Abstract

LUNSFORD III PHILIP J. The Frequency Domain Behavioral Modeling and Simulation of Nonlinear Analog Circuits and Systems. (Under the direction of Michael B. Steer.)

A new technique for the frequency-domain behavioral modeling and simulation of nonautonomous nonlinear analog subsystems is presented. This technique extracts values of the Volterra nonlinear transfer functions and stores these values in binary files. Using these files, the modeled substem can be simulated for an arbitrary periodic input expressed as a finite sum of sines and cosines. Furthermore, the extraction can be based on any circuit simulator that is capable of steady state simulation. Thus a large system can be divided into smaller subsystems, each of which is characterized by circuit level simulations or lab measurements. The total system can then be simulated using the subsystem characterization stored as tables in binary files.

Using known ideal nonlinear circuits, the method extraction technique was tested for nonlinearities up to seventh order. The Volterra nonlinear transfer functions of an equalizing circuit were extracted from the results of transient simulations which utilized the shooting method. The resulting tables were then used to simulate an arbitrary waveform and the results closely matched the equivalent time domain circuit simulation.
The Frequency Domain Behavioral Modeling and Simulation of Nonlinear Analog Circuits and Systems

by

Philip J. Lunsford, II

A thesis submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

Department of Electrical and Computer Engineering

Raleigh, NC

1993

Approved By:

Co-Chairman/Advisory Committee

Co-Chairman/Advisory Committee
Acknowledgements

I wish to thank the following people for the support they have given me during my time at North Carolina State University.

To Dr. Michael Steer for his guidance, support, and technical direction.

To IBM for supporting me and this research.

To everyone in Daniels 333 and 336 for their outlook, work ethic, and sense of humor.

To my wife, Anne, for providing moral support and the correct perspective, and to Jean Elizabeth Lunsford, my daughter, for giving me incentive to no longer be a graduate student.
## Table of Contents

List of Figures  
List of Tables  

### 1 Introduction
1.1 Motivations and Objectives of This Study ........................................... 1  
1.2 Overview .................................................. 2  
1.3 Summary of Original Contributions .............................................. 2  
1.3.1 Published Work ............................................. 2  
1.3.2 Unpublished Work ............................................. 2  

### 2 Review of Steady-State Behavioral Modeling and Simulation  
2.1 Introduction .................................................. 3  
2.2 Definitions & Issues ............................................ 3  
2.2.1 Time Domain vs Frequency Domain .................................. 4  
2.3 Modeling Techniques ............................................. 4  
2.3.1 Time Domain Modeling ........................................... 5  
2.3.2 Generalized Power Series Modeling ................................... 5  
2.3.3 Power Series Ratios ......................................... 6  
2.3.4 Volterra Series ............................................... 8  
2.4 Simulation ................................................... 10  
2.4.1 Shooting Methods .............................................. 10  
2.4.2 Harmonic Balance ............................................. 10  
2.4.3 Method of Nonlinear Currents .................................... 12  
2.5 Behavioral Model Extraction ........................................... 12  
2.5.1 Hand Written in Modeling Language .................................. 13  
2.5.2 Error Minimization ............................................. 13  
2.5.3 Maximum Likelihood Estimator .................................... 14  
2.5.4 Volterra Series Model Extraction ................................... 15  
2.6 Conclusion ................................................... 17  

### 3 Behavioral Modeling and Simulation  
3.1 Introduction ................................................... 18  
3.2 Bivariate Generalized Power Series ........................................... 19  
3.3 The Relationship between Volterra Series and Power Series ............. 22  
3.3.1 Univariate Case .............................................. 22  
3.3.2 Bivariate Case ............................................... 29  
3.3.3 Discussion .................................................. 34  
3.4 Volterra Harmonic Balance Behavioral Simulation  
3.4.1 Univariate Volterra and Power Series Simulation ..................... 40  
3.4.2 Bivariate Volterra and Power Series Simulation ..................... 40  
3.5 Conclusion ................................................... 50
4 Volterra Behavioral Model Extraction
  4.1 Introduction ........................................... 52
  4.2 Extraction procedure ................................ 52
    4.2.1 Output Decomposition .............................. 54
    4.2.2 Transfer Function Calculation ....................... 55
    4.2.3 Algorithm Flow Description ......................... 62
    4.2.4 Summary ............................................ 75
  4.3 Conclusion ............................................. 79

5 Verification
  5.1 Introduction ........................................... 80
  5.2 Bivariate Generalized Power Series – Ring Diode Mixer .... 80
  5.3 Signal Flow Analysis for Bivariate Volterra Series .......... 84
  5.4 New Volterra Extraction ................................ 87
    5.4.1 Linear RC Filter .................................. 87
    5.4.2 Power Series Device 7th order ...................... 89
  5.5 Four Megabit Token Ring Equalizer Circuit ................ 92
    5.5.1 Problems ......................................... 107
    5.5.2 Discussion ......................................... 107

6 Conclusion
  6.1 Summary ................................................ 112
  6.2 Suggestions for Further Work ............................ 113

References ................................................. 114

Appendices ............................................... 133
  A. Algorithm for Bivariate GPSSA Indexing .................. 133
  B. Properties of Volterra Nonlinear Transfer Functions .... 138
    B.1 Introduction ......................................... 138
    B.2 Basic Formula ....................................... 138
    B.3 Symmetry of Kernels and Transfer Functions ........... 139
    B.4 Negative Argument of a Transfer Function .............. 140
    B.5 Balanced Transfer Function Arguments that Add to Zero 140
    B.6 Conclusion .......................................... 141
  C. MAPLE Code for Ring Diode Mixer ........................ 142
    C.1 Formulation ......................................... 142
    C.2 Solution ............................................. 144
  D. Instructions for Running Extraction Code ................ 153
    D.1 Overview ............................................. 153
    D.2 Customizing for a Simulator .......................... 153
    D.3 Creating *.status file ................................ 154
    D.4 Initializing .......................................... 155
    D.5 Creating Simulation Files .............................. 155
    D.6 Running Simulations ................................... 156
    D.7 Extracting Values .................................... 156
    D.8 Calculating from Tables ................................ 156

iv
## List of Figures

2.1 Simplified spectrum for a mixer circuit \((f_1 = 2f_2 - f_3)\) .................................. 8
2.2 Volterra Analysis ................................................................. 11
2.3 Linear–Nonlinear Partitioning of Circuit .................................. 12

3.1 Univariate GPSA ................................................................. 36
3.2 Volterra Analysis ................................................................. 37
3.3 Univariate Equivalency ........................................................ 38
3.4 Bivariate Equivalency .......................................................... 39
3.5 Volta–Voltage Black Box Model ............................................. 41

4.1 Overview of extraction procedure .......................................... 53
4.2 Overview of initialization procedure ....................................... 63
4.3 The input file for low–pass RC filter ...................................... 64
4.4 Flow chart for determining simulation conditions ....................... 66
4.5 Flow chart for determining indeterminant conditions .................. 69
4.6 Flow chart for determining additional required runs .................... 71
4.7 Overview of calculation procedure ......................................... 76
4.8 Calculation procedure detail .................................................. 77
4.9 Procedure for multiple runs with the same input frequencies ........ 78

5.1 Ring Diode Mixer ................................................................. 81
5.2 Conversion Gain ................................................................. 83
5.3 Full MESFET Circuit ............................................................ 84
5.4 Signal Flow Graph ............................................................... 86
5.5 Two Tone Results ............................................................... 88
5.6 Schematic of simple RC filter .................................................. 89
5.7 The results for \(H_0, H_1, \text{and} H_2\) of a low–pass RC filter ............ 90
5.8 Sample of results for \(H_3\) of low–pass RC filter ......................... 91
5.9 Scattergram of the extracted value by order for 7th order system .... 93
5.10 Schematic of equalizer circuit ............................................... 95
5.11 Equalizer circuit at 4MHz .................................................... 96
5.12 Equalizer circuit at 8 MHz ................................................... 97
5.13 Simulated vs Predicted equalizer circuit at 500mVT4 MHz input .... 102
5.14 Simulated vs Predicted equalizer circuit at 500mVT8 MHz input .... 103
5.15 Simulated vs Predicted equalizer circuit at 500mVT4 MHz input .... 104
5.16 Simulated vs Predicted equalizer circuit at 500mVT8 MHz input .... 105
5.17 Plot of sample output predicted by extracted Volterra model for arbitrary input with max amplitude ........................................ 108
5.18 Plot of sample output predicted by extracted Volterra model for arbitrary input with max amplitude 10 percent over ....................... 109
5.19 Plot of sample output predicted by extracted Volterra model for pulse train input .......................................................... 110
5.20 Plot of sample AC output predicted by extracted Volterra model for pulse train input .......................................................... 111

A.1 Algorithm for indexing calculation ........................................ 133
List of Tables

2.1 Partial listing of IPD’s ........................................ 7
3.1 Relationship between IPD and arguments to Volterra nonlinear transfer
function for $\alpha = 0$ ........................................ 46
3.2 Example of relationship between IPD and arguments to Volterra nonlinear
transfer function for nonzero $\alpha$ .......................... 47
4.1 Simple FIPD table ........................................ 57
4.2 Runs for 1 Hz and 2 Hz ..................................... 58
4.3 List of runs with no peak minimization\(\)no additional runs\(\)no time shift\(\)
and no amp adjustment. ...................................... 67
4.4 List of runs with peak minimization\(\)no additional runs\(\)no time shift\(\)
and no amp adjustment. ...................................... 68
4.5 List of runs with peak minimization\(\)additional runs\(\)no time shift\(\)and
no amp adjustment. ............................................ 70
4.6 List of runs with peak minimization\(\)additional runs\(\)time shift\(\)and no
amp adjustment. .............................................. 72
4.7 List of runs with peak minimization\(\)additional runs\(\)time shift\(\)and amp
adjustment. ....................................................... 74
5.1 Element Values ............................................. 85
5.2 Bivariate Coefficients .................................... 85
5.3 General output format for $n$th order $H$’s ................. 89
5.4 Number of points calculated for each $H$ for 7th order system ... 92
5.5 Lower Bound of the Number of Simulations ................. 98
5.6 Number of Simulations ................................... 99
5.7 Number of Points extracted for $H$ .......................... 99
5.8 Sample of output phasors predicted by extracted Volterra model for an
input of a pulse train with truncated harmonics ................ 100
5.9 Sample of output phasors predicted by extracted Volterra model for ar-
bitrary input ..................................................... 106
List of Symbols

! factorial
\( \cdot \) complex conjugate
\( \cdot _k \) complex conjugate for \( n_k < 0 \) no operation otherwise
\( \cdot _k^\dagger \) complex conjugate for \( n_k \geq 0 \) no operation otherwise
\( A, a, b, c, d \) Coefficients of real or complex numbers
\( B \) memoryless bivariate function
\( C \) intermediate complex variable
\( C_i \) capacitance value of the \( i \)th component
\( E \) error
\( E(\sigma, x) \) intermediate function used to define calculation
\( F() \) an analytic function
\( F(\rho, z) \) intermediate function used to define calculation
\( G \) a linear transfer function
\( H_{n}(()) \) \( n \)th order Volterra nonlinear transfer function
\( H_{m,n}(() \) \( m, n \)th order Volterra nonlinear transfer function of a
bivariate system
\( I \) current
\( I_{\omega k} \) Phasor of the current corresponding to frequency \( \omega_k \) at the
interface to a linear subcircuit
\( I'_{\omega k} \) Phasor of the current corresponding to frequency \( \omega_k \) at the
interface to a nonlinear subcircuit
\( L \) inductance
\( P \) a vector of parameters that characterize a model
\( R_i \) resistance value of the \( i \)th resistor
\( R \) the frequency domain representation of the response to a system
\( T \) subcomponent of the output
\( U \) intermediate complex variable
\( V \) voltage
\( V_i \) thermal voltage
\( V_{\omega k} \) Phasor of the voltage corresponding to frequency \( \omega_k \) at the
interface to a linear subcircuit
\( V'_{\omega k} \) Phasor of the voltage corresponding to frequency \( \omega_k \) at the
interface to a nonlinear subcircuit
\( W \) intermediate complex variable
\( X_k \) phasor representation of the \( k \)th component of the input of a
univariate system or of the input of the first port in a bivariate system
\( Y(f) \) frequency domain representation of the output
\[ Y_n(f) \] nth order frequency domain representation of the output

\[ Y_q \] phasor representation of the \( q \)th component of the output

\[ Z_k \] frequency domain representation (phasor) of the \( k \)th component of the input to the second port in a bivariate system

\( \beta \) intermediate real variable

\( \Gamma \) phasor used in intermediate calculation that always has a length of 1

\( \Lambda \) phasor used in intermediate calculation that always has a length of 1

\( \Phi \) subcomponent of the output

\( \alpha \) indexing variable that represents the number of +/- pairs of frequencies in the corresponding FIPD or a real scaling variable

\( \bar{f}_i \) \( i \)th input component frequency to the first port of a bivariate system

\( \delta() \) impulse function

\( \epsilon_k \) Neumann factor \( \Gamma \epsilon_k = 1 \) for \( k=0 \), \( \Gamma \epsilon_k = 2 \) otherwise

\( \eta \) indexing variable used to index other indexing variables

\( \bar{f}_i \) \( i \)th input component frequency to the second port of a bivariate system

\( \lambda \) time delay variable associated with the second port of a bivariate system

\( \nu \) indexing variable corresponding to the number of different values of the scaling variable \( \alpha \)

\( \omega_k \) the output radian frequency of the \( k \)th output component

\( \phi_k \) phase shift of the \( k \)th component of the input of a univariate system or the first port of a bivariate system

\( \pi \) the ratio of the circumference to the diameter of a circle = 3.141592...

\( \tau \) time delay or dummy time variable

\( \theta_k \) phase shift of the \( k \)th component of the input to the second port of a bivariate system

\( \zeta \) indexing variable used to index other indexing variables

\( e \) base for natural logarithm = 2.7128281...

\( f_i \) frequency for the \( i \)th component

\( g \) impulse response function corresponding to the transfer function \( G \)

\( g(V_{RF}, V_{LO}) \) real bivariate polynomial that characterizes a ring diode mixer

\( h_n() \) \( n \)th order Volterra kernel

\( h_{m,n}() \) \( m, n \)th order Volterra kernel of a bivariate system

\( i, k, l, m, p, q, s, \mu \) indexing variables

\( j \) square root of -1

\( m \) indexing variable representing the order of the calculation corresponding to the second port of a bivariate system

\( n \) indexing variable representing the order of the calculation

\( n_i \) indexing variable usually represents the order of the \( i \)th
subcomponent being calculated

\( p_i \)  
ith component of \( P \)

\( r \)  
the time domain representation of the response to a system

\( t \)  
time

\( x(t) \)  
time domain representation of the input for a single-input system

\( x_0 \)  
value of input used for Taylor expansion

\( x_k(t) \)  
time domain representation of the \( k \) input component

\( y \)  
output

\( y(t) \)  
time domain representation of the output

\( y_n(t) \)  
time domain representation of the \( n \)th order output component

\( z(t) \)  
time domain representation of the input to the second port in a bivariate system

\( z_k(t) \)  
time domain representation of the \( k \)th component of the input to the second port in a bivariate system

\( \text{Im}\{\} \)  
Imaginary part of the expression in braces

\( \text{Re}\{\} \)  
Real part of the expression in braces
<table>
<thead>
<tr>
<th>Abreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLSI</td>
<td>Very Large Scale Integration</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>MESFET</td>
<td>Metal–Semiconductor Field–Effect Transistor</td>
</tr>
<tr>
<td>RC</td>
<td>Resistor–Capacitor</td>
</tr>
<tr>
<td>Hz</td>
<td>Hertz</td>
</tr>
<tr>
<td>KVL</td>
<td>Kirchhoff’s Voltage Law</td>
</tr>
<tr>
<td>KCL</td>
<td>Kirchhoff’s Current Law</td>
</tr>
<tr>
<td>dB</td>
<td>decibel</td>
</tr>
<tr>
<td>IPD</td>
<td>Intermodulation Product</td>
</tr>
<tr>
<td>DC</td>
<td>Direct Current</td>
</tr>
<tr>
<td>IF</td>
<td>Intermodulation Frequency</td>
</tr>
<tr>
<td>LO</td>
<td>Local Oscillator</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>AOM</td>
<td>Arithmetic Operation Method</td>
</tr>
<tr>
<td>GPSA</td>
<td>Generalized Power Series Analysis</td>
</tr>
<tr>
<td>MLE</td>
<td>Maximum Likelihood Estimator</td>
</tr>
<tr>
<td>IP</td>
<td>Intermodulation Product</td>
</tr>
<tr>
<td>FIPD</td>
<td>Frequency Intermodulation Product Description</td>
</tr>
<tr>
<td>ASTAP</td>
<td>A Statistical Analysis Program</td>
</tr>
<tr>
<td>MHz</td>
<td>Megahertz</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>GHz</td>
<td>Gigahertz</td>
</tr>
<tr>
<td>TL</td>
<td>Transmission Line</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivations and Objectives of This Study

The design complexity of analog circuits and systems is forever increasing. With this complexity comes the problem of simulation of these large analog systems. VLSI circuit dimensions are decreasing thus requiring more accurate simulations and chip sizes are increasing thus allowing larger more complex circuits on a single chip. With VLSI technology design iterations are costly both in time and money and accurate simulation capability is a necessity for designing these large chips. Furthermore when these chips are put together to form a complex analog system the system design becomes complex and system simulation even more demanding.

Circuit simulators such as SPICE [1] are impractical for system level simulation because the cost of simulations is $O(N^3)$ where $N$ is the number of nodes. Recent advances such as waveform relaxation [2] have resulted in $O(N^\alpha)$ where $\alpha$ is typically 1.2 to 1.5. However no such speed up of global simulation has been achieved for analog circuit functions.

