A NOVEL AUTOMATED MODEL GENERATION ALGORITHM FOR HIGH LEVEL FAULT MODELING OF ANALOG CIRCUITS

MUHAMMAD UMER FAROOQ

MASTER OF SCIENCE
ELECTRICAL AND ELECTRONIC
ENGINEERING DEPARTMENT

UNIVERSITI TEKNOLOGI PETRONAS

MARCH 2013

STATUS OF THESIS

Title of thesis	A Novel Automated Model Generation Algorithm for High Level Fault Modelling of Analog Circuits

I MUHAMMAD UMER FAROOQ

hereby allow my thesis to be placed at the Information Resource Center (IRC) of Universiti Teknologi PETRONAS (UTP) with the following conditions:

1. The thesis becomes the property of UTP
2. The IRC of UTP may make copies of the thesis for academic purposes only.
3. This thesis is classified as

☐ Confidential

☑ Non-confidential

If this thesis is confidential, please state the reason:

___

The contents of the thesis will remain confidential for ______ years.

Remarks on disclosure:

___

Signature of Author

Permanent address:
House No.1037, Street # 41,
Sector G-10/4, Islamabad,
Pakistan
Date: 07/03/2013

Endorsed by

Signature of Supervisor

Name of Supervisor
Dr. Likun Xia

Date : 07/03/13

DECLARATION OF THESIS

Title of thesis	A Novel Automated Model Generation Algorithm for High Level Fault Modelling of Analog Circuits

I MUHAMMAD UMER FAROOQ

hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTP or other institutions.

Witnessed by

Signature of Author

Signature of Supervisor

Permanent address:
House No.1037, Street # 41, Sector G-10/4, Islamabad, Pakistan
Date: 07-03-2013

Name of Supervisor
Dr. Likun Xia

Date : 07/03/2013

DEDICATION

To my parents and teachers

ACKNOWLEDGEMENTS

I thank Allah the Almighty, for giving me the strength and opportunity to achieve what has been done in this work. I would like to thanks my supervisor Dr. Likun Xia, Dr. Fawnizu Azmadi Hussin and Dr. Aamir Saeed Malik for their help and supervision in conduction this research. I greatly appreciate their care, dedication and constructive criticism. I have truly benefited and enjoyed working with them.

I thank to the management and authorities of Universiti Teknologi PETRONAS for the generous financial support and administrative facilities throughout my stay with Electrical and Electronic Department.

Thanks to those with whom I have worked and during the course of my study. To my friends in Center of Intellignet Signal and Imaging Research Lab, Wajid Mumtaz, Junaid, Roshahslinie Ramli, Rishu Gupta, Ateeq-ur-rehman shaheen, Marwan Shaheen, Duc Lee, Duc Tra, Ghazanfar. My special thanks to Rao Ahmed Rauf Subhani and Jawad Hamayun who has been my great support during my stay in Malaysia. I also express my sincere appreciation to all my friends and country mates in UTP for their support during the progress of this work.

Last but not the least a very special gratitude towards my parents, my wife and all the family members for their prayers and consistent moral support in every thick and thin. Special thanks to my In-laws for their great moral support and timely back-up.

ABSTRACT

High level modelling techniques have been used by researchers from few decades to increase fault simulation speed of analog circuits. However, due to manual model generation, the techniques are tedious and time consuming and unable to reduce analog testing time. To overcome manual modelling limitation, researchers adopt algorithmic support and start using automated model generation (AMG) methods to generate models for high level modelling of analog circuits. AMG models successfully perform HLFM but unfortunately fail to increase high level fault simulation (HLFS) speed compared to full SPICE-circuit simulations. The failure is mainly occurred due to the consumption of multiple models and computational overhead of model switching required capturing nonlinear effects.

In this dissertation we propose a novel AMG approach termed automated model generation using Chebyshev and Newton interpolating polynomials (AMG-CNIP). Unlike existing white-box AMG methods that use multiple Taylor polynomial models, AMG-CNIP generate single model by employing a fusion of Chebyshev and Newton interpolating polynomials in nonlinear state-space (ss) model structure. AMG-CNIP model successfully capture nonlinear effects generated by fault-free and faulty analog circuits.

High level models are generated by translating AMG-CNIP MATLAB model into VHDL-AMS models. High level models are evaluated at system level using benchmark nonlinear transmission line circuits and simulation speed is compared with full SPICE circuit simulation. Experimentation results shows that high level models shows good accuracy at system level and achieves simulation speedup compared to full SPICE circuit simulations.

ABSTRAK

Teknik-teknik pemodelan peringkat tinggi telah digunakan oleh penyelidik sejak beberapa dekad untuk meningkatkan kelajuan simulasi kesalahan litar analog. Walau bagaimanapun, disebabkan oleh generasi model manual, teknik-teknik tersebut adalah merumitkan dan memakan masa dan tidak mampu mengurangkan masa ujian analog. Untuk mengatasi had-had model secara manual, penyelidik mengamalkan algoritma sokongan dan mula menggunakan kaedah model generasi automatik (AMG) bagi menjana model-model untuk pemodelan litar analog peringkat tinggi. Model AMG berjaya melaksanakan HLFM namun gagal untuk meningkatkan kelajuan simulasi kesalahan peringkat tinggi (HLFS) berbanding simulasi litar penuh SPICE. Kegagalan utama yang berlaku adalah disebabkan oleh penggunaan pelbagai model dan melebihi pengiraan yang diperlukan oleh model pensuisan untuk merakam kesan tidak linear.

Dalam disertasi ini, kita mencadangkan pendekatan baru AMG yang dipanggil generasi model automatik menggunakan Chebyshev dan Newton interpolasi polinomial (AMG-CNIP). Tidak seperti kaedah AMG-kotak putih yang sedia ada yang menggunakan model polinomial berganda Taylor, AMG-CNIP menjana model tunggal dengan menggunakan gabungan Chebyshev dan Newton interpolasi polinomial dalam model struktur keadaan-ruang (ss) tidak linear. Model AMG-CNIP berjaya merakam kesan tidak linear yang dijana oleh litar yang bebas kerosakan dan litar analog yang mempunyai kerosakan.

Model tahap tinggi dihasilkan dengan menukarkan model AMG CNIP MATLAB kepada model VHDL-AMS. Sistem model tahap tinggi dinilai dengan menggunakan penanda aras talian penghantaran litar tidak linear dan kelajuan simulasi dibandingkan dengan simulasi litar SPICE penuh. Keputusan eksperimen menunjukkan bahawa model tahap tinggi menunjukkan ketepatan yang baik pada peringkat sistem dan mencapai kecepatan simulasi berbanding simulasi litar SPICE penuh.

In compliance with the terms of the Copyright Act 1987 and the IP Policy of the university, the copyright of this thesis has been reassigned by the author to the legal entity of the university,

Institute of Technology PETRONAS Sdn Bhd.

Due acknowledgement shall always be made of the use of any material contained in, or derived from, this thesis.

© Muhammad Umer Farooq, 2013
Institute of Technology PETRONAS Sdn Bhd
All rights reserved.

TABLE OF CONTENTS

A NOVEL AUTOMATED MODEL GENERATION ALGORITHM FOR HIGH LEVEL FAULT MODELING OF ANALOG CIRCUITS II

LIST OF TABLES xii

CHAPTER 1 INTRODUCTION 2

1.1 Objectives 4

1.2 Dissertation Organization 5

CHAPTER 2 LITERATURE REVIEW 6

2.1 Classification of AMG Approaches 6

2.1.1 Linear Time Invariant (LTI) System 6

2.1.2 Linear Time Varying (LTV) System 8

2.1.3 Nonlinear Systems 8

2.2 SI AMG Approaches 9

2.2.1. SI AMG for Nonlinear Systems 10

2.3 MOR AMG Approaches 17

2.3.1. MOR AMG for Nonlinear Systems 18

CHAPTER 3 METHODOLOGY 26

3.1. Introduction 26

3.2. Automated Model Generation Algorithm Using Chebyshev and Newton Interpolating Polynomial (AMG-CNIP) 31

3.2.1 Model generation 31

3.2.1.1. 1D CN Polynomial Formulation 32

3.2.1.2. 2D CN Polynomial Formulation 32

3.2.1.3. Automatic Range Finder (ARF) 33

3.2.1.4. CN Polynomial Representation in SS 34

3.2.1.5. Automatic Polynomial Tuner (APT) 34

3.3. Model Evaluation 36

3.4. AMG-CNIP Experimentation 38

3.5. Conclusion 44

CHAPTER 4 HIGH LEVEL MODELING AND HIGH LEVEL FAULT MODELING USING AMG-CNIP MACROMODELS 46

4.1. Introduction..46
4.2. Automatic Model Conversion (AMC) 46
4.3. AMG-CNIP Algorithm for HLM and HLFM............................48
4.4. HLM Using AMG-CNIP Macromodels...................................50
4.5. HLFM Using AMG-CNIP Macromodels..................................62
  4.5.1 HLFM for Short Fault at Diode D1 62
  4.5.2 HLFM for Short Fault at Diode D2...................................... 71
4.6. Conclusion 76
CHAPTER 5 CONCLUSION AND FUTURE RECOMMENDATIONS.................78
5.1. Future Work and Recommendations...79
REFERENCES 82
PUBLICATIONS 91
APPENDICES... 92
Appendix A analog circuits as state space..93
Appendix B matlab code for amg-cnip algorithm...............................97
B.1. Main Routine...98
B.2. AMG-CNIP Model Generation Routine...................................99
  B.2.1 Simulate Circuit Routine... 99
    B.2.1.1 Nltxline Routine... *100*
  B.2.2 ARF Routine.. 101
  B.2.3 1D CN Polynomial Routine.. 102
    B.2.3.1 1D NDD Routine.. *102*
  B.2.4 2D CN Polynomial Routine.. 103
    B.2.4.1. 2D NDD Routine *103*
  B.2.5 APT Routine.. 104
B.3. AMG-CNIP Model Evaluation Routine 108
B.4. Automatic Model Conversion (AMC) Routine.......................109
B.5. Plotting Routines 111

LIST OF TABLES

Table 3-1: Algorithm for Automatic Polynomial Tuner (APT)..........................36

Table 3-2: AMG-CNIP algorithm for model generation and simulation................38

Table 4-1: AMG-CNIP algorithm for model generation, HLM and HLFM...............49

Table 4-2: Configurations for AMG-CNIP HLM Testing..53

Table 4-3: Summary of simulation speed up and accuracy for AMG-CNIP HLM.....61

Table 4-4: Configurations for AMG-CNIP HLFM testing (short fault at D1)............62

Table 4-5: Summary of simulation speed up and accuracy for AMG-CNIP HLFM (fault at diode D1)..70

Table 4-6: Configurations for AMG-CNIP HLFM testing (short fault at D2)............71

Table 4-7: Summary of simulation speed up and accuracy for AMG-CNIP HLFM (short fault at diode D2)..75

LIST OF FIGURES

Figure 2-1: Classification of AMG techniques 7

Figure 2-2: Linear Time Invariant (LTI) block [1] 8

Figure 2-3: Linear Time Varying (LTV) 8

Figure 2-4: NLARX model structure, see equation (2-1) [17] 10

Figure 2-5: Hammerstein-Wiener (H-W) model [17] 11

Figure 3-1: Multiple models using Taylor polynomials 27

Figure 3-2: Comparison of convergence properties of Taylor and Chebyshev polynomials 28

Figure 3-3: Generation of the linearized models along a trajectory of nonlinear system in two-dimension ss 29

Figure 3-4: AMG-CNIP model generation and evaluation flow 36

Figure 3-5: Schematic of transmission line circuit (N=5 nodes) with $i_d(x) = \exp(40x) - 1$ 38

Figure 3-6: Training input waveform 39

Figure 3-7: Transient response of 1D polynomial macromodels for input amplitude ±2.0A and random *freq* and θ 40

Figure 3-8: Transient response of 1D polynomial macromodels for input amplitude ±5.0A that is ±2.5A greater than training input amplitude 41

Figure 3-9: Transient response of 2D polynomial macromodels for input amplitude ±2.0A and random *freq* and θ 42

Figure 3-10: Transient response of 2D polynomial macromodels for input amplitude ±5.0A that is ±2.5A greater than training input amplitude 43

Figure 4-1: An example of VHDL-AMS behavioral model for nonlinear transmission line circuit 47

Figure 4-2: Complete picture of AMG-CNIP model generation and performing HLM and HLFM 48

Figure 4-3: Nonlinear transmission line circuit wit nonlinear current function *(fx)* from SPICE2G6 in SystemVision 50

Figure 4-4: 3-stage cascaded nonlinear transmission line circuit 50

Figure 4-5: MMGSD and AMG-CNIP response for fault-free NLTx line circuit 51

Figure 4-6: MMGSD and AMG-CNIP response for fault-free NLTx circuit (ignoring initial 150us response) 52

Figure 4-7: Transient response of MMGSD and AMG-CNIP for Config1 circuit 54

Figure 4-8: Transient response of AMG-CNIP HLM for Config2 56

Figure 4-9: Transient response of AMG-CNIP HLM for Config3 (node 5) 57

Figure 4-10: Transient response of AMG-CNIP HLM for Config3 (node 11) 57

Figure 4-11: Transient response of AMG-CNIP HLM for Config4 (nodes 11, 12 and 13) 58

Figure 4-12: Transient response of AMG-CNIP HLM for Config5 59

Figure 4-13: Transient response of AMG-CNIP HLM for Config6 (nodes 1and 5)...60

Figure 4-14: NLTx circuit with short fault at D1 62

Figure 4-15: MMGSD and AMG-CNIP response for NLTx line faulty circuit 63

Figure 4-16: Transient response of AMG-CNIP HLM and HLFM for Config1 (nodes 1, 2 and 3) simulation result 65

Figure 4-17: Transient response of AMG-CNIP HLFM for Config2 (nodes 5 and 6) 66

Figure 4-18: Transient response of AMG-CNIP HLFM for Config3 (nodes 10 and 11) 67

Figure 4-19: AMG-CNIP HLFM Config4 (nodes 11 and 12) simulation result 68

Figure 4-20: Transient response of AMG-CNIP HLFM for Config5 (nodes 5, 6 and 11) 69

Figure 4-21: Transient response of AMG-CNIP HLFM for Config6 (nodes 10 and 11) 70

Figure 4-22: NLTx circuit with short fault at D2 71

Figure 4-23: MMGSD and AMG-CNIP response for Config1 fault at D2 72

Figure 4-24: Transient response of AMG-CNIP HLM and HLFM for Config1 (node 2) with fault at D2 72

Figure 4-25: Transient response of AMG-CNIP HLFM for Config2 (nodes 2 and 6) with fault at D2 73

Figure 4-26: Transient response of AMG-CNIP HLFM for Config3 (nodes 7 and 11) with fault at D2 73

Figure 4-27: Transient response of AMG-CNIP HLFM for Config4 (node 12) with fault at D2 74

Figure 4-28: Transient response of AMG-CNIP HLFM for Config5 (nodes 2, 7 and 11) with fault at D2 74

Figure 4-29: Transient response of AMG-CNIP HLFM for Config6 (nodes 1, 6 and 11) with fault at D2 75

LIST OF ABBREVIATIONS

AMG	Automated Model Generation
HLM	High Level Model(ing)
HLFM	High Level Fault Model(ing)
HLS	High Level Simulation
HLFS	High Level Fault Simulation
TLM	Transistor Level Model
TLFM	Transistor Level Fault Model
TLS	Transistor Level Simulation
TLFS	Transistor Level Fault Simulation
MOR	Model Order Reduction
SI	System Identification
CN	Chebyshev and Newton
DAE	Differential and Algebraic Equation
AMG-CNIP	Automated Model Generation algorithm using CN polynomials
WNS	Weakly Nonlinear System
SNS	Strongly Nonlinear System
LTI	Linear and Time Invariant
LTV	Linear and Time Varying
ss	State-space
1D	One Dimensional
2D	Two Dimensional
ARF	Automatic Range Finder
APT	Automatic Polynomial Tuner
AMC	Automatic Model Conversion
NLTx	Nonlinear Transmission
MMGSD	Multiple Model Generation System using Delta Operator

CHAPTER 1

INTRODUCTION

The progress in silicon fabrication processes drastically reduces feature size and allows more transistors on a die enabling larger chip and more complicated functionality out of a single chip. Meanwhile, fabrication industry lacks of providing efficient facilities to build circuits on chips. This may result in higher possibility of manufacture defects. Therefore, it is vital to test circuits before going to the production stage.

Generally analog and mixed signal (AMS) circuit testing is performed at two stages: initially at design stage and then after chip manufacture. Once design is finalized, faults are injected in the circuit and simulate to determine the possible responses to a given test in the presence of fault. The faulty responses are stored and a fault library is generated. To detect faults after chip manufacture, one tries to match the actual results of test experiment with one of the pre-computed expected results stored in library.

In this dissertation we are concerned with the testing of circuits at design stage as it is more critical, expensive and time consuming.

Several mature methods exist for digital circuits testing, e.g., Design for Testability (DFT) [1] and Built-In-Self-Test (BIST) [2] etc. and can be easily applied to all circuits. Unfortunately, in analogue domain, these techniques can only be employed for certain classes of circuits [3], [4]. Analogue test is very time consuming and expensive, its costs dominate approximately 90% of the whole testing cost in modern AMS ICs [1],[2]. Analogue test suffers from a lack of automated test pattern generation (ATPG) algorithms, long testing time and unknown test quality [7], [8]. Analog testing strongly influences time-to-market and time-to-profit and unfortunately still a major bottleneck in developing mixed-signal ICs.

Traditional analog faulty circuit simulation use SPICE-like simulators that works at the device level, with full scale device models, resulting in extensive simulation run times [9]. It is well known that transistor is a basic building block of modern semiconductor devices. Therefore, transistor level simulation (TLS) plays a major role in fault modeling and simulation. To test faults in transistor level circuit, one need to inject fault in each transistor and run transistor level fault simulation (TLFS) separately for each fault. To test a modern analog and mixed signal circuit containing millions of transistors, TLFS may take impractically long time to complete.

During the last few years, high level modeling (HLM) and high level fault modeling (HLFM) techniques have been proposed for testing and verification of modern complex AMS design due to its high speed [10–14] . High level fault-free modeling may simply indicate (linear or nonlinear) behavior of a fault-free circuit, but it is not able to cope with faulty conditions (that can also be linear or nonlinear). The only way to model faulty circuit at high level is to employ HLFM.

To perform HLM and HLFM *macromodel* of an anlog circuit is required. Macrmodel is simplified representation of the original transistor level circuit that capture the important behaviour of the circuit while discarding the redundant information. Macromodel can be generated manually or automatically. For the former approach, macromodel generation is a tedious process and often fails to capture critical nonlinear effects stemming from unanticipated interactions between blocks, or from second order phenomena in device models. It is in this context that people start seeking possibilities of automated model generation (AMG). AMG techniques take a detailed description of a block – for example a SPICE-level netlist – and generate, via an automated computational procedure, a much simpler macromodel [15].

A number of AMG approaches have proposed, which can roughly be divided into two branches: the fitting approaches and the constructive approaches [16]. The fitting approaches are also know as "black-box" modeling, which treat the system being modeled as a black-box, a prametric model is pre-selected and only the input/output data (from SPICE-level simulations) is used to fit model parameters [17–19]. Typical parametric models that can be adopted include, e.g., lookup tables [20], radial basis functions (RBF) [21], artificial neural networks (ANN) [22], [23] and its derivations such as fuzzy logic (FL) [24] and neural-fuzzy network (NF) [25], and regression [20],[21]. The constructive approaches are completely opposite, which start from detailed SPICE-level descriptions and abstract macromodel automatically via mathematical algorithms in a bottom-up fashion, with no manual knowledge of underlying internal circuit structure or operation required. Compared with black-box counterparts, one of the advantages of such methods is that nonlinearities and dynamics from increasingly sophisticated device models can be naturally and automatically abstracted into the macromodels. However, white-box AMG may produce high order models of excessive complexity (e.g., [28–30]), in which case model order reduction (MOR) techniques are required [31]. A survey paper [32] discusses MOR techniques with respect to various model types, and in a variety of contexts: LTI MOR [33], LTV MOR [34], [35] and weakly nonlinear methods including polynomial-based [36], trajectory piecewise linear (TPWL) [37], piecewise polynomial (PWP) [38], scalable trajectory piecewise linear (STPWL) [39] and transfer function trajectory (TFT) [40].

Unfortunately, there are not many papers describing the use of AMG approaches for HLFM at a system level. For straightforward system simulation relatively simple models may be adequate, but they can prove inadequate during HLFM and HLFS. The accuracy and speedup of existing models may be doubted when fault simulation is implemented because faulty behavior may force (non-faulty) subsystems into highly nonlinear regions of operation, which may not be covered by fault-free models. Authors in [18] first time use AMG approach called multiple model generation system using detal operator (MMGSD) to perform HLM and HLFM at system level. Unfortunately, authors fail to achieve simulation speedup at system level due to computational overhead of model switching for which high level simulator is not optimized.

The general aim of this study is to prove that models generated using automated model generation can be employed to implement HLFM and a simulation speedup at system level can be achieve compared to full SPICE-level circuit simulations.

1.1 Objectives

The objectives of this dissertation are summarized as follows.

1. To develop an AMG algorithm for macromodeling of linear and nonlinear analog circuits.
2. To perform HLM and HLFM at system level and achieve simulation speedup compared to standard SPICE circuit simulations.

1.2 Dissertation Organization

The remaining of the thesis is organized as follows.

- Literature Review: Chapter 2 reviews nonlinear AMG approaches for black-box and white-box types. The approaches are discussed in the context of HLM.
- AMG-CNIP approach: Chapter 3 introduces the major contribution of this dissertation, the AMG algorithm termed automated model generation using Cheybshev and Newton interpolating polynomials (AMG-CNIP). AMG-CNIP generates single model for a given analog circuit (faulty or fault-free) and avoid model switching problem that is the main hurdle in achieving simulation speedup at system level.
- HLM and HLFM using AMG-CNIP: Chapter 4 presents high level modeling (HLM and HLFM) using models generated in last chapter. Two benchmark circuits: nonlinear transmission line circuit and a 3-stage cascaded nonlinear transmission line circuits are used to perform HLM and HLFM at system level.
- Conclusion and Future Recommendations: Chapter 5 concludes this dissertation and provides several suggestions for future improvement of AMG-CNIP.

CHAPTER 2
LITERATURE REVIEW

It is known that a model of a system (also called macromodel) can be generated manually or automatically. This thesis mainly focuses on the latter, i.e., development of an AMG approach for analog circuits. The aim of this chapter is to review some model generation approaches for HLM of nonlinear analog circuits.

2.1 Classification of AMG Approaches

Depending on the amount of information available for modeling analog circuit, either black-box or white-box techniques can be employed. If only system input/output data is available then one can only employ black-box AMG. If system detailed schematic is available and information about internal node voltages and branch currents, then white-box AMG can be used. In this chapter we review System Identification (SI) based black-box AMG and Model Order Reduction (MOR) based white-box AMG.

A fundamental decision in selecting an AMG approach is to choose the right model structure. These model structures are selected according to the type of the circuit or system being analyzed. The main system types include [32]: Linear Time Invariant (LTI), Linear Time Varying (LTV) and nonlinear systems. Once the system type is analyzed, the macromodel can be generated using any model structure, e.g. ss (in form of differential and algebraic equations (DAEs)), transfer function and polynomials etc. Figure 2-1 illustrates the taxonomy of various AMG techniques in pyramid form. A brief description of system types of analog circuits is given in next subsections.

2.1.1 Linear Time Invariant (LTI) System

LTI systems are widely used in electronics design and the AMG techniques developed for them are mature e.g. [8],[9]. The basic structure of a LTI block for mixed signal circuits is illustrated in figure 2-2, where *u(t)* and *y(t)* represent inputs, and outputs to the system in the time domain, respectively. Correspondingly, *U(s)* and *Y(s)* represent *u(t)* and *y(t)* in the Laplace domain.

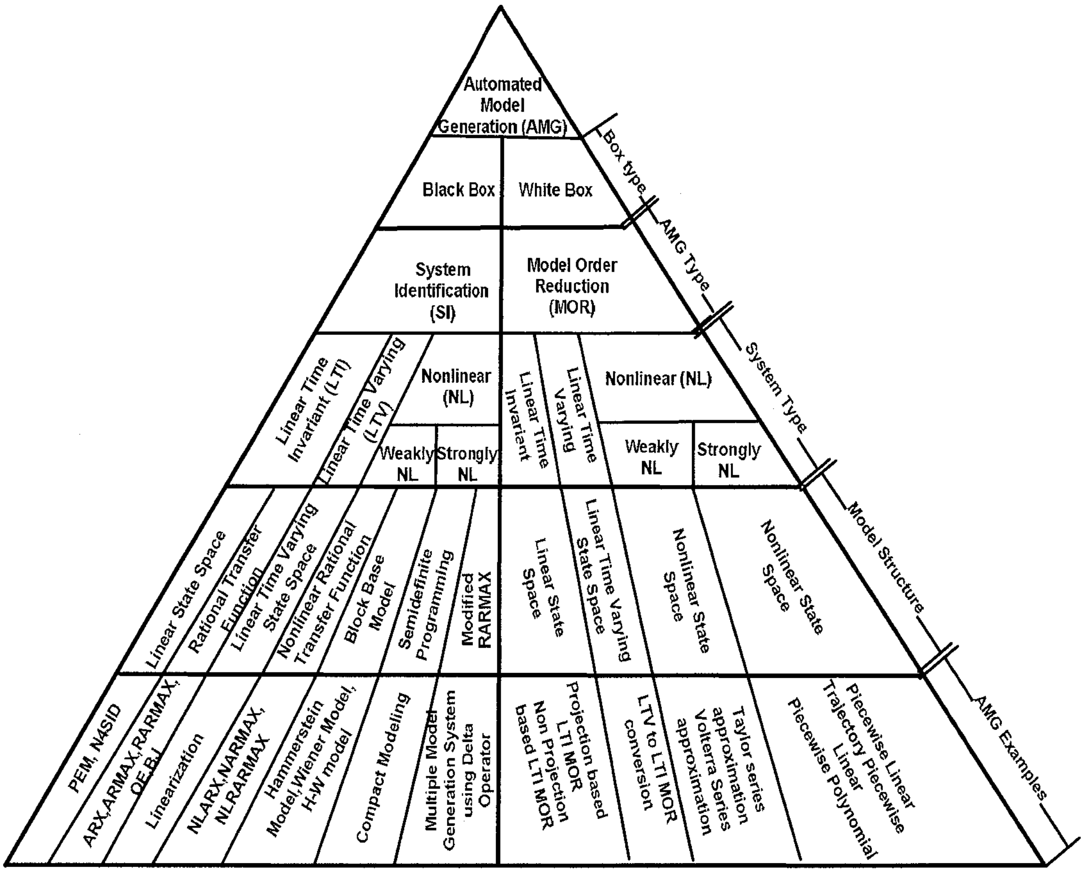

Automated Model Generation (AMG)
Black Box
White Box
System Identification (SI)
Model Order Reduction (MOR)
Linear Time Invariant (LTI)
Linear Time Varying (LTV)
Nonlinear (NL)
Weakly NL
Strongly NL
Linear Time Invariant
Linear Time Varying
Nonlinear (NL)
Weakly NL
Strongly NL
Linear State Space
Rational Transfer Function
Linear Time Varying State Space
Nonlinear Rational Transfer Function
Block Base Model
Semidefinite Programming
Modified RARMAX
Linear State Space
Linear Time Varying State Space
Nonlinear State Space
Nonlinear State Space
PEM, N4SID
ARX,ARMAX,RARMAX, OE,BJ
Linearization
NLARX,NARMAX, NLRARMAX
Hammerstein Model,Wiener Model, H-W model
Compact Modeling
Multiple Model Generation System using Delta Operator
Projection based LTI MOR
Non Projection based LTI MOR
LTV to LTI MOR conversion
Taylor series approximation
Volterra Series approximation
Piecewise Linear
Trajectory Piecewise Linear
Piecewise Polynomial
Box type
AMG Type
System Type
Model Structure
AMG Examples

Figure 2-1: Classification of AMG techniques

An LTI system can be characterized by convolution of the input with an impulse response *h*(*t*) in the time-domain, which transforms in the frequency/Laplace domain into a multiplication relationship, i.e., the input-output relationship can be expressed by partial differential equations (PDEs) or ordinary differential equations (ODEs). These models can be implemented using HDLs such as VHDL-AMS or Verilog-AMS. Examples of LTI systems include RLC interconnects circuits generated by parasitic extractors of digital circuits and linear amplifiers, etc.

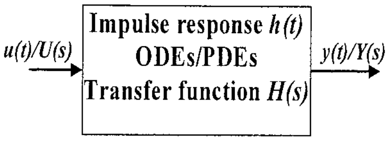


Figure 2-2: Linear Time Invariant (LTI) block [1]

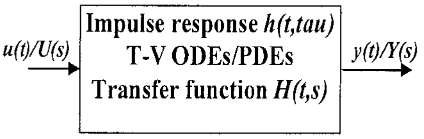


Figure 2-3: Linear Time Varying (LTV)

2.1.2 Linear Time Varying (LTV) System

LTV systems are used in practice because most real-world systems are time-varying as a result of system parameters changing as a function of time. LTV systems can be described by impulse responses in the time domain or transfer functions in the frequency/Laplace domain. The basic structure of an LTV system is depicted in figure 2-3, where $u(t)$ and $y(t)$ represent the inputs and outputs of the system in the time domain, respectively. $U(s)$ and $Y(s)$ are the corresponding signals in the Laplace domain.

The main difference between LTV and LTI systems is that the latter follow the rule that if there is a time-shift in the input, the same time-shift also occurs in output, which does not necessarily occur in LTV systems. LTV systems are able to efficiently model variations with time using the ss form and thus the modeling of LTV systems can be formulated as problems of nonlinear systems, which obey the scalability property, but do not obey the time shift property. More details on nonlinear systems are provided in the next subsection. Examples of LTV systems may include RF mixers, switched capacitors, and sampling circuits.

2.1.3 Nonlinear Systems

Essentially circuits that contain semiconductor devices are nonlinear, most obviously for devices such as diodes and silicon controlled rectifiers where the I-V characteristics change abruptly. Transistors can be modeled as a linear device for small signals, but for large signals they are significantly nonlinear.

A nonlinear circuit or system can be divided into weakly nonlinear system (WNS) or strongly nonlinear system (SNS). The former can be approximated using linearization techniques or modeled using simple nonlinear approximation methods such as Taylor and Volterra series. Strongly nonlinear circuits cannot be modeled using simple series approaches because the useful information lies not only in low order derivatives, but also in high order derivatives that are existed in strong nonlinear effects [42]. Some examples of strong nonlinear effects include the responses generated by high speed digital logic gates, switches, comparators and so on, where rapid switching occurs between two states. Though the switching behavior is inherent in digital systems, the same behavior is also observed in analog circuits such as sampling circuits, switching mixers, analog-to-digital converters (ADC), digital-to-analog converter (DAC) etc.

Unlike linear systems, no technique has been developed yet that can be confidently adopted to capture the characteristics of all kind of nonlinear systems [2]. This is mainly because of extremely complex dynamical behavior in nonlinear systems. During the last decade significant efforts have been put for developing nonlinear AMG. Unfortunately, the majority of these techniques [43–47] are application specific. In next subsections SI and MOR AMG approaches for nonlinear circuits and systems are discussed.

2.2 SI AMG Approaches

SI is the art and science of generating mathematical models from the descriptions of a dynamical system [48]. In general, SI AMG methods generate models using three major steps:

1. Collect data, either from experimentation or simulation of the original system; dividing the data into two parts, one for model generation and the other for model verification.
2. Choose a model set.
3. Pick the 'best' model from the set.

It is quite likely that the first model obtained will not pass model validation tests. In which case the identification process in continued until the model is validated [49–51]. Successful model generation requires suitable stimulus inputs that can excite all the possible states of the system.

A number of SI approaches exist for the modeling of linear systems [52–54]. Details on modeling of nonlinear circuits and systems using nonlinear AMG is given in next section.

2.2.1. SI AMG for Nonlinear Systems

Transfer function modeling techniques of linear systems can be extended for AMG of WNS. One example is nonlinear ARX (NLARX) [27], whose model structure is shown in (2-1).

$$y(t) = F(y(t-1),...,y(t-n_a),u(t),u(t-1),...,u(t-n_b)) \qquad (2\text{-}1)$$

The function F depends on a finite number of previous inputs u and outputs y. n_a, n_b represent the number of delayed output and input values that are used for the prediction of the current output. The model structure is shown in block form in figure 2-4.

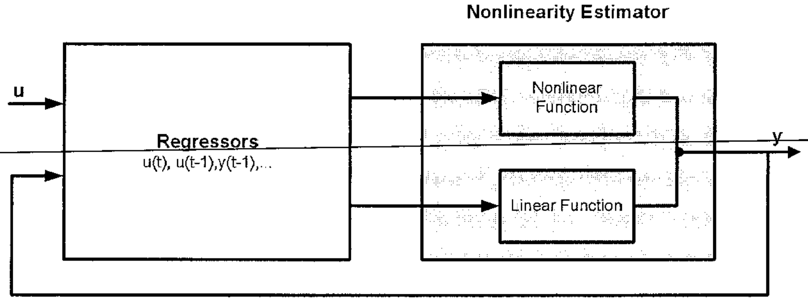


Figure 2-4: NLARX model structure, see equation (2-1) [17]

F is a nonlinear function and the inputs to F are model regressors, composed of delayed input and output values, $u(t),u(t-1),...,u(t-n_b)$ $y(t-1),...,y(t-n_a)$.

The first block in figure 2-4 is a regressor block that computes the regressor on the basis of the current and past input and output values. The second block is a nonlinearity estimator. This block implements two functions, one a linear function and the other a nonlinear function. The outputs from the regressor block are mapped to the model output y using these functions. The combined function $F(x)$ implemented by nonlinearity estimator is shown in (2-2).

$$F(x) = L^T(x-r) + d + g(Q(x-r)) \qquad (2\text{-}2)$$

where x is a regressor vector. $L^T(x)$ is the output of the linear function block and $g(Q(x-r))$ is the output of nonlinear function block. Examples of nonlinear estimators used in the nonlinear function block include sigmoid networks, tree partitions, wavelet networks and neural networks [27].

Another approach to decomposing a system into linear and nonlinear functions involves the dynamics of the system being controlled through linear functions, with the nonlinear behavior captured through functions consuming the inputs and outputs of a linear block. The Hammerstein-Wiener (*H-W*) model [55] shown in figure 2-5 is an example of this configuration. Nonlinearity in both input and output is treated in separate blocks with a linear block connecting them.

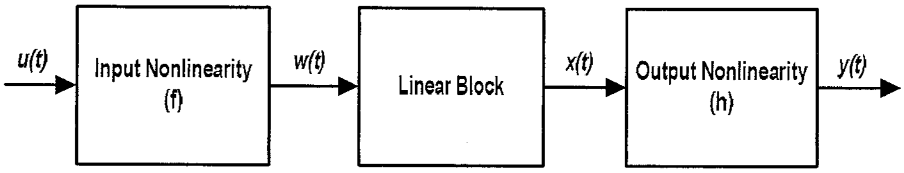


Figure 2-5: Hammerstein-Wiener (H-W) model [17]

where

- Block 1 implements nonlinear function *(f)* on input *u(t)* and generates output *w(t)*;
- Block 2 implements a linear transfer function between nonlinear input *w(t)* and linear output *x(t)*;
- Block 3 implements nonlinear function *(h)* and generates nonlinear output *y(t)* by consuming inputs *x(t)* from linear block.

As the block 1 takes care of nonlinearities in input, the function *f* is called the *input nonlinearity*. Similarly, function *h* (block 3) is called the *output nonlinearity*. If the model structure contains only block 1 for nonlinearity, i.e., treating nonlinearities only in input, then it is called a *Hammerstein* model. Similarly, if the model structure treats nonlinearities only in the output; it is a *Wiener* model. The combination of both nonlinearities in a single model is called an *H-W* model. Several algorithms are available for nonlinearity estimators *f* and *h*, such as piecewise linear, one layer, sigmoid network, wavelet network, saturation and dead zone [27].

Partitioning of a nonlinear system in to linear and nonlinear blocks in an *H-W* model generates better output than the NLARX model [17]. However, due to lack of feedback, the model may become unstable for larger systems. An inherent shortcoming in this model structure is that a *H-W* model considers nonlinearity as a "static" characteristic of a system, which is usually the case with control systems.

However, in modern mixed signal electronic circuits, nonlinear behavior is highly dynamic (nonlinearity changes with time), e.g., nonlinear effects generated by cross talk, noise, and intermodulation phenomena etc. The *H-W* model is also discrete time like NLARX and uses the same nonlinear functions, such as sigmoidnet, wavenet etc., for nonlinearity estimation.

