden layer and i^{th} neuron of l^{th} hidden layer in an MLP network
w_{i0}^l	The bias parameter of i^{th} neuron of l^{th} hidden layer in an MLP network
$\tilde{w}_{ki}$	The weight of the link between i^{th} neuron of the hidden layer and k^{th} neuron of the output layer in an RBF network

$\mathbf{w}$	The vector of weight parameters representing the connections in a neural network
$\mathbf{w}_{initial}$, $\mathbf{w}_{old}$, $\mathbf{w}_{now}$, and $\mathbf{w}_{next}$	The vector of weight parameters at the initial epoch, last epoch, present epoch, and next epoch of training, respectively
$\mathbf{W}_s$	A matrix of the connection weights between state variables and all the hidden neurons of $\mathbf{f}_{ANN}$ in SSDNN formulation
$\mathbf{W}_u$	A matrix of the connection weights between external input signals and all the hidden neurons of $\mathbf{f}_{ANN}$ in SSDNN formulation
$\mathbf{W}_s^{(i)}$ and $\mathbf{W}_u^{(i)}$	A vector containing the i^{th} row of $\mathbf{W}_s$ and $\mathbf{W}_u$, respectively
$\mathbf{x}(t)$	The vector of the state variables in SSDNN formulation
$\bar{\mathbf{x}}(t)$	The vector containing state variables for original nonlinear circuits
$\hat{\mathbf{x}}(t)$	A vector of state variables in the adjoint SSDNN model
$\mathbf{x}_e$	A vector of the equilibrium points of the SSDNN model
$\mathbf{x}_n(k)$, $n=1,2$	The regressor vector that collects time-delayed signals of voltages and currents at port n in the piece-wise RBF formulation
$\tilde{\mathbf{y}}$	A vector containing the outputs from the output neurons of a neural network
$\mathbf{y}(t)$	A vector of time-domain output signals of a nonlinear circuit

$y^j(t)$	A vector of neural model responses due to the j^{th} input training waveform
$y_d(t)$	The output waveform training data from a nonlinear circuit
$y_d^j(t)$	A vector containing the samples for the j^{th} output training waveform
$\bar{Y}(\omega)$	The output spectrums for DNN training
z_i^l	The output of i^{th} neuron of l^{th} hidden layer in an MLP network
$\sigma(.)$	The activation function of hidden neurons in a neural network
$\sigma(.)$	A vector of nonlinear activation functions for hidden neurons in f_{ANN}
δ_i^l	The local gradient at i^{th} neuron of l^{th} layer in an MLP network
κ_{ij}	The standard deviations of radial basis activation functions in an RBF network
λ_i	The i^{th} eigenvalues of the test matrices for stabilities, <i>i.e.</i> , A_{local} or A_{global}
ε	The maximum number of training samples that is allowed per sub-region
η	A positive step size along the r direction during gradient-based training
η^*	The optimal step size of η
α	The momentum factor used in BP algorithms

β_j	The j th spreading (scale parameter) of the Gaussian basis functions in the regressor space x_n of the piece-wise RBF formulation
ρ	A vector representing the position of the sub-region containing the present training sample
ω_E	The fundamental frequency of the generic excitation in the steady-state case
Γ	The Jacobian matrix
$\Theta_n, n = 1, 2$	A vector of internal parameters in the submodels of the piece-wise RBF formulation

Nomenclature

ADNN	Adjoint dynamic neural network
ADS	Advanced Design System
ANN	Artificial neural network
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BP	Back propagation
BPTT	Back propagation through time
CAD	Computer-aided design
CPU	Central processing unit
dc	Direct current
DQPSK	Differential quadrature phase shift keying
DNN	Dynamic neural network
EDA	Electrical design automation
EM	Electromagnetics
GaN	Gallium Nitride
GAS	Global asymptotical stability
GRNN	Globally recurrent neural network
HB	Harmonic balance

HBT	Heterojunction bipolar transistor
HEMT	High electron mobility transistor
HSTL	High-Speed Transceiver Logic
IBIS	Input/output buffer information specification
IC	Integrated circuit
IMD3	Third-order intermodulation
MLP	Multilayer perceptron
MOR	Model order reduction
RBF	Radial basis function
RF	Radio frequency
RNN	Recurrent neural network
SI	Signal integrity
SM	Space mapping
SSDNN	State-space dynamic neural network
<i>tanh</i>	hyperbolic tangent
TI	Texas Instruments
TPWL	Trajectory-based piecewise-linear
VLSI	Very large scale integration

Chapter 1: Introduction

1.1 Background

In recent years, rapid growth of today's radio frequency (RF) and microwave circuits has led to shrinking device dimensions and increasing functionalities. The advanced computer-aided design (CAD) tools that allow efficient design, analysis and optimization are highly demanded for first-pass success design and time-to-market. Through the effective use of CAD tools, the essential characteristics of circuits and systems could be simulated and optimized before building any physical hardware prototype.

The success of modern CAD algorithms depends on the availability, accuracy and speed of the models of each individual device/circuit. Therefore, modeling continues to be a key step towards the successful CAD of high-frequency components and circuits. In the past, a variety of modeling approaches have been introduced for high-frequency devices and circuits, such as detailed theoretical models, empirical models and equivalent circuit models. However, with the rapid growth of RF and microwave technologies, the conventional approaches may not be adequate to fully address the accuracy and speed requirements, especially for the newly involved devices/circuits. As such, there is a constant demand for the development of more advanced modeling techniques to meet the growing challenges for high-frequency CAD.

Over the past decade, artificial neural networks (ANN) have received much attention as unconventional and effective computational tools for modeling, simulation and optimization of modern RF and microwave devices and circuits [1]. ANNs are information-processing networks inspired by the human brain's ability of learning and generalizing. Neural networks have been successfully applied to many complicated tasks in areas such as control, telecommunications, biomedicine, remote sensing, pattern recognition, manufacturing etc. In the microwave area, neural networks have been used to model a wide variety of circuit components and the developed neural models have been used in circuit design and optimization [2][3]. Neural models can be developed through a computer-driven optimization process called training. The neural models can learn and abstract the device/circuit characteristics from the measured/simulated training data (samples) without any prior knowledge of the application. Theoretically, neural networks are capable of mapping arbitrary continuous nonlinear relationships between inputs and outputs. On the other hand, neural models are very fast because the structure of neural network is simple and the evaluation only involves several basic arithmetic operations (multiplications and additions). The attractive features including excellent combination of accuracy and speed capabilities have made ANN a good candidate for modeling high-frequency components and circuits.

1.2 Motivations

Efficient modeling of nonlinear blocks is of great importance to the simulation and design of today's complex nonlinear RF/microwave circuits and systems. Simulation of

nonlinear circuits based on complete and detailed circuit descriptions (*e.g.*, transistor-level models) is often a cumbersome task for design engineers due to its computational complexity and computer memory demands. It becomes computationally prohibitive to perform simulation and design of complete RF/Microwave systems built of many nonlinear circuit blocks. One possible solution is to use simplified but sufficiently accurate behavioral models to represent each nonlinear functional block in the systems. These behavioral models are high-level abstractions of the nonlinear functional blocks, enabling fast and accurate predictions of the overall system performance.

Recent cutting-edge research on neural network techniques has led to the development of neural network approaches targeted to modeling nonlinear dynamic behaviors. Various time-domain neural network formulations, such as time-delay neural networks [4], recurrent neural networks (RNN) [5], and dynamic neural networks (DNN) [6][7], have been presented for modeling the steady-state large signal behavior of nonlinear circuits including amplifiers and mixers. Recently, an adjoint dynamic neural network (ADNN) approach [8][9] was developed, paving the way for training neural models for nonlinear transient applications. The main motivation of this thesis is to efficiently and accurately model the time-domain transient behavior of the high-speed nonlinear devices and circuits, by further exploring the potentials of neural networks.

Nonlinear transient modeling is an important subject for signal integrity (SI) analysis and timing analysis for complex high-speed digital systems. For fast behavioral modeling of high-speed nonlinear circuits, such as high-speed I/O buffers, a generic time-domain neural network approach is desired, which should have the capability of modeling the

dynamic relationship between the time-domain input and output signals. Another important requirement for the new formulation is that it should be robust to handle high-frequency noise in the actual signals, which is often unavoidable when performing the measurements of the devices/circuits. In other words, the modeling technique should be able to produce models with satisfactory accuracies even if the input and output signals are in the presence of noise. Furthermore, the model formulation should have a straightforward circuit representation that enables convenient incorporation of the trained model into commercial CAD tools for system simulations and verifications. In addition, it is highly desired that the proposed modeling technique could take into account model stability property, which is a fundamental property of the model to ensure its reliability in nonlinear simulation. Stability criteria for local and global stabilities based on the model internal parameters are required in order to perform stability verifications of the trained models prior to their use in actual nonlinear simulations. As a further direction, the stability criteria should be considered during the model generation process such that the resulting models can automatically satisfy the requirement of stability.

Another strong motivation is the continuous demand for ANN-based transient behavioral models of nonlinear circuits with high robustness and accuracy for system-level simulation. Generally speaking, nonlinear circuit modeling is a more challenging task compared to linear circuit modeling. In the case of linear circuit modeling, the magnitude and phase of the transfer function over frequency provide complete information about the behavior of the original circuit. In other words, circuit terminations (loads) and signal shapes have no impact on the modeling process of the linear circuit. On

the contrary, for nonlinear circuit modeling, the choice of waveform shapes and terminations directly affects the training of the ANN model, because different waveform shapes and terminations produce different behavioral information of the nonlinear circuit. During system-level simulations, the ANN model is often presented with circuit waveforms and terminations whose details may not be known during the model development stage. In this case, it is a challenging task to ensure the accuracy and robustness of the ANN model considering the uncertainty about potential waveform shapes and terminations that the model has to deal with. In this thesis, we are also aimed at developing a new training approach for neural-network-based nonlinear modeling, which enables the trained model to faithfully respond to versatile waveform shapes and terminations.

1.3 Contributions of the Thesis

The overall direction of the thesis is to develop ANN-based algorithms for transient behavioral modeling of high-speed nonlinear circuits and to enhance the robustness and accuracy of ANN-based nonlinear models during high-level simulations. The main objectives are, (a) to develop a generic time-domain neural network formulation to model the transient behavior of high-speed nonlinear circuits without having to rely on the circuit details, (b) to derive a set of stability criteria that allow stability verifications and enforcement for the transient behavioral models based on neural networks, (c) to develop a new training approach for nonlinear transient modeling using RNN formulations, such that the resulting model maintains high accuracy for various shapes of waveforms that are not known during training. Accomplishing these goals could make the ANN-based CAD

technology become an attractive and viable approach for practical high-speed/high-frequency nonlinear CAD, offering combined benefits in terms of accuracy and speed. Corresponding to the aforementioned research objectives, the major contributions of this thesis are summarized as follows:

- A transient modeling technique of high-speed nonlinear circuits is developed, combining the continuous recurrent network and the state-space concept. The proposed technique, namely a state-space dynamic neural network (SSDNN) [10]-[13], extends the existing DNN approach into a generalized and robust formulation for representing transient behaviors. A training algorithm exploiting the adjoint sensitivity computation is developed to enable SSDNN to efficiently learn from the transient input and output waveform data without relying on the circuit internal details. An exact circuit representation of the proposed SSDNN is derived such that the trained SSDNN model can be conveniently used in commercial SPICE-like circuit simulators.
- For the first time, stability analysis methods are presented for ANN-based transient modeling of nonlinear microwave circuits. The stability criteria are derived for both local and global stabilities of SSDNN models [11]. Stability test matrices are formulated based on SSDNN internal weight parameters. These criteria can be conveniently applied to the stability verification of a trained SSDNN model using the eigenvalues of the test matrices. In addition, a constrained training algorithm is introduced by formulating the proposed stability

criteria as training constraints such that the resulting SSDNN models satisfy both the accuracy and stability requirements.

- A new training approach for developing robust and accurate transient models of nonlinear circuits is presented [14]. The proposed algorithm, which is formulated using RNN modeling structures, addresses a significant challenge of neural-based nonlinear circuit modeling where training information depends on the shapes of circuit waveforms and/or circuit terminations. To relate information of training waveforms with that of test waveforms, an internal space of the RNN, called internal input-neuron space, is judiciously exploited. A new circuit block combining a generic load and a generic excitation is proposed to terminate the nonlinear circuit to be modeled. Sweeping the coefficients of the proposed circuit block enables the systematic generation of training waveforms to cover effectively the region of interest in the internal-input-neuron space. To enhance the model training efficiency, a data reduction algorithm is formulated to reduce the amount of training data while maintaining the necessary modeling information.
- The proposed ANN-based modeling techniques are implemented in an objected-oriented fashion using MATLAB [15] and/or C++, and are further integrated into the general modeling framework of NeuroModelerPlus software [16]. In addition, programs based on C++ are developed for incorporating the trained SSDNN and RNN models into commercial CAD tools such as Agilent Advanced Design System (ADS) [17] and Synopsys HSPICE [18]. These computer programs have

been used to obtain the numerical results for the validation of the techniques presented in this thesis.

1.4 Thesis Organization

The rest of the thesis is organized as follows:

In Chapter 2, an overview of ANN-based RF and microwave modeling techniques, as well as existing transient behavioral modeling techniques for nonlinear circuits is presented. Relevant neural network structures and training algorithms for microwave modeling are described. State-of-the-art of behavioral modeling approaches and their applications to modeling transient behavior of nonlinear circuits are presented.

The SSDNN technique for nonlinear transient modeling and the stability criteria for trained SSDNN models are presented in Chapter 3. The formulation of nonlinear SSDNN models is introduced and its weight parameters are reformatted for stability analysis purposes. The stability analysis of the SSDNN model is then described, leading to a set of practically usable stability criteria. Based on the adjoint sensitivity concept, the SSDNN sensitivities for model training are derived. Furthermore, a constrained training scheme is proposed that automatically ensures the global asymptotical stability (GAS) of a trained SSDNN model. Several numerical examples for modeling high-speed receiver and driver buffers are given to validate the proposed technique.

In Chapter 4, a new training approach is presented for robust transient modeling of nonlinear circuits, considering RNN formulations. The concept of RNN formulation, which is capable of modeling both transient and steady-state nonlinear behaviors, is first

introduced. The internal input-neuron space of the RNN is then defined and further used to formulate a new circuit block for systematic generation of training data for RNN. To improve training efficiency, a data reduction method is described for shrinking the size of the training data. Application examples of modeling high-speed I/O buffers and an RF amplifier are presented to demonstrate the robustness and accuracy of the RNN models by using the proposed training approach.

Finally, in Chapter 5, a summary of the thesis is presented highlighting the key contributions of the thesis. Future research directions beyond the scope of the current work are also recommended.

Chapter 2: Literature Review

2.1 Overview of ANNs for RF and Microwave Applications

The recent advances of telecommunication and wireless industries have resulted in the continuous improvement of the performance of RF and microwave circuits. The wireless industry's emphases on time-to-market, low manufacturing cost, and first-pass-design, are putting higher demands on the reliability and efficiency of CAD tools for RF/microwave circuit design. New modeling techniques with high accuracy and fast speed are required for enabling cost-effective circuit optimization and design.

Neural networks, also called ANNs, are information processing systems with their design inspired by the studies of ability of human-brain to learn from observations and to generalize by abstraction [1]. The fact that neural networks can be trained to learn any arbitrary nonlinear input-output relationships from corresponding data has resulted in their use in a number of areas such as pattern recognition, control, speech processing, etc. Over the last decade, a CAD approach based on ANN gained recognition in RF, microwave and very large scale integration (VLSI) communities [3][19]. Fast, accurate, and reliable neural network models (or simply neural models) can be directly developed from measured/simulated microwave data. Once developed, these neural models can be

used to replace computationally intensive physics/electromagnetics (EM) models of active/passive devices to speed up microwave design. Through recent intensive researches, neural network techniques have been demonstrated to model a wide variety of microwave devices/circuits such as transmission line components [20]-[29], bends [30][31], vias [32]-[34], CPW components [34], microstrip components [35], FETs [36]-[40], Heterojunction Bipolar Transistors (HBT) [41][42], High Electron Mobility Transistors (HEMT) [43][44], waveguides [45], amplifiers [46]-[51], mixers [5][6], and antennas [52]-[57]. Neural networks have also been used in object wave propagation [58][59], impedance matching [60][61], inverse modeling [62]-[65], circuit optimization and synthesis [66][67], and yield prediction [68][69]. Neural models are much faster than conventional detailed physics/EM models, more accurate than polynomial and empirical models, allow more dimensions than table lookup models, and are easier to develop when a new device/technology is introduced [2].

To develop neural models, many crucial and nontrivial steps are involved such as data generation, data scaling, selecting neural network structures, neural network training, etc. Two of the most essential steps in the development of neural models are the determination of suitable neural network structures and the training of constructed neural networks.

Microwave modeling problems could be broadly divided into two categories, *i.e.*, static nonlinear problems and dynamic nonlinear problems. In the case of static problems, the input and output relationship of the problem is algebraic, and can be represented by various feedforward neural networks [1], which are effective tools for modeling multi-

dimensional and nonlinear problems. The most widely used neural network structure in this category for microwave modeling is Multilayer Perceptron (MLP) [1][3]. Other popular choices in this category include radial basis function (RBF) networks, wavelet neural networks, and so on.

Recently, nonlinear dynamic modeling has become an important subject for the CAD of nonlinear RF/Microwave circuits. In order to characterize the dynamic analog behavior of nonlinear circuits, where time domain information needs to be considered, conventional feedforward neural networks are not sufficient. To address this, neural network formulations taking into account the dynamic effects of circuit input and output signals are introduced. Two recent developments in this category are discrete-time RNN [5][70] and continuous-time DNN [6].

The ability of neural networks to capture the nonlinearity of the problem depends on the number of hidden neurons in the networks. Theoretically speaking, the more nonlinear the problem is, the more hidden neurons are needed. But too many hidden neurons may lead to the overlearning phenomenon [1], where the model only memorizes the training information but has poor generalization capabilities. On the other hand, too few hidden neurons may not represent the problem sufficiently, which is called underlearning. In an attempt to solve this problem, an automatic model generation algorithm [71] featuring dynamic adjustment of the network size has been proposed.

To begin with, a neural network does not know any information about the microwave device/circuit problem. In order to represent the device/circuit behavior, the neural network has to be trained to learn the information from the corresponding data. As such,

neural network training is a crucial step towards the development of accurate models. A good collection of the training data, which is well distributed, sufficient, and accurately measured/simulated, is the basic requirement to obtain an accurate model. Beyond that, choice of suitable training algorithms also plays a very important role in the success of model development. A good training algorithm will speed up the training procedure with better accuracy. In mid 1980's, backpropagation (BP) [72] scheme was introduced to provide the derivatives of output error with respect to each weight in the MLP network. Based on this scheme, basic weight updating and its variation methods were proposed firstly. After that, second-order gradient-based optimization methods have been applied to the ANN training, motivated by the fact that the training process is essentially an optimization procedure. Recently, global optimization techniques, such as simulated annealing [73] and genetic algorithm [74], were also proposed to overcome the problem of local minimum in the gradient-based training methods.

In the following sections, several popular neural network structures are reviewed according to their functionalities and applications, covering both static feedforward neural networks and time-domain neural networks. In addition, we describe commonly used training methods for microwave applications, including BP algorithms, gradient-based training algorithms, and global training algorithms.

2.2 Neural Network Structures

Let $\tilde{N}_u$ and $\tilde{N}_y$ represent the number of input and output neurons of the neural network.

Let $\tilde{\mathbf{u}}$ be an $\tilde{N}_u$ -vector containing the external inputs (stimuli) to the neural network, $\tilde{\mathbf{y}}$

be an $\tilde{N}_y$ -vector containing the outputs from the output neurons, and $\mathbf{w}$ be a vector containing all the weight parameters representing the connections in the neural network. The function $\tilde{\mathbf{y}} = \tilde{\mathbf{y}}(\tilde{\mathbf{u}}, \mathbf{w})$ mathematically represents a neural network. The definition of $\mathbf{w}$ and the manner in which $\tilde{\mathbf{y}}$ is computed from $\tilde{\mathbf{u}}$ and $\mathbf{w}$, determine the structure of the neural network.

2.2.1 Multilayer Perceptron Networks

An MLP network is one of the most popular types of neural networks that has been used in many different microwave applications [1][3]. In the MLP structure, the neurons are grouped into several layers. The neurons in the same layer have no connection to each other and the neurons between consecutive layers are fully connected. Typically, an MLP neural network consists of an input layer, one or more hidden layers, and an output layer, as shown in Figure 2.1. Suppose the total number of layers is L . The first layer is the input layer, the L^{th} layer is the output layer, and layers 2 to $L-1$ are hidden layers. Suppose the number of neurons in l^{th} layer is N_l , $l = 1, \dots, L$. Let w_{ij}^l represent the weight of the link between j^{th} neuron of $(l-1)^{th}$ hidden layer and i^{th} neuron of l^{th} hidden layer, w_{i0}^l and z_i^l be the bias parameter and the output of i^{th} neuron of l^{th} hidden layer, respectively. The vector of all trainable weights in an MLP is given by

$$\mathbf{w} = \begin{bmatrix} w_{10}^2 & w_{11}^2 & w_{12}^2 & \dots & w_{N_L N_{L-1}}^L \end{bmatrix}^T. \quad (2.1)$$

Let $\tilde{u}_i$ represent the i^{th} input parameter to the MLP. Then the following equations hold:

$$z_i^1 = \tilde{u}_i, \quad i = 1, \dots, N_1 \quad (2.2)$$

Figure 2.1: MLP neural network structure. Typically, an MLP network consists of one input layer, one or more hidden layers, and one output layer.

$$z_i^l = \sigma \left(\sum_{j=1}^{N_{l-1}} w_{ij}^l z_j^{l-1} + w_{i0}^l \right), \quad i = 1, \dots, N_l, \quad l = 2, \dots, L-1 \quad (2.3)$$

where $\sigma(\cdot)$ is the activation function of hidden neurons. The most commonly used form of $\sigma(\cdot)$ is the logistic sigmoid function given by

$$\sigma(\gamma) = \frac{1}{(1 + e^{-\gamma})}. \quad (2.4)$$

It is a smooth switch from 0 to 1 as γ varies from negative infinity to positive infinity.

Other possible candidates for $\sigma(\cdot)$ could be arctangent function

$$\sigma(\gamma) = \left(\frac{2}{\pi}\right) \arctan(\gamma) \quad (2.5)$$

or hyperbolic tangent (*tanh*) function

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

All of these activation functions are bounded, continuous, monotonic and continuously differentiable.

The outputs of MLP are produced by linear functions in the output neurons, and can be computed as

$$\tilde{y}_k = \sum_{i=1}^{N_{L-1}} w_{ki}^L z_i^{L-1} + w_{k0}^L, \quad k = 1, 2, \dots, N_L \quad (2.7)$$

where $\tilde{y}_k$ is the k th output of MLP. The universal approximation theorem of MLP was proved by Cybenko [75] and Hornik *et al.* [76] in 1989. According to the universal approximation theorem, provided sufficient hidden neurons, a three-layer perceptron network is theoretically capable of approximating an arbitrarily continuous, multi-dimensional, real, static nonlinear function to any desired accuracy. But in practice, how to choose the number of hidden neurons is still an open question. A definite answer is that the number of hidden neurons depends on the degree of nonlinearity and the dimension of the original problem. More nonlinear and more dimensional problems need

more neurons. But too many hidden neurons will result in lots of redundant free variables in $\mathbf{w}$, thus leading to the problem of overlearning. Several algorithms have been proposed to find a proper size of MLP network such as constructive algorithm [77] and network pruning [78].

In practice, three-layer or four-layer perceptron networks are commonly used for RF/Microwave applications. Intuitively, a four-layer perceptron network would perform better for tasks that have certain common localized behavior in different regions of the problem space. For the same task, a three-layer perceptron network may need lots of hidden neurons to repeat the same behavior in different input regions. In [79], it is pointed out that three-layer perceptron networks are preferred in function approximation where generalization capability is of a major concern. In [80], it was demonstrated that four-layer perceptron networks would have better performance in boundary definitions and therefore were favored for pattern classification tasks.

The MLP neural network can only represent the microwave problem through a process called training. The objective of training is to find an optimal set of weight parameters $\mathbf{w}$, such that the MLP outputs $\tilde{\mathbf{y}}$ closely matches the training data from the original device or circuit. The training data are in the form of sample pairs $(\tilde{\mathbf{u}}_k, \mathbf{d}_k)$, $k \in I_s$, where $\mathbf{d}_k$ is the desired outputs of the neural model for inputs $\tilde{\mathbf{u}}_k$, and I_s is an index set of training samples. During training, the neural network performance is evaluated by computing the difference between actual neural network outputs and desired outputs, for all the training samples. This difference, also known as the error (cost) function during training, is quantified by,

$$E(\mathbf{w}) = \frac{1}{2} \sum_{k \in I_S} \sum_{j=1}^{\tilde{N}_y} (\tilde{y}_j(\tilde{\mathbf{u}}_k, \mathbf{w}) - d_{jk})^2 \quad (2.8)$$

where d_{jk} is the j^{th} element of $\mathbf{d}_k$ and $\tilde{y}_j(\tilde{\mathbf{u}}_k, \mathbf{w})$ is the j^{th} neural network output for input $\tilde{\mathbf{u}}_k$. If we use only one training sample at a time to update $\mathbf{w}$, then a per-sample error function E_k can be defined as

$$E_k = \frac{1}{2} \sum_{j=1}^{\tilde{N}_y} (\tilde{y}_j(\tilde{\mathbf{u}}_k, \mathbf{w}) - d_{jk})^2. \quad (2.9)$$

During training, $\mathbf{w}$ is updated along the negative direction of the gradient of E or E_k , until E or E_k becomes small enough. In 1986, Rumelhart, Hinton, and Williams [72] proposed a systematic neural network training approach. One of the significant contributions in their work is the error BP algorithm. The following procedure describes how the error BP process can be used to compute the gradient information $\frac{\partial E_k}{\partial \mathbf{w}}$.

Using the definition of E_k in (2.9), the derivative of E_k with respect to the weight parameters of the l^{th} layer w_{ij}^l can be computed by simple differentiation as

$$\frac{\partial E_k}{\partial w_{ij}^l} = \frac{\partial E_k}{\partial z_i^l} \cdot \frac{\partial z_i^l}{\partial w_{ij}^l} \quad (2.10)$$

and

$$\frac{\partial z_i^l}{\partial w_{ij}^l} = \frac{\partial \sigma}{\partial \gamma} \cdot z_j^{l-1}. \quad (2.11)$$

The gradient $\frac{\partial E_k}{\partial z_i^l}$ can be initialized at the output layer as

$$\frac{\partial E_k}{\partial z_i^L} = (\tilde{y}_i - d_{ik}) \quad (2.12)$$

using the error between neural network outputs and desired outputs (training data).

Subsequent derivatives $\frac{\partial E_k}{\partial z_i^l}$ are computed by back-propagating this error from $l+1^{th}$

layer to l^{th} layer as

$$\frac{\partial E_k}{\partial z_i^l} = \sum_{j=1}^{N_{l+1}} \frac{\partial E_k}{\partial z_j^{l+1}} \cdot \frac{\partial z_j^{l+1}}{\partial z_i^l}. \quad (2.13)$$

For example, if the MLP uses sigmoid (2.4) as hidden neuron activation function, then we have

$$\frac{\partial \sigma}{\partial \gamma} = \sigma(\gamma) (1 - \sigma(\gamma)) \quad (2.14)$$

$$\frac{\partial z_i^l}{\partial w_{ij}^l} = z_i^l (1 - z_i^l) z_j^{l-1} \quad (2.15)$$

and

$$\frac{\partial z_i^l}{\partial z_j^{l-1}} = z_i^l (1 - z_i^l) w_{ij}^l. \quad (2.16)$$

Let δ_i^l represent the local gradient at i^{th} neuron of l^{th} layer, which is given by

$$\delta_i^L = (\tilde{y}_i - d_{ik}) \quad (2.17)$$

$$\delta_i^l = \left(\sum_{j=1}^{N_{l+1}} \delta_j^{l+1} w_{ji}^{l+1} \right) z_i^l (1 - z_i^l) \quad l = L-1, L-2, \dots, 2. \quad (2.18)$$

By using (2.17) and (2.18), which defines the BP process, the derivatives of E_k with respect to the weights can be computed by

$$\frac{\partial E_k}{\partial w_{ij}^l} = \delta_i^l z_j^{l-1} \quad l = L, L-1, \dots, 2. \quad (2.19)$$

Once the error derivatives are obtained by the BP process, the MLP training can be carried out by various gradient-based optimization methods to be presented in Chapter 2.3.3.