A standard approach to system design is to identify the important characteristics of the subsystems and define figures of merit for each subsystem. These figures of merit can be budgeted for the total system. For example for a Local Area Network (LAN) using a token ring protocol each station or subsystem contributes to the overall jitter of the signal as it travels around the ring. In order to assure that the system jitter is below some maximum the maximum jitter contribution of a single station is restricted to the total jitter budget divided by the number of stations. This approach assumes that mechanics of how jitter is added is well understood. As designs become more complex the accumulation characteristics of such figures of merit become more complicated and harder to predict. Identifying and accurately defining and budgeting these becomes more difficult.

System simulators such as SABER [3,4] address the problem of system simulation by providing the user with a language (e.g. MAST) to write behavioral models of large circuits or small subsystems. It is up to the designer to write and verify these models. The complexity of writing accurate models and the overhead and possible mistakes in verifying these models can render this method of system simulation unreliable.

The objectives of this study are to develop new ways to simulate nonlinear circuits and systems and to develop new ways to automatically extract the subsystem models to be used. This study is restricted to steady-state simulation meaning all of the signals can be expressed as the sum of periodic signals.
1.2 Overview

The organization of this thesis is divided into six chapters. Chapter one is an introductory chapter outlining the entire thesis and stating the motivation and original contributions of this study. Chapter two gives a review of the state of the art in steady-state behavioral modeling and simulation. In particular, modeling and simulation techniques that can be used to analyze the steady-state behavior of analog circuits and systems are reviewed. Also, the current technology in behavioral model extraction is reviewed.

Chapter three is a presentation of new modeling and simulation techniques that are restricted to steady-state simulation as they are based in the frequency domain. Special attention is paid to bivariate or two input models both as a behavioral black box (as in a ring diode mixer) and as a circuit element model (as in a current source modeling the drain-to-source current in a MESFET controlled by both the gate voltage and the drain voltage). Chapter four is a detailed description of a new model extraction technique for Volterra behavioral models. This technique is applicable to black box model extraction by both laboratory measurements and by device level simulation. Chapter five is a review of verification for the techniques presented in chapters four and five and include simulation results for a ring diode mixer, a MESFET amplifier, simple RC filter, a memoryless 7th order device, and a nonlinear equalization circuit. Chapter six concludes this work by summarizing and giving suggestions for further work.

1.3 Summary of Original Contributions

1.3.1 Published Work

  (sections 3.2 and 5.2)

  (sections 3.3 and 5.3)

1.3.2 Unpublished Work

- Volterra Behavioral Model Extraction (chapter 4)

- Frequency Domain Harmonic Balance using Volterra Behaviorals (section 3.4 and appendix A)
Chapter 2

Review of Steady–State Behavioral Modeling and Simulation

2.1 Introduction

The current state-of-the-art of steady-state modeling and simulation of analog circuits and systems will be discussed in this chapter. More specifically, simulation and modeling techniques that are applicable to steady-state modeling are reviewed. In this context, the term “steady-state” refers to input and output signals that can be expressed as a sum of periodic signals. In general, a steady-state signal is not required to be periodic. For example, a signal consisting of the sum of two sinusoids, one at 1 Hz and one at $\sqrt{2}$ Hz would be a steady-state signal, but it would not be a periodic signal. However, most techniques discussed here are specific to periodic signals.

First, several definitions of terms are discussed and remarks are made giving overall advantages and disadvantages of the techniques described by these terms. Secondly, modeling and simulation techniques are reviewed, and lastly, behavioral model extraction is reviewed.

2.2 Definitions & Issues

Several terms merit discussion. Here, macromodeling versus behavioral modeling, circuit versus system simulation, and time versus frequency domain simulation are discussed.

Two terms that are sometimes used interchangeably but usually carry different meaning are “macromodel” and “behavioral model.” A macromodel is a simplified circuit that represents the behavior of a more complex circuit. Macromodeling is commonly used for the simulation of complex circuits based on operational amplifiers [5]. A behavioral model, on the other hand, is usually defined as an input–output characteristic described by an equation, algorithm, or table, but a macromodel is also a behavioral model.

The terms “circuit simulation” and “system simulation” are also terms that carry different meaning. Circuit level simulation is a transistor level simulation where branches of a circuit topology are described to a simulator and the voltage–current constitutive relationship for each branch is known. The simulator then solves Kirchhoff’s voltage and current laws (KVL and KCL) to predict the performance of the circuit. A system level simulator usually has an input where the branches describing the topology have direction. In other words, the connections to the primitive blocks in the system are either inputs or outputs. An output of one block connects to the input(s) of another block(s). No two outputs are connected together. In a system simulator, the effects of loading on a signal are not taken into account. Thus in a
system simulator, the performance of a subsystem is dependent only on the signals of the input, not what is connected to the output. However, a circuit simulator can be used as a pseudosystem simulator by creating devices with the voltage-current relationship described by a behavioral model. Thus through the use of behavioral modeling, a circuit simulation can perform a mix of circuit and system simulation.

Other types of simulation (or design) techniques used to analyze high level systems are signal flow graphs [6] pp. 542-558 and spreadsheet techniques that use defined figures of merit [7]. Signal flow analysis is confined to linear systems and budgeting defined figures of merit requires a simplification and often linearization of the characteristics of the subsystems.

2.2.1 Time Domain vs Frequency Domain

Simulators capable of steady-state simulation can be divided into two types: time domain and frequency domain simulators. A time domain simulator is the most intuitive. The signals are expressed as values at points in time. On the other hand, a frequency domain steady-state simulator expresses signals as coefficients of a Fourier series or set of Fourier series. Usually the phasor form of these coefficients is used, thus each signal is a vector of complex numbers. Each complex number has an associated frequency. The magnitude and phase of the phasor corresponds to the magnitude and phase of the frequency component.

Compared to time-domain simulators, frequency domain simulators can be more efficient, especially for the case of a linear circuit or system. Moreover, non-periodic steady-state signals can be more easily represented in the frequency domain than in the time domain. Circuits which have steady-state responses can typically be simulated several orders of magnitude faster. The simulation dynamic range (defined as the ratio of the magnitude of the largest sinusoidal signal component present in the circuit to the smallest component simultaneously present) exceeds 200 dB for a frequency domain simulation [8] and approaches 400 dB for some specialized forms. On the other hand, the dynamic range for time-domain simulation can be as little as 60 dB. [9]. To put this in perspective, the accurate simulation of a 16–bit digital-to-analog converter requires that the dynamic range of the simulator be at least 100 dB. But time domain simulation is more intuitive and easier to integrate physical based semiconductor models. The transients related to non-periodic affects (e.g. start up) can also be analyzed with a time domain simulator and no such information is available with frequency domain simulators.

2.3 Modeling Techniques

For a simulator to be useful, it must be supplied with a description of the circuit or system that is to be simulated. Techniques of modeling are usually heavily dependent on the simulator that will be used. For instance, a frequency domain description of a component is more easily used in a frequency domain simulator.
2.3.1 Time Domain Modeling

Time domain modeling is reasonably intuitive since the time domain is the domain of physical reality. Other domains are just mathematical representations that enable analysis from different perspectives. For a macromodel used in circuit simulation, an input-output characteristic or current-versus-voltage characteristic can be described by a time domain equation or table so that \( y(t) = F(x(t)) \) where \( F \) is some analytic function. A behavioral model for an idealized subsystem can then be the idealized function. For example, an ideal gain stage can be modeled by simply setting the output equal to the input multiplied by the gain. This simplified model neglects skew rate limitations and other distortion factors but enables simple and efficient simulation. Behavior level time domain simulators such as SABER [314] and PROFILE [1011] use specialized languages which enable the user to model arbitrary systems. Of course trade-offs between simplicity, efficiency, and accuracy of a model are made when creating the model. But each model must be coded by hand and the resulting model is only as good as the understanding of the designer.

2.3.2 Generalized Power Series Modeling

Generalized power series analysis is a frequency domain nonlinear circuit simulation technique which utilizes generalized power series descriptions of nonlinear components. It was developed by Steer and Khan in 1983 [12] based on earlier work by Sea and Vacroux [13-15] and Heiter [16]. Examples of this simulation technique have been published for microwave circuit simulation [17-19]. The work described here forms one of the bases of the macromodeling techniques developed in this thesis.

Generalized power series requires that the constitutive relations (i.e. current versus voltage) of the nonlinear elements be described by a power series that can contain time delays. The steady-state input \( x(t) \) is described by a series of sinusoids or a vector of phasors. Each phasor \( X_k \) represents the amplitude and phase of the corresponding sinusoid.

\[
x(t) = \sum_{k=0}^{K} e_k |X_k| \cos(\omega_k t + \phi_k) = \sum_{k=-K}^{K} X_k e^{j\omega_k t}
\]  

(2.1)

The general form of the output \( y(t) \) (e.g. the current through a branch as a function of the voltage across the branch) is characterized by \( A, A_i, a_{n,i}, b_{k,i} \) and \( \tau_{k,n,i} \).

\[
y(t) = A \sum_{i=1}^{I} A_i \sum_{n=0}^{\infty} a_{n,i} \left[ \sum_{k=-K}^{K} b_{k,i} x_k(t - \tau_{k,n,i}) \right]^n
\]  

(2.2)

The resulting output phasor \( Y_q \) at frequency \( \omega_q \) is given by

\[
Y_q = \sum_{n=0}^{\infty} \sum_{n_1, \ldots, n_K} \text{Re} \{ \epsilon_n T \}_{\omega_q} \left[ n_1 + \cdots + n_K \right] = n
\]  

(2.3)
where the output frequency is given by

\[ \omega_q = \sum_{k=1}^{K} n_k \omega_k \]  

(2.4)

and \( T \) and \( \Phi \) are defined by

\[ T = \sum_{a=0}^{\infty} \sum_{s_1 + \cdots + s_K = a} (n + 2a)! \ a_{n+2a} \ \Phi \]  

(2.5)

\[ \Phi = \prod_{k=1}^{K} \left( \frac{(b_k)^{n_k + 2s_k}(X_k^\dagger)^{s_k} \Gamma_k^\dagger^{s_k}}{s_k!(n_k + s_k)!} \right) \]  

(2.6)

\[ \Gamma_{k,n} = e^{-j\omega_k \tau_{k,n}} \]  

(2.7)

Using Kirchhoff’s current and voltage laws the harmonic balance system of equations can be solved using iterative techniques such as Newton-Raphson. The key to this technique is the efficiency of the nonlinear calculation. For a given set of frequencies, an IPD (Intermodulation Production Description) table is created during the initialization of the simulation. Each entry in this table is a valid set of \( n_k \) (2.4). This table is referred to during each iteration. The table can also be saved in a file to be used in subsequent simulations that use the same set of frequencies and that contain elements with the same maximum order of the power series. For example, figure 2.1 shows a simple spectrum of a mixer circuit.

A partial list of the IPD table for this spectrum is given in table 2.1. The \( n \) and \( n_1, n_2, \) and \( n_3 \) variables are used in (2.3). Note that zero Hz (DC) is not an input that is mixed with the other frequencies. The only DC input component is when \( n = 0 \) for the DC output value for zero input. This is because we can simplify the mixing calculations by changing the form of the polynomial. For instance, take the polynomial of \( x \)

\[ y = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \]  

(2.8)

If the DC input to the input \( x \) is given by \( x_0 \) then (2.8) can be expressed in the form

\[ y = b_0 + b_1(x - x_0) + b_2(x - x_0)^2 + \ldots + b_n(x - x_0)^n \]  

(2.9)

where

\[ b_i = a_i + a_{i+1}x_0 + a_{i+2}x_0^2 + \ldots + a_n x_0^n \]  

(2.10)

This recalculation of the coefficients is inexpensive and reduces the number of IPD’s needed to calculate the output.

### 2.3.3 Power Series Ratios

In 1989 C.R. Chang introduced a new technique to calculate power series in the frequency domain [20]. Termed the Arithmetic Operator Method or AOM, this technique maps the time domain arithmetic operation into frequency domain matrix
<table>
<thead>
<tr>
<th>Output Frequency</th>
<th>n</th>
<th>n₁</th>
<th>n₂</th>
<th>n₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1 \Gamma IF$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>-1</td>
<td>2</td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>3</td>
<td>-2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>-2</td>
<td>3</td>
<td>-3</td>
</tr>
<tr>
<td>$f_2 \Gamma LO$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>-1</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>-2</td>
<td>3</td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>3</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>$f_3 \Gamma RF$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>-2</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>2</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>-3</td>
<td>3</td>
<td>-2</td>
</tr>
<tr>
<td>$f_4 \Gamma DC$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>-2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2.1: Partial listing of IPD’s when DC is not an input to the algebraic formula
operations. Since multiplication in the time domain corresponds to a matrix multiplication in the frequency domain, division in the time domain can be mapped to matrix inversion and multiplication in the frequency domain. Thus saturating curves such as $\tanh(x)$ can be efficiently modeled by a ratio of power series.

$$y = \frac{a_0 + a_1 x + a_2 x^2 + a_3 x^3 \ldots}{b_0 + b_1 x + b_2 x^2 + b_3 x^3 \ldots}$$

(2.11)

This ability to allow division in the model algebra reduces the number of terms needed for many models, especially models that have a saturating behavior similar to $\tanh(x)$. With GPSA, saturating curves such as $\tanh(x)$ require many terms. Thus AOM increases the efficiency and extends the modeling capability of GPSA.

### 2.3.4 Volterra Series

In the late 1800’s and early 1900’s Volterra developed functional series now known as Volterra series [21,22]. A summary of this work is given in [22]. This has provided a basis for modeling and simulation in a wide variety of disciplines. It has been used in the field of biology to model the thermal dependency of the heart rate [23] and the properties of the auditory system [24,25]. In the field of nonlinear hydrodynamics [26] it has been used to study ship and platform motion and stabilization [26–32]. In addition to hydrodynamics, other mechanical engineering problems have been studied using Volterra series including elastomer properties [33] and structural mechanics [34–36]. It has been used to study the optical [37] and magnetic [38]
properties of materials. The input–output properties of a laser diode have also been modeled [39].

Volterra series has been extensively used in circuit design often tailored for a specific application. For example, oscillators [40,41], power conversion systems [41], transconductance–capacitor filters [42], operational amplifiers [43,44], diode mixers [45,46], hard-limiters [47], radio receivers [48], and traveling-wave tubes [49] have all been analyzed using Volterra series. However, the most extensive use of Volterra series has been in analyzing microwave amplifiers [50–55]. There has also been a vast amount of theoretical research on Volterra series. Most of this deals with convergence properties [56,57], conditions where Volterra series exist [58–62], techniques to analytically derive Volterra descriptions for a given system [63], and relationships to other mathematical theory [64–68]. There has also been work in developing insight and techniques that will allow Volterra theory to be used in the design of nonlinear controls [69–71]. The frequency domain form of the Volterra Series forms the basis for behavioral modeling techniques developed in this thesis.

Time Domain Volterra Series

Time continuous time–domain expression for the output $y(t)$ of a function described by a Volterra series with input $x(t)$ is

$$y(t) = \sum_{n=0}^{\infty} y_n(t)$$  \hspace{1cm} (2.12)

where

$$y_n(t) = \int_{\tau_n=-\infty}^{+\infty} \int_{\tau_{n-1}=-\infty}^{+\infty} \ldots \int_{\tau_1=-\infty}^{+\infty} h_n(\tau_1, \tau_2, \ldots, \tau_n) \left[ \prod_{i=1}^{n} x(t - \tau_i) \right] d\tau_1 \, d\tau_2 \ldots d\tau_n$$  \hspace{1cm} (2.13)

The $n$th order Volterra kernel is $h_n(\tau_1, \tau_2, \ldots, \tau_n)$ and is not unique unless some further restriction is applied. Requiring that $h_n$ be symmetric can be done without loss of generality and results in a unique set of Volterra kernels for a given system. This restriction is usually applied and the following discussion assumes that the kernels are symmetric.

Frequency Domain Volterra Series

The $n$–fold Fourier transform of the Volterra kernel is known as the Volterra nonlinear transfer function $H_n$. 

$$H_n(f_1, f_2, \ldots, f_m) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \ldots \int_{-\infty}^{+\infty} h_n(\tau_1, \tau_2, \ldots, \tau_n) e^{-j2\pi(f_1\tau_1 + f_2\tau_2 + \ldots + f_m\tau_m)} d\tau_1 \, d\tau_2 \ldots d\tau_n$$  \hspace{1cm} (2.14)
Using a Fourier transform $\mathcal{F}(2,13)$ can be used to give the frequency domain description of the $n$th order output $Y_n(f)$

$$Y_n(f) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_n(f_1, f_2, \ldots, f_m) \delta(f - f_1 - f_2 - \ldots - f_n) \left[ \prod_{i=1}^{n} X(f_i) \right] df_1 \, df_2 \cdots df_n$$ (2.15)

Figure 2.2 gives a graphical representation of the frequency domain calculation of the output. The output of each order calculation is summed to give the output of the system. The output of each order calculation is calculated from the frequency domain description of the input and the Volterra nonlinear transfer function corresponding to that order. The 0th order calculation is just the DC offset for zero input and is represented by the value of the 0th order Volterra nonlinear transfer function $H_0$.

### 2.4 Simulation

Three different simulation techniques can be used for steady-state simulation. They are shooting methods, harmonic balance, and the Volterra method of nonlinear currents.

#### 2.4.1 Shooting Methods

Shooting methods [72-77] are restricted to periodic simulation and are based on time domain simulation. Since a time domain simulator can simulate a period of time equal to the period of the signal, the only remaining problem is to find the initial condition such that the initial state of the circuit is the same as the ending state of the circuit. In other words, there is a boundary condition that all signals at the end of the period simulation must be equal to the value of the signals at the beginning of the simulation. For a circuit simulator, the value of all of the signals can usually be uniquely determined by the value of all of the capacitor voltages and inductor currents. For the special case of a circuit with a transmission line, the problem becomes much more difficult because the state of a transmission line requires a continuous representation. Thus the initial state of the circuit must include a vector of time samples representing the continuous waves traveling in the transmission line. This makes shooting methods impractical for most circuits that contain transmission line models.