We start investigation for the development of AMG using *NLARX* and *H-W* AMGs. According to the results [19], [56], [57] it is clear that both AMGs are able to capture weak nonlinear effects, but become computationally expensive when run at system level after conversion into VHDL-AMS behavioral models. Furthermore, both AMG approaches perform HLM with poor accuracy, and high level simulation (HLS) speed is significantly slower than standard transistor level SPICE circuit simulation (gold reference).

To deal with dynamical nonlinear behavior, Simeu *et al.* [58] propose another approach termed as Situation Dependent ARX (SDARX) model. SDARX extends the idea of the general nonlinear ARX (NARX) (given in equation 2-3) by adding situation dependent coefficients in the regressor term $\gamma(t)$ of the model, shown in (2-4), where, $\phi_o(\gamma(t)), \phi_i^y(\gamma(t))\big|_{1\le i\le n_y}$ and $\phi_i^u(\gamma(t))\big|_{1\le i\le n_u}$ are the data dependent coefficients of the model. A Radial Basis Function (RBF) method is used to estimate these situation dependent coefficients. The main idea of the SDARX approach is to divide the parameter search space into two subspaces: linear weight subspace and nonlinear parameter subspace.

$$y(t) = f(y(t-1),\ldots,y(t-n),u(t-1),\ldots,u(t-n_u)) + v(t) \qquad (2\text{-}3)$$
$$y(t) = f(\gamma(t)) + v(t)$$

$$y(t) = \phi_o(\gamma(t)) + \sum_{i=1}^{n_y} \phi_i^y(\gamma(t))y(t-i) + \sum_{i=1}^{n_u} \phi_i^u(\gamma(t))u(t-i) + \varepsilon(t) \qquad (2\text{-}4)$$

Then a search is made for the best model from these search spaces, applying optimization techniques to get accurate results. To avoid computational overhead, optimization is applied only to the linear subspace. SDARX models weak dynamical nonlinear behavior with good accuracy. Unfortunately, this approach is computationally complex and hence may become unsuitable for HLM, as it may not be able to increase HLS speed. In addition the SDARX model is only applicable to SISO systems.

SI AMG approaches discussed above are only able to capture weak nonlinear effects. To model strong nonlinear circuits and systems based on input/output data, more powerful identification AMGs are required. There are several techniques available for SI based AMG for SNS. Two recently proposed techniques are discussed: Compact Modeling (CM) [17] and Multiple Model Generation System using Delta operator (MMGSD) [18].

In CM, the model identification procedure is based on minimizing the model error over a given training data set subject to an incremental stability constraint, which is formulated as a semidefinite optimization problem [17]. Initially the system of implicit form seen in (2-5) is linearized to get a linearized output error upper bound r over the training data set X.

$$\begin{aligned} &F(v[t],\ldots,v[t-m],u[t],\ldots,u[t-k])=0,\\ &G(y[t],v[t])=0 \end{aligned} \tag{2-5}$$

where $v[t] \in R^N$ is a vector of internal variables, $y[t] \in R^{Ny}$ is the output, $u[t] \in R^{Nu}$ is the input, $F \in R^N$ is a dynamical relationship between the internal variables and the input, and $G \in R^{Ny}$ is the static relationship between internal variables and output. Then a stability constraint is enforced on the system through the use of a storage function H.

The formulations that enforce both stability and accuracy on the linearized output error upper bound r at the same time are shown in (2-6).

$$\min_{r,F,G,H} \sum\nolimits_t r_t \ \textit{subject to}$$

$$r_t + 2\delta_o^T \overline{F}_t(\Delta) + 2\xi^T \overline{G}_t(\delta_o,\xi) - |\xi|^2 + h_{t-1}(\Delta-) - h_t(\Delta+) \geq 0 \quad \forall t,\Delta,\xi \tag{2-6}$$

where r_t output error upper bound is a function of training data set $\widetilde{X}[t] = (\widetilde{y}[t],\widetilde{V}[t],\widetilde{U}[t])$ given by $r_t = r(\widetilde{y}[t],\widetilde{V}[t],\widetilde{U}[t])$, F and G are dynamical and static relationships of input and outputs represented respectively by $\overline{F}_t(\Delta) = \overline{F}(\widetilde{V}[t],\widetilde{U}[t],\Delta)$, $\overline{G}_t = \overline{G}(\widetilde{y}[t],\widetilde{v}[t],\delta_o,\xi)$, Δ and ξ represents the incremental variables used for incremental stability. h_t is the storage function that depends on internal state $\widetilde{V}$ and incremental variable Δ, given by $h_{t-1}(\Delta-) = h(\widetilde{V}-,\Delta-)$, and $h_t(\Delta+) = h(\widetilde{V}-,\Delta-)$ [17].

All functions *F, G, H* and *r* in equation (2-6) are unknowns. Estimation of these unknowns can be treated as an optimization problem and can be converted into semidefinite program (SDP) by allowing the unknown functions *F, G, H* and *r* to be chosen as linear combinations of a finite set of basis function Ø, given in (2-7), where $Ø^F$, $Ø^G$, $Ø^H$, $Ø^r$, $\in \phi$, and α^F, α^G, α^H, α^r are the free variables solved through SDP.

$$F=\sum_{j\in N_f}\alpha_j^F\phi_j^F(V,U), G=\sum_{j\in N_g}\alpha_j^G\phi_j^G(y,v_o)$$
$$H=\sum_{j\in N_h}\alpha_j^H\phi_j^H(V), r=\sum_{j\in N_r}\alpha_j^r\phi_j^r(y,V,U) \qquad (2\text{-}7)$$

A SDP is a problem whose objective function is linear and whose constraints require matrices which are positive semidefinite (PSD)[17]. SDP is a special case of convex optimization concerned with the optimization of a linear objective function subject to the constraint that the affine combination of symmetric matrices is positive semidefinite. Such a constraint is nonlinear, or not smooth, but convex in nature. Hence, SDP is considered to be a special case of convex optimization. A general form of SDP that minimizes a linear function of a variable $x \in R^m$ subject to a matrix inequality is shown in equation (2-8), where $F(x) \cong F_o + \sum_{i=1}^{m} x_i F_i$.

$$\begin{array}{ll} \textit{Minimize} & C^T x \\ \textit{Subject to} & F(x) \geq 0 \end{array} \qquad (2\text{-}8)$$

The SDP problem data in (11) are the vector $c \in R^m$ and $m+1$ symmetric matrices $F_0, \ldots, F_m \in R^{n*n}$. $F(x) \geq 0$means that $F(x)$ is positive semidefinite, i.e., $z^T F(x) z \geq 0$ for all $z \in R^n$.

The benefit of formulating the optimization problem (2-6) as an SDP (2-8) is that it can be solved efficiently using readily available software routines [59–61]. However, the complexity of the optimization problem (2-7) depends heavily on the choice of the basis function Ø for the unknowns *F, G, H* and *r*. Therefore, the selection of the basis is critical to obtain a feasible solution [18]. CM utilizes a polynomial basis to efficiently solve the optimization problem. This allows CM to identify a system as a rational model structure. However, only models that are linear in unknown state variables are considered. To achieve compactness in the identified rational model, reduction of states for larger systems are attained through conventional projection methods and further reduction of polynomial basis is obtained through a fitting procedure.

The model generated by CM very closely resembles the original system [62] and guarantees stability in the generated compact model. For example, in [17] a compact model of a Micro Electro-Mechanical Systems (MEMS) device of order 400 is generated. Simulation results for its nonlinear behavior show good accuracy indicating that CM is able to generate stable reduced models for originally high order systems.

Another AMG for SNS targeted HLM and HLFM of analogue circuits has been proposed by Xia *et al.* [18]. The algorithm, termed multiple model generation system (MMGS) consists of two parts: the Automated Model Estimator (AME) and the Automated Model Predictor (AMP). The AME implements the model generation process using Recursive Maximum Likelihood (RML) [27] method and AMP uses these models to predict signals in the simulation [63]. The AME comprises three stages: pre-analysis, estimator and post-analysis. In pre-analysis stage initial conditions of MMGS are setup e.g., input range and the number of submodels. This step is only performed once in MMGS. The estimator implements RML and provides output responses and error measures. The post-analysis step implements the model generation process.

MMGS produces a new model for different nonlinear regions error against input voltage. The model structure used by MMGS for these regions is shown in (2-9) [63].

$$y(t) = -a_1 y(t-1) - a_2 y(t-2) - \ldots a_{na} y(t-na) + b_1 u(t-1) + b_2 u(t-2) + \ldots b_{nb} u(t-nb) \qquad (2\text{-}9)$$

The model is calculated using the linear regression in (2-10), where θ is the parameter vector shown in (2-11), $\varphi(t)$ is the regression vector displayed in (2-12).

$$y(t) = \varphi^T(t)\theta \qquad (2\text{-}10)$$

$$\theta = [a_1\ a_2\ \ldots\ a_{\mathrm{na}}\ b_1\ b_2\ \ldots\ b_{\mathrm{nb}}]^T \qquad (2\text{-}11)$$

$$\varphi^T(t) = [-y(t-1)\ldots - y(t-na) \quad u(t-1)\ldots u(t-nb)] \qquad (2\text{-}12)$$

MMGS generates discrete time models that are able to accurately model faults for low frequency circuits. However, these models get contaminated with the effects of aliasing and severe phase shift for high frequency circuits [64].

The shortcomings in MMGS are improved by introducing a delta operator *(δ)* in MMGS model structure that leads to the development of MMGSD which handles nonlinearity over time. An attractive property of the *delta* operator is that it produces model coefficients that approximate the discrete time models as continuous time models.

To present the model structure in continuous time, the inverse Laplace transform of the transfer function shown in (2-13).

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_0 s^n + b_1 s^{n-1} + \dots b_n s^0}{s^m + a_1 s^{m-1} + \dots a_m s^0} \tag{2-13}$$

It can be used if the sampling interval is sufficiently short [26]. The resulting equation in time domain with the *delta* operator can be presented as equation (2-14) [63].

$$G(\delta) = \frac{y(t)}{u(t)} = \frac{b_0 \delta^n + b_1 \delta^{n-1} + \dots b_n \delta^0}{\delta^m + a_1 \delta^{m-1} + \dots a_m \delta^0} \tag{2-14}$$

After arranging terms in (2-14), the system in (2-15) is obtained.

$$y(t)\delta^m = -(a_1 \delta^{m-1} + \dots + a_m) y(t) + (b_0 \delta^n + \dots + b_n) u(t) \tag{2-15}$$

System in equation (2-15) is solved using RML appropriately modified using the *delta* operator [63]. Unlike in [17], [37], [65], Xia *et al.* use single training data to excite all possible states by superimposing pseudorandom binary sequence (PRBS) signal on a linear waveform . In addition, HLM and high level fault modelling (HLFM) are performed by converting the MMGSD model into a VHDL-AMS model through a process called multiple model conversion system using a delta operator (MMCSD). MMCSD loads MMGSD model parameters and generates HLM and HLFM implemented in VHDL-AMS. Accurate simulation results are achieved. Unfortunately, simulation speed up in high level fault simulation (HLFS) is not achieved compared with transistor level fault simulation (TLFS) because of computational overhead in the HDL simulation. The overhead is mainly due to the fact that the MMGSD has to discretely switch between multiple models for a single fault during simulation. Speed can be improved by implementation of fault collapsing, to prevent repetition of effort, and by optimizing the number of models generated to prevent unnecessary model switching from consuming computational effort. Intelligent setting of thresholds for model switching during MMGSD may achieve this.

In conclusion SI AMG approaches are able to capture nonlinear effects generated by analogue circuits based on only input/output data of system. However, based on existing literature and through our experimentation results in [19], [56], [57], SI AMG approaches are not efficient for performing HLM at system level because:

- Majority of available SI AMGs are discrete time, whereas analogue circuits are continuous time in nature, hence continues time model structures are more suitable for their modeling.
- Since no circuit internal information is used during modeling process, AMG has to make its own guess and increase model order in order to accurately capture nonlinear effects resulting in computational overhead.
- HLM and HLFM generated using SI AMG becomes computationally expensive and are unable to achieve simulation speedup as compared to standard SPICE circuit simulation.

2.3 MOR AMG Approaches

When circuit internal information (node voltages, branch currents etc.) are available during macromodeling process, then white-box MOR AMG approaches can be used. A MOR AMG approach mainly captures circuit internal information through a ss model structure shown in equation (2-16) for macromodeling of linear systems.

$$E\dot{x} = Ax(t) + Bu(t), \qquad (2\text{-}16)$$

$$y(t) = C^T x(t) + Du(t)$$

where $x(t) \in R^n$ is the internal state variable consists of node voltages and branch currents, $u(t) \in R^m$ and $y(t) \in R^p$ are the input and output waveforms. Matrices $A \in R^{nxn}, E \in R^{nxn}, B \in R^{nxm}$, and $C \in R^{pxn}$ are constants.

First equation in (2-16) is called 'state equation' and second equation is called 'output equation', where output $y(t)$ is a linear combination of states $x(t)$.

Since ss captures complete circuit internal information, the order (dimensions) of resulting state variable x is high. It is inefficient to simulate macromodel in its original form as it is computationally expensive due to high model order. Therefore, model order reduction (MOR) methods are applied. MOR methods projects original ss to a lower dimension subspace that capture only critical information of the circuit and confiscate redundant information. Therefore, task of a MOR method is to produce a model of lower order whose response matches with original system response with good accuracy. Also, the reduced model should maintain the desired properties of the original system with minimum computational overhead, less memory requirements and shortest evaluation time.

There are two types of MOR techniques for linear systems: projection based and non-projection based techniques. Projection methods reduce model order (*n*) by projecting original ss (of dimension *n*) to lower dimension ss (of dimension *q*) by using projection matrices *U* and *V*. The most widely used approaches are Proper Orthogonal Decomposition (POD) methods [66–69], Krylov-subspace and shifted Krylov-subspace methods [70–74], and Balancing-based methods [75–77]. Non-projection methods do not employ projection matrices to reduce ss dimensions. Instead these methods directly approximate lower order ss from original higher order ss by applying fitting and optimization methods etc. Some of the examples of non-projection methods includes Hankel optimal model reduction [78], singular perturbation method [79], and various optimization-based methods.

In the next subsection MOR approaches for nonlinear systems are reviewed.

2.3.1. MOR AMG for Nonlinear Systems

If a circuit contains nonlinear components such as diodes, transistors etc. then the linear ss models structure shown in equation (2-16) cannot be used for macromodeling of nonlinear analog circuits. A standard nonlinear system formation shown in (2-17) is needed based on a set of nonlinear differential-algebraic equations (DAEs).

$$\begin{aligned} \dot{q}(x(t)) &= f(x(t)) + bu(t) \\ y(t) &= c^T x(t) \end{aligned} \qquad (2\text{-}17)$$

where $x \in R^n$, *n* is the order of matrices, *x*(*t*) represents the state vector that can represent node voltages and branch currents, *y*(*t*) is the vectors of outputs that can be again node voltages and branch currents, *u*(*t*) is the input to the circuit, *q*(.) and *f*(.) are nonlinear vector functions. Function *q*(*x*(*t*)) represents nonlinear charge contribution in the circuit that is a function of state vector *x(t)* and function *f*(*x*(*t*)) represents nonlinear current in the circuit that is also a function of state variable $x(t)$. b and c are the constant input and output matrices, respectively.

In the systems where charge (and capacitance) is considered to be linear, the nonlinear ss representation (2-17) is reduced to the form shown in equation (2-18), where E is a constant matrix.

$$\begin{aligned} E\frac{d}{dt}x(t) &= f(x(t)) + bu(t) \\ y(t) &= c^T x(t) \end{aligned} \qquad (2\text{-}18)$$

Critical task in macromodeling of nonlinear circuits using equation (2-17) is the representation of nonlinear functions $q(.)$ and $f(.)$ in ss. Conventional approach is to expand the functions using Taylor polynomials as shown in equation (2-19).

$$\begin{aligned} &\frac{d}{dt}(q(x_0)+G_1(x-x_0)+G_2(x-x_0)\otimes(x-x_0)+\cdots G_i(x-x_0)^{(i)}) = \\ &f(x_0)+A_1(x-x_0)+A_2(x-x_0)\otimes(x-x_0)+\cdots A_i(x-x_0)^{(i)}+bu(t) \end{aligned}$$

(2-19)

where $\otimes$ is the Kronecker tensor product operator and G_i and A_i are the i^{th} order derivative of nonlinear functions $q(x)$ and $f(x)$ respectively and are given as:

$$A_i = \frac{1}{i!}\frac{\partial^i f}{\partial x^i}\Big|_{x=x_0} \in R^{nxn^i}, \quad G_i = \frac{1}{i!}\frac{\partial^i q}{\partial x^i}\Big|_{x=x_0} \in R^{nxn^i}$$

It is important to notice here that nonlinear functions $q(.)$ and $f(.)$ in equation (2-17) do not have explicit formulations, instead these terms come directly from circuit simulator. Basically MOR AMG methods use standard SPICE circuit simulation data to obtain training information for macromodel generation. SPICE linearizes nonlinear components of the circuit using Taylor polynomial in its Newton Raphson loop [80]. MOR AMG methods directly extract Taylor polynomial terms from SPICE simulator and use as linearization information in nonlinear ss form shown in equation (2-19) for macromodel generation.

Once a nonlinear system is linearized, linear MOR techniques (projection or non-projection based) mentioned earlier can be applied to reduce the system order.

As discussed earlier in section 2.1.3 nonlinear analog circuit can be classified as WNS or SNS. If a system response can be approximated using single Taylor polynomial, then the system is considered as WNS. If a system response cannot be captured with single polynomial then the system is considered as SNS. For later case, piecewise linearization (PWL) approach is used. PWL linearizes a nonlinear system at different operating points using Taylor polynomials and the final model is a combination of different linearized models.

The basic idea behind macromodeling a WNS is to approximate a nonlinear system with Taylor polynomial such that MOR methods for linear system can be utilized. As long as higher order terms are in Taylor polynomial, the system cannot be consider as linear. Therefore, authors in [81] and [82] apply perturbation and relaxation approaches to realize Taylor polynomial system as a linear system. Basically these methods consider each term in Taylor polynomial as a separate linear system in which the input of current system is the output of preceding system. However, the authors do not suggest method for the model order reduction of resulting system. To solve this shortcoming, Roychowdhury *et al.* [83] improve the relaxation approach by applying a separate projection basis at each stage to obtain a reduced model. Although authors achieve good accuracy for reduced model, but resulting model is still not reduced as the final model is the union of separate reduced models. Li *et al.* [36] combine and extend Volterra and projection approaches above using a method termed NORM (Nonlinear Model Order Reduction Method) to achieve reduced model size. Instead of generating separate projection basis for each stage, NORM generate a single projection basis for all stages to reduce the system shown in equation (2-19) as proposed in [36]. Compared with existing projection based reduction models, such as[34][84], it provides a significant reduction in model size.

The polynomial based techniques mentioned above attempt to linearize the nonlinear part of a system and then apply model reduction [83],[36],[85]. Other techniques perform the nonlinear model reduction process using various approaches by splitting a system into linear and nonlinear parts, then a reduction technique is applied only to the linear part. After that the linear part is stitched back with the untreated nonlinear part to get the overall reduced nonlinear model [86],[87]. However, these approaches suppose less number of nonlinear elements in the circuits that is not a practical assumption for modern semiconductor nonlinear analogue and mixed signal circuits.

It is known that strong nonlinear effects often appear in higher order derivatives [38]. Unfortunately, the MOR based AMG techniques for WNS are only able to capture nonlinear effects that lie within low order derivatives of a system transfer function. To overcome the issue above , other methods such as piecewise approximation [15] can be used to achieve better solutions. The simplest approach is to represent a nonlinear system using PWL approach. Each small region in the nonlinear response is readily linearized, and a combination of these linear pieces can approximate an overall nonlinear system. However, a major shortcoming is that the number of pieces (regions) increases drastically for strong nonlinear systems in order to achieve sufficient accuracy, which will result in long computation process.

To overcome this problem of explosion of regions, Rewienski *et al.* [27] developed an approach termed trajectory piecewise-linear (TPWL). Initially they select multiple points along a trajectory in the ss of a nonlinear system to generate an equal number of linear approximations. These points are called “center points” and are generated using application specific training inputs. A model is generated if the current state point x is ‘close enough’ to the last linearized point x_i, i.e., $\|x - x_i\| < \varepsilon$, which means that x lies within a circle of radius of ε and centered at x_i. Each of the linearized models takes the form shown in (2-20), with expansions around states x_0 ,..., x_{s-1}, where $f(.)$ evaluated at states x_i. x_0 is the initial state of the system and A_i are the Jacobians evaluated at x_i.

$$\frac{dx}{dt} = f(x_i) + A_i(x - x_i) + Bu \tag{2-20}$$

A Krylov subspace projection method is then required to reduce the order of the linear model within each piecewise region. Rewienski *et al.* then combine all s linear models according to a weighting equation in (2-21), where $\widetilde{w}_i(x)$ are the weights depending on state x.

$$\frac{dx}{dt} = \sum_{i=0}^{s-1} \widetilde{w}_i(x) f(x_i) + \sum_{i=0}^{s-1} \widetilde{w}_i(x) A_i(x - x_i) + Bu \tag{2-21}$$

Rewienski *et al.* state that TPWL is more suitable for circuits that exhibit strong nonlinear effects such as comparators. They also prove that TPWL is more efficient than PWL with respect to the regions explosion problem. They employ application specific training inputs for macromodel generation, which implies that the macromodel covers only those ss trajectories that are generated through specific inputs. TPWL macromodels are able to model strongly nonlinear systems, but the capability to model weak nonlinear systems is limited due to poor approximation properties of linearized models for higher order derivatives. Also such macromodels may be unable to generate correct responses for inputs not covered by training stimuli. Moreover, the computational cost involved in generating reduced macromodels for SNS is high.

To improve this, Vasileyev *et al.* [88] introduce a two-step hybrid reduction method called TBR-TPWL model reduction. Initially they reduce the ss matrices dimension using a conventional Krylov subspace method and further reduction is achieved through use of TBR projection. Simulation of macromodels generated using TPWL-TBR provides better accuracy than models generated using only the Krylov subspace method. Also, computational cost is significantly reduced, but unfortunately it does not guarantee stability.

TPWL and its variants discussed above can model SNS with good accuracy, but they have poor approximation properties for WNS. The major reason of failure is that they use only first order term of Taylor polynomial in each region which limits the scope of macromodel local to an operating region. In other words, local approximation properties of Taylor polynomials are proved to be poor due to consumption of pure linear model in each piecewise region. Therefore Dong *et al.* [65],[38] proposed a piecewise polynomial (PWP) extension of TPWL. This is a combination of polynomial model reduction with the trajectory piecewise linear method. PWP uses higher order Taylor polynomial in each piece wise region. It is able to improve TPWL by dividing the nonlinear ss into several regions that are approximated with a high order Taylor polynomial model around the center expansion point. A training simulation employing DC sweeps can be used to generate these expansion points. The resulting model is gradually developed by introducing new regions until the desired accuracy is achieved. Firstly a polynomial function is expanded into many points, and then simplified through approximation of the nonlinear functions in each region to obtain much smaller size models. The resulting models are then combined as a single model. A scalar weight function is used to ensure fast and flat switching from one region to another.

PWP is implemented in [44] for extracting broadly applicable general-purpose macromodels from SPICE netlists in which the PWP model is able to model different nonlinear behaviors such as loading effects, simultaneous switching noise (SSN), crosstalk noise and so on. A simulation speed up of eight times is reported [89].

One of the major advantages of PWP is that it can model not only weakly nonlinear effects (such as distortion and inter-modulation) but also strongly nonlinear system dynamics (such as clipping and slewing). Moreover, fidelity in large-swing and large-signal analysis can be retained. However, a combination of several training data is used to excite different operating regions.

Authors in [39] focus on the interpolation properties of TPWL models and propose an algorithm called scalable trajectory piecewise linear (STPWL). Original TPWL macromodel extract linearization information from circuit simulator in the form of Taylor polynomials using application specific training inputs. During evaluation process macromodel interpolate between excited and non-excited states. Tiwary *et al.* show that interpolation properties of existing TPWL method are poor and propose a scalable architecture using Gaussian Mixture Models (GMM). The GMM method increases the likelihood of generating unreachable states in the ss that were not excited during model training. Tiwary *et al.* also propose the implementation of STPWL models in commercial SPICE3f5 [90] simulator as a voltage controlled voltage source (VCVS) SPICE component, which means that macromodel can be simulated at system level with other SPICE components. Their method shows a simulation speed up (~3.8*X*) at system-level.

Dimitri *et al.* [40] further improve scalability property of TWPL macromodels and propose a method called Transfer Function Trajectory (TFT). TFT transforms state linearization matrices (obtained through training trajectories) into frequency domain and cast the problem of finding non-reachable states as a regression fitting problem. They use vector fitting (VF) algorithms to construct model for non-reachable states in frequency domain. The authors also translate macromodels into VHDL-AMS behavioral models and achieve speed up to 10.3 X to 110 X compared to existing TPWL approaches. However, both authors in [39] and [40] perform system level simulation under fault-free conditions and fault-modeling for faulty-analog circuits is not performed. Further survey on development of trajectory based MOR methods for nonlinear circuits and systems can be seen in a recent paper [91].

Overall review reviles that the white-box nonlinear AMG approaches use Taylor polynomial as their foundation for linearization of nonlinear elements in the ss. The authors target to improve properties of Taylor polynomials macromodels somehow by proposing different methodologies/architecture on the top of basic Taylor implementation. This implies that root problem cause still exists in all techniques discussed. Therefore, a more proficient way is to replace Taylor polynomial with an efficient one that has good interpolation/approximation properties and enable macromodel to run at higher speed. It is also noticed that except in few cases the existing AMG approaches only show speedup in MATLAB environment, and macromodels are not actually translated to HLM to run at system level and show simulation speed up at system level as compared to standard SPICE circuit simulations. Similarly except for few authors [18] , none of the technique is practically used to perform HLFM at system level.

We propose a novel AMG method in which multiple Taylor polynomials macromodel is replace with single polynomial macromodel. Polynomial is generated through a fusion of Chebyshev and Newton interpolating polynomial. Resulting macromodel is then employed in nonlinear ss model structure to linearize all nonlinear components in the circuit. The proposed macromodeling process is discussed in the next chapter and then HLM and HLFM will be performed at system level using proposed AMG macromodels.

CHAPTER 3

METHODOLOGY

3.1. Introduction

In this chapter we introduce the methodology termed automated model generation using Chebyshev Newton interpolating polynomials (AMG-CNIP). It is the fusion of Chebyshev and Newton (CN) interpolating polynomials (CNIP).

It is discussed in chapter 2 that white-box AMG use multiple Taylor polynomial models in each region to capture nonlinear behavior. Intuitively an efficient solution to avoid model switching can be to use single high order Taylor polynomial. However, it is known that Taylor polynomial has poor convergence properties due to non-uniform approximation error distribution [92]. Approximation error is less as long as input stays close to expansion point, but increases rapidly as input moves away from expansion point. This means that for inputs far from training inputs, even higher order Taylor polynomials cannot predict the accurate response [93].

The situation depicted in figure 3-1 shows why Taylor polynomial based macromodel require multiple models to model compete circuit response. Solid line (above) is circuit input and solid line (below) is the circuit response. Both input and output are shown on same (time vs. amplitude) axes. To approximate complete circuit response, Taylor polynomial macromodel require multiple models generated at multiple linearization points (black dots in figure 3-1). A Taylor polynomial linearized model generated around a linearization point (black dots in figure 3-1) can be considered as linear in that region and is valid only for that region. Outside the region, this model cannot predict response of other region and new linearized model is required to be generated for next region with new linearization point. This means that macromodel generated using Taylor polynomial cannot efficiently interpolate between excited and non-excited states in ss with *single* linearzed Taylor model.

It can be deduced here that if a polynomial has good approximation properties also for large magnitude inputs, then macromodel can efficiently interpolate between different states to predict correct response with a single model. Hence we cast the state prediction problem as an interpolation problem. This implies that macromodel response for non-excited states mainly depends on the interpolation properties of a polynomial used for the linearization of nonlinear components in the circuit. And in turn interpolation properties depend on the approximation properties of a polynomial.

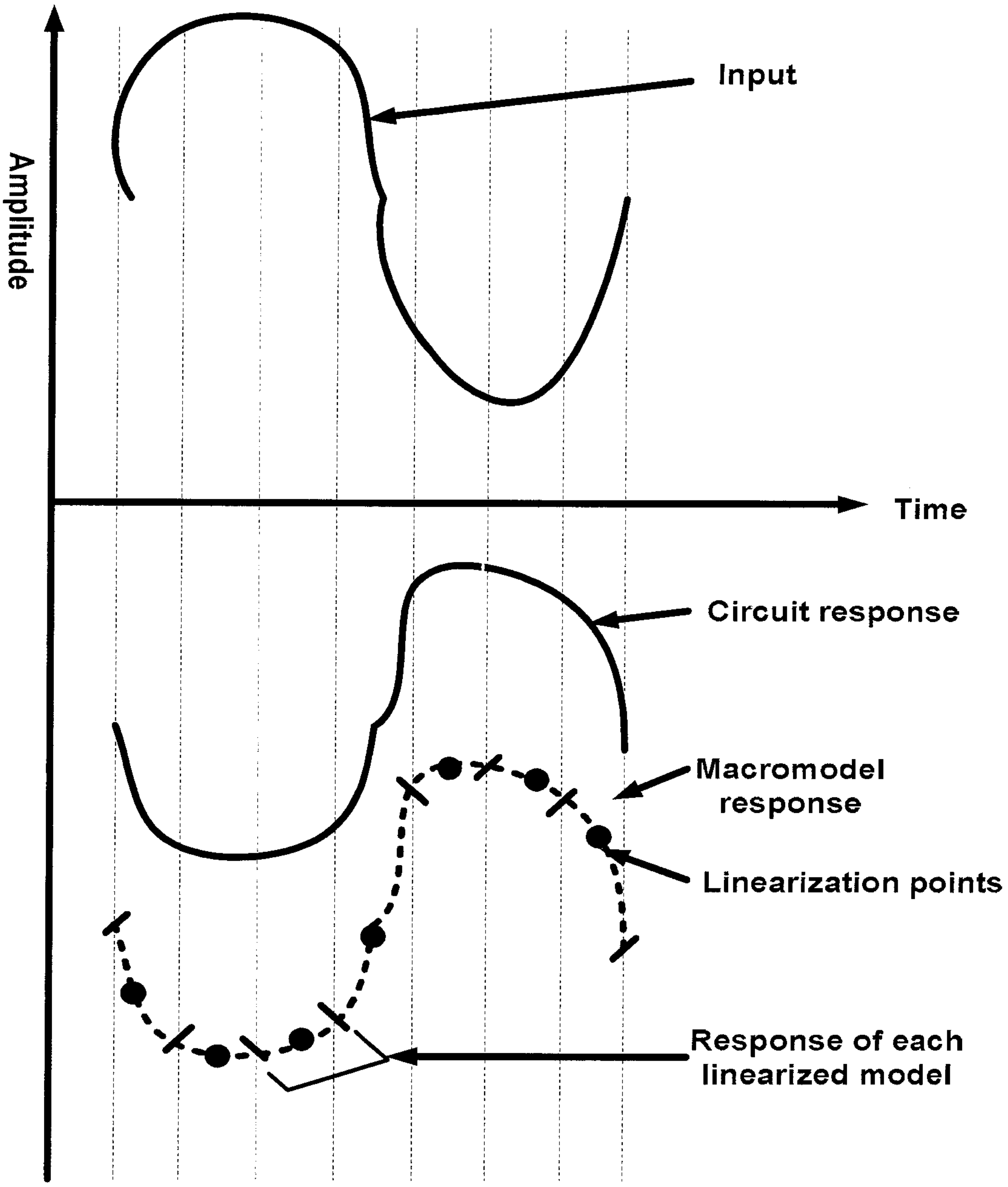


Figure 3-1: Multiple models using Taylor polynomials

To clarify the rationale here, we explain the problem through a numerical example. Consider a nonlinear function $f(x)=\frac{x^2 \exp^{-\frac{x^2}{2}}}{\sqrt{2\pi}}+x\sin(2\pi x)$, which is evaluated at 100 points of x = -2.5:0.05:2.5, shown in blue with circles in figure 3-2. To approximate this function with Taylor polynomial, Taylor series expansions of different orders are performed around the origin. One can observe from figure 3-2 that as long as input x is small (within the range of (-1, 1)), fit for all Taylor polynomials is good. As input value x become larger, Taylor polynomials response start diverging from the solution and although Taylor polynomial of order 45 fit the complete response but it is impractical to use such high order polynomial in ss to model an analog circuit. This clearly shows that Taylor polynomials are effective only for inputs have small magnitude and their approximation is poor for input signal having large magnitude.

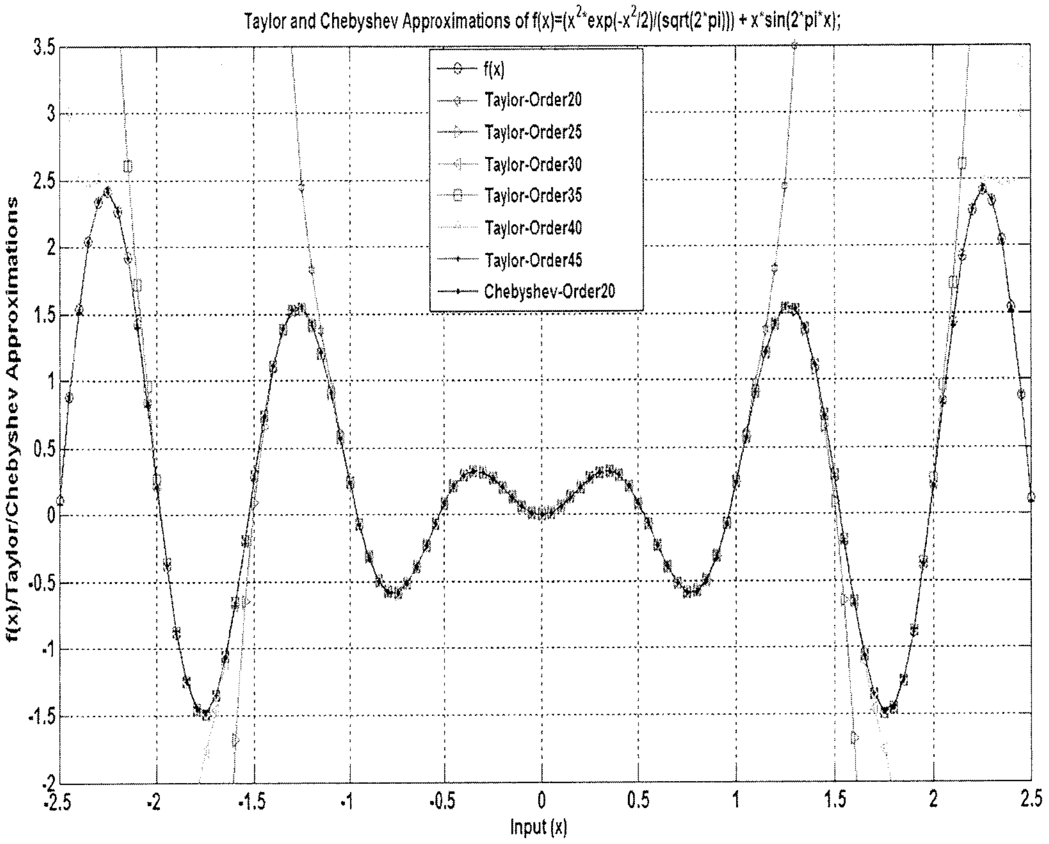


Figure 3-2: Comparison of convergence properties of Taylor and Chebyshev polynomials

To further demonstrate approximation properties of Taylor polynomial in ss, consider an example illustrated in figure 3-3 [94].