2.2.2 Radial Basis Function Networks

RBF networks are special types of feedforward neural networks that contain a single layer of hidden neurons with RBFs as the activation functions. RBF networks have been applied to various microwave modeling cases, at both device and circuit levels, including microwave transistors [81], RF and microwave power amplifiers [47], and high-speed I/O ports of integrated circuits (IC) [82][83].

A typical RBF neural network is shown in Figure 2.2. The RBF neural network has an input layer, a radial basis hidden layer, and an output layer. Commonly used radial basis activation functions are Gaussian and multi-quadratic. The Gaussian function is given by,

$$\sigma(\gamma) = \exp(-\gamma^2) \quad (2.20)$$

and the multi-quadratic function is given by,

$$\sigma(\gamma) = \frac{1}{(\varpi^2 + \gamma^2)^\alpha}, \quad \alpha > 0 \quad (2.21)$$

where ϖ is a constant.

Given the inputs $\tilde{\mathbf{u}}$, the total input to the i^{th} hidden neuron γ_i is given by,

$$\gamma_i^2 = \sum_{j=1}^{\tilde{N}_u} \left(\frac{\tilde{u}_j - \tilde{c}_{ij}}{\kappa_{ij}} \right)^2, \quad i = 1, 2, \dots, N_h \quad (2.22)$$

where N_h is the number of hidden neurons and the parameters $\tilde{c}_{ij}$ and κ_{ij} , $i = 1, 2, \dots, N_h$, $j = 1, 2, \dots, \tilde{N}_u$, are centers and standard deviations of radial basis activation functions, respectively. The output value of the i^{th} hidden neuron is $z_i = \sigma(\gamma_i)$, where $\sigma(\gamma)$ is an RBF. Finally, the outputs of the RBF network are computed from hidden neurons as,

$$\tilde{y}_k = \sum_{i=1}^{N_h} \tilde{w}_{ki} z_i, \quad k = 1, 2, \dots, \tilde{N}_y \quad (2.23)$$

where $\tilde{w}_{ki}$ is the weight of the link between i^{th} neuron of the hidden layer and k^{th} neuron of the output layer. Training parameters $\mathbf{w}$ of the RBF network include $\tilde{w}_{k0}, \tilde{w}_{ki}, \tilde{c}_{ij}, \kappa_{ij}$, $k = 1, 2, \dots, \tilde{N}_y$, $i = 1, 2, \dots, N_h$, $j = 1, 2, \dots, \tilde{N}_u$.

2.2.3 Time Domain Neural Networks

In this part, we review some special types of neural networks, *i.e.*, DNNs and RNNs, which are capable of representing time dynamics and are suitable for dynamic modeling of nonlinear circuits. Let M and K are the numbers of dynamic input and output signals of a nonlinear circuit, respectively. Let $\mathbf{u}$ be an M -vector representing time-domain input signals of a nonlinear circuit, *e.g.*, input voltages and currents, and $\mathbf{y}$ be a K -vector representing time-domain output signals of a nonlinear circuit, *e.g.*, output voltages and currents. These time-domain neural networks provide effective (reduced-order) means to represent the input-output relationship between $\mathbf{u}$ and $\mathbf{y}$.

Figure 2.2: RBF neural network structure.

Recurrent Neural Networks

Conventional feed-forward neural networks are well known for their learning and generalization capabilities. However, they are only capable of mapping static input-output relationships. To model nonlinear circuit responses in time-domain, a neural network that can include temporal information is necessary. RNN has been found to be a suitable candidate to accomplish this task. In the past, RNNs were successfully used in various engineering applications such as system control [84][85], speech recognition [70], etc.

For microwave dynamic modeling, the structure of a typical RNN is shown in Figure 2.3 [5]. Let $\bar{\mathbf{p}}$ represent an $\bar{N}_p$ -vector containing the time-invariant inputs of RNN. As

seen in Figure 2.3, the input layer of the RNN includes $\mathbf{u}$ and $\bar{\mathbf{p}}$ and the RNN outputs are the time varying signal $\mathbf{y}$. The first hidden layer of RNN contains buffered (time-delayed) history of $\mathbf{y}$ fed back from the output layer, buffered history of $\mathbf{u}$, and $\bar{\mathbf{p}}$. The second hidden layer contains neurons with sigmoid activation functions. The last layer of the RNN structure is the output layer whose outputs are linear functions of the responses of hidden neurons in the second hidden layer. As we can observe from Figure 2.3, the overall RNN structure, including both the feed forward part and feedback connections, realizes the following nonlinear dynamic relationship [5]

$$\mathbf{y}(k) = f_{FFNN}(\mathbf{y}(k-1), \dots, \mathbf{y}(k-n), \mathbf{u}(k-1), \dots, \mathbf{u}(k-m), \bar{\mathbf{p}}, \mathbf{w}). \quad (2.24)$$

Here the number of delay buffers m and n represent the order of dynamics in the RNN.

f_{FFNN} represents a static mapping function to be represented by a standard feedforward neural network (*e.g.*, MLP or RBF).

The RNN model can represent the nonlinear microwave circuit behavior only after it has been trained with proper training data. For nonlinear dynamic modeling, training data applied are a set of input and output waveforms of the original nonlinear circuit. They can be collected through simulations and/or measurements. Let $[T_1, T_2]$ be the time interval for input and output waveforms. Let $\mathbf{u}_d(t)$ and $\mathbf{y}_d(t)$ represent the input and output waveforms of the original nonlinear circuit. Let training data be represented by input-output waveform pair $(\mathbf{u}_d^j(t), \mathbf{y}_d^j(t))$, $T_1 \leq t \leq T_2$, $j = 1, 2, \dots, N_W$, where $\mathbf{u}_d^j(t)$ and $\mathbf{y}_d^j(t)$ are the j th input and output waveforms, respectively, and N_W is the total number of waveforms.

Figure 2.3: The discrete-time RNN based macromodel structure for nonlinear dynamic modeling [5]. τ represents the delay step.

The process of training is to adjust the trainable parameters of RNN such that the difference between RNN output response $y(t)$ and circuit waveform data $y_d(t)$ is minimized. Let N_t be the number of time samples in $[T_1, T_2]$. Let $y_i^j(k)$ be the i th output signal at the k th time sample in the j th trajectory of RNN and $y_{d,i}^j(k)$ be the corresponding

waveform sample from the original nonlinear circuit. For the k th training sample in the j th waveform pair $(\mathbf{u}_d^j(k), \mathbf{y}_d^j(k))$, the l_2 error of the RNN model is:

$$E_R^j(k) = \sum_{i=1}^K \frac{1}{2} (y_i^j(k) - y_{di}^j(k))^2. \quad (2.25)$$

And, the overall training error E_R , which is cumulated over all the time samples per waveforms as well as all the waveforms, is given by:

$$E_R = \frac{1}{2} \sum_{j=1}^{N_w} \sum_{i=1}^K \sum_{k=1}^{N_t} (y_i^j(k) - y_{di}^j(k))^2. \quad (2.26)$$

To enable efficient RNN training through gradient based optimization methods, it is necessary to calculate the derivatives of the error function with respect to each parameter in the RNN. As can be noticed from Figure 2.3, the outputs of the RNN model are fed back to the input layer through tapped delay lines. Specifically, output $\mathbf{y}$ is not only a function of input $\mathbf{u}$, but also a function of time-delayed histories of itself. Due to the recurrency between inputs and outputs, the conventional BP-based method is not suitable for the RNN training. A more advanced training scheme based on Back Propagation Through Time (BPTT) [5], [70] has been introduced, taking into account the effects of feedback connections in the RNN. A brief overview of the BPTT-based algorithm is summarized in the following to derive the derivative of the training error E_R [5].

To obtain derivatives of $E_R^j(k)$ with respect to weights $\mathbf{w}$ for training, we need $\frac{dy_i^j(k)}{d\mathbf{w}}$ which can be computed recursively from its history information. For $k = 1$,

initialize $\frac{dy_i^j(k)}{dw} = \frac{\partial y_i^j(k)}{\partial w}$, where the left hand side is the derivative of the RNN macromodel and $\frac{\partial y_i^j(k)}{\partial w}$ in the right hand side is the derivative of the feedforward neural network part in the macromodel, which can be computed by the normal BP. The subsequent derivative $\frac{dy_i^j(k)}{dw}$ for $k > 1$ can be computed by using the history of $\frac{dy_i^j(k)}{dw}$ through a process called recurrent BP. The recurrent BP includes two parts. First, normal BP is done through the feedforward neural network from the output layer to the input layer, obtaining the partial derivative $\frac{\partial y_i^j(k)}{\partial w}$, which represents the sensitivity of circuit output waveform with respect to macromodel internal parameters. Secondly, the derivative of $y_i^j(k)$ with respect to each neuron in the input layer is obtained by performing a further BP to the input layer of the feedforward neural network. This derivative represents the dependency of circuit output waveform between adjacent time points. Both these two derivatives contribute to the final value of $\frac{dy_i^j(k)}{dw}$ at a particular time point.

A special type of RNN structures that received much attention is Hopfield networks [86][87]. The overall structure of the network is composed of a single layer. Let $\bar{N}$ be the total number of neurons in a Hopfield network. The Hopfield network is constructed in such a way that each neuron, say the i^{th} neuron, can accept stimulus from (a) an external dynamic input $u_i, i = 1, 2, \dots, \bar{N}$; (b) output signals from other neurons $y_j, j = 1, 2, \dots,$

$\bar{N}$, $j \neq i$; (c) the output signal of the neuron itself, *i.e.*, y_i . The dynamics of the Hopfield network is governed by an energy function (*i.e.*, Lyapunov function), such that as the neural network dynamically changes with respect to time t , the total energy function of the network will be minimized.

Dynamic Neural Networks

For the purpose of circuit simulation, the most desirable format to describe nonlinear dynamics is that in the continuous-time domain. In theory this format best describes the fundamental essence of nonlinear behavior, and in practice it is flexible to fit most or all the needs for nonlinear circuit simulation. With these motivations, a technique based on the DNN in differential format [6], also known as differential DNN, was presented for large signal modeling of nonlinear microwave circuits and systems.

In time domain, the original nonlinear circuit can be generally described in state equation form as,

$$\begin{aligned}\dot{\bar{\mathbf{x}}}(t) &= \boldsymbol{\varphi}(\bar{\mathbf{x}}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) &= \boldsymbol{\psi}(\bar{\mathbf{x}}(t), \mathbf{u}(t))\end{aligned}\tag{2.27}$$

where $\bar{\mathbf{x}}$ is a vector containing state variables of the original circuit; $\boldsymbol{\varphi}$ and $\boldsymbol{\psi}$ represent nonlinear functions. In a modified nodal formulation [88], the state vector $\bar{\mathbf{x}}$ includes nodal voltages, currents of inductors, currents of voltage sources and charge of nonlinear capacitors.

At system level, such detailed state equations are too large, computationally expensive, and sometimes even unavailable. Therefore, a simpler (reduced order) model

approximating the same dynamic input-output relationships is highly required. Let n_D be the order of differential DNN representing the reduced order for original nonlinear circuit. Let each $\mathbf{v}_i$ be a K -vector, $i = 1, 2, \dots, n_D$. Let $\mathbf{f}_{ANN}$ represent an MLP neural network [1] with input neurons representing $\mathbf{y}$, $\mathbf{u}$, their derivatives $d^i \mathbf{y}/dt^i$, $i = 1, 2, \dots, n_D - 1$, and $d^k \mathbf{u}/dt^k$, $k = 1, 2, \dots, n_D$; and the output neuron representing $d^{n_D} \mathbf{y}/dt^{n_D}$. In [6], a differential DNN model was formulated as

$$\begin{aligned} \dot{\mathbf{v}}_1(t) &= \mathbf{v}_2(t) \\ &\vdots \\ \dot{\mathbf{v}}_{n_D-1}(t) &= \mathbf{v}_{n_D}(t) \\ \dot{\mathbf{v}}_{n_D}(t) &= \mathbf{f}_{ANN}(\mathbf{v}_{n_D}(t), \mathbf{v}_{n_D-1}(t), \dots, \mathbf{v}_1(t), \mathbf{u}^{(n_D)}(t), \mathbf{u}^{(n_D-1)}(t), \dots, \mathbf{u}(t)) \end{aligned} \quad (2.28)$$

where inputs and outputs of the DNN model are $\mathbf{u}(t)$ and $\mathbf{y}(t) = \mathbf{v}_1(t)$, respectively.

The overall DNN model (2.28) is in a standardized format for typical nonlinear circuit simulators. For example, the left-hand-side of the equation provides the charge or the capacitor part, and the right-hand-side provides the current part, which are the standard representation of nonlinear components in many harmonic balance (HB) simulators. In this way, DNN can provide dynamic current-charge parameters for general nonlinear circuits with any number of internal nodes in original circuit.

The order n_D (or the number of hidden neurons in $\mathbf{f}_{ANN}$) represents the effective order (or the degree of nonlinearity) of the original circuit that is visible from the input-output data. Therefore the size of the DNN reflects the internal property of the original circuit rather than external signals, and as such the model does not suffer from the curse of

dimensionality in multi-tone simulation. On the other hand, by adjusting order n_D and number of hidden neurons in f_{ANN} , the DNN model creation becomes much easier than conventional equivalent circuit based approaches [89], where manual trial-and-error may be needed to create/adjust the equivalent circuit topology and the nonlinear equation terms in it. A typical DNN structure for modeling large-signal behavior of nonlinear microwave circuits is shown in Figure 2.4, where $\bar{U}(\omega)$ and $\bar{Y}(\omega)$ represent the input and output spectrums, respectively, and matrix $B(\omega, t)$ represents the coefficients of Inverse Fourier Transform [90]. The DNN technique has been demonstrated to provide increased efficiency and flexibility in modeling and simulation of nonlinear microwave circuits such as RF amplifiers and mixers.

2.3 Training Algorithms

2.3.1 Training Objective

A neural network needs to be trained with corresponding device/circuit data before it can faithfully reproduce the response of the original circuit/device. For microwave modeling purposes, the development of neural models is essentially a curve-fitting problem in a multi-dimensional space. Let $\tilde{y} = \tilde{y}(\tilde{x}, \mathbf{w})$ represent the input-output relationship of the neural network. The objective of training is to find variables $\mathbf{w}$ such that the cost function $E(\mathbf{w})$, which represents the error between the neural network predictions and the training data, is minimized. The cost function $E(\mathbf{w})$ is usually a highly nonlinear function with respect to the adjustable parameter $\mathbf{w}$. Due to the complexity of $E(\mathbf{w})$,

iterative algorithms are often used to explore the parameter space efficiently. Various training algorithms ranging from gradient-based training algorithms to global training algorithms have been established. Every training algorithm has its unique way for updating the weights in neural networks. In the next few sections, we review some of the commonly used neural network training algorithms for RF and microwave applications [91]-[105].

Figure 2.4: A general DNN structure and its training process in time-domain using spectrum data, where $B^{(i)}$ is the time derivative operator [6].

2.3.2 Back Propagation Algorithms

BP scheme and the corresponding basic neural network training algorithm were developed by Rumelhart *et al.* in 1986 [71]. Nowadays, the BP scheme has become a standardized algorithm for training MLP neural networks. In this technique, the weights of the neural network weights $\mathbf{w}$ are updated along the negative gradient direction in the weight space. The update formula is given by [72]

$$\Delta \mathbf{w}_{now} = \mathbf{w}_{next} - \mathbf{w}_{now} = -\eta \left. \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} \right|_{\mathbf{w} = \mathbf{w}_{now}} \quad (2.29)$$

or

$$\Delta \mathbf{w}_{now} = \mathbf{w}_{next} - \mathbf{w}_{now} = -\eta \left. \frac{\partial E_k(\mathbf{w})}{\partial \mathbf{w}} \right|_{\mathbf{w} = \mathbf{w}_{now}} \quad (2.30)$$

where the step size η , called the learning rate, controls the step size of weight update. In (2.29), the weights are updated after all the training samples have been used to teach the neural network. This mode is called batch mode. On the other hand, another mode represented by (2.30) is called sample-by-sample mode where the weights are updated after each sample is presented to the network.

It is usually difficult to determine the learning rate η in the basic BP algorithm described above. The smaller η is, the more iterations are needed for training. But if η is too large in order to speed up the learning process, the training may be unstable due to weight oscillation [92]. A simple method to improve this situation is the addition of a momentum term to weight update formulae as proposed by [72]:

$$\Delta \mathbf{w}_{now} = -\eta \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} \bigg|_{\mathbf{w}=\mathbf{w}_{now}} + \alpha \Delta \mathbf{w}_{old} = -\eta \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} \bigg|_{\mathbf{w}=\mathbf{w}_{now}} + \alpha (\mathbf{w}_{now} - \mathbf{w}_{old}) \quad (2.31)$$

$$\Delta \mathbf{w}_{now} = -\eta \frac{\partial E_k(\mathbf{w})}{\partial \mathbf{w}} \bigg|_{\mathbf{w}=\mathbf{w}_{now}} + \alpha \Delta \mathbf{w}_{old} = -\eta \frac{\partial E_k(\mathbf{w})}{\partial \mathbf{w}} \bigg|_{\mathbf{w}=\mathbf{w}_{now}} + \alpha (\mathbf{w}_{now} - \mathbf{w}_{old}) \quad (2.32)$$

where constant α is the momentum factor which controls the influence of the last weight update direction on the current weight update, and $\mathbf{w}_{old}$ represents the last point of $\mathbf{w}$.

There have been a number of techniques that are aimed at further improving the robustness and efficiency of the standard BP algorithms. To accelerate the convergence of BP, an adaptation scheme is introduced in [93] that allows the learning rate and the momentum factor to be adaptively determined according to the training error. Another adaptive algorithm based on a few heuristics is proposed by [94]. Firstly, every weight in the neural network should have its own learning rate that is adaptive to the training process individually. Secondly, if a weight has kept changing in the same direction for several consecutive iterations, a larger step size needs to be applied. Otherwise, if a weight is in an oscillation state for several iterations, a smaller step size should be used. In [95], an enhanced BP algorithm is introduced where the learning rate of a weight in the neural network will be increased if the weight is large and is going to be larger. Each iteration of the training procedure involves updating the weights of the neural network along a computed direction with a certain step size. The standard BP algorithm uses learning rate as the step size. A more efficient way is to choose an optimal step size using linear search methods along the computed update direction. Examples in this category are

the line search based on quadratic model [96], and the line search based on linear interpolation [97].

2.3.3 Gradient-Based Optimization Methods

The BP algorithms described in the above section are simple to understand and relatively easy to implement. However, there is no sound theory on choosing the suitable values of learning rate and momentum factor. As a result, these algorithms often suffer from slow convergence or oscillation around the ravine area. Because supervised training of neural networks can be considered as a functional optimization procedure, nonlinear high-order optimization methods using gradient information can be used to improve the rate of convergence. Compared to the BP algorithms and their variants, these methods have a sound theoretical basis and guaranteed convergence for most smooth functions. Some of the early work in this area was presented in [98], by developing second-order-training algorithms for neural networks. In [91], various first- and second-order optimization methods for feedforward neural networks training are reviewed.

As mentioned before, the cost function $E(\mathbf{w})$ of training is defined as the summation of the square of difference between output of neural networks and training data. It is a parametric function depending on all the training parameters $\mathbf{w}$ in the neural network. Because $E(\mathbf{w})$ is generally complicated and nonlinear with respect to the weight parameters $\mathbf{w}$, iterative descent algorithms are usually used in the training process. In these algorithms, the next point of weights $\mathbf{w}_{next}$ is computed based on the current point of weights $\mathbf{w}_{now}$ and a direction vector $\mathbf{r}$,

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

The optimal step size η^* can be found by a line search along the $\mathbf{r}$ direction to minimize $E(\mathbf{w})$:

$$\eta^* = \min_{\eta > 0} E(\mathbf{w}_{now} + \eta \mathbf{r}) . \quad (2.34)$$

If the direction vector $\mathbf{r}$ is determined based on the gradient of the cost function $E(\mathbf{w})$, which is denoted by $\mathbf{g}$, the method is then called a gradient-based descent method. In the following part, we describe two commonly used gradient-based methods, *e.g.*, conjugate gradient method and quasi-Newton method.

Conjugate Gradient Method

Conjugate gradient method is originally derived from quadratic minimization. The direction vector $\mathbf{r}$ is chosen as the conjugate direction and is computed in a recursively way [99]. With an initial guess of weights $\mathbf{w}_{initial}$, the conjugate gradient method sets

initial gradient $\mathbf{g}_{initial} = \left. \frac{\partial E}{\partial \mathbf{w}} \right|_{\mathbf{w}=\mathbf{w}_{initial}}$, and direction vector $\mathbf{r}_{initial} = -\mathbf{g}_{initial}$. Then the vector

sequences $\mathbf{g}$ and $\mathbf{r}$ are constructed recursively by [99]

$$\mathbf{g}_{next} = \mathbf{g}_{now} + \lambda_{now} \mathbf{H} \mathbf{r}_{now} \quad (2.35)$$

$$\mathbf{r}_{next} = -\mathbf{g}_{next} + \gamma_{now} \mathbf{r}_{now} \quad (2.36)$$

$$\lambda_{now} = \frac{\mathbf{g}_{now}^T \mathbf{g}_{now}}{\mathbf{r}_{now}^T \mathbf{H} \mathbf{r}_{now}} \quad (2.37)$$

$$\gamma_{now} = \frac{\mathbf{g}_{next}^T \mathbf{g}_{next}}{\mathbf{g}_{now}^T \mathbf{g}_{now}} \quad (2.38)$$

$$\text{or,} \quad \gamma_{now} = \frac{(\mathbf{g}_{next} - \mathbf{g}_{now})^T \mathbf{g}_{next}}{\mathbf{g}_{now}^T \mathbf{g}_{now}} \quad (2.39)$$

where $\mathbf{H}$ is the Hessian matrix of the cost function $E(\mathbf{w})$. Equation (2.38) is called the Fletcher-Reeves formula and (2.39) is called the Polak-Ribiere formula.

To avoid the need of Hessian matrix for deriving the conjugate direction, an alternative way can be used to compute the conjugate direction. First, compute $\mathbf{w}_{next}$ by proceeding $\mathbf{w}$ from $\mathbf{w}_{now}$ along the direction $\mathbf{r}_{now}$ to the local minimum through line

minimization, and then set $\mathbf{g}_{next} = \left. \frac{\partial E}{\partial \mathbf{w}} \right|_{\mathbf{w}=\mathbf{w}_{next}}$. This $\mathbf{g}_{next}$ is then used as the result vector of

(2.35). In this way, only line minimization computation is needed to find the conjugate direction. The descent direction $\mathbf{r}$, i.e., the conjugate direction, can be accumulated without involving computationally intensive matrix inversions. As such, conjugate gradient methods are very efficient and scale well with the size of parameters in the neural network.

Quasi-Newton Training Algorithms

Quasi-Newton method employs second-order information about the cost function $E(\mathbf{w})$ to determine the weight update without requiring the knowledge of the Hessian matrix $\mathbf{H}$. Due to the proper approximation of inverse Hessian matrix, this method has faster convergence rate than conjugate gradient method. In this method, the descent direction $\mathbf{r}$ is computed by modifying the gradient vector $\mathbf{g}$ using a matrix $\mathbf{A}$, which is an estimation of the inverse of the Hessian matrix. The weight update formulae are given by [100]

$$\mathbf{w}_{next} - \mathbf{w}_{now} = -\eta \mathbf{r} = -\eta \mathbf{A}_{now} \mathbf{g}_{now} \quad (2.40)$$

$$\mathbf{A}_{now} = \mathbf{A}_{old} + \frac{\Delta \mathbf{w} \Delta \mathbf{w}^T}{\Delta \mathbf{w}^T \Delta \mathbf{g}} - \frac{\mathbf{A}_{old} \Delta \mathbf{g} \Delta \mathbf{g}^T \mathbf{A}_{old}}{\Delta \mathbf{g}^T \mathbf{A}_{old} \Delta \mathbf{g}} \quad (2.41)$$

or

$$\mathbf{A}_{now} = \mathbf{A}_{old} + \left(1 + \frac{\Delta \mathbf{g}^T \mathbf{A}_{old} \Delta \mathbf{g}}{\Delta \mathbf{w}^T \Delta \mathbf{g}}\right) \frac{\Delta \mathbf{w} \Delta \mathbf{w}^T}{\Delta \mathbf{w}^T \Delta \mathbf{g}} - \frac{\Delta \mathbf{w} \Delta \mathbf{g}^T \mathbf{A}_{old} + \mathbf{A}_{old} \Delta \mathbf{g} \Delta \mathbf{w}^T}{\Delta \mathbf{w}^T \Delta \mathbf{g}} \quad (2.42)$$

where

$$\Delta \mathbf{w} = \mathbf{w}_{now} - \mathbf{w}_{old} \quad \text{and} \quad \Delta \mathbf{g} = \mathbf{g}_{now} - \mathbf{g}_{old} \quad (2.43)$$

By using the update formula in (2.41) we obtain the Davidon-Fletcher-Powell algorithm. On the other hand, the update formula in (2.42) gives the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [100].

Assume there are N_{wp} weight parameters in the neural network structure. For standard Quasi-Newton methods, N_{wp}^2 storage space and a line search are required to maintain an approximation of the inverse Hessian matrix and calculate a reasonably accurate step size. In limited-memory or one-step BFGS method [101], the inverse Hessian approximation is reset to the identity matrix after every iteration. The space of storing matrices is consequently saved. When N_{wp} becomes large, the consumption of storage memory and computation on line search become the bottleneck [102] for efficiency and scaling. In [102], a second-order learning algorithm is proposed so that the decent direction $\mathbf{r}$ is computed on the basis of a partial BFGS update with less memory. A reasonably accurate step size is efficiently calculated as the minimal point of a second-order approximation of the cost function. In [103], various approaches to the parallel

implementation of second-order gradient-based MLP training algorithms, such as full and limited memory BFGS algorithms, were presented. Through the estimation of inverse Hessian matrix, quasi-Newton has faster convergence rate than conjugate gradient method.

2.3.4 Global Optimization Methods

Gradient-based training methods share the common drawback that the final solution after convergence is only a local minimum. To allow the training algorithms to escape from local minima and converge to the global minimum, some training methods using random optimization techniques have been proposed. These methods are characterized by a random search element in the training process. Two representative algorithms in this class are the simulated annealing algorithm and the genetic algorithm. In [73], the simulated annealing algorithm is described in such a manner that analogs the annealing process of atoms inside metal. The random parameter in the optimization process is controlled by a parameter called temperature, which determines the search range a local minimum can jump out. In [74], the genetic training algorithm is introduced to train the neural network. The genetic algorithm simulates the biological evolution and the weights of the neural network are coded as chromosomes. The local convergence of the algorithm is achieved by crossover operation and selection procedure, and the global search is accomplished by the random mutation. In [104], noise is added to the weights during training based on a Langevin updating rule. A stochastic minimization algorithm is developed in [105] for neural networks training through Iterated Adaptive Memory Stochastic Search.

Because of the random search during the neural network training, training methods based on global optimizations method could be very slow to converge. A hybrid method that combines the conjugate gradient method and random optimization was proposed in [96]. During the training with conjugate gradient method, if a flat error surface is encountered, the training algorithm switches to the random optimization method. After the training escapes from the flat error surface, it switches back to the conjugate gradient algorithm.