#### 2.4.2 Harmonic Balance

Harmonic balance simulators [8178] work in the frequency domain and thus signals are expressed as vectors of phasors. Each phasor has a corresponding frequency and this list of frequencies is usually part of the specification of the simulation. In other words, for the simulation to be accurate, one must know apriori the discrete values of the significant frequencies that will be present in the circuit.
Figure 2.2: Illustration of nonlinear analysis using Volterra series.
The overall circuit solution technique of harmonic balance simulators can be seen in figure 2.3. The circuit is grouped into two subcircuits. One subcircuit contains only linear elements and the other subcircuit contains only nonlinear elements. The error signal that is minimized during the solution iteration is defined by the voltages and currents at this interface.

\[
E = \sum_{k=0}^{N} \left| I_{\omega k} + I'_{\omega k} \right|^2 + \left| V_{\omega k} - V'_{\omega k} \right|^2
\]  

(2.16)

2.4.3 Method of Nonlinear Currents

The method of nonlinear currents is a simulation technique based on the mathematics of Volterra series that allows direct calculation of the frequency response of nonlinear circuits where the constitutive relationships of the nonlinear elements are described by a power series [79] pp 190–207, [47] pp 80–87. This method is particularly efficient when the nonlinearity is weak and has been shown to be efficient in calculating the distortion in weakly nonlinear microwave circuits.

2.5 Behavioral Model Extraction

As analog circuits and systems get larger the need for more complex simulations increases. One way to simplify the simulation is to create a behavioral model for portions of the system and use this simplified model in simulation. However the problem of creating the behavioral model is not simple. Even if the form or topology of the model has been chosen the specific model parameters must be created or extracted in some way. The creation or extraction of a specific model depends on the simulator that is being used. This section will discuss several aspects of behavioral model extraction using a written modeling language, using error minimization, using the maximum likelihood estimator, and extracting a general Volterra series model.
2.5.1 Hand Written in Modeling Language

Time domain circuit simulators that are tailored for behavioral simulation like SABER [4] and PROFILE [10][11] use a structured language to describe the model. The most common behavioral blocks are assumed to be linear and are easy to characterize. Moreover, many circuits are intentionally designed to be linear, therefore the assumption of linearity is as good as the design of the subcircuit being modeled. Furthermore, sometimes the model can be an idealization of the intended function of the intended circuit, thus no extraction is needed. Moreover, saturation effects can be simply modeled by using thresholding functions.

2.5.2 Error Minimization

Most model extraction can be thought of as an error minimization problem. Assuming that the topology of the model has been determined (e.g. the form of the equation used to model the input–output behavior) then the known behavior of the circuit can be compared to the modeled behavior of the circuit. The difference in the behavior can be thought of as the error $\epsilon$ of the model and the parameters $P$ of the model can be varied until the error is as small as possible.

$$\epsilon = f(P) \quad (2.17)$$

Linear Least Square Fit

If the relationship between the parameters $P = [p_1, p_2 \ldots p_N]$ and the behavior of the circuit can be expressed in the matrix form of (2.18) then the parameters $P$ can be directly calculated [88][pp. 194–199]. This assumes that $N$ is less than or equal to $M$ and the error to be minimized is the $L_2$ norm of the difference between the left and right side of equation 2.18. This is a common method for extracting parameters but is not always applicable because equation (2.18) cannot be formulated or because the resulting error is inappropriate.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ a_{31} & a_{32} & \cdots & a_{3N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{bmatrix} \quad (2.18)$$

Nonlinear Minimization with derivative information

Where the error minimization problem cannot be set up as a linear problem nonlinear minimization must be used. Several methods of nonlinear minimization use derivative information. These methods only find a local minimum in the parameter space but are relatively fast and efficient. The major difficulty is calculating the derivative information.
Nonlinear Minimization without derivative information – Simulated Annealing

There are also nonlinear minimization techniques which do not require derivative information. The main advantage of these techniques is that they tend to find the global instead of local minimum of the error function. These are usually termed “annealing” techniques because the algorithm has an analogy in the physics of a metal being annealed [89]. The error value is analogous to the total energy of the metal being annealed. As a metal cools each molecule has a probability of changing its energy level. The change of the state usually results in a decrease in energy. But in some cases the energy can increase and the probability of an increase in energy is greater at higher temperatures. Thus the cooling schedule used is an important factor in the efficiency and the quality of the resulting solution. A fast cooling schedule is quicker but less likely to find the global minimum.

Conventional simulated annealing is not directly applicable to a continuous parameter space; the technique called tree annealing developed by Bilbro et al [90] is very efficient. The history of previous points in the parameter space is used to bias the random choices of possible better solutions. This information is stored in a multi-dimensional binary tree and thus the term tree annealing. As with all nonlinear minimization algorithms the user must specify a range of the valid parameter space. Tree annealing is efficient at finding a global minimum especially if an appropriate cooling schedule is used and the calculation of the error function is not too expensive.

2.5.3 Maximum Likelihood Estimator

A recent development in parameter extraction that deserves mention is the use of the maximum likelihood estimator (MLE) [91–93]. This technique is based on the mathematics of Volterra series and has the advantage of minimizing the effect of gaussian white noise present in the system. Noise is always a factor that can affect the accuracy of the extracted model. MLE assumes that random noise is added to the input and output of the system due to the measurement technique. The system is first characterized by measuring the noise characteristics of the input and output with no signal applied. Thus MLE is applicable only to models based on measurements taken in a laboratory not simulated measurements derived by simulation. After the zero-input noise is characterized the output is measured with an input stimulus. The problem is then formulated “what is the most likely true input (the intended input minus the noise added) and true output (the measured output minus the noise added) for this system with the known intended input and measured output”. This formulation has the advantage of minimizing the effects of noise coupled into the system but is expensive because the true input and output are now unknown. Thus both the parameter space and the input–output space must be solved simultaneously. This translates into a larger number of unknowns. Thus MLE is especially useful in parameter extraction where the model has a small number of unknown parameters and where measurement noise is significant.
2.5.4 Volterra Series Model Extraction

General behavioral models based on Volterra series can be used to accurately represent many different systems [56]. The general Volterra model is a behavioral model where the input–output characteristics are described by a Volterra series as given in (2.13) or (2.15). The problem of extracting, storing, and using these models presents difficulties. Since a general Volterra model consists of continuous functions either the time-domain kernels or the frequency-domain nonlinear transfer functions the range and method of storage must be determined. Typically discrete points are selected. Since the time domain kernels are integrated over time the equivalent of the entire kernels must be saved. For the case of the frequency-domain nonlinear transfer function a limited set of points can be saved if the simulation frequencies are known in advance. The amount of data needed to store the model is heavily dependent on the order of the model assumed. The higher the order the more data needed. Both the Volterra kernels and the nonlinear transfer functions have \( n \) inputs (where \( n \) is the order of the kernel or nonlinear transfer function). Thus a small increase in the order of the model can vastly increase the amount of data that needs to be stored. For most applications the model must be extracted by exciting the system with several different inputs and measuring the responses. If there is a system of equations already describing the system the Volterra series can be calculated directly [63] but in that case it would usually be more efficient to base the behavioral model directly on the known set of equations.

White Noise Probing for the Development of Functional Models

The idea of using noise as a probing input to extract Volterra models of a system was first suggested by N. Wiener in the 1940’s [94]. To simplify the procedure Wiener developed G-Functionals which are based on Volterra functionals (i.e. Volterra series) but unlike Volterra functionals G-Functionals are orthogonal with respect to gaussian white noise. Unlike G-Functionals responses Volterra functional responses are homogeneous of degree \( n \) where \( n \) is the order of the functional. This extraction procedure was further developed and demonstrated on various circuits in the 1960’s by Lee and Schetzen [95][96] used in the 1960’s for various circuits and systems. In the 1980’s this method gained interest again [97–100] and gaussian noise has been used to extract Volterra based models for electrical systems [101] structural components [102][103] vertebrate photoreceptors [104] and other biological and physiological systems [105].

The method is appealing because of the simplicity of the measurement. Only a noise generator, variable delays, multipliers, and an averager are needed to extract the models. These procedures are time consuming and certain points are difficult to obtain namely \( h_n(\tau_1, \tau_2, \ldots, \tau_n) \) where \( \tau_i = \tau_j \) for all \( i \neq j \).

Two-tone probing

A second way to extract a Volterra representation of a system is to use two-tone probing [106][107]. This is very simple and straightforward but it requires that the system be of order two and thus is only applicable to a small set of systems. Since the system is of order two when the system is excited by the sum of two sinusoids the
output spectrum has a limited number of discrete components and each component characterizes a single point in one of the Volterra nonlinear transfer functions.

Quick Method

In order to maximize the information derived from a single steady-state measurement, Boyd et al developed the so-called quick method for measuring second-order Volterra kernels [108]. The procedure uses an input of the sum of several sinuoids. The relative frequencies are very carefully chosen so that most of the output frequencies are associated with only one point on a Volterra nonlinear transfer function. The method requires very careful control of the complex input signal but the output measurement and the calculation of the Volterra nonlinear transfer functions is straightforward. The system is not however applicable to extraction based on shooting method circuit simulation because of the extremely long periods of the input signal. Work by Chua and Liao published in 1989 [109] extended this work to higher order measurements. Although the theory presented is for any arbitrary order, the authors show data for third order extraction and report that this method is practical only up to fourth order.

This method is good for finding the general shapes of the nonlinear transfer functions but it is limited since there are several points that cannot be measured directly from a single measurement. For instance if the system is of order 3 or higher all points on the first order transfer function cannot be extracted from a single measurement. The output frequencies associated with the first order transfer functions will always have components associated with the third order transfer function. For example if the input spectrum contains a 4 Hz sinusoid then in general the output will contain a 4 Hz sinusoid. Part of the sinusoid will be associated with the point \( H_1(4) \) but part will also be associated with the point \( H_3(-4, 4, 4) \).

In order to separate the different orders associated with the same output frequencies several measurements can be taken using the same input frequencies and the same relative amplitudes at each frequency but the total amplitude of the signal is varied. Since the \( n \)th order response \( y_n \) is homogeneous of degree \( n \) then \( x(t) \rightarrow y_n(t) \) implies \( \alpha x(t) \rightarrow \alpha^n y_n(t) \). Thus the measurement is taken \( m \) times each time with the input \( x \) multiplied by a scalar \( \alpha_i \). If \( r_i \) is the measured response for \( \alpha_i x(t) \) then the \( n \)th order response \( y_n \) can be calculated by solving (2.19) in the least square error sense. This method of separation previously has been used for second order separation and will be used for higher order as part of the extraction method proposed in chapter 4.

\[
\begin{bmatrix}
\alpha_1 & \alpha_1^2 & \cdots & \alpha_1^N \\
\alpha_2 & \alpha_2^2 & \cdots & \alpha_2^N \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_m & \alpha_m^2 & \cdots & \alpha_m^N \\
\end{bmatrix}
\begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_N \\
\end{bmatrix}
=
\begin{bmatrix}
r_1 \\
r_2 \\
r_3 \\
\vdots \\
r_m \\
\end{bmatrix}
\] (2.19)
2.6 Conclusion

The current state of the art behavioral simulation techniques are usually based on an idealized time domain model. Volterra series techniques offer the generality needed to model arbitrary non-autonomous causal systems but techniques for model extraction and efficient simulation are lacking. Most system level simulation today is done using time domain simulators that require the equivalent of simplified macromodels. Although the modeling languages for these simulators provide general modeling capability, model synthesis and extraction is specific to the subsystem being modeled and requires extensive knowledge of the subsystem.

Some of the work described in this chapter forms the basis of the behavioral modeling technique discussed in later chapters. These are:

- Generalized Power Series indexing techniques
- Error Minimization with tree annealing
- Frequency domain Volterra models
- Two tone probing for Volterra model extraction
- Quick Method for Volterra model extraction
Chapter 3
Behavioral Modeling and Simulation

3.1 Introduction

In order to simulate large circuits efficiently, the circuit must be partitioned into smaller subcircuits and then each subcircuit characterized or modeled. The behavioral model of a subcircuit needs to have sufficient generality so that the subcircuit can be modeled to the desired accuracy. Volterra techniques provide a mathematical base that will enable a large class of input-output behaviors to be modeled. It also provides a built-in measure of complexity, the order of the model, that can be increased to improve the accuracy of the model or decreased to improve the computation speed of the model.

This chapter discusses the relationship of power series modeling to the more general case of Volterra series modeling. This relationship is important for several reasons. First, since the present state of the art includes circuit simulation based on power series modeling, this relationship gives us a foundation to extend this simulation technique to the more general case of models based on Volterra series. Secondly, since techniques already exist to extract power series models with order dependent delays, a way to translate these existing power series models into Volterra models is provided. Lastly, by understanding the relationship between power series and Volterra series, we can gain insight into the limitations of the present power series analysis.

This chapter first introduces power series analysis for the general case of a two-input one-output system. This work gives the basis for using two-input models in a power series based simulator. Secondly, the relationship between power series analysis and Volterra series analysis will be shown. The univariate (single input) case of the output described by a power series is derived first to an alternate form. Then the univariate Volterra description will also be shown in that same form, thus providing a means to derive the direct equation for the Volterra nonlinear transfer function given that the power series description is known. The same is then done for the bivariate (two-input) case. First deriving a form for the output of a bivariate power series and then by deriving the same form for a bivariate Volterra series a direct equation for the bivariate Volterra nonlinear transfer function as a function of the bivariate power series description will be given. Using the derived relationship, the differences and limitations of power series and Volterra series modeling is discussed. Although the univariate relationship has previously been established, the method used did not lend itself to the bivariate case.

The second section of this chapter shows how the indexing scheme for power series simulation can be modified so that Volterra descriptions can be used. First the simpler case of a univariate description is given and then the more complicated case of a bivariate description. This modification provides the basis of a general Volterra simulator derived from an existing power series simulator thus allowing a
practical implementation of Volterra simulator.

3.2 Bivariate Generalized Power Series

This section presents the development of an algorithm for determining the steady-state frequency domain description of the output of a system described by a bivariate (i.e. two independent inputs) power series. This power series can have complex coefficients and frequency-dependent time delays. Since the development is restricted to the steady-state response of the system, we can restrict the inputs to be represented as the sums of sinusoids.

A nonlinear element or system having the two multifrequency inputs $x(t)$ and $z(t)$ (each having $N$ components)

$$x(t) = \sum_{k=1}^{N} x_k(t) = \sum_{k=1}^{N} |X_k| \cos(\omega_k t + \phi_k)$$

(3.1)

and

$$z(t) = \sum_{k=1}^{N} z_k(t) = \sum_{k=1}^{N} |Z_k| \cos(\omega_k t + \theta_k)$$

(3.2)

can be represented by the bivariate generalized power series

$$y(t) = \sum_{\sigma=0}^{\infty} \sum_{\rho=0}^{\infty} a_{\sigma,\rho} E(\sigma, x) F(\rho, z)$$

(3.3)

with

$$E(\sigma, x) = \left( \sum_{k=1}^{N} b_k x_k(t - \tau_{k,\sigma}) \right)^{\sigma}$$

(3.4)

and

$$F(\rho, z) = \left( \sum_{k=1}^{N} d_k z_k(t - \lambda_{k,\rho}) \right)^{\rho}.$$  

(3.5)

In these expressions $a_{\sigma,\rho}$ is a complex coefficient, $b_k$ and $d_k$ are real, and $\tau_{k,\sigma}$ and $\lambda_{k,\rho}$ are time delays that depend on both the order of the power series and the index of the input frequency components. With these parameters a large variety of systems can be modeled. Our aim is to rewrite (3.3) in terms of phasors. The $x$ input can be expressed as

$$x_k(t - \tau_{k,\sigma}) = |X_k| \cos(\omega_k t + \phi_k - \omega_k \tau_{k,\sigma}) = \frac{1}{2} X_k \Gamma_{k,\sigma} e^{j\omega_k t} + \frac{1}{2} X_k^* \Gamma_{k,\sigma}^* e^{-j\omega_k t}$$

(3.6)

where $X_k$ is the phasor of the component of $x_k(t)$ at the radian frequency $\omega_k$ and

$$\Gamma_{k,\sigma} = e^{-j\omega_k \tau_{k,\sigma}}$$

(3.7)
Similarly, for the other input $z(t)$ we write

$$z_k(t - \lambda_{k,\rho}) = |Z_k| \cos(\omega_k t + \theta_k - \omega_k \lambda_{k,\rho})$$

$$= \frac{1}{2} Z_k \Upsilon_{k,\rho} e^{j\omega_k t} + \frac{1}{2} Z_k^* \Upsilon^*_{k,\rho} e^{-j\omega_k t}$$

(3.8)

where $Z_k$ is the phasor of the component of $z_k(t)$ at the radian frequency $\omega_k$ and

$$\Upsilon_{k,\rho} = e^{-j\omega_k \lambda_{k,\rho}}$$

(3.9)

Using the multinomial expansion theorem [111] we write (3.4) as

$$E(\sigma, x) = \sum_{l_1, \ldots, l_N, m_1, \ldots, m_N} \left\{ \exp \left( j \sum_{k=1}^{N} (l_k - m_k) \omega_k t \right) \right\} \sigma!$$

$$\times \prod_{k=1}^{N} \left( \frac{\left( \frac{1}{2} b_k \right)^{l_k + m_k} (X_k)^{l_k} (X_k^*)^{m_k} (\Gamma_k,\sigma)^{l_k} (\Gamma_k^*,\sigma)^{m_k}}{l_k! m_k!} \right)$$

(3.10)

where the summation is over all combinations of the integers $l_1, \ldots, l_N, m_1, \ldots, m_n$ such that $\sum_{k=1}^{N} l_k + m_k = \sigma$. Similarly, we can expand $F$ in (3.5) and so the product of $E$ and $F$ becomes

$$E(\sigma, x) F(\rho, z) = \sum_{l_1, \ldots, l_N, m_1, \ldots, m_N} \left\{ \exp \left( j \sum_{k=1}^{N} (l_k + i_k - m_k - j_k) \omega_k t \right) \right\} \sigma! \rho! \Psi$$

(3.11)

where

$$\Psi = \prod_{k=1}^{N} \left( \frac{\left( \frac{1}{2} d_k \right)^{i_k + j_k} (X_k)^{i_k} (X_k^*)^{j_k} (Z_k)^{i_k} (Z_k^*)^{j_k} (\Gamma_k^*,\sigma)^{i_k} (\Gamma_k,\sigma)^{j_k}}{l_k! m_k! i_k! j_k!} \right)$$

(3.12)

and the above summation is over all combinations of the nonnegative integers $l, m, i, j$ such that $\sum_{k=1}^{N} l_k + m_k = \sigma$ and $\sum_{k=1}^{N} i_k + j_k = \rho$. As with the single-variable power series, the frequency of each component is

$$\omega = \sum_{k=1}^{N} n_k \omega_k$$

(3.13)

where $n_k$ is a set of integers. An intermodulation product description (IPD) where $n_k = l_k + i_k - m_k - j_k$. The intermodulation order is $n \Gamma$ where $n = \sum_{k=1}^{N} |n_k|$.
Letting $(p_k + q_k)$ equal the larger of $(l_k + i_k)$ and $(m_k + j_k)$ and $(r_k + s_k)$ equal the smaller we have $p_k + q_k - r_k - s_k = n_k$ where $p_k, q_k, r_k, s_k \geq 0$.