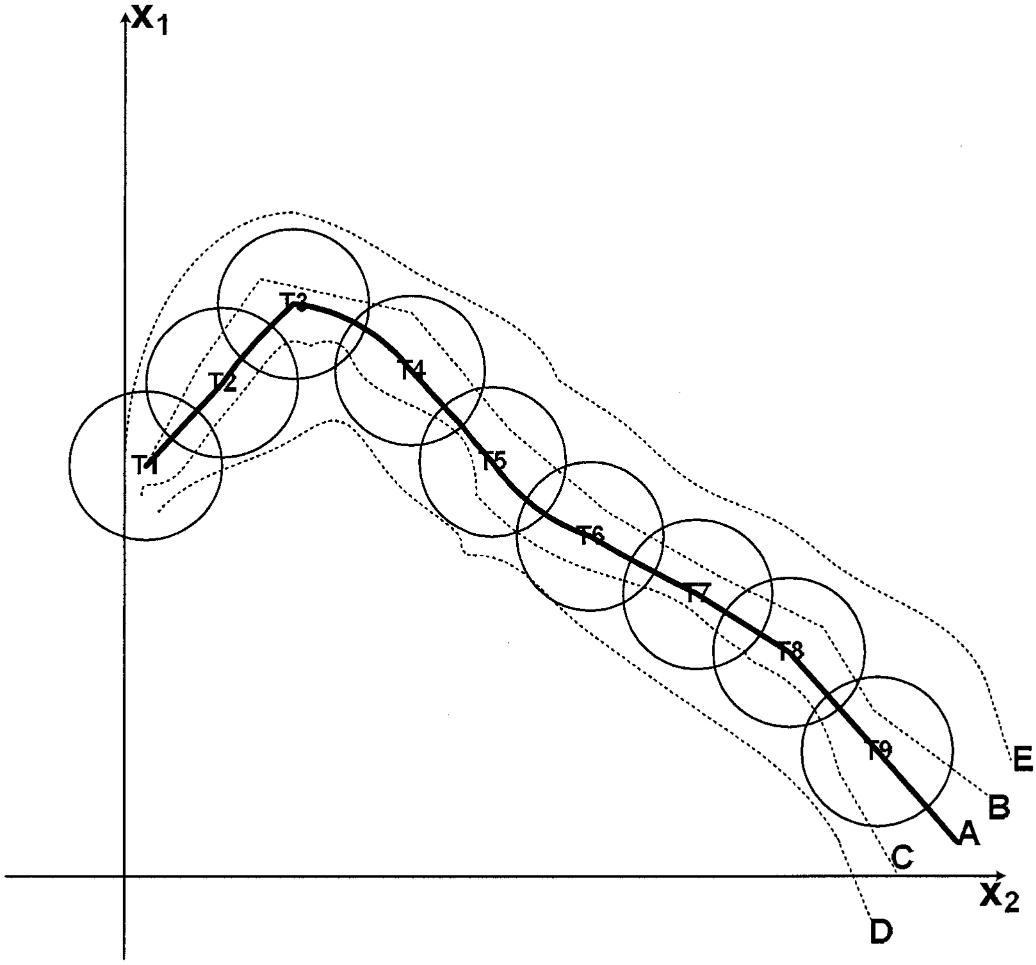


Figure 3-3: Generation of the linearized models along a trajectory of nonlinear system in two-dimension ss

Nine linearization points (T_1 to T_9) are generated along a training trajectory *A* using Taylor polynomials. Other trajectories (*B, C, D*, and *E*) are generated using inputs different than training input for trajectory *A*. Trajectories *B* and *C* are generated for inputs that are closer in magnitude to the input for training trajectory *A* which means the difference in inputs magnitudes for trajectory *A* and trajectories *B* and *C* is small. Whereas trajectories *D* and *E* are generated using the inputs that are far or larger in magnitude than training input for trajectory *A*. The balls represent the region of convergence of Taylor polynomials. It is clear from figure 3-3 that as long as input magnitude is closer to training input magnitude, the resulting trajectories will fall into the region of convergence of Taylor polynomial and macromodel can suitably approximate original nonlinear system response (i.e. can predict correct *B* and *C* trajectories in figure 3-3). On the other hand, if input is far or larger in magnitude than the training input magnitude, then trajectories *D* and *E* do not fall in these balls and macromodel will not be able to accurately predict *D* and *E* trajectories. It should be stressed here that the *shape* of inputs that generate trajectory *A* can be totally different from the inputs that generate trajectories *B* and *E* e.g. training trajectory *A* can be generated using a step function whereas *B* and *E* can be generated using square and sawtooth inputs etc. The important point to notice here is that input magnitude *scale* (e.g. amplitude of input voltage) actually affects the macromodel response. The poor approximation property of Taylor polynomial is the major cause of using multiple linearization models in ss.

It is clear from the above discussion that an efficient polynomial is required that has better approximation as well as interpolation properties and can predict correct response for the inputs that are larger/ far from the training input. Furthermore, to avoid model switching, there should be single polynomial in the ss that can approximate response for the complete trajectory.

A number of interpolating polynomials exists in literature, e.g., Chebyshev polynomials, Newton polynomials, Lagrange polynomials, Hermite polynomials etc. It is demonstrated in [92][95] that Chebyshev polynomials have shown good interpolation and approximation properties due to their discrete orthogonality property and minimax approximation of a function. Therefore, in this work we replace Taylor polynomials with Chebyshev polynomials in nonlinear ss model structure. A comparison of Taylor and Chebyshev polynomial approximation properties can be seen from figure 3-2. Single Chebyshev polynomial of order *20* fit the complete curve, which shows that single Chebyshev polynomial can be employed in ss to approximate complete trajectory with single model. Similarly it is also clear that single Chebyshev polynomial is effective for large input magnitudes as well.

3.2. Automated Model Generation Algorithm Using Chebyshev and Newton Interpolating Polynomial (AMG-CNIP)

The AMG-CNIP methodology consists of two steps: model generation and model evaluation. The application of MOR methods at AMG-CNIP macromodels is outside the scope of this dissertation.

3.2.1 Model generation

We use a fusion of CN interpolating polynomial for nonlinear function *f*(*x*) in ss model structure shown in chapter 1, repeated in (3-1).

$$\begin{aligned} E\dot{x} &= f(x(t)) + bu(t), \\ y &= Cx \end{aligned} \tag{3-1}$$

CN polynomial generation is a three step procedure described below.

1. Obtain a set of input data points (call it x_k') for nonlinear function using Chebyshev node formulas.
2. Obtain a set of target points (call it $f(x_k')$) by using Chebyshev nodes calculated above and input to original nonlinear function $f(.)$.
3. Obtain final interpolating polynomial by employing Newton divided difference (NDD) method on data set $(x_k', f(x_k'))$.

Nonlinear function *f(x)* represents current through nonlinear components (e.g. diodes, transistors etc.) in the circuit. To generate CN polynomial original nonlinear function equation is required. Original nonlinear function (e.g. nonlinear diode current equation or nonlinear transistor drain current equation) can be obtained from SPICE-like simulators e.g. [96]. In our case we obtain nonlinear current equation *f(x)* from SystemVision software that run SPICE2G6 simulation engine [97].

The nonlinear current *f*(*x*) can be a function of either single node voltage (when nonlinear component is connected to single node and other end is grounded) or a function of two node voltages (when nonlinear component is connected to two different nodes) in the circuit. This means that we need one dimensional (1D) and two dimensional (2D) mathematical formulations to obtain CN polynomials. All nonlinear components are classified as 1D and 2D according to their node connections and then *single* 1D CN polynomial is calculated for *all* 1D components and *single* 2D CN polynomial is calculated for *all* 2D components. Please note that unlike existing AMGs that store multiple linearization information (multiple Taylor polynomial models) for each nonlinear component in the circuit, we only store two polynomials: one CN polynomial for 1D components and one CN polynomial for 2D components. Thus our methodology is also memory storage efficient.

3.2.1.1. *1D CN Polynomial Formulation*

To obtain 1D CN polynomial three-step procedure shown below can be used.

1. Simulate original circuit and measure the maximum and minimum magnitude range $[a,b]$ (amplitude of current) of nonlinear function $f(x)$ between which it swings. Then use Chebyshev node formula given in equation (3-2) [95].

$$x'_k = \frac{b-a}{2}\cos\frac{2N+1-2k}{2(N+1)}\pi + \frac{a+b}{2} \tag{3-2}$$

where a is the lower limit and b is the upper limit of the range $[a,b]$. $k=0,1,2,\ldots,N$ and N is the number of nodes to generate and also the degree of polynomial.

2. Use Chebyshev nodes x'_k obtained in the last step as inputs to original nonlinear function and calculate corresponding target points $f(x'_k)$. Thus obtain a set of interpolation points $(x'_k, f(x'_k))$.

3. Using a set of interpolation points $(x'_k, f(x'_k))$ obtained above, apply 1D NDD method [95] to obtain polynomial in the form shown in equation (3-3).

$$n_N(x) = a_0 + a_1(x-x'_0) + a_2(x-x'_0)(x-x'_1) + \ldots + a_n(x-x'_0)\ldots(x-x'_N) \tag{3-3}$$

where $a_0, a_1, \ldots, a_N$ are coefficients of the polynomial calculated using NDD and x'_k are the Chebyshev nodes calculated in step 1.

3.2.1.2. *2D CN Polynomial Formulation*

If $f()$ is function of two node voltages, e.g., for variables x and y, then 2D CN polynomial can be obtained using three-step procedure shown below.

1. Simulate original circuit and measure the range of *f()*. Calculate Chebyshev nodes (x'_k, y'_k) for two voltages (*x,y*) using equation (3-4).

$$x'_k = \frac{b-a}{2}\cos\left(\frac{2N+1-2k}{2(N+1)}\pi\right)+\frac{a+b}{2};\quad y'_k = \frac{b-a}{2}\cos\left(\frac{2N+1-2k}{2(N+1)}\pi\right)+\frac{a+b}{2} \qquad (3\text{-}4)$$

2. Calculate target nodes using set of Chebyshev nodes (x'_k, y'_k) as input to nonlinear function *f()* and obtain a set of interpolation points ((x'_k, y'_k),f(x'_k, y'_k)) to obtain CN polynomial.

3. Using a set of interpolation points obtained in the last step; apply 2D NDD [98] method to obtain 2D CN polynomial of the form shown in equation (3-5).

$$n_N(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{n-i} a_{i,j}x'^{i}y'^{j} \qquad (3\text{-}5)$$

3.2.1.3. *Automatic Range Finder (ARF)*

We can apply the heuristic approach to manually set the range [*a,b*] for generating CN polynomial. However, an efficient way is to automatically measure the range [*a,b*] for nonlinear function *f(x)*. To automatically measure the range a simple procedure called automatic range finder (ARF) can be applied.

1. Simulate original circuit with training input and store state vector *x(t)*.
2. Simulate nonlinear current function *f(x(t))* with state vector *x(t)* as its input.
3. Find maximum and minimum value of current *f(x(t))* and apply as [*a,b*] range for both 1D and 2D formulation given in subsections 3.2.1.1 and 3.2.1.2.

Suitable setting of range [*a,b*] is critical to get accurate macromodel response. Incorrect settings can badly affect the macromodel response. The value *a* here represents the minimum current value calculated for all nonlinear component and *b* represent maximum value for all nonlinear components. The generation of CN polynomial based on these extreme values will automatically cover lower value cases for all nonlinear components in the circuit. In this way we do not need to compute different CN polynomial for each nonlinear component and single CN polynomial can be used for linearization of all nonlinear components in ss.

3.2.1.4. *CN Polynomial Representation in SS*

Once CN polynomial model for nonlinear function *f(x)* is generated using three-step procedure described above, CN polynomial can be employed, in the nonlinear ss model structure, as shown in equation (3-6).

$$\begin{aligned} E\frac{d}{dt}x(t) &= n_N(x(t)) + bu(t) \\ y(t) &= c^T x(t) \end{aligned} \tag{3-6}$$

The critical task in model generation process is the tuning of order (N) for CN polynomial because unnecessarily high order can affect the simulation speed. Order can be tuned by simulating model (3-6) with different training inputs and measure the output error (the difference between model output and original circuit output). There are two ways to tune the order. Either we can manually assign best guess to the order and test for different training data to achieve best output. Other way is to automatically tune the model order. A simple procedure called automatic polynomial tuner (APT) can automatically tune the polynomial order based on training inputs.

3.2.1.5. *Automatic Polynomial Tuner (APT)*

Given original circuit output *y(t)* for a set of training inputs and a pre-defined error tolerance γ, following simple algorithm (Table 3-1) can be implemented to tune N.

Table 3-1: Algorithm for Automatic Polynomial Tuner (APT)

Algorithm 1: Algorithm for Automatic Polynomial Tuner (APT)

```
4.  Start with arbitrary order $N$. The initial value is usually set in range of 1 to 3.
5.  Select training input waveform
6.  for each transient time step do
7.      Compute input $u(t)$ for current time $t$.
8.      Calculate macromodel output (call it $y'(t)$) using (3-6)).
9.      Measure absolute error $\varepsilon = |y'(t) - y(t)|$
10.     If $\varepsilon > \gamma$
11.         $N = N + 1$
12.         Apply three-step procedure (sec 3.2.1.1 and 3.2.1.2) with new $N$ to re-calculate CN polynomial for 1D and 2D.
13.     else
14.         If transient time finish
15.             Exit
16.         else
17.             Increment transient time step $t$ and go to step 4.
18.         end if
19.     end if
20. end for
```

Please notice that APT tune order for 1D and 2D CN polynomials at the same time. Once polynomial order (N) is tuned, macromodel coefficients (matrices E, B, C and CN polynomial coefficients $(a_0, a_1, \ldots, a_N)$ for 1D and 2D) are used for model evaluation with evaluation input waveforms.

3.3. Model Evaluation

In model evaluation process the polynomial coefficients calculated after model tuning are obtained from model generation process and the model is simulated with different input waveforms than the training input. During model evaluation the macromodel interpolate between excited and non-excite states that appear due to different evaluation inputs. The macromodel is expected to output the same response for evaluation input as the response generated by original system for the evaluation input. The situation is depicted in figure 3-4.

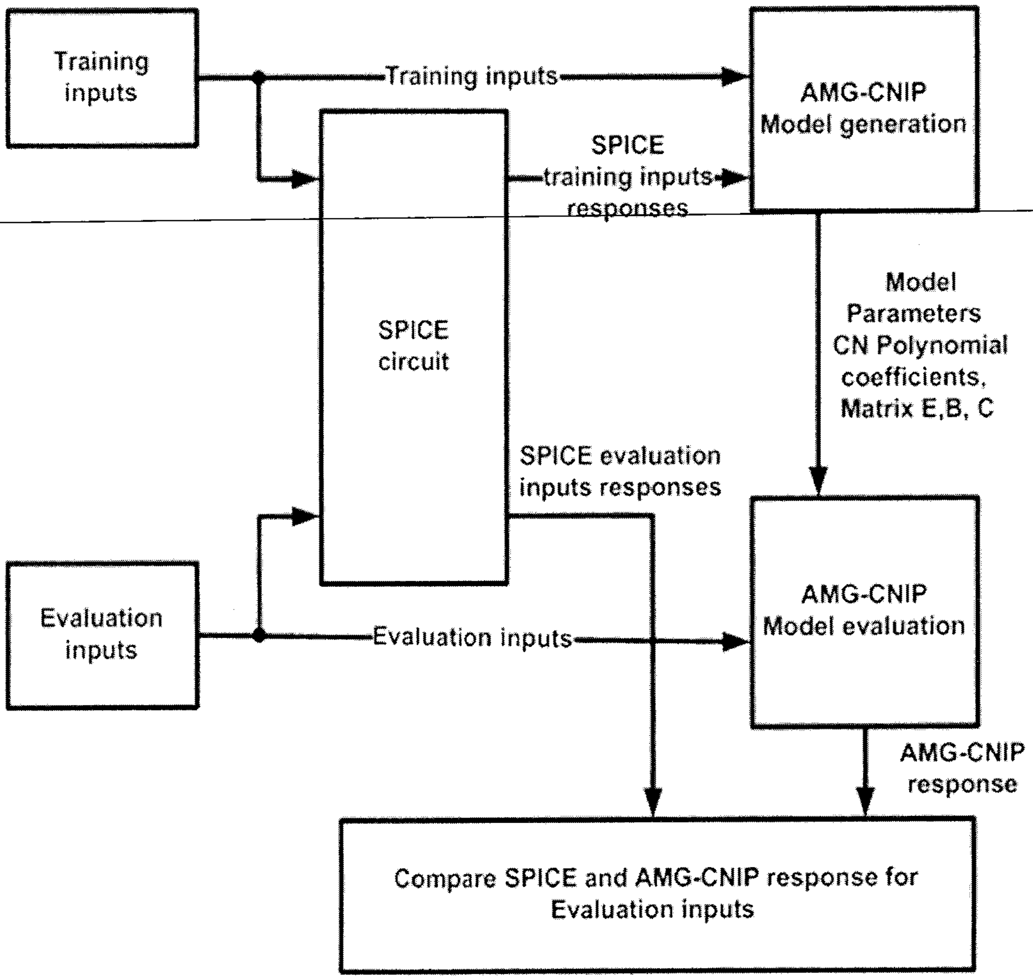

Training inputs
Training inputs
AMG-CNIP Model generation
SPICE training inputs responses
SPICE circuit
Model Parameters CN Polynomial coefficients, Matrix E,B, C
SPICE evaluation inputs responses
Evaluation inputs
Evaluation inputs
AMG-CNIP Model evaluation
AMG-CNIP response
Compare SPICE and AMG-CNIP response for Evaluation inputs

Figure 3-4: AMG-CNIP model generation and evaluation flow

To conclude this section, we provide full algorithm details for macromodel generation and evaluation through AMG-CNIP in the Table 3-2

.Table 3-2: AMG-CNIP algorithm for model generation and simulation

Algorithm 2 AMG-CNIP algorithm for model generation and simulation

```
Model Generation
Formulate circuit KCL equations in ss form (3-1) (Appendix A).
Simulate original circuit with training input and automatically measure range (to use for CN polynomial generation) using ARF (section 3.2.1.3)
for all 1D nonlinear component do
    Apply 1D formulation to obtain single CN polynomial for all single ended nonlinear components (sec 3.2.1.1)
end for
for all 2D nonlinear components do
    Apply 2D formulation to obtain single CN polynomial for all nonlinear components with two node connection (sec 3.2.1.2)
end for
Employ 1D and 2D CN polynomials in ss to obtain AMG-CNIP model (equ. 3-6)
Given training input, simulate AMG-CNIP model (3-6) and tune CN polynomial order ($N$) using algorithm 3 (sec. 3.2.1.5)

Model Evaluation
Given an evaluation input waveform and AMG-CNIP model parameters
for all time steps do
    Interpolate CN polynomial in ss
    Solve ss equations to obtain output $y$
end for
```

Complete MATLAB code for model generation and model evaluation of AMG-CNIP algorithm is given in Appendix B.

3.4. AMG-CNIP Experimentation

To prove that our proposed AMG-CNIP has better approximation and interpolation properties compared to Taylor polynomial macromodel, we carry out an experimentation in which we generate macromodels for AMG-CNIP and white-box AMG that use single CN and Taylor polynomial respectively. A nonlinear transmission line circuit is employed shown in figure 3-5.

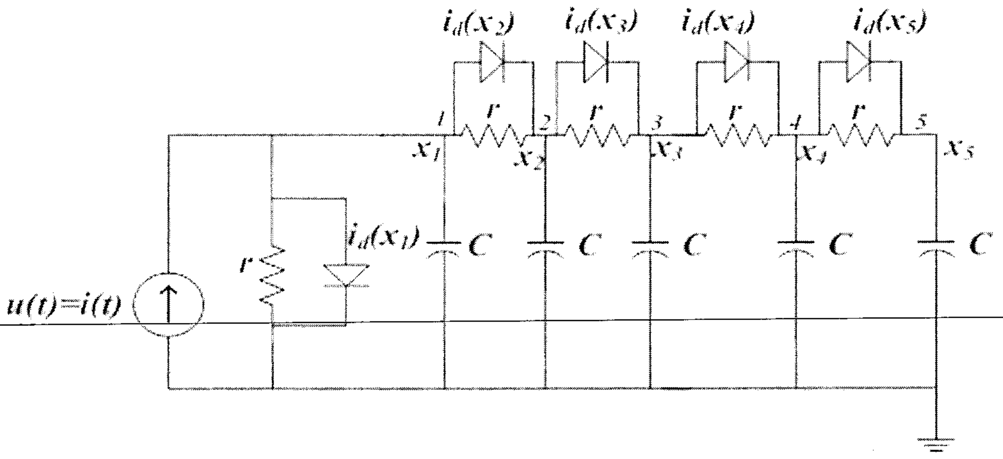


Figure 3-5: Schematic of transmission line circuit (N=5 nodes) with $i_d(x) = \exp(40x) - 1$

It consists of resistors, capacitors and diodes with simplified current model $i_d(x) = \exp(40x) - 1$. All resistors and capacitors have unit value, i.e., $R=C=1$. The single input is the current source entering node 1, $u(t) = i(t)$; the single output is the voltage at node N, $y(t) = x_N(t)$.

First consider simple case of 1D polynomial. For 1D case, there is only one diode, resistor and capacitor in the circuit. As one end of diode is connected to ground, diode current $f(x)$ is a function of only one node voltage across it. We generate training input of the form shown in equation (3-7).

$$u(t) = \sum_{i=1}^{5} Amp_i . \sin(2\pi . freq_i . t + \theta_i) \qquad (3\text{-}7)$$

For this experimentation the training input is generated by adding 5 waveforms. The value of amplitude (*Amp*) varies randomly between ±2.5A. Similarly random values of frequency (*freq*) are chosen from 1Hz to 100 Hz; and random values of phase (θ_i)

are used. Using these parameter values training input waveform generated is shown in figure 3-6.

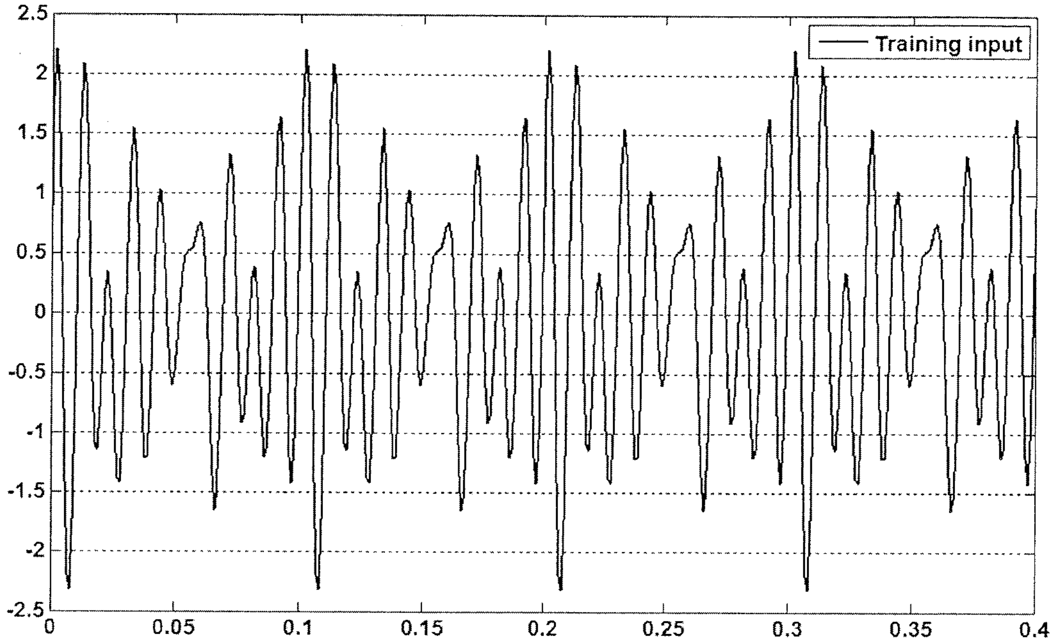


Figure 3-6: Training input waveform

Macromodel is simulated with the training input (figure 3-3) and ARF find the range [-0.8, 1.5797]. APT tunes CN polynomial order to 6 with error tolerance γ=1.0e-3. For comparison we also generate 1D Taylor polynomial macromodel of order 6.

To evaluate AMG-CNIP and Taylor polynomial macromodel, an evaluation input signal is generated by combining two waveform of the form (3-7) whose parameters are randomly selected different than training input parameters such that input current swings between ±2.0A. A transient simulation of 400msec is run and simulation result is shown in figure 3-7.

Simulation result shows that both marcomodel show good accuracy, however AMG-CNIP response is indistinguishable from original system response. Both macromodels are tested for maximum input amplitude ±2.0A that is ±0.5A less than maximum training input amplitude of ±2.5A. Therefore, both are able to successfully interpolate between excited and non-excited states in the ss.

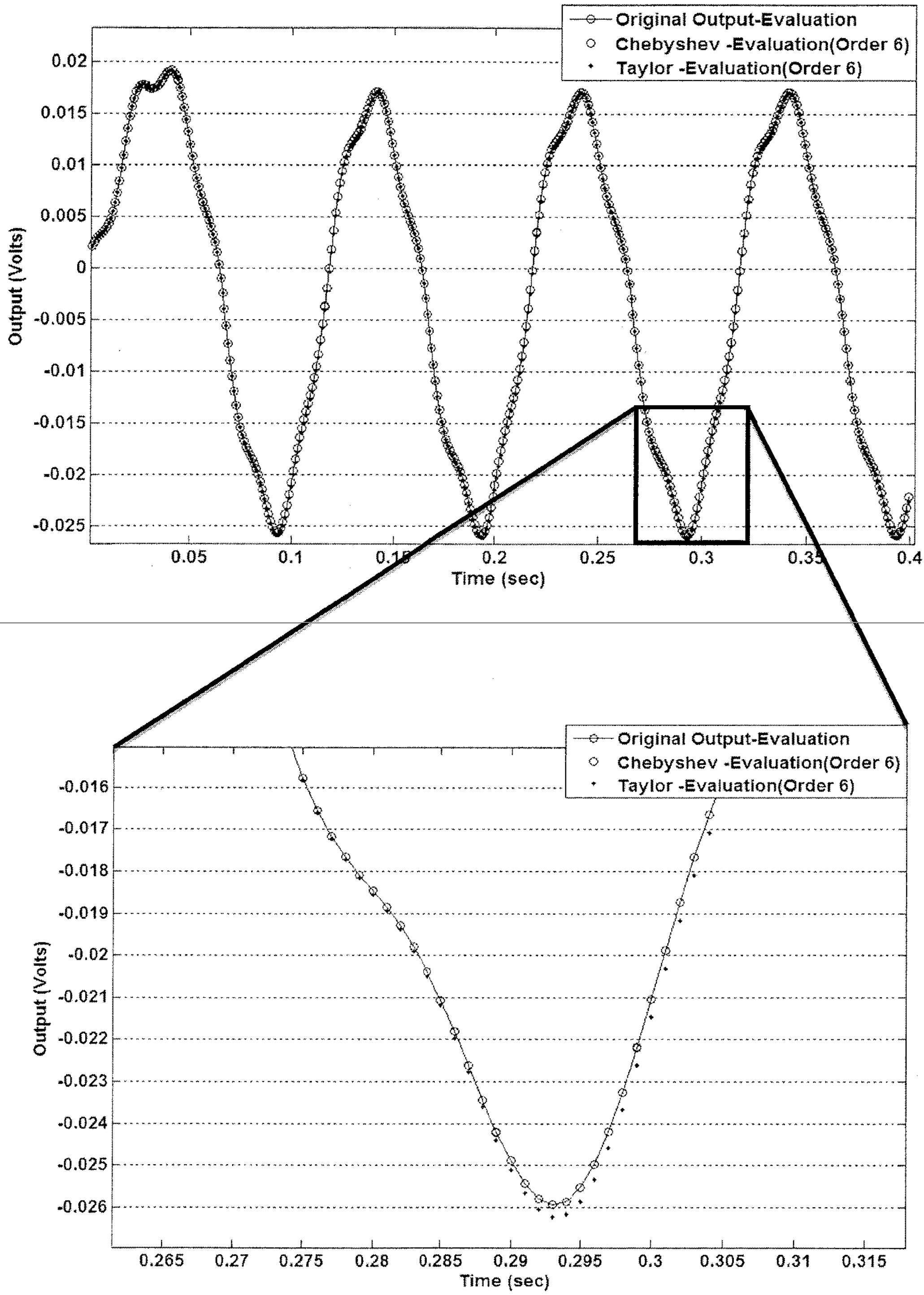


Figure 3-7: Transient response of 1D polynomial macromodels for input amplitude ±2.0A and random *freq* and θ

Another simulation is run with input amplitude ±5.0A with random *freq* and θ. Now evaluation input amplitude (±5.0A) exceeds maximum training input maximum amplitude (±2.5A) by an amount ±2.5A. The simulation result is shown in figure 3-8.

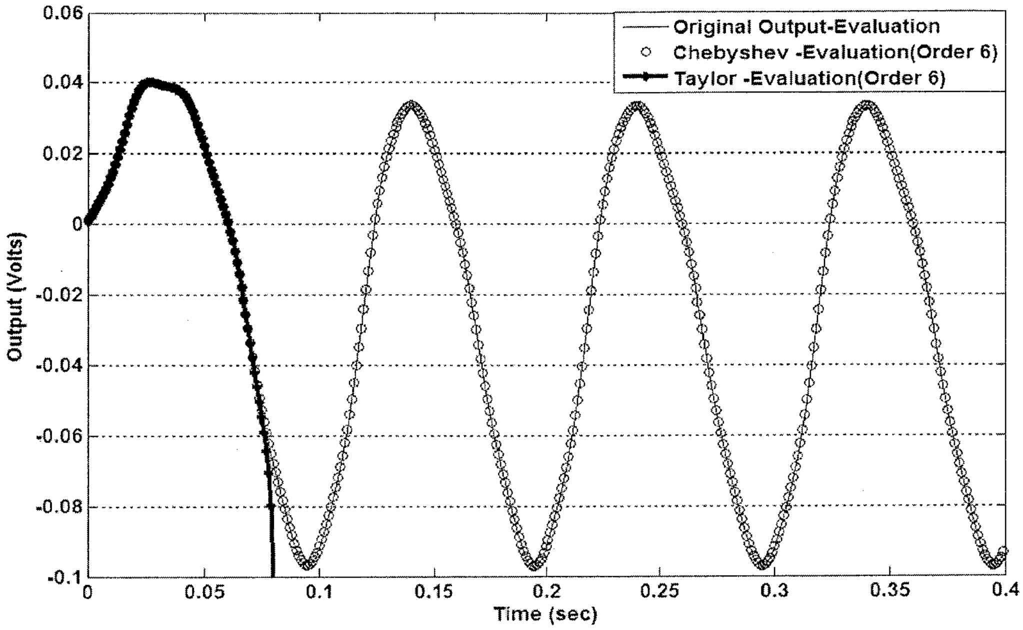


Figure 3-8: Transient response of 1D polynomial macromodels for input amplitude ±5.0A that is ±2.5A greater than training input amplitude

Since amplitude of evaluation input is larger than the training input maximum amplitude, the large input pushes the ss to the regions that falls outside the region of convergence of Taylor polynomial (as discussed in sec. 3.1). Therefore, Taylor polynomial macromodel response (dotted line in figure 3-8) diverges from original system response (solid line). On the other hand, AMG-CNIP response (circled line) shows good accuracy due to its better approximation and interpolation properties.

In next experimentation, there are five nodes in the circuit shown in figure 3-5 and resulting ss equation is shown in equation (3-6). Here 1D formulation is used only for nonlinear current function $i_d(x_1)$ for first diode whose one end is connected to ground. 2D formulation is used for remaining diodes current functions $[i_d(x_2)\ i_d(x_3)\ i_d(x_4)\ i_d(x_5)]$. APT tunes the CN polynomial order to 10. CN polynomial order is high in this case as there are more diodes in the circuit and they significantly contribute to nonlinear behavior of circuit. For comparison, we also generate a Taylor polynomial macromodel with order 10. Similar training input of form (3-7) is generated as for 1D experimentation. The output is voltage is taken at node 5 i.e. N=5.

A transient simulation of 3 seconds is run with input ±2.0A and simulation result is shown in figure 3-9.

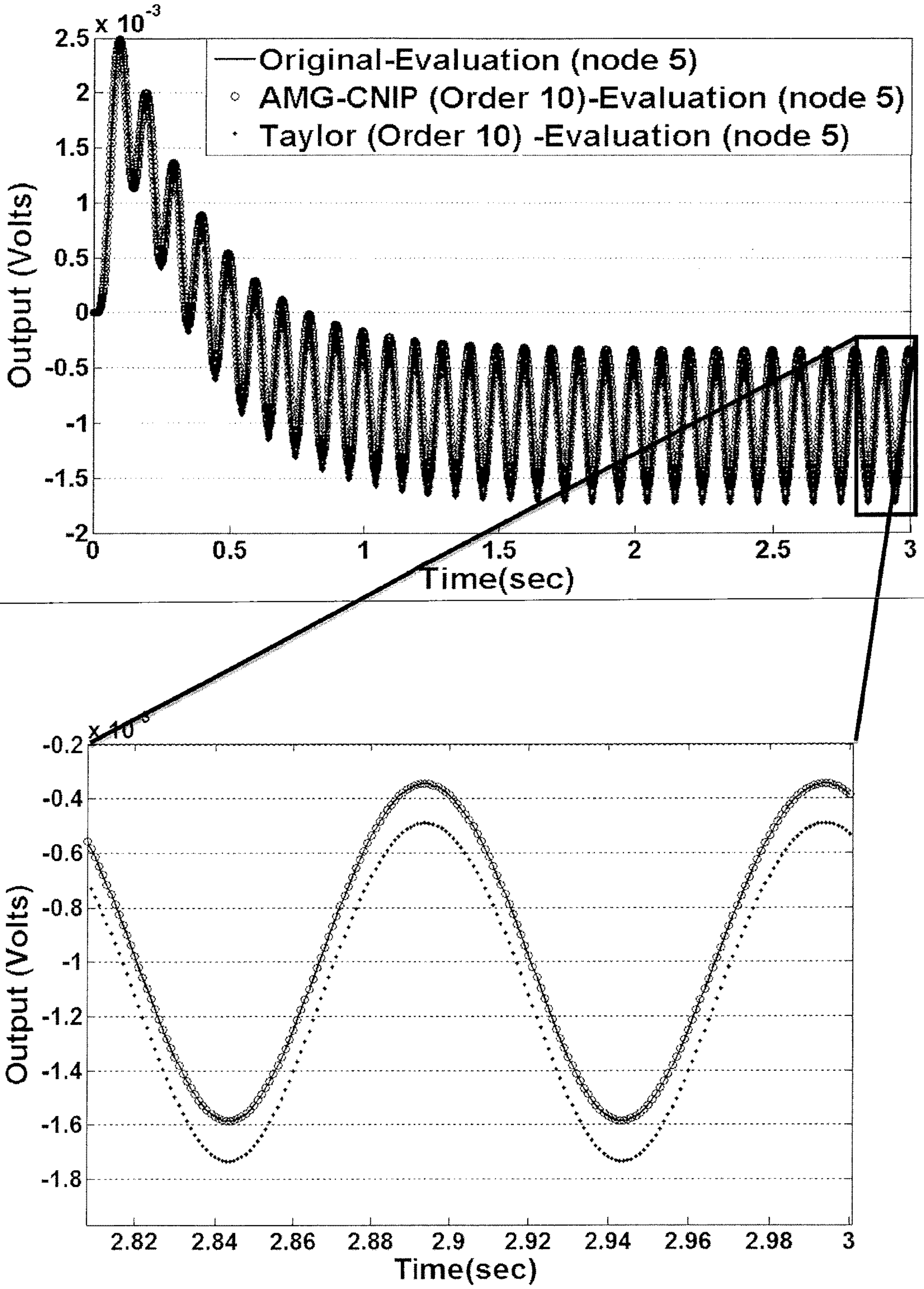


Figure 3-9: Transient response of 2D polynomial macromodels for input amplitude ±2.0A and random *freq* and θ

Zoomed portion of last 100ms shows that accuracy of AMG-CNIP macromodel is better than Taylor polynomial macromodel. Another transient simulation is run for 1.5

seconds with input amplitude ±5.0A. Simulation result is shown in figure 3-10.