2.4 Existing Transient Behavioral Modeling Approaches for Nonlinear Circuits

With the ever-increasing of the signal speed and complexity in today's high-speed VLSI system, minimizing SI effects such as signal delays, crosstalks, and ground-bounce noises has become a challenging task [106]. Efficient and accurate representation of the SI effects is a key step towards ensuring the reliable signal transmissions in the high-speed signal propagation path, including transmission lines and nonlinear digital I/O buffers. The nonlinear I/O buffer, *i.e.*, nonlinear drivers and receivers, are important nonlinear circuit blocks for high-speed design that are used to drive and load the interconnect structures. Recently, the behavioral modeling of nonlinear digital buffers has become an important topic. Behavioral models are simplified representations of the essential nonlinear behavior of the detailed transistor-level description of the buffer. When used in the system level SI analysis, these behavioral models can lead to faster central processing time (CPU) time and less memory consumption, compared to the detailed transistor-level

models. In this section, we provide an overview of the recent approaches for behavioral modeling of the nonlinear I/O buffers.

2.4.1 Equivalent-Circuit-Based Approaches

High-speed I/O buffers can be most accurately represented by SPICE transistor-level models [18], which are often directly extracted from the detailed circuit design of the buffer. The detailed transistor-level model can accurately account for the second-order effects (*e.g.*, ground bounce) and is suitable for small-scale simulations of interconnects with a few buffers. However, because transistor-level models contain detailed process and design information, they tend to be slow and are not well suited for the simulation of the complex board consisting of a large number of nets and drivers. Another drawback of the transistor-level model is that it reveals the sensitive information of the design and manufacturing process. Due to this reason, it is not easy to obtain the transistor-level models from the manufacturers.

The input/output buffer information specification (IBIS) [107] is a behavioral modeling method for representing the analog behavior of digital IC buffers aimed at providing fast and accurate simulations for the SI effects transmission lines. It was originally developed in late 1990s by the semiconductor industry to provide input-output information of ICs to circuit simulators, without having to revealing the proprietary process and design information. IBIS model adopts a standard parsed file format, *i.e.*, a table-based format, to store only the essential behavioral information about input-output characteristics of ICs. Due to this standard format, IBIS model file does not contain any

confidential information about the original circuit to be modeled. Another advantage is that IBIS models tend to run much faster than the detailed transistor-level model since it is a behavioral-level representation of the original device. Furthermore, this standard is widely supported by most electrical design automation (EDA) vendors and semiconductor manufacturers, leading to a large set of model libraries of commercial devices.

Figure 2.5 shows the basic structure of an IBIS to model the behavior of an output buffer [107]. As can be noticed, there are five basic component used in the model structure. Elements 1 and 2 are the pull-down and the pull-up devices, respectively. Element 3 contains the protection device that describes the effects of ground and power clamp diodes where the C_{comp} is the input capacitance of the input pin. Element 4 contains the ramp time for the pull-up and pull-down devices. Element 5 is the package parasitics where L_{pkg} , R_{pkg} , and C_{pkg} are the package parameters. Each element of the model is represented by a data-based table in the IBIS ASCII file. Each table begins with a keyword in bracket, such as [Pull_down], which indicates the nature of the table that proceeds. This format makes it easy for the simulator to either directly use the data in the ASCII file or translate the data into a format used by a specific software.

IBIS models can be generated either by measurement, which requires having a well-controlled environment and measurement devices, or through multiple SPICE simulations to obtain the necessary voltage/current tables and voltage transition tables. Once all the necessary voltage/current data and other parameters are obtained, one can create the IBIS ASCII file using the file format defined in the IBIS standard. Next, the model file is

checked by using the “Golden Parser” to detect any possible syntax errors. If the model passes this check, the model is then incorporated into an EDA tool and validated for accuracy by using the measured or simulated data from the actual device. If the model has good correlation with actual device, the model is ready to be used for the system-level SI simulations.

Figure 2.5: The basis structure of the IBIS model for an output buffer.

2.4.2 ANN-Based Approaches

In recent years, ANN-based techniques have emerged as an attractive alternative approach to the equivalent circuit based approach, such as IBIS model. ANNs are information-processing systems that are inspired by the abilities of the human brains to learn from observation and generalize by abstraction [1]. ANNs can be trained by an optimization process called training to learn the input-output behavior of the nonlinear I/O buffer. The trained ANN models can then be used in SI-based analysis and design for quick and accurate estimation of the performance of the high-speed

digital systems. In this subsection, several ANN-based techniques for modeling nonlinear I/O buffers are reviewed.

In [82], a parametric model based on the theory from input-output system identifications has been introduced. This method is a useful complement of the conventional IBIS models, taking into account more physical effects of the I/O drivers and maintaining higher-level of accuracy and efficiency. In this method, the nonlinear relationship between the currents and voltages of the driver output port is represented by a piece-wise linear formulation with RBF neural networks [82]

$$i_d(k) = w_1(k)f_1(\Theta_1, \mathbf{x}_n(k)) + w_2(k)f_2(\Theta_2, \mathbf{x}_n(k)), n=1,2 \quad (2.44)$$

$$f_n(\Theta_n, \mathbf{x}_n(k)) = \sum_{j=1}^{N_h} \tilde{w}_{nj} \exp(-|\mathbf{x}_n(k) - \mathbf{c}_{nj}|^2 / \beta_j^2), n=1,2 \quad (2.45)$$

$$\Theta_n = [\tilde{w}_{n1}, \dots, c_{n1}^T, \dots, \beta_1, \dots]^T, n=1,2 \quad (2.46)$$

where $f_1(\Theta_1, x(k))$ and $f_2(\Theta_2, x(k))$ are the Gaussian RBF submodels that describe the port nonlinear behavior of the driver when the drive inputs are at HIGH and LOW logic states, respectively, and i_d represents the driver output port current. The vector $\mathbf{x}_n$ is called the regressor vector that collects time-delayed signals of port voltage v_d and the current of the submodel at the past r time instants, *i.e.*,

$$\mathbf{x}_n(k) = [i_n(k-1), \dots, i_n(k-r), v_d(k-1), \dots, v_d(k-r)]^T, n=1,2. \quad (2.47)$$

As can be noticed from (2.45) and (2.46), the submodel functions f_1 and f_2 realize a dynamic relationship by relating the history of the model input and output signals to the

present output signals through the sums of Gaussian RBF functions. Each Gaussian basis function is defined by its position in the regressor space $\mathbf{x}_n$ (center $\mathbf{c}_{nj}$) and by its spreading (scale parameter β_j). Here, N_h is the total number of nonlinear basis functions used in each RBF submodel. Θ_1 and Θ_2 denote the internal weight parameters of the RBF functions, which are estimated by standard methods, such that the model responses can fit those of the reference model for the original circuit under consideration. In the piece-wise model (2.44), $w_1(k)$ and $w_2(k)$ are time varying weight coefficients that act as the switches for submodels f_1 and f_2 , depending on the transitions of the input port state. The overall model can be generated from the input-output transient waveforms at the device ports following a systematic procedure. A detailed description of the modeling estimation process can be found in [82]. This technique has been used for parametric modeling of single-ended CMOS buffers [82], as well as the low voltage differential signaling buffers [108]. An extended formulation of this approach for modeling the effects of device temperatures and nonlinear power supplies can be found in [83].

As a related technique to the piece-wise formulation with RBF functions, RNN structures are exploited in [109], for modeling the highly nonlinear driver circuits. RNN models are a generic type of discrete-time neural network that can be used to effectively model the nonlinear patterns with memory or feedback. In this case, the nonlinear submodels f_1 and f_2 in the piece-wise linear formulation in (2.44) are represented by RNN functions which contains a three-layer MLP neural network [109]. The training of the RNN is carried out by using the BPTT algorithm to estimate the internal weight

parameters of the RNN. By taking into account the effect of the feedback of RNN outputs during training, the BPTT algorithm can lead to more robust RNN models for representing the highly nonlinear buffer behavior. In [109], it is demonstrated that the RNN model maintains a similar accuracy as the detailed transistor level but being 6-7 times faster in terms of computational time.

2.4.3 Nonlinear Model Order Reduction Techniques

Recently, there have been some growing interests in automated CAD approaches for extracting accurate and quick macromodels of nonlinear circuit blocks from their detailed circuit equations for SPICE-level representations [110]-[116]. These techniques are known as nonlinear model order reduction (MOR) techniques, which are extensions to the well-established reduction-based methods for linear systems, such as PRIMA [117] and Krylov-subspace methods [118].

Nonlinear circuit-reduction techniques based on Krylov-subspace methods [110] or congruent transformation techniques [111] have been proposed for modeling a certain type of nonlinear circuits, which includes high-speed interconnect networks. An important feature of these techniques is that the order of the original system can be reduced to a user specified number q while the first q derivatives of the time-response of the original system are retained. Specifically, the nonlinear state-based dynamic equations of the original large nonlinear circuit are first formulated in time domain. Taylor expansion is then applied to time-related variables in both sides of the nonlinear circuit equations with respect to time, which includes system states and input signals. The Krylov subspace of the original nonlinear system is formulated as a matrix by putting

together the first q Taylor coefficients of all the system states. The reduced system is composed of a set of new states, which is computed through a congruent transformation of the original system states using the Q matrix resulted from QR decomposition. The new system after reduction is able to preserve theoretically the first q orders of the original nonlinear system. By replacing the original nonlinear circuit with the new reduced-order system, a significant improvement of the speed in circuit simulation is demonstrated.

For an important class of nonlinear circuit where the nonlinearities can be sufficiently approximated by polynomials, techniques based on Volterra series expansions have been proposed to obtain reduced-order nonlinear models. These approaches use the Krylov-subspace methods to generate the linearized system and further incorporate high order non-linearities based on distortion inputs. As an example, in [112], a compact nonlinear MOR method has been proposed, targeted to the MOR of time-varying weakly nonlinear circuits that can be characterized by low-order Volterra series. The automatically extracted macromodels capture not only the first-order (linear) system properties, but also the important second-order effects of interest that cannot be neglected for a broad range of applications. Unlike the existing projection-based reduction methods for weakly nonlinear systems, this technique begins with the general matrix-form Volterra nonlinear transfer functions to derive a set of minimum Krylov subspaces for order reduction. Moment matching of the nonlinear transfer functions is performed by projecting the original system onto this set of minimum Krylov subspaces, leading to a significant reduction of model size. To improve the model compactness for nonlinear system

reduction, a multipoint version of the compact nonlinear MOR method is also presented. Results have shown that this technique provides an effective MOR framework for both time-invariant and time-varying weakly nonlinear systems.

For general nonlinear system with strong nonlinearities, a trajectory-based piecewise-linear (TPWL) techniques was pioneered in [113] and followed by several enhancements [114][115]. The basic idea of the TPWL approach is to represent the original nonlinear system with a piecewise-linear system and subsequently reduce each of the pieces with Krylov subspace projection methods. Rather than approximating the individual components as piecewise-linear and subsequently composing the resulting components to make a reduced-order system, a small set of linearizations about the state trajectory, which is the response to a “training input”, is generated. A procedure for a posteriori estimation of the simulation error is also presented, which may be used to determine the accuracy of the extracted TPWL reduced-order models. Through the numerical examples of nonlinear circuits, it is shown that this MOR technique is computationally inexpensive, and that the macromodels can be constructed “on the fly” to accelerate simulation of the system response.

A recent work in [116] presents a hybrid method that combines the trajectory-based techniques and the weakly nonlinear MOR algorithms. It is known that the weakly polynomial macromodels accurately capture the small signal distortions around the expansion point, but they become rapidly inaccurate for large input excitations. The TPWL-based macromodels, on the other hand, are well-suited for capturing strong nonlinearities in wider range; however, its accuracy in representing weak nonlinearities

could be limited. The hybrid method, namely piece-wise-polynomial method, utilizes the framework of the TPWL methodology, but approximates each region with higher order (tensor) polynomials instead of using purely linear representations. In this way, the piece-wise-polynomial method is capable of delivering a unified macromodel that can represent strong and weakly nonlinearities simultaneously. As such, the generated macromodels can not only reproduce small-signal distortion and intermodulation properties well, but also retain fidelity in large-signal transient analyses, such as the SI simulations of high-speed I/O buffers. These reduced models can be used to achieve fast system-level simulation using a variety of time- and frequency-domain analyses (such as direct current (dc), transient, HB, etc.).

2.5 Summary

In this chapter, a comprehensive review of the ANN-based techniques for RF and microwave modeling and design has been given. Different neural network structures that are commonly used in nonlinear microwave modeling are described. In addition, gradient-based and global training algorithms for developing ANN models are systematically described. The introduction of these fundamental concepts serves as the foundation of the advanced ANN-based model formulations and training approaches to be presented.

Conventional methods for transient behavioral modeling of high-speed nonlinear circuits, such as high-speed I/O buffers, have been presented. Detailed transistor-level models capture the full analog nonlinear behavior, but are too computationally demanding for high-level design. Equivalent-circuit-based approaches provide great

efficiency but offer only limited accuracy. The MOR-based methods can produce reduce-order nonlinear models with high-level of accuracy and efficiency, but usually it has to rely on the analytical formulation of the original circuits. The ANN-based approaches have shown great potentials as fast and accurate alternatives to these conventional approaches, and can be applied even if the internal details of the nonlinear circuits are not available. Motivated by the attractive features of ANNs, in the next few chapters, several advanced ANN techniques are presented for transient modeling and simulation of nonlinear circuits for high-speed applications.

Chapter 3: Nonlinear Transient Modeling and Stability Analysis for High-Speed Applications Exploiting State-Space Dynamic Neural Networks

In this chapter, one of the major contributions of the thesis, a transient behavioral modeling technique for nonlinear circuits, namely the SSDNN technique, is presented. The proposed SSDNN technique directly addresses the transient modeling of the nonlinear circuits, which is of great importance for the SI analysis in high-speed digital design. Through a systematic training algorithm, the SSDNN model can be developed from the input-output waveforms of the original circuits, and the resulting accurate model can be subsequently used to enable efficient transient simulation and SI analysis at the system level. Another important contribution is stability criteria of the SSDNN model derived through Lyapunov methods, which facilitate convenient stability verifications of the trained SSDNN models. In addition, a further advancement is achieved where an advanced constrained training approach is developed, enabling the trained SSDNN model to meet both accuracy and stabilities requirements.

3.1 Introduction

Over the past decade, ANNs have emerged as useful tools for modeling and design optimization in RF and microwave CAD [1]. A recent trend in this area is to utilize ANN to enhance the CAD of nonlinear microwave devices and circuits [8][9][50][6][119]. In [6], a DNN technique was introduced for modeling the steady-state behavior of nonlinear circuits such as amplifiers and mixers. The DNN models can be developed directly from input-output data without having to rely on circuit internal details. The DNN approach maintained the ANN speed and accuracy capabilities with extra flexibilities for steady-state HB simulations in both single-tone and multi-tone cases. A further study in [50] provided the theoretical basis for the order determination of DNN formulations. An ADNN approach was developed in [8][9], paving the way for training neural models for nonlinear transient applications. Most recently, the DNN concept has also been considered for developing HEMT models from large signal measurements [119].

A recent work to be presented in this chapter addresses a new area of DNN applications, *i.e.*, neural-based transient modeling of nonlinear circuits such as nonlinear drivers and receivers in high-speed VLSI interconnect analysis. With the continuous increase of signal speed and frequency, the SI in VLSI packages becomes important. Fast and accurate representations of the nonlinear analog behavior of driver/receiver buffers are the key to the success of SI-based design of high-speed interconnects with nonlinear terminations [120]-[123]. There are two types of conventional methods in popular use today: detailed transistor level modeling approach such as SPICE models, and empirical behavioral modeling approach such as the IBIS models [107]. The former approach

offers better accuracy but becomes increasingly inefficient due to the growing size and complexity of IC designs. The IBIS model is a data-based model which is fast but usually has only limited accuracy and flexibility. Therefore developing efficient buffer models for transient applications has become an important topic [122][123]. In this chapter, we explore the accuracy and flexibility of neural networks and further extend the DNN concept to the transient modeling of high-speed IC drivers and receivers.

One of the most important aspects of nonlinear transient modeling is the model stability. It is a fundamental requirement for ensuring the model reliability in circuit simulations. A non-stable model is non-practical and can be potentially detrimental to the overall system simulation. In general, determining the stability for nonlinear systems is a challenging task. In the neural network community, GAS and global exponential stability have been studied for some special classes of dynamic networks, *e.g.*, Hopfield neural networks [124][125], RNNs [126][127], and discrete-time state-space neural networks [128]. Recently, the GAS has been also studied for RNNs with time delays [129]-[131]. Various stability criteria are introduced in the literature in terms of linear matrix inequalities [132]. These existing methods are useful for applications such as neural optimizations, signal processing, and classifications. However, for the DNN microwave behavioral models, these existing stability criteria cannot be directly used because of the differences in the model structures, where the DNNs are structured for efficiently addressing analog microwave modeling. So far for DNN behavioral modeling [6], stability analysis methods are still not yet available.

In this work, a nonlinear SSDNN technique is presented for modeling nonlinear transient behavior. The SSDNN formulation expands the existing DNN structure into a general and flexible formulation for transient-oriented nonlinear modeling and design. This SSDNN nonlinear model can be developed directly from the input and output transient signals without requiring any detailed knowledge of original nonlinear circuits. Training of the SSDNN is realized in the time-domain by employing a new transient adjoint technique for fast and accurate derivative calculation. The exact circuit representation of the SSDNN model is also presented to allow the model to be conveniently incorporated into general SPICE-like circuit simulators.

In addition, for the first time, stability analysis of neural models is studied for behavioral modeling of nonlinear circuits using the SSDNN modeling framework. A set of stability criteria is derived to enable the verifications of both local and global stabilities of the SSDNN model upon the completion of model training. Furthermore, a new training algorithm, which incorporates the proposed stability criteria into model training as constraints, is developed such that the trained SSDNN model is guaranteed to be globally asymptotically stable. The final SSDNN model, which are both accurate and globally stable, can be implemented into general SPICE-like circuit simulators delivering fast and accurate transient responses.

3.2 Proposed SSDNN Modeling Technique

There are two main categories of formulations to describe nonlinear dynamic systems: one is input-output format and the other is state-space format. The existing DNN approaches have been based on the input-output format. Here we apply state-space concept to neural networks for nonlinear circuit modeling. Based on combining state-space concept and continuous RNN method [133], we propose the following SSDNN nonlinear model:

$$\begin{cases} \dot{\mathbf{x}}(t) = -\mathbf{x}(t) + \tau \mathbf{f}_{ANN}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}) \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) \end{cases} \quad (3.1)$$

with initial condition $\mathbf{x}(0) = \mathbf{x}_0$, where $\mathbf{u}(t) \in \mathbb{R}^M$ and $\mathbf{y}(t) \in \mathbb{R}^K$ are the circuit input and output waveforms, respectively, as defined in Chapter 2.2.3, $\mathbf{x}(t) \in \mathbb{R}^N$ is the vector of the state variables, and N denotes the dimension of state-space, or the order of the model. Here $\mathbf{f}_{ANN}$ is a vector function representing a feed-forward MLP neural network [1], which has $N+M$ input neurons and N output neurons with weight parameters denoted as $\mathbf{w}$ and a constant scaling parameter τ . $\mathbf{C} \in \mathbb{R}^{K \times N}$ is the output measurement matrix, which maps state-space into output-space.

There are several advantages of the SSDNN technique for modeling nonlinear circuit. The model accuracy can be proven through the universal approximation property [1] of MLP neural networks. Furthermore, this new format advocates first-order differential equation format as opposed to the direct high-order format in DNN-based approaches [6], thus enhancing the robustness of the model in dealing with noisy training data. In addition, the existing DNN can be indirectly derived as a

special case from this new SSDNN formulation. In this sense, the SSDNN is a progressive generalization of the existing DNN.

The discrete-time counterpart of the SSDNN, namely, state-space neural networks, has been studied, *e.g.*, [128], [134]-[137], and used in various applications of nonlinear system identification and control. If such discrete time structure is used for circuit modeling, certain approximations need to be made when incorporating the model into circuit simulators. Our SSDNN provides an attractive alternative in this aspect because the SSDNN implementation in circuit simulation can be achieved exactly via circuit representations [10].

The SSDNN also provides further advances in analog circuit modeling over several other existing discrete-time neural network architectures, which were mostly developed from signal processing and control applications. An example of such existing methods is the Elman recurrent neural network [138], which contains a feedback connection over each hidden neuron. Another example of such architectures is to extend feedforward MLP structure to include global feedbacks and delays from MLP outputs to inputs [84], which will be referred to as globally recurrent neural network (GRNN) in this work. Another way to have discrete time recurrency is locally recurrent neural network where the classical synapse is replaced by a finite impulse response filter or an infinite impulse response filter [139]. The fundamental difference between the SSDNN and the above neural networks is that the SSDNN is formulated in continuous time while the other structures are in discrete time. To implement discrete-time models into analog

equivalent circuits for circuit simulation, certain approximations are needed. On the other hand, the SSDNN has an exact circuit implementation for nonlinear time-domain simulation [10]. By using recurrence over MLP inputs and outputs instead of recurrence over individual neurons, the SSDNN also offers additional flexibilities in modeling nonlinear analog behavior of the circuit.

3.3 Stability Analysis of SSDNN models

3.3.1 SSDNN Parameters for Stability Analysis

We develop stability criteria with respect to the neural network structures and parameters in SSDNN formulations. Let $\mathbf{f}_{ANN}$ in (3.1) represent a commonly used three-layer MLP neural network. Let N_h be the number of hidden neurons in the MLP. To facilitate stability derivation, we split the internal weights of $\mathbf{f}_{ANN}$ into three groups, described in matrix form as

$$\mathbf{W}_u = \begin{bmatrix} w_{11}^2 & w_{12}^2 & \cdots & w_{1M}^2 \\ w_{21}^2 & w_{22}^2 & \cdots & w_{2M}^2 \\ \vdots & \vdots & \cdots & \vdots \\ w_{N_h 1}^2 & w_{N_h 2}^2 & \cdots & w_{N_h M}^2 \end{bmatrix} \quad (3.2)$$

which represents the connection weights between external input signals and all the hidden neurons,

$$\mathbf{W}_s = \begin{bmatrix} w_{1,M+1}^2 & w_{1,M+2}^2 & \cdots & w_{1,M+N}^2 \\ w_{2,M+1}^2 & w_{2,M+2}^2 & \cdots & w_{2,M+N}^2 \\ \vdots & \vdots & \cdots & \vdots \\ w_{N_h,M+1}^2 & w_{N_h,M+2}^2 & \cdots & w_{N_h,M+N}^2 \end{bmatrix} \quad (3.3)$$

which represents the connection weights between state variables and all the hidden neurons, and

$$W_o = \begin{bmatrix} w_{11}^3 & w_{12}^3 & \cdots & w_{1N_h}^3 \\ w_{21}^3 & w_{22}^3 & \cdots & w_{2N_h}^3 \\ \vdots & \vdots & \cdots & \vdots \\ w_{N1}^3 & w_{N2}^3 & \cdots & w_{NN_h}^3 \end{bmatrix} \quad (3.4)$$

which represents the connection weights between all the hidden neurons and outputs of f_{ANN} . Figure 3.1 shows the structure of a three-layer MLP neural network together with the corresponding locations of weight matrices W_u , W_s and W_o . For the convenience of stability analysis, we rewrite (3.1) in the vector form as

$$\dot{\mathbf{x}}(t) = -\mathbf{x}(t) + \tau W_o \boldsymbol{\sigma}(W_u \mathbf{u}(t) + W_s \mathbf{x}(t)) \quad (3.5)$$

where $\boldsymbol{\sigma}(\cdot) = [\sigma_1(\cdot) \ \sigma_2(\cdot) \ \dots \ \sigma_{N_h}(\cdot)]^T$ is a vector of nonlinear activation functions for hidden neurons. We follow the assumptions that the activation functions belong to a class of functions that are bounded, smooth, and monotonically increasing. Examples of such functions include commonly used neuron activations functions such as sigmoid functions, arctangent functions and *tanh* functions.

Figure 3.1: A three-layer MLP neural network used in the SSDNN. The MLP connection weights are grouped into three parts, as defined by W_u , W_s and W_o . The MLP input neurons include SSDNN overall input signals $u_i(t)$ and state variables $x_i(t)$. The MLP output neurons are f_{ANN} in state-space equations.

3.3.2 Stability Analysis via Lyapunov Method

In this subsection, we perform stability analysis of the SSDNN model via Lyapunov's indirect and direct methods [140]. The Lyapunov method has been widely used for deriving the stability conditions for RNNs with and without time delays, *e.g.* [124]-[127], [129], and [130]. In [125], the global exponential stability

of Hopfield neural networks is studied by means of Lyapunov method. Through another form of Lyapunov energy functions, several sufficient conditions are derived in [130] for RNNs with time varying delays and Lipschitz continuous activation functions. Recently, this method has also been used in the study of global robust stability of neural network with time delays [129]. These works have demonstrated the applications of the Lyapunov method for a general category of RNN structures where the feedback connection is applied locally at each neuron. However for our SSDNN structure where the global feedbacks from MLP outputs to inputs are present, these existing results can not be directly used. By following the ideas in [126][127], here we apply the Lyapunov method for the stability analysis of the SSDNN and derive sufficient criteria for investigating both local and global stabilities.

Let the state vector $\mathbf{x}_e$ be the equilibrium point of the SSDNN model that satisfies $\dot{\mathbf{x}}_e = \mathbf{0}$, *i.e.*,

$$-\mathbf{x}_e + \tau \mathbf{W}_o \sigma(\mathbf{W}_u \mathbf{u} + \mathbf{W}_s \mathbf{x}_e) = \mathbf{0}. \quad (3.6)$$

To examine stability, we choose $\mathbf{u}(t)$ in (3.6) to be a constant external input. Let $\mathbf{u}_0$ be the constant value for $\mathbf{u}(t)$, $t > 0$. By performing the variable transformation $\mathbf{z} = \mathbf{x} - \mathbf{x}_e$, we can rewrite (3.6) in terms of $\mathbf{z}$ as

$$\dot{\mathbf{z}}(t) = -\mathbf{z}(t) + \tau \mathbf{W}_o \mathbf{f}(\mathbf{W}_s \mathbf{z}(t)) \quad (3.7)$$

where $\mathbf{f}(\mathbf{W}_s \mathbf{z})$ is a vector function defined as

$$\mathbf{f}(\mathbf{W}_s \mathbf{z}) = \begin{bmatrix} f_1(\mathbf{W}_s^{(1)} \mathbf{z}) & f_2(\mathbf{W}_s^{(2)} \mathbf{z}) & \dots & f_{N_h}(\mathbf{W}_s^{(N_h)} \mathbf{z}) \end{bmatrix}^T \quad (3.8)$$

and

$$\begin{aligned} f_i(\mathbf{W}_s^{(i)} \mathbf{z}) &= \sigma_i(\mathbf{W}_s^{(i)}(\mathbf{x}_e + \mathbf{z}) + \mathbf{W}_u^{(i)} \mathbf{u}_0) \\ &\quad - \sigma_i(\mathbf{W}_s^{(i)} \mathbf{x}_e + \mathbf{W}_u^{(i)} \mathbf{u}_0) \end{aligned} \quad (3.9)$$

where $\mathbf{W}_s^{(i)}$ and $\mathbf{W}_u^{(i)}$ represent the i^{th} row of $\mathbf{W}_s$ and $\mathbf{W}_u$, respectively. Because of the monotonically increasing property of σ , we have

$$\begin{aligned} \sigma_i(\mathbf{W}_s^{(i)}(\mathbf{x}_e + \mathbf{z}) + \mathbf{W}_u^{(i)} \mathbf{u}_0) &> \\ &\quad \sigma_i(\mathbf{W}_s^{(i)} \mathbf{x}_e + \mathbf{W}_u^{(i)} \mathbf{u}_0), \text{ if } \mathbf{W}_s^{(i)} \mathbf{z} > 0 \\ \sigma_i(\mathbf{W}_s^{(i)}(\mathbf{x}_e + \mathbf{z}) + \mathbf{W}_u^{(i)} \mathbf{u}_0) &< \\ &\quad \sigma_i(\mathbf{W}_s^{(i)} \mathbf{x}_e + \mathbf{W}_u^{(i)} \mathbf{u}_0), \text{ if } \mathbf{W}_s^{(i)} \mathbf{z} < 0 \end{aligned} \quad (3.10)$$

In addition, since the activation functions σ are smooth switch functions, $f_i(\mathbf{W}_s^{(i)} \mathbf{z}) / (\mathbf{W}_s^{(i)} \mathbf{z})$ is non-negative and bounded, *i.e.*,

$$0 \leq f_i(\mathbf{W}_s^{(i)} \mathbf{z}) / (\mathbf{W}_s^{(i)} \mathbf{z}) \leq k_i, k_i > 0, \mathbf{W}_s^{(i)} \mathbf{z} \neq 0 \quad (3.11)$$

where k_i is known as the Lipschitz constant for σ_i [127], and can be determined by finding the maximum magnitude of the derivative of σ_i . After the above transformation, the origin $\mathbf{z} = \mathbf{0}$ now becomes the equilibrium point of the new SSDNN representation in (3.7). To investigate the stability of (3.5) at $\mathbf{x} = \mathbf{x}_e$, it is equivalent to test the stability of (3.7) at $\mathbf{z} = \mathbf{0}$.