For a given set of $n_k$'s specifying an individual intermodulation product (IP) the relevant components of $E(\sigma, x) F(\rho, z)$ can be written as the sum of two terms (for $n \neq 0$)

\[
\left(\frac{1}{2}C\right) e^{j\omega_q t} + \left(\frac{1}{2}C\right)^* e^{-j\omega_q t} = \left\{ \begin{array}{ll}
\left(\frac{1}{2}U'_q\right) e^{j\omega_q t} + \left(\frac{1}{2}U'_q\right)^* e^{-j\omega_q t} & \text{for } \omega_q \neq 0 \\
U'_q & \text{for } \omega_q = 0
\end{array} \right.
\]

(3.14)

where

\[
C = 2 \sum_{p_1, \ldots, p_N, q_1, \ldots, q_N} \left\{ \left(\frac{1}{2}\right)^{p_1 + p_N} \sigma! \rho! \Phi \right\}
\]

(3.15)

and with

\[
\Phi = \prod_{k=1}^{N} \frac{(b_k)^{p_k+r_k}(d_k)^{q_k+s_k}(X^*_k)^{p_k}(X_k)^{r_k}(Z_k^*)^{q_k}(Z_k)^{s_k}(\Gamma_k^\dagger)^{p_k}(\Gamma_k)^{q_k}(\Gamma_k^\dagger)^{r_k}(\Gamma_k)^{s_k})}{p_k! r_k! q_k! s_k!}
\]

(3.16)

Here we define

\[
X_k^\dagger = \begin{cases} X_k & \text{for } n_k \geq 0 \\
X_k^* & \text{for } n_k < 0
\end{cases}
\]

(3.17)

(where $X_k^*$ denotes the complex conjugate of $X_k$) and

\[
X_k^\dagger = \begin{cases} X_k^* & \text{for } n_k \geq 0 \\
X_k & \text{for } n_k < 0
\end{cases}
\]

(3.18)

$(Z_k^\dagger \Gamma_k \Gamma_k^\dagger)^{p_k} \Gamma_k \Gamma_k^\dagger \Gamma_k^\dagger \Gamma_k$ and $\Gamma_k \Gamma_k^\dagger \Gamma_k^\dagger$ are similarly defined.) Thus

\[
U'_q = \left\{ \begin{array}{ll}
\frac{1}{2}(C + C^*) = \text{Re}(C) & \text{for } \omega_q \neq 0 \\
\frac{1}{2}(C - C^*) = \text{Im}(C) & \text{for } \omega_q = 0
\end{array} \right.
\]

(3.19)

Note that $U'_q$ is the contribution to $E(\sigma, x) F(\rho, y)$ of one intermodulation product (IP). The two terms in (3.14) occur for $n \neq 0$ and $n = 0$ replace two sets of $i_k \Gamma_k \Gamma_k \Gamma_k$ and $m_k \Gamma_k \Gamma_k \Gamma_k$ one set corresponding to $(l_k + i_k) \geq (m_k + j_k)$ resulting in the $e^{j\omega_q t}$ term and a set corresponding to $(l_k + i_k) < (m_k + j_k)$ resulting in the $e^{-j\omega_q t}$ term. For $n = 0$ there is only one set corresponding to $(l_k + i_k) = (m_k + j_k)$. Thus the $U'_q$ expression is one-half that in (3.19) for the case $n = 0$. For (3.19) to hold we make the restriction that no IPD be equal to the negative of another IPD. If $U'_q$ is the component of $Y$ due to a single intermodulation product then

\[
U'_q = \sum_{\sigma=0}^{\infty} \sum_{\rho=0}^{\infty} a_{\sigma, \rho} U''_q
\]

(3.20)
Using the Neumann factor $\Gamma \epsilon_n$ ($\epsilon_n = 1$ if $n = 0$; $\epsilon_n = 2$ if $n \neq 0$):

$$U_q = \text{Re} \{ \epsilon_n T \}_\omega$$

(3.21)

where $\text{Re} \{ \}_\omega$ is defined such that it is ignored for $\omega \neq 0$ but for $\omega = 0$ the real part of the expression in brackets is taken. In (3.21)

$$T = \sum_{n=0}^{\infty} \sum_{\frac{p_1 + \cdots + p_N - r_1 - \cdots - r_N}{q_1 + \cdots + q_N + s_1 + \cdots + s_N} = \sigma} \frac{(\sigma! \rho! \sigma,\rho)}{2^{(n+2\alpha)}} \Phi$$

(3.22)

and $\Phi$ is given by (3.16). The phasor of the $\omega_q$ component of the output $y(t)$ is then given by

$$Y_q = \sum_{n=0}^{\infty} \sum_{n_1, \cdots, n_N} U_q, \quad \left| n_1 \right| + \cdots + \left| n_N \right| = n$$

(3.23)

We have thus derived an algebraic formula for the output of a bivariate power series having two multifrequency inputs. These formulas reduce to those presented in [12] for a single variable power series with the elimination of the appropriate variables and subsequent grouping of terms.

### 3.3 The Relationship between Volterra Series and Power Series

In this section an explicit relationship is developed between Volterra series and power series both for the univariate (single input) and bivariate (two-input) case. The strategy is to take both the equation for power series and the equation for Volterra series and derive a form for both equations that show the same structure. The relationship between the two can then be inferred by inspection. For clarity, the simpler univariate case is shown first and then the more general bivariate case is presented in the same format.

#### 3.3.1 Univariate Case

Generalized Power Series Analysis introduced by Steer and Khan [12] expresses the output $y(t)$ as a function of the input $x(t)$ by the equation

$$y(t) = A \sum_{i=1}^{l} \sum_{n=0}^{\infty} a_{n,i} \left[ \sum_{k=1}^{K} b_{k,i} x_k(t - \tau_{k,n,i}) \right]^n$$

(3.24)
Where the input is steady–state given by

\[ x(t) = \sum_{k=1}^{K} x_k(t) \]  

(3.25)

and \( x_k(t) \) is a single frequency component of radian frequency \( \omega_k \). Thus \( x_k(t) \) can be expressed as

\[ x_k(t) = X_k e^{j\omega_k t} \]  

(3.26)

Note that we must choose the \( x_k(t) \)'s such that (3.25) yields a real \( x(t) \). For the discussion here we will restrict \( A \Gamma A_i \Gamma a_{n,i} \Gamma b_{k,i} \Gamma \) and \( \tau_{k,n,i} \) to be real. Now we wish to put (3.24) in a form which is close to a conventional Volterra series form. Thus we rewrite (3.24) as

\[
y(t) = c_0 + \sum_{i=1}^{l} \sum_{n=1}^{\infty} \left[ \sum_{k=1}^{K} \hat{c}_{k,i,n} x_k(t - \tau_{k,n,i}) \right]^n \]

(3.27)

where

\[ c_0 \equiv A \sum_{i=1}^{l} a_{0,i} \]  

(3.28)

and

\[ \hat{c}_{k,i,n} \equiv [A A_i a_{n,i}]^{1/2} b_{k,i} \]  

(3.29)

Changing the summation order then gives

\[
y(t) = c_0 + \sum_{n=1}^{\infty} \sum_{i=1}^{l} \left[ \sum_{k=1}^{K} c_{k,i,n} x_k(t) \right]^n \]

(3.30)

Where

\[ c_{k,i,n} \equiv \hat{c}_{k,i,n} e^{-j\omega_k \tau_{k,n,i}} \]  

(3.31)

Or

\[
y(t) = \sum_{n=0}^{\infty} y_n(t) \]

(3.32)

Where

\[
y_n(t) = \sum_{i=1}^{l} \left[ \sum_{k=1}^{K} c_{k,i,n} x_k(t) \right]^n \]

(3.33)

and

\[ y_0(t) = c_0 \]  

(3.34)

By adding the new summation variable \( k_1 \) we can factor the term in brackets in (3.33)

\[
\left[ \sum_{k=1}^{K} c_{k,i,n} x_k(t) \right]^n = \left[ \sum_{k=1}^{K} c_{k_1,i,n} x_{k_1}(t) \right]^{n-1} \left[ \sum_{k=1}^{K} c_{k,i,n} x_k(t) \right] \]

(3.35)
And again with $k_2$ yields
\[
\left[ \sum_{k=1}^{K} c_{k,i,n} x_k(t) \right]^n = \left[ \sum_{k_1}^{K} c_{k_1,i,n} x_{k_1}(t) \right] \left[ \sum_{k_2}^{K} c_{k_2,i,n} x_{k_2}(t) \right] \cdots \left[ \sum_{k_n}^{K} c_{k_n,i,n} x_{k_n}(t) \right]^{n-2}
\] (3.36)

Continuing this $n$ times gives
\[
\left[ \sum_{k=1}^{K} c_{k,i,n} x_k(t) \right]^n = \left[ \sum_{k_1}^{K} c_{k_1,i,n} x_{k_1}(t) \right] \left[ \sum_{k_2}^{K} c_{k_2,i,n} x_{k_2}(t) \right] \cdots \left[ \sum_{k_n}^{K} c_{k_n,i,n} x_{k_n}(t) \right]^{n-1}
\] (3.37)

Moving all the terms to the right
\[
\left[ \sum_{k=1}^{K} c_{k,i,n} x_k(t) \right]^n = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \cdots \sum_{k_n=1}^{K} c_{k_1,i,n} x_{k_1}(t) c_{k_2,i,n} x_{k_2}(t) \cdots c_{k_n,i,n} x_{k_n}(t) \] (3.38)

Or
\[
\left[ \sum_{k=1}^{K} c_{k,i,n} x_k(t) \right]^n = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \cdots \sum_{k_n=1}^{K} \prod_{\zeta=1}^{n} c_{k_\zeta,i,n} x_{k_\zeta}(t) \] (3.39)

Substituting (3.39) into (3.33) and changing the summation order gives
\[
y_n(t) = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \cdots \sum_{k_n=1}^{K} \sum_{i=1}^{l} \prod_{\zeta=1}^{n} c_{k_\zeta,i,n} x_{k_\zeta}(t) \] (3.40)

or
\[
y_n(t) = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \cdots \sum_{k_n=1}^{K} \left( \prod_{y=1}^{n} x_{k_y}(t) \right) \sum_{i=1}^{l} \prod_{\zeta=1}^{n} c_{k_\zeta,i,n} \] (3.41)

This is close to the form in which the output of Volterra series analysis is evaluated. We will now consider the output of Volterra series analysis and list it in a form corresponding to (3.41).

In Volterra series analysis the output of a nonlinear system is
\[
y(t) = \sum_{n=0}^{\infty} y_n(t) \] (3.42)

where
\[
y_n(t) = \int_{\tau_n=-\infty}^{+\infty} \int_{\tau_{n-1}=-\infty}^{+\infty} \cdots \int_{\tau_1=-\infty}^{+\infty} h_n(\tau_1, \tau_2 \ldots \tau_n) \prod_{i=1}^{n} x(t - \tau_i) \, d\tau_1 \, d\tau_2 \ldots \, d\tau_n \] (3.43)

This is the integral form of Volterra series analysis and is analogous to a multidimensional convolution. The $h_n()$ terms are known as Volterra kernels. For a linear
system only the \( h_0() \) and \( h_1() \) kernels are non-zero so that only \( y_0(t) \) and \( y_1(t) \) are non-zero. The \( y_0 \) term

\[
y_0(t) = h_0
\]  

(3.44)
is just a DC output for no input. Equation (3.43) is a more general form of the Volterra integral equation than is usually reported as most authors consider \( 1 \leq n \leq \infty \) and so neglect the DC offset. Now we want to put (3.43) in frequency domain form and so we will present a number of Fourier transforms which will be used during the development.

\[
X(f) \equiv \int_{-\infty}^{+\infty} x(t)e^{-j2\pi ft} dt  
\]

(3.45)
\[
X(f)e^{+j2\pi ft_i} = \int_{-\infty}^{+\infty} x(t - \tau_i)e^{+j2\pi ft_i}d\tau_i  
\]

(3.46)
\[
x(t) = \int_{-\infty}^{+\infty} X(f)e^{+j2\pi ft} df  
\]

(3.47)
\[
Y_n(f) \equiv \int_{-\infty}^{+\infty} y_n(t)e^{-j2\pi ft} dt  
\]

(3.48)
\[
y_n(t) = \int_{-\infty}^{+\infty} Y_n(f)e^{+j2\pi ft} df  
\]

(3.49)
\[
H_n(f_1, f_2 \ldots f_n) \equiv \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} h_n(\tau_1, \tau_2 \ldots \tau_n)e^{-j2\pi (f_1\tau_1 + f_2\tau_2 + \ldots f_n\tau_n)} d\tau_1 d\tau_2 \ldots d\tau_n  
\]

(3.50)
\[
H_0 \equiv h_0  
\]

(3.51)
and

\[
h_n(\tau_1, \tau_2 \ldots \tau_n) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_n(f_1, f_2 \ldots f_n)e^{+j2\pi (f_1\tau_1 + f_1\tau_2 + \ldots f_n\tau_n)} df_1 df_2 \ldots df_n  
\]

(3.52)
Thus by substituting (3.52) into (3.43) and rearranging the terms and order of integration yields

\[
y_n(t) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_n(f_1, f_2 \ldots f_n)
\]

\[
\left( \int_{-\infty}^{+\infty} x(t - \tau_1)e^{+j2\pi f_1\tau_1} d\tau_1 \right)
\]

\[
\left( \int_{-\infty}^{+\infty} x(t - \tau_2)e^{+j2\pi f_2\tau_2} d\tau_2 \right)
\]

\[
\vdots
\]

\[
\left( \int_{-\infty}^{+\infty} x(t - \tau_n)e^{+j2\pi f_n\tau_n} d\tau_n \right)
\]

\[
df_1 df_2 \ldots df_n  
\]

(3.53)
Using (3.46) this reduces to

\[
g_n(t) = \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_n(f_1, f_2 \ldots f_n) \prod_{i=1}^{n} X(f_i) e^{j2\pi f_i t} \, df_1 df_2 \ldots df_n
\]  

(3.54)

Substituting (3.54) into (3.48) and rearranging yields

\[
Y_n(f) = \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_n(f_1, f_2 \ldots f_n) \prod_{i=1}^{n} X(f_i) \left( \int_{-\infty}^{+\infty} e^{j2\pi f_1 f} \left[ e^{j2\pi(f_1 + f_2 + \ldots + f_n)} \right] \, df \right) 
\]

or

\[
Y_n(f) = \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_n(f_1, f_2 \ldots f_n) \prod_{i=1}^{n} X(f_i) \delta(f - f_1 - f_2 - \ldots - f_n)
\]

(3.55)

(3.56)

Where \( \delta(\cdot) \) is the impulse function. This is the form of the Volterra series used for frequency domain analysis. Note that the first order response \( Y_1 \) reduces to

\[
Y_1(f) = \int_{-\infty}^{+\infty} H_1(f_1) \delta(f - f_1) X(f_1) \, df_1
\]

(3.57)

This integral is trivial to evaluate because of the \( \delta(f - f_1) \) in the integrand. Thus we get the more common form of the frequency domain equation used for linear analysis.

\[
Y_1(f) = H_1(f) X(f)
\]

(3.58)

Note that (3.58) is valid for any frequency domain description of \( X(f) \). Thus in general \( Y_1(f) \) could be considered a continuous spectrum representing any arbitrary signal. Or alternately if \( X(f) \) was a phasor representing a single tone input (i.e. an impulse in the frequency domain) then \( Y_1(f) \) would also be a single phasor representing the single tone output. Furthermore if \( X(f) \) were a set of phasors each phasor representing the phase and amplitude of a single tone at a different frequency then \( Y(f) \) would be a set of phasors corresponding to a frequency of the resulting sum of output tones.