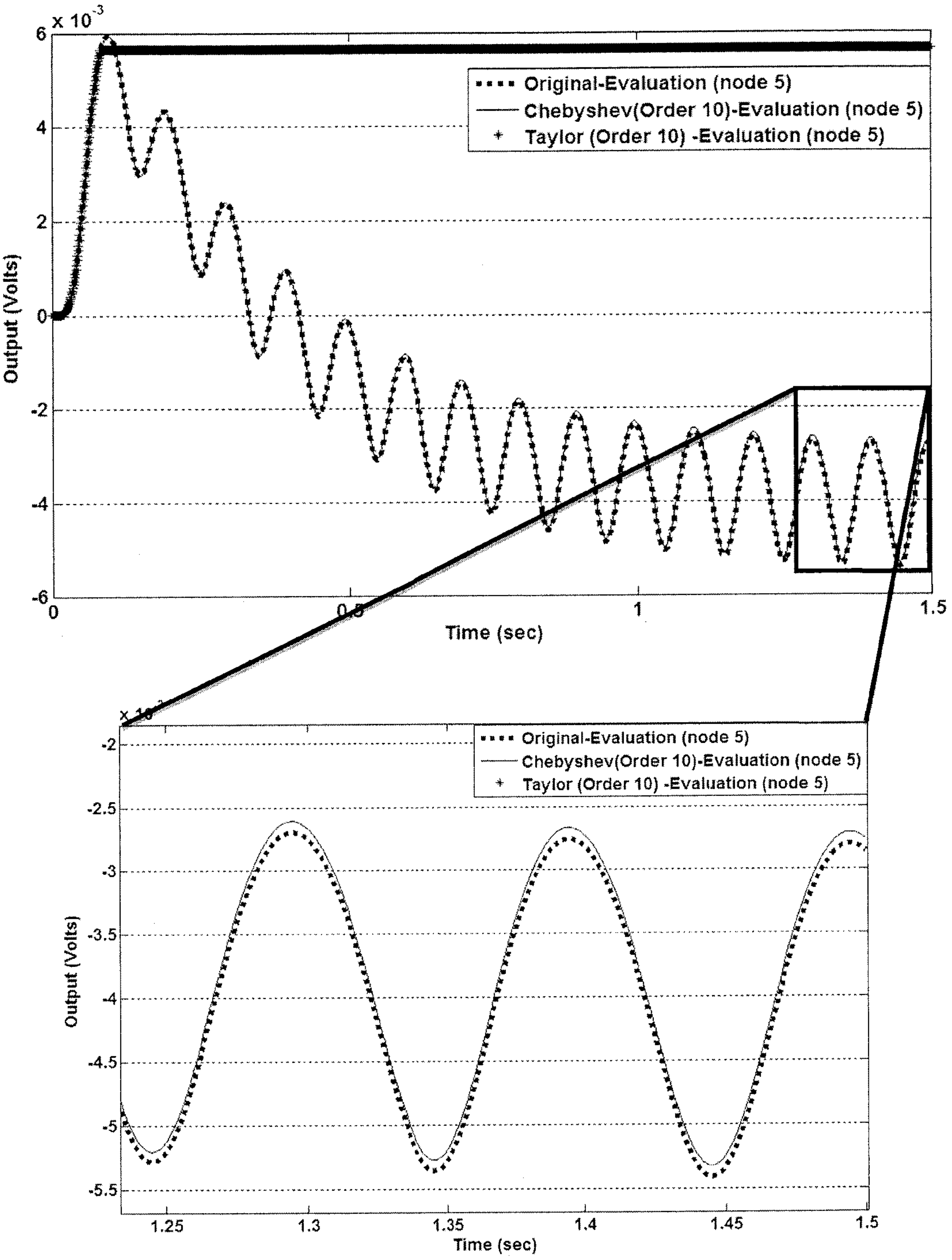


Figure 3-10: Transient response of 2D polynomial macromodels for input amplitude ±5.0A that is ±2.5A greater than training input amplitude

Again Taylor macromodel response diverges from original solution and become straight line (dotted line), whereas AMG-CNIP response follows the original output due to its good interpolation properties. The accuracy of AMG-CNIP macromodel in this case can be easily improved by increasing CN polynomial order.

Another important conclusion is that from the situation of large amplitude evaluation input that macromodel with CN polynomial proves to be stable than Taylor polynomial macromodel. Authors in [37][65][39] do not guarantee the stability of macromodels generated using Taylor polynomials.

3.5. Conclusion

In this chapter we present an automated model generation algorithm called AMG-CNIP that employs a fusion of Chebyshev and Newton interpolating polynomial for linearization of nonlinear components in analog circuits. It employs single polynomial for complete training input as compared to existing Taylor polynomial macromodel that consumes multiple linearized models. Main purpose of this chapter is to show that CN polynomial has advantages than Taylor polynomial in terms of approximation and interpolation properties. The *single* linearized model can be used to model nonlinear analog circuits as compared to multiple linearized models. Simulation results show better accuracy than Taylor polynomial macromodels for single model assumption.

CHAPTER 4

HIGH LEVEL MODELING AND HIGH LEVEL FAULT MODELING USING AMG-CNIP MACROMODELS

4.1. Introduction

In this chapter we perform HLM and HLFM of analog circuits using AMG-CNIP algorithm proposed in chapter 3. Section 4.2 details a routine called Automatic Model Conversion (AMC) that converts an AMG-CNIP MATLB model into a VHDL-AMS behavioral model. In section 4.3 complete details of AMG-CNIP algorithm are provided to perform HLM and HLFM. In section 4.4 HLM is performed at system level based on AMG-CNIP fault-free macromodels using benchmark nonlinear transmission line circuit. Similarly, in section 4.5 HLFM is performed at system level using AMG-CNIP faulty macromodels using nonlinear transmission line circuit. Finally a conclusion is drawn in section 4.6.

4.2. Automatic Model Conversion (AMC)

To practically utilize AMG-CNIP macromodel in electronic domain and simulate at system level (integrated with SPICE circuit blocks) one needs to translate AMG-CNIP mathematical macromodel into an electrical circuit. Verilog-AMS and VHDL-AMS are general choices of language to model mathematical models as electrical circuits.

In this work we use VHDL-AMS language for translating AMG-CNIP MATLAB macromodel into electronic domain behavioral model to simulate at system level. VHDL-AMS is an extension of standard VHDL language used for modeling of digital circuits and systems. AMS (analog and mixed signal) extension in language is provided to model analog and mixed signal systems. To simulate a hybrid of VHDL-

AMS model and SPICE circuit blocks, we use system level simulator from Mentor Graphics called SystemVision [97].

Macromodel generated by AMG-CNIP is in the form of differential equations that contain CN polynomial coefficients (1D and 2D), ss matrices A, B and C. AMC routine loads CN polynomial coefficients (and ss matrices) from MATLAB and code differential equation using VHDL-AMS built-in attribute *'integ'* [99]. Attribute *'integ'* apply numerical integration on differential equation and the desired result (e.g. voltage) is output to the terminal. In this dissertation we use nonlinear transmission line circuit (figure 3-2 in sec. 3.3) to perform HLM and HLFM. An example of VHDL-AMS model for fault-free nonlinear transmission line circuit is shown in figure 4-1.

```
library IEEE;
use ieee.math_real.all;
use IEEE.electrical_systems.all;
entity nltxline_Cheb is
  port (
  terminal u_t,v1_hlm,v2_hlm,v3_hlm,v4_hlm,v5_hlm,g : electrical);
end entity nltxline_Cheb;
 Ideal Architecture
architecture ideal of nltxline_Cheb is
 -- Declare Branch Quantities
  quantity vi across i_in     through u_t     to g;
  quantity v1 across i_out1   through v1_hlm  to g;
  quantity v2 across i_out2   through v2_hlm  to g;
  quantity v3 across i_out3   through v3_hlm  to g;
  quantity v4 across i_out4   through v4_hlm  to g;
  quantity v5 across i_out5   through v5_hlm  to g;
  quantity temp1: real;
  quantity temp2: real;
  quantity temp3: real;
  quantity temp4: real;
  quantity temp5: real;
  --CN polynomial parameters
  constant a_1d_1 : real :=5.571831423275393e-09;
  constant a_1d_2 : real :=3.353644694695958e-10;
  constant a_1d_3 : real :=7.273126596624195e-16;
 begin
temp1  == (-((a_1d_1*v1**2) + (a_1d_2*v1) + a_1d_3) -  (a_1d_1*(v1-v2)**2) + ((a_1d_2*(v1-v2)) + a_1d_3) - (2.0*v1)+v2 + i_in);
v1     == temp1'integ;
temp2  == (( (a_1d_1*(v1-v2)**2) + (a_1d_2*(v1-v2)) + a_1d_3) - ((a_1d_1*(v2-v3)**2) + (a_1d_2*(v2-v3)) + a_1d_3)+ v1- (2.0*v2) + v3);
v2     == temp2'integ;
temp3  == (((a_1d_1*(v2-v3)**2) + (a_1d_2*(v2-v3)) + a_1d_3) - ((a_1d_1*(v3-v4)**2) + (a_1d_2*(v3-v4)) + a_1d_3)+ v2- (2.0*v3) + v4);
v3     == temp3'integ;
temp4  == (((a_1d_1*(v3-v4)**2) + (a_1d_2*(v3-v4)) + a_1d_3) - ((a_1d_1*(v4-v5)**2) + (a_1d_2*(v4-v5)) + a_1d_3)+ v3- (2.0*v4) + v5);
v4     == temp4'integ;
temp5  == (((a_1d_1*(v4-v5)**2) + (a_1d_2*(v4-v5)) + a_1d_3) + v4 - v5);
v5     == temp5'integ;
end architecture ideal;
```

Figure 4-1: An example of VHDL-AMS behavioral model for nonlinear transmission line circuit

Please notice that same CN polynomial coefficients (ald_1 to ald_3) are used for each nonlinear component (each diode current) in all differential equations as compared to existing white box AMG approaches that use different Taylor polynomial coefficients for each nonlinear component.

MATLAB code for AMC routine is given in Appendix B.

4.3. AMG-CNIP Algorithm for HLM and HLFM

A complete picture of performing HLM and HLFM using AMG-CNIP is shown in figure 4-2.

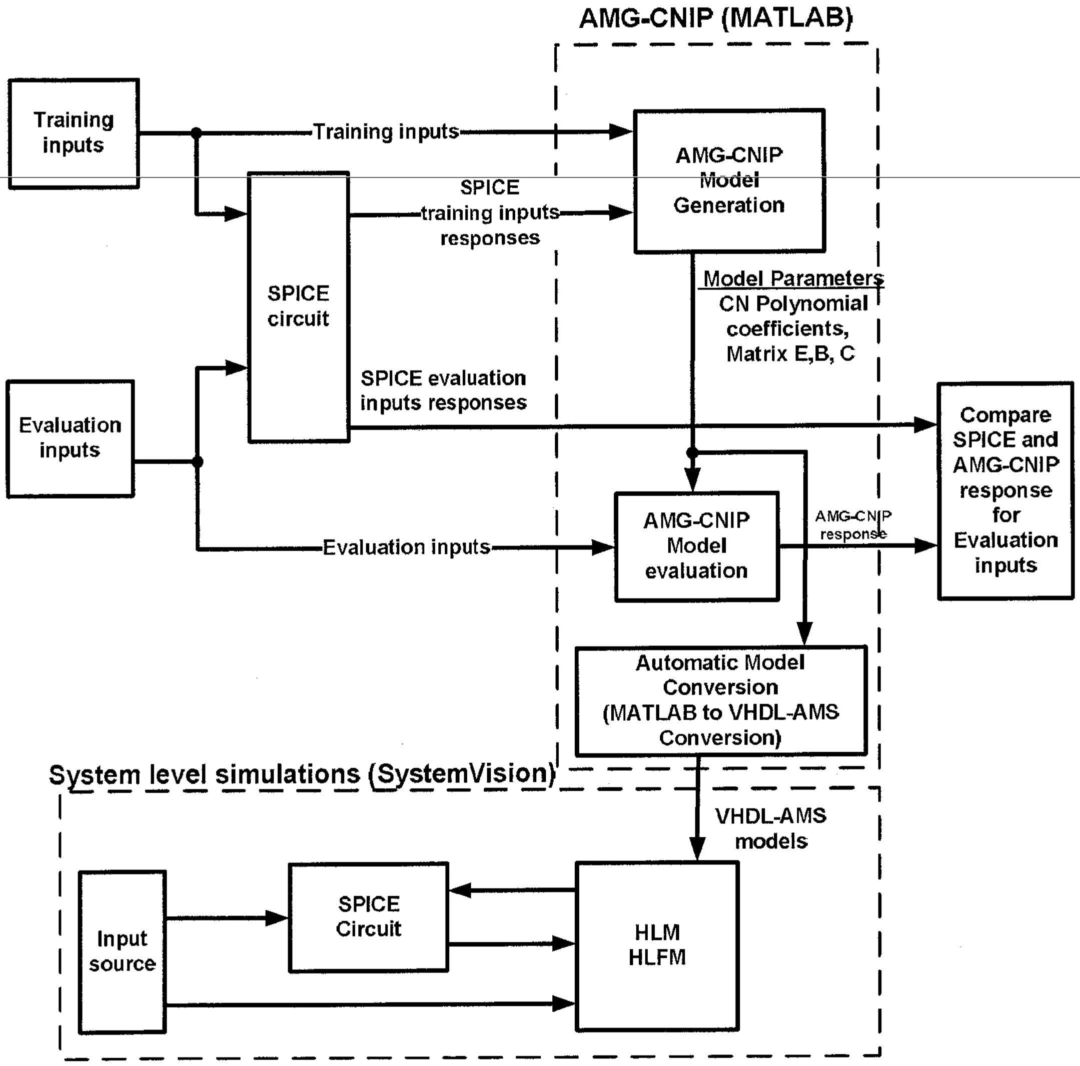

AMG-CNIP (MATLAB)
Training inputs
Training inputs
SPICE training inputs responses
AMG-CNIP Model Generation
SPICE circuit
Model Parameters
CN Polynomial coefficients, Matrix E,B, C
Evaluation inputs
SPICE evaluation inputs responses
Compare SPICE and AMG-CNIP response for Evaluation inputs
Evaluation inputs
AMG-CNIP Model evaluation
AMG-CNIP response
Automatic Model Conversion (MATLAB to VHDL-AMS Conversion)
System level simulations (SystemVision)
VHDL-AMS models
Input source
SPICE Circuit
HLM HLFM

Figure 4-2: Complete picture of AMG-CNIP model generation and performing HLM and HLFM

AMG-CNIP algorithm for performing HLM and HLFM using AMG-CNIP macromodels is given in Table 4-1.

Table 4-1: AMG-CNIP algorithm for model generation, HLM and HLFM

Algorithm 3 AMG-CNIP algorithm for model generation, HLM and HLFM

```
Model Generation
  Formulate circuit KCL equations in ss form (3-1) (Appendix A)
  Simulate original circuit with training input and automatically measure range (to use for CN polynomial generation) using ARF (sec. 3.2.1.3)
  for all 1D nonlinear component do
      Apply 1D formulation to obtain single CN polynomial for all single ended nonlinear components (sec 3.2.1.1)
  end for
  for all 2D nonlinear components do
      Apply 2D formulation to obtain single CN polynomial for all nonlinear components with two node connection (sec 3.2.1.2)
  end for
  Employ 1D and 2D CN polynomials in ss to obtain AMG-CNIP model (equ. 3-6)
  Given training input, simulate AMG-CNIP model (3-6) and tune CN polynomial order ($N$) using algorithm 3 (sec. 3.2.1.5)

Model Evaluation
  Given an evaluation input waveform and AMG-CNIP model parameters
  for all time steps do
      Interpolate CN polynomial in ss
      Solve ss equations to obtain output $y$
  end for

HLM and HLFM using AMG-CNIP
  Given AMG-CNIP parameters for fault-free and faulty circuit; use AMC to generate VHDL-AMS behavioral models (sec. 4.2) i.e. HLM and HLFM.
  Simulate HLM and HLFM at system level using SystemVision and compare with SPICE circuit simulations (sec. 4.4).
```

4.4. HLM Using AMG-CNIP Macromodels

To perform HLM, two benchmark circuits are used. Nonlinear transmission line circuit (NLTx) shown in figure 4-3 is same circuit as shown in figure 3-5 but use nonlinear diode current function *f(x)* obtained from SPICE2G6 [97], instead of simplified diode current model described in section 3.4.

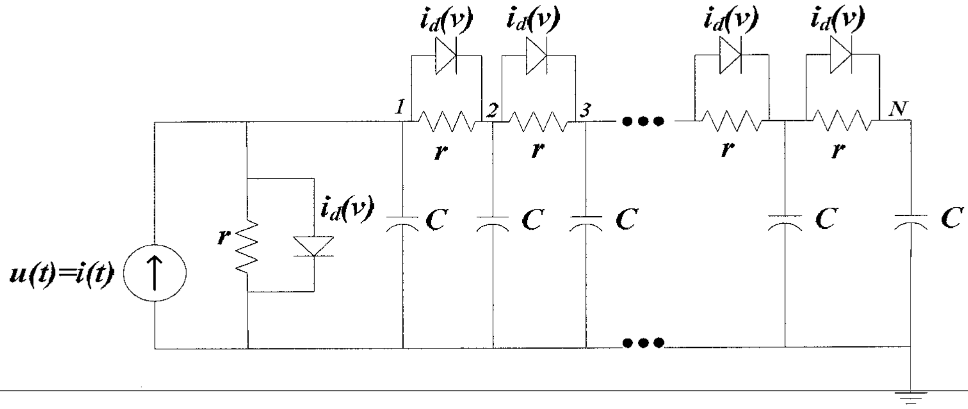


Figure 4-3: Nonlinear transmission line circuit wit nonlinear current function *(fx)* from SPICE2G6 in SystemVision

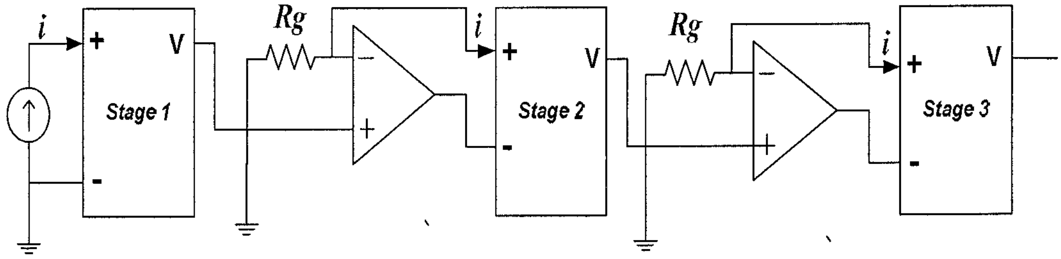


Figure 4-4: 3-stage cascaded nonlinear transmission line circuit

The circuit in figure 4-4 is a 3-stage cascaded NLTx circuit where each stage consists of NLTx line circuit displayed in figure 4-3.

In this work we also implement an AMG from literature called Multiple Model Generation using Delta Operator (MMGSD) [100] for comparison of AMGs performance for multiple model versus single model assumption. Both AMGs (AMG-CNIP and MMGSD) are initially evaluated in MATLAB for accuracy and speed up comparison using NLTx line circuit in figure 4-4 and then HLM is performed.

For macromodel generation a similar training input (given in equation 3-7, sec 3.3) is used to generate MMGSD and AMG-CNIP macromodels. MMGSD use 3 models to achieve reasonable accuracy at output. APT module tunes polynomial order *N* to 3. ARF find range for nonlinear function *f(x)* to be [-0.003, 0.008]. A transient simulation is performed for 10ms with evaluation input generated by combining sinusoidal of random amplitudes, frequencies and phases. Simulation result is shown in figure 4-5.

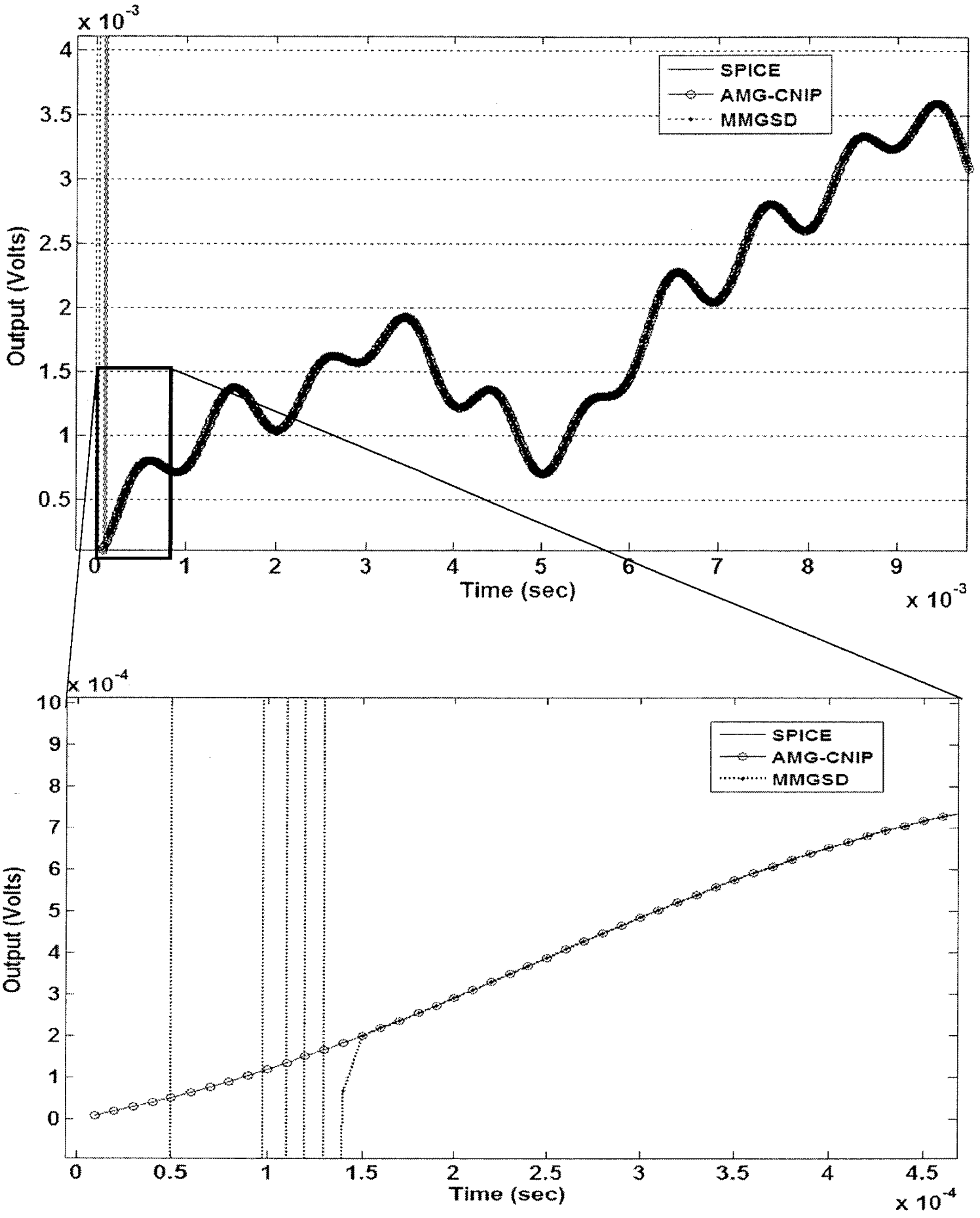


Figure 4-5: MMGSD and AMG-CNIP response for fault-free NLTx line circuit

Figure 4-5 shows that MMGSD macromodel consume approximately 150us to start predicting the correct response. During this time, response tends towards infinity (output voltage is ≈ $7\text{x}10^{24}$ volts). Hence computational overhead involved here for MMGSD is to attain stable state due to model switching until best model is selected.

Moreover, it is clear from figure 4-5 that AMG-CNIP response is settled from the start of transient simulation. The reason for settling time comparison is that macromodel response for first cycle of evaluation input waveform is important, since same response is repeated for remaining waveform cycles. Therefore, an AMG that leads in start predicting correct response during first input waveform cycle is considered to be computationally efficient and fast. As clear from above experimentation, MMGSD takes 150us to stabilize and predict correct response whereas AMG-CNIP response is stable from beginning.

By observing from 150us, the response for remaining time period is shown in figure 4-6.

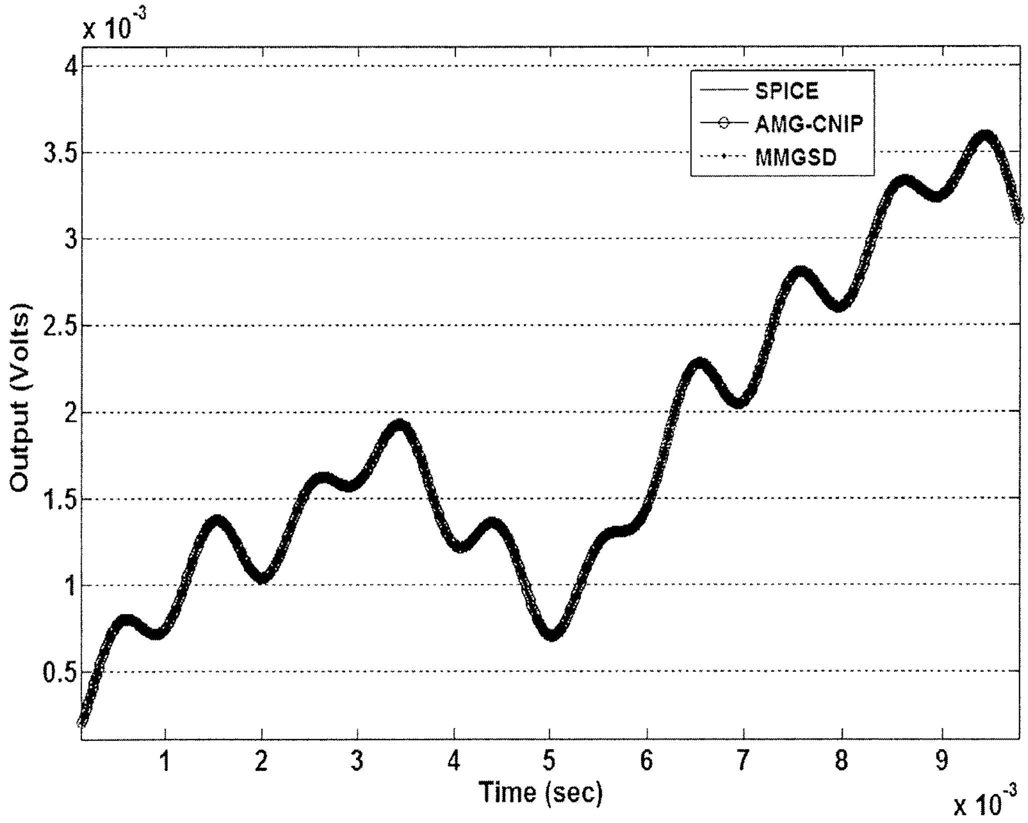


Figure 4-6: MMGSD and AMG-CNIP response for fault-free NLTx circuit (ignoring initial 150us response)

It is clear that waveforms of MMGSD and AMG-CNIP are indistinguishable from original SPICE circuit response. For fair speed comparison we perform 30 simulation runs for both AMGs and on average MMGSD consume approximately 155us to settle the response. AMG-CNIP response is settled from beginning of simulation in all simulation runs. This clearly shows that AMG-CNIP is faster to predict correct response as compared to MMGSD.

To perform HLM, we convert AMG-CNIP and MMGSD macromodels into VHDL-AMS behavioral models using AMC routine in section 4.2. Please notice that for MMGSD a separate AMC routine is developed to generate VHDL-AMS behavioral model as MMGSD model structure is different from AMG-CNIP model structure. HLMs are tested in different configurations (Table 4-2) using circuits shown in figure 4-3 and figure 4-4.

Table 4-2: Configurations for AMG-CNIP HLM Testing

Configuration No	**Settings**
Config 1	NLTx line
Config 2 (3-stage Cascaded)	SPICE – HLM – SPICE
Config 3 (3-stage Cascaded)	HLM – SPICE – HLM
Config 4 (3-stage Cascaded)	SPICE – SPICE – HLM
Config 5 (3-stage Cascaded)	HLM – HLM – HLM
Config 6 (3-stage Cascaded)	HLM – SPICE – SPICE

In Config1, NLTx line interface with input current source only. In remaining configurations, HLM interface with standard SPICE circuit modules to perform system level simulations. There are total of 15 nodes in three stage cascaded nonlinear transmission line circuit and output voltage can be taken at any node while performing system level simulations.

In first experimentation for Config1, a transient simulation of 200 sec is run in SystemVision. Input signal is a current sine wave with 500Hz frequency and ±2.0A amplitude. AMG-CNIP HLM, MMGSD HLM and SPICE circuit are run simultaneously and the simulation result is shown in figure 4-7.

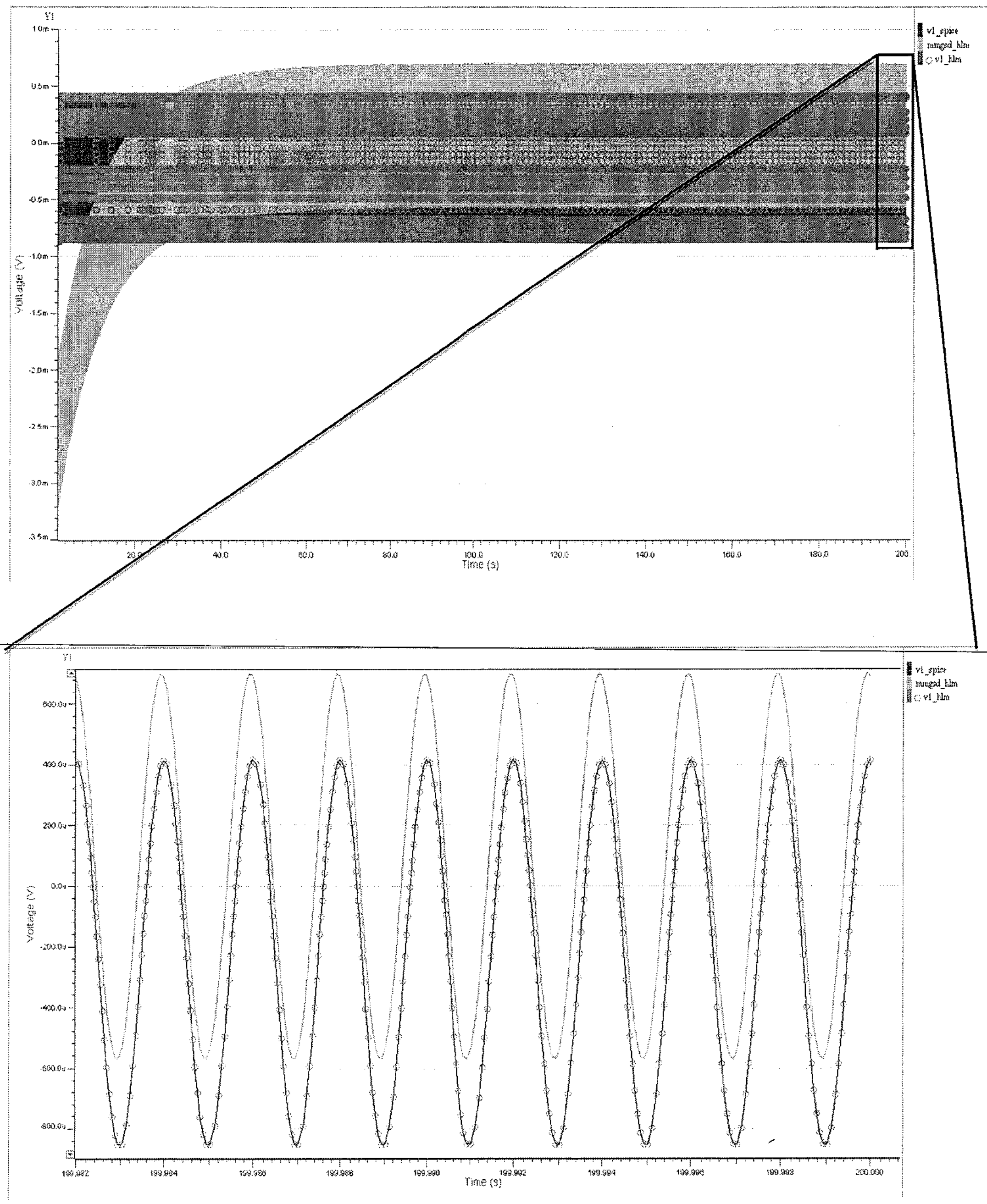


Figure 4-7: Transient response of MMGSD and AMG-CNIP for Config1 circuit

Transient response in figure 4-7 shows that MMGSD is unable to perform high level modeling. It is clear from figure 4-7 that MMGSD HLM response (mmgsd_hlm in figure 4-7) takes approximately 20 seconds to settle. Also, the zoomed portion of last 100ms shows that MMGSD response is significantly different from original SPICE response (v1_spice in figure 4-7) whereas, AMG-CNIP HLM response (v1_hlm in figure 4-7) overlaps accurately with SPICE response from start of simulation. For quantitative comparison, we measure the relative error of two AMG responses w.r.t. SPICE response. To measure relative error in percentage equation 4-1 is used, where *i* is the number of samples obtained from SystemVision simulation.

$$\mathrm{Re}l.error = \frac{\sum_i \left| \dfrac{y\_SPICE_i - y\_AMG_i}{y\_SPICE_i} \right|}{length(y\_SPICE)} \times 100 \qquad (4\text{-}1)$$

Relative error percentage for AMG-CNIP for Config1 circuit is 5.49e-8%, which is almost negligible. Whereas, MMGSD relative error is calculated to be 582% that means MMGSD failed to perform HLM.

We also observe CPU time taken by SystemVision to simulate HLM and SPICE circuits. After the completion of transient simulation SystemVision shows CPU time consume by simulator to solve system of equations for a given circuit.

To accurately note CPU time, we perform 30 simulation runs for each circuit independently and calculate the mean and maximum values. On average SPICE circuit CPU time is 15.65 sec. AMG-CNIP average CPU time is 8 sec and MMGSD average CPU time is 240sec. The reason of high CPU time for MMGSD is that AMG has to switch between three models during simulation and SystemVision is not optimized for discrete switching, thereby model switching becomes a computational overhead.

We also calculate simulation speed up achieved by AMG compared to SPICE circuit. The simulation speedup is calculated using the formula given in equation 4-2

$$Simulation\_Speedup = \frac{Mean(SPICE\_CPU\_time)}{Mean(AMG - CNIP\_CPU\_time)} \qquad (4\text{-}2)$$

In this experimentation AMG-NIP achieve a simulation speedup of 1.96x compared to standard SPICE circuit simulation.

For system level testing, we start with Config2. In Config2, second stage of SPICE circuit is replaced with AMG-CNIP HLM. A transient simulation of 15sec is performed and simulation result is compared with full SPICE circuit simulation result (all stages with SPICE modules). Both Config2 and full SPICE circuit run simultaneously and simulation result is shown in figure 4-8. Similar input signal is given to the circuit as for Config1 and output voltage is taken at node 6 of 3-stage cascaded circuit (which is first output of HLM).

v6_spice is full SPICE circuit response and v6_hlm is Config2 circuit response. It can be seen that AMG-CNIP achieves good accuracy with relative error of 0.72%. Mean CPU time of full SPICE circuit for 30 runs is 28.3sec and mean CPU time of Config2 circuit for 30 runs is 25.26 sec.

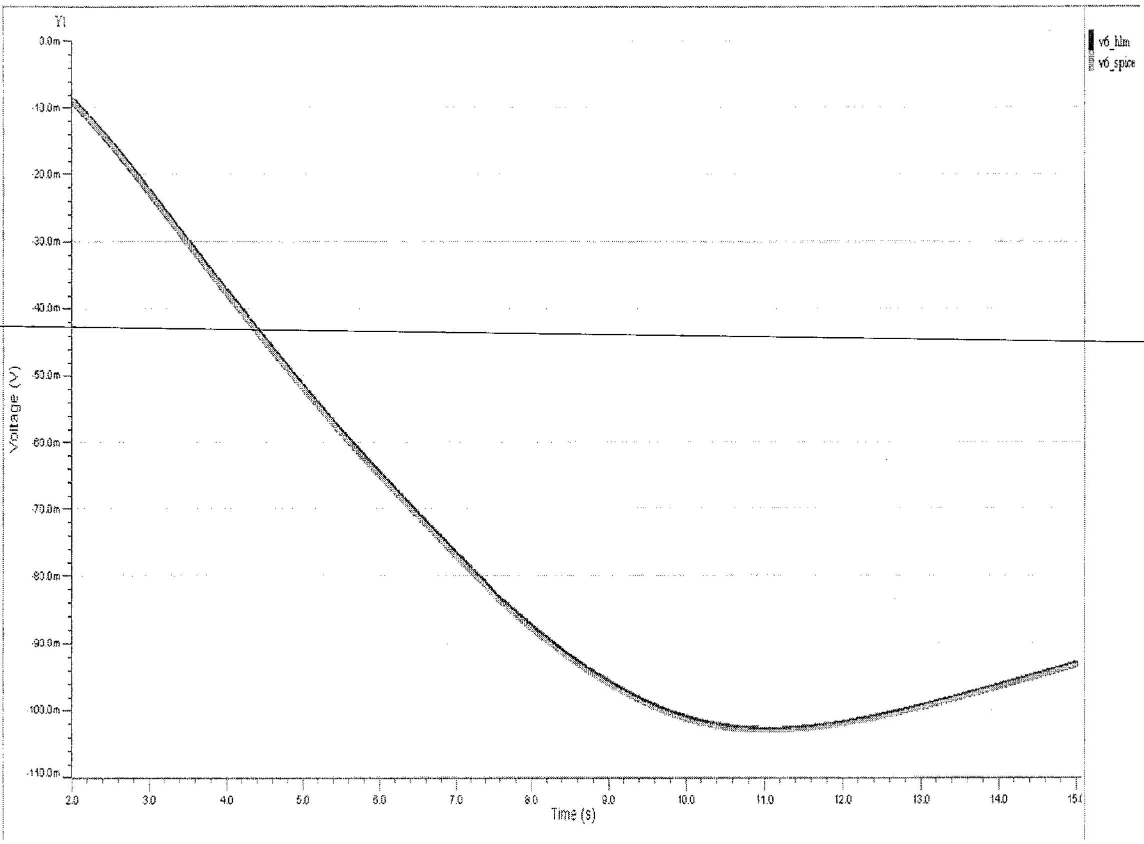


Figure 4-8: Transient response of AMG-CNIP HLM for Config2

Hence a simulation speed up of 1.12x is achieved compared to full SPICE circuit. This shows that AMG-CNIP is able to increase simulation speed of a circuit at system level with reasonable accuracy.