Proposed Method for Verifying Local Stability

The most convenient way to test the local stability of a nonlinear system is by investigating the associated linearized system, which is known as the Lyapunov's

indirect method [140]. From (3.7), we define a nonlinear vector function

$F(\mathbf{z}) = -\mathbf{z}(t) + \tau \mathbf{W}_o \mathbf{f}(\mathbf{W}_s \mathbf{z}(t))$. Linearizing (3.7) at $\mathbf{z} = \mathbf{0}$ yields

$$\left. \frac{dF}{d\mathbf{z}^T} \right|_{\mathbf{z}=\mathbf{0}} = -\mathbf{I}_l + \tau \mathbf{W}_o \left. \frac{d\mathbf{f}}{d(\mathbf{W}_s \mathbf{z})^T} \right|_{\mathbf{z}=\mathbf{0}} \mathbf{W}_s \quad (3.12)$$

where $\mathbf{I}_l \in \mathfrak{R}^{N \times N}$ is an identity matrix and $\left. \frac{d\mathbf{f}}{d(\mathbf{W}_s \mathbf{z})^T} \right|_{\mathbf{z}=\mathbf{0}}$ is the Jacobian matrix of $\mathbf{f}$,

computed at the equilibrium point $\mathbf{z} = \mathbf{0}$. For simplicity, let $\mathbf{\Gamma} \in \mathfrak{R}^{N_h \times N_h}$ be the above Jacobian matrix

$$\mathbf{\Gamma} = \left. \frac{d\mathbf{f}}{d(\mathbf{W}_s \mathbf{z})^T} \right|_{\mathbf{z}=\mathbf{0}} = \text{diag}[\sigma_1'(\mathbf{W}_s^{(1)} \mathbf{x}_e + \mathbf{W}_u^{(1)} \mathbf{u}_0), \dots, \sigma_{N_h}'(\mathbf{W}_s^{(N_h)} \mathbf{x}_e + \mathbf{W}_u^{(N_h)} \mathbf{u}_0)] \quad (3.13)$$

where σ_i' is the derivative of σ_i , $i=1, \dots, N_h$. For example, if we use *tanh* functions for σ , $\mathbf{\Gamma}$ can be simplified as

$$\mathbf{\Gamma} = \text{diag}[1 - \sigma_1^2(\mathbf{W}_s^{(1)} \mathbf{x}_e + \mathbf{W}_u^{(1)} \mathbf{u}_0), \dots, 1 - \sigma_{N_h}^2(\mathbf{W}_s^{(N_h)} \mathbf{x}_e + \mathbf{W}_u^{(N_h)} \mathbf{u}_0)] \quad (3.14)$$

The test matrix for local stability is then given by

$$\mathbf{A}_{local} = -\mathbf{I}_l + \tau \mathbf{W}_o \mathbf{\Gamma} \mathbf{W}_s \quad (3.15)$$

Let $\lambda_1, \lambda_2, \dots, \lambda_N$ be the eigenvalues of the test matrix $\mathbf{A}_{local}$ in (3.15). We present the following criterion for the local stability of a SSDNN nonlinear model through Lyapunov's indirect method [140].

Proposed Local Stability Criterion: The equilibrium point $\mathbf{z} = \mathbf{0}$ of the SSDNN model (3.7) is said to be asymptotically (locally) stable if the real parts of all the

eigenvalues for A_{local} of (3.15) are negative, *i.e.*, $\text{Re}(\lambda_i) < 0$ for all i , $i = 1, 2, \dots, N$.

The proposed criterion can be used to determine the asymptotical stability of a trained SSDNN model given the constant model input $\mathbf{u}_0$. Since the SSDNN model is a reduced-order representation of the original nonlinear circuit, the order N of the model is typically very small. As the size of the test matrix A_{local} of (3.15) is $N \times N$, the computational cost when applying the proposed local stability criterion is low. However, the above criterion is valid only if the values of the state variables are in close neighborhood of the equilibrium point, which is not always guaranteed during the usage of the SSDNN model in circuit design. Next we present a complementary method that can be used to investigate the global stability of the model.

Proposed Method for Verifying Global Stability

The equilibrium point $\mathbf{z} = \mathbf{0}$ of the SSDNN model (3.7) is said to be globally asymptotically stable if it is locally stable in the sense of Lyapunov and globally attractive, where the global attractivity means that every trajectory of the original SSDNN system (3.5) would converge to a unique equilibrium point $\mathbf{x}_e$ as $t \rightarrow \infty$.

The GAS, also called global stability for convenience, of the SSDNN model can be investigated based on the concept of Lyapunov's direct method [140]. We assume $k_i = 1$ for $i = 1, 2, \dots, N_h$ in (3.11), which is valid for all the popular neural network activation functions, such as sigmoid functions, arctangent functions and *tanh* functions. In the following, we present a sufficient condition for the GAS of the SSDNN model.

Proposed Condition for Global Stability: The equilibrium point $\mathbf{z} = \mathbf{0}$ of the SSDNN model in (3.7) is said to be globally asymptotically stable if the following test matrix

$$\mathbf{A}_{global} = -\mathbf{I}_g + \tau \frac{(\mathbf{W}_s \mathbf{W}_o) + (\mathbf{W}_s \mathbf{W}_o)^T}{2} \quad (3.16)$$

is negative definite, where $\mathbf{I}_g \in \mathbb{R}^{N_h \times N_h}$ is an identity matrix.

Derivation: We start by defining a candidate of the Lyapunov function as

$$F_L(\mathbf{z}) = \sum_{i=1}^{N_h} \int_0^{W_s^{(i)} \mathbf{z}} f_i(r) dr \quad (3.17)$$

where f_i is defined in (3.9). It is obvious that $F_L(\mathbf{z}) = 0$ for $\mathbf{z} = \mathbf{0}$. Using property

(3.10), each integral term $\int_0^{W_s^{(i)} \mathbf{z}} f_i(r) dr$ in (3.17) is always positive for any $\mathbf{z}$, $\mathbf{z} \neq \mathbf{0}$.

Therefore $F_L(\mathbf{z})$ is a positive definite function, satisfying the basic requirement of the Lyapunov function. By differentiating $F_L(\mathbf{z})$ along the SSDNN trajectory defined by (3.7), we obtain

$$\begin{aligned} \frac{dF_L(\mathbf{z})}{dt} &= \left(\frac{dF_L(\mathbf{z})}{d\mathbf{z}} \right)^T \dot{\mathbf{z}} \\ &= \sum_{i=1}^{N_h} f_i(W_s^{(i)} \mathbf{z}) W_s^{(i)} \dot{\mathbf{z}} \\ &= \sum_{i=1}^{N_h} f_i(W_s^{(i)} \mathbf{z}) W_s^{(i)} [-\mathbf{z} + \tau W_o \mathbf{f}(W_s \mathbf{z})] \\ &= \mathbf{f}(W_s \mathbf{z})^T W_s [-\mathbf{z} + \tau W_o \mathbf{f}(W_s \mathbf{z})] \\ &= -\mathbf{f}(W_s \mathbf{z})^T W_s \mathbf{z} + \mathbf{f}(W_s \mathbf{z})^T [\tau W_s W_o] \mathbf{f}(W_s \mathbf{z}) \end{aligned} \quad (3.18)$$

Based on (3.11) and with the assumptions of $k_i = 1$, we have

$$f_i^2(W_s^{(i)}\mathbf{z}) \leq f_i(W_s^{(i)}\mathbf{z})W_s^{(i)}\mathbf{z}, \quad i=1, \dots, N_h \quad (3.19)$$

From (3.19), it follows that

$$\begin{aligned} & \left[f_1(W_s^{(1)}\mathbf{z}) \right]^2 + \left[f_2(W_s^{(2)}\mathbf{z}) \right]^2 + \dots + \left[f_{N_h}(W_s^{(N_h)}\mathbf{z}) \right]^2 \leq \\ & f_1(W_s^{(1)}\mathbf{z})W_s^{(1)}\mathbf{z} + f_2(W_s^{(2)}\mathbf{z})W_s^{(2)}\mathbf{z} + \dots + f_{N_h}(W_s^{(N_h)}\mathbf{z})W_s^{(N_h)}\mathbf{z} \end{aligned} \quad (3.20)$$

The matrix form of (3.20) reads

$$\mathbf{f}(W_s\mathbf{z})^T \mathbf{f}(W_s\mathbf{z}) \leq \mathbf{f}(W_s\mathbf{z})^T W_s \mathbf{z} \quad (3.21)$$

Combining (3.18) and (3.21) gives

$$\begin{aligned} \frac{dF_L(\mathbf{z})}{dt} & \leq -\mathbf{f}(W_s\mathbf{z})^T \mathbf{f}(W_s\mathbf{z}) + \mathbf{f}(W_s\mathbf{z})^T [\tau W_s W_o] \mathbf{f}(W_s\mathbf{z}) \\ & = \mathbf{f}^T [-I_g + \tau W_s W_o] \mathbf{f} \end{aligned} \quad (3.22)$$

In general, the matrix $W_s W_o$ in (3.22) is not symmetrical. Since

$\mathbf{f}^T W_s W_o \mathbf{f} = \mathbf{f}^T W_o^T W_s^T \mathbf{f}$, we can further derive

$$\frac{dF_L(\mathbf{z})}{dt} \leq \mathbf{f}^T \left[-I_g + \tau \frac{(W_s W_o) + (W_s W_o)^T}{2} \right] \mathbf{f} \quad (3.23)$$

It is obvious that if the matrix A_{global} in (3.16) is negative definite, $dF_L(\mathbf{z})/dt$ remains negative for any value of state variable $\mathbf{z}$. Therefore $F_L(\mathbf{z})$ is a valid Lyapunov function for the SSDNN model (3.7). Since f_i is monotonic everywhere with $f_i(0)=0$, the integral $\int_0^{W_s^{(i)}\mathbf{z}} f_i(r) dr$ in (3.17) is unbounded as $\|\mathbf{z}\| \rightarrow \infty$. Hence the Lyapunov function $F_L(\mathbf{z})$ is radially unbounded. This property further ensures that the model equilibrium point is globally attractive [35]. Thus the SSDNN model (3.7) is globally asymptotically stable at the equilibrium point $\mathbf{z} = \mathbf{0}$.

It is to be noted that by formulating the test matrix A_{global} to be a symmetrical and real matrix in (3.23), the proposed GAS condition can be conveniently used in practice through the computation of the eigenvalues of A_{global} . Let $\lambda_1, \lambda_2, \dots, \lambda_{N_h}$ be the eigenvalues of the test matrix A_{global} in (3.16). We have the following equivalent criterion for global stability.

Proposed Global Stability Criterion (GAS Criterion): the equilibrium point x_e of (3.5) is globally asymptotically stable if all the eigenvalues of the test matrix A_{global} of (3.16) are negative, *i.e.*, $\lambda_i < 0$ for all i , $i = 1, 2, \dots, N_h$.

Discussions

The stability criteria derived by the Lyapunov method are not unique. Different formulations of Lyapunov functions can lead to different criteria. Because the Lyapunov stability conditions are sufficient conditions, if SSDNN violates a specific set of stability conditions, it is still possible for SSDNN to be stable as long as it can satisfy the stability conditions derived from another Lyapunov function. The present work fills a gap in developing stability conditions for SSDNN. It is possible to develop various other forms of Lyapunov functions, *e.g.*, the multi-term formulation of Lyapunov function [129][130], leading to a variety of criteria for examining the stability of SSDNN.

3.4 SSDNN Sensitivity and Model Training

3.4.1 Adjoint Sensitivity for SSDNN

The SSDNN can be used to represent the nonlinear transient behavior only after we

train it with suitable input and output data. For our formulation, training is directly done in time-domain. Consequently, our training data are in the format of sampled input and output waveforms, which can be obtained from the original nonlinear circuit by simulation or measurement. Let N_w denote the total number of transient waveforms used for training. Let $\mathbf{u}_d^m(t)$ and $\mathbf{y}_d^m(t)$ represent the m^{th} input and output transient waveform sampled in the time-interval $[0, T]$ respectively. Let $\mathbf{y}^m(t)$ represent the SSDNN response due to the m^{th} input waveform $\mathbf{u}_d^m(t)$. The goal of training is to adjust parameters in $\mathbf{w}$ and $\mathbf{C}$ such that the objective function (overall training error)

$$E_d = \sum_{m=1}^{N_w} E_d^m = \frac{1}{2} \sum_{m=1}^{N_w} \int_0^T \left\| \mathbf{y}^m(t) - \mathbf{y}_d^m(t) \right\|^2 dt \quad (3.24)$$

is minimized. Here E_d^m is defined as the per-waveform training error for the m^{th} waveform

$$E_d^m = \frac{1}{2} \int_0^T \left\| \mathbf{y}^m(t) - \mathbf{y}_d^m(t) \right\|^2 dt. \quad (3.25)$$

Since training is essentially an optimization process, fast and accurate sensitivity information is important in order to utilize gradient-based training methods. To achieve this, we need to derive SSDNN sensitivities by modifying the adjoint sensitivity technique of DNN presented in [8][9]. Following Appendix A, we derive an adjoint SSDNN model as

$$\dot{\hat{\mathbf{x}}}(t) = \hat{\mathbf{x}}(t) - \tau \mathbf{W}_s^T \mathbf{G}(t) \mathbf{W}_o^T \hat{\mathbf{x}}(t) + \mathbf{C}^T (\mathbf{y}^m(t) - \mathbf{y}_d^m(t)) \quad (3.26)$$

where $G(t)$ contains the derivatives of the hidden neuron activation functions

$$G(t) = \text{diag}[\sigma_1'(W_s^{(1)}x(t) + W_u^{(1)}u(t)), \dots, \sigma_{N_h}'(W_s^{(N_h)}x(t) + W_u^{(N_h)}u(t))] \quad (3.27)$$

and $\hat{x} \in \mathfrak{R}^N$ is a vector of state variables for the adjoint SSDNN model. The boundary conditions for (3.26) are imposed at the upper time limit as $\hat{x}(T) = \mathbf{0}$. Based on the adjoint SSDNN in (3.26) along with the boundary conditions, the general form of training error sensitivity for the m^{th} waveform can be derived as

$$\frac{dE_d^m}{dw_{ij}^l} = - \int_0^T \hat{x}^T \left[\tau \frac{dW_o}{dw_{ij}^l} \sigma + \tau W_o G \left(\frac{dW_u}{dw_{ij}^l} u + \frac{dW_s}{dw_{ij}^l} x \right) \right] dt. \quad (3.28)$$

Depending on the location of the variable w_{ij}^l , the general sensitivity expression of (3.28) can be further specified. For example, using $\tanh$ functions for σ , the expression of (3.28) is further simplified as

$$\frac{\partial E_d^m}{\partial w_{ij}^l} = \begin{cases} - \int_0^T \tau \hat{x}_i \sigma_j dt, & \text{for } l=3 \\ - \sum_{k=1}^N \int_0^T \tau \hat{x}_k w_{ki}^3 (1 - \sigma_i^2) u_j dt, & \text{for } 1 \leq j \leq M, l=2 \\ - \sum_{k=1}^N \int_0^T \tau \hat{x}_k w_{ki}^3 (1 - \sigma_i^2) x_{j-M} dt, & \text{for } j > M, l=2 \end{cases} \quad (3.29)$$

To compute the sensitivity of the overall training error E_d , we perform the sensitivity analysis using (3.5), (3.26), and (3.29) for each training waveform m to obtain $\frac{dE_d^m}{dw_{ij}^l}$, $m=1,2,\dots,N_w$. Finally, the total error sensitivity for training is calculated as

$$\frac{dE_d}{dw_{ij}^l} = \sum_{m=1}^{N_w} \frac{dE_d^m}{dw_{ij}^l} \quad (3.30)$$

and

$$\frac{dE_d}{dC_{ij}} = \sum_{m=1}^{N_w} \int_0^T (y_i^m - y_{di}^m) x_j dt \quad (3.31)$$

where y_i^m and y_{di}^m represent the i^{th} output of the SSDNN and training data for the m^{th} training waveform, respectively and C_{ij} is an element in matrix C .

3.4.2 SSDNN Training

A basic training method for the SSDNN model is to solve the unconstrained optimization problem by minimizing E_d (3.24) based on a set of input and output training waveforms [10]. In this basic method, there are two steps involved in training SSDNN to learn nonlinear transient data. Firstly, training is carried out by applying explicit integration method (*e.g.*, Runge-Kutta method) to solve both the SSDNN (3.1) and its associated adjoint system (3.26). This training is numerically efficient since at each time instance no Newton iterations are required. These approximate solutions provide good starting point for the refined training in the next step. The SSDNN is further trained based on implicit integration (*e.g.*, Trapezoidal Rule) of (3.1) and (3.26) to improve the accuracy of the final model.

Because the weight update is purely based on minimizing the training error, the training method in [10] only enforces accuracy but not stability. To overcome this problem, we propose a new formulation for SSDNN training, which automatically

ensures the GAS property of the SSDNN model once the training is accomplished. Here we use our proposed GAS criterion as the constraint for training. The modified SSDNN training is a constrained optimization problem,

$$\begin{aligned} & \underset{w, C}{\text{minimize}} E_d \\ & \text{subject to } \text{Max}_{i \in [1, N_h]} [\lambda_i] < 0 \end{aligned} \quad (3.32)$$

where λ_i is the i^{th} eigenvalue of the matrix A_{global} defined in (3.16). The constrained training is implemented in MATLAB 7.0 using the *fmincon* function from the optimization toolbox [141].

During the constrained training, the derivatives of E_d with respect to model internal parameters are calculated analytically by following (3.5), (3.26), and (3.29) – (3.31) while the derivative of the constraint in (3.32) is numerically computed by the finite-difference method. Weight parameters are initialized by uniform random distribution between $[-0.1, 0.1]$ and are used as the starting points for the constrained training. Figure 3.2 is the block diagram illustrating the proposed constrained training algorithm. Once the training is completed, the accuracy of the trained SSDNN model is further validated with an independent set of test waveforms.

After the successful verifications of accuracy and stability, the trained SSDNN model can be conveniently incorporated into a SPICE-like circuit simulator for high-level circuit simulation and design, following the exact circuit representation as shown in Figure 3.3. The proposed method provides a further progress beyond the existing treatment for stability of state-space neural networks [128][135], where stability theorems were mainly

used for verifications. The proposed constrained training of SSDNN achieves good model accuracy and at the same time enforces the model stability.

Figure 3.2: Flowchart for the proposed SSDNN constrained training algorithm for enforcing the GAS of a trained SSDNN model. The training error sensitivities are efficiently computed by exploiting the adjoint SSDNN model.

Figure 3.3: Circuit representation of the SSDNN nonlinear model. All the resistor and capacitor values are unity.

3.5 Demonstration Examples

3.5.1 CMOS Inverter

This example is to demonstrate the validity of our proposed constrained training algorithm using the GAS criterion as the constraint. Two SSDNN models, namely model I and model II, are trained to learn transient behavior of a CMOS inverter using the original unconstrained training method and the proposed constrained training method, respectively. The CMOS inverter is a standard circuit configuration containing two complementary

MOSFETs connected with each other [142]. Our SSDNN model contains one input v_i (input voltage) and one output v_o (output voltage). Training waveforms of the inverter are generated using level-1 HSPICE MOSFET models [18], for different rise time of the input pulse source [1ns, 2ns step 0.2ns]. The total training data set contains 6 waveforms. The same data set is also used for testing purpose. To investigate the GAS property of the model, normally distributed random noise signals are added to the input waveforms. For this modeling case, the three-layer MLP neural network has been applied to represent the f_{ANN} in the SSDNN structure of (3.1). The $\tanh$ function has been used as the activation function in the hidden layer of the MLP. The SSDNN structure with order $N = 2$ and number of hidden neurons $N_h = 5$ was sufficient to assure model test error within 1%. This is at similar level of accuracies for circuit modeling as existing literature, *e.g.*, [120], [143].

After training, we check the stability of the trained SSDNN models based on our local and global stability criteria and the eigenvalues of the test matrices A_{local} and A_{global} are summarized in Table 3.1. As observed, both models can satisfy the local stability criterion and the model developed by the constrained training method is globally asymptotically stable as well. To verify the GAS by simulation, we randomly select 20 initial values of SSDNN state variables around the model equilibrium point and simulate both models up to 15ns. Figure 3.4 shows the comparison of state trajectories for these two models under the same initial conditions. It is noticed that for Model I which is only locally stable but not globally stable, the

trajectories converge to non-unique equilibriums, meaning that the state variables may have wrong solutions depending upon initial conditions. It is also observed that Model II which satisfies our proposed GAS criterion, indeed has a unique equilibrium point that attracts all the trajectories. This example demonstrates the importance of stability property of the model. The example also confirms that using our proposed constrained training method the SSDNN model can be guaranteed to preserve the GAS.

Table 3.1: Comparisons of Matrix Eigenvalues using Different Stability Criteria for the CMOS Inverter Example. Model I is Obtained by Original Unconstrained Training Method and Model II is Obtained by Our Proposed Constrained Training Method.

	Eigenvalues of the test matrix for stability criteria		Stability condition satisfied?	
	Model I	Model II	Model I	Model II
Local Stability	-9.1750e-001 +j3.5036e-002 -9.1750e-001 -j3.5036e-002	-2.4778e+000 -9.9561e-001	Yes	Yes
GAS	-9.2069e+000 -4.2158e+000 -1.0000e+000 1.5992e+000 5.4592e+000	-1.1667e+001 -1.2446e+000 -1.0000e+000 -7.8768e-001 -4.1635e-001	No	Yes

(a)

(b)

Figure 3.4: The state trajectories of the SSDNN for modeling a CMOS inverter under 20 random initial conditions: (a) Model I without GAS property where state variables converge to different equilibrium points, meaning that none of the equilibrium points are globally asymptotically stable. (b) Model II with GAS property where state variables converge to a unique equilibrium point.

3.5.2 Receiver Buffer

A high-speed receiver buffer circuit based on HSPICE level-47 BSIM3 MOSFET models [18] has been modeled through the SSDNN technique. The input current and voltage of the receiver are chosen as the input and output of SSDNN, respectively. The training waveforms are generated by exciting the circuit with pulse signals with different rise time [0.05ns – 0.3ns, step 0.05ns] and different amplitudes [-0.5v – 2.5v, step 0.1v] to maximally stimulate the nonlinear dynamic behavior of the circuit in different working regions. The resulting training data set has a total of 186 waveforms. In addition, a separate data set with 150 test waveforms is also generated for the validation of the trained SSDNN model. Using the SSDNN structure of $N = 2$ and $N_h = 10$, an accurate SSDNN model has been achieved with average test error around 0.13%. The $\tanh$ function was used as the activation function for each hidden neuron. The model accuracy achieved here is similar to that reported in the existing literature for modeling this type of circuit, *e.g.*, [120], [143]. It is to be noted that the order of the SSDNN represents the effective order (*i.e.*, the reduced order) of the original circuit that is observable from circuit external transient signals. Therefore, our proposed model is much more compact than the state-space representation of the detailed nonlinear circuit. For all the examples in this chapter, the suitable order of the SSDNN is determined through an adaptive process. Starting with the minimum value, the order is gradually increased until the SSDNN model satisfies the desired accuracy based on the test signals. Furthermore, the SSDNN model has been verified in a practical circuit

configuration where the receiver is connected to a high-speed interconnect [10]. In [10], a set of noisy training data, where white noise is applied to the output waveforms, has been used to train both SSDNN models and existing DNN models. The SSDNN model has been shown to give more reliable solutions than the standard DNN models.

Here, we apply our proposed stability criteria to verify the stability of the trained SSDNN models. Table 3.2 shows the comparison of the eigenvalues of the test matrices A_{local} and A_{global} using various types of stability criteria presented in Chapter 3.3.2. From the table, we observe that the receiver SSDNN model developed by the original unconstrained training in [10] is locally stable. However, the global stability is not guaranteed. To enforce the GAS of the trained SSDNN model, we re-train a new SSDNN model with the same model structure ($N = 2$ and $N_h = 10$) as the original SSDNN by applying the new constrained training algorithm in Chapter 3.4.2. The new SSDNN model has been trained to have similar accuracy as the original one. After the training is completed, we verify the local stability and GAS of the model using the criteria proposed in Chapter 3.3.2. For checking stabilities of the SSDNN model, we compute the eigenvalues of the test matrices A_{local} and A_{global} , given in Table 3.2. It can be observed that with the new constrained training algorithm both the local stability and GAS of the model have been ensured, demonstrating further advances over [10].

Table 3.2: Eigenvalues for Stability Verification of the SSDNN Model for the Receiver Example, and Comparison between Proposed Constrained Training versus Unconstrained Training

	Eigenvalues of the test matrix for stability criteria		Stability condition satisfied?	
	Model with unconstrained training	Model with proposed constrained training	Model with unconstrained training	Model with proposed constrained training
Local Stability	-1.9374e+001 -2.5352e+000	-7.5674e+000 -4.6456e+000	Yes	Yes
GAS	-3.5620e+001 -4.4540e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 7.7986e+000 1.4522e+001	-8.2460e+001 -1.1416e+001 -1.5686e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -2.3522e-002	No	Yes

3.5.3 Physics-Based Driver Buffer

In this example, we demonstrate the application of the SSDNN technique for physics-based transient modeling. We consider a four-stage CMOS driver including 8 transistors connected with each other [10]. The driver load is a lossy transmission line ($R=36\Omega/\text{m}$, $L=360\text{nH}/\text{m}$, $C=100\text{pF}/\text{m}$, and $G=0.01\text{S}/\text{m}$) terminated with a 5pF capacitor. Transient training waveforms are generated for different values of pulse rise-time [0.25ns–0.75ns, step 0.25ns], pulse amplitude [4.5v–5.5v, step 0.5v], and interconnect length [0.08m–0.14m, step 0.02m] using physics-based simulator MINIMOS-NT [144]. The training data set contains 36 waveforms in total. The SSDNN inputs and outputs are defined as $\mathbf{u}(t)=[v_{in}(t), i_{out}(t)]^T$ and $\mathbf{y}(t)=[v_{out}(t)]^T$, respectively. The SSDNN structure of $N = 2$ and $N_h = 15$ has been utilized and the average test error achieved is 0.21% using an independent data set containing 12 waveforms. The $\tanh$ function was used as the activation function of the hidden neurons in MLP. The level of accuracy of the SSDNN model is similar to that in existing literature for nonlinear buffer modelling, *e.g.*, [120], [143]. In [10], it has been demonstrated that for 50 waveform evaluations, the total CPU time was reduced from 11.6 hours using MINIMOS down to 9 seconds using the SSDNN model. A set of noisy training data has also been applied to train both SSDNN and DNN models. It was observed in [10] that the SSDNN consistently achieves better accuracy compared to the standard DNN when there is noise in the training data.

Here, we perform additional comparison of the SSDNN with several standard RNN approaches, namely, Elman network [138], GRNN (MLP with delayed inputs and outputs) [84] and DNN [6]. As shown in Table 3.3, all the methods except the Elman network have achieved very good accuracy for modeling the physics-based driver. It demonstrates that by adopting external recurrence over the MLP structure as opposed to internal feedback over each neuron, GRNN, DNN and SSDNN have enough degree of freedom to represent the nonlinear analog behavior of the driver. To further compare the model performance, we incorporated the four different models, *i.e.*, Elman, GRNN, DNN, and SSDNN models, into HSPICE and verified that they can be used to perform transient circuit simulations. Due to their discrete-time natures, Elman and GRNN models need certain approximations when converting the time-difference equations into the differential equations in order to obtain an equivalent circuit representation. The DNN and SSDNN models, on the other hand, have exact circuit representations in SPICE-like circuit simulators owing to their continuous-time model formats. The SSDNN technique provides attractive overall quality of model accuracy, flexibility for circuit simulation, and reliability in the noisy environment.