For \( n = 2 \) (3.56) becomes

\[
Y_2(f) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} H_1(f_1, f_2) \delta(f - f_1 - f_2) X(f_1) X(f_2) \, df_1 df_2
\]

(3.59)

Here the double integral is not trivial to evaluate. So for frequency–domain Volterra analysis to be useful for nonlinear circuits a restriction must be applied to the input
of the circuit. Therefore, we require the input of the system to be steady-state. Thus we can express \( x(t) \) in the form of (3.25) where \( x_k(t) \) is defined by (3.26). Then from (3.47)

\[
X(f) = \sum_{k=1}^{K} X_k \delta(f - \frac{\omega_k}{2\pi})
\]  

(3.60)

Note that the \( \omega_k \)'s are the radian frequencies of the input and \( f \) is the frequency of the output. We will use this convention for clarity. Thus

\[
Y_n(f) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_n(f_1, f_2 \ldots f_n) \delta(f - f_1 - f_2 \ldots - f_n) \delta(f_1 - \frac{\omega_{k_1}}{2\pi}) \delta(f_2 - \frac{\omega_{k_2}}{2\pi}) \cdots \delta(f_n - \frac{\omega_{k_n}}{2\pi}) df_1 df_2 \ldots df_n
\]

(3.61)

Expanding the multiplication by introducing the summation variables \( k_1, k_2 \ldots k_n \) yields

\[
Y_n(f) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_n(f_1, f_2 \ldots f_n) \delta(f - f_1 - f_2 \ldots - f_n) \left[ \sum_{k_1=1}^{K} X_{k_1} \delta(f_1 - \frac{\omega_{k_1}}{2\pi}) \right] \left[ \sum_{k_2=1}^{K} X_{k_2} \delta(f_2 - \frac{\omega_{k_2}}{2\pi}) \right] \cdots \left[ \sum_{k_n=1}^{K} X_{k_n} \delta(f_n - \frac{\omega_{k_n}}{2\pi}) \right] df_1 df_2 \ldots df_n
\]

(3.62)

Changing the order of summation and integration

\[
Y_n(f) = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \cdots \sum_{k_n=1}^{K} \left[ \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_n(f_1, f_2 \ldots f_n) \delta(f - f_1 - f_2 \ldots - f_n) \delta(f_1 - \frac{\omega_{k_1}}{2\pi}) \delta(f_2 - \frac{\omega_{k_2}}{2\pi}) \cdots \delta(f_n - \frac{\omega_{k_n}}{2\pi}) \right] df_1 df_2 \ldots df_n
\]

(3.63)
Carrying out the integration is relatively simple since the only nonzero terms of the integrand occur when \( f_i = \omega_{k_i}/2\pi \) thus

\[
Y_n(f) = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \cdots \sum_{k_n=1}^{K} \left[ H_n\left(\frac{\omega_{k_1}}{2\pi}, \frac{\omega_{k_2}}{2\pi}, \ldots, \frac{\omega_{k_n}}{2\pi}\right) \delta\left(f - \frac{\omega_{k_1}}{2\pi} - \frac{\omega_{k_2}}{2\pi} - \cdots - \frac{\omega_{k_n}}{2\pi}\right)\right] X_{k_1} X_{k_2} \ldots X_{k_n}
\]  

(3.64)

Now using (3.49) this becomes

\[
y_n(t) = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \cdots \sum_{k_n=1}^{K} \left[ \prod_{\eta=1}^{n} x_{k_\eta}(t) \right] H_n\left(\frac{\omega_{k_1}}{2\pi}, \frac{\omega_{k_2}}{2\pi}, \ldots, \frac{\omega_{k_n}}{2\pi}\right) e^{j(\omega_{k_1} + \omega_{k_2} + \cdots + \omega_{k_n}) t} X_{k_1} X_{k_2} \ldots X_{k_n}
\]  

(3.65)

and using (3.26) we have the desired form of the output of a nonlinear system described by Volterra nonlinear transfer functions.

\[
y_n(t) = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \cdots \sum_{k_n=1}^{K} \left[ \prod_{\eta=1}^{n} x_{k_\eta}(t) \right] H_n\left(\frac{\omega_{k_1}}{2\pi}, \frac{\omega_{k_2}}{2\pi}, \ldots, \frac{\omega_{k_n}}{2\pi}\right)
\]  

(3.66)

By comparing the output of a nonlinear system described by generalized power series (3.41) with (3.66) we see that univariate Volterra series analysis is identical to univariate generalized power series analysis when the \( n \)th order Volterra nonlinear transfer function is given by

\[
H_n\left(\frac{\omega_{k_1}}{2\pi}, \frac{\omega_{k_2}}{2\pi}, \ldots, \frac{\omega_{k_n}}{2\pi}\right) = \sum_{i=1}^{l} \prod_{\zeta=1}^{n} c_{k_\zeta,i,n}
\]  

(3.67)

That is using (3.29) and (3.31) \( \Gamma \) then

\[
H_0 = c_0 \sum_{i=1}^{l} a_{0,i}
\]  

(3.68)

and for \( n = 1 \Gamma \)

\[
H_n\left(\frac{\omega_{k_1}}{2\pi}, \frac{\omega_{k_2}}{2\pi}, \ldots, \frac{\omega_{k_n}}{2\pi}\right) = A \sum_{i=1}^{l} A_i a_{n,i} \prod_{\zeta=1}^{n} b_{k_\zeta,i} e^{-j\omega_{k_\zeta} \tau_{k_\zeta,n,i}}
\]  

(3.69)

Thus if we are now given a univariate generalized power series description \( \Gamma \) we can directly calculate a Volterra nonlinear transfer function.
### 3.3.2 Bivariate Case

Now that the relationship between univariate Volterra series and power series has been developed, the bivariate case is considered. First a form of bivariate generalized power series is derived and then an equivalent form for bivariate Volterra series is also derived. As with the univariate case, a direct relationship between the two forms is demonstrated and a formula for the Volterra nonlinear transfer function as a function of the generalized power series description is derived.

Changing the subscripts to be consistent with bivariate Volterra analysis, (3.3) can be expressed as

$$ y(t) = A \sum_{i=1}^{I} A_i \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} a_{m,n,i} \left[ \sum_{k=1}^{K} b_{k,i} x_k(t - \tau_{k,n,i}) \right] \left[ \sum_{l=1}^{L} d_{l,i} z_l(t - \lambda_{l,m,i}) \right]^{m} $$

(3.70)

The sum of the first port is given by $x(t)$ and is described by (3.25) and (3.26) as was the case for the univariate system. The input at the second port is given by $z(t)$ and is given by

$$ z(t) = \sum_{i=1}^{I} z_i(t) $$

(3.71)

and $z_i(t)$ is a single frequency component of radian frequency $\omega_i$. Thus $z_i(t)$ can be expressed as

$$ z_i(t) = Z_i e^{j\omega_i t} $$

(3.72)

As with $x_k(t)$, the values of $z_i(t)$ must be in complex conjugate pairs for $\omega_i \neq 0$ such that $z(t)$ is real. The output can be expressed as

$$ y(t) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} y_{m,n}(t) $$

(3.73)

Where

$$ y_{m,n}(t) = \sum_{i=1}^{I} \hat{a}_{m,n,i} \left[ \sum_{k=1}^{K} \hat{b}_{k,i,n} x_k(t) \right] \left[ \sum_{l=1}^{L} \hat{d}_{l,i,n} z_l(t) \right]^{m} $$

(3.74)

$$ \hat{a}_{m,n,i} = A_i a_{m,n,i} $$

(3.75)

$$ \hat{b}_{k,i,n} = b_{k,i} e^{-j\omega_k \tau_{k,n,i}} $$

(3.76)

$$ \hat{d}_{l,i,n} = d_{l,i} e^{-j\omega_l \lambda_{l,n,i}} $$

(3.77)

As was done in (3.35), the exponentation in (3.74) can be factored to yield

$$ y_{m,n}(t) = \sum_{i=1}^{I} \hat{a}_{m,n,i} \left[ \sum_{k=1}^{K} \hat{b}_{k,i,n} x_k(t) \right]^{n-1} \left[ \sum_{l=1}^{L} \hat{d}_{l,i,n} z_l(t) \right]$$

(3.78)

29
Continuing this process results in

\[
y_{m;n}(t) = \sum_{i=1}^{L} \hat{a}_{m,n,i} \left[ \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \cdots \sum_{k_n=1}^{K} \prod_{\zeta=1}^{n} \hat{b}_{k_{\zeta},i,n} x_{k_{\zeta}}(t) \right] \left[ \sum_{l_1=1}^{L} \sum_{l_2=1}^{L} \cdots \sum_{l_{\mu}=1}^{L} \prod_{\nu=1}^{m} \hat{d}_{l_{\nu},i,n} z_{l_{\nu}}(t) \right]
\]

or

\[
y_{m;n}(t) = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \cdots \sum_{k_n=1}^{K} \sum_{l_1=1}^{L} \sum_{l_2=1}^{L} \cdots \sum_{l_{\mu}=1}^{L} \prod_{\zeta=1}^{n} x_{k_{\zeta}}(t) \prod_{\nu=1}^{m} \hat{b}_{k_{\nu},i,n} \sum_{i=1}^{L} \hat{a}_{m,n,i} \prod_{\nu=1}^{m} \hat{d}_{l_{\nu},i,n} z_{l_{\nu}}(t)
\]

(3.80)

The output of a system described by a bivariate Volterra series is given by

\[
y(t) = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} y_{m;n}(t)
\]

(3.82)

where

\[
y_{m;n}(t) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} h_{m;n}(\tau_1, \tau_2, \ldots, \tau_m; \bar{\tau}_1, \bar{\tau}_2, \ldots, \bar{\tau}_n) \left[ \prod_{k=1}^{m} z(t - \tau_k) \right] \left[ \prod_{l=1}^{n} x(t - \bar{\tau}_l) \right] \frac{d\tau_1}{2\pi} \frac{d\tau_2}{2\pi} \cdots \frac{d\tau_m}{2\pi} \frac{d\bar{\tau}_1}{2\pi} \frac{d\bar{\tau}_2}{2\pi} \cdots \frac{d\bar{\tau}_m}{2\pi}
\]

(3.83)

in the sense that

\[
y_{0;0}(t) = h_{00}
\]

(3.84)

The \(y_{0;0}\) term is usually omitted but is kept here for generality. As in the univariate case we will use the Fourier transform pairs of (3.45) through (3.47) and

\[
Y_{m;n}(f) \equiv \int_{-\infty}^{+\infty} y_{m;n}(t) e^{-j2\pi ft} \, dt
\]

(3.85)

\[
y_{m;n}(t) = \int_{-\infty}^{+\infty} Y_{m;n}(f) e^{+j2\pi ft} \, df
\]

(3.86)

\[
Z(f) \equiv \int_{-\infty}^{+\infty} z(t) e^{-j2\pi ft} \, dt
\]

(3.87)

\[
z(t) = \int_{-\infty}^{+\infty} Z(f) e^{+j2\pi ft} \, df
\]

(3.88)
\[ H_{m,n}(\hat{f}_1, \hat{f}_2, \ldots \hat{f}_m; \tilde{f}_1, \tilde{f}_2, \ldots \tilde{f}_n) \equiv \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} \]

\[ h_{m,n}(\hat{\tau}_1, \hat{\tau}_2, \ldots \hat{\tau}_m, \tilde{\tau}_1, \tilde{\tau}_2, \ldots \tilde{\tau}_n) \]

\[ e^{-j2\pi (\hat{f}_1 \hat{\tau}_1 + \hat{f}_2 \hat{\tau}_2 + \cdots + \hat{f}_m \hat{\tau}_m + \tilde{f}_1 \tilde{\tau}_1 + \tilde{f}_2 \tilde{\tau}_2 + \cdots + \tilde{f}_n \tilde{\tau}_n)} \]

\[ d\hat{\tau}_1 d\hat{\tau}_2 \ldots d\hat{\tau}_m d\tilde{\tau}_1 d\tilde{\tau}_2 \ldots d\tilde{\tau}_n \]  

(3.89)

and

\[ H_{0,0} \equiv h_{0,0} \]  

(3.90)

By substituting (3.91) into (3.83) and rearranging the terms and order of integration

\[ y_{m,n}(t) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_{m,n}(\hat{f}_1, \hat{f}_2, \ldots \hat{f}_m; \tilde{f}_1, \tilde{f}_2, \ldots \tilde{f}_n) \]

\[ \left( \int_{-\infty}^{+\infty} z(t - \hat{\tau}_1) e^{+j2\pi \hat{f}_1 \hat{\tau}_1} d\hat{\tau}_1 \right) \]

\[ \left( \int_{-\infty}^{+\infty} z(t - \hat{\tau}_2) e^{+j2\pi \hat{f}_2 \hat{\tau}_2} d\hat{\tau}_2 \right) \]

\[ \vdots \]

\[ \left( \int_{-\infty}^{+\infty} z(t - \hat{\tau}_m) e^{+j2\pi \hat{f}_m \hat{\tau}_m} d\hat{\tau}_m \right) \]

\[ \left( \int_{-\infty}^{+\infty} x(t - \tilde{\tau}_1) e^{+j2\pi \tilde{f}_1 \tilde{\tau}_1} d\tilde{\tau}_1 \right) \]

\[ \left( \int_{-\infty}^{+\infty} x(t - \tilde{\tau}_2) e^{+j2\pi \tilde{f}_2 \tilde{\tau}_2} d\tilde{\tau}_2 \right) \]

\[ \vdots \]

\[ \left( \int_{-\infty}^{+\infty} x(t - \tilde{\tau}_n) e^{+j2\pi \tilde{f}_n \tilde{\tau}_n} d\tilde{\tau}_n \right) \]

\[ d\hat{f}_1 d\hat{f}_2 \ldots d\hat{f}_m d\tilde{f}_1 d\tilde{f}_2 \ldots d\tilde{f}_n \]  

(3.92)

Using (3.45) and (3.87) this reduces to

\[ y_{m,n}(t) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_{m,n}(\hat{f}_1, \hat{f}_2, \ldots \hat{f}_m; \tilde{f}_1, \tilde{f}_2, \ldots \tilde{f}_n) \]

\[ \prod_{h=1}^{m} Z(\hat{f}_h) e^{+j2\pi \hat{f}_h t} \left[ \prod_{i=1}^{n} X(\tilde{f}_i) e^{+j2\pi \tilde{f}_i t} \right] \]

\[ d\hat{f}_1 d\hat{f}_2 \ldots d\hat{f}_m d\tilde{f}_1 d\tilde{f}_2 \ldots d\tilde{f}_n \]  

(3.93)
Substituting this value of \( y_{m,n}(t) \) into (3.85) yields

\[
Y_{m,n}(t) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_{m,n}(\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_m; \bar{f}_1, \bar{f}_2, \ldots, \bar{f}_n)
\]
\[
\delta(f - \hat{f}_1 - \hat{f}_2 - \ldots - \hat{f}_m - \bar{f}_1 - \bar{f}_2 - \ldots - \bar{f}_n)
\]
\[
\left[ \prod_{i=1}^{m} Z(\hat{f}_i) \right] \left[ \prod_{i=1}^{n} X(\bar{f}_i) \right]
\]
\[
d\hat{f}_1 d\hat{f}_2 \ldots d\hat{f}_m d\bar{f}_1 d\bar{f}_2 \ldots d\bar{f}_n
\]  
(3.94)

or

\[
Y_{m,n}(t) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_{m,n}(\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_m; \bar{f}_1, \bar{f}_2, \ldots, \bar{f}_n)
\]
\[
\delta(f - \hat{f}_1 - \hat{f}_2 - \ldots - \hat{f}_m - \bar{f}_1 - \bar{f}_2 - \ldots - \bar{f}_n)
\]
\[
\left[ \prod_{i=1}^{m} Z(\hat{f}_i) \right] \left[ \prod_{i=1}^{n} X(\bar{f}_i) \right]
\]
\[
d\hat{f}_1 d\hat{f}_2 \ldots d\hat{f}_m d\bar{f}_1 d\bar{f}_2 \ldots d\bar{f}_n
\]  
(3.95)

Assuming \( x(t) \) is in the form of (3.25) where \( x_k(t) \) is defined by (3.26) and \( z(t) \) is in the form of

\[ z(t) = \sum_{i=1}^{L} z_i(t) \]  
(3.96)

where \( z_i(t) \) is defined by

\[ z_i(t) = Z_i e^{+j\omega_i t} \]  
(3.97)

then from (3.88)

\[ Z(f) = \sum_{i=1}^{L} Z_i \delta(f - \frac{\omega_i}{2\pi}) \]  
(3.98)

Substituting the values of \( X(f) \) and \( Z(f) \) given by (3.60) and (3.98) into (3.95) gives

\[
Y_{m,n}(t) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_{m,n}(\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_m; \bar{f}_1, \bar{f}_2, \ldots, \bar{f}_n)
\]
\[
\delta(f - \hat{f}_1 - \hat{f}_2 - \ldots - \hat{f}_m - \bar{f}_1 - \bar{f}_2 - \ldots - \bar{f}_n)
\]
\[
\left[ \prod_{i=1}^{m} \sum_{l=1}^{L} Z_l \delta(\hat{f}_i - \frac{\omega_i}{2\pi}) \right] \left[ \prod_{i=1}^{n} \sum_{k=1}^{K} X_k \delta(\bar{f}_i - \frac{\omega_k}{2\pi}) \right]
\]
\[
d\hat{f}_1 d\hat{f}_2 \ldots d\hat{f}_m d\bar{f}_1 d\bar{f}_2 \ldots d\bar{f}_n
\]  
(3.99)

Expanding the multiplication by introducing the summation variables \( l_1, l_2 \ldots l_m \) and \( k_1, k_2 \ldots k_n \) yields

\[
Y_{m,n}(t) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} H_{m,n}(\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_m; \bar{f}_1, \bar{f}_2, \ldots, \bar{f}_n)
\]  
32
Changing the order of summation and integration

\[
\begin{align*}
\delta(f - \bar{f}_1 - \bar{f}_2 - \cdots - \bar{f}_m - \bar{f}_1 - \bar{f}_2 - \cdots - \bar{f}_n) \\
\left[ \sum_{l_1=1}^L Z_{l_1} \delta(\bar{f}_1 - \frac{\omega_{l_1}}{2\pi}) \right] \left[ \sum_{l_2=1}^L Z_{l_2} \delta(\bar{f}_2 - \frac{\omega_{l_2}}{2\pi}) \right] \cdots \left[ \sum_{l_m=1}^L Z_{l_m} \delta(\bar{f}_m - \frac{\omega_{l_m}}{2\pi}) \right] \\
\left[ \sum_{k_1=1}^K X_{k_1} \delta(\tilde{f}_1 - \frac{\omega_{k_1}}{2\pi}) \right] \left[ \sum_{k_2=1}^K X_{k_2} \delta(\tilde{f}_2 - \frac{\omega_{k_2}}{2\pi}) \right] \cdots \left[ \sum_{k_n=1}^K X_{k_n} \delta(\tilde{f}_n - \frac{\omega_{k_n}}{2\pi}) \right] \\
d\bar{f}_1 d\bar{f}_2 \cdots d\bar{f}_m d\tilde{f}_1 d\tilde{f}_2 \cdots d\tilde{f}_n
\end{align*}
\]