In Config3 circuit we replace first and last stages with AMG-CNIP HLM and perform 15sec transient simulation with same input. Output is taken at node 5 and node 11 shown in figure 4-9 and 4-10.

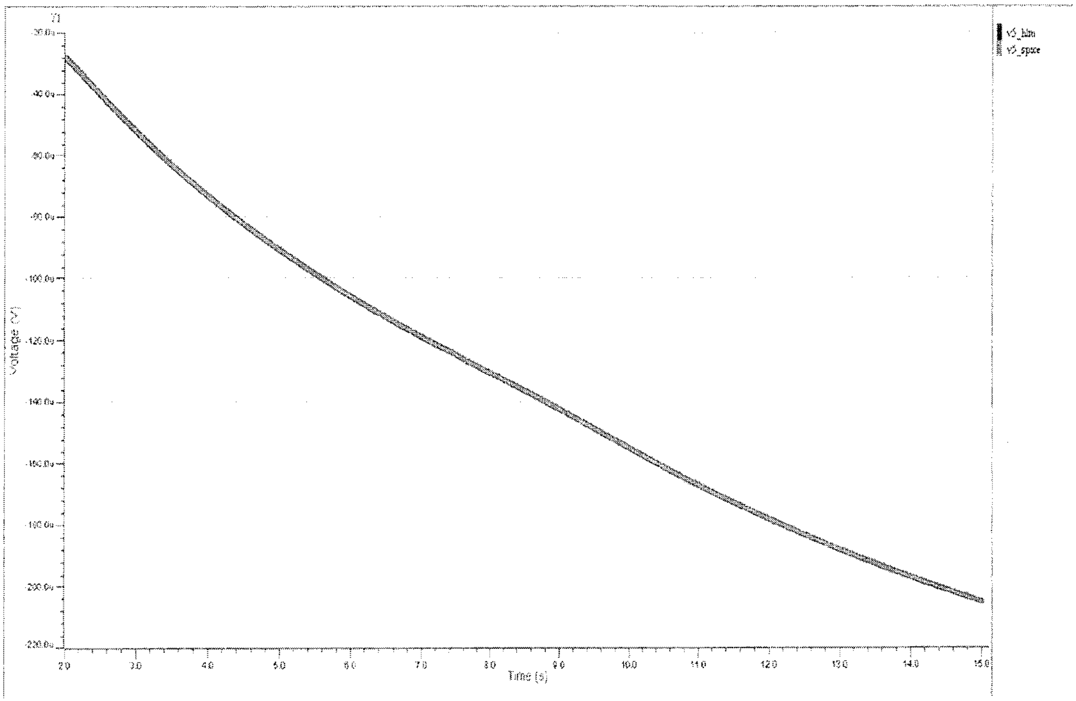


Figure 4-9: Transient response of AMG-CNIP HLM for Config3 (node 5)

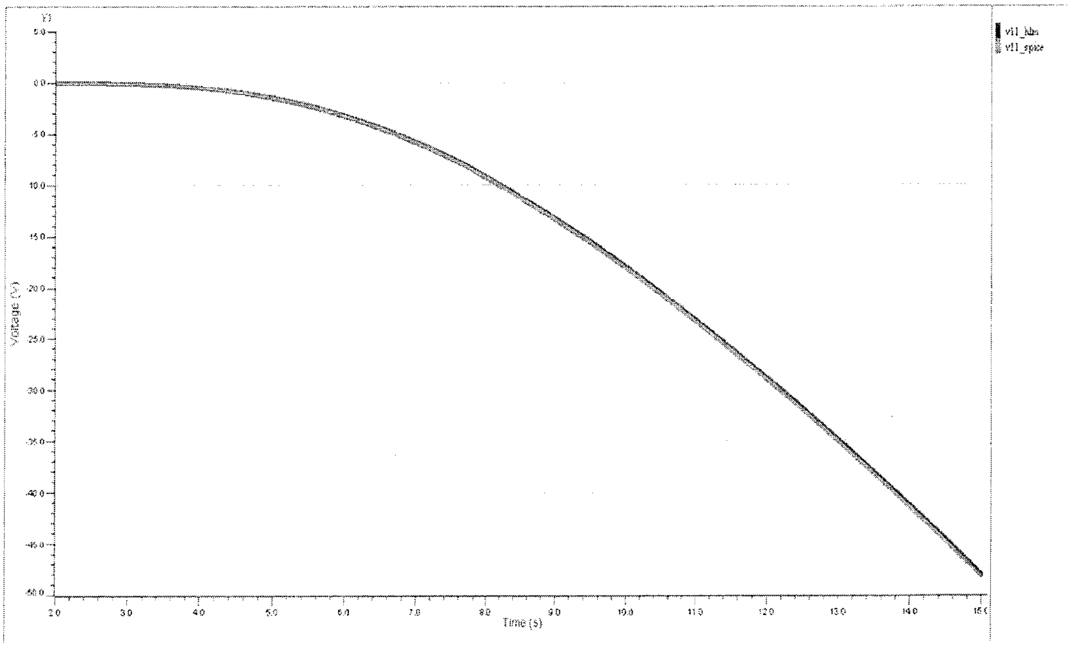


Figure 4-10: Transient response of AMG-CNIP HLM for Config3 (node 11)

Simulation results show that Config3 circuit with two HLMs in circuit is working with good accuracy. The relative error (measure at node 11) is 0.71%. Average CPU time for Config3 is 12.75sec and SPICE full circuit average simulation time is

28.3sec. Hence, a simulation speed up of 2.2x is achieved. This shows that with increasing number of HLMs in the circuit, circuit simulation speed is increasing accordingly and CPU time is decreased.

In Config4 we replace last stage of cascaded nonlinear transmission line circuit with AMG-CNIP HLM. Output voltage is taken at nodes 11, 12 and 13. Simulation result is shown in figure 4-11.

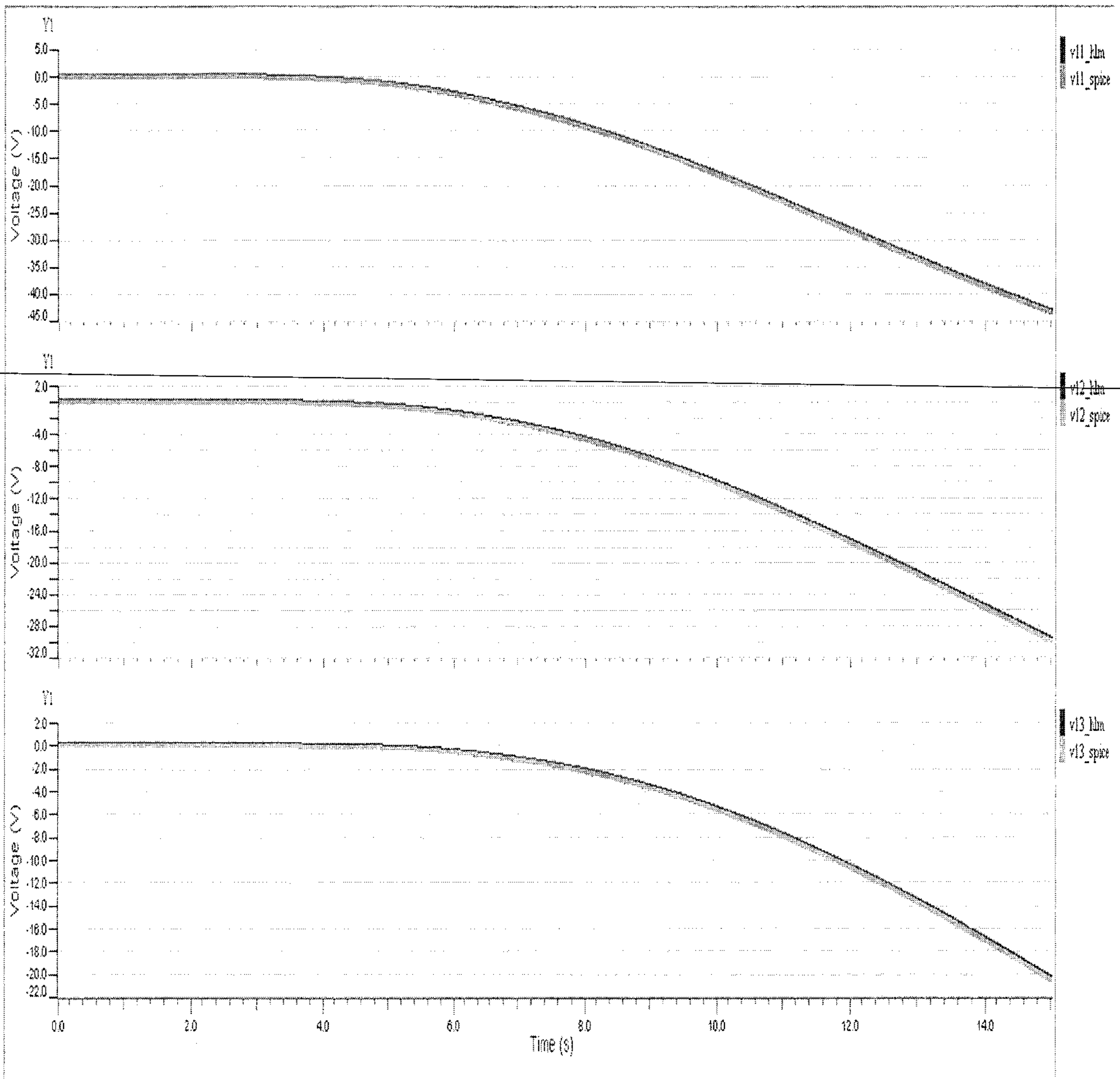


Figure 4-11: Transient response of AMG-CNIP HLM for Config4
(nodes 11, 12 and 13)

Please note that output voltage at node 11 reaches to -45 volts. Similarly voltages at node 12 and 13 reach up to -30 volts and -21 volts respectively and AMG-CNIP HLM is able to predict accurate responses. Above response shows excellent interpolation and approximation properties of CN polynomials because during training, macromodel was trained to predict output responses that swing within the range of millivolts. Whereas, in this configuration output voltages reach to unexpectedly high voltage (-45 volts) and AMG-CNIP HLM can still predict response with good accuracy. Relative error measured at node 11 is 0.0085%, at node 12 is 0.863% and at node 13 is 1.12%. The average CPU time for Config4 circuit is 25.31sec. The simulation speed up achieved is 1.11x.

In Config5, we replace all SPICE modules with AMG-CNIP HLM and compares simulation result with full SPICE circuit. The output is taken at node 5, 10 and 15 to show that each stage is predicting correct response even with little output error in preceding stage. Simulation result is shown in figure 4-12.

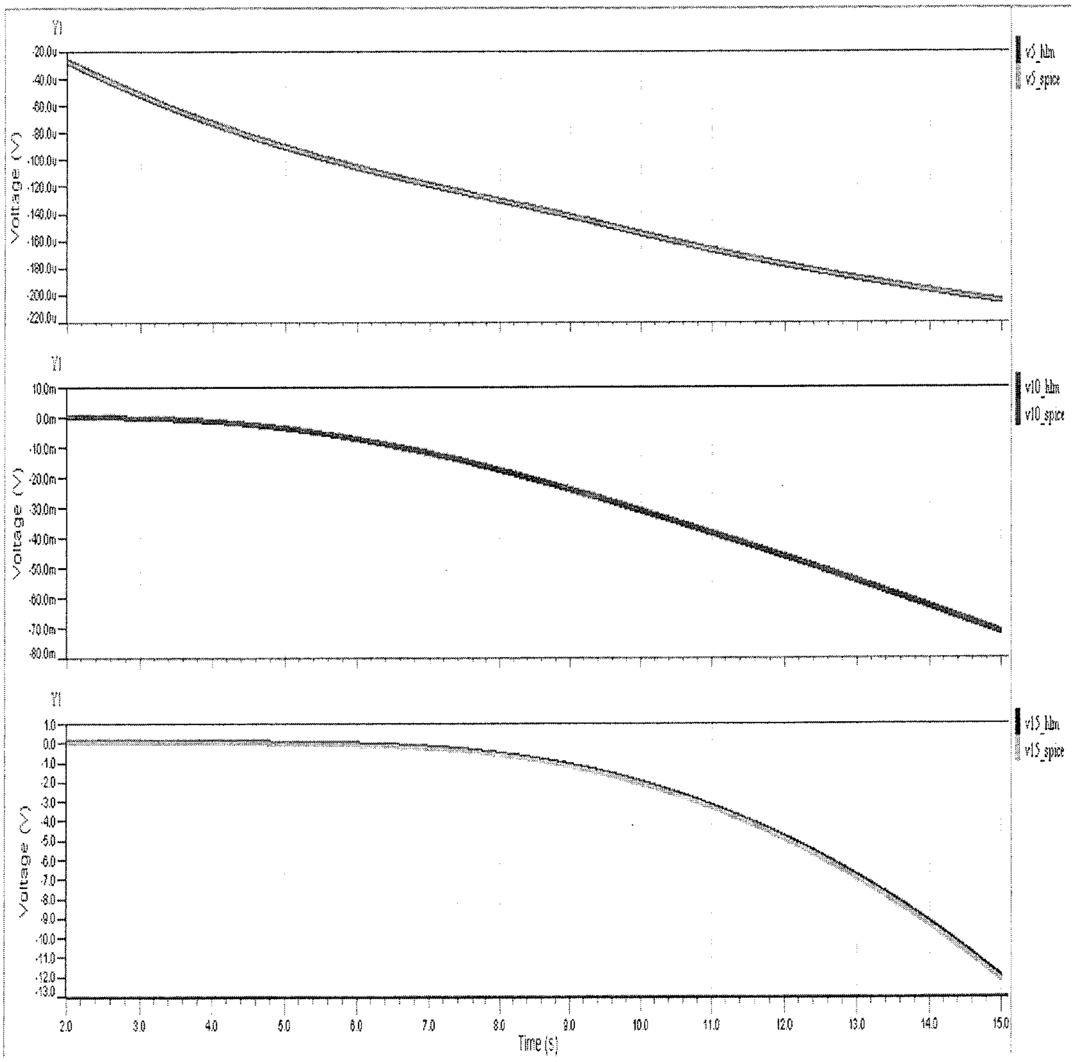


Figure 4-12: Transient response of AMG-CNIP HLM for Config5 (nodes 5, 10 and 15)

It is clear from simulation result AMG-CNIP HLM with all three stages replaced can still show good accuracy. The relative error measured at node 5 is 8.4e-7%, at node 10 is 0.68% and at node 15 is 1.54%. Average CPU time for Config5 circuit is 17 sec and speed up achieved is 2.62x. This is the highest simulation speed up achieved in different HLM configurations and it shows that by using more HLMs in the circuit, higher simulation speed up can be achieved.

In another configuration (Config6) we replace first stage with HLM. Outputs are taken at node 1 and node 5. Simulation result is shown in figure 4-13.

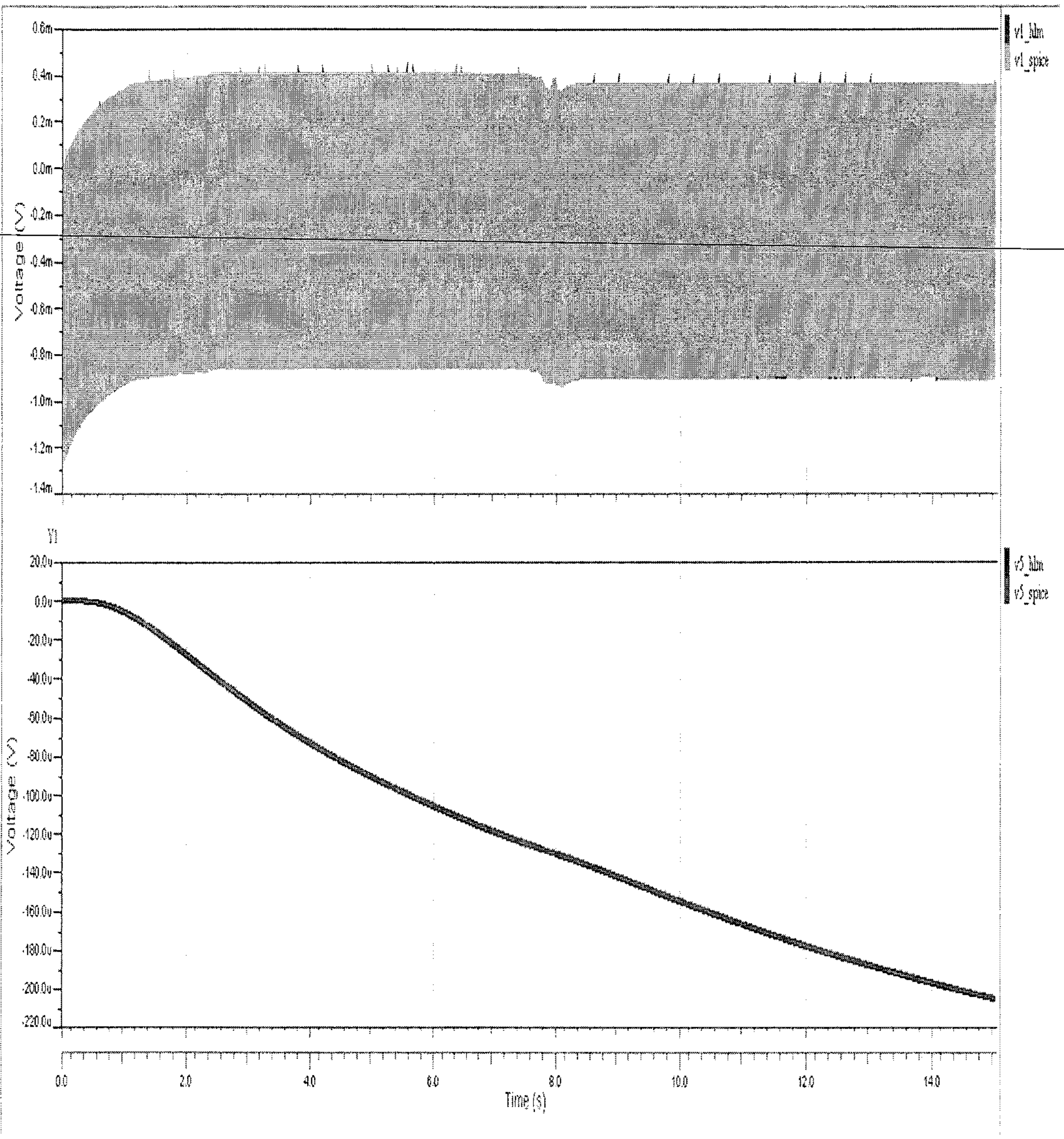


Figure 4-13: Transient response of AMG-CNIP HLM for Config6 (nodes 1and 5)

Mean CPU time for Config6 is 12.68 sec and simulation speedup achieved is 2.23x. Relative error measured at node 1 is 3.7e-7 and 0.05% at node 5.

Complete summary of simulation speed up and accuracy for six different configurations of AMG-CNIP HLM is given in Table 4-3.

Table 4-3: Summary of simulation speed up and accuracy for AMG-CNIP HLM

Configuration Fault Free circuit	SPICE (sec)		AMG-CNIP HLM (sec)		Speed up $\frac{SPICE}{AMG\_CNIP}$	Accuracy Rel. error (%)
	Mean	Max	Mean	Max		
NLTX line	15.65	30.2	8.0	30.3	1.96x	5.49e-8
SPICE_HLM_SPICE	28.3	52.2	25.26	45.8	1.12x	0.72
HLM_SPICE_HLM	28.3	52.2	12.75	20.6	2.2x	0.71
SPICE_SPICE_HLM	28.3	52.2	25.32	47.8	1.11x	1.12
HLM_HLM_HLM	28.3	52.2	10.8	17	2.62x	1.54
HLM_SPICE_SPICE	28.3	52.2	12.68	30	2.23x	3.7e-7

4.5. HLFM Using AMG-CNIP Macromodels

In this section we perform HLFM by introducing a short fault at a diode in NLTx circuit. A short fault is introduced by manually adding a short resistor (r_s) of very small value (0.000001Ω) across a diode. One fault is introduced at a time in the circuit. In this experimentation we perform HLFM by introducing short faults at diodes D1 and D2.

4.5.1 HLFM for Short Fault at Diode D1

First consider a short fault is introduced at diode D1, shown in figure 4-14.

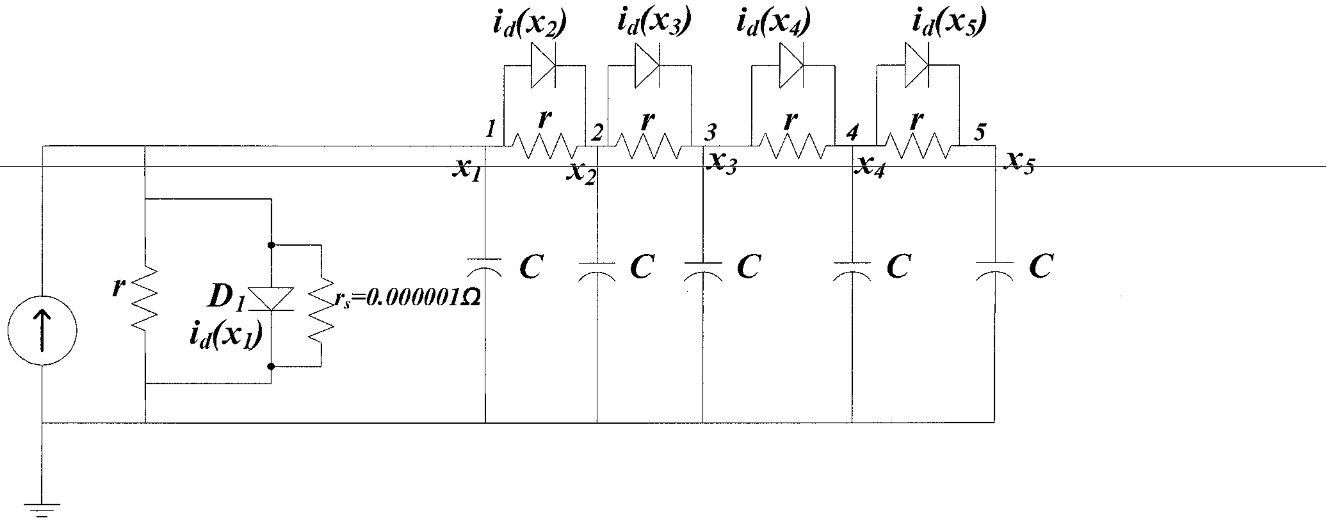


Figure 4-14: NLTx circuit with short fault at D1

HLFM is performed using circuit in figure 4-14 in six different configurations shown in Table 4-4.

Table 4-4: Configurations for AMG-CNIP HLFM testing (short fault at D1)

Configuration No.	**Settings**
Config 1	NLTX line
Config 2 (3-stage Cascaded)	HLFM – SPICE – SPICE
Config 3 (3-stage Cascaded)	SPICE – HLFM – SPICE
Config 4 (3-stage Cascaded)	SPICE – SPICE – HLFM
Config 5 (3-stage Cascaded)	HLM – HLFM – SPICE
Config 6 (3-stage Cascaded)	HLM – HLM – HLFM

The experimentation started by generating AMG-CNIP and MMGSD models in MATLAB. A 3rd order CN polynomial is generated for AMG-CNIP macromodel and ARF find range for nonlinear function *f(x)* to be [-0.0001, 0.0001]. Three models are used for MMGSD macromodel.

MMGSD again show similar behavior and on average consume ~158us to settle the response. On the other hand AMG-CNIP response is settled from the start of simulation. The simulation result is shown in figure 4-15.

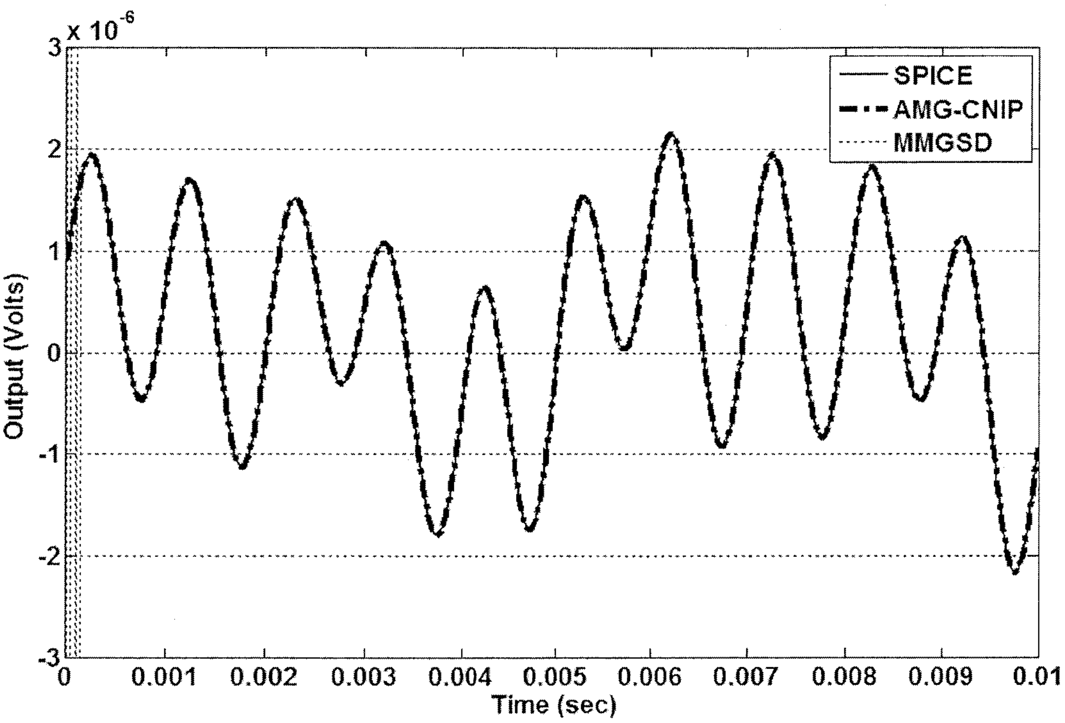
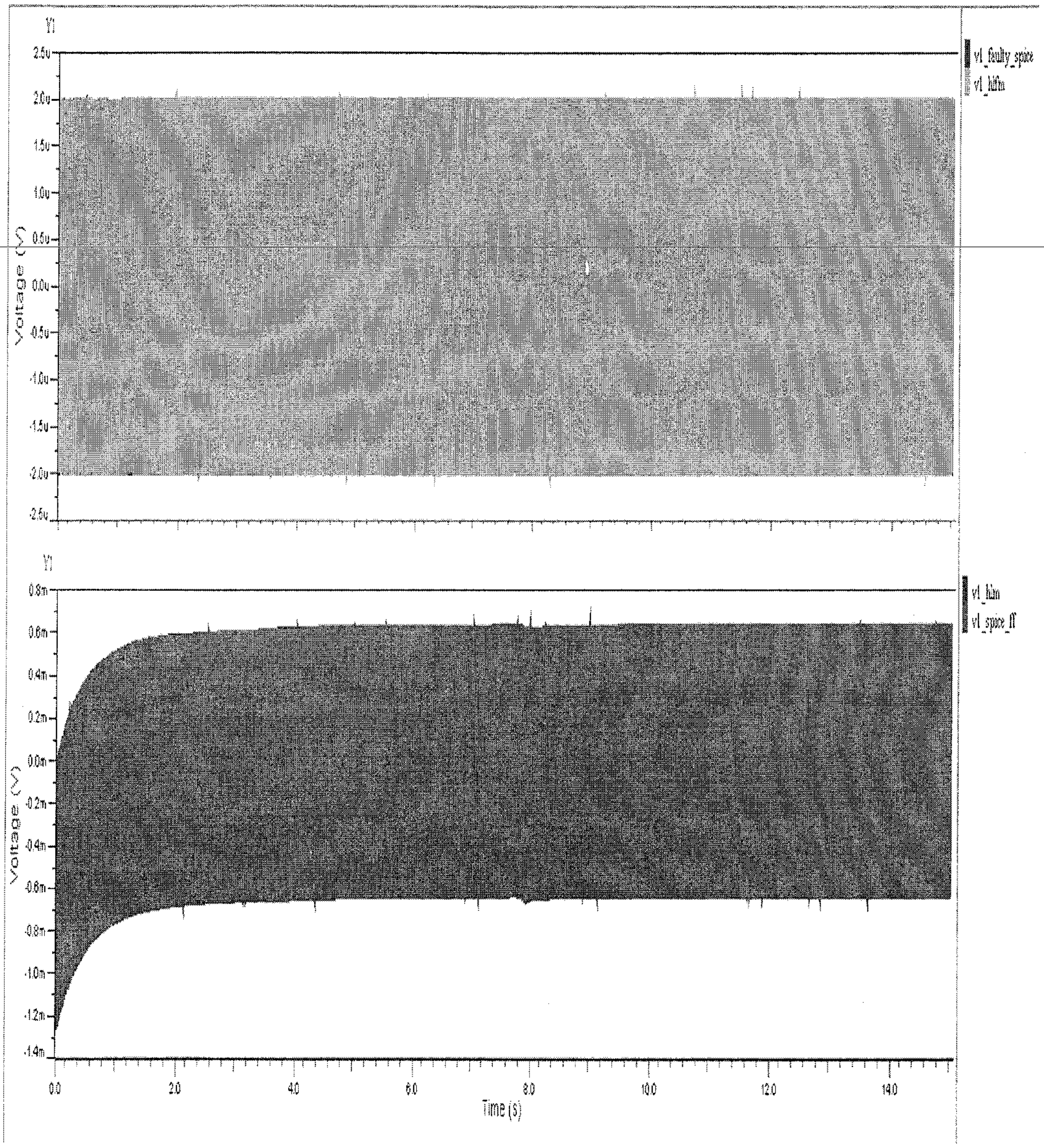


Figure 4-15: MMGSD and AMG-CNIP response for NLTx line faulty circuit

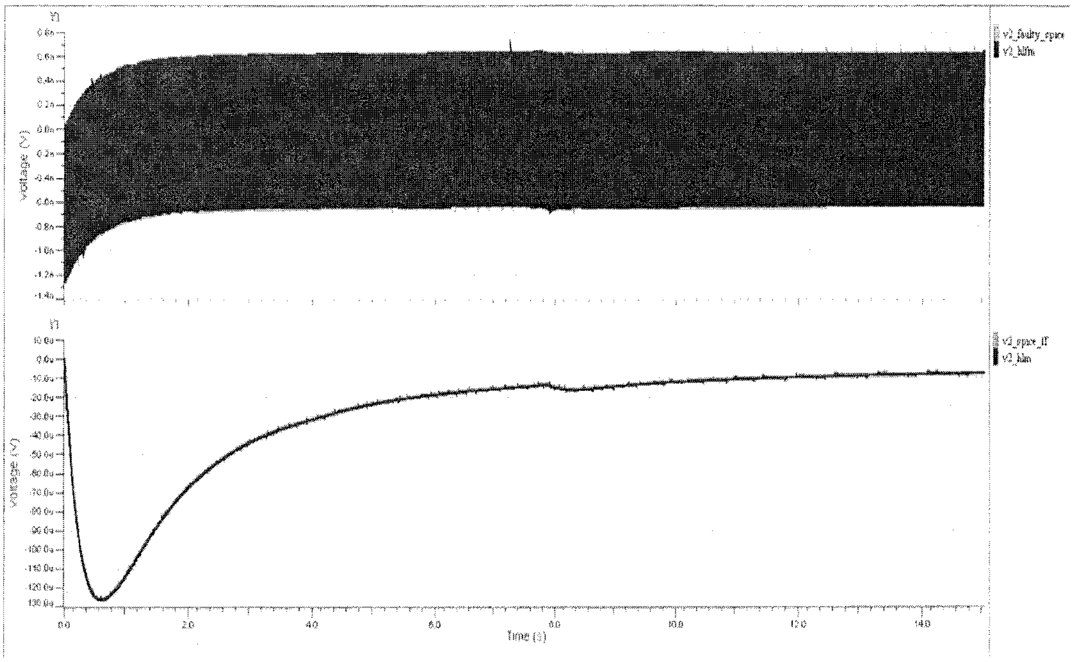
We employ AMC routine to generate AMG-CNIP and MMGSD HLFMs. An input sine waveform is applied with 500 Hz frequency and ±2.0A current amplitude and transient simulation of 15 sec is run.

MMGSD fails to perform HLFM and show very high relative error percentage (>550%) and on average consume more than 300 seconds of CPU time to simulate Config1 circuit in SystemVision. The simulation result of AMG-CNIP for Config1 circuit is shown in figure 4-16.

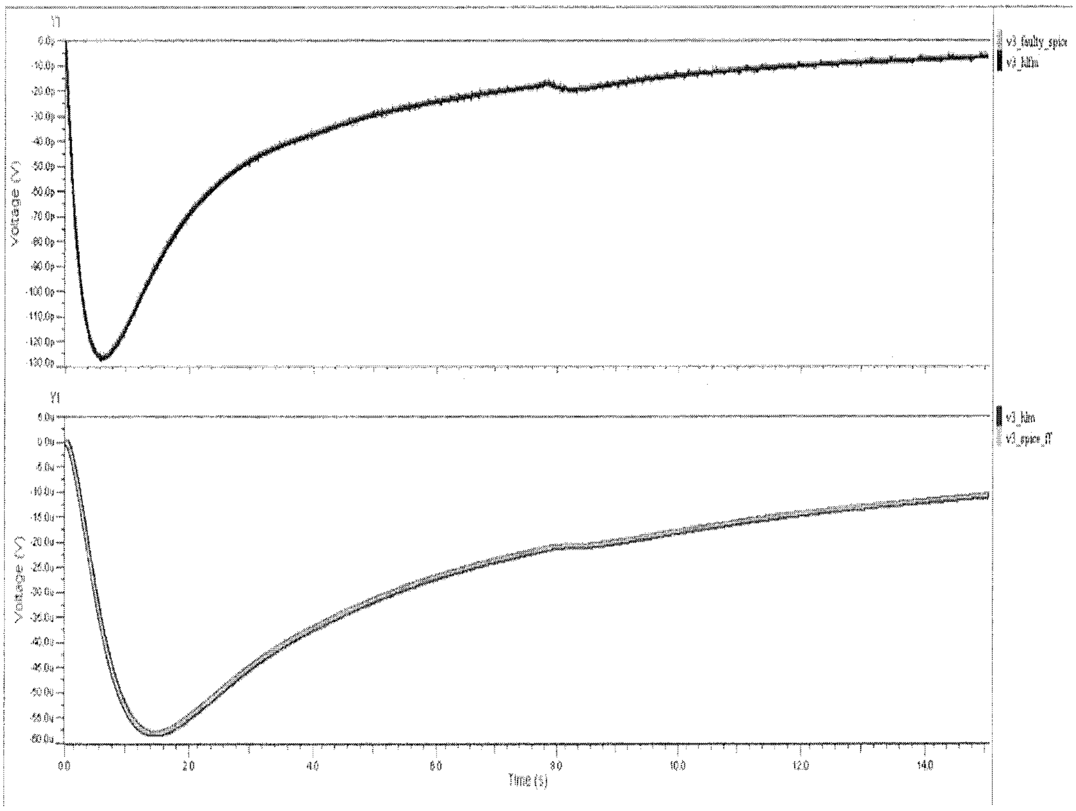
The outputs are taken at nodes 1, 2 and 3. We also show fault-free responses at nodes 1, 2 and 3. It is very important to notice here that HLM with similar AMG-CNIP settings cannot predict correct response for faulty circuit as is clear from the difference between faulty and fault free response at respective nodes. E.g. voltage at node 1 (figure 4-16(a)) swings between -1.2mV and 0.6mV for fault-free circuit (lower waveform), whereas voltage at node 1 swings between -2uV and 2uV for faulty circuit (upper waveform). Similarly, the difference in amplitudes is obvious at node 2 and 3 (figure 4-16(a), 4-16(b)) for fault-free and faulty circuit responses. It clearly shows that one cannot simply use HLM for faulty circuit, one need to generate fault model while performing HLFM.



a Transient response of AMG-CNIP HLFM for Config1 at node 1



b Transient response of AMG-CNIP HLFM for Config1 at node 2



c Transient response of AMG-CNIP HLFM for Config1 at node 3

Figure 4-16: Transient response of AMG-CNIP HLM and HLFM for Config1 (nodes 1, 2 and 3) simulation result

Relative error for Config1 measured at node 1 is 3.4e-12, at node 2 is 1.28e-6% and at node 3 is 4.21e-10 that is almost negligible. Average CPU time (for 30 runs) of AMG-CNIP HLFM is 7.3 seconds and average CPU time of SPICE circuit is 16.1 seconds. Simulation speed up achieved is 2.2x.