Next we explore the robustness of the SSDNN models by performing the model stability analysis with both types of criteria proposed in Chapter 3.3.2. Table 3.4 shows the comparison of the eigenvalues of the test matrices A_{local} and A_{global} by applying different types of stability criteria to the SSDNN models. The table confirms that the SSDNN buffer model trained with unconstrained training used in

[10] is locally stable. However, there is no guarantee that the model has the superior property of global stability. To further achieve the GAS, we train another SSDNN model ($N = 2$ and $N_h = 15$) by applying the new constrained training scheme in Chapter 3.4.2. After training, the average test error is found to be 0.65%, confirming the model accuracy. For this new model, we again check the stability using our proposed criteria. As shown in Table 3.4, all the eigenvalues of the test matrices A_{local} and A_{global} are found to be negative indicating that both the local stability and GAS have been achieved successfully. From this study, it is demonstrated that through the proposed constrained training method, the SSDNN model can be developed to achieve good accuracy with guaranteed GAS, showing further advances over [10].

Table 3.3: Comparison of the Model Test Error between Discrete RNN, DNN and SSDNN for Transient Modeling of a Physics-Based Driver.

	Elman	GRNN	DNN	SSDNN
Model Test Error	5.26e-2	3.15e-3	2.52e-3	2.10e-3

Table 3.4: Eigenvalues for Stability Verification of the SSDNN Model for the Physics-Based Driver Example, and Comparison between Proposed Constrained Training versus Unconstrained Training.

	Eigenvalues of the test matrix for stability criteria		Stability condition satisfied?	
	Model with unconstrained training	Model with proposed constrained training	Model with unconstrained training	Model with proposed constrained training
Local Stability	-1.0561e+001 $j1.7316e+000$ -1.0561e+001 $-j1.7316e+000$	-1.1569e+001 -1.6446e+000	Yes	Yes
GAS	-3.8325e+001 -1.6530e+001 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 2.1715e+000 8.9110e+000	-6.3185e+000 -2.2598e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.0000e+000 -1.7287e-001 -1.3254e-003	No	Yes

3.5.4 8-Bit High-Speed Driver

In this example, we consider the transient modeling of a commercial IC package, namely an octal driver SN74AHCT540 from Texas Instruments (TI) Inc., as shown in Figure 3.5. For this chip, both detailed transistor-level model and IBIS model are available [145], where the former model is slow and accurate, and the latter is faster but less accurate. Here we aim to use the SSDNN model to represent the nonlinear dynamic behavior between the input pin and output pin. We define $\mathbf{u}(t) = [v_{in}(t), i_{out}(t)]^T$ and $\mathbf{y}(t) = [v_{out}(t)]^T$ to be the inputs and outputs of the SSDNN model respectively. Buffer input pin is excited by a pulse signal. The buffer output pin is terminated with a transmission line ($Z_0 = 50\Omega$ and $T_D = 0.5\text{ns}$) in series with a 50pF capacitor load. We simulate the driver package using transistor-level model in HSPICE to gather different training waveforms. Specifically, the training waveforms are obtained by varying pulse rise time T_r [0.5ns – 2.0ns, step 0.5ns] and the length of the transmission line load d [1cm – 3cm, step 0.5cm]. The entire training set contains 20 waveforms. The f_{ANN} used in the SSDNN model is a three-layer MLP with $\tanh$ function as the activation function for the hidden layer. The model structure used in this example has order $N = 2$ and number of hidden neurons $N_h = 8$. SSDNN model training is performed following the constrained training algorithm as summarized in Chapter 3.4.2.

After training, the SSDNN model accuracy is confirmed by an independent set of test waveforms with 12 waveforms that are never applied during the training. We

then incorporate the trained SSDNN model into HSPICE and perform the transient simulation of the model for SI analysis. Figure 3.6 shows the comparison of the voltages at the output pin using three types of models, *i.e.*, detailed transistor-level model, IBIS model and the proposed SSDNN model. It can be observed that SSDNN solutions can match the signal variations from the detailed transistor-level model more accurately than that of the IBIS model. Therefore, the proposed SSDNN provides better accuracy than IBIS model. We also compare the three different models in terms of relative error and CPU time in HSPICE transient simulation, as given in Table 3.5. The SSDNN provides similar level of accuracies as the existing literature for I/O buffer modeling, *e.g.*, [120], [143]. The CPU time in the table is for the simulation of output waveforms of driver-receiver buffers with transmission line load for 50 variations of T_r and d . The SSDNN model is shown to be 12 times faster than the detailed transistor-level model, without sacrificing the accuracy.

To verify the stability of the trained SSDNN model, the eigenvalues of the test matrix A_{global} in (3.16) are computed, and the maximum eigenvalue is found to be -0.0059 , indicating that our SSDNN model is globally asymptotically stable. To further validate the GAS property, we perform the transient simulation of the SSDNN model for randomly selected initial conditions around the equilibrium point and the obtained trajectories of SSDNN state variables are shown in Figure 3.7. It can be observed that all the trajectories asymptotically converge to the unique equilibrium point of the model as time proceeds. This example demonstrates that the SSDNN technique provides best overall efficiency, being more accurate than IBIS

model and faster than detailed transistor-level model. The proposed constrained training algorithm leads to fast, accurate and robust modeling solutions for high-speed IC design.

Figure 3.5: The schematic of a TI 8-bit high-speed driver to be modeled by SSDNN. Training waveforms of $v_{in}(t)$, $v_{out}(t)$, and $i_{out}(t)$ are generated by HSPICE simulations using detailed transistor-level models for developing the SSDNN model.

(a)

(b)

Figure 3.6: Sample comparisons of the output voltage responses using detailed transistor-level model (—), IBIS model (---), and proposed SSDNN model (o) for the TI driver example. The comparisons shown here are the cases for (a) $T_r = 1.25\text{ns}$ and $d = 1.25\text{cm}$, and (b) $T_r = 1.75\text{ns}$ and $d = 2.75\text{cm}$. Good agreement is achieved between the responses of the proposed SSDNN models and those of detailed transistor-level models even though such test waveforms are never used in training.

Figure 3.7: The state-space trajectories of the SSDNN model in the TI driver example. These trajectories are generated by transient simulation of the model with randomly selected initial conditions, as $t \rightarrow \infty$. The SSDNN model is trained to be globally asymptotically stable through our proposed constrained training algorithm. The model state variables x_1 and x_2 all converge to a unique equilibrium point as observed from the figure.

Table 3.5: Comparisons of CPU Time and Accuracy for Various Transient Models for HSPICE Simulation in the TI Driver Example.

	Transistor-level Model	Proposed SSDNN Model	IBIS Model
CPU Time	2052 Seconds	151 Seconds	36 Seconds
Error relative to the transistor-level model	0.0% (reference for comparison)	0.8%	22.3%

3.6 Summary

We have presented a generalized neural-based modeling technique to represent the transient behavior of nonlinear circuits for high-speed applications, such as IC drivers and receivers. The SSDNN technique efficiently modifies and extends the DNN formulation, established mainly for frequency-domain steady state modeling, to cover time-domain transients. To address the robustness of the SSDNN model, we have derived several stability criteria that facilitate convenient investigations of both local stability and global stability based on the SSDNN model parameters. As a further step towards enhancing model robustness, an advanced constrained training scheme has been developed that can automatically produce the SSDNN models with

good accuracy and guaranteed GAS. The proposed constrained training algorithm has been demonstrated through the modeling of circuit-based and physics-based I/O buffers. The proposed technique can be used for achieving efficient CAD of high-speed ICs including high-speed interconnect and nonlinear driver/receiver buffers [12].

Chapter 4: A Robust Training Approach for Transient Modeling of Nonlinear Circuits Based on Recurrent Neural Networks

In this chapter, another major contribution of the thesis is presented, addressing the development of accurate and robust transient neural models for system-level simulation and design. With the RNN as the modeling vehicle, a new training approach for transient modeling of nonlinear circuits is proposed, overcoming the conventional limitations where training information depends on the shapes of circuit waveforms and/or circuit terminations. By exploiting an internal space of an RNN, called the internal input-neuron space, a new circuit block combining a generic load and a generic excitation is formulated as a generalized circuit termination. A rich combination of training waveforms is systematically obtained by sweeping the coefficients of the proposed circuit block, leading to effective coverage of the region of interest in the internal input-neuron space. A new data reduction method is developed to significantly reduce the size of training data while maintaining the necessary modeling information. The proposed method is demonstrated through examples of RNN modeling of high-speed drivers and an RF amplifier. It is confirmed that, for different terminations and test waveforms, the model trained with the proposed technique yields better accuracy and robustness than that using the existing training methods.

4.1 Introduction

Recently, ANN techniques have emerged as useful approaches to nonlinear behavioral modeling and optimization for microwave devices [146], [147], [148] and circuits [6], [8]-[12], [46], [149]. This chapter further explores the methodology of developing robust and accurate ANN models for nonlinear circuits. Generally speaking, nonlinear circuit modeling is a more challenging task compared to linear circuit modeling. In the case of linear circuit modeling, the magnitude and phase of the transfer function over frequency provide complete information about the behavior of the original circuit. In other words, circuit terminations (loads) and signal shapes have no impact on the modeling process of the linear circuit. In contrast, for nonlinear circuit modeling, the choice of waveform shapes and terminations directly affects the training of the ANN model, because different waveform shapes and terminations produce different behavioral information of the nonlinear circuit. In existing methods, training data are often generated based on sweeping the parameters in the physical loads that are similar to the test loads, such as lumped components and transmission lines [10], [11]. Sweeping the parameters in physical sources, in the form of time-domain step-like sources [82], [109] or frequency-domain power sources [50], [147], has also been used to generate training data for the development of neural-network models for nonlinear circuits. These methods are important advances in the area of nonlinear circuit modeling using ANNs. However, the existing methods may not be adequate to ensure the accuracy of general time-domain neural networks, especially when the trained neural-network model deals with new

waveform shapes and circuit terminations that are not known during training. For example, such a situation could happen during system verifications, where the model may be presented with the terminations that are never seen in training due to the interaction between the model and other circuit blocks. Another example is the nonlinear simulation with ANN models where the input signals of the ANN model, determined by the iterative nonlinear solvers, can have different shapes compared to the training signals. Therefore, ensuring the model accuracy for versatile signal waveforms and circuit loads is an important task for neural-network-based nonlinear modeling and simulation.

In this chapter, we address these issues by investigating the use of a finite set of waveforms with basic shapes for ANN training, enabling the trained model to respond accurately to test waveforms of unknown shapes. We develop our technique using a fundamental type of ANN structures, *i.e.*, RNNs, which is a generic formulation for discrete-time neural networks used in nonlinear microwave modeling. In the RNN formulation, a feedforward neural network is used to map the histories of circuit input and output signals to the output signals at the present time. In this way, the RNN establishes a dynamic representation of the behavior of the original circuit. In this work, we propose to exploit an internal space of the RNN, called the internal input-neuron space, as the vehicle to relate the information of training waveforms with that of test waveforms. We propose a novel circuit block combining a generic load and a generic excitation to terminate the original circuit. Utilizing the proposed circuit block, we present a new RNN training approach that can produce different patterns of voltage and current signals to cover the region of interest in the internal input-neuron space. Through

this approach, the RNN model can be trained to handle test waveforms with completely different shapes compared to training waveforms. To facilitate efficient model training, we develop an algorithm to reduce substantially the amount of training data while retaining the relevant modeling information about the original circuit. The validity of the proposed method is confirmed by the transient behavioral modeling of high-speed I/O drivers and steady-state large signal behavioral modeling of an RF amplifier.

4.2 Overview of Nonlinear Transient Modeling using RNN formulations

As defined in Chapter 2.2.3, $u(t)$ represents the input signals of a nonlinear circuit, *e.g.*, the port voltage of a high-speed I/O driver, and $y(t)$ denotes the output signals of a nonlinear circuit, *e.g.*, the port current of a high-speed I/O driver. The original detailed circuit may be computationally expensive to be used for the simulation and design at the system-level. To enhance the model efficiency, a fast neural-network model can be developed through an optimization process called training to learn the u - y relationship using various signals from the original circuit [1].

In this chapter, we develop our technique using a generic time-domain neural network formulation, *i.e.*, an RNN, which is capable of modeling both transient and large signal steady-state behavior. The method based on RNNs has been a useful technique for nonlinear behavioral modeling of RF amplifiers [49], [51], [5], and high-speed I/O buffers [109]. Compared to non-recurrent neural models, RNN models have been shown to have increased dynamic modeling capability to represent the behavior of nonlinear devices and circuits [5].

In an RNN, $y(t)$ is a function of $u(t)$ as well as the time-delayed histories of $u(t)$ and $y(t)$. Let m and n represent the number of delay steps for input signals $u(t)$ and output signals $y(t)$, respectively. m and n are also called the input and output orders of the RNN model, respectively. The generic RNN formulation for representing the nonlinear dynamic input-output relationship can be described as

$$y(t) = f_{FFNN}(u(t), u(t-h), u(t-2h), \dots, u(t-mh), y(t-h), y(t-2h), \dots, y(t-nh)) \quad (4.1)$$

where f_{FFNN} represents a static feedforward neural network, such as an MLP network [1], and h is the sampling time interval. To ensure that all the information of the original input and output signals is retained after sampling, h is chosen to be smaller than the Nyquist interval, which is equal to the reciprocal of twice the highest frequency of the input and output signals. Note that once h is determined, it will be used as a fixed constant for discretizing the continuous-time waveforms of input and output signals. As such, the value of h affects the amount of discrete-time training samples used in model training.

In this chapter, we use an RNN as a simplified representation of a two-port nonlinear device or circuit. Let $u(t) = [v_i(t) \ v_o(t)]^T$ and $y(t) = [i_o(t)]$, where $v_i(t)$ and $v_o(t)$ represent the input and output port voltages, respectively, and $i_o(t)$ denotes the output port current. In this case, the RNN formulation in (4.1) becomes

$$i_o(t) = f_{FFNN}(v_i(t), v_i(t-h), v_i(t-2h), \dots, v_i(t-mh), v_o(t), v_o(t-h), v_o(t-2h), \dots, v_o(t-mh), i_o(t-h), i_o(t-2h), \dots, i_o(t-nh)) \quad (4.2)$$

For the system-level simulation and design, a good RNN model is desired to provide accurate solutions when it is connected to other circuit elements in the system. In other words, the RNN model should maintain good accuracy for various test waveforms and circuit terminations. Since the potential test waveforms could have unknown shapes, it is a challenging task to generate a finite set of training waveforms that can sufficiently account for the broad information embedded in various test waveforms. In the next section, we will present a systematic approach for RNN training such that the information regarding the unknown test waveforms is properly represented at the training stage.

4.3 Proposed Training Approach for RNN-Based Nonlinear Modeling

4.3.1 Problem Statement

An RNN model is composed of a feedforward neural network (e.g., MLP) and time delayed signals of model inputs and outputs, as shown in Figure 4.1. Similar to the notation in Chapter 2, we use vector $\tilde{\mathbf{u}}$ to represent the values supplied to the input neurons of the feedforward neural network. Notice that the input neurons of the feedforward neural network are internal neurons for the overall RNN. For the convenience of our description, the space spanned by $\tilde{\mathbf{u}}$ will be called the internal input-neuron space. The $\tilde{\mathbf{u}}$ -vector will be referred to as the input dimensions of the feedforward neural network. For the simplified two-port formulation, $\tilde{\mathbf{u}}$ contains the input and output port signals at the present time as well as their time-delayed histories at previous time instants, *i.e.*,

$$\tilde{\mathbf{u}} = [v_i(t) \ v_i(t-h) \ v_i(t-2h) \ \dots \ v_i(t-mh) \ v_o(t) \ v_o(t-h) \ v_o(t-2h) \ \dots \ v_o(t-mh) \ i_o(t-h) \ i_o(t-2h) \ \dots \ i_o(t-nh)]^T. \quad (4.3)$$

Figure 4.1: Illustration of the external waveform space of an RNN and the internal input-neuron space. The entire diagram represents the RNN and the box with a dashed border represents the feedforward neural network.

The waveform space of an RNN, *i.e.*, $v_i(t)$, $v_o(t)$, and $i_o(t)$, is a functional space because different waveform shapes of voltages and currents carry different modeling information. The conventional way of representing a functional space is to use a linear combination of base functions (waveforms). However, this concept is not applicable in our case since our circuit is nonlinear and superposition (of base functions) does not help. Therefore, representing the information of a generic unknown waveform in the waveform space (functional space) is not a straightforward task. Here we propose to exploit the specific structure of the RNN, which consists of the time-delayed history of the waveform (a short waveform segment) and a static feedforward neural network. We propose to use the internal input-neuron space to represent the dynamic information of a short segment of the waveform. This space is a parametric space, and the information about the space can be represented by data samples in different locations in the space. Based on this concept, the training and test waveforms with different shapes can be related to different locations in the internal input-neuron space. This is achieved by transforming the signal from the waveform space into the trajectory in the internal input-neuron space. The objective of the proposed training approach is to facilitate systematic generation of training data, which can effectively cover various locations in the internal input-neuron space. In this way, the information about unknown test waveforms can be adequately accounted for during model training. For simplicity, we assume that the input port voltages $v_i(t)$ of the original circuit have predetermined shapes. For example, in the case of high-speed I/O drivers, the driver input port is

normally excited with digital signals of trapezoidal shapes. Using this prior knowledge, this chapter focuses on effectively covering a sub-part of the internal input-neuron space, which is spanned by $v_o(t)$ and $i_o(t)$ as well as their time-delayed signals. Good coverage of this subspace will allow the trained RNN model to represent the effects of different circuit loads and different waveform shapes of the signals at the circuit output port.

4.3.2 Formulation to Explore the Internal Input-Neuron Space

We start by describing our new formulation in transient domain followed by its complementary version for steady-state modeling. In an effort to develop a generic training approach, we consider two ways conceptually to terminate the original circuit for gathering training information. One way is to specify a generic load, and the other way is to specify a generic excitation. The generic load and excitation should contain both linear and nonlinear load information. Since the spaces of generic excitation and generic load are also functional spaces, it is difficult to represent the effects of different excitations and loads using a finite set of training waveforms. To solve this problem, we again exploit the specific structure of the RNN. More specifically, we exploit the finite length of the time-delayed history of model input and output signals. Based on the input order m of the RNN, we propose a generic excitation $v_E(t)$ that contains different piece-wise linear signals with $V_i, i = 1, 2, \dots, m+1$, representing the signal amplitude at $m+1$ consecutive time points. On the other hand, we propose a generic load in the form of a rational function representing a complex admittance:

$$D(s) = \frac{a_0 + a_1 s + \dots + a_m s^m}{1 + b_1 s + \dots + b_n s^n} \quad (4.4)$$

where $\mathbf{a} = [a_0 \ a_1 \ \dots \ a_m]^T$ and $\mathbf{b} = [b_1 \ b_2 \ \dots \ b_n]^T$ are the coefficients of the rational function. The orders m and n in the rational function $D(s)$ are specially designed to match the input and output orders of the RNN, respectively. To explore the internal input-neuron space, we combine the proposed generic load and excitation to form a new circuit block that is used as a general termination of the original circuit during data generation, as shown in Figure 4.2. Here V_{dd} and V_{ss} represent the high and low power supply voltages of the original circuit, respectively.

In time-domain, the proposed formulation in Figure 4.2 can be described by

$$\begin{aligned} & a_0(v_o(t) - v_E(t)) + a_1(v_o(t) - v_E(t))' + \dots + \\ & a_m \frac{d^m(v_o(t) - v_E(t))}{dt^m} = i_o(t) + b_1 i_o(t)' + \dots + b_n \frac{d^n i_o(t)}{dt^n}. \end{aligned} \quad (4.5)$$

To express the proposed formulation with the input dimensions of the feedforward neural network, we utilize a backward-difference approximation of time derivatives to convert (4.5) into the discrete-time version as

$$\begin{aligned} & a_0(v_o(t) - v_E(t)) + a_1 h^{-1} \sum_{k=0}^1 (-1)^k \binom{1}{k} (v_o(t - kh) - v_E(t - kh)) + \\ & \dots + a_m h^{-m} \sum_{k=0}^m (-1)^k \binom{m}{k} (v_o(t - kh) - v_E(t - kh)) = i_o(t) + \\ & b_1 h^{-1} \sum_{k=0}^1 (-1)^k \binom{1}{k} i_o(t - kh) + \dots + b_n h^{-n} \sum_{k=0}^n (-1)^k \binom{n}{k} i_o(t - kh). \end{aligned} \quad (4.6)$$

Rearranging the terms in (4.6) yields

Figure 4.2: The proposed circuit block with a generic load and excitation to explore the internal input-neuron space for training RNN models.

$$\sum_{k=0}^m \hat{a}_k (v_o(t-kh) - v_E(t-kh)) = i_o(t) + \sum_{k=1}^n \hat{b}_k i_o(t-kh). \quad (4.7)$$

The coefficients $\hat{\mathbf{a}} = [\hat{a}_0 \ \hat{a}_1 \ \dots \ \hat{a}_m]^T$ and $\hat{\mathbf{b}} = [\hat{b}_1 \ \hat{b}_2 \ \dots \ \hat{b}_n]^T$ in (4.7) can be respectively related to the coefficients $\mathbf{a}$ and $\mathbf{b}$ as

$$\begin{bmatrix} \hat{a}_0 \\ \hat{a}_1 \\ \vdots \\ \hat{a}_m \end{bmatrix} = \gamma \begin{bmatrix} 1 & (-1)^0 \binom{1}{0} h^{-1} & \dots & (-1)^0 \binom{m}{0} h^{-m} \\ 0 & (-1)^1 \binom{1}{1} h^{-1} & \dots & (-1)^1 \binom{m}{1} h^{-m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & (-1)^m \binom{m}{m} h^{-m} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} \quad (4.8)$$

and

$$\begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \vdots \\ \hat{b}_n \end{bmatrix} = \gamma \begin{bmatrix} (-1)^1 \binom{1}{1} h^{-1} & (-1)^1 \binom{2}{1} h^{-2} & \dots & (-1)^1 \binom{n}{1} h^{-n} \\ 0 & (-1)^2 \binom{2}{2} h^{-2} & \dots & (-1)^2 \binom{n}{2} h^{-n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & (-1)^n \binom{n}{n} h^{-n} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad (4.9)$$

where γ is a scaling factor, given by $\gamma = 1 / (1 + \sum_{i=1}^n b_i h^{-i})$. Notice that the voltage and current variables in (4.7) are also the input dimensions of the feedforward neural network, therefore (4.7) realizes a linear hyper-plane in the sub-part of the internal input-neuron space, which contains $v_o(t)$ and $i_o(t)$ as well as their time-delayed signals. The coefficients $\hat{a}$ and $\hat{b}$ in (4.7) control the direction of the hyper-plane. Because the matrices in (4.8) and (4.9) are both upper-triangular matrices with non-zero diagonal elements, the equations in (4.8) and (4.9) have unique solutions. In other words, for any given values of $\hat{a}$ and $\hat{b}$, the coefficients a and b of the rational function can be uniquely determined by (4.8) and (4.9), respectively. Thus by sweeping a and b , the hyper-plane in (4.7) can be allowed to have various angles. On the other hand, $v_E(t)$ and its time-delayed signals $v_E(t-h), \dots, v_E(t-mh)$ represent the individual shift of the hyper-plane along the input dimensions $v_o(t), v_o(t-h), \dots, v_o(t-mh)$ of the feedforward neural network, respectively, as defined in (4.7). By controlling the amount of the shift, the hyper-plane can be moved from one end to the other in the internal input-neuron space. As such, by

simultaneously sweeping coefficients $\mathbf{a}, \mathbf{b}$, and $v_E(t)$, the hyper-plane defined in (4.7) can be designed to explore different places in the internal input-neuron space. Figure 4.3 shows the examples of the hyper-plane in the sub-part of the internal input-neuron space containing $v_o(t), v_o(t-h)$, and $i_o(t-h)$.

Our proposed formulation is designed to allow the trajectory of the training data to explore the internal input-neuron space extensively. In our formulation, we judiciously choose the signal shape of the generic excitation $v_E(t)$ that leads to different signal patterns of $v_o(t)$. As a special case in our formulation, if the rational function $D(s)$ represents a short circuit, *i.e.*, $a_0 = \infty$ with other coefficients being zero, then $v_E(t)$ is directly applied to the output port of the original circuit, *i.e.*, $v_o(t) = v_E(t)$. By sweeping the coefficients of $v_E(t)$, *i.e.*, $V_i, i = 1, 2, \dots, m+1$, different combinations of signal levels and slopes are created corresponding to various points in the subspace of $v_o(t)$ and its time-delayed signals. When there is no recurrency in the RNN model, *i.e.*, $n = 0$, sweeping the parameters of $v_E(t)$ alone can adequately provide training data to cover the internal input-neuron space. However, when recurrency exists, *i.e.*, $n > 0$, the internal input-neuron space $\mathbf{x}$ includes not only $v_o(t)$ and its history but also the history of $i_o(t)$. In this case, it may not be easy to control $v_o(t)$ and $i_o(t)$ simultaneously by only sweeping the parameters of $v_E(t)$. In this work, we propose to use a generic load together with $v_E(t)$ to provide different combinations of voltages and currents to enrich the training information.

The generic load (rational function) $D(s)$ in our formulation provides a means to alter the relationship of $v_o(t)$, $i_o(t)$, and their time derivatives. The rational function $D(s)$ is a reduced order representation of a general passive termination of the original circuit. As a special case, when $v_E(t) = 0$, by sweeping the coefficients $\mathbf{a}$ and $\mathbf{b}$ of $D(s)$, the proposed formulation can account for a wide range of passive terminations. With the combination of the rational function $D(s)$ and the excitation $v_E(t)$, the region of interest in the internal input-neuron space can be systematically explored, leading to a comprehensive set of training data for developing RNN models. Let $\mathbf{p}$ be the vector containing all the parameters in the proposed generic load and excitation

$$\mathbf{p} = [a_0 \ a_1 \ \dots \ a_m \ b_1 \ b_2 \ \dots \ b_n \ V_1 \ V_2 \ \dots \ V_{m+1}]^T. \quad (4.10)$$

By sweeping the parameters in $\mathbf{p}$, we can systematically generate different port voltage and current waveforms from the original nonlinear circuit. Such waveforms become the candidate data for training RNN models. Notice that the slope of the piece-wise signal $v_E(t)$ at each sampling interval h can be computed based on the signal amplitudes at two consecutive time instants. Therefore, by individually sweeping the amplitudes of the piece-wise signal $V_i, i = 1, 2, \dots, m+1$, the slope information is indirectly supplied to the proposed training approach for developing recurrent-neural-network models.

Figure 4.3: Illustration of the hyper-plane realized by the proposed formulation in the internal input-neuron space. (a) $\hat{a}_0 = 0.02$, $\hat{a}_1 = 0.01$, $\hat{b}_1 = 0.1$, and $v_E(t) = 2$. (b) $\hat{a}_0 = 0.02$, $\hat{a}_1 = 0.07$, $\hat{b}_1 = 0.6$, and $v_E(t) = -2$. By sweeping the coefficients in the proposed formulation, the hyper-plane of (4.7) is allowed to have different angles and positions in the internal input-neuron space.

For steady-state large signal modeling, the proposed formulation as shown in Figure 4.2 can be applied in the following way. We define a generic excitation $v_E(t)$ in the form of a multiple harmonic excitation to excite the circuit output port. Let ω_E represent the fundamental frequency of the generic excitation. Let R_0 and R_i represent the voltage magnitude of the generic excitation at dc and the i^{th} harmonic frequency, $i = 1, 2, \dots, N_f$, respectively, where N_f is the number of harmonics to be considered. By changing R_i , $i = 0, 1, \dots, N_f$, and ω_E , we can conveniently adjust $v_E(t)$ and its time derivatives such that the hyper-plane in (4.7) is positioned differently in the internal input-neuron space. In the case of large signal modeling, the list of parameters in the proposed generic load and excitation is given by

$$\mathbf{p} = [a_0 \ a_1 \ \dots \ a_m \ b_1 \ b_2 \ \dots \ b_n \ R_0 \ R_1 \ \dots \ R_{N_f} \ \omega_E]^T. \quad (4.11)$$

By sweeping the parameters in $\mathbf{p}$, we can control the hyper-plane in (4.7) to explore different parts in the internal input-neuron space. After we have obtained the frequency-domain spectrum data of port voltages and currents, inverse Fourier transform will be applied to convert the frequency-domain spectrums into the time-domain waveforms that become the training data for RNN modeling.