(3.100)

Again carrying out the integration is relatively easy since the integrand is nonzero only when \( \bar{f}_i = \omega_{k_i}/2\pi \) and \( \tilde{f}_i = \omega_{l_i}/2\pi \) thus

\[
\begin{align*}
Y_{m;n}(t) &= \sum_{k_1=1}^K \sum_{k_2=1}^K \cdots \sum_{k_n=1}^K \sum_{l_1=1}^L \sum_{l_2=1}^L \cdots \sum_{l_m=1}^L \nonumber \\
& \quad \cdot H_{m;n}(\frac{\omega_{l_1}}{2\pi}, \frac{\omega_{l_2}}{2\pi}, \cdots, \frac{\omega_{l_m}}{2\pi}; \frac{\omega_{k_1}}{2\pi}, \frac{\omega_{k_2}}{2\pi}, \cdots, \frac{\omega_{k_n}}{2\pi}) \\
& \quad \cdot \delta(f - \frac{\omega_{l_1}}{2\pi} - \frac{\omega_{l_2}}{2\pi} - \cdots - \frac{\omega_{l_m}}{2\pi} - \frac{\omega_{k_1}}{2\pi} - \frac{\omega_{k_2}}{2\pi} - \cdots - \frac{\omega_{k_n}}{2\pi}) \\
& \quad \cdot Z_{l_1} Z_{l_2} \cdots Z_{l_m} X_{k_1} X_{k_2} \cdots X_{k_n}
\end{align*}
\]

(3.102)

Now substituting this value of \( Y_{m;n}(f) \) into (3.86)

\[
\begin{align*}
y_{m;n}(t) &= \sum_{k_1=1}^K \sum_{k_2=1}^K \cdots \sum_{k_n=1}^K \sum_{l_1=1}^L \sum_{l_2=1}^L \cdots \sum_{l_m=1}^L \\
& \quad \cdot H_{m;n}(\frac{\omega_{l_1}}{2\pi}, \frac{\omega_{l_2}}{2\pi}, \cdots, \frac{\omega_{l_m}}{2\pi}; \frac{\omega_{k_1}}{2\pi}, \frac{\omega_{k_2}}{2\pi}, \cdots, \frac{\omega_{k_n}}{2\pi}) e^{+j(\omega_{l_1} + \omega_{l_2} + \cdots + \omega_{l_m} + \omega_{k_1} + \omega_{k_2} + \cdots + \omega_{k_n})t} \\
& \quad \cdot Z_{l_1} Z_{l_2} \cdots Z_{l_m} X_{k_1} X_{k_2} \cdots X_{k_n}
\end{align*}
\]

(3.103)

and using (3.26) and (3.97)

\[
y_{m;n}(t) = \sum_{k_1=1}^K \sum_{k_2=1}^K \cdots \sum_{k_n=1}^K \sum_{l_1=1}^L \sum_{l_2=1}^L \cdots \sum_{l_m=1}^L
\]

33
As we did in the univariate case by comparing (3.41) and (3.66), we now do the same in the bivariate case by comparing (3.104) and (3.81). Thus we observe that

\[
H_{m,n}(\frac{\omega_1}{2\pi}, \frac{\omega_2}{2\pi}, \frac{\omega_3}{2\pi}, \frac{\omega_4}{2\pi}, \cdots, \frac{\omega_k}{2\pi}) = \sum_{i=1}^{l} \hat{a}_{m,n,i} \left[ \prod_{\xi=1}^{n} \hat{b}_{j_{\xi},i,n} \right] \left[ \prod_{\mu=1}^{m} \hat{d}_{j_{\mu},i,n} \right]
\]

or

\[
H_{m,n}(\frac{\omega_1}{2\pi}, \frac{\omega_2}{2\pi}, \frac{\omega_3}{2\pi}, \frac{\omega_4}{2\pi}, \cdots, \frac{\omega_k}{2\pi}) = A \sum_{i=1}^{l} A_i a_{m,n,i} \left[ \prod_{\xi=1}^{n} b_{j_{\xi},i,n} e^{-j\omega_{j_{\xi}} T_{j_{\xi},i,n}} \right] \left[ \prod_{\mu=1}^{m} d_{j_{\mu},i,n} e^{-j\omega_{j_{\mu}} T_{j_{\mu},i,n}} \right]
\]

Chapter 5 will give an example of a bivariate Volterra description derived from a bivariate power series used to describe the current source of a MESFET as a function of the gate-to-source and drain-to-source voltages.

### 3.3.3 Discussion

In the previous two subsections the relationship between Volterra series analysis and generalized power series analysis was derived. Although Volterra series is more general than power series, the complexity of determining the nonlinear transfer functions used to characterize a Volterra system can be overwhelming. A power series, if it exists, is much easier to obtain. If the system can be described by a univariate or bivariate power series, we have provided a direct and straightforward means to calculate the Volterra nonlinear transfer function.

A generalized power series can contain frequency dependent delays, and these delays are only calculated for a discrete set of input frequencies; thus the resulting Volterra nonlinear transfer functions are only defined for points defined by those input frequencies. However, as would be expected, these points are exactly the ones needed for any simulation that contains only the prescribed input frequencies.

Further insight to the system representation of these two kinds of analyses can be seen using graphical representations of the system model. In order to simplify the discussion, for now we restrict \( A = A_i = I = 1 \) in (3.24) and (3.70). The
diagrams shown can be extended to arbitrary \( A, A_i \) and \( \Pi \Gamma \) but the comparisons between Volterra and power series are more easily seen for the simpler case.

First let us look at equation (3.24) and at the effects of the parameters \( b_k \) and \( \tau_{k,n} \). Here the \( k \) subscript indicates the input frequency \( \omega_k \Gamma \) and the \( n \) subscript indicates the order of the calculation. The parameter \( b_k \) is the amplification of the signal at the input frequency \( \omega_k \Gamma \) and \( \tau_{k,n} \) is the time delay equivalent to a phase shift at the input frequency \( \omega_k \). Therefore for a given \( n \Gamma \) the effect of \( b_k \) and \( \tau_{k,n} \) is equivalent to passing the input through a linear system characterized by amplitude \( b(\omega) \) and delay \( \tau_n(\omega) \). We define \( G_{b,\tau,n} \) to be this linear system characterized by \( b_k \) and \( \tau_{k,n} \).

In generalized power series analysis the coefficients \( a_n \Gamma \), can be complex. At first glance this appears to be impossible. The left side of (3.24) is a time domain signal and thus must be a pure real function. If the \( a_n \) on the right side are complex how can the right side be guaranteed to be real? The answer lies in the form of \( x_k(t) \) which is complex as given in (3.26) but \( x(t) \) is real. Thus the chosen values of \( x_k(t) \) must be in complex conjugate pairs so that (3.25) can remain real on the right side. In the calculation of the output of a generalized power series the requirement of the complex conjugate pairs is used to reduce the complexity of calculating the output. We know that the right side of (3.24) expands to a sum of complex conjugate pairs each pair corresponding to a single absolute output frequency but one component of the pair representing the positive frequency and the other component representing the negative frequency. Since we know \textit{a priori} that these will be complex conjugate pairs only the positive frequencies need to be calculated. Thus when the \( a_n \) coefficient is referred to as being complex it is really implicitly a function of the output frequency. For the negative output frequencies the complex conjugate of \( a_n \) is used. But in practice the phasors for the negative frequencies are inferred from the values calculated for the positive frequencies. Thus the negative frequency phasors are not directly calculated and in practice the coefficients \( a_n \) can be complex numbers.

Previously we defined \( G_{b,\tau,n} \) to be the linear system characterized by \( b_k \) and \( \tau_{k,n} \). Likewise we define \( G_{a,n} \) to be the linear system characterized by \( a_n \). We have already stated that \( a \) is implicitly a function of output frequency because we allowed them to be complex. We can extend GPSA such that \( a \) is explicitly a function of the output frequency. Thus \( G_{a,n} \) can be any arbitrary linear system. Moreover to simplify the calculation we can restrict \( \tau \) to be independent of \( n \) for conversely we could let \( b \) be dependent on \( n \) thus generalizing \( G_{b,\tau,n} \) to be any arbitrary linear system.

From examination of (3.24) we can see that the system representation for univariate GPSA can be given as shown in Figure 3.1. Note that \( a_0 \) is just the DC output for the system when no inputs are applied. Figure 3.2 gives the standard system representation for Volterra series analysis. By comparing figures 3.1 and 3.2 we can easily see that GPSA is equivalent to Volterra analysis if we restrict the Volterra nonlinear transfer functions \( H_n \) to be representable as shown in figure 3.3. The same representation can be applied to bivariate Volterra and GPSA analysis. Figure 3.4 shows the equivalent bivariate Volterra nonlinear transfer function that is represented by bivariate GPSA.

As previously stated the equivalent linear systems \( (G) \) that we have used to characterize power series analysis can be any arbitrary linear system. Furthermore these linear systems are not restricted to causal systems. For example take the
Figure 3.1: Univariate power series system representation.
Figure 3.2: Illustration of nonlinear analysis using Volterra series.
Figure 3.3: Equivalency of univariate Volterra analysis and generalized power series analysis (GPSA).
Figure 3.4: Equivalency of bivariate Volterra analysis and generalized power series analysis (GPSA).
simple case of \( G_a \) when \( a \) is an arbitrary complex constant. As we discussed before, \( a \) is implicitly a function of the output frequency. Thus even though we use a constant to implement the calculation of \( y(t) \), the full function of \( G_a \) is

\[
G_a(f) = \begin{cases} 
  a & \text{if } f > 0 \\
  \Re\{a\} & \text{if } f = 0 \\
  a^* & \text{if } f < 0
\end{cases}
\]  

(3.108)

We can also express this function using the signum function \((\text{sgn})\) defined by

\[
\text{sgn}(x) \equiv \begin{cases} 
  1 & \text{if } x > 0 \\
  0 & \text{if } x = 0 \\
  -1 & \text{if } x < 0
\end{cases}
\]

(3.109)

Thus (3.108) is equivalent to

\[
G_a(f) = \Re\{a\} + j \Im\{a\} \text{sgn}(f)
\]

(3.110)

To determine causality we can take the inverse fourier transform of the transfer function. The inverse Fourier transform of (3.110) is given by

\[
g(t) = \Re\{a\} \delta(t) - \frac{\Im\{a\}}{\pi t}
\]

(3.111)

We can now see that the impulse response of the system described by (3.108) has nonzero values for \( t < 0 \) and thus the system is not causal. Of course it is impossible to build a system that is not causal but the non-causal system here is only a section of a system model used to represent the steady-state response of nonlinear systems.

### 3.4 Volterra Harmonic Balance Behavioral Simulation

Given that one has the Volterra descriptions of several subcircuits, one would like to use those descriptions to simulate the entire system whose subcircuits have been characterized. If the Volterra description is independent of the input and output loading effects of the rest of the circuit, then the subcircuit can be modeled as a black box voltage or current source. For instance, assuming that the input nodes had high impedance and the output nodes had low impedance, the subcircuit could be modeled as a voltage-controlled voltage source as shown in figure 3.5. The output voltage is described by Volterra nonlinear transfer functions. If the input and output loading effects were linear, then these loading effects could be taken into account by adding an input impedance across the two input terminals and an output impedance in series with one of the output terminals. Thus a very general behavioral model can be added to a circuit simulator allowing it to act both as a circuit and as a system simulator. The simulator, however, must be able to use the Volterra description of the input–output relationship of the voltage-controlled voltage source. The scheme presented here is to include this in a harmonic balance.
Figure 3.5: Equivalent Circuit model for Voltage controlled voltage source described by Volterra nonlinear transfer functions.

simulator such as FREDA [20] since the Volterra description used is most useful in the frequency domain.

The idea of using a black box voltage-controlled voltage source can easily be extended to a current-controlled current source or a voltage-controlled current source. The scheme has several advantages. The simulator can be used for small circuits or large systems. Any mix of circuit components and behavioral black box models can easily be simulated. Furthermore, the calculation of the Volterra series description can use the same frequency indexing scheme that is used for GPSA after some modification. The scheme also can be extended to bivariate models e.g. a voltage source controlled by two input ports (four terminals).

If the time domain form of Volterra series as given in (3.43) was used instead of the frequency domain form as given in (3.66) there would be several disadvantages. The first disadvantage would be the amount of data needed for a table based model. The time domain calculation requires n-fold convolution integrals using the Volterra kernels. Thus the entire continuous functions \( h_n(t_1, t_2, \ldots, t_n) \) must be stored. Not only must they be stored but it also must be somehow measured or derived. In the scheme presented here we have restricted the input to our simulation to be represented by a set of Fourier series thus if we know which input frequencies will be used we know the exact points on the nonlinear transfer functions that will be needed. Note we do not need to know the values of the phases or amplitudes of the input phasors just a list of all of the possible input frequencies. So instead of storing entire continuous n dimensional kernel functions we only have to store a defined set of points of the n dimensional nonlinear transfer functions. Furthermore, calculating the n-fold convolution integrals in the time domain would be very expensive in terms of computation time. The integral calculation can be simplified if the form of the kernel is known in advance but such a restriction would limit either the accuracy of the model or limit the class of circuits that could be modeled.
3.4.1 Univariate Volterra and Power Series Simulation

In order to use a Volterra description of a subcircuit, one needs a simulator that can calculate the output of the system described by the Volterra nonlinear transfer functions $H_0 \Gamma H_1(f_1) \Gamma H_2(f_1, f_2) \Gamma H_3(f_1, f_2, f_3) \Gamma \ldots H_n(f_1, f_2, f_3 \ldots f_n)$. Here we present a modification to an already existing indexing scheme for GPSA to allow a GPSA simulator to use univariate Volterra nonlinear transfer functions as the characterization of a black box model.

Consider the case of a single input consisting of a finite sum of sinusoids

$$x(t) = \sum_{k=1}^{N} x_k(t) = \sum_{k=1}^{N} |X_k| \cos(\omega_k t + \phi_k)$$  \hspace{1cm} (3.112)

where the output can be described as a power series

$$y(t) = \sum_{\ell=0}^{\infty} \left[ a_\ell \left( \sum_{k=1}^{N} x_k(t) \right) \right]^\ell$$  \hspace{1cm} (3.113)

The input can be expressed as a sum of complex exponentials

$$x_k(t) = |X_k| \cos(\omega_k t + \phi_k)$$

$$= \frac{1}{2} X_k e^{j\omega_k t} + \frac{1}{2} X_k^* e^{-j\omega_k t}$$

where $X_k$ is the phasor of $x_k$. As discussed in subsection 2.3.2, the phasor of the component of the output at frequency $\omega_q$ is given by

$$Y_q = \sum_{n=0}^{\infty} \sum_{n_1, \ldots, n_N} U_q$$

$$\left| n_1 \right| + \cdots + \left| n_N \right| = n$$

$$|n_1| + \cdots + |n_N| = n$$

where $\omega_q = \sum_{k=1}^{N} n_k \omega_k$. A set of $n_k$'s defines an intermodulation product description (called an IPD) and $n$ is the order of intermodulation. The second summation is over all possible combinations of $n_1, \ldots, n_N$ such that $|n_1| + \cdots + |n_N| = n$.

$$U_q = \text{Re} \left\{ e_n T \right\}_{\omega_q}$$  \hspace{1cm} (3.115)

where

$$T = \sum_{n=0}^{\infty} \sum_{s_1, \ldots, s_N} \left\{ \left( \frac{(n + 2\alpha)!}{2^{(n+2\alpha)}} \right) a_{n+2\alpha} \Phi \right\}$$  \hspace{1cm} (3.116)

and

$$\Phi = \prod_{k=1}^{N} \frac{(X_k^*)^{n_k}}{s_k!} \left( \frac{X_k}{n_k + s_k} \right)^{2s_k}$$  \hspace{1cm} (3.117)
Where $s_k \geq 0$. In these expressions $X_k$ is the phasor of $x_k$ \( \Gamma \)

\[
X_k^\dagger = \begin{cases} 
X_k & n_k \geq 0 \\
X_k^* & n_k < 0 
\end{cases}, 
\]

(3.118)

\[
\epsilon_n = \begin{cases} 
1 & n = 0 \\
2 & n \neq 0 
\end{cases}, 
\]

(3.119)

and $\text{Re}\{\omega\}$ is defined such that for $\omega_j \neq 0$ it is ignored and for $\omega_j = 0$ the real part of the expression in braces is taken.

Assuming that the series (3.113) is convergent then the summation may be truncated and still represent an output arbitrarily close to the true value. Thus we can specify

\[
a_{\ell} = \begin{cases} 
a_{\ell} & \ell \leq n_{\text{max}} \\
0 & \ell > n_{\text{max}} 
\end{cases}, 
\]

(3.120)

Thus the infinite summation limits in (3.113)\(\Gamma\)(3.114) and (3.116) can be replaced by $n_{\text{max}} \Gamma n_{\text{max}} \Gamma$ and $n_{\text{max}}/2$. For the general case in which $N$ components are considered as inputs\(\Gamma\) a set of integers\(\Gamma\) denoted $n_k \Gamma$ are used to specify the frequency \(\Gamma\)

\[
\omega = \sum_{k=1}^{N} n_k \omega_k 
\]

As before the set of $n_k$’s is an IPD and the order of intermodulation is given by

\[
n = \sum_{k=1}^{N} | n_k | .
\]

IPD’s up to a maximum order $n_{\text{max}}$ are predetermined and stored in a database. In the evaluation of the algebraic formula\(\Gamma\) all intermodulation products of the same order are calculated and added to the total response for that frequency component until the desired fractional accuracy is obtained.