In Config2 we replace first stage with HLFM and compare with full SPICE 3-stage cascaded NLTx line circuit whose first stage is SPICE faulty block. Simulation result for Config2 is shown in figure 4-17 and the output voltage is taken at node 5 and 6. Good response at node 6 shows that first stage of Config1 generates less error and correct behavior is propagated to next stage.

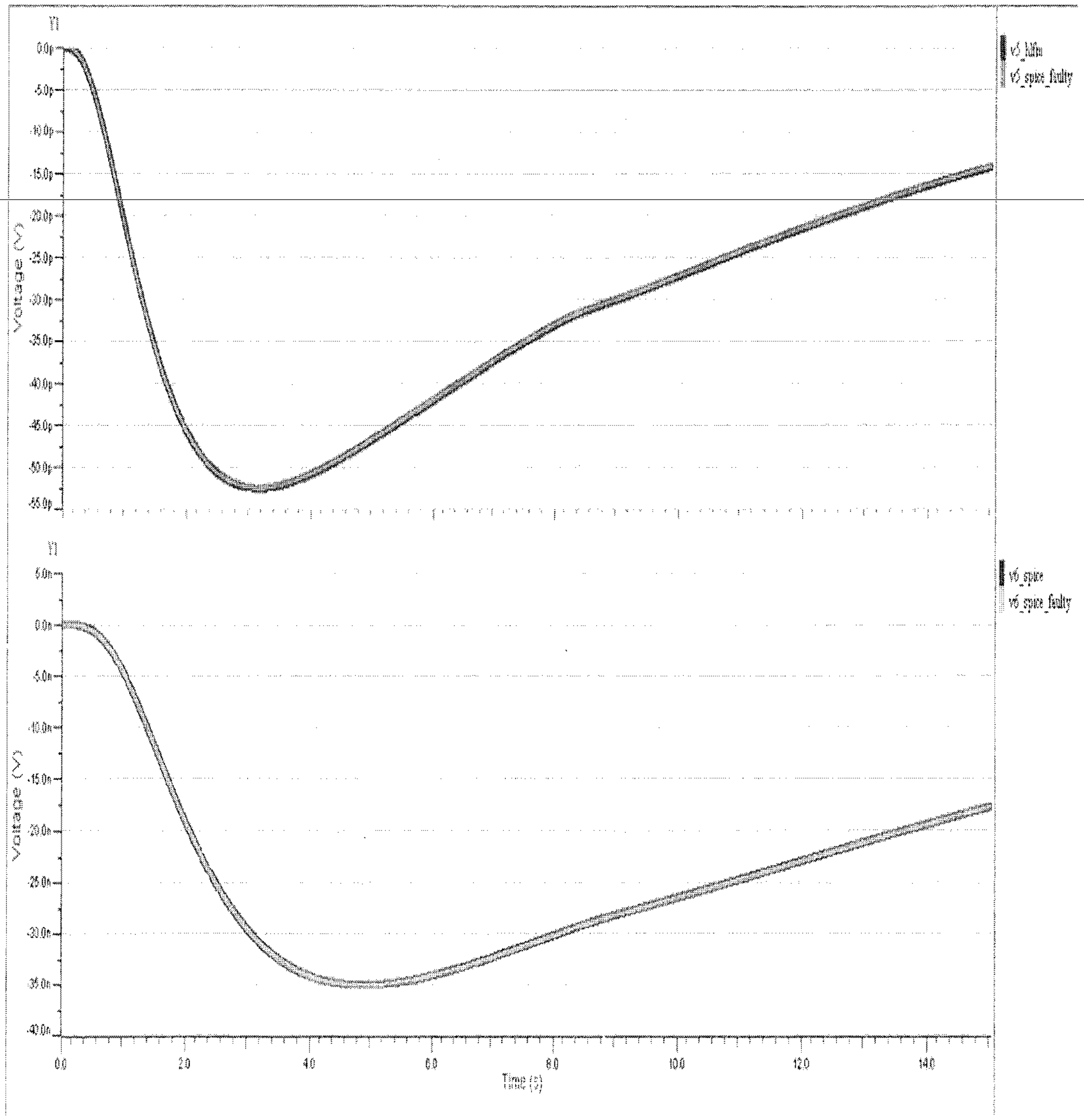


Figure 4-17: Transient response of AMG-CNIP HLFM for Config2 (nodes 5 and 6)

Percentage of relative error measured at node 5 is 3.8e-7 and at node 2 is 3.02e-7 that is negligible. HLFM average CPU time is 12.1 seconds. Average CPU time for full SPICE faulty circuit is 32.7 seconds and a simulation speed up of 2.7x is achieved.

In Config3 and Config4 we replace middle and last stage with AMG-CNIP HLFMs respectively. A transient simulation of 15 seconds is performed and output is taken at nodes 10 and 11 for Config3 and at nodes 11 and 12 for Config4. Simulation results are shown in figure 4-18 and figure 4-19 respectively.

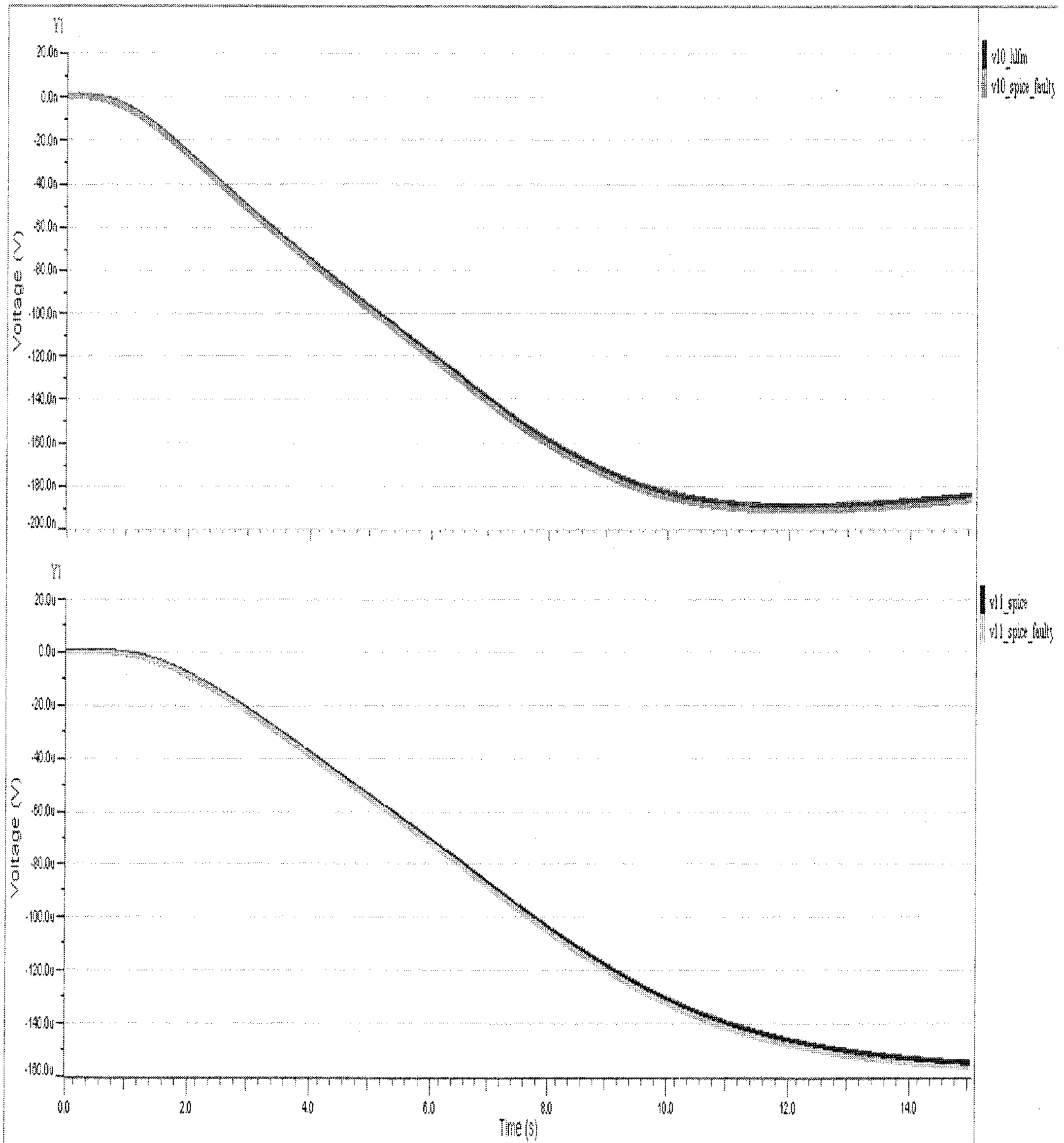


Figure 4-18: Transient response of AMG-CNIP HLFM for Config3 (nodes 10 and 11)

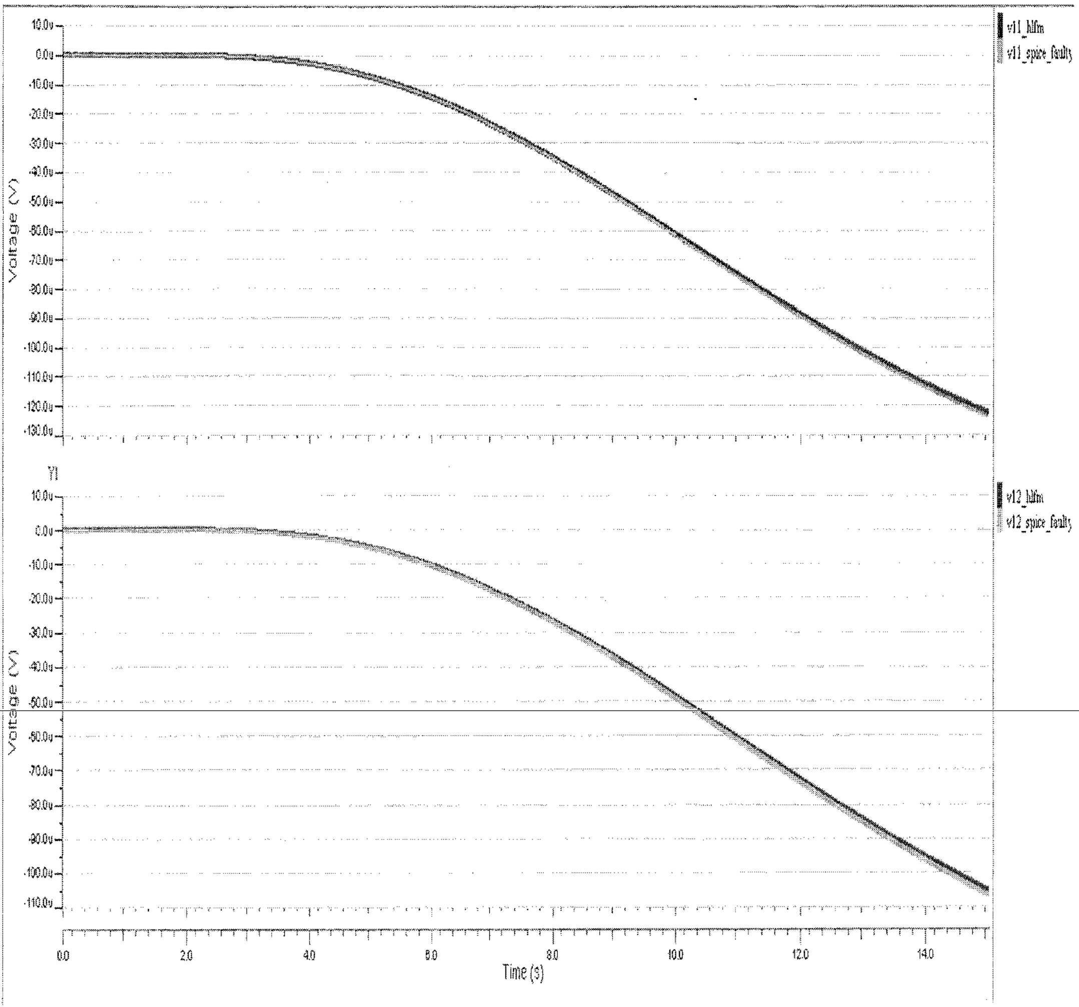


Figure 4-19: AMG-CNIP HLFM Config4 (nodes 11 and 12) simulation result

Speedup achieved for Config3 is 1.09x and speed up achieved for Config4 is 1.08x as compared to full SPICE faulty circuits. It is important to note that simulation speed up achieved in Config2 (i.e. 2.7x) is larger than in latter two cases (Config3 and Config4) i.e. 1.09x and 1.08x, although single stage is replaced in all cases (Config2, Config3 and Config4). Config2 and Config1 show almost similar speedup. In Config2 HLFM come across with similar operating conditions as Config1 (i.e. both directly interface with input current source and configuration is similar to training configuration) whereas in Config3 and Config4, operating conditions are different for HLFM as they are interfaced with SPICE circuit instead of input current source. Therefore, HLFMs are evaluated more rigorously in these configurations and thus show less speedup with showing reasonable accuracy.

In Config5 we replace first stage with HLM and second stage with HLFM and last stage remains SPICE circuit. Outputs are taken at node 5, 6 and 11. Simulation result is shown in figure 4-20.

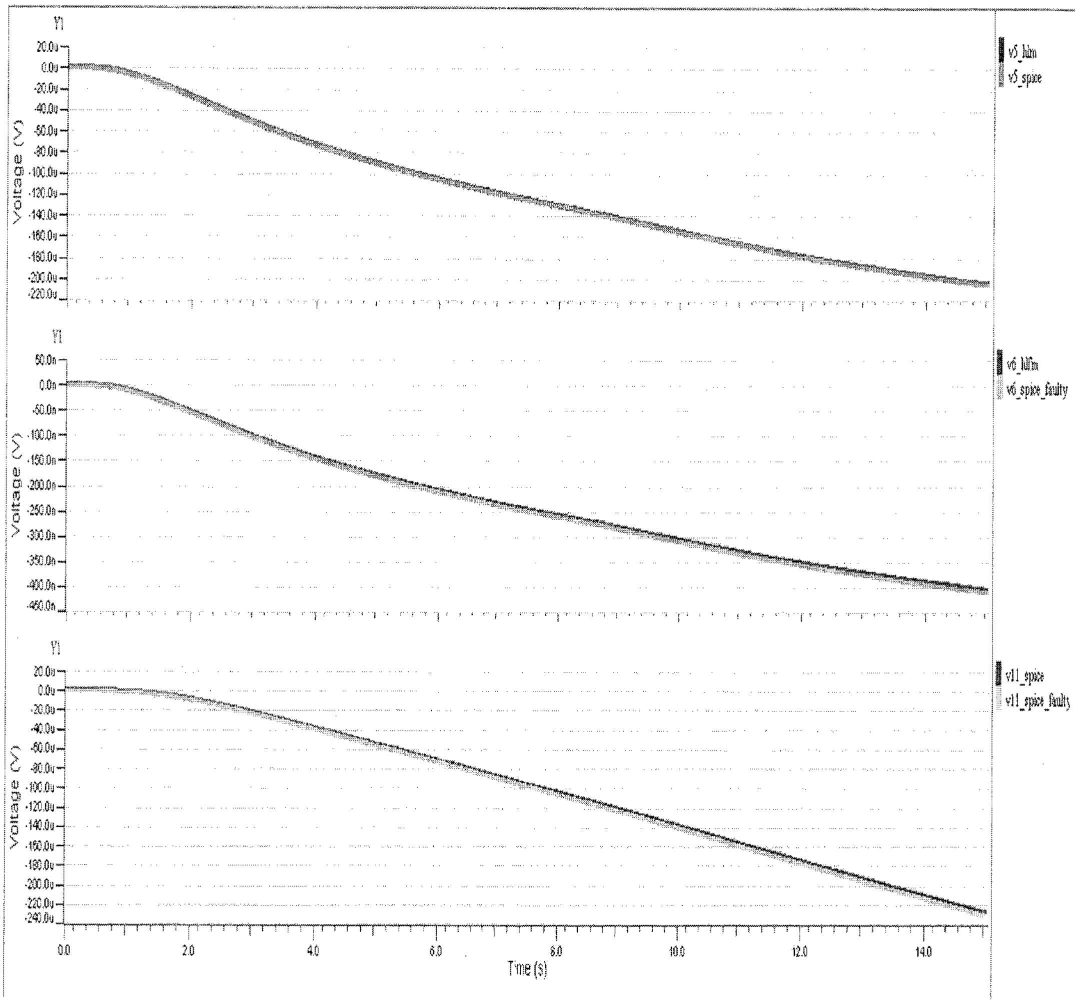


Figure 4-20: Transient response of AMG-CNIP HLFM for Config5 (nodes 5, 6 and 11)

Relative error measured at node 5 is 1.518e-6, at node 6 is 0.01% and at node 11 is 0.09% and simulation speedup achieved is 1.78x. A little more speedup is achieved in this case (compared to previous cases) as we replace first two stages with HLM and HLFM respectively as compared to single stage replacement in Config1, Cong2 and Config3.

In Config6 first two stages are replaced with HLM and last stage is replaced with HLFM. The purpose of this experiment is to test if more speedup can be achieved by using more HLMs in circuit at system level. The output is taken at node 10 (HLM output) and node 11 (HLFM output). Simulation result is shown in figure 4-21.

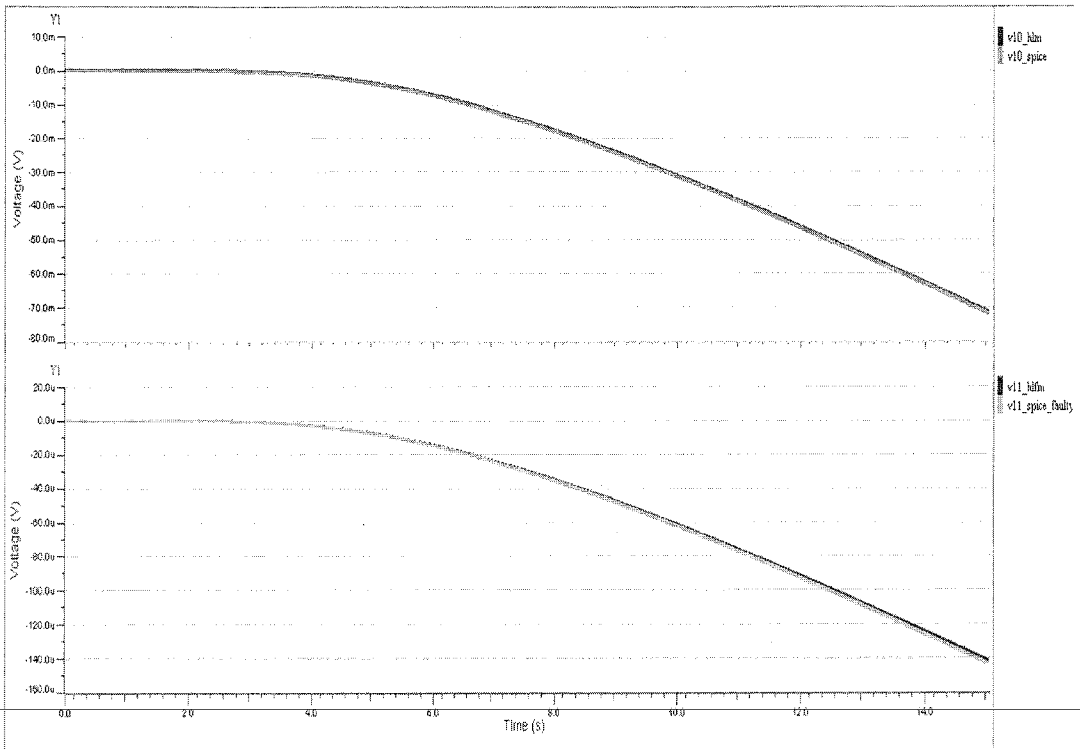


Figure 4-21: Transient response of AMG-CNIP HLFM for Config6 (nodes 10 and 11)

Relative error percentage measure at node 11 is 0.01% and a simulation speed up of 2.85x is achieved. This shows that replacing circuit with more HLMs can achieve greater simulation speed up compared to full SPICE circuit.

A summary of simulation speed up and relative error percentages for six configurations is given in Table 4-5.

Table 4-5: Summary of simulation speed up and accuracy for AMG-CNIP HLFM (fault at diode D1)

Configuration D1 Faulty	SPICE (sec)		AMG-CNIP HLFM (sec)		Speed up $\frac{SPICE}{AMG\_CNIP}$	Accuracy Rel. error (%)
	Mean	Max	Mean	Max		
NLTX line	16.1	20.5	7.3	11.8	2.2x	3.4e-12
HLFM_SPICE_SPICE	32.7	36.3	12.1	25.6	2.7x	3.8e-7
SPICE_HLFM_SPICE	29.96	33.5	27.3	31.7	1.09x	0.001
SPICE_SPICE_HLFM	30.2	38	27.5	36.6	1.08x	0.01
HLM_HLFM_SPICE	29.96	33.5	12.1	16.8	1.78x	0.09
HLM_HLM_HLFM	30.2	38	10.59	16.1	2.85x	0.01

4.5.2 HLFM for Short Fault at Diode D2

In second experimentation we introduce a fault at diode D2 shown in figure 4-22 and test HLFM in different six configurations shown in Table 4-6.

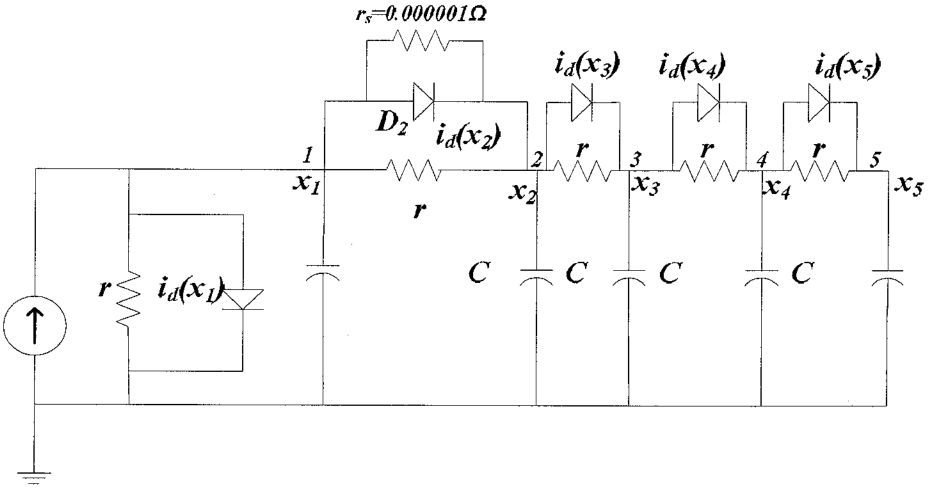


Figure 4-22: NLTx circuit with short fault at D2

Again we start by generating AMG-CNIP macromodel and MMGSD macromodel in MATLAB. AMG-CNIP use 3rd order CN polynomial; with ARF range for nonlinear function *f(x)* to be [-1e-6, 1e-6].

Table 4-6: Configurations for AMG-CNIP HLFM testing (short fault at D2)

Configuration No.	**Settings**
Config 1	NLTx line
Config 2 (3-stage Cascaded)	HLFM – SPICE – SPICE
Config 3 (3-stage Cascaded)	SPICE – HLFM – SPICE
Config 4 (3-stage Cascaded)	SPICE – SPICE – HLFM
Config 5 (3-stage Cascaded)	HLM – HLFM – SPICE
Config 6 (3-stage Cascaded)	HLM – HLM – HLFM

MMGSD has approximately 154us of settling time and AMG-CNIP macromodel response is settled from beginning. Ignoring initial 155us, the response of AMG-CNIP and MMGSD for evaluation input is shown in figure 4-23.

HLFS results for 6 configurations (in Table 4-6) with fault at diode D2 are shown in figure 4-24 through figure 4-29. Again notice the difference between fault-free and faulty circuit response in figure 4-24 at node 2.

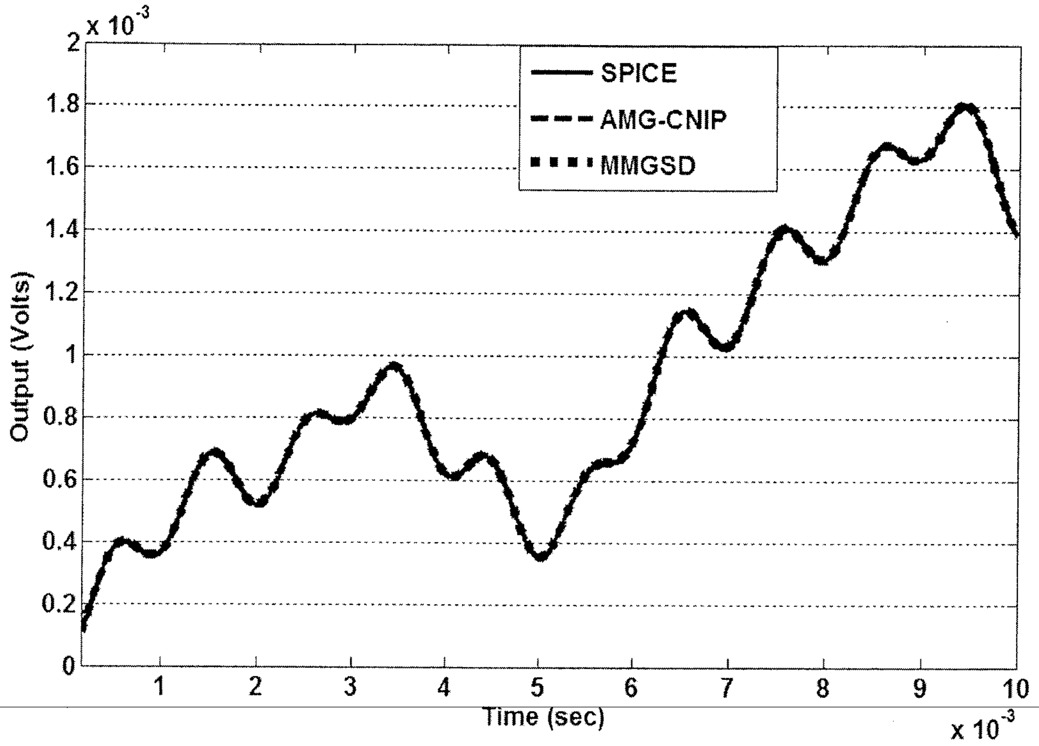


Figure 4-23: MMGSD and AMG-CNIP response for Config1 fault at D2

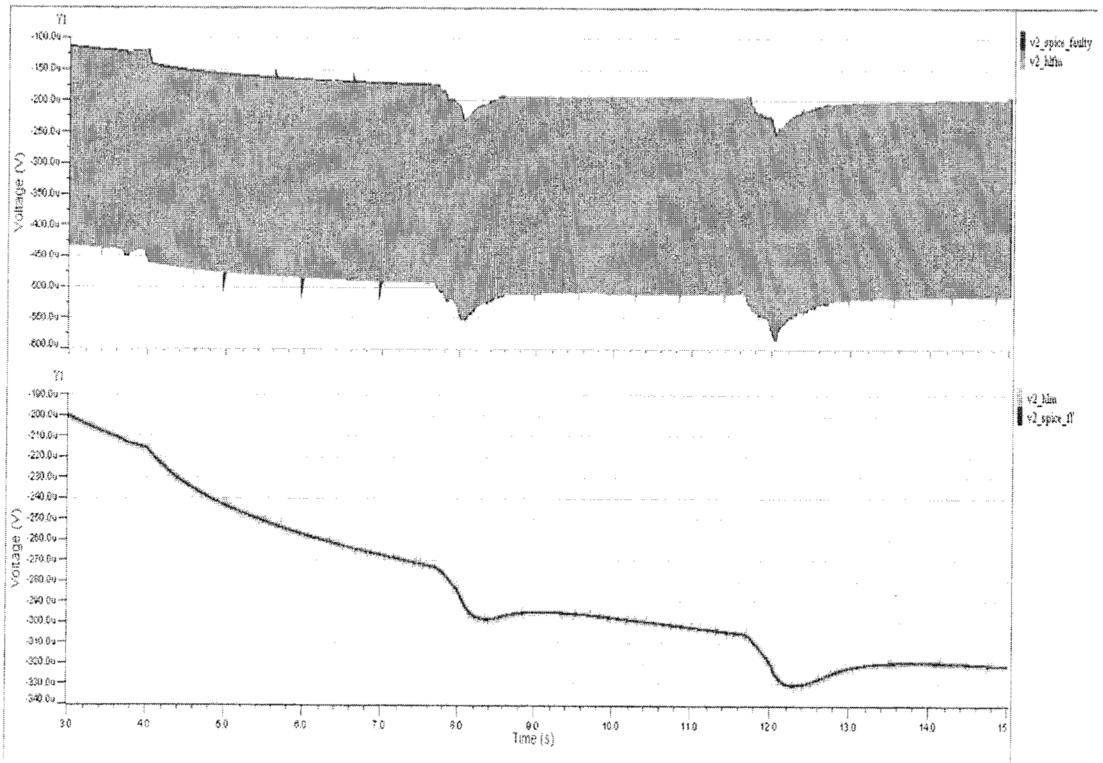


Figure 4-24: Transient response of AMG-CNIP HLM and HLFM for Config1 (node 2) with fault at D2

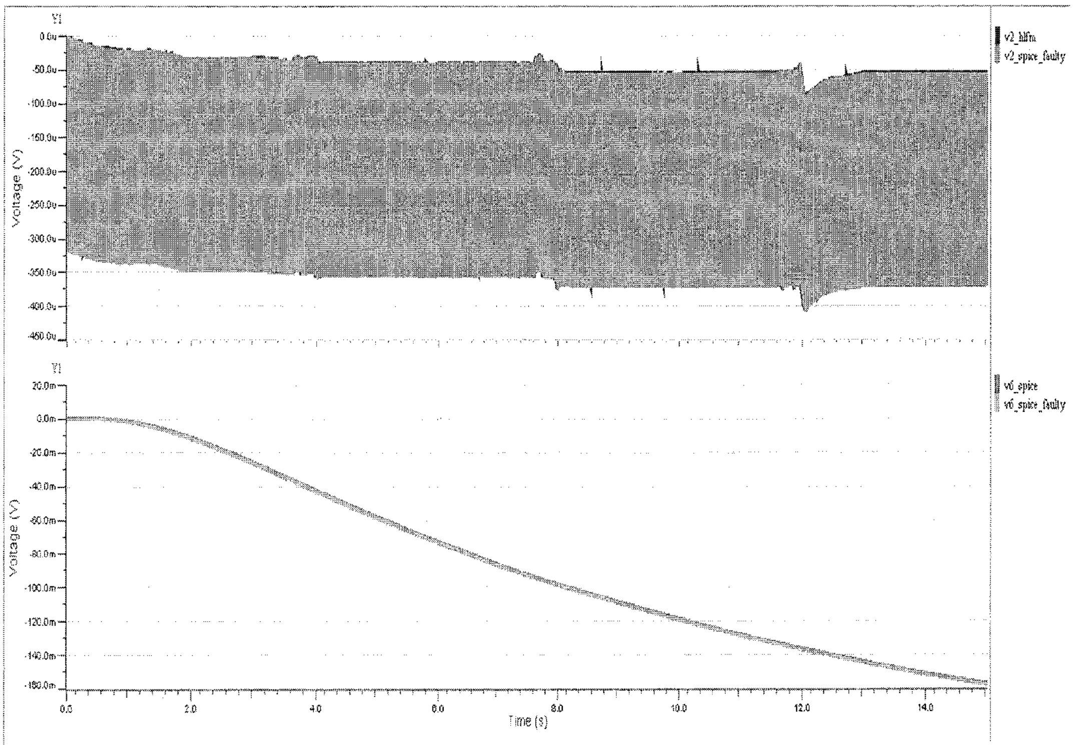


Figure 4-25: Transient response of AMG-CNIP HLFM for Config2 (nodes 2 and 6) with fault at D2

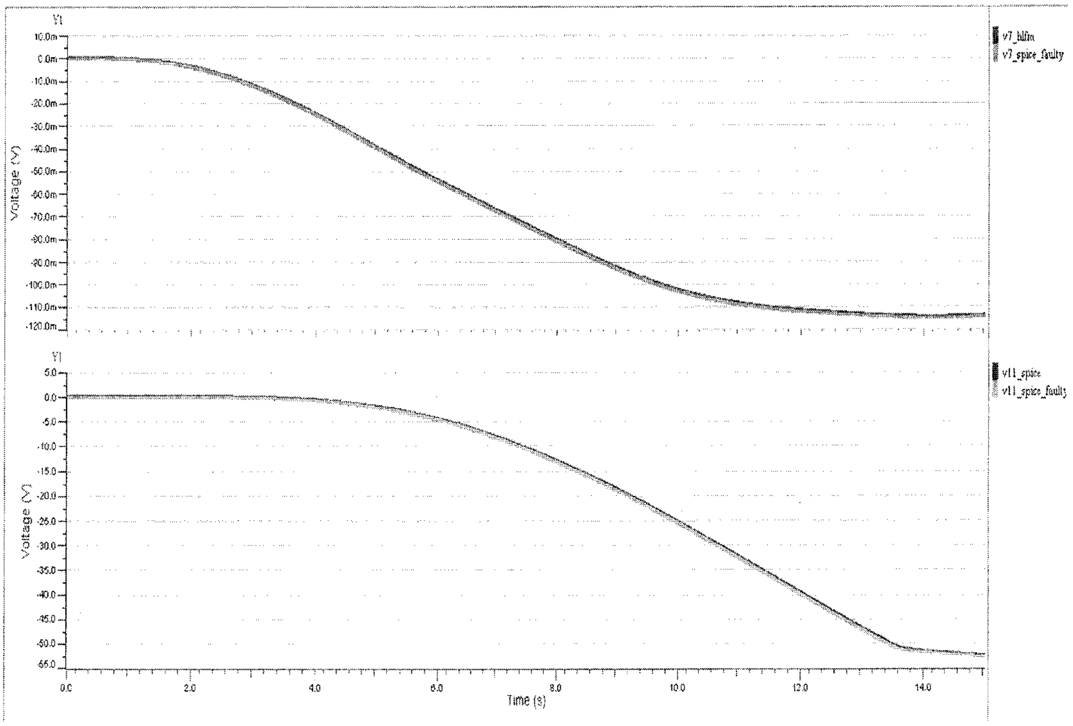


Figure 4-26: Transient response of AMG-CNIP HLFM for Config3 (nodes 7 and 11) with fault at D2

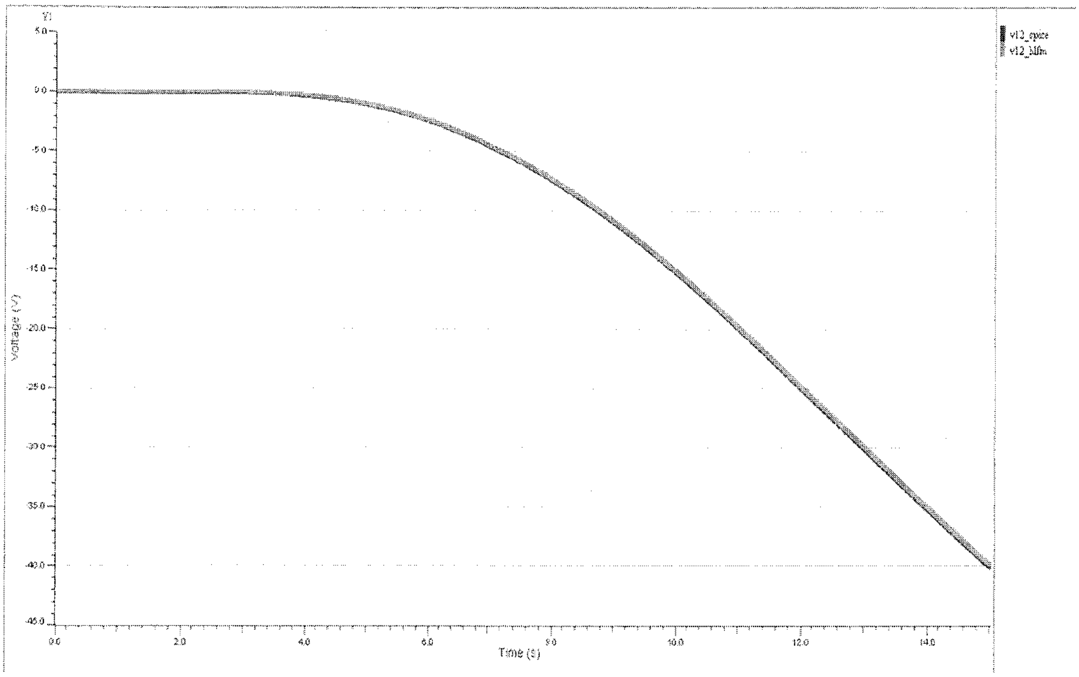


Figure 4-27: Transient response of AMG-CNIP HLFM for Config4 (node 12) with fault at D2