In summary, by sweeping a predefined set of parameters in the proposed formulation, we can systematically generate a rich combination of training signal patterns to achieve good coverage of the internal input-neuron space. Consequently, the model trained using

such effective data will have better generalization capabilities for a variety of test loads and test waveforms.

As more parameter sweeps produce more useful signal patterns, this process may also lead to a large amount of training data. If such raw data were directly applied for training RNN models, the training would take a long time and/or consume a large amount of memory. To improve efficiency, a reduction method will be presented in the next subsection to prune the raw training data by removing the unnecessary portion of samples.

4.3.3 Data Reduction Method

Each training waveform generated by following the proposed formulation can be converted to a set of training samples $(\tilde{\mathbf{u}}, \mathbf{y})$ corresponding to the inputs and outputs of the feedforward neural network, respectively. To obtain a sample of $\tilde{\mathbf{u}}$, we extract a short segment from each waveform of $v_i(t)$, $v_o(t)$, and $i_o(t)$. We then place the segments at the appropriate locations in the vector $\tilde{\mathbf{u}}$ according to (4.3). The length of the segments is determined by the RNN orders m and n . Different samples of $\tilde{\mathbf{u}}$ can be obtained from the same waveform by extracting the segments at different time instants in the waveform. Different segments of circuit waveforms carry the information of different signal levels and slopes. If two segments within the same waveform or in separate waveforms lead to the same or nearly the same $\tilde{\mathbf{u}}$ values, then one of them is considered redundant and can be removed. Our reduction method is formulated to eliminate the redundant waveform segments, *i.e.*, redundant training samples, while retaining the necessary modeling information about the original circuit.

To perform data reduction, we first identify the locations of redundant training samples in the internal input-neuron space. Let $\mathbf{R}$ represent the region of interest in the internal input-neuron space. Let $\mathbf{J}^L = [J_1^L \ J_2^L \ \dots \ J_{\tilde{N}_u}^L]^T$ and $\mathbf{J}^U = [J_1^U \ J_2^U \ \dots \ J_{\tilde{N}_u}^U]^T$ represent the lower and upper bounds of the region $\mathbf{R}$, respectively. To detect the redundant samples, we propose to divide the overall region $\mathbf{R}$ into sub-regions by splitting the desired range for each input $\tilde{u}_i$ of the feedforward neural network, *i.e.*, $[J_i^L, J_i^U]$, into q_i sub-intervals, $i=1,2,\dots,\tilde{N}_u$. Let $\mathbf{q} = [q_1 \ q_2 \ \dots \ q_{\tilde{N}_u}]^T$ represent the number of sub-intervals for all the input dimensions of the feedforward neural network. For each training sample $(\tilde{\mathbf{u}}, \mathbf{y})$, we analyze each input dimension to determine the location of the present sample in the internal input-neuron space:

$$\left\{ \begin{array}{l} \rho_i = l, \text{ if } \frac{(l-1)(J_i^U - J_i^L)}{q_i} + J_i^L \leq \tilde{u}_i \leq \frac{l(J_i^U - J_i^L)}{q_i} + J_i^L \\ \rho_i = 0, \text{ otherwise} \end{array} \right. \quad l = 1, 2, \dots, q_i; i = 1, 2, \dots, \tilde{N}_u \quad (4.12)$$

where $\boldsymbol{\rho} = [\rho_1 \ \rho_2 \ \dots \ \rho_{\tilde{N}_u}]^T$ is an index vector representing the position of the sub-region containing the present training sample. Each ρ_i represents an index number between 1 and q_i that indicates a unique sub-interval for $\tilde{u}_i$ at the i th input dimension of the feedforward neural network, $i = 1, 2, \dots, \tilde{N}_u$, *i.e.*,

$$\tilde{u}_i \in \left[\frac{(J_i^U - J_i^L)(\rho_i - 1)}{q_i} + J_i^L, \frac{(J_i^U - J_i^L)\rho_i}{q_i} + J_i^L \right]. \quad (4.13)$$

If $\rho_i = 0$, then it means that $\tilde{u}_i$ is outside the predefined range $[J_i^L, J_i^U]$. Once the locations of redundant data are determined, the next step is to eliminate the redundant samples such that each sub-region only retains a minimum number of training samples.

Let N_r represent the total number of sub-regions in R , which is calculated by $N_r = \prod_{i=1}^{\tilde{N}_u} q_i$.

Let ε represent the maximum number of training samples that is allowed per sub-region.

The present training sample $(\tilde{u}, y)$ is accepted if it is within one of the sub-regions and the present number of training samples in that particular sub-region is no more than ε , *i.e.*,

$$\begin{cases} \rho_i > 0 \text{ for all } i, i = 1, 2, \dots, \tilde{N}_u \\ N_s \leq \varepsilon \end{cases} \quad (4.14)$$

where the subscript s denotes the index number of the sub-region occupied by the present sample, computed by

$$s = \rho_{\tilde{N}_u} + \sum_{i=1}^{\tilde{N}_u-1} (\rho_i - 1) \prod_{l=i+1}^{\tilde{N}_u} q_l \quad (4.15)$$

and N_s represents the present number of training samples in the sub-region s . On the other hand, if any of the conditions in (4.14) are violated, the training sample $(\tilde{u}, y)$ is considered redundant and excluded from the training data set.

The entire set of sub-regions is now divided into two groups: empty sub-regions, which are not occupied by any training sample, and sub-regions occupied by training samples. To measure the quality of training data, we generate different test data based on a variety

of circuit loads. Using test data instead of training data, we also identify the group of sub-regions occupied by test samples following (4.12). Let I_T and I_V represent the index set for the group of sub-regions occupied by training and test samples, respectively. To test the effectiveness of the training data, we define two quality metrics Q_{all} and Q_{te} representing the coverage rate of the internal input-neuron space during and after data generation, respectively, as

$$Q_{all} = \frac{size(I_T)}{N_r} \quad (4.16)$$

$$Q_{te} = \frac{size(I_T)}{size(I_V)}. \quad (4.17)$$

When sweeping the parameters of the proposed formulation, Q_{all} is periodically calculated to monitor if the newly generated training data cover the sub-regions that are still empty. Q_{te} indicates how well the final training data cover the region of test data. If Q_{te} is close to 100 per cent, then the training data are ready to be used for developing RNN models.

4.3.4 Discussion

For high-speed I/O drivers, the magnitude of the circuit output voltage $v_o(t)$ is often limited by the power supply voltages V_{dd} and V_{ss} . Since $v_o(t)$ is one of the input dimensions of the feedforward neural network, the boundaries of the training region in the internal input-neuron space are dependent on the circuit power supply voltages. If the

training data with fixed power supply voltages were used for RNN training, the model accuracy near the boundaries of the training region would be limited. To enhance the model accuracy and robustness, we add the power supply voltages V_{dd} and V_{ss} into the list of parameters $\mathbf{p}$ in (4.10) to be swept during data generation. In this way, training data containing rich information can be generated to enhance the model accuracy near the boundaries of the training region.

In the proposed technique, sweeping the coefficients of the generic excitation $v_E(t)$ offers different combinations of signal levels and slopes for port voltages. In case if a specific circuit simulator does not allow the direct implementation of a piece-wise linear source at finite number of time points, we may sweep the parameters of the available time-domain source, such as a trapezoidal source, to generate different signal patterns of port voltages. In this case, the proposed data reduction method helps to select the useful segments and remove the redundant segments from various candidate waveforms.

For practical considerations, the proposed generic load in terms of transfer function $D(s)$ should preserve both stability and passivity. The stability of the network can be conveniently verified by checking if the poles of $D(s)$ fall in the left half of the complex plane. Passivity means that the network should not generate any new energy. In this work, we adopt the criterion in [150] to verify the passivity of $D(s)$, which involves computing the eigenvalues of the Hamiltonian matrix. Each set of values for $D(s)$ coefficients $\mathbf{a}$ and $\mathbf{b}$ is subject to stability and passivity verifications prior to the actual data generation. If any stability or passivity violation is detected, the present values of $\mathbf{a}$ and $\mathbf{b}$ will be skipped.

With the increase of the number of coefficients $\mathbf{p}$ in the proposed training approach, uniform sweeping of each coefficient in $\mathbf{p}$ may become inefficient. In this work, we use adaptive sampling method to select the values for coefficients $\mathbf{p}$ within their sweeping ranges. For a given region of the coefficients $\mathbf{p}$ in (4.10) or (4.11), we split the whole region into smaller areas and carry out data generation individually for all of the areas. Based on the training data for each area of $\mathbf{p}$, we evaluate and record the quality metric Q_{all} following (4.16). Because Q_{all} indicates the effectiveness of training data, we judiciously use Q_{all} to guide the adaptive sampling method to explore different area of $\mathbf{p}$. During data generation, if Q_{all} increases over its previous value, we will choose dense samples of $\mathbf{p}$ in the current area. Otherwise, sparse samples will be used to speed up data generation. In this way, our method selects an effective set of values for the coefficients $\mathbf{p}$ in the proposed formulation and subsequently reduces the CPU cost of data generation.

The proposed data reduction method is specially tailored to address our modeling case where a large amount of training data are required in order to cover sufficiently the internal input-neuron space. Without data reduction, a brute-force approach could be to use blindly all the available training data generated by sweeping the list of coefficients in (4.10) or (4.11). Because the size of the training data may be large, and the training data may be unevenly distributed in the internal input-neuron space, training a neural network to learn such data could be a challenging task, requiring considerable time and effort to obtain a reliable neural model. Through the detection and elimination of the redundant data, the proposed data reduction method yields a smaller yet effective set of data, thus

improving the efficiency and robustness of the training process. Compared with the existing iterative approaches for data reduction, such as vector quantization and clustering, the proposed method provides a more simplified way to reduce the data size and keep a uniform distribution of the reduced data in the regions that are occupied by data samples. This allows the RNN to learn the information about various parts of the internal input-neuron space, using only a small amount of training data.

4.3.5 Overall Training Algorithm

The overall algorithm for our proposed training approach can be described as follows:

- Step 1: Initialize the orders m and n of the RNN model. Based on the selected model orders, define the voltage and current variables in the internal input-neuron space.
- Step 2: Define the region of interest $\mathbf{R}$ in the internal input-neuron space by selecting desired lower and upper bounds for each input dimension of the feedforward neural network, *i.e.*, $\mathbf{J}^L$ and $\mathbf{J}^U$. Divide the region $\mathbf{R}$ into N_r sub-regions based on the number of sub-intervals q .
- Step 3: Based on the model orders, determine the number of parameters to be swept in the proposed formulation. Set the sweeping range for each parameter of $\mathbf{p}$ in (4.10) or (4.11). Set the iteration count to $i = 1$.
- Step 4: Determine a new set of $\mathbf{p}$ values according to uniform sweeping or the sampling method discussed in Chapter 4.3.4. Perform stability and passivity verifications

of the generic load for the current set of $\mathbf{a}$ and $\mathbf{b}$ values. If the stability and passivity are preserved, proceed to the next step. Otherwise, repeat Step 4.

- Step 5: Perform data generation with the proposed formulation using the current set of $\mathbf{p}$ values. Update the index set I_T using the newly obtained training waveforms. Compute Q_{all}^i using (4.16) where the superscript i is the iteration count. Compare the present Q_{all}^i with its history at previous iterations. If Q_{all}^i does not improve for several consecutive iterations, *i.e.*, $Q_{all}^i = Q_{all}^{i-l}$ for several values of l ($l < i$), then terminate the data generation and proceed to the next step. Otherwise, set $i = i + 1$ and repeat Step 4.
- Step 6: Using (4.12), (4.14), and (4.15), perform the proposed data reduction on the set of waveform training data generated in Steps 4 and 5. Generate test data from the original circuit based on a variety of test loads. Use the test data to determine the index set I_V .
- Step 7: Validate the effectiveness of the final training data by calculating Q_{te} based on (4.17). If Q_{te} is close to 100 per cent, then proceed to the next step. Otherwise, repeat Step 1 with the modified orders of the RNN model.
- Step 8: Using the final training data from Step 6, train the feedforward neural network with the standard training method, such as Quasi-Newton method [1]. Optionally, using the original waveform data in Steps 4 and 5, further refine the accuracy of the RNN model with conventional recurrent training based on BPTT [5], [70].

Figure 4.4 shows a flowchart illustrating the procedure of the proposed method for developing RNN models for nonlinear circuits.

Figure 4.4: A flowchart illustrating the overall algorithm of the proposed training approach for developing RNN models of nonlinear circuits.

After the training of the RNN model, we further validate the model accuracy using an independent set of test data that is not used during the model training. Next, to facilitate the direct use of the trained RNN model in a circuit simulator, an equivalent circuit representation of the model is implemented, following a similar procedure as described in [82]. Although the sampling time interval h is not directly used as a parameter in the training of the RNN, it plays a role during the process of converting a discrete-time RNN model into a continuous-time equivalent circuit. In particular, during the conversion process, h is used to determine the values of the dynamic elements (such as capacitors or inductors) in the circuit representation. Once the continuous-time equivalent circuit is obtained, the RNN model can be applied in various circuit simulations, even if the time step size used in these simulations is different from the h used in training.

4.4 Demonstration Examples

4.4.1 Transient Behavioral Modeling of a High-Speed Driver

This example demonstrates the application of the proposed method to the transient modeling of a high-speed driver, which is a 130 nm High-Speed Transceiver Logic (HSTL) I/O cell, featuring IBM CU-11 technology. The driver circuit is composed of a differential amplifier input buffer and a push-pull output buffer. I/O driver models are useful for the design of high-speed VLSI packages and interconnects. The original model of the driver is an encrypted transistor-level model in HSPICE [18], which is accurate but slow. Here we aim to develop a fast behavioral model to speed up SI-based analysis and design. We demonstrate that by training an RNN to learn the input and output waveforms

of the driver, a fast RNN behavioral model can be developed, even though the internal details of the original driver are not available. In this example, the input and output orders of the RNN model are $m=2$ and $n=1$, respectively. Correspondingly, the internal input-neuron space $\tilde{\mathbf{u}}$ is defined as

$$\tilde{\mathbf{u}} = [v_i(t) \ v_i(t-h) \ v_i(t-2h) \ v_o(t) \ v_o(t-h) \ v_o(t-2h) \ i_o(t-h)]^T. \quad (4.18)$$

The detailed transistor-level model of the driver is used as the reference model. For data generation, the driver input port is excited by a pulse signal with a rise/fall time of 0.2 ns and a pulse width of 2.3 ns. The sampling interval h that is used in this example is 0.02 ns. Following the proposed technique in Chapter 4.3.5, the training data are systematically generated by sweeping the coefficients $\mathbf{p}$ in (4.10). Specifically, uniform sweeping of $v_E(t)$ coefficients V_1, V_2 , and V_3 is performed where each coefficient varies from -0.2 V to 2.2 V, with step-size 0.2 V. In addition, the sampling method discussed in Chapter 4.3.4 is applied to sweep $D(s)$ coefficients a_0 and b_1 from 0.01 to 1 and from 0.1 to 10, respectively. The b_1 values are relative to the frequency variable in GHz. The other coefficients of $D(s)$ are set to be zero. Furthermore, power supply voltages V_{dd} and V_{ss} are also adjusted around their nominal values, *i.e.*, 1.8 V and 0 V, respectively.

After data generation is complete, we then apply our proposed data reduction method to eliminate the redundant data. During the reduction, the region of training data in the internal input-neuron space is split into smaller sub-regions by using five sub-intervals for $v_i(t)$, $v_i(t-h)$ and $v_i(t-2h)$, and 20 sub-intervals for the remaining inputs in $\tilde{\mathbf{u}}$, *i.e.*,

$q = [5 \ 5 \ 5 \ 20 \ 20 \ 20 \ 20]^T$. The maximum number of training samples that is allowed per sub-region is five, *i.e.*, $\varepsilon = 5$. Through the proposed data reduction, the size of the training data is significantly reduced from 128500 to 31512. To validate the quality of training data, various test data are also generated when the device is terminated with quite different linear and nonlinear loads. The linear load consists of a lossless transmission line (characteristic impedance $Z_0 = 50 \ \Omega$ and line delay T_d ranging from 0.1 ns to 0.5 ns, with step-size 0.1 ns) and a 1-pF capacitor. The nonlinear load is composed of a lossy transmission line, a via, and a nonlinear receiver. The length of the lossy transmission line is swept from 1000 mil to 4000 mil to create different sets of test data. Using the test data for both the linear and nonlinear loads, the quality of the training data is confirmed with $Q_{te} = 99.32\%$, which means that over 99 per cent of the sub-regions containing test data are covered by the training data. Figure 4.5 shows the sample distribution of the training data from the proposed method and the test data in the internal input-neuron space where good coverage of the region of test signals with various shapes is achieved. Subsequently, we train an RNN model with the method in Chapter 4.3.5 and the final average test error is 0.84% for various test data generated using both linear and nonlinear loads.

Figure 4.5: Sample distribution of the training data generated with the proposed technique (·) and the test data (—) in the high-speed driver example. Good coverage of the internal input-neuron space allows this model to respond accurately to test signals of unknown shapes.

For comparison purposes, two types of existing methods, *i.e.*, physical load sweeping and physical source sweeping, are also applied to develop RNN models. In this work, we use the terms physical source and physical load to refer to the type of sources and loads that are used in the normal operation of the original circuit being modeled. For the method of physical load sweeping, a capacitor load (capacitor value ranging from 1 pF to 10 pF, with step-size 1 pF) and a lossless transmission line load ($Z_0 = 50 \, \Omega$ and T_d ranging from 0.1 ns to 0.5 ns, with step-size 0.1 ns) have been used separately to generate

different sets of training data. To perform data generation with the method of physical source sweeping, we use a transient source of ramp signals at the driver output port, which is defined by the low dc level V_{start} , the high dc level V_{stop} , and the rise/fall time T_r . During data generation, the source parameters are individually swept. Specifically, T_r varies from 0.1 ns to 5 ns, with step-size 0.5 ns. V_{start} and V_{stop} vary from $-3V_{dd}$ to $4V_{dd}$, with step-size $V_{dd}/10$ and $V_{dd}=1.8$ V. Using the training data from these existing methods, three RNN models are subsequently developed. Table 4.1 summaries the average model test accuracies for the test data under different driver loads. For testing with a simple capacitor load, the existing and new methods both produce good results. However, as the test load becomes different from that in training, the proposed method leads to more accurate RNN models than those from the existing methods.

Next, we verify the performance of the RNN models through simulations for different load effects not considered in training. In Figure 4.6, the output voltages of the driver are shown for two different lengths of the transmission line in the nonlinear load. The test data used here have completely different shapes compared to the training data. It is observed that the RNN model trained with the proposed technique provides accurate predictions of signal variations.

Table 4.1: Comparison of the Test Accuracy of the RNN Model between the Existing and Proposed Methods for Various Test Loads in the High-Speed Driver Example

Test load	Average model test error			
	Model trained with the existing method (physical load sweeping)		Model trained with the existing method (physical source sweeping)	Model trained with the proposed technique
	Capacitor load	Transmission line load		
Capacitor load	0.11%	1.38%	1.19%	0.64%
Linear load	3.67%	0.86%	8.57%	0.72%
Nonlinear load	5.74%	4.25%	6.41%	1.06%

As an additional verification, the RNN model is used for crosstalk simulation with coupled transmission lines. In this test setup, a lossy two-conductor transmission line is driven by two high-speed drivers from IBM. One driver is an active driver, which is excited with a pulse signal whereas the other driver is a quiet driver, which is set idle at 0V. The coupled transmission line is terminated with 1 pF capacitors. As shown in Figure 4.7, the responses of the RNN model trained with the proposed method have a good agreement with the test waveforms of crosstalk signals. This is achieved even though the test load is quite different from that in training.

(a)

(b)

Figure 4.6: Output voltages of the high-speed driver for different nonlinear loads. (a) The nonlinear load contains a transmission line with length of 2000 mil. (b) The nonlinear load contains a transmission line with length of 4000 mil. The proposed method leads to accurate RNN models for a variety of test loads that are never applied in training.

Figure 4.7: Crosstalk signals at the output port of the quiet driver in the high-speed driver example. The proposed technique enables the RNN model to produce accurate results of the crosstalk signals that are never seen during training.

As a further test of the model performance under the test signals that differ from the training signals, we use the RNN model trained with the proposed technique for the SI simulation of a complete signal propagation path. The entire signal path includes a nonlinear driver, a lossy transmission line, a via, and a nonlinear receiver. In this test, the driver input excitation is a $2^7 - 1$ pseudorandom binary sequence with bit rate 2.5 Gb/s. The simulation is carried out for the duration of 1000 input bits and the eye diagram generated by the RNN model agrees with that of the transistor-level model. The CPU times for the various test cases are summarized in Table 4.2, where the RNN model runs

2-3 times faster than the transistor-level model for various SI simulations. The RNN model, trained with the proposed training approach, is a good alternative to the detailed transistor-level model in terms of speed and accuracy. For this particular example, since the complexity of the original circuit to be modeled is relatively low, the CPU speed-up of the RNN model is limited. However, as circuit complexity or the level of simulation details increases, the CPU speed-up would become more significant by using the RNN model. When a very detailed simulation is required for the original model, such as in the case of a physics-based model, the recurrent-neural-network model offers a substantial speed-up of CPU compared to the original model, as we will demonstrate in the four-stage driver example.

Table 4.2: CPU Comparison between the Detailed Transistor-Level Model and the RNN Model Trained with the Proposed Technique in the High-Speed Driver Example

Driver input	Test load	CPU time	
		Original detailed transistor-level model	RNN model trained with the proposed technique
One pulse	Linear	6.4 seconds	2.1 seconds
One pulse	Nonlinear	12.3 seconds	3.9 seconds
Pseudorandom binary sequence of 1000 bits	Nonlinear	465.9 seconds	224.7 seconds

4.4.2 Large Signal Behavioral Modeling of an Amplifier

This example demonstrates the application of the proposed method to the large signal behavioral modeling of an RF amplifier based on RNNs. The original amplifier, consisting of nine transistors, is a detailed transistor-level nonlinear circuit from Agilent ADS [17], as shown in Figure 4.8. In this example, the input and output orders of the RNN model are $m = 2$ and $n = 2$, respectively. Based on the orders of the model, the internal input-neuron space $\tilde{\mathbf{u}}$ is defined as

$$\tilde{\mathbf{u}} = [v_i(t) \ v_i(t-h) \ v_i(t-2h) \ v_o(t) \ v_o(t-h) \ v_o(t-2h) \ i_o(t-h) \ i_o(t-2h)]^T. \quad (4.19)$$

The training data for the model are generated from ADS simulations by using the proposed technique presented in Chapter 4.3.5. The amplifier input port is connected to a power source with the input power level P_{in} ranging from -20 dBm to 10 dBm, and the fundamental frequency ω_E ranging from 800 MHz to 1200 MHz. Sweeping of the coefficients $\mathbf{p}$ in (4.11) is carried out where each of the coefficients a_0, a_1, b_1, a_2 , and b_2 is swept from 0 to 1 using the sampling method discussed in Chapter 4.3.4. In addition, $v_E(t)$ coefficients R_1 and R_2 are uniformly swept from 0 V to 1 V with step-size 0.1 V, and from 0 V to 0.1 V with step-size 0.01 V, respectively. The spectrum data generated from ADS simulations are converted to time-domain waveforms with 50 samples per waveform.

Figure 4.8: The RF amplifier circuit to be modeled by RNNs.

After the data generation is finished, we apply the proposed data reduction algorithm to eliminate the redundant data. During this process, we divide the region of training data into smaller sub-regions with ten sub-intervals for each input dimension of the feedforward neural network. A maximum of five samples is allowed for each sub-region, *i.e.*, $\varepsilon = 5$. Through the proposed reduction method, the size of training data is significantly reduced from 2650 000 to 21846. To test the quality of the training data, different test data are also generated with various linear and nonlinear loads of the amplifier. Specifically, the linear complex load is implemented as a complex impedance Z_L with its real part ranging from 25 to 75 and its imaginary part ranging from -1 to 1 . The nonlinear load is implemented as a nonlinear resistor whose characteristic is

represented by a second-order polynomial function $i = 0.02v + \alpha v^2$ with α ranging from -0.3 to 0.3 . Using these test data, the coverage rate of the internal input-neuron space, *i.e.*, Q_{te} in (4.17), is 97.8%, which means that over 97 per cent of the sub-regions containing test data are covered by the training data. Figure 4.9 shows the sample distribution of the training data and various test data in the internal input-neuron space. As shown in the figure, the proposed method leads to a systematic set of training data that can completely cover the region of various test data whose shapes are not known during training. An RNN model is subsequently developed with the method in Chapter 4.3.5 and the final model test error is 0.58% based on various test waveforms that are never applied in training.

For comparison purposes, we also apply the existing method based on physical source sweeping to produce different training data. In this case, we apply power sources with various power levels and fundamental frequencies at the input and output ports of the amplifier. Specifically, the input and output powers are swept from dc to 10 dBm and the fundamental frequency varies from 800 MHz to 1200 MHz. Such training data from the existing method are subsequently used for developing an RNN model. In Table 4.3, comparisons of different RNN models are made in terms of the coverage rate of the internal input-neuron space (Q_{te}) and the model test accuracy. As can be observed, when the test data are based on the linear loads (resistive or complex loads), both the existing and proposed methods provide good coverage rate and model test accuracy. However, when the nonlinear load is applied, the proposed method leads to higher coverage rate and better model accuracy compared to the existing method.

Figure 4.9: Sample distribution of training data using the proposed technique (·) and the test data (—) in the internal input-neuron space for the amplifier example. Through the proposed technique, the training data can sufficiently cover the region of the test waveforms whose shapes are not known during training.

As a further validation, we implement the RNN model into ADS software and use the model in HB simulation under various test loads that differ from those in training. Figure 4.10 shows the comparison of amplifier output power levels for one-tone signals, where the responses of the RNN model have a good match with those obtained from the original circuit. In addition, we investigate the reliability of the RNN model during the system-level simulation. For comparison purposes, we also use the existing training method of physical load sweeping to develop different RNN models at various input power levels.

In this case, we use a resistor load, which is swept from $10\ \Omega$ to $100\ \Omega$. We then test the model performance in HB simulation using the RNN models trained with the existing and proposed methods. As shown in Table 4.4, the proposed technique yields a better model convergence rate especially at higher power levels, confirming that the proposed method improves the model reliability in large signal HB simulation.

Table 4.3: Comparison of the Coverage Rate of the Internal Input-Neuron Space (Q_{te}) and the Model Test Accuracy for Different Test Loads in the Amplifier Example

Test load	Q_{te}		Model average test error	
	The existing method (physical source sweeping)	The proposed technique	Model trained with the existing method (physical source sweeping)	Model trained with the proposed technique
Resistive load	100%	100%	0.22%	0.36%
Complex load	91.2%	99.4%	1.09%	0.44%
Nonlinear load	69.8%	96.1%	3.77%	0.89%

Figure 4.10: The amplifier output power versus input power at different loads and fundamental frequencies. (a) $Z_L = 40 - j$ and $\omega_E = 950$ MHz. (b) $Z_L = 70 + 0.5j$ and $\omega_E = 1050$ MHz. Good agreements are achieved between the RNN model trained with the proposed technique and the original amplifier even though such test loads are never considered during training.

Table 4.4: Convergence Rate of the RNN Model in HB Simulation for Various Input Power Levels and Resistor Loads in the Amplifier Example

		Model convergence rate	
		Model trained with the existing method (physical load sweeping)	Model trained with the proposed technique
Input power level	- 14 dBm	100%	100%
	- 5 dBm	85%	100%
	5 dBm	70%	100%

We also examine the performance of the RNN model trained with the proposed technique under two-tone test signals. In this case, the amplifier is excited by a two-tone power source with the power level at each tone swept from -10 dBm to 5 dBm, with step-size 1 dBm and frequency spacing swept from 30 kHz to 70 kHz , with step-size 20 kHz. Figure 4.11 shows the comparisons of the third-order intermodulation (IMD3) between the original amplifier circuit and the RNN model using the proposed technique. As can be observed, for different complex loads that are not applied in training, the IMD3 results of the RNN model agree well with those based on the original circuit.