Although a device described by a generalized power series\(\Gamma\)(3.113)\(\Gamma\) has constant Volterra nonlinear transfer functions

\[
H_n(f_1, f_2 \ldots f_n) = a_n 
\]

(3.121)

the techniques used in evaluating generalized power series cannot be directly applied to evaluating the output of a system described by Volterra nonlinear transfer functions. In order to apply GPSA techniques to Volterra analysis\(\Gamma\) we need to rearrange (3.114) - (3.117). The value of $n$ in (3.114) is the order of the IPD\(\Gamma\) but not necessarily the order of the corresponding Volterra nonlinear transfer function. As seen in (3.64) the $\mu$th order nonlinear transfer function $H_\mu$ is multiplied by $\mu$ phasor components of the input. By observing the form of (3.116) - (3.117) we can define $\mu$ to be

\[
\mu = n + 2\alpha 
\]

(3.122)
Thus we will rewrite (3.116) as

\[
T = \sum_{\mu=-n}^{\infty} \sum_{s_1, \ldots, s_N: s_1 + \cdots + s_N = (\mu-n)/2} \frac{\mu!}{2^\mu} a_\mu \Phi
\]  

(3.123)

If we rearrange the summation order of (3.114) – (3.117) and use (3.123) instead of (3.116)

\[
Y_q = \sum_{\mu=0}^{\infty} \sum_{n=0,1}^{\mu} \sum_{n_1, \ldots, n_N: n_1 + \cdots + n_N = n} U_q
\]

(3.124)

\[
U_q = \text{Re} \left\{ \epsilon_n \sum_{s_1, \ldots, s_N} a_\mu \frac{\mu!}{2^\mu} \prod_{k=1}^{N} \frac{(X_k^\dagger)^{n_k} | X_k |^{2s_k}}{s_k!(n_k + s_k)!} \right\}_{\omega_q}
\]

(3.125)

We now have an equation that is in the form for both GPSA and Volterra analysis. Since \(a_\mu\) is equivalent to the Volterra nonlinear transfer function\(V\) we can use (3.125) to calculate the response to a system characterized by a Volterra series if we interpret the argument of the nonlinear transfer function correctly. Thus

\[
U_q = \text{Re} \left\{ \epsilon_n \sum_{s_1, \ldots, s_N} H_\mu(f_1, f_2, \ldots, f_\mu) \frac{\mu!}{2^\mu} \prod_{k=1}^{N} \frac{(X_k^\dagger)^{n_k} | X_k |^{2s_k}}{s_k!(n_k + s_k)!} \right\}_{\omega_q}
\]

(3.126)

Here the arguments of \(H_\mu\) are determined by the two sets of numbers \(n_k\) and \(s_k\). The value of \(s_i\) is the number of positive and negative pairs of \(f = \pm \omega_i/2\pi\). Since these are pairs whose frequencies add to zero, they have no effect on the output frequency. The value of a given \(|n_i|\) gives the number of additional times that \(f = \pm \omega_i/2\pi\) appears in the argument. A positive value of \(n_i\) indicates that exactly \(|n_i|\) values are \(+\omega_i/2\pi\) and a negative value of \(n_i\) indicates that exactly \(|n_i|\) values are \(-\omega_i/2\pi\). Since without loss of generality we can restrict the nonlinear transfer functions to be symmetric [106] (see the discussion in section B.3) the order of the \(f_i\)’s has no effect.

For example suppose \(N = 3\) \(\Gamma n_1 = 0\) \(\Gamma n_2 = -1\) \(\Gamma n_3 = 1\) \(\Gamma s_1 = 1\) \(\Gamma s_2 = 2\) \(\Gamma\) and \(s_3 = 0\). Thus

\[
\mu = \sum_{i=1}^{N} |n_i| + 2s_i = 8
\]

(3.127)
Thus for our example the arguments of $H$ would be

$$H_8\left(\frac{\omega_1}{2\pi}, \frac{-\omega_1}{2\pi}, \frac{-\omega_1}{2\pi}, \frac{\omega_2}{2\pi}, \frac{\omega_2}{2\pi}, \frac{-\omega_2}{2\pi}, \frac{-\omega_2}{2\pi}, \frac{\omega_3}{2\pi} \right)$$

$$s_1 = 1 \quad s_2 = 2 \quad n_2 = -1 \quad n_3 = +1$$

Thus with a few modifications a GPSA simulator can be turned into a general purpose Volterra simulator. Furthermore the method of nonlinear currents could be used to solve the circuit equations. This is a direct calculation and does not require iteration thus also not requiring neither the evaluation of a Jacobian matrix nor its inversion.

For the previous example with the spectrum shown in figure 2.1 tables 3.1 and 3.2 show the correspondence between the IPD and the arguments of a Volterra nonlinear transfer function. The arguments of the Volterra nonlinear transfer function is a set of frequencies and the value of the Volterra nonlinear transfer function at that point corresponds to the intermodulation of those frequencies of the order of that Volterra nonlinear transfer function. Thus we term this list of frequencies a frequency intermodulation product description or FIPD. The order of the FIPD is defined as the number of frequencies in the FIPD and is the same as the order of the corresponding Volterra nonlinear transfer function.

Table 3.1 gives the simpler case for $\alpha = 0$. For $\alpha = 0$ the order of the IPD is the same as the order of the FIPD. The absolute value of $n_1$ corresponds to the number of times frequency $f_1$ appears in the FIPD. The sign of $n_1$ is the same as the sign of the corresponding $f_1$ entries. Thus if $n_1 = -2$ then $-f_1$ would appear in the FIPD twice. The same relationship holds for $n_2$ and $f_2\Gamma n_3$ and $f_3\Gamma$ and so on.

The case of $\alpha \neq 0$ is more complicated and is shown in table 3.2. For nonzero $\alpha$ the order of the calculation (e.g., the order of the FIPD) is not equal to the order of the IPD. Instead the order of the corresponding FIPD is the order of the IPD plus $2\alpha$. For our example of three input frequencies the value of $\alpha$ is the sum of $s_1 + s_2 + s_3$. The value of $s_1$ corresponds to the number of $+/-$ pairs of $f_1$ present in the FIPD. Thus if $s_1 = 2\Gamma$ then we would find two $+/-$ pairs of $f_1$ (i.e. $f_1\Gamma - f_1\Gamma f_1\Gamma - f_1$) in the FIPD. These occurrences of $f_1$ are in addition to any occurrences associated with the value of $n_1$. The same relationship holds for $s_2$ and $f_2\Gamma n_3$ and $f_3\Gamma$ and so on.
Table 3.1: Relationship between IPD and arguments to Volterra nonlinear transfer function for $\alpha = 0$

<table>
<thead>
<tr>
<th>Output Frequency</th>
<th>$n$</th>
<th>IPD</th>
<th>FIPD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$n_1$</td>
<td>$n_2$</td>
</tr>
<tr>
<td>$f_1$ΓIF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>2</td>
<td>-2</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>-2</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>-2</td>
<td>3</td>
<td>-3</td>
</tr>
<tr>
<td>$f_2$ΓLO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>-2</td>
<td>3</td>
<td>-2</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>$f_3$ΓRF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>-2</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>-3</td>
<td>3</td>
<td>-2</td>
</tr>
</tbody>
</table>

46
Table 3.2: Example of relationship between IPD and arguments to Volterra nonlinear transfer function for nonzero $\alpha$

<table>
<thead>
<tr>
<th>$n_1$</th>
<th>$n_2$</th>
<th>$n_3$</th>
<th>$\alpha$</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>FIPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$f_2, -f_3$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$f_2, -f_3,  \underbrace{f_1, -f_1}_{s_1}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$f_2, -f_3,  \underbrace{f_2, -f_2}_{s_2}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$f_2, -f_3,  \underbrace{f_3, -f_3}_{s_3}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$f_2, -f_3,  \underbrace{f_1, -f_1, f_2, -f_2}_{s_1, s_2}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$f_2, -f_3,  \underbrace{f_2, -f_2, f_3, -f_3}_{s_2, s_3}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>$f_2, -f_3,  \underbrace{f_1, -f_1, f_1, -f_1}_{s_1, s_3}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>$f_2, -f_3,  \underbrace{f_2, -f_2, f_2, -f_2}_{s_2}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>$f_2, -f_3,  \underbrace{f_2, -f_3, f_3, -f_3}_{s_3}$</td>
</tr>
</tbody>
</table>
### 3.4.2 Bivariate Volterra and Power Series Simulation

In the previous section we discussed how a GPSA simulator can be modified to accept univariate Volterra nonlinear transfer functions as the subcircuit descriptions. Here we present the same for the bivariate case.

For the case of a dual input structure with each input $x(t)$ and $z(t)$ consisting of a finite sum of sinusoids

$$
x(t) = \sum_{k=1}^{N} x_k(t) = \sum_{k=1}^{N} |X_k| \cos(\omega_k t + \phi_k) \tag{3.129}
$$

$$
z(t) = \sum_{k=1}^{N} z_k(t) = \sum_{k=1}^{N} |Z_k| \cos(\omega_k t + \theta_k) \tag{3.130}
$$

and where the output can be described as a power series

$$
y(t) = \sum_{\sigma=0}^{\infty} \sum_{\rho=0}^{\infty} a_{\sigma,\rho} [x(t)]^\sigma [z(t)]^\rho \tag{3.131}
$$

$$
\omega = \sum_{k=1}^{N} n_k \omega_k \tag{3.132}
$$

The phasor of the $\omega_q$ component of the output $y(t)$ is then given by

$$
Y_q = \sum_{n=0}^{\infty} \sum_{n_1, \ldots, n_N} U_q \left| n_1 + \cdots + n_N \right| = n \tag{3.133}
$$

$$
U_q = \text{Re} \left\{ \epsilon_n T \right\}_{\omega_q} \tag{3.134}
$$

The intermodulation order is $n \Gamma$ where $n = \sum_{k=1}^{N} |p_k|$, where $p_k \Gamma q_k \Gamma r_k \Gamma s_k \geq 0$, $\epsilon_n$ is the Neumann factor $\Gamma(\epsilon_n = 1 \Gamma n = 0; \epsilon_n = 2 \Gamma n \neq 0) \Gamma$ and $\text{Re} \left\{ \right\}_{\omega_q}$ is defined such that it is ignored for $\omega_q \neq 0$ but for $\omega_q = 0$ the real part of the expression in brackets is taken.

$$
T = \sum_{\sigma=0}^{\infty} \left( \sum_{p_1, \ldots, p_N, q_1, \ldots, q_N, r_1, \ldots, r_N, s_1, \ldots, s_N} \right) \left\{ \left( \frac{\sigma! \rho! a_{\sigma,\rho}}{2^{(n+2\alpha)}} \right) \Phi \right\} \tag{3.135}
$$

$$
\Phi = \prod_{k=1}^{N} \frac{(X_k^*)^{p_k} (X_k^*)^{r_k} (Z_k^*)^{q_k} (Z_k^*)^{s_k}}{p_k! r_k! q_k! s_k!} \tag{3.136}
$$

$$
X_k^+ = \begin{cases} X_k & \text{for } n_k \geq 0 \\ X_k^* & \text{for } n_k < 0 \end{cases} \tag{3.137}
$$
and
\[ X_k^\dagger = \begin{cases} 
X_k^* & \text{for } n_k \geq 0 \\
X_k & \text{for } n_k < 0 
\end{cases} \]

\((Z_1^\dagger \Gamma \text{ and } Z_k^\dagger \text{ are similarly defined.})\)

As with the univariate case, our aim is to rewrite the equations with the leftmost summation over the order of the response. For the bivariate case, the order is \(\mu = \rho + \sigma\). Thus (3.135) can be rewritten as

\[
T = \sum_{\mu = n}^{\infty} \sum_{\substack{p_1, \ldots, p_N; q_1, \ldots, q_N; r_1, \ldots, r_N \text{ odd for } n \text{ odd} \\
p_1 + \cdots + p_N + q_1 + \cdots + q_N = \sigma \\
p_k + q_k - r_k - s_k = \lfloor n_k \rfloor \text{ odd for } n \text{ even}}}
\left\{ \left( \frac{\sigma! \rho! \alpha_{\sigma, \rho}}{2^\mu} \right) \Phi \right\} (3.137)
\]

As in the univariate case, we now rearrange the order of summation

\[
Y_i = \sum_{\mu = 0}^{\infty} \sum_{n = 0}^{\infty} \sum_{n_1, \ldots, n_N \text{ odd for } n \text{ odd}}^{n_1 + \cdots + n_N = n} \overline{U}_q \quad (3.138)
\]

Where

\[
\overline{U}_q = \text{Re} \left\{ \epsilon_n \sum_{\substack{p_1, \ldots, p_N; q_1, \ldots, q_N; r_1, \ldots, r_N \text{ odd for } n \text{ odd} \\
p_1 + \cdots + p_N + q_1 + \cdots + q_N = \sigma \\
p_k + q_k - r_k - s_k = \lfloor n_k \rfloor \text{ odd for } n \text{ even}}}
\left( \frac{\alpha_{\sigma, \rho}}{2} \right) \Phi \right\} (3.139)
\]

As in the univariate case, \(\alpha_{\sigma, \rho}\) can be replaced by the bivariate Volterra nonlinear transfer function \(H_{\sigma, \rho}(f_1, f_2, \ldots, f_m; \bar{f}_1, \bar{f}_2, \ldots, \bar{f}_m)\) if the values of the \(\bar{f}_i\)'s and \(\bar{f}_i\)'s are
correctly interpreted as functions of the values of $p_k \Gamma q_k \Gamma r_k \Gamma$ and $s_k$ for all $k$. Thus

$$
U_q = \text{Re} \left\{ e_n \sum_{p_1, \ldots, p_N; q_1, \ldots, q_N; r_1, \ldots, r_N} H_{\sigma, \rho}(f_1, f_2, \ldots; \tilde{f}_1, \tilde{f}_2, \ldots) \frac{\sigma! \rho!}{2^{\mu}} \Phi \right\}
$$

where

$$
p_1 + \cdots + p_N + r_1 + \cdots + r_N = \sigma
$$

$$
q_1 + \cdots + q_N + s_1 + \cdots + s_N = \rho
$$

$$
p_k + q_k - r_k - s_k = |n_k|
$$

$$
\sigma + \rho = \mu
$$

The roll of the index vectors in the bivariate case is different than that of the univariate case. The values of $p_k$ and $q_k$ represent the number of times $+\frac{\omega_k}{2\pi}$ occurs in the $\tilde{f}$ and $f$ arguments respectively. The values of $r_k$ and $s_k$ represent the number of times $-\frac{\omega_k}{2\pi}$ occurs in the $\tilde{f}$ and $f$ arguments respectively. For the example of

$$
p_2 = 3
$$

$$
p_4 = 1
$$

$$
r_4 = 2
$$

$$
q_3 = 1
$$

all others $= 0$

would correspond to

$$
H_{6,1}(+\frac{\omega_2}{2\pi}, +\frac{\omega_2}{2\pi}, +\frac{\omega_4}{2\pi}, +\frac{\omega_4}{2\pi}, -\frac{\omega_4}{2\pi}, +\frac{\omega_3}{2\pi})
$$

$$
s_2 = 3 \quad s_4 = 1 \quad r_4 = 2 \quad q_3 = 1
$$

Thus we can modify the indexing scheme for bivariate GPSA to make it more general and to handle both bivariate GPSA models and bivariate Volterra models in the same simulation. Thus a simulator could handle both bivariate power series models and the more general bivariate Volterra series models.

### 3.5 Conclusion

In order to explore ways of general frequency domain modeling, this chapter has extensively reviewed two types of frequency domain analysis: generalized power series analysis and Volterra analysis. The relationship between these has been shown for both the single input for univariate system and the two input for bivariate system. By understanding this relationship, the indexing scheme for GPSA was modified to provide a basis for simulation of Volterra models for both univariate and bivariate. Since the Volterra model is more general, a Volterra simulation could include both
Volterra models and GPSA models. This kind of simulation would extend the modeling capability available and provide a basis where complex subsystems could be simulated using Volterra descriptions and device level circuits could in the same simulation be modeled using power series techniques. Thus frequency domain circuit simulation could be generalized to include system level verification. However this system level simulation is not possible without the ability to extract Volterra models. This extraction is examined in the next chapter.
Chapter 4
Volterra Behavioral Model Extraction

4.1 Introduction

The advantage of circuit and system simulation is the fact that a circuit can be designed and tested before it is built thus giving the designer a fast and inexpensive way to test designs. But of course for any simulation the results are only as good as the models. For simulating large circuits or systems the compute resource required can be prohibitive if transistor level SPICE-like simulations are used. In order to provide a practical means of simulating large systems the Volterra harmonic balance technique proposed in section 3.4 can be used. But the work to date on extracting table-based Volterra behavioral models has been restricted to simple models and extraction techniques based on lab measurements. Thus a circuit can only be modeled after it is built. The technique proposed here is shown to be applicable to extraction by running circuit level simulations thus allowing the designer to simulate the entire system before the subcircuits are built. Furthermore if the subcircuit description can be simulated for various process variations multiple behavioral models can be extracted to represent the performance variation of the manufactured subcircuits and thus the manufactured system. This ability to predict not only the nominal performance but also the performance variation due to manufacturing tolerances becomes increasingly important as systems become larger and more complex.

This chapter will present a procedure to extract the Volterra nonlinear transfer function \(H_N(f_1, f_2, \ldots, f_n)\). This procedure requires only a circuit level description of the subcircuit to be modeled and a circuit simulator that can simulate the steady-state behavior of the circuit. This can be either a time-based simulator that uses shooting methods [72-75] or a frequency-based simulator that uses harmonic balance techniques [81]. One also needs to know the possible discrete input frequencies and the range of input amplitudes. An additional parameter that must be specified by the user is the maximum order of the system which will be the highest order Volterra nonlinear transfer function to be extracted. The extraction process uses this order as the order of multiple polynomial curve fits and assume \(H_n = 0\) for all \(n\) greater than the specified maximum order. The error of this polynomial curve fit can be used to determine if the specified maximum order is sufficiently large.