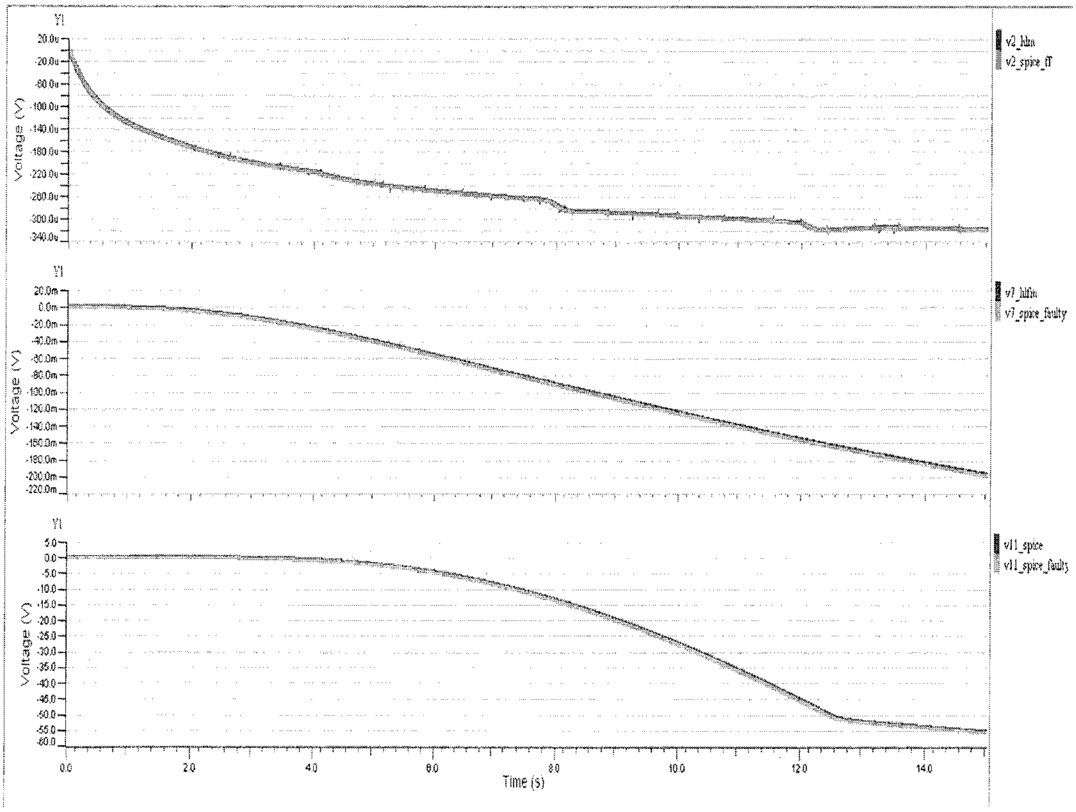


Figure 4-28: Transient response of AMG-CNIP HLFM for Config5 (nodes 2, 7 and 11) with fault at D2

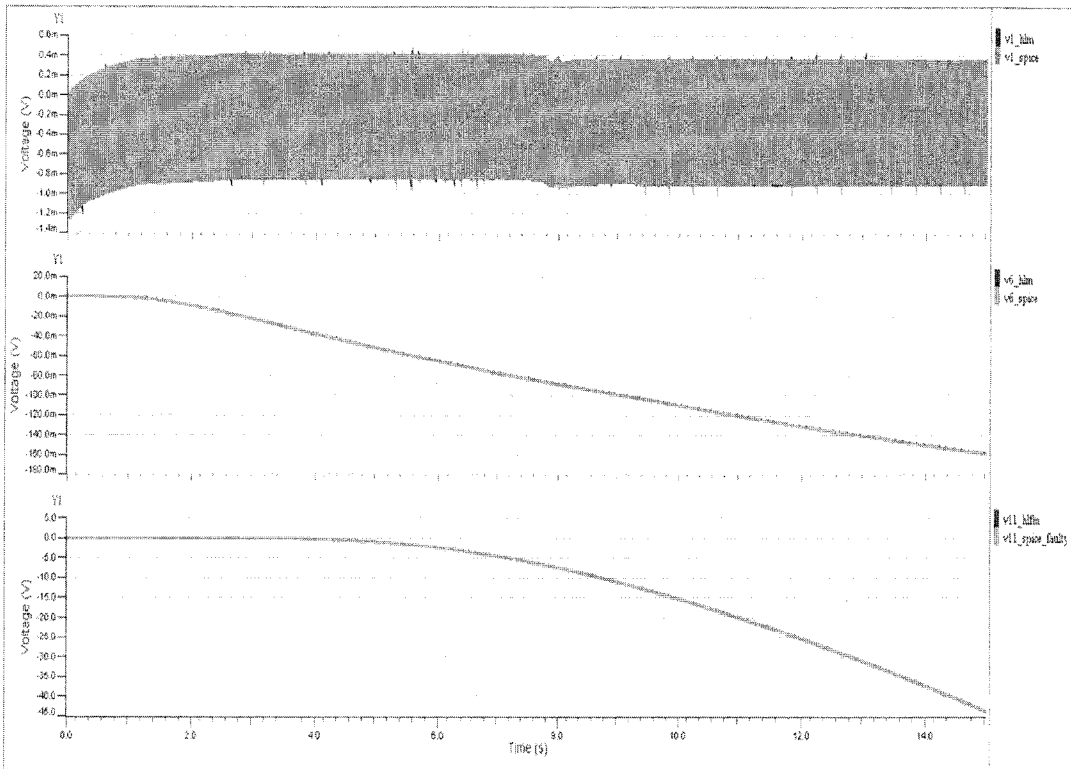


Figure 4-29: Transient response of AMG-CNIP HLFM for
Config6 (nodes 1, 6 and 11) with fault at D2

Table 4-7: Summary of simulation speed up and accuracy for AMG-CNIP HLFM
(short fault at diode D2)

Configuration D2 Faulty	SPICE (sec)		AMG-CNIP HLFM (sec)		Speed up $\frac{SPICE}{AMG\_CNIP}$	Accuracy Rel. error (%)
	Mean	Max	Mean	Max		
NLTX line	17.2	21	12.4	20.6	1.38x	3.75e-10
HLFM_SPICE_SPICE	34.5	38.9	20.5	25.6	1.68x	2.656e-7
SPICE_HLFM_SPICE	50.9	57.3	50.1	56	0.9x	0.005
SPICE_SPICE_HLFM	40.1	47.2	37.5	46.1	1.06x	0.013
HLM_HLFM_SPICE	48.9	54	22.8	27.6	2.14x	0.005
HLM_HLM_HLFM	40.1	47.2	17.6	30.1	2.26x	0.007

One should note that voltage at node 12 in Config4 (figure 4-25) reaches to -40 volts and AMG-CNIP HLFM is able to predict correct response with relative error percentage of 0.013% and a simulation speed up of 1.06x.

A summary of simulation speed up and accuracy measure for fault at diode D2 in six configurations is given in Table 4-7. Again replacing all stages in cascaded configuration (Config6) with behavioral models, highest speedup of 2.26x is achieved.

4.6. Conclusion

In this chapter we perform HLM and HLFM using AMG-CNIP macromodels. A MATLAB routine called AMC is developed for translation of AMG-CNIP MATLAB macromodel to VHDL-AMS behavioral model. HLM and HLFM are performed using two nonlinear transmission line circuits, and tested in different configurations. Simulation results show that simulation speed up from 0.9x to 2.85x is achieved compared to full SPICE circuit simulation with a sensible accuracy (overall percentage relative error < 2%). In cases where relative percentage error is slightly high, accuracy can be easily improved by increasing CN polynomial order.

It is important to notice that the number of nonlinear components used in the circuits is moderate and HLM and HLFM show reasonable simulation speed up. It is expected that by using circuits having large number of nonlinear components, higher simulation speedup can be achieved.

CHAPTER 5

CONCLUSION AND FUTURE RECOMMENDATIONS

In this dissertation, we studied the problem of generating efficient and accurate models for nonlinear analog circuits and systems with particular emphasis on applications related to the domain of simulation of fault-free and faulty analog circuits. Analog circuits provide a unique set of challenges in circuit design. Although typically analog circuits are much smaller than their digital counterparts; the circuits are extremely nonlinear.

A fault is defect in an analogue circuit that is undesirable and can take other parts of a circuit out of their normal operating regions. Modeling and simulation of faulty circuit is a challenging task compared to fault free circuits.

Our approach to handling the problem by generating efficient simulation models for fault-free and faulty analog circuits is based on white-box ss model structure. Existing methods model nonlinear components of an analogue circuit through Taylor polynomials in nonlinear ss model structure. Multiple linearized Taylor polynomial models are obtained at different operating points to achieve best accuracy at output. Existing methods show good accuracy to model weak as well as strongly nonlinear behavior. However, main focus is on generating models for fault-free analog circuit and very little attention is given by researchers to generate faulty model and test at system level for simulation speedup compared to standard SPICE circuit simulations. Few cases, e.g., Xia et. al. [18] actually implements fault-free and faulty circuit models in electronic domain to perform HLM and HLFM. Unfortunately to date, none of the authors show simulation speedup at system level while performing HLFM.

In this dissertation we develop a novel AMG algorithm termed AMG-CNIP that focuses on the objectives of achieving simulation speedup at system level while performing HLM and HLFM. The novelty in the work is to linearize nonlinear components in the circuit using single polynomial generated through a fusion of Chebyshev and Newton interpolating polynomials instead of using multiple Taylor polynomial models. Our work in AMG development mainly focuses on following aspects of generating/using a macromodel:

- *Computational Efficiency*: Single CN polynomial is used to linearize all nonlinear components in the ss. Single model macromodel avoids model switching that consumes significant time during simulation and become computational overhead. Hence, with single CN polynomial approach, speedups are achieved at system level for fault-free and faulty analog circuit simulations compared to standard SPICE fault-free and faulty circuit simulations.
- *Modeling Accuracy*: CN polynomial is introduced in the ss with the fact keeping in mind that AMG-CNIP macromodel will be evaluated at system level by performing HLM and HLFM and model may come across with unexpected operating conditions that are not covered during model training. Best approximation/ interpolation properties of CN polynomial enable HLM and HLFM to achieve good accuracy.

As a demonstration of our proposed methodology, we generate AMG-CNIP macromodels for nonlinear transmission line circuit problem. Initially macromodels are validated in MATLAB and then translated into VHDL-AMS behavioral models to perform HLM and HLFM at system level in different configurations. Simulation results show that overall speedup of 0.98x to 2.8x is achieved as compared to full SPICE circuit simulations. It is also seen that good accuracy is obtained at system level with relative error percentage less than ~2% in all cases. In some cases where relative error is slightly high, accuracy can be easily improved by increase CN polynomial order.

5.1.Future Work and Recommendations

Our proposed algorithm successfully achieved the objectives of the project of achieving simulation speedup at system level with reasonable accuracy. However, there are many potential areas that require further development in the AMG-CNIP algorithm.

- Immediate development in AMG-CNIP is to extend the algorithm for modeling of analog circuits containing transistors, e.g., MOSFET (level 1, 2, 3, BSIM1, BSIM2, BSIM3 etc). In our current implementation charge (or capacitance) i.e., nonlinear function $q(x)$ is taken as linear function of state variable x, whereas in transistors charge (capacitance) is a nonlinear function of state variable x. Similar procedure of obtaining CN polynomial for nonlinear function $f(x)$ can be applied to obtain another CN polynomial for nonlinear function $q(x)$. One would need to apply numerical integration method along with Newton-Raphson iterative method to solve set of nonlinear differential equations of AMG-CNIP macromodel.

- The extended algorithm can be implemented to perform HLM and HLFM of transistor level circuits. Since modern circuits consists of MOSFET as their basic building block, so it is critical to evaluate the AMG-CNIP algorithm for transistor level circuit, perform HLM and HLFM of transistor circuits and check for simulation speed up at system level.
- An important development is to invent a technique for multi-dimensional multivariate interpolating polynomial. This is the core of our algorithm that we expect to break through in the field of modeling. Existing interpolation methods are only multivariate, not multidimensional. Here multidimensional means that a polynomial interpolates between more than one nonlinear functions simultaneously, e.g., in our current implementation a diode works in its single operating region and we generate CN interpolating polynomial only for *one* nonlinear current function. For devices like MOSFET, switching between different operating regions (e.g., sub-threshold, linear, saturation regions etc.) there is separate nonlinear current function for each region. To generate single polynomial for all regions, one would need to generate a polynomial that interpolates between different nonlinear functions simultaneously. Similar multidimensional interpolating polynomial will be required for nonlinear charge (capacitance) function.
- Another important development is the application of MOR methods on AMG-CNIP macromodels. Current MOR methods are applicable for Taylor based macromodels, whereas to reduce order of AMG-CNIP macromodel, one would need to develop MOR technique for CN polynomial.
- In our current implementation, there are moderate numbers of nonlinear

components in the circuit and small numbers of speedup are shown. One can test the algorithm with large number of nonlinear components to check for higher number of speedup achievement.

- AMG-CNIP shows stable response compared to existing algorithms, but a stronger mathematical formulation is required for stability preserving analysis of CN polynomial based AMG-CNIP macromodels.

REFERENCES

[1] C. Wang, F, *Digital Circuit Testing: A Guide to DFT and Other Techniques.* Academic Press, 1991.

[2] S. D. Dasnurkar and J. a. Abraham, "Real-time dynamic hybrid BiST solution for Very-Low-Cost ATE production testing of A/D converters with controlled DPPM," in *2010 11th International Symposium on Quality Electronic Design (ISQED)*, 2010, pp. 562–569.

[3] S. J. Spinks, *Fault simulation for structural testing of analogue integrated circuits.* Hull Univ. (United Kingdom), 1998.

[4] S. SUNTER and M. GRAPHICS, "Essential principles for practical analog BIST," *Electronic Design News*, 2010.

[5] J. A. Abraham and M. Soma, "Mixed-Signal Test-Tutorial C," in *The European Design and Test Conference (ED&TC)*, 1995.

[6] K. Arabi, "Special session 6C: New topic mixed-signal test impact to SoC commercialization," in *VLSI Test Symposium (VTS), 2010 28th*, 2010, pp. 212–212.

[7] E. K. Bartsch, "Improved Analogue Fault Simulation Through High Level Modeling," University of Hull, 1999.

[8] P. Kalpana and K. Gunavathi, "Test-generation-based fault detection in analog VLSI circuits using neural networks," *ETRI journal*, vol. 31, no. 2, pp. 209–214, Apr. 2009.

[9] B. a. a. Antao and F. M. El-Turky, "Automatic Analog Model Generation For Behavioral Simulation," *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 12.2.1–12.2.4, 1992.

[10] P. R. Wilson, Y. Kilic, J. N. Ross, M. Zwolinski, and A. D. Brown, "Behavioural modelling of operational amplifier faults using analogue hardware description languages," in *Proceedings of the Fifth IEEE International Workshop on Behavioral Modeling and Simulation. BMAS 2001 (Cat No.01TH8601)*, pp. 106–112.

[11] Y. Kilic and M. Zwolinski, "Behavioral Fault Modeling and Simulation Using VHDL-AMS to Speed-Up Analog Fault Simulation," *Analog Integrated circuits and signal processing*, vol. 39, no. 2, pp. 177–190, 2004.

[12] Y. Joannon, V. Beroulle, C. Robach, S. Tedjini, and J.-L. Carbonero, "Choice of a High-Level Fault Model for the Optimization of Validation Test Set Reused for Manufacturing Test," *VLSI Design*, vol. 2008, pp. 1–10, 2008.

[13] G. Gronthoud and H. G. Kerkhoff, "Reducing analogue fault-simulation time by using high -level modelling in dotss for an industrial design," *IEEE European Test Workshop, 2001.*, pp. 61–67, 2001.

[14] E.K.Bartsch and I. Bell, "High Level Analogue fault simulation using linear and non linear models," *Radioengineering Journal*, vol. 8, pp. 32–34, 1999.

[15] J. Roychowdhury, "Algorithmic macromodelling methods for mixed-signal systems," *17th International Conference on VLSI Design. Proceedings.*, pp. 141–147, 2004.

[16] N. Dong, "Automated nonlinear macromodeling for fast analog circuit simulation," University of Minnesota, 2006.

[17] B. N. Bond, Z. Mahmood, Y. Li, R. Sredojevi, A. Megretski, V. Stojanovi, Y. Avniel, and L. Daniel, "Compact Modeling of Nonlinear Analog Circuits Using System Identification Via Semidefinite Programming and Incremental Stability Certification," *IEEE Trans. on Computer-Aided Design of Int. Circuits and Systems*, vol. 29, no. 8, pp. 1149–1162, 2010.

[18] L. Xia, I. M. Bell, and A. J. Wilkinson, "Automated Model Generation Algorithm for High-Level Fault Modeling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 7, pp. 1140–1145, Jul. 2010.

[19] M. U. Farooq, L. Xia, F. A. Hussin, and A. S. Malik, "Fault Modeling of Analog Circuits Using System Identification Automated Model Generation Approaches from SPICE level Descriptions," *Advanced Science Letters (ASL)*, vol. 19, no. 5, pp. 1276–1280, 2013.

[20] B. Yang and B. Mcgaughy, "An Essentially Non-Oscillatory (ENO) High-Order Accurate Adaptive Table Model for Device Modeling," in *Design Automation Conference*, 2004, pp. 864–867.

[21] M. Swaminathan, B. Mutnury, and J. Libous, "Macro-modelling of non-linear I/O drivers using spline functions and finite time difference approximation," in *Electrical Performance of Electronic Packaging*, 2003, pp. 273–276.

[22] E. Davalo and P. Naïm, *Neural Networks.* Macmillan Education Ltd., 1991.

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

[24] H. B. Verbruggen and R. Babuska, *Fuzzy Logic Control: Advances in Applications.* World Scientific Publishing Company, 1999.

[25] F. J. Uppal and R.J. Patton, "Neuro-fuzzy uncertainty de-coupling: a multiple-model paradigm for fault detection and isolation," *INTERNATIONAL JOURNAL OF ADAPTIVE CONTROL AND SIGNAL PROCESSING*, vol. 19, no. 4, pp. 281–304, 2005.

[26] R. . Middleton and G. . Goodwin, *Digital Control and Estimation - A Unified Approach.* Prentice-Hall, 1990.

[27] L. Ljung, *System Identification Theory for User.* Upper Saddle River: Prentice Hall PTR, 1999.

[28] C. S. Gathercole and H. a. Mantooth, "Modeling nonlinear dynamics in analog circuits via root localization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 7, pp. 895–907, Jul. 2003.

[29] S. X.-D. Tan and C.-J. R. Shi, "Efficient DDD-based term generation algorithm for analog circuit behavioral modeling," in *Proceedings of the ASP-DAC Asia and South Pacific Design Automation Conference, 2003.*, 2003, pp. 789–794.

[30] Y. Wei and A. Doboli, "Systematic Development of Nonlinear Analog Circuit Macromodels through Successive Operator Composition and Nonlinear Model Decoupling," in *Design*, 2006, pp. 1023–1028.

[31] G. Gielen, T. McConaghy, and T. Eeckelaert, "Performance space modeling for hierarchical synthesis of analog integrated circuits," *Proceedings of the 42nd annual ...*, pp. 881–886, 2005.

[32] J. Roychowdhury, "Automated Macromodel Generation for Electronic Systems," in *Behavioral Modeling and Simulation (BMAS)*, 2003, pp. 11–16.

[33] L. T. Pillage and R. A. Rohrer, "Asymptotic Waveform Evaluation for timing analysis," *IEEE Trans. on Computer -Aided Design*, vol. 9, no. 4, pp. 352–366, 1990.

[34] J. R. Phillips, "Model reduction of time-varying linear systems using approximate multipoint Krylov-subspace projectors," in *1998 IEEE/ACM*

International Conference on Computer-Aided Design. Digest of Technical Papers (IEEE Cat. No.98CB36287), 1998, pp. 96–102.

[35] J. Roychowdhury, "Reduced-order modeling of time-varying systems," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 46, no. 10, pp. 1273–1288, 1999.

[36] P. Li and L. T. Pileggi, "NORM : Compact Model Order Reduction of Weakly Nonlinear Systems," in *Order A Journal On The Theory Of Ordered Sets And Its Applications*, 2003, pp. 1–6.

[37] M. Rewienski and J. White, "A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 2, pp. 155–170, Feb. 2003.

[38] J. Roychowdhury, "Piecewise polynomial nonlinear model reduction," in *Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)*, 2003, pp. 484–489.

[39] S. K.Tiwary and R. A. Rutenbar, "Scalable trajectory methods for on-demand analog macromodel extraction," in *Proceedings of the 42nd annual Design and Automation*, 2005, pp. 403–408.

[40] D. De Jonghe and G. Gielen, "Characterization of Analog Circuits Using Transfer Function Trajectories," *IEEE Transcation on Circuits and Systems I*, vol. 59, no. 8, pp. 1796–1804, Aug. 2012.

[41] E. . Grimme, "Krylov projection methods for model reduction," University of Illinios Urbana-Champaign, 1997.

[42] J. Roychowdhury, "Algorithmic methods for bottom-up generation of system-level RF macromodels," in *Control, Communications and Signal Processing, 2004. First International Symposium on*, 2004, pp. 57–62.

[43] A. Odabasioglu, M. Celik, and L. T. Pileggi, "PRIMA: passive reduced-order interconnect macromodeling algorithm," *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD) ICCAD-97*, pp. 58–65, 1997.

[44] N. Dong and J. Roychowdhury, "Automated extraction of broadly applicable nonlinear analog macromodels from spice-level descriptions," in *Custom*

Integrated Circuits Conference, 2004. Proceedings of the IEEE 2004, 2004, pp. 117–120.

[45] R. W. Freund, “SPRIM: structure-preserving reduced-order interconnect macromodeling,” *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.*, pp. 80–87, 2005.

[46] B. N. Bond and L. Daniel, “A Piecewise-Linear Moment-Matching Approach to Parameterized Model-Order Reduction for Highly Nonlinear Systems,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 12, pp. 2116–2129, Dec. 2007.

[47] Z. Mahmood, B. Bond, T. Moselhy, A. Megretski, and L. Daniel, “Passive reduced order modeling of multiport interconnects via semidefinite programming,” in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2010, pp. 622–625.

[48] L. Ljung, “Perspectives on system identification,” *Annual Reviews in Control*, vol. 34, no. 1, pp. 1–12, Apr. 2010.

[49] K. J. Astrom, “Lectures on the identification problem- the least square method,” Division of Automatic Control. Lund Institute of Technology. Lund. Sweden, 1968.

[50] T. C. Hsia, *System identification: least-squares methods*, no. x. Lexington Books, Lexington.: , 1977.

[51] J. M. Mendel, *Discrete techniques of parameter estimation: the equation error formulation.* Marcel Dekker, Newyork: , 1973.

[52] T. Soderstron, P. Stoica, and T. Soderstrom, *System Identification.* Englewood Cliffs: Prentice-Hall, 1989.

[53] R. Pintelon and J. Schoukens, *System Identification: A Frequency Domain Approach.* Picataway: IEEE Press, 2001.

[54] V. Overschee and B. DeMoor, *N4SID subspace algorithms for the identification of combined deterministic-stochastic systems*, vol. 10, no. 2–3. 1994, pp. 75–93.

[55] “Identifying Nonlinear ARX Models Nonlinear Black-Box Model Identification (System Identification Toolbox^{TM}).” .

[56] M. U. Farooq, L. Xia, F. Hussin, and A. Malik, "Performance Evaluation Of Nonlinear Automated Model Generation Approaches For High Level Fault Modeling," in *7th IEEE Conference on Industrial Electronics and Applications (ICIAS)*, 2012, pp. 1842–1846.

[57] M. U. Farooq, L. Xia, F. A. Hussin, and A. S. Malik, "High level fault modeling and fault propagation in analog circuits using NLARX automated model generation technique," in *2012 4th International Conference on Intelligent and Advanced Systems (ICIAS2012)*, 2012, pp. 846–850.

[58] E. Simeu and S. Mir, "Diagnosis in Linear and Nonlinear Mixed-Signal Systems : a Parameter Identification Based Technique," 2005.

[59] "STINS A Matlab tool for Stable Identification of Nonlinear systems via Semidefinite programming." .

[60] "SPOT: Syst. Polynomial Optimization Toolbox [Online]. Available: http://web.mit.edu/ameg/www/." .

[61] J. F. Sturm, "Using SeDuMi 1 . 02 , a MATLAB toolbox for optimization over symmetric cones 1 Introduction to SeDuMi," *Online*, pp. 1–24, 1998.

[62] B. N. Bond and L. Daniel, "Automated Compact Dynamical Modeling : An Enabling Tool for Analog Designers," in *Computer-Aided Design*, 2010, pp. 415–420.

[63] L. Xia, I. M. Bell, and A. J. Wilkinson, "A Robust approach for automated model generation," in *4th IEEE International Conference on Design and Technology of Integrated systems of Nanoscale era*, 2009, pp. 281–286.

[64] L. Xia, I. M. Bell, and A. J. Wilkinson, "Automated macromodel generation for high level modeling," *3rd International Conference on Design and Technology of Integrated Systems in Nanoscale Era*, pp. 1–6, Mar. 2008.

[65] N. Dong and J. Roychowdhury, "General-Purpose Nonlinear Model-Order Reduction Using Piecewise-Polynomial Representations," *Computer-Aided Design*, vol. 27, no. 2, pp. 249–264, 2008.

[66] I. W. Congress, "EMPIRICAL MODEL REDUCTION OF CONTROLLED NONLINEAR SYSTEMS Sanjay Lall , Jerrold E . Marsden , Sonja Glavaˇ," *World*, pp. 473–478, 1999.

[67] K. Kunisch and S. Volkwein, "Galerkin proper orthogonal decomposition methods for parabolic problems," *Numerische Mathematik*, vol. 90, no. 1, pp. 117–148, Nov. 2001.

[68] S. Volkwein, "Model reduction using proper orthogonal decomposition," *Lecture Notes, Institute of Mathematics and Scientific Computing, University of Graz. see http://www. uni-graz. at/imawww/volkwein/POD. pdf*, pp. 1–42, 2008.

[69] M. Bergmann, C. Bruneau, and A. Iollo, "Improvement of reduced order modeling based on proper orthogonal decomposition," *Methods*, no. x, pp. 1–2, 2008.

[70] P. M. Vienna, I. Troch, F. Breitenecker, B. Salimbahrami, B. Lohmann, T. Bechtold, and J. G. Korvink, "Two-sided Arnoldi Algorithm and Its Application in Order Reduction of MEMS," *Computer*, vol. 2003, no. February, pp. 1021–1028, 2003.

[71] C. Lanczos, "An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators," *Journal Of Research Of The National Bureau Of Standards*, vol. 45, no. 4, pp. 255–282, 1950.

[72] R. Freund, "Krylov-subspace methods for reduced-order modeling in circuit simulation," *Journal of Computational and Applied Mathematics*, vol. 123, no. 1–2, pp. 395–421, Nov. 2000.

[73] Z. Bai, "Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems," *Applied Numerical Mathematics*, vol. 43, no. 1–2, pp. 9–44, Oct. 2002.

[74] Z. Bai, "Krylov Subspace Techniques for Reduced-Order Modeling of Nonlinear Dynamical Systems Linearization method," *Sciences-New York*, no. x, pp. 1–6.

[75] E. Jonckheere and L. Silverman, "A new set of invariants for linear systems–Application to reduced order compensator design," *Automatic Control, IEEE Transactions on*, vol. 28, no. 10, pp. 953–964, Oct. 1983.

[76] A. Antoulas, "A survey of balancing methods for model reduction," in *Proc. European Control Conference ECC*, 2003, vol. 2, no. 2.

[77] D. Enns, "Model reduction with balanced realizations: An error bound and a frequency weighted generalization," in *Decision and Control, 1984. The 23rd IEEE*, 1984, no. December, pp. 127–132.

[78] K. Fujimoto and J. M. a. Scherpen, "Model reduction for nonlinear systems based on the differential eigenstructure of Hankel operators," in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, 2001, vol. 4, pp. 3252–3257.

[79] K. J. Åström and T. Bohlin., "Numerical identification of linear dynamic systems from normal operating records," in *IFAC Symposium on Self-Adaptive Systems*, 1965.

[80] L. W. Nagel, "SPICE2 A Computer Program to simulate semiconductor circuits," 1975.

[81] Martin Schetzen, *The Volterra and Wiener theories of nonlinear systems.* John Wiley, 1980.

[82] A. H. Nayfeh and B. Balachandran, *Applied Nonlinear Dynamics: Analytical, Computational, and Experimental Methods.* Wiley, 1995.

[83] J. Roychowdhury, "Reduced-order modelling of time-varying systems," *Proceedings of the ASP-DAC '99 Asia and South Pacific Design Automation Conference 1999 (Cat. No.99EX198)*, vol. 46, no. 10, pp. 1273–1288, 1999.

[84] J. R. Phillips, "Automated extraction of nonlinear circuit macromodels," in *Proceedings of the IEEE 2000 Custom Integrated Circuits Conference (Cat. No.00CH37044)*, pp. 451–454.

[85] R. Batra, P. Li, L. T. Pileggi, and Y. T. Chien, "A methodology for analog circuit macromodeling," in *Behavioral Modeling and Simulation Conference, 2004. BMAS 2004. Proceedings of the 2004 IEEE International*, 2004, pp. 41–46.

[86] A. Steinbrecher and T. Stykel, "Model Order Reduction of Nonlinear Circuits," *International Journal of Circuit Theory and Application*, vol. 5, no. 9, 2012.

[87] M. Heinkenschloss and T. Reis, "Model Reduction for a Class of Nonlinear Electrical Circuits by Reduction of Linear Subcircuits," 2010.

[88] D. Vasilyev, M. Rewienski, and J. White, "A TBR-based trajectory piecewise-linear algorithm for generating accurate low-order models for nonlinear analog

circuits and MEMS," in *Proceedings of the 40th conference on Design automation - DAC '03*, 2003, p. 490.

[89] N. Dong and J. Roychowdhury, "Automated Nonlinear Macromodelling of Output Buffers for High-speed Digital Applications," in *IEEE 42nd Annual Design Automation Conference*, 2005, no. 51, pp. 51–56.

[90] "http://embedded.eecs.berkeley.edu/pubs/downloads/spice/index.htm." .

[91] S. A. Nahvi and S. Janardhanan, "Trajectory based methods for nonlinear MOR: Review and perspectives," in *2012 IEEE International Conference on Signal Processing, Computing and Control (ISPCC)*, 2012, pp. 1–6.

[92] J. P. Boyd, *Chebyshev and Fourier Spectrum Methods*. Section 2.7, Dover Publication, 2001.

[93] M. U. Farooq and L. Xia, "Local Approximation Improvement of Trajectory Piecewise Linear Macromodels through Chebyshev Interpolating Polynomials," in *18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2013.

[94] M. J. Rewienski and J. K. White, "A Trajectory Piecewise-Linear Approach to Model Order Reduction of Nonlinear Dynamical Systems," PhD thesis, Massachusetts Institute of Technology, USA, 2003.

[95] W. Y. Yang, W. Cao, T. S. Chung, and J. Morris, *Applied Numerical Methods Using MATLAB*. Wiley-Interscience, 2005, p. 528.

[96] T. L.Quarles, "SPICE 3 Implementation Guide," 1989.

[97] "SystemVision SPICE reference," Mentor Graphics Corporation, 2010.

[98] D. N. Varsamis and N. P. Karampetakis, "On the Newton Multivariate Polynomial Interpolation with Applications," in *7th International Workshop on Multidimensional (nD) Systems (nDs)*, 2011, vol. 0, pp. 1–8.

[99] P. J. Ashenden, G. D. Peterson, and D. A. Teegarden, *The System Designer's Guide to VHDL-AMS*. Morgan Kaufmann Publisher, 2003.

[100] L. Xia, I. M. Bell, and A. J. Wilkinson, "Approach for Automated Model Generation," in *4th International Conference on Design and Techonology of Integrated Systems on Nanascale Era (DTIS09)*, 2009, pp. 281–286.

PUBLICATIONS

[1] Muhammad Umer Farooq, Likun Xia, "Automated Model Generation Approach to Perform Behavioral Modeling (VHDL-AMS) in Analogue Circuits and Systems", Patent No 12MY51.

[2] M.U.Farooq, L.Xia, F.A.Hussin,A.S.Malik,"A Novel Algorithm for Automated Model Generation of Analog Circuits Using Chebyshev-Newton Interpolation", in Adv. Sci. Lett. No. 5, Vol.19, pp.1520-1524, May 2013.

[3] M.U.Farooq, L.Xia, F.A.Hussin,A.S.Malik,"Fault Modeling of Analog Circuits Using System Identification Automated Model Generation Approaches from SPICE level Descriptions", in in Adv. Sci. Lett. No. 5, Vol.19, pp.1276-1280, May 2013.

[4] M.U.Farooq, L.Xia, F.A.Hussin, "A Critical Survey on Automated Model Generation Techniques for High level Modeling and High level Fault Modeling", in IEEE National Postgraduate Conference (NPC 2011), Bandar Seri Iskandar, Universiti Teknologi PETRONAS,Malaysia, 19-20 September, 2011.

[5] M.U.Farooq, L.Xia, F.A.Hussin, A.S.Malik, "Performance Evaluation of Nonlinear Automated Model Generation Approaches for High Level Fault Modeling", in The 7th IEEE Conference on Industrial Electronics and Applications (ICIEA 2012), Singapore,18-20 July, 2012.

[6] M.U.Farooq, L.Xia, F.A.Hussin, A.S.Malik, "High Level Fault Modeling and Fault Propagation in Analog Circuits using NLARX Automated Model Generation Technique", in The 4th IEEE Conference on Intelligent and Advanced Systems (ICIAS 2012), Kuala lumpur, Malaysia ,12-14 June, 2012.

[7] M.U.Farooq, L.Xia, F.A.Hussin, A.S.Malik," A Novel Algorithm for Automated Model Generation of Analog Circuits Using Chebyshev-Newton Interpolation", in International Conference on Advanced Electrical Engineering (ICAEE 2012), Hong Kong , China, 4-5 September, 2012.

[8] M.U.Farooq, L.Xia, F.A.Hussin,A.S.Malik," Fault Modeling of Analog Circuits Using System Identification Automated Model Generation Approaches from SPICE level Descriptions", in International Conference on Advanced Electrical Engineering (ICAEE 2012), Hong Kong , China, 4-5 September, 2012.

[9] M.U.Farooq, L.Xia, F.A.Hussin, A.S.Malik," Improvement of Local Approximation of Trajectory Piecewise Linear Macromodels through Chebyshev Interpolating Polynomials," in the IEEE/ACM 18th Asia and South Pacific Design Automation Conference (ASP-DAC2013), Yokohama, Japan, 22-25 January 2013.

[10] M.U.Farooq, L.Xia, F.A.Hussin, A.S.Malik," Automated Model Generation of Analog Circuits Through Modified Trajectory Piecewise Linear Approach with Chebyshev Newton Interpolating Polynomials," in the 4th International Conference on Intelligent Systems, Modelling and Simulation (ISMS2013),Bangkok, Thailand, 29-31 January,2013.