(b)

Figure 4.11: Output IMD3 power of the amplifier circuit for different test loads: (a) Lower side IMD3, (b) Upper side IMD3. The frequency spacing of input signals is 50 kHz. Accurate output IMD3 results are obtained using the RNN model trained with the proposed technique, even though the test loads are never seen in training.

As an additional verification, we apply the RNN model trained with the proposed technique to spectrum regrowth simulation under modulated test signals. For this test, the amplifier is driven by a $\pi/4$ differential quadrature phase shift keying (DQPSK) input signal with input power at 1-dB gain compression point and center frequency of 950 MHz. The channel bandwidth (chip rate) is 24.3 KHz. Different complex loads have been used for this test. As shown in Figure 4.12, the RNN model trained with the proposed technique can also predict the spectrum regrowth under modulated signals. It is to be noticed that the time step of the modulated signal does not have to be the same as the sampling time h used in the model development. In fact, because the trained RNN model has been converted into a continuous-time equivalent circuit in ADS, the model can be used in the environment of modulated signals, regardless of the time step value used during the circuit simulation. To generate the power spectrum in Figure 4.12(a), the detailed amplifier circuit requires 5.69 seconds whereas the RNN model requires only 0.98 seconds, which shows the efficiency of the RNN model in the system-level simulation. The proposed technique can be used to develop large signal steady-state RNN models that are accurate for a variety of linear and nonlinear loads and flexible for different simulation needs, such as one-tone, two-tone, and modulated signals.

(a)

(b)

Figure 4.12: The output power spectrum of the amplifier circuit under modulated test signals. Different test loads are used for this validation: (a) $Z_L = 45 + j$. (b) $Z_L = 65 + j$. By using the proposed technique, accurate spectrum regrowth predictions of the RNN model are achieved for different amplifier loads that are never used in training.

4.4.3 Transient Behavioral Modeling of a Four-Stage Driver from a Physics-Based Simulator

In this example, we demonstrate the proposed training approach through the RNN-based transient modeling of an I/O driver from a physics-based simulator. The original circuit is a four-stage driver composed of eight complementary metal oxide semiconductor transistors [151], as shown in Figure 4.13. Each transistor is implemented using the detailed physics-level model from a semiconductor simulator MINIMOS-NT [144]. Here we aim to use RNNs for fast behavioral modeling of this driver. The input and output orders of the RNN model used in this example are $m=1$ and $n=1$, respectively. Based on the orders of the model, the internal input-neuron space $\tilde{\mathbf{u}}$ is defined as $\tilde{\mathbf{u}} = [v_i(t) \ v_i(t-h) \ v_o(t) \ v_o(t-h) \ i_o(t-h)]^T$. The driver input port is excited by a pulse signal with a rise/fall time of 0.5 ns and a pulse width of 2 ns. The sampling interval h that is used in this example is 0.05 ns.

Following the proposed technique presented in Chapter 4.3.5, the training data are generated by sweeping the coefficients $\mathbf{p}$ in (4.10). In particular, a_0 , a_1 , and b_1 are swept from 0.01 to 1, 0.001 to 0.1, and 0.1 to 10, respectively, using the sampling method discussed in Chapter 4.3.4. Since the direct implementation of $v_E(t)$ is not feasible in MINIMOS-NT, we sweep the parameters of the available time-domain source (pulse source) to generate different patterns of voltage signals. Specifically, the pulse amplitude and rise/fall time are uniformly swept from -1 V to 6 V, with step-size 0.5 V, and from 0.25 ns to 0.75 ns, with step-size 0.25 ns, respectively. Furthermore, power supply voltages V_{dd} and V_{ss} are also adjusted around their nominal values, *i.e.*, 5.0 V and 0 V,

respectively. To confirm the quality of the data obtained using the proposed technique, different test data are also generated based on various test loads of the CMOS driver. In this case, the driver load consists of a lossless transmission line with line parameters as $L=360$ nH/m and $C=100$ pF/m, and a capacitor of 5 pF. The transmission line length varies between 1 cm and 3 cm to create different test waveforms.

Figure 4.13: The four-stage CMOS driver circuit to be modeled by an RNN. The data generation is carried out by driving a physics-based simulator MINIMOS-NT.

After data generation is complete, we apply the proposed data reduction method to minimize the size of training data. During the reduction, the region of training data in the internal input-neuron space is divided into smaller sub-regions by using five sub-intervals for $v_i(t)$ and $v_i(t-h)$, and 20 sub-intervals for the remaining inputs in $\tilde{\mathbf{u}}$, *i.e.*, $\mathbf{q} = [5 \ 5 \ 20 \ 20 \ 20]^T$. The maximum number of data samples that is allowed for each sub-region is three, *i.e.*, $\varepsilon = 3$. By using the proposed reduction algorithm, the size of the actual training data is reduced from 68500 down to 2677, showing the effectiveness of

our data reduction method. The quality of the final training data is confirmed with $Q_{te} = 96.55\%$, computed using (4.17), which means that over 96 per cent of the sub-regions containing test data are covered by the training data with the proposed technique. Figure 4.14 shows the sample distribution of the final training data and the test data in the internal input-neuron space. It is demonstrated that the region of test data, whose shapes are not known during training, is sufficiently covered by the training data from the new training approach. Subsequently, we train an RNN model with the method in Chapter 4.3.5 and the final model test error is 0.77% for various test data not applied in training.

For comparison purposes, we develop two additional RNN models using the data from the existing methods. In particular, the first method is based on physical load sweeping where a capacitor load is used with its values ranging from 1 pF to 10 pF, with step-size 1 pF. The second method is based on physical source sweeping where a transient source with ramp signals is applied at the driver output port. The transient source is defined by the low dc level V_{start} , the high dc level V_{stop} , and the rise/fall time T_r . During data generation, the parameters of the physical source are individually swept. Specifically, T_r is swept from 0.01 ns to 5 ns, with step-size 0.5 ns. V_{start} and V_{stop} are swept from $-V_{dd}$ to $2V_{dd}$ with various number of sample points N_s and $V_{dd} = 5.0$ V. Table 4.5 compares the coverage rate of the internal input-neuron space, *i.e.*, Q_{te} in (4.17), among different training methods where the proposed method offers the highest overall coverage rate for the test data from various loads of the four-stage driver (capacitor loads and transmission line loads).

Figure 4.14: Sample distribution of the training data with the proposed technique (o) and the test data (—) in the internal input-neuron space for the four-stage driver example. The proposed technique leads to good coverage of the region of the test data whose shapes are not known during training.

We also test the accuracy of the RNN model using the test waveforms of various shapes. Table 4.6 shows the comparison of test accuracies of the RNN models between the existing and proposed methods. For a simple test load such as a capacitor load, the existing and new methods both produce accurate results. However, as the test load becomes different from that in training, the proposed training approach leads to the RNN model with better accuracy compared to the existing methods. Figure 4.15 shows driver output waveforms for the case of transmission line load. As observed in the figure, the

RNN model trained with the proposed technique maintains good accuracy even though the test waveforms are quite different from those in training.

Table 4.5: Comparison of the Coverage Rate $\mathcal{Q}_{te}$ between the Existing Methods and the Proposed Technique for Various Test Loads in the Four-Stage Driver Example

Test load	The existing method (physical load sweeping)	The existing method (physical source sweeping)			The proposed technique
		$N_s = 10$	$N_s = 25$	$N_s = 30$	
Capacitor	100%	82.1%	85.9%	88.5%	97.4%
Transmission line and capacitor	42.2%	30.3%	39.6%	40.2%	96.6%

Using the RNN model developed by the proposed method, we also perform the Monte Carlo analysis of the four-stage driver circuit with the length of the transmission line load as the statistical variable. For 50 random perturbations around the nominal value of the length parameter, the total simulation time with our RNN model is 2.45 minutes whereas the original physics-based simulation requires 2970 minutes, which shows a significant reduction of the CPU time using our model. The RNN model generated with our proposed technique is fast, accurate, and robust for handling various signal shapes and circuit terminations that are needed during high-level design.

Table 4.6: Comparison of the Test Accuracy of the RNN Model between the Existing Methods and the Proposed Technique for Different Test Loads in the Four-Stage Driver Example

Test load	Model average test error		
	Model trained with the existing method (physical load sweeping)	Model trained with the existing method (physical source sweeping)	Model trained with the proposed technique
Capacitor	0.08%	1.15%	0.31%
Transmission line and capacitor	5.85%	4.57%	0.77%

Figure 4.15: Output voltage of the four-stage driver for the transmission line loads with different lengths. Such test loads have not been used for training the RNN model.

4.5 Summary

We have presented a new training approach for developing accurate and robust RNN models of nonlinear circuits. The proposed technique systematically performs data generation targeted to developing transient or large signal models based on RNNs. By exploiting the internal input-neuron space, the proposed generic formulation has enabled the RNN model, which is trained with a limited set of training waveforms, to maintain good accuracy for test waveforms of unknown shapes. In addition, a data reduction method has been developed to enhance the model training efficiency by detecting and eliminating the redundant data. Examples have shown that through the proposed technique, the trained RNN models have good performance for various loads and signal shapes that are never seen in training. Compared to the existing training methods, the proposed technique provides a further advance in developing accurate and robust RNN models for efficient CAD of nonlinear high-frequency/high-speed circuits.

Chapter 5: Conclusions and Future Research

5.1 Conclusions

Recent advances in the RF and microwave industry have resulted in a dramatic increase in the complexities and functionalities of microwave circuits. Conventional design procedures became increasingly difficult with ever-growing circuit complexities coupled with tightened design tolerances. Computer-aided high-frequency design has emerged as a strong alternative to the conventional design procedure, which is cumbersome and computationally expensive. Modeling continues to be a major bottleneck for efficient CAD and optimization of RF/microwave components and circuits. The advanced high-frequency CAD algorithms that advocate the use of accurate and fast neural models in place of computationally prohibitive theoretical models have gained strong recognition in RF/microwave communities.

This thesis has presented state-of-the-art research in the neural-network-based high-frequency/high-speed CAD area, aimed at enhancing this emerging technology to higher level of maturity and broader areas of applications. The central objectives of the thesis have been the development of neural-network-based algorithms for efficient and robust transient modeling and simulation of nonlinear circuits in high-speed systems. It is envisioned that meeting these key objectives through focused and methodical research in

this area could enable the success of the proposed methodologies in terms of accuracy, cost-effectiveness, speed, and feasibility in practical CAD applications. Specifically, in view of the aforementioned objectives, the following significant contributions have been made:

Firstly, a new SSDNN technique has been developed for transient behavioral modeling of nonlinear circuits in high-speed applications [10]-[13]. The SSDNN nonlinear model is a more robust formulation than the conventional DNN-based models, improving the model reliability in the case where the training data are contaminated with high-frequency noise. The SSDNN nonlinear model can be developed directly from the input and output transient signals without requiring any internal knowledge of original nonlinear circuits. Training of the SSDNN has been realized in the time-domain by employing a new adjoint sensitivity technique for fast and accurate derivative calculations. The exact circuit representation of the SSDNN model has also been presented to allow the model to be conveniently incorporated into general SPICE-like circuit simulators.

Secondly, for the first time, the stabilities of the transient nonlinear models based ANNs have been addressed, which are crucial for ensuring the model reliability in circuit simulations. By following the concepts of Lyapunov methods, several stability criteria have been derived that facilitate convenient investigations of local and global stabilities based on the SSDNN model parameters. As a further step towards enhancing model robustness, an advanced constrained training scheme has been developed, enabling the development of the SSDNN models with good accuracies and guaranteed GAS. The

proposed constrained training algorithm has been demonstrated through the modeling of circuit-based and physics-based high-speed I/O buffers as well as a commercial packaged IC device.

Thirdly, a novel training approach has been presented for developing nonlinear transient models using RNNs, which are robust and accurate for versatile signal shapes and circuit loads [14]. By sweeping a predetermined set of parameters based on the orders of RNN models, the proposed training approach systematically performs data generation targeted to developing transient or large signal models based on RNNs. By allowing the generated data to cover extensively the internal input-neuron space, the proposed generic formulation has enabled the RNN model, which is trained with a limited set of training waveforms, to maintain high accuracy for test waveforms of unknown shapes. To facilitate efficient model training through the proposed approach, a data reduction method has been developed featuring smart detection and elimination of the redundant data samples. Through the proposed training approach, the trained RNN models have been shown to maintain good performance for various loads and signal shapes in both transient and large signal simulation environments, outperforming the existing training methods in terms of the model accuracy and robustness.

Finally, in order to accelerate the practical use of the algorithm proposed in this thesis, a generalized implementation in the form of general-purpose computer softwares is desirable. Object-oriented computer programs for the proposed ANN-based modeling techniques have been developed using C++ and/or MATLAB [15]. These programs have been used for obtaining the numerical results in this thesis and integrated into a trial

version of the NeuroModelerPlus software [16]. The research work presented in this thesis has made important contributions to the efficient and robust neural-network-based CAD of high-frequency/high-speed nonlinear circuits.

5.2 Further Directions

ANNs have been increasingly recognized as an efficient and flexible computational tool in RF/microwave and high-speed modeling, simulation and optimization. The CAD approaches based on ANNs, including those presented in this thesis, aim at achieving highest-levels of speed, precision and flexibility, in an attempt to meet the unprecedented challenges posed by the next generation high-frequency/high-speed design. The combination of artificial intelligence concepts with the state-of-the-art CAD technologies fosters numerous opportunities for technical innovations and industrial applications at various stages of the high-frequency/high-speed CAD, including modeling, simulation and design.

The proposed SSDNN technique has created the foundation for general transient neural modeling of high-speed nonlinear circuits, such as IC drivers and receivers. The next step is to further explore the potentials of this modeling technique, addressing the design challenges for emerging RF/microwave technologies. One important direction for future research is to develop efficient CAD methodologies for high-power nonlinear active devices in the millimetre-wave range. With the recent advances in modern semiconductor device technologies, Gallium Nitride (GaN) devices have become one of the most promising candidates for high-power and high-frequency applications, owing to their high breakdown voltages and high sheet carrier densities [152]. Efficient nonlinear

time-domain models of GaN devices are of great importance for the simulation and design of power amplifiers and oscillators [153]. However, accurate and fast time-domain modeling of GaN devices can be challenging, considering the highly nonlinear behaviors and strong memory/thermal effects. Recently, SSDNN techniques [10][11] have emerged as accurate and robust vehicles for modeling nonlinear transient behavior of high-speed IC devices and circuits. Advances in space mapping (SM) techniques, on the other hand, have resulted in major breakthroughs in high-frequency CAD [154][155], achieving efficient EM-based optimization by exploiting detailed models (called “fine models”) as well as approximate and fast models (called “coarse models”). For efficient analysis and optimization of high-frequency/high-power devices, one could create a comprehensive nonlinear CAD framework, combining the dynamic learning capability of ANN algorithms with elevated modeling efficiency through SM environments. Modern commercial semiconductor physics simulators (*e.g.*, Synopsys-Medici, Silvaco-ATLAS) could be used as fine models in the proposed research. Quick reduced-order nonlinear models, generated by the state-of-the-art SSDNN techniques, could be implemented as coarse models. A generalized training algorithm could be developed to adjust the weight parameters of the SSDNN as well as the parameters in the mapping networks, such that the overall model can accurately represent the nonlinear dynamic and thermal effects of the GaN devices. This entirely new CAD framework, which is built on the recent significant advances in SM optimization approaches and ANN techniques, is expected to enable rapid CAD of nonlinear devices for high-frequency/high-power applications, while preserving high accuracies offered by detailed physics-level models.

Another direction to be investigated is to expand the concept of SSDNN for modeling transient envelopes of modulated RF signals and to achieve efficient system-level envelope simulation based on the SSDNN behavioral models. The circuit envelope simulation plays a vital role in the CAD of today's wireless telecommunication circuits driven by complex modulated RF signals [156]. For the success of system-level envelope simulations, fast and accurate envelope-domain models are of great importance. The SSDNN could be a suitable candidate for circuit envelope modeling due to its learning capabilities of nonlinear dynamic behavior. Instead of directly using the complex modulated RF signals as training data, the envelope-domain SSDNN models could be developed from base-band in-phase and quadrature signals without involving the high frequency RF carriers. Adjoint sensitivities for SSDNN could be employed to enable efficient training of envelope-domain SSDNN models using the transient in-phase and quadrature waveforms. Unlike the existing envelope simulation methods where mixed frequency and time domain analyses are required, a new simulation technique could be developed for the efficient simulation of circuit envelopes based on the trained SSDNN models. It is expected that the envelope-domain SSDNN model could accurately reproduce the spectrum regrowths, and AM/AM and AM/PM conversions of the original power amplifier with enhanced simulation speed.

Combining the learning and generalization capabilities of ANNs with the existing RF and microwave engineering knowledge continues to be an important subject in high-frequency modeling and design. The objective is to extensively utilize the available prior circuit knowledge to develop knowledge-based ANN model architectures with highest

possible levels of model accuracies and efficiency. This is especially beneficial when limited training data are available or the generation of training data involves highly expensive computations. An interesting topic could be to incorporate dynamic knowledge of the original nonlinear circuits into the existing SSDNN modeling framework. As described in Chapter 3, the proposed SSDNN technique is a black-box modeling approach in the sense that the model can be developed directly from the input and output waveforms of the nonlinear circuits. In practice, certain form of circuit knowledge, such as empirical functions or equivalent circuits, could exist. For example, for high-speed I/O buffers, the industrially prevailing IBIS models are often available, which provide a quick and approximate estimation of the buffer behavior for system-level analysis. In such a case, since the available knowledge provides reasonable approximations of the device/circuit under test, it should help achieve higher accuracies with less training data or time, as well as enhanced model extrapolation capabilities beyond the model training range. One of the ongoing research subjects is to develop a new knowledge-based formulation that combines existing circuit/device knowledge in the form of IBIS models with the proposed SSDNN formulation. It is expected that the new hybrid model will maintain the existing benefits from IBIS models, such as fast simulation speed and improved model convergence, and further provide higher model accuracies and flexibilities when the model is terminated with a variety of printed circuit board structures.

The proposed robust training approach has enabled the development of ANN-based transient models of nonlinear circuits that maintain good accuracy for different waveform

shapes and load variations. Presently, the proposed training approach is applied to a special type of time-domain neural networks, called RNNs. Since the SSDNN technique has been proven to be an accurate and efficient vehicle for modeling nonlinear transient behavior, an interesting direction would be to further expand the proposed training approach to cover the SSDNN formulation. In this way, an advanced nonlinear transient modeling technique based on SSDNNs will be established, providing fast and accurate transient responses even if the circuit waveforms shapes and/or circuit terminations are not known during training. This will bring the ANN-based nonlinear modeling techniques, like those proposed in this thesis, one step closer to becoming a practical and industrial approach in high-speed system analysis and design.

Last but not least, another significant direction that is of great practical importance is to make the neural network modeling algorithms and techniques available to RF/microwave communities in the form of user-friendly software tools. These tools could enable quick technology transfer of the advanced ANN-based CAD methodologies to frontline circuit/system designers. The feedback from the designer can further stimulate in-depth scientific research in the direction of neural network modeling. In practice, it is often possible that the commercially available high-frequency/high-speed CAD tools do not always have the infrastructures for linking the externally developed neural models with internal component/circuit models. A key step towards the use of neural models in industrial applications is to develop plug-in software tools that provide convenient and seamless integration of neural models into the commercial simulation and design

environments. These activities have tremendous potentials for advancing the ANN-based modeling techniques into a mature mainstream CAD technology.

In conclusion, ANNs with their distinct capabilities of learning and abstraction, coupled with the attractive features such as speed, accuracy and flexibility, will continue to be one of the most promising and powerful modeling vehicles, tackling the growing challenges in the CAD of today's high-frequency/high-speed circuits and systems.

Appendix A: Derivation of Adjoint Sensitivity for SSDNN

The adjoint sensitivity for the existing DNN was introduced in [8][9]. Here we derive the sensitivity for the new state-space formulation of DNN. To derive the new SSDNN sensitivities, we utilize the concept of Lagrange function [88] and formulate our task as a nonlinear optimization problem of (3.25) with the SSDNN equations of (3.5) as dynamic constraints. We define a Lagrange function L as

$$L = \frac{1}{2} \left\| \mathbf{y}^m(t) - \mathbf{y}_d^m(t) \right\|^2 + \hat{\mathbf{x}}(t)^T \left[\dot{\mathbf{x}}(t) + \mathbf{x}(t) - \tau \mathbf{W}_o \boldsymbol{\sigma}(\mathbf{W}_u \mathbf{u}(t) + \mathbf{W}_s \mathbf{x}(t)) \right] \quad (\text{A.1})$$

where $\hat{\mathbf{x}} \in \mathbb{R}^N$ is a vector representing time-dependent Lagrange parameters. Let E_d^m be the per-waveform training error defined in (3.25). Subject to (3.5), derivatives of E_d^m with respect to the f_{ANN} weight w_{ij}^l can be expressed in terms of L as

$$\begin{aligned} \frac{dE_d^m}{dw_{ij}^l} = & \int_0^T (\mathbf{y}^m - \mathbf{y}_d^m)^T \mathbf{C} \frac{d\mathbf{x}}{dw_{ij}^l} dt \\ & + \int_0^T \hat{\mathbf{x}}^T \frac{d\dot{\mathbf{x}}}{dw_{ij}^l} dt + \int_0^T \hat{\mathbf{x}}^T \frac{d\mathbf{x}}{dw_{ij}^l} dt \\ & - \int_0^T \hat{\mathbf{x}}^T \left[\tau \frac{d\mathbf{W}_o}{dw_{ij}^l} \boldsymbol{\sigma} + \tau \mathbf{W}_o \mathbf{G} \left(\frac{d\mathbf{W}_u}{dw_{ij}^l} \mathbf{u} + \frac{d\mathbf{W}_s}{dw_{ij}^l} \mathbf{x} + \mathbf{W}_s \frac{d\mathbf{x}}{dw_{ij}^l} \right) \right] dt \end{aligned} \quad (\text{A.2})$$

where $\mathbf{G}$ is defined in (3.27). Integrating the term $\int_0^T \hat{\mathbf{x}}^T \frac{d\mathbf{x}}{dw_{ij}^l} dt$ of (A.2) by parts,

we obtain

$$\begin{aligned} \frac{dE_d^m}{dw_{ij}^l} &= \int_0^T (\mathbf{y}^m - \mathbf{y}_d^m)^T \mathbf{C} \frac{d\mathbf{x}}{dw_{ij}^l} dt + \hat{\mathbf{x}}^T \frac{d\mathbf{x}}{dw_{ij}^l} \Big|_0^T \\ &\quad - \int_0^T \dot{\hat{\mathbf{x}}}^T \frac{d\mathbf{x}}{dw_{ij}^l} dt + \int_0^T \hat{\mathbf{x}}^T \frac{d\mathbf{x}}{dw_{ij}^l} dt - \int_0^T \tau \hat{\mathbf{x}}^T \mathbf{W}_o \mathbf{G} \mathbf{W}_s \frac{d\mathbf{x}}{dw_{ij}^l} dt . \\ &\quad - \int_0^T \hat{\mathbf{x}}^T \left[\tau \frac{d\mathbf{W}_o}{dw_{ij}^l} \boldsymbol{\sigma} + \tau \mathbf{W}_o \mathbf{G} \left(\frac{d\mathbf{W}_u}{dw_{ij}^l} \mathbf{u} + \frac{d\mathbf{W}_s}{dw_{ij}^l} \mathbf{x} \right) \right] dt \end{aligned} \quad (\text{A.3})$$

Equation (A.3) can be rearranged as

$$\begin{aligned} \frac{dE_d^m}{dw_{ij}^l} &= \int_0^T \left[-\dot{\hat{\mathbf{x}}}^T + \hat{\mathbf{x}}^T - \tau \hat{\mathbf{x}}^T \mathbf{W}_o \mathbf{G} \mathbf{W}_s + (\mathbf{y}^m - \mathbf{y}_d^m)^T \mathbf{C} \right] \frac{d\mathbf{x}}{dw_{ij}^l} dt \\ &\quad + \hat{\mathbf{x}}^T \frac{d\mathbf{x}}{dw_{ij}^l} \Big|_0^T - \int_0^T \hat{\mathbf{x}}^T \left[\tau \frac{d\mathbf{W}_o}{dw_{ij}^l} \boldsymbol{\sigma} + \tau \mathbf{W}_o \mathbf{G} \left(\frac{d\mathbf{W}_u}{dw_{ij}^l} \mathbf{u} + \frac{d\mathbf{W}_s}{dw_{ij}^l} \mathbf{x} \right) \right] dt . \end{aligned} \quad (\text{A.4})$$

As (A.4) holds true for arbitrary choice of $\hat{\mathbf{x}}^T$, to eliminate the terms containing

$\frac{d\mathbf{x}}{dw_{ij}^l}$, we choose $\hat{\mathbf{x}}^T$ to satisfy the differential equation in the form

$$-\dot{\hat{\mathbf{x}}}(t)^T + \hat{\mathbf{x}}(t)^T - \tau \hat{\mathbf{x}}(t)^T \mathbf{W}_o \mathbf{G}(t) \mathbf{W}_s + (\mathbf{y}^m(t) - \mathbf{y}_d^m(t))^T \mathbf{C} = \mathbf{0} . \quad (\text{A.5})$$

This becomes the adjoint SSDNN equation in (3.26). Using the adjoint SSDNN of

(3.26) together with its boundary conditions, (A.4) can be simplified as

$$\begin{aligned} \frac{dE_d^m}{dw_{ij}^l} &= -\hat{\mathbf{x}}^T(0) \frac{d\mathbf{x}(0)}{dw_{ij}^l} \\ &\quad - \int_0^T \hat{\mathbf{x}}^T \left[\tau \frac{d\mathbf{W}_o}{dw_{ij}^l} \boldsymbol{\sigma} + \tau \mathbf{W}_o \mathbf{G} \left(\frac{d\mathbf{W}_u}{dw_{ij}^l} \mathbf{u} + \frac{d\mathbf{W}_s}{dw_{ij}^l} \mathbf{x} \right) \right] dt . \end{aligned} \quad (\text{A.6})$$

In general, $\mathbf{x}(0)$ is independent of the f_{ANN} weight w_{ij}^l , *i.e.*, $\frac{d\mathbf{x}(0)}{dw_{ij}^l} = \mathbf{0}$. Equation (A.6) can then be simplified by eliminating the first term. Thus the derivation of (3.28) is complete.

Bibliography

- [1] Q. J. Zhang and K. C. Gupta, *Neural Networks for RF and Microwave Design*. Norwood, MA: Artech House, 2000.
- [2] J. E. Rayas-Sánchez, “EM-based optimization of microwave circuits using artificial neural networks: the state-of-the-art,” *IEEE Trans. Microw. Theory Tech.*, vol. 52, no. 1, pp. 420–435, Jan. 2004.
- [3] Q. J. Zhang and Y. Cao, “Time-domain neural network approaches to EM modeling of microwave components,” Springer Proceedings in Physics: *Time Domain Methods in Electrodynamics*, pp. 41–53, Sep. 2008.
- [4] T. Liu, S. Boumaiza, and F. M. Ghannouchi, “Dynamic behavioral modeling of 3G power amplifier using real-valued time-delay neural networks,” *IEEE Trans. Microw. Theory Tech.*, vol. 52, no. 3, pp. 1025–1033, Mar. 2004.
- [5] Y. H. Fang, M. C. E. Yagoub, F. Wang, and Q. J. Zhang, “A new macromodeling approach for nonlinear microwave circuits based on recurrent neural network,” *IEEE Trans. Microwave Theory Tech.*, vol. 48, no. 12, pp. 2335–2344, Dec. 2000.
- [6] J. J. Xu, M. C. E. Yagoub, R. T. Ding, and Q. J. Zhang, “Neural based dynamic modeling of nonlinear microwave circuits,” *IEEE Trans. Microwave Theory Tech.*, vol. 50, no. 12, pp. 2769–2780, Dec. 2002.

- [7] Y. Cao, L. Zhang, J.J. Xu and Q.J. Zhang, "Efficient harmonic balance simulation of nonlinear microwave Circuits with dynamic neural models," *IEEE MTT-S Int. Microwave Symp.*, San Francisco, CA, USA, pp. 1423 –1426, Jun. 2006.
- [8] Y.Cao, J.J. Xu, V.K. Devabhaktuni, R.T. Ding and Q.J. Zhang, "An adjoint sensitivity technique for dynamic neural-network modeling and design of high-speed interconnect," *International Journal RF and Microwave CAE*, vol. 16, pp. 385-399, Jul. 2006.
- [9] Y. Cao, J.J. Xu, V.K. Devabhaktuni, R.T. Ding and Q.J. Zhang, "An adjoint dynamic neural network technique for exact sensitivities in nonlinear transient modeling and high-speed interconnect design," *IEEE MTT-S Int. Microwave Symp. Dig.*, Philadelphia, PA, USA, pp. 165 –168, Jun. 2003.
- [10] Y. Cao, R.T. Ding and Q.J. Zhang, "A new nonlinear transient modeling technique for high-speed integrated circuit applications based on state-space dynamic neural network," *IEEE MTT-S Int. Microwave Symp.*, Fort Worth, TX, USA, pp. 1553 –1556, Jun. 2004.
- [11] Y. Cao, R.T. Ding and Q.J. Zhang, "State-space dynamic neural network technique for high-speed IC applications: modeling and stability analysis," *IEEE Trans. Microwave Theory Tech.*, vol. 54, pp. 2398-2409, Jun. 2006.
- [12] Y. Cao and Q.J. Zhang, "State-space dynamic neural Network technique for high-speed IC buffer modeling," *International Symposium on Signals, Systems and Electronics, (ISSSE) 2007*, Montreal, QC, Canada, pp. 71 –74, Aug. 2007.

- [13] Y. Cao, Q.J. Zhang and I. Erdin, "Neural-based transient behavioral modeling of IC buffers for high-speed interconnect design," *Progress in Electromagnetics Research Symposium (PIERS) 2007*, Beijing, China, pp. 1316 –1318, Mar. 2007.
- [14] Y. Cao and Q.J. Zhang, "A new training approach for robust recurrent-neural-network modeling of nonlinear circuits," accepted in *IEEE Trans. Microwave Theory Tech.*, Jun. 2009.
- [15] MATLAB 7.0, The Mathworks Inc., Natick, MA, USA.
- [16] Q. J. Zhang, NeuroModelerPlus. Dept. Electron., Carleton Univ., Ottawa, ON, Canada
- [17] Advanced Design System ver. 2006A, Agilent Technologies, Inc., Palo Alto, CA, 2006.
- [18] HSPICE ver. 2007.03, Synopsys, Inc., Mountain View, CA, 2007.
- [19] P. Burrascano, S. Fiori, and M. Mongiardo, "A review of artificial neural networks applications in microwave computer-aided design", *Int. J. RF and Microwave CAE*, vol. 9, pp. 158-174, May. 1999.
- [20] F. Wang and Q.J. Zhang, "Knowledge-based neural models for microwave design," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 2333-2343, Dec. 1997.
- [21] 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, Dec. 1998.

- [22] 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, Nov. 1997.
- [23] G. Antonini and A. Orlandi, "Gradient evaluation for neural-networks-based electromagnetic optimization procedures," *IEEE Trans. Microwave Theory Tech.*, vol. 48, pp. 874-876, May 2000.
- [24] V.K. Devabhaktuni and Q.J. Zhang, "Neural network training-driven adaptive sampling algorithm for microwave modeling," *Proc. 30th European Microwave Conf.*, Paris, France, Oct. 2000, pp. 222-225.
- [25] R.L. Mahajan, "Design and optimization through physical/neural network models," *IEEE MTT-S Int. Microwave Symp. Workshop on Appl. of ANN to Microwave Design*, Denver, CO, Jun. 1997, pp. 1-16.
- [26] Q.J. Zhang, "A neural network paradigm for microwave modeling, simulation and optimization," *IEEE MTT-S Int. Microwave Symp. Workshop on Appl. of ANN to Microwave Design*, Denver, CO, Jun. 1997, pp. 87-107.
- [27] 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, Jun. 1999, pp. 269-272.
- [28] 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, Oct. 1997.

- [29] 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, May 1997.
- [30] 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 APS Int. Symp. Dig.*, Orlando, FL, Jul. 1999, pp. 2572-2575.
- [31] J.W. Bandler, M.A. Ismail, J.E. Rayas-Sanchez, and Q.J. Zhang, "Neuromodeling of microwave circuits exploiting space mapping technology," *IEEE MTT-S Int. Microwave Symp. Dig.*, Anaheim, CA, Jun. 1999, pp. 149-152.
- [32] 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, Jun. 1998, pp. 9-12.
- [33] P.M. Watson, K.C. Gupta, and R.L. Mahajan, "Applications of knowledge-based artificial neural network modeling to microwave components," *Int. J. RF and Microwave CAE*, vol. 9, pp. 254-260, May 1999.
- [34] P. Watson, G. Creech, and K. Gupta, "Knowledge based EM-ANN models for the design of wide bandwidth CPW patch/slot antennas," *IEEE APS Int. Symp. Dig.*, Orlando, FL, Jul. 1999, pp. 2588-2591.

- [35] J.W. Bandler, M.A. Ismail, J.E. Rayas-Sánchez and Q. J. Zhang, "Neuromodeling of microwave circuits exploiting space mapping technology," *IEEE Trans. Microwave Theory Tech.*, vol. 47, , pp. 2417-2427, Dec. 1999.
- [36] D. Schreurs, J. Verspecht, S. Vandenberghe, and E. Vandamme, "Straightforward and accurate nonlinear device model parameter-estimation method based on vectorial large-signal measurements," *IEEE Trans. Microwave Theory Tech.*, vol. 50, pp. 2315-2319, Oct. 2002.
- [37] Y. Harkouss, J. Rousset, H. Chehade, E. Ngoya, D. Barataud, and J.P. Teyssier, "Modeling microwave devices and circuits for telecommunications system design," *Proc. IEEE Int. Conf. Neural Networks*, Anchorage, Alaska, May 1998, pp. 128-133.
- [38] 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, May 1994, pp. 393-396.
- [39] 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, Jun. 1999, pp. 145-148.
- [40] G. Kothapalli, "Artificial neural networks as aids in circuit design," *Microelectronics J.*, vol. 26, pp. 569-578, Sep. 1995.
- [41] V.K. Devabhaktuni, C. Xi, and Q.J. Zhang, "A neural network approach to the modeling of heterojunction bipolar transistors from S-parameter data,"

- Proc. 28th European Microwave Conf.*, Amsterdam, Netherlands, Oct. 1998, pp. 306-311.
- [42] M. Vai and S. Prasad, "Qualitative modeling heterojunction bipolar transistors for optimization: A neural network approach," *Proc. IEEE/Cornell Conf. Adv. Concepts in High Speed Semiconductor Dev. and Circuits.*, Aug. 1993, pp. 219-227.
 - [43] K. Shirakawa, M. Shimizu, 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, Sep. 1997.
 - [44] 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, Oct. 1998.
 - [45] G. Fedi, S. Manetti, G. Pelosi, and S. Selleri, "Design of cylindrical posts in rectangular waveguide by neural network approach," *IEEE APS Int. Symp. Dig.*, Salt Lake City, UT, Jul. 2000, pp. 1054-1057.
 - [46] V. Rizzoli, A. Neri, D. Masotti, and A. Lipparin, "A new family of neural network-based bi-directional and dispersive behavioral models for nonlinear RF/Microwave subsystems," *Int. J. RF and Microwave CAE*, vol. 12, pp. 51-70, Jan. 2002.
 - [47] M. Isaksson, D. Wisell, and D. Rönnow, "Wide-band dynamic modeling of power amplifiers using radial-basis function neural networks," *IEEE Trans. Microw. Theory Tech.*, vol. 53, no. 11, pp. 3422-3428, Nov. 2005.

- [48] J. Wood, M. LeFevre, D. Runton, J.-C. Nanan, B. H. Noori, and P. H. Aaen, "Envelope-domain time series (ET) behavioral model of a Doherty RF power amplifier for system design," *IEEE Trans. Microw. Theory Tech.*, vol. 54, no. 8, pp. 3163–3172, Aug. 2006.
- [49] B. O'Brien, J. Dooley, and T. J. Brazil, "RF power amplifier behavioral modeling using a globally recurrent neural network," in *2006 IEEE MTT-S Int. Microwave Symp. Dig.*, San Francisco, CA, Jun. 2006, pp. 1089–1092.
- [50] J. Wood, D. E. Root, and N. B. Tufillaro, "A behavioral modeling approach to nonlinear model-order reduction for RF/microwave ICs and systems," *IEEE Trans. Microw. Theory Tech.*, vol. 52, no. 9, pp. 2274–2284, Sep. 2004.
- [51] H. Sharma and Q. J. Zhang, "Automated time domain modeling of linear and nonlinear microwave circuits using recurrent neural networks," *Int. J. RF Microw. Comput.-Aided Eng.*, vol. 18, no. 3, pp. 195–208, May 2008.
- [52] 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 APS Int. Symp. Dig.*, Orlando, FL, Jul. 1999, pp. 2604-2607.
- [53] C. Christodoulou, A. El Zooghby, and M. Georgiopoulos, "Neural network processing for adaptive array antennas," *IEEE APS Int. Symp. Dig.*, Orlando, FL, Jul. 1999, pp. 2584-2587.
- [54] R. Mishra and A. Patnaik, "Neurospectral analysis of coaxial fed rectangular patch antenna," *IEEE APS Int. Symp. Dig.*, Salt Lake City, UT, Jul. 2000, pp. 1062-1065.

- [55] 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, Dec. 1998.
- [56] A.H. El 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, Dec. 1998.
- [57] 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, Jul. 1999.
- [58] H. Yang, C. He, W. Song, and H. Zhu, "Using artificial neural network approach to predict rain attenuation on earth-space path," *IEEE APS Int. Symp. Dig.*, Salt Lake City, UT, Jul. 2000, pp. 1058-1061.
- [59] J. Lee and A. Lai, "Function-based and physics-based hybrid modular neural network for radio wave propagation modeling," *IEEE APS Int. Symp. Dig.*, Salt Lake City, UT, Jul. 2000, pp. 446-449.
- [60] 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, May. 1995.
- [61] M. Vai and S. Prasad, "Applications of neural networks optimized by the genetic algorithm to microwave systems," *IEEE APS Int. Symp. Dig.*, Orlando, FL, Jul. 1999, pp. 2580-2583.

- [62] E. Bermani, S. Caorsi, and M. Raffetto, "Time domain inverse scattering by means of neural networks," *IEEE APS Int. Symp. Dig.*, Salt Lake City, UT, Jul. 2000, pp. 1760-1763.
- [63] 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, Oct. 1998.
- [64] M. Vai and S. Prasad, "Neural networks in microwave circuit design - Beyond black box models," *Int. J. RF and Microwave CAE*, vol. 9, pp. 187-197, May 1999.
- [65] H. Kabir, Y. Wang, M. Yu and Q.J. Zhang, "Neural network inverse modeling and applications to microwave filter design," *IEEE Trans. Microwave Theory Tech.*, vol. 56, No. 4, , pp. 867-879, Apr. 2008.
- [66] 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, Jun. 2000, pp. 879-882.
- [67] 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 and Microwave CAE*, vol. 9, pp. 175-186, May 1999.
- [68] J. E. Rayas-Sánchez and V. Gutiérrez-Ayala, "EM-based Monte Carlo analysis and yield prediction of microwave circuits using linear-input neural-output space

- mapping,” *IEEE Trans. Microwave Theory Tech.*, vol. 54, pp. 4528-4537, Dec. 2006.
- [69] J. W. Bandler, J. E. Rayas-Sánchez and Q. J. Zhang, “Yield-driven electromagnetic optimization via space mapping-based neuromodels,” *Int. J. RF and Microwave CAE*, vol. 12, pp. 79-89, Jan. 2002.
 - [70] S. Haykin, *Neural Networks: A Comprehensive Foundation*. 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999, ch. 15, pp. 751–756.
 - [71] 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, Dec. 2001.
 - [72] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume I, pp. 318-362, MIT Press, Cambridge, Massachusetts, 1986.
 - [73] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, pp. 671-680, 1983.
 - [74] A. J. F. van Rooij, L. C. Jain, R. P. Johnson, *Neural Network Training Using Genetic Algorithms*, World Scientific, 1996.
 - [75] C. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Math. Control Signals Systems*, vol. 2, pp. 303-314, 1989.
 - [76] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, pp. 359-366, May 1989.

- [77] 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, May 1997.
- [78] R. Reed, "Pruning algorithms – a survey," *IEEE Trans. Neural Networks*, vol. 4, pp. 740-747, Sep. 1993.
- [79] J. de Villiers and E. Barnard, "Backpropagation neural nets with one and two hidden layers," *IEEE Trans. Neural Networks*, vol. 4, pp. 136-141, Jan. 1992.
- [80] S. Tamura and M. Tateishi, "Capabilities of a four-layered feedforward neural network: four layer versus three," *IEEE Trans. Neural Networks*, vol. 8, pp. 251-255, Mar. 1997.
- [81] J.A. Garcia, A. Tazon, A. Mediavilla, I. Santamaria, M. Lazaro, C.J. Pantaleon, and J.C. Pedro, "Modeling MESFET's and HEMT's intermodulation distortion behavior using a generalized radial basis function network," *Int. Journal of RF and Microwave CAE*, vol. 9, pp. 261-276, Apr. 1999.
- [82] I. S. Stievano, I. A. Maio, and F. G. Canavero, "Parametric macromodels of digital I/O ports," *IEEE Trans. Adv. Packag.*, vol. 25, no. 2, 255-264, May 2002.
- [83] I. S. Stievano, I. A. Maio, and F. G. Canavero, "Mpilog macromodeling via parametric identification of logic gates," *IEEE Trans. Adv. Packag.*, vol. 27, no. 1, 15-23, Feb. 2004.

- [84] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4-27, Mar. 1990.
- [85] J. Aweya, Q. J. Zhang, and D. Y. Montuno, "A direct adaptive neural controller for flow control in computer networks," in *Proc. IEEE Int. Neural Networks Conf.*, Anchorage, Alaska, pp. 140-145, May 1998.
- [86] J. Hopfield and D. Tank, "Neural computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, pp. 141-152, Mar. 1985.
- [87] J.A. Freeman and D.M. Skapura, *Neural Networks: Algorithms, Applications and Programming Techniques*, Reading, Mass: Addison-Wesley, 1992.
- [88] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*. New York, NY: Van Nostrand Reinhold, 1993.
- [89] G. Casinovi and A. Sangiovanni-Vincentelli, "A macromodeling algorithm for analog circuits," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 150-160, Feb. 1991.
- [90] K.S. Kundert, G.B. Sorkin, and A. Sangiovanni-vincentelli, "Applying harmonic balance to almost-periodic circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 36, pp. 366-378, Feb. 1988.
- [91] R. Battiti, "First and second-order methods for learning: between steepest descent and Newton's method," *Neural Computation*, vol. 4, pp. 141-166, Feb. 1992.

- [92] 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.
- [93] *Neural Network Toolbox Version 5.0, For Use with Matlab*, The MathWorks Inc., Natick, Massachusetts, 2006.
- [94] R. A. Jacobs, "Increased rate of convergence through learning rate adaptation," *Neural Networks*, vol. 1, pp. 295-307, 1988.
- [95] M. Arisawa and J. Watada, "Enhanced back-propagation learning and its application to business evaluation," *In Proc. IEEE Intl. Conf. Neural Networks*, vol. I, pp. 155-160, Orlando, Florida, Jul. 1994.
- [96] 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.
- [97] P. P. Van-Der-Smagt, "Minimization methods for training feedforward neural networks," *Neural Networks*, vol. 7, pp. 1-11, 1994.
- [98] R. L. Watrous, "Learning algorithms for connectionist networks: applied gradient methods of nonlinear optimization," *In Proc. IEEE First Intl. Conf. Neural Networks*, vol. II, pp. 619-627, San Diego, California, 1987.
- [99] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Cambridge, Massachusetts: Cambridge University Press, 1986.

- [100] T. R. Cuthbert, *Optimization using Personal Computers, chapter Quasi-Newton Methods and Constraints*, pp. 233-314. John Wiley and Sons, New York, NY, 1987.
- [101] P. Gill, W. Murray and M. Wright, *Practical Optimization*, Academic Press, London, England, 1981.
- [102] K. R. Nakano, "Partial BFGS update and efficient step-length calculation for three layer neural network," *Neural Computation*, vol. 9, pp. 123-141, 1997.
- [103] S. Mcloone and G. W. Irwin, "Fast parallel off-line training of multilayer perceptrons," *IEEE Trans. Neural Networks*, vol. 8, pp. 646-653, May. 1997.
- [104] T. Rognvaldsson, "On Langevin updating in multilayer perceptron," *Neural Computation*, vol. 6, pp. 916-926, 1994.
- [105] R. Brunelli, "Training neural nets through stochastic minimization," *Neural Networks*, vol. 7, pp. 1405-1412, 1994.
- [106] R. Achar and M. S. Nakhla, "Simulation of high-speed interconnects," *Proc. IEEE*, vol. 89, no. 5, pp. 693-728, May 2001.
- [107] *I/O Buffer Information Specification (IBIS) Ver. 4.1*: Electronic Industries Alliance, Jan. 2003. Available: <http://eda.org/pub/ibis/ver4.1/>
- [108] I. S. Stievano, I. A. Maio, and F. G. Canavero, "Parametric macromodels of differential drivers and receivers," *IEEE Trans. Adv. Packag.*, vol. 28, no. 2, 189-196, May 2005.
- [109] B. Mutnury, M. Swaminathan, and J. P. Libous, "Macromodeling of non-linear digital I/O drivers," *IEEE Trans. Adv. Packag.*, vol. 29, no. 1, 102-113, Feb. 2006.

- [110] P. K. Gunupudi and M. S. Nakhla, "Model-reduction of nonlinear circuits using Krylov-space techniques," in *Proc. IEEE Int. Design Automation Conf.*, New Orleans, Louisiana, pp. 13-16, Jun. 1999.
- [111] P. Gunupudi and M. Nakhla, "Nonlinear circuit-reduction of high-speed interconnect networks using congruent transformation techniques," *IEEE Trans on Advanced Packaging*, vol. 24, no. 3, pp. 317-325, Aug. 2001.
- [112] P. Li and L.T. Pileggi, "Compact reduced-order modeling of weakly nonlinear analog and RF circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 2, pp. 184–203, Feb. 2005.
- [113] M. Rewienski and J. White, "A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 2, pp. 155–170, Feb. 2003.
- [114] N. Dong and J. Roychowdhury, "Automated extraction of broadly applicable nonlinear analog macromodels from SPICE-level descriptions," in *Proc. IEEE Custom Integr. Circuits Conf.*, Oct. 2004, pp. 117–120.
- [115] S. Tiwary and R. A. Rutenbar, "On-the-fly fidelity assessment for trajectory-based circuit macromodels," in *Proc. IEEE Custom Integr. Circuits Conf.*, Sep. 2006, pp. 185–188.
- [116] N. Dong and J. Roychowdhury, "General-purpose nonlinear model-order reduction using piecewise-polynomial representations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 2, pp. 249–264, Feb. 2008.

- [117] A. Odabasioglu, M. Celik, and L. Pileggi, "PRIMA: Passive reduced order interconnect macromodeling algorithm," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 1997, pp. 58–65.
- [118] R. W. Freund, "Krylov-subspace methods for reduced-order modeling in circuit simulation," *J. Comput. Appl. Math.*, vol. 123, no. 1/2, pp. 395–421, Nov. 2000.
- [119] D. Schreurs, J. Verspecht, E. Vandamme, N. Vellas, C. Gaquiere, M. Germain, and G. Borghs, "ANN model for AlGaIn/GaN HEMTs constructed from near-optimal-load large-signal measurements," in *IEEE MTT-S Int. Microwave Symp. Dig.*, Philadelphia, PA, Jun. 2003, pp. 447–450.
- [120] I. A. Maio, I. S. Stievano, and F. G. Canavero, "NARX approach to black-box modeling of circuit elements," in *Proc. IEEE Int. Symp. Circuits and Systems*, Monterey, CA, Jun. 1998, pp. 411–414.
- [121] S. Lum, M. Nakhla, and Q. J. Zhang, "Sensitivity analysis of lossy coupled transmission lines with nonlinear terminations," *IEEE Trans. Microwave Theory Tech.* vol. 42, no. 4, pp. 607–615, Apr. 1994.
- [122] I.S. Stievano, Z. Chen, D. Becker, F.G. Canavero, G. Katopis and I.A. Maio, "Behavioral modeling of digital IC input and output ports," in *Proc. Electrical Performance of Electronic Packaging*, Cambridge, MA, Oct. 2001, pp. 331–334.
- [123] B. Mutnury, M. Swaminathan, and J. Libous, "Macro-modeling of non-linear I/O drivers using spline functions and finite time-difference approximation," in *Proc. Electrical Performance of Electronic Packaging*, Princeton, NJ, Oct. 2003, pp. 273–276.

- [124] Z. H. Guan, G. R. Chen, and Y. Qin, "On equilibria, stability, and instability of Hopfield neural networks," *IEEE Trans. Neural Networks*, vol. 11, no. 3, pp. 534-540, Mar. 2000.
- [125] J. Cao, "Global exponential stability of Hopfield neural networks," *Int. J. Syst. Sci.*, vol.32, no. 2, pp. 233-236, Feb. 2001.
- [126] T. G. Chu, C. S. Zhang, and Z. D. Zhang, "Necessary and sufficient condition for absolute stability of normal neural networks," *Neural Networks*, vol. 16, no. 8, pp. 1223-1227, Oct. 2003.
- [127] S. Q. Hu and J. Wang, "Global stability of a class of continuous-time recurrent neural networks," *IEEE Trans. Circuits Syst.*, vol. 49, no. 9, pp. 1534-1547, Sep. 2002.
- [128] J. M. Zamarreño and P. Vega, "State space neural network: properties and application," *Neural Networks*, vol. 11, no. 6, pp. 1099-1112, Aug. 1998.
- [129] J. Cao and J. Wang, "Global asymptotic and robust stability of recurrent neural networks with time delays," *IEEE Trans. Circuits Syst.*, vol. 52, no. 2, pp. 417-426, Feb. 2005.
- [130] J. Cao and J. Wang, "Global asymptotic stability of a general class of recurrent neural networks with time-varying delays," *IEEE Trans. Circuits Syst.*, vol. 50, no. 1, pp. 34-44, Jan. 2003.
- [131] T. L. Liao and F. C. Wang, "Global stability of cellular neural networks with time delays," *IEEE Trans. Neural Networks*, vol. 11, no. 11, pp. 1481-1484, Nov. 2000.

- [132] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*. Philadelphia, PA: SIAM, 1994.
- [133] L. Leistriz, M. Galicki, H. Witte, and E. Kochs, "Training trajectories by continuous recurrent multilayer networks," *IEEE Trans. Neural Networks*, vol. 15, no. 3, pp. 283-291, Mar. 2002.
- [134] J. M. Zamarreño, P. Vega, L. D. García, and M. Francisco, "State-space neural network for modelling, prediction and control," *Control Engineering Practice*, vol. 8, no. 9, pp. 1063-1075, Sep. 2000.
- [135] I. Rivals and L. Personnaz, "Black-box modeling with state-space neural networks," in *Neural Adaptive Control Technology*, R. Zbikowski and K. J. Hunt Ed. New Jersey: World Scientific, 1996, pp. 237-264.
- [136] P. Gil, A. Dourado, and J. O. Henriques, "State space neural networks and the unscented Kalman filter in online nonlinear system identification," *IASTED International Conference on Intelligent Systems and Control*, Tampa, FL, Nov. 2001, pp. 337-342.
- [137] J. O. Henriques, A. Dourado, and P. Gil, "State space neural networks in nonlinear adaptive system identification and control," *IFAC Workshop on Advanced Fuzzy/Neural Control 2001*, Valencia, Spain, Oct. 2001, pp. 99-104.
- [138] J. L. Elman, "Finding structure in time", *Cogn. Sci.*, vol.14, pp.179-211, 1990.
- [139] P. Campolucci, A. Uncini, F. Piazza, and B. D. Rao, "On-line learning algorithms for locally recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 10, no. 2, pp. 253-271, Mar. 1999.

- [140] M.Vidyasagar, *Nonlinear Systems Analysis*, 2nd Edition. Englewood Cliffs, NJ: Prentice-Hall, 1993.
- [141] *Optimization Toolbox, For Use with Matlab*, The MathWorks Inc., Natick, Massachusetts, 2005.
- [142] A. S. Sedra and K. C. Smith, *Microelectronic Circuits*, 4th Edition. New York, NY: Oxford University Press, 1998.
- [143] J. L. Rossello and J. Segura, "Charge-based analytical model for evaluation of power consumption in submicron CMOS buffer," *IEEE Trans. Computer-Aided Design*, vol. 21, no. 4, pp. 433-448, Apr. 2002.
- [144] MINIMOS-NT v.2.0, Institute for Microelectronics, Technical University Vienna, Vienna, Austria.
- [145] *Inverting Buffers and Drivers Library*, Texas Instruments Inc., Dallas, TX, USA.
- [146] L. Zhang, J. J. Xu, M. C. E. Yagoub, R. T. Ding, and Q. J. Zhang, "Efficient analytical formulation and sensitivity analysis of neuro-space mapping technique for nonlinear microwave device modeling," *IEEE Trans. Microw. Theory Tech.*, vol. 53, no. 9, pp. 2752-2767, Sep. 2005.
- [147] D. Schreurs, J. Wood, N. Tuffillaro, L. Barford, and D. E. Root, "Construction of behavioral models for microwave devices from time domain large-signal measurements to speed up high-level design simulations", *Int. J. RF and Microwave CAE*, vol. 13, no. 1, pp. 54-61, Jan. 2003.
- [148] D. Schreurs, S. Vandenberghe, H. Taher, and B. Nauwelaers, "ANN model for SiGe HBTs constructed from time-domain large-signal measurements," in *Proc.*

10th European Gallium Arsenide and Other Semiconductors Application Symposium, Milano, Italy, Sep. 2002, pp. 85–88.

- [149] V. Rizzoli, A. Costanzo, D. Masotti, A. Lipparini, and F. Matri, “Computer-aided optimization of nonlinear microwave circuits with the aid of electromagnetic simulation,” *IEEE Trans. Microwave Theory Tech.*, vol. 52, no. 1, pp. 362–377, Jan. 2004.
- [150] D. Saraswat, R. Achar, and M. Nakhla, “A fast algorithm and practical considerations for passive macromodeling of measured/simulated data,” *IEEE Trans. Adv. Packag.*, vol. 27, no. 1, pp. 57–70, Feb. 2004.
- [151] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective*. Upper Saddle River, NJ: Prentice-Hall, 1996, ch. 8, pp. 490–493.
- [152] U. K. Mishra *et al.*, “GaN-based RF power devices and amplifier,” *Proc. IEEE*, vol. 96, no. 2, pp. 287–305, Feb. 2008.
- [153] P.M. Cabral, J.C. Pedro, and N.B. Carvalho, “Nonlinear device model of microwave power GaN HEMTs for high power-amplifier design,” *IEEE Trans. Microwave Theory Tech.*, vol. 52, no. 11, pp. 2585–2592, Nov. 2004.
- [154] J. Bandler *et al.*, “Space mapping: the state of the Art,” *IEEE Trans. Microwave Theory Tech.*, vol. 52, no. 1, pp. 337–361, Jan. 2004.
- [155] J.W. Bandler, S. Koziel, and K. Madsen, “Editorial—surrogate modeling and space mapping for engineering optimization,” *Optimization and Engineering, Special Issue on Surrogate Modeling and Space Mapping for Engineering Optimization*, vol. 9, no. 4, pp. 307–310, Apr. 2008.

- [156] R. Frevert, J. Haase, R. Jancke, U. Knöchel, P. Schwarz, R. Kakerow and M. Darianian, *Modeling and Simulation for RF System Design*, Dordrecht: Springer, 2005.