APPENDICES

APPENDIX A
ANALOG CIRCUITS AS STATE SPACE

A fault-free or faulty analog circuit can be represented with a set of nonlinear differential equations shown in A-1.

$$E\dot{x} = f(x(t)) + bu(t),\\ y = Cx \tag{A-1}$$

In this appendix, a simple example of an analog fault-free circuit (figure A-1) [37] that can be cast as a set of nonlinear differential equations (A.1) is presented. The circuit has five diodes, with a capacitor and resistor connected to each node. For analog circuits, the state vector $(x(t))$ is the voltage at capacitive nodes and current through inductors. In our case, the state vector is the voltages at capacitive node and is of length 5, thus having five elements ($x(t) = [x_1(t)\ x_2(t)\ x_3(t)\ x_4(t)\ x_5(t)]$). $x_1(t)$ represents voltage at node 1, $x_2(t)$ is voltage at node 2 and so on. Consider all resistors and capacitors have unity value and nonlinear current through a diode is denoted with $g(x)$.

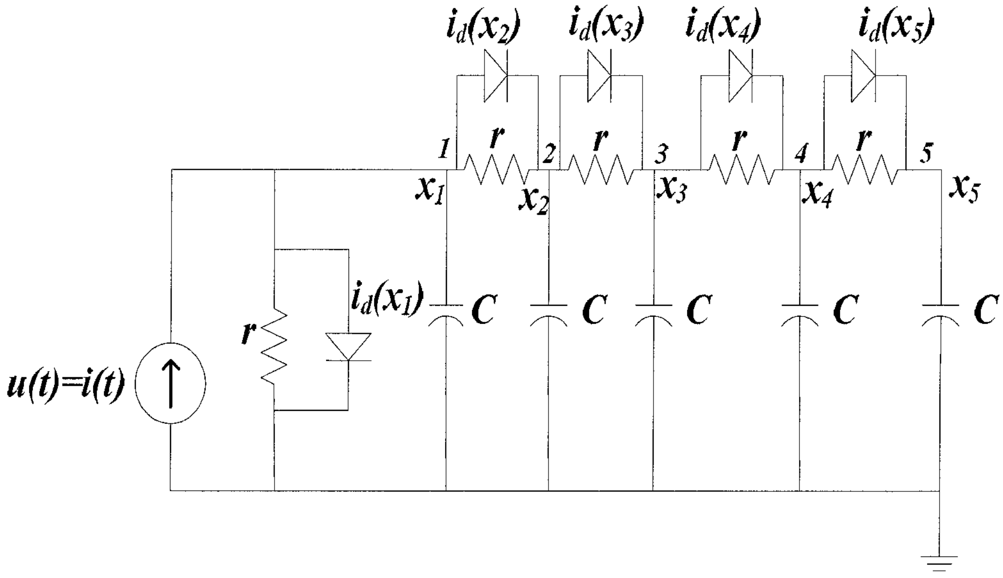


Figure A- 1: Schematic of transmission line circuit ($N=5$ nodes)

Writing Kirchhoff's Current Law (KCL) equations at node 1 to node 5 produce following set of equations:

$$\left.\begin{aligned}
&\text{KCL at node 1:}\\
&\qquad i(t)=C\frac{dx_1}{dt}+g(x_1-x_2)+g(x_1)+\frac{x_1}{r}+\frac{(x_1-x_2)}{r}\\
&\text{KCL at node 2:}\\
&\qquad g(x_1-x_2)+\frac{(x_1-x_2)}{r}=C\frac{dx_2}{dt}+g(x_2-x_3)+\frac{(x_2-x_3)}{r}\\
&\text{KCL at node 3:}\\
&\qquad g(x_2-x_3)+\frac{(x_2-x_3)}{r}=C\frac{dx_3}{dt}+g(x_3-x_4)+\frac{(x_3-x_4)}{r}\\
&\text{KCL at node 4:}\\
&\qquad g(x_3-x_4)+\frac{(x_3-x_4)}{r}=C\frac{dx_4}{dt}+g(x_4-x_5)+\frac{(x_4-x_5)}{r}\\
&\text{KCL at node 5:}\\
&\qquad g(x_4-x_5)++\frac{(x_4-x_5)}{r}=C\frac{dx_5}{dt}
\end{aligned}\right\}\text{(A-2)}$$

Re-arranging and re-writing these equations in matrix form, we get a set of equations similar to equation (A-1).

$$\frac{d}{dt}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\\x_5\end{bmatrix}=\begin{bmatrix}-g(x_1)-g(x_1-x_2)+\frac{(x_2-2x_1)}{r}\\ g(x_1-x_2)-g(x_2-x_3)+\frac{(x_1-2x_2+x_3)}{r}\\ g(x_2-x_3)-g(x_3-x_4)+\frac{(x_2-2x_3+x_4)}{r}\\ g(x_3-x_4)-g(x_4-x_5)+\frac{(x_3-2x_4+x_5)}{r}\\ g(x_4-x_5)+\frac{(x_4-x_5)}{r}\end{bmatrix}+\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix}i(t) \tag{A-3}$$

Comparing equation (A-3) with equation (A-1) we get,

$$x=\begin{bmatrix}x_1\\x_2\\x_3\\x_4\\x_5\end{bmatrix};\ f(x)=\begin{bmatrix}-g(x_1)-g(x_1-x_2)+\dfrac{(x_2-2x_1)}{r}\\ g(x_1-x_2)-g(x_2-x_3)+\dfrac{(x_1-2x_2+x_3)}{r}\\ g(x_2-x_3)-g(x_3-x_4)+\dfrac{(x_2-2x_3+x_4)}{r}\\ g(x_3-x_4)-g(x_4-x_5)+\dfrac{(x_3-2x_4+x_5)}{r}\\ g(x_4-x_5)+\dfrac{(x_4-x_5)}{r}\end{bmatrix};\ b=\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix};\ u(t)=i(t)$$

Here, x represents the state vector, *f(x)* represents the nonlinear current function through diodes, b is column vector describing the number of inputs (single input in this case) and *u(t)* is the input to circuit. Hence, analog circuit information is automatically abstracted into white-box ss model structure.

APPENDIX B

MATLAB CODE FOR AMG-CNIP ALGORITHM

B.1. Main Routine

```
%%%%%% TOP LEVEL ROUTINE %%%%%%%%%%%%%%%%%%
% This routine is the main routine for algorithm model generation,
model evaluation and automatic model conversion.

close all
clear all
clc

%%%%%% Parameters from SPICE2G6 running in SystemVision
t0=0;tf=0.01;dt=1e-5;abs_tol = 1e-6;
Isat = 1e-14;Gmin = 1e-12;vte  = 0.02586418638;
nodes = 5; c= 1;

sim_constants=struct;
sim_constants.t0=t0;
sim_constants.tf=tf;
sim_constants.dt=dt;
sim_constants.tol=abs_tol;

spice_constants=struct;
spice_constants.Isat=Isat;
spice_constants.Gmin=Gmin;
spice_constants.vte = vte;
spice_constants.nodes=nodes;
spice_constants.c = c;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% AMG_CNIP MODEL GENERATION%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[a_1d,a_2d,x1k,x2k,N_tuned,CKT_TrainOut,AMGCNIP_TrainOut,T] =
AMGCNIP_MODEL_GENERATION(spice_constants,sim_constants);

PLOT_TRAIN(CKT_TrainOut,AMGCNIP_TrainOut,T); %%% plotting trained
model outputs. Just a sanity check.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% AMG_CNIP MODEL EVALUATION  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


[AMGCNIP_EvalOut, CKT_EvalOut,T]  =
AMG_CNIP_EVALUATION(a_1d,a_2d,x1k,x2k,N_tuned,spice_constants,sim_con
stants);
PLOT_EVAL(CKT_EvalOut,AMGCNIP_EvalOut,T);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% AUTOMATIC MODEL CONVERSION %%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This routine loads polynomial coefficients from model generation
%%% routine and generate VHDL-AMS model that can be directly use as
drop-in
%%% module in system vision for performing HLM at system level
Automatic_Model_Converter('D:\Umer
Data\My_Work\Chebyshev_SPICE_Coding\AMG_CNIP\AMG_CNIP_HLM.vhdl',N_tun
ed,a_1d,a_2d)
```

B.2. AMG-CNIP Model Generation Routine

```
 function [
a_1d,a_2d,x1k,x2k,N_tuned,CKT_TrainOut,AMGCNIP_TrainOut,T] =
AMGCNIP_MODEL_GENERATION(spice_constants,sim_constants)
%AMGCNIP_MODEL_GENERATION: This routine generate AMG-CNIP model
according to  the algorithm given in chapter 3.
%Input parameters: spice_constants: to get spice parameters
%                  sim_constants: to get timing information
%Output parameters:a_1d, a_2d: CN polynomial coefficients for 1D and
2D
%                 N_tuned: Tuned CN polynomial order
%                 CKT_TrainOut, AMGCNIP_TrainOut: outputing original
%                 circuit and AMGCNIP response for ploting with
%timing information in T.

%%%%% Simulating original circuit in MATLAB to find range
%%%%% Chapter 3 -section 3.2.1.3
ckt_outputs = simulate_ckt(spice_constants,sim_constants);

%%%%% Simulating nonlinear current functions with state vector
(output voltages)
%%%%% to find range [a,b] using function ARF described in section
3.2.1.3
[a,b]= ARF(ckt_outputs,spice_constants);
range = [a,b];

 %%%%%%  Generating 1D Chebyshev - Newton Interpolating polynomial as
%%%%%%  described in section 3.2.1.1
N = 3; %% Initiaing polynomial order with some random value
a_1d = CN_poly_1D(range,spice_constants,N);

%%%%%%  Generating 2D Chebyshev - Newton Interpolating polynomial as
%%%%%%  described in section 3.2.1.2

[x1k,x2k,a_2d] = CN_poly_2D(range,spice_constants,N);

 %%%%% Simulate ss with CN polynomial coefficients (1D and 2D) and
APT tunes
%%%%% the polynomial order N as described in section 3.2.1.5
 [a_1d,a_2d,x1k,x2k,N_tuned,CKT_TrainOut,AMGCNIP_TrainOut,T] =
APT(a_1d,a_2d,x1k,x2k,range,N,spice_constants,sim_constants);

end
```

B.2.1 Simulate Circuit Routine

```
function [ckt_outputs] = simulate_ckt(spice_constants,sim_constants )
% This routine simulate original circuit in ss form with original
nonlinear diode current function to obtain original circuit outputs.

% Initialization of different variables for transient simulation
```

```
t0 = sim_constants.t0;
tf = sim_constants.tf;
dt = sim_constants.dt;
T  = t0:dt:tf;
nodes        = spice_constants.nodes;
IC           = zeros(nodes,1);
ckt_outputs = zeros(1:length(T)-1,nodes);
flag = 1; % using the ode routine for training

for i=1:length(T)-1
    % selecting time step for transient simulation
    Tspan           =[T(i) T(i+1)];
    % solving differential equations using MATLAB ode function
    [Time V]
=ode15s(@(t,v)nltxline(t,v,spice_constants,flag),Tspan,IC);
    %storing only last output value of integration step at Time=tf
    IC                = V(end,:);
    ckt_outputs(i,:)  = V(end,:);

end
end
```

B.2.1.1 Nltxline Routine

```
function dv = nltxline(t,v,spice_constants,flag)
% This routine simulates original circuit and is also used during
model
% evaluation. Flag test either to calculates training input or
evaluation
% input.

c= spice_constants.c;
if flag ==1
ut = compute_train_input(t);
else if flag == 0
        ut = compute_eval_input(t);
    end
end
dv = zeros(5,1);

id1 = calculate_id(v(1),spice_constants);
id2 = calculate_id(v(1)-v(2),spice_constants);
id3 = calculate_id(v(2)-v(3),spice_constants);
id4 = calculate_id(v(3)-v(4),spice_constants);
id5 = calculate_id(v(4)-v(5),spice_constants);


dv(1)  =  (-id1-id2+v(2)-(2*v(1))+ ut)/c;
dv(2)  =  (v(1)-(2*v(2))+v(3)+id2-id3)/c;
dv(3)  =  (v(2)-(2*v(3))+v(4)+id3-id4)/c;
dv(4)  =  (v(3)-(2*v(4))+v(5)+id4-id5)/c;
dv(5)  =  (v(4)-v(5)+id5)/c;

dv = dv(:);
end
```

```
function id = calculate_id(vd,spice_constants)
% This routine calculate orginal nonlinear current function at every
time
% point for original circuit simulation and this is what is replace
by CN
% polynimial in ss by AMG-CNIP algorithm

Isat = spice_constants.Isat;
Gmin = spice_constants.Gmin;
vte  = spice_constants.vte;

evd = exp(vd/vte);
id  = Isat*(evd-1) + (Gmin*vd);

end
```

B.2.2 ARF Routine

```
function [a,b ] = ARF( ckt_outputs,spice_constants )
%ARF: This function find range [a,b] to calculate nodes for Chebyshev
%polynomials.

% extracting voltages from circuit outputs
vd1 = ckt_outputs(:,1);
vd2 = ckt_outputs(:,2);
vd3 = ckt_outputs(:,3);
vd4 = ckt_outputs(:,4);
vd5 = ckt_outputs(:,5);

n = length(vd1);

% simulating diode nonlinear current functions with corresponding
voltages
for i=1:n
fx1(i) = Diode_nl_func(spice_constants,vd1(i));
fx2(i) = Diode_nl_func(spice_constants,vd1(i)-vd2(i));
fx3(i) = Diode_nl_func(spice_constants,vd2(i)-vd3(i));
fx4(i) = Diode_nl_func(spice_constants,vd3(i)-vd4(i));
fx5(i) = Diode_nl_func(spice_constants,vd4(i)-vd5(i));
end

% find minimum value of each nonlinear function. also find maximum
value
% from all nonlinear funtions. it is done to cover range for all
nonlinear functions so that we do not need to calculate different CN
polynomial for each nonlinear function.
a1 = min(fx1); b1=max(fx1);
a2 = min(fx2); b2=max(fx2);
a3 = min(fx3); b3=max(fx3);
a4 = min(fx4); b4=max(fx4);
a5 = min(fx5); b5=max(fx5);

aa = [a1,a2,a3,a4,a5];
bb = [b1,b2,b3,b4,b5];
a = min(aa);% minmum from all nonlinear functions
b = max(bb);% maximum from all nonlinear functions
```

```
end

function [ fx ] = Diode_nl_func( spice_constants,vd)

evd_t = exp(vd/spice_constants.vte);
fx = (spice_constants.Isat*(evd_t-1)+ (spice_constants.Gmin*vd));

end
```

B.2.3 1D CN Polynomial Routine

```
function [ a_1d] = CN_poly_1D( range,spice_constants,n)
%CN_POLY_1D: This function calculates CN polynomial for 1D components
i.e.for the components whose one end is grounded and other end is
connected with a node.
% Input parameters
% range : Used to calculate Chebyshev nodes
% N : polynomial order
% spice_constants : parameters to use durign calculation of Chebyshev
nodes
% Outpt parameters
% a_1d : 1D CN polynomial coefficients calculated using 1D Newton
Divivded Difference method

% initializing range [a,b] for calculating Chebyshev nodes
a = range(1); b = range(2);

% Three-step procedure for obtaining CN polynomial
for i=1:n
% Step1: Obtaining input nodes using Cos formula as descrubed in
section 3.2.1.2
xk(i) = ((a+b)/2)+ ((b-a)/2)*cos((( (2*n)+1-(2*i))*pi)/(2*n+2));
evd = exp(xk(i)/spice_constants.vte);
% Step 2: obtaining output nodes using nonlinear function from
SPICE2G6
yk(i) =(spice_constants.Isat*(evd-1) + (spice_constants.Gmin*xk(i)))
;
end
%step3: calculating CN polynomial coefficients using NDD method from
set of
%interpilation points calculated in step 2.
a_1d = newton1d(xk,yk);

end
```

B.2.3.1 1D NDD Routine

```
function [n,DD] = newton1d(x,y)
%Input : x = [x0 x1 ... xN]
% y = [y0 y1 ... yN]
%Output: n = Newton polynomial coefficients of degree N


N = length(x)-1;
```

```
DD = zeros(N + 1,N + 1);
DD(1:N + 1,1) = y';
for k = 2:N + 1
for m = 1: N + 2 - k %Divided Difference Table
DD(m,k) = (DD(m + 1,k - 1) - DD(m,k - 1))/(x(m + k - 1)- x(m));
end
end
a = DD(1,:); %Eq.(3.2.6) in book "Applied numerical methods using
MATLAB"
n = a(N+1); %Begin with Eq.(3.2.7)
for k = N:-1:1 %Eq.(3.2.7)
n = [n a(k)] - [0 n*x(k)]; %n(x)*(x - x(k - 1))+a_k - 1
end
```

B.2.4 2D CN Polynomial Routine

```
function [ x1k,x2k,a_2d ] = CN_poly_2D( range,spice_constants,n )
%CN_POLY_2D: Generates 2D CN polynomial for components both sides
connected
%with different nodes in circuit
%  Input : range for calculating Chebyshev nodes
%          spice_constants : to be used in nonlinear function
%          n : order of polynomial and number of interpolating points
%  Output : x1k, x2k: interpolation points for 2D CN polynomial
%           a_2d : 2D CN polynomial coefficients

a = range(1); b = range(2);% initilizing range [a,b] from ARF
% Obtaining Chebyshev nodes for 2D formulation, x1k represents first
% variable and x2k represents 2nd variable for formulation
for i=1:n
x1k(i) = ((a+b)/2)+ ((b-a)/2)*cos((( (2*n)+1-(2*i))*pi)/(2*n+2));
x2k(i) = ((a+b)/2)+ ((b-a)/2)*cos((( (2*n)+1-(2*i))*pi)/(2*n+2));
end
% obtaining output points using nonlinear function and input
interpolation
% points calculated above
for i=1:n
    for j=1:n
        evd      = exp((x1k(i)-x2k(j))/spice_constants.vte);
        fxk(j,i) =(spice_constants.Isat*(evd-1) +
(spice_constants.Gmin*(x1k(i)-x2k(j)))) ;
    end
end
 % obtaining coefficients for 2D CN polynomial using 2D NDD method
a_2d = newton2d(x1k,x2k,fxk);% 2D coefficients

end
```

B.2.4.1. 2D NDD Routine

```
function A = newton2d(x,y,g)

%% 2D method mathematical details are taken from the paper
%% D. N. Varsamis and N. P. Karampetakis, "On the Newton Multivariate
%% Polynomial Interpolation with Applications," in 7th International
%% Workshop on Multidimensional (nD) Systems (nDs), 2011, vol. 0, pp.
1-8.

n = length(x);
DD = zeros(n,n,n);
%Creating Zero order table
for i=1:n
    for j=1:n
        if j > n+1-i
            DD(j,i,1) =0;
        else
            DD(j,i,1) = g(j,i);
        end
    end
end

% Creating k=2 to n order tables
for k=2:n
for i=1:n
    for j=1:n
        if j>n+1-i
            DD(j,i,k)=0;
        else
            DD(j,i,k) = DD(j,i,k-1); % this is assignment of old
table values in new tables, values that passed on directly without
any calculation
                if j<= k-1 && i > k-1 % these are three conditions to
calcuate enteries in each table based on i,j,k indecies
                    DD(j,i,k) = (DD(j,i,k-1) - DD(j,i-1,k-1)) / (x(i)
- x(i-k+1) );
                else
                    if i<= k-1 && j>k-1
                      DD(j,i,k) =  (DD(j,i,k-1)- DD(j-1,i,k-1)) /
(y(j)-y(j-k+1));
                    else
                        if j>k-1 && (i+j-1) <= n
                            DD(j,i,k) = (DD(j,i,k-1) + DD(j-1,i-1,k-
1) - DD(j-1,i,k-1) - DD(j,i-1,k-1)) /( (x(i)-x(i-k+1)) * (y(j) - y(j-
k+1)) );


                        end
                    end
                end


        end
    end
end
end
 A = DD(:,:,n); % Returning last table enteries as polynomial
coefficients
%end
```

B.2.5 APT Routine

```
function [a_1d,a_2d,x1k,x2k,N_tuned,CKT_TrainOut,AMGCNIP_TrainOut,T]
= APT(a_1d,a_2d,x1k,x2k,range,N,spice_constants,sim_constants)
```

```
%APT: This routine tunes the order N of polynomial based on output
error.

% Initialization of different variables for transient simulation
t0 = sim_constants.t0;
tf = sim_constants.tf;
dt = sim_constants.dt;
T  = t0:dt:tf;
nodes        = spice_constants.nodes;
IC           = zeros(nodes,1);
IC_cheb      = zeros(nodes,1);
ckt_outputs = zeros(1:length(T)-1,nodes);
ss_cheb_outputs = zeros(1:length(T)-1,nodes);
flag = 1; % using ode routines for training inputs
N_tuned = N;% Initializing N_tuned with N

% Main loop, simulating original circuit and ss with CN polynomial at
same
% time to calculate output error at each time step and tune
polynomial
% order
for i=1:length(T)-1
    Tspan           =[T(i) T(i+1)];
    [Time V]
=ode15s(@(t,v)nltxline(t,v,spice_constants,flag),Tspan,IC);
    [T_cheb V_cheb]
=ode15s(@(t_cheb_train,v_cheb_train)SS_CN_poly(t_cheb_train,v_cheb_tr
ain,a_1d,a_2d,x1k,x2k,spice_constants,flag),Tspan,IC_cheb);%
Approximation of training trajectory through Chebyshev-Newton
polynomials used in state space
    IC              = V(end,:);
    IC_cheb         = V_cheb(end,:);
    ckt_outputs(i,:) = V(end,:);
    ss_cheb_outputs(i,:)= V_cheb(end,:);

    %%%%%%%% polynomial tuner check to increase order N based on
condition
    %%%%%%%% below
    if abs(ckt_outputs(i,1) - ss_cheb_outputs(i,1)) >
sim_constants.tol
        N_tuned = N_tuned + 1;
        a_1d            = CN_poly_1D(range,spice_constants,N_tuned);
        [x1k,x2k,a_2d] = CN_poly_2D(range,spice_constants,N_tuned);
    end
end

CKT_TrainOut      = ckt_outputs;
AMGCNIP_TrainOut = ss_cheb_outputs;

end
```

B.2.5.1. CN Polynomial SS Simulation Routine

```
 function [ dv ] =
SS_CN_poly(t,v,a_1d,a_2d,x1k,x2k,spice_constants,flag)
%SS_CN_POLY_TRAIN : In this routine we replace also use 1D
coefficients for
%2D functions by taking difference of two voltages and make it single
```

```
%voltage. Complete code for 2D ss calculation is also given in
comments
%below and output same response.

c= spice_constants.c;
dv = zeros(5,1);

if flag ==1
ut = compute_train_input(t);
else if flag == 0
        ut = compute_eval_input(t);
    end
end


P_1d   = polyval(a_1d,v(1));
v12 = v(1) - v(2);
P1_2d  = polyval(a_1d,v12);
v23 = v(2) - v(3);
P2_2d  = polyval(a_1d,v23);
v34 = v(3) - v(4);
P3_2d  = polyval(a_1d,v34);
v45 = v(4) - v(5);
P4_2d  = polyval(a_1d,v45);

dv(1)= (-P_1d  - P1_2d+ v(2) - (2*v(1)) + ut)/c;
dv(2)= (P1_2d  - P2_2d + v(1) - (2*v(2)) + v(3))/c;
dv(3)= (P2_2d  - P3_2d + v(2) - (2*v(3)) + v(4))/c;
dv(4)= (P3_2d  - P4_2d + v(3) - (2*v(4)) + v(5))/c;
dv(5)= (P4_2d + v(4) - v(5))/c;
%
% %initializing variables for calcuation of newton 2D polynomial
% n=length(x1k);% getting the order of the system
% X1_P1=zeros(n,1);X2_P1=zeros(n,1);
% X1_P2=zeros(n,1);X2_P2=zeros(n,1);
% X1_P3=zeros(n,1);X2_P3=zeros(n,1);
% X1_P4=zeros(n,1);X2_P4=zeros(n,1);
%
% X1_P1(1)=1;X2_P1(1)=1;
% X1_P2(1)=1;X2_P2(1)=1;
% X1_P3(1)=1;X2_P3(1)=1;
% X1_P4(1)=1;X2_P4(1)=1;
%
% c1_P1=1;c2_P1=1;
% c1_P2=1;c2_P2=1;
% c1_P3=1;c2_P3=1;
% c1_P4=1;c2_P4=1;
%
%
% %% To generate 2d newton polynomials two matrices X1 and X2 are
built at run time, these are generated here recursively using for
loop.
% % In each chunk of code below, separate polynomials are generate
for each function. Each polynomial depends on two node voltages and
in each chunk of code node voltages are changed. Therefore need to
replicate this code the no. of polynomials we want to generate.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% for i=2:n
%      c1_P1 = (v(1)-x1k(i-1))* c1_P1;
%      X1_P1(i)=c1_P1;
% end
%
% for i=2:n
%      c2_P1 = (v(2)-x2k(i-1))* c2_P1;
%      X2_P1(i)=c2_P1;
% end
%
% P1_2d = (X2_P1'*a_2d)*X1_P1;% Calculation of Newton 2D polynomial
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% for i=2:n
%      c1_P2 = (v(2)-x1k(i-1))* c1_P2;
%      X1_P2(i)=c1_P2;
% end
%
% for i=2:n
%      c2_P2 = (v(3)-x2k(i-1))* c2_P2;
%      X2_P2(i)=c2_P2;
% end
%
% P2_2d = (X2_P2'*a_2d)*X1_P2;
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% for i=2:n
%      c1_P3 = (v(3)-x1k(i-1))* c1_P3;
%      X1_P3(i)=c1_P3;
% end
%
% for i=2:n
%      c2_P3 = (v(4)-x2k(i-1))* c2_P3;
%      X2_P3(i)=c2_P3;
% end
%
% P3_2d = (X2_P3'*a_2d)*X1_P3;
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% for i=2:n
%      c1_P4 = (v(4)-x1k(i-1))* c1_P4;
%      X1_P4(i)=c1_P4;
% end
%
% for i=2:n
%      c2_P4 = (v(5)-x2k(i-1))* c2_P4;
%      X2_P4(i)=c2_P4;
% end
%
% P4_2d = (X2_P4'*a_2d)*X1_P4;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% finally assiging polynomial values to nodes. These values are the
%%% output of current integration step.
% dv(1)= -P_1d  - P1_2d+ ut;
% dv(2)= P1_2d  - P2_2d;
% dv(3)= P2_2d   - P3_2d;
% dv(4)= P3_2d   - P4_2d;
% dv(5)= P4_2d;

dv= dv(:);

end
```

B.3. AMG-CNIP Model Evaluation Routine

```
function [ AMG_CNIP_EvalOut, Ckt_EvalOut,T ] =
AMG_CNIP_EVALUATION(a_1d,a_2d,x1k,x2k,N,spice_constants,sim_constants
)
%AMG_CNIP_EVALUATION: This routine loads coefficients from model
generation module and evaluate the model with evaluation input.

%  Initialization of different variables for transient simulation
t0 = sim_constants.t0;
tf = sim_constants.tf;
dt = sim_constants.dt;
T  = t0:dt:tf;
nodes        = spice_constants.nodes;
IC           = zeros(nodes,1);
IC_cheb      = zeros(nodes,1);
ckt_outputs  = zeros(1:length(T)-1,nodes);
ss_cheb_outputs = zeros(1:length(T)-1,nodes);
flag = 0; % using ode routine for evaluation inputs


for i=1:length(T)-1
    Tspan           =[T(i) T(i+1)];
    [Time V]
=ode15s(@(t,v)nltxline(t,v,spice_constants,flag),Tspan,IC);
    [T_cheb V_cheb]
=ode15s(@(t_cheb_train,v_cheb_train)SS_CN_poly(t_cheb_train,v_cheb_tr
ain,a_1d,a_2d,x1k,x2k,spice_constants,flag),Tspan,IC_cheb);%
Approximation of training trajectory through Chebyshev-Newton
polynomials used in state space
    IC              = V(end,:);
    IC_cheb         = V_cheb(end,:);
    ckt_outputs(i,:)  = V(end,:);
    ss_cheb_outputs(i,:)= V_cheb(end,:);

end

AMG_CNIP_EvalOut = ss_cheb_outputs;
Ckt_EvalOut      = ckt_outputs;

end
```

B.4. Automatic Model Conversion (AMC) Routine

```
 function Automatic_Model_Converter(filename,n,a_1d,a_2d)
% open a file for writing signals
%fid = fopen('d:\Hdl_generation\vhd_g\vhd_g.vhd','w');

fid = fopen(filename,'w');

fprintf(fid,'%s\n','library ieee;');
fprintf(fid,'%s\n','use ieee.math_real.all;');
fprintf(fid,'%s\n','use IEEE.electrical_systems.all;');
fprintf(fid,'\n');
fprintf(fid,'%s\n','entity AMGCNIP_HLM is ');
fprintf(fid,'%s\n','port (');
fprintf(fid,'%s\n','terminal u_t,v1_hlm,v2_hlm,v3_hlm,v4_hlm,v5_hlm,g
: electrical);');
fprintf(fid,'%s\n','end entity AMGCNIP_HLM;');
fprintf(fid,'\n');
fprintf(fid,'%s\n','architecture ideal of AMGCNIP_HLM is');
fprintf(fid,'%s\n',' -- Declare Branch Quantities');
fprintf(fid,'%s\n','  quantity vi across i_in     through u_t     to
g;');
fprintf(fid,'%s\n','  quantity v1 across i_out1   through v1_hlm  to
g;');
fprintf(fid,'%s\n','  quantity v2 across i_out2   through v2_hlm  to
g;');
fprintf(fid,'%s\n','  quantity v3 across i_out3   through v3_hlm  to
g;');
fprintf(fid,'%s\n','  quantity v4 across i_out4   through v4_hlm  to
g;');
fprintf(fid,'%s\n','  quantity v5 across i_out5   through v5_hlm  to
g;');
fprintf(fid,'\n');

for i=1:5
fprintf(fid,'\n quantity temp%d : real;',i);
end
fprintf(fid,'\n');
for i=1:n
fprintf(fid,'\n constant a_1d_%d : real :=%3.30g;',i,a_1d(i));
end

fprintf(fid,'\n');
fprintf(fid,'\n');

fprintf(fid,'%s\n','begin');
fprintf(fid,'%s\n','vi   ==1.0*i_in;');
fprintf(fid,'\n');

%%%%%%%%%%%% temp 1 code
fprintf(fid,'%s','temp1 == -(');

for i=1:n-1
fprintf(fid,'(a_1d_%d*v1**%d)',i,n-i);
fprintf(fid,'+');
end
fprintf(fid,'a_1d_%d)',n);

fprintf(fid,'-(');
for i=1:n-1
fprintf(fid,'(a_1d_%d*(v1-v2)**%d)',i,n-i);
fprintf(fid,'+');
end
fprintf(fid,'a_1d_%d)+ v1 - (v1-v2)+i_in;',n);
fprintf(fid,'\n');
fprintf(fid,'%s\n','v1 == temp1''integ;');

%%%%%%%%%%%% temp 2 code
fprintf(fid,'\n');

fprintf(fid,'%s','temp2 == (');

for i=1:n-1
fprintf(fid,'(a_1d_%d*(v1-v2)**%d)',i,n-i);
fprintf(fid,'+');
end
fprintf(fid,'a_1d_%d)',n);

fprintf(fid,'-(');

for i=1:n-1
fprintf(fid,'(a_1d_%d*(v2-v3)**%d)',i,n-i);
fprintf(fid,'+');
end
fprintf(fid,'a_1d_%d) + (v1-v2) - (v2-v3);',n);

fprintf(fid,'\n');
fprintf(fid,'%s\n','v2 == temp2''integ;');

%%%%%%%%%%%% temp 3 code

fprintf(fid,'\n');

fprintf(fid,'%s','temp3 == (');

for i=1:n-1
fprintf(fid,'(a_1d_%d*(v2-v3)**%d)',i,n-i);
fprintf(fid,'+');
end
fprintf(fid,'a_1d_%d)',n);

fprintf(fid,'-(');

for i=1:n-1
fprintf(fid,'(a_1d_%d*(v3-v4)**%d)',i,n-i);
fprintf(fid,'+');
end
fprintf(fid,'a_1d_%d) + (v2-v3) - (v3-v4);',n);

fprintf(fid,'\n');
fprintf(fid,'%s\n','v3 == temp3''integ;');
```

```
%%%%%%%%%%%% temp 4 code

fprintf(fid,'\n');

fprintf(fid,'%s','temp4 == (');

for i=1:n-1
fprintf(fid,'(a_1d_%d*(v3-v4)**%d)',i,n-i);
fprintf(fid,'+');
end
fprintf(fid,'a_1d_%d)',n);

fprintf(fid,'-(');

for i=1:n-1
fprintf(fid,'(a_1d_%d*(v4-v5)**%d)',i,n-i);
fprintf(fid,'+');
end
fprintf(fid,'a_1d_%d) + (v3-v4) - (v4-v5);',n);

fprintf(fid,'\n');
fprintf(fid,'%s\n','v4 == temp4''integ;');

%%%%%%%%%%%% temp 5 code

fprintf(fid,'\n');

fprintf(fid,'%s','temp5 == (');

for i=1:n-1
fprintf(fid,'(a_1d_%d*(v4-v5)**%d)',i,n-i);
fprintf(fid,'+');
end
fprintf(fid,'a_1d_%d)+ (v4-v5);',n);


fprintf(fid,'\n');
fprintf(fid,'%s\n','v5 == temp5''integ;');
fprintf(fid,'\n');

fprintf(fid,'%s\n','end architecture ideal;');
```

B.5. Plotting Routines

```
function PLOT_TRAIN(CKT_TrainOut,AMGCNIP_TrainOut,T )
%PLOT_TRAIN : Plot training circuit output and AMG-CNIP trained model
%outputs
figure
plot(T(1:end-1),CKT_TrainOut(:,1),'o-',T(1:end-
1),AMGCNIP_TrainOut(:,1),'.-')
```

```
legend('CIRCUIT-TRAIN-OUTPUT (node 1)','AMGCNIP-TRAIN-OUTPUT (node
1)')
grid

figure
plot(T(1:end-1),CKT_TrainOut(:,2),'o-',T(1:end-
1),AMGCNIP_TrainOut(:,2),'.-')
legend('CIRCUIT-TRAIN-OUTPUT (node 2)','AMGCNIP-TRAIN-OUTPUT (node
2)')
grid

figure
plot(T(1:end-1),CKT_TrainOut(:,3),'o-',T(1:end-
1),AMGCNIP_TrainOut(:,3),'.-')
legend('CIRCUIT-TRAIN-OUTPUT (node 3)','AMGCNIP-TRAIN-OUTPUT (node
3)')
grid

figure
plot(T(1:end-1),CKT_TrainOut(:,4),'o-',T(1:end-
1),AMGCNIP_TrainOut(:,4),'.-')
legend('CIRCUIT-TRAIN-OUTPUT (node 4)','AMGCNIP-TRAIN-OUTPUT (node
4)')
grid

figure
plot(T(1:end-1),CKT_TrainOut(:,5),'o-',T(1:end-
1),AMGCNIP_TrainOut(:,5),'.-')
legend('CIRCUIT-TRAIN-OUTPUT (node 5)','AMGCNIP-TRAIN-OUTPUT (node
5)')
grid

end


function PLOT_EVAL(CKT_EvalOut,AMGCNIP_EvalOut,T )
%PLOT_TRAIN :Plot evaluation circuit outputs and AMG-CNIP model
evaluation outputs

figure
plot(T(1:end-1),CKT_EvalOut(:,1),'o-',T(1:end-
1),AMGCNIP_EvalOut(:,1),'.-')
legend('CIRCUIT-EVAL-OUTPUT (node 1)','AMGCNIP-EVAL-OUTPUT (node 1)')
grid

figure
plot(T(1:end-1),CKT_EvalOut(:,2),'o-',T(1:end-
1),AMGCNIP_EvalOut(:,2),'.-')
legend('CIRCUIT-EVAL-OUTPUT (node 2)','AMGCNIP-EVAL-OUTPUT (node 2)')
grid

figure
plot(T(1:end-1),CKT_EvalOut(:,3),'o-',T(1:end-
1),AMGCNIP_EvalOut(:,3),'.-')
legend('CIRCUIT-EVAL-OUTPUT (node 3)','AMGCNIP-EVAL-OUTPUT (node 3)')
grid
```

```
figure
plot(T(1:end-1),CKT_EvalOut(:,4),'o-',T(1:end-
1),AMGCNIP_EvalOut(:,4),'.-')
legend('CIRCUIT-EVAL-OUTPUT (node 4)','AMGCNIP-EVAL-OUTPUT (node 4)')
grid

figure
plot(T(1:end-1),CKT_EvalOut(:,5),'o-',T(1:end-
1),AMGCNIP_EvalOut(:,5),'.-')
legend('CIRCUIT-EVAL-OUTPUT (node 5)','AMGCNIP-EVAL-OUTPUT (node 5)')
grid

end
```