4.2 Extraction procedure

A high-level flow chart of the overall extraction procedure is given in figure 4.1. The initialize portion of the program includes the determination of both how many simulations are to be run and exactly what the input stimulus of each simulation will be. In order to understand how this selection is determined we must first review how the Volterra nonlinear transfer function is calculated in the last step.
Begin

Initialize

Create Simulation Files

Simulate

Calculate $H_n$

End

Figure 4.1: High level flow chart for the procedure to extract the nonlinear Volterra transfer functions using circuit simulation
4.2.1 Output Decomposition

The extraction procedure is based on the fact that the mapping of the nth order response \( y_n \) is homogeneous of degree \( n \) with respect to the input \( x \). That is, for the mapping \( x \rightarrow y_n \) for any scalar \( \alpha \) then \( \alpha x \rightarrow \alpha^n y_n \). This property can be seen by studying equation reeq:volt,yn. As would be expected \( \Gamma \) this property also holds for the frequency domain description. Thus

\[
\alpha X \rightarrow \alpha^n Y_n
\]  

(4.1)

Our procedure now is to pick several sets of “runs”. Each run is defined by a single set of input frequencies and relative amplitudes of these frequencies. If a time-domain shooting method simulator is going to be used then these frequencies must be commensurable. For each run \( \nu \) simulations are performed \( \Gamma \) where \( \nu \) is greater than or equal to the assumed order \( \Gamma n \) \( \Gamma \) of the system. Each simulation has the same input frequencies and the relative amplitudes and phases of the input frequencies are the same with respect to each other \( \Gamma \) but the total input signal amplitude is changed for each simulation. Thus the inputs of each simulation are only scalar multiples \( \Gamma \) of a reference input \( x \). We will characterize each simulation amplitude by the scalar \( \alpha_i \) so the input amplitudes for the simulations of a given run are \( \alpha_1 x, \alpha_2 x, \ldots, \alpha_\nu x \). If we refer to the simulated responses of the system as \( r_i \) where \( 1 \leq i \leq \nu \) then

\[ r_i = y_0 + \alpha_i y_1 + \alpha_i^2 y_2 + \ldots + \alpha_i^n y_n \]  

(4.2)

Note here that \( r_i \) and \( \alpha_i \) are known and \( y_n \) is the unknown nth order system response as given in (3.43) for the known reference input \( x \) for a given run. Here \( y_0 \) is just the DC offset of the output. We can easily determine \( y_0 \) by simulating the system with a zero value input. In our procedure \( \Gamma \) this is the first simulation and the resulting DC output is subtracted from all subsequent simulations. For simplicity \( \Gamma \) we will drop the \( y_0 \) term and assume that this value has been subtracted from \( r_i \).

We can also look at the frequency domain representation of (4.2)

\[ R_{q,i} = \alpha_i Y_{q,1} + \alpha_i^2 Y_{q,2} + \ldots + \alpha_i^n Y_{q,n} \]  

(4.3)

Here the capital letter indicates that the value is a complex number representing a phasor for a particular output frequency \( y_q \). \( R_{q,i} \) represents the measured output at frequency \( \omega_q \) for the \( i \)th simulation of a given run. \( Y_{q,j} \) is the \( j \)th order response of the output frequency \( \omega_q \) of the system simulated with the reference input \( x \). Since \( \alpha \) is a real number \( \Gamma (4.3) \) can be easily separated into real and imaginary parts I or

\[
\begin{align*}
\text{Re} \{ R_{q,i} \} &= \alpha_i \text{Re} \{ Y_{q,1} \} + \alpha_i^2 \text{Re} \{ Y_{q,2} \} + \ldots + \alpha_i^n \text{Re} \{ Y_{q,n} \} \\
\text{Im} \{ R_{q,i} \} &= \alpha_i \text{Im} \{ Y_{q,1} \} + \alpha_i^2 \text{Im} \{ Y_{q,2} \} + \ldots + \alpha_i^n \text{Im} \{ Y_{q,n} \}
\end{align*}
\]  

(4.4)

(4.5)

Using all \( \nu \) simulations for a given run \( \Gamma \) we can express (4.3) in matrix form.

\[
\begin{bmatrix}
\text{Re} \{ R_{q,1} \} \\
\text{Re} \{ R_{q,2} \} \\
\text{Re} \{ R_{q,3} \} \\
\vdots \\
\text{Re} \{ R_{q,\nu} \}
\end{bmatrix} =
\begin{bmatrix}
\alpha_1 & \alpha_1^2 & \alpha_1^3 & \cdots & \alpha_1^n \\
\alpha_2 & \alpha_2^2 & \alpha_2^3 & \cdots & \alpha_2^n \\
\alpha_3 & \alpha_3^2 & \alpha_3^3 & \cdots & \alpha_3^n \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\alpha_\nu & \alpha_\nu^2 & \alpha_\nu^3 & \cdots & \alpha_\nu^n
\end{bmatrix}
\begin{bmatrix}
\text{Re} \{ Y_{q,1} \} \\
\text{Re} \{ Y_{q,2} \} \\
\text{Re} \{ Y_{q,3} \} \\
\vdots \\
\text{Re} \{ Y_{q,n} \}
\end{bmatrix}
\]  

(4.6)
and
\[
\begin{bmatrix}
\text{Im } \{R_{g,1}\} \\
\text{Im } \{R_{g,2}\} \\
\text{Im } \{R_{g,3}\} \\
\vdots \\
\text{Im } \{R_{g,n}\}
\end{bmatrix} = \begin{bmatrix}
\alpha_1 & \alpha_1^2 & \alpha_1^3 & \cdots & \alpha_1^n \\
\alpha_2 & \alpha_2^2 & \alpha_2^3 & \cdots & \alpha_2^n \\
\alpha_3 & \alpha_3^2 & \alpha_3^3 & \cdots & \alpha_3^n \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\alpha_\nu & \alpha_\nu^2 & \alpha_\nu^3 & \cdots & \alpha_\nu^n
\end{bmatrix}
\begin{bmatrix}
\text{Im } \{Y_{f,1}\} \\
\text{Im } \{Y_{f,2}\} \\
\text{Im } \{Y_{f,3}\} \\
\vdots \\
\text{Im } \{Y_{f,n}\}
\end{bmatrix}
\] (4.7)

Thus for each run the response at output frequency $\omega_y$ can be decomposed into the $n$th order responses by solving (4.6) and (4.7) for $Y_{g,i}$. This is equivalent to a polynomial curve fit of a real function. Of course the $\alpha$-based matrix must not be singular but we are free to choose any $\alpha$. For small $n = \nu \leq 3$ the problem of choosing a set of $\alpha$’s such that the matrix is as well conditioned as possible has been solved in closed form [109] but for the general case the equations become too complicated to be practical. For our measurement procedure we will use $\alpha_i$’s that alternate in sign and whose absolute values are equally spaced in the interval $0 < |\alpha_i| \leq 1$

\[
\alpha_i = -1^{(i+1)} \frac{i}{\nu}
\]

For some sets of input frequencies especially those with few distinct input frequencies some output frequencies may have only even or odd order components. These cases are known a priori and thus we can reduce the size of the $\alpha$ matrix and simplify the decomposition. For example for the case of a single input frequency (single-tone excitation) only the even order responses contribute to the first harmonic output frequency and so (4.6) and (4.7) reduce to (assuming even $n$)

\[
\begin{bmatrix}
\text{Re } \{R_{g,2}\} \\
\text{Re } \{R_{g,4}\} \\
\text{Re } \{R_{g,6}\} \\
\vdots \\
\text{Re } \{R_{g,\nu}\}
\end{bmatrix} = \begin{bmatrix}
\alpha_1^2 & \alpha_1^4 & \alpha_1^6 & \cdots & \alpha_1^n \\
\alpha_2^2 & \alpha_2^4 & \alpha_2^6 & \cdots & \alpha_2^n \\
\alpha_3^2 & \alpha_3^4 & \alpha_3^6 & \cdots & \alpha_3^n \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\alpha_\nu^2 & \alpha_\nu^4 & \alpha_\nu^6 & \cdots & \alpha_\nu^n
\end{bmatrix}
\begin{bmatrix}
\text{Re } \{Y_{f,2}\} \\
\text{Re } \{Y_{f,4}\} \\
\text{Re } \{Y_{f,6}\} \\
\vdots \\
\text{Re } \{Y_{f,n}\}
\end{bmatrix}
\] (4.9)

and

\[
\begin{bmatrix}
\text{Im } \{R_{g,2}\} \\
\text{Im } \{R_{g,4}\} \\
\text{Im } \{R_{g,6}\} \\
\vdots \\
\text{Im } \{R_{g,\nu}\}
\end{bmatrix} = \begin{bmatrix}
\alpha_1^2 & \alpha_1^4 & \alpha_1^6 & \cdots & \alpha_1^n \\
\alpha_2^2 & \alpha_2^4 & \alpha_2^6 & \cdots & \alpha_2^n \\
\alpha_3^2 & \alpha_3^4 & \alpha_3^6 & \cdots & \alpha_3^n \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\alpha_\nu^2 & \alpha_\nu^4 & \alpha_\nu^6 & \cdots & \alpha_\nu^n
\end{bmatrix}
\begin{bmatrix}
\text{Im } \{Y_{f,2}\} \\
\text{Im } \{Y_{f,4}\} \\
\text{Im } \{Y_{f,6}\} \\
\vdots \\
\text{Im } \{Y_{f,n}\}
\end{bmatrix}
\] (4.10)

### 4.2.2 Transfer Function Calculation

Our goal is to calculate the Volterra nonlinear transfer function $H_n$ for all $n$. Now that we know the $i$th order output response at frequency $\omega_y Y_{g,i}$ and the input $X$ the problem is to calculate the appropriate $H_i$’s. In order to understand the problems of this calculation we need to look at how $Y_{g,i}$ can be calculated from $H_i$ and $X$. From (3.65) we know that the $n$th order steady-state response at an output
frequency of \( \omega_q \) of a system characterized by Volterra nonlinear transfer functions can be represented in the form

\[
Y_{q,n} = \sum_j^J \beta_j H_{n}^{(j)} \prod_i^n X_{i,j}
\]  

(4.11)

Where the arguments to \( H_{n}^{(j)} \) are the frequencies corresponding to the frequencies associated with \( X_{i,j} \). We will refer to a given set of these frequencies as a Frequency Intermodulation Product Description or FIPD. Thus an FIPD is just the arguments to the Volterra nonlinear transfer function consisting of \( n \) frequencies. The sum of the frequencies in an FIPD will be the value of the output frequency.

\[
\sum_i^n \omega_i = \omega_q
\]  

(4.12)

Note that the frequencies in an FIPD can be negative. Thus \( J \) is the number of FIPD’s of order \( n \) that contribute to the \( \omega_q \). The value of \( \beta_j \) is known and a function of the FIPD and is equivalent to the \((n + 2\alpha)!/(s_1!(n_1 + s + k)!)\) term in (3.116). The product \( \prod_i^n X_{i,j} \) is the set of input phasors that correspond to the appropriate components of an FIPD for the output frequency \( \omega_q \). Letting \( W^{(j)} = \beta_j \prod_i^n X_{i,j} \) represent the known values of the equation, \( \sum_j^J \) becomes

\[
Y_{q,n} = \sum_j^J H^{(j)} W^{(j)}
\]  

(4.13)

Letting an \( re \) subscript indicate the real part of the complex number and an \( im \) subscript represent the imaginary part of a complex number we can express (4.13) as a real matrix equation.

\[
\begin{bmatrix}
W^{(1)}_{re} & -W^{(1)}_{im} & W^{(2)}_{re} & -W^{(2)}_{im} & \ldots & W^{(J)}_{re} & -W^{(J)}_{im}
W^{(1)}_{im} & W^{(1)}_{re} & W^{(2)}_{im} & W^{(2)}_{re} & \ldots & W^{(J)}_{im} & W^{(J)}_{re}
W^{(1)}_{im} & W^{(1)}_{re} & W^{(2)}_{im} & W^{(2)}_{re} & \ldots & W^{(J)}_{im} & W^{(J)}_{re}
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots
\end{bmatrix}
\begin{bmatrix}
H^{(1)}_{re} \\
H^{(1)}_{im} \\
H^{(2)}_{re} \\
H^{(2)}_{im} \\
\vdots \\
H^{(J)}_{re} \\
H^{(J)}_{im}
\end{bmatrix} = \begin{bmatrix}
Y_{q,n,rr} \\
Y_{q,n,ri} \\
Y_{q,n,ir} \\
Y_{q,n,ii}
\end{bmatrix}
\]  

(4.14)

For \( J = 1 \) the number of knowns equal the number of unknowns and we can solve for \( H_{re}^{(1)} \) and \( H_{im}^{(1)} \) as long as the resulting \( 2 \times 2 \) matrix is not singular. For \( J > 1 \) we need either to know some of \( H \) or we need to add some rows to the matrix for a combination of both.

We will now examine when the number of unknowns in (4.14) is greater than the number of rows in the left matrix. For any first order system the output at any frequency is due only to a single input frequency; thus the number of unknowns for a single output frequency is at most two: one for \( \text{Im}\{H_1\} \) and one for \( \text{Re}\{H_1\} \).
Table 4.1: All FIPD’s for a second order system with a discrete input spectrum of 1Hz and 2Hz.

<table>
<thead>
<tr>
<th>Transfer Function</th>
<th>FIPD</th>
<th>Order</th>
<th>Output Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) $H_0$</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(b) $H_1(1)$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(c) $H_1(2)$</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>(d) $H_2(-1, 1)$</td>
<td>-1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>(e) $H_2(-2, 2)$</td>
<td>-2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>(f) $H_2(1, 1)$</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>(g) $H_2(-1, 2)$</td>
<td>-1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>(h) $H_2(1, 2)$</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>(i) $H_2(2, 2)$</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

The exception to this is the DC or zero output frequency point. It can have two components—one from $H_0$ that is not associated with any input frequency and one from $H_1(0)$ if a DC input is present. However, we can easily determine the $H_0$ value by a single simulation with no inputs but we must do this before we try to calculate $H_1(0)$. Thus the order that we determine the values of $H$ is important.

For systems of order less than or equal to two (4.14) can always be solved by using previously calculated values of $H$ if the correct runs (i.e., sets of input frequencies) are simulated and the Volterra nonlinear transfer functions calculated in the correct order. As with the first order case above, the value of $H_0$ must be calculated before $H_1(0)$ can be found. The next question is what values of $H_1$ are needed before some value of $H_2$ can be calculated and how can one in general always have the needed values of $H_1$ available when calculating $H_2$. This can easily be accomplished by picking which simulations are to be performed and in which order. All of the single-tone inputs are simulated first, then the two-tone inputs are simulated. Since each FIPD is limited to at most two frequencies for the case of a second order system, more than two tones are not required to find any value of the two transfer functions $H_1$ and $H_2$. For instance, suppose that we were trying to calculate the values of $H$ for a second order system in order to be able to predict the response of that system for any phase and amplitude combination of 1 Hz and 2 Hz. The needed values of the Volterra nonlinear transfer functions would be as shown in Table 4.1.

In this example the four different runs indicated in Table 4.2 are required. Here run (A) has no input (B) has a single 1 Hz input (C) has a single 2 Hz input and (D) had both 1 Hz and 2 Hz inputs. Run (A) would determine $H_0$. Run (B) would have two output frequencies 0 Hz and 2 Hz. The 0 Hz output component would be a sum of the effects of (a) and (f) but run (A) already determined the value of $H_0$ thus leaving only $H_2(-1, 1)$ unknown. Run (C) would also have two output frequencies 0 Hz and 4 Hz. As before the 0 Hz output component would be a sum
Table 4.2: List of simulation runs for extracting all Volterra nonlinear transfer functions points needed to calculate an output for a second order system with 1 Hz and 2 Hz input phasors

<table>
<thead>
<tr>
<th>Input Frequencies</th>
<th>Output Frequencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A) no input</td>
<td>0</td>
</tr>
<tr>
<td>(B) 1 Hz</td>
<td>0, 1, 2</td>
</tr>
<tr>
<td>(C) 2 Hz</td>
<td>0, 2, 4</td>
</tr>
<tr>
<td>(D) 1 Hz and 2 Hz</td>
<td>0, 1, 2, 3, 4</td>
</tr>
</tbody>
</table>

of two FIPD's one from (a) and one from (e). And again \( H_0 \) is already known leaving just \( H_2(-2, 2) \). However run (D) is more complicated. The output has all of the components due to all of the FIPD's. But careful examination indicates that at most only one value of \( H \) is now unknown for each output frequency. All of the values of \( H \) contributing to an output frequency of 0 Hz (all \( \pm \)) are already known. For the output frequency of 1 Hz only \( H_2(-1, 2) \) is unknown. Both of the values of \( H \) for an output frequency of 2 Hz are known. And there is only one FIPD for each of the remaining output frequencies 3 Hz and 4 Hz. This method of measuring transfer functions for low order systems is referred to as the harmonic probing method and previously has been successfully used to measure systems of order less than or equal to two [107].

For assumed system order of a value greater than two the problem becomes more complex. There are four situations where care must be taken when trying to solve (4.14) for balanced DC FIPD, unbalanced DC FIPD, FIPD with an unbalanced DC FIPD subset and other unique FIPD's with the same order and frequencies.

Balanced DC FIPD

When an FIPD is balanced with respect to zero (e.g. \( H_4(-2, -1, 1, 2) \)) the value of the nonlinear transfer function will be real. To be a balanced FIPD each positive frequency must have a corresponding negative frequency in the FIPD. For this case we know that the value of \( H \) must be real. See Appendix B, section B.5 for further discussion. This case of a pure real transfer function is not a problem as then (4.14) will have the form

\[
\begin{bmatrix}
W_{rc}^{(1)} & 0 \\
0 & W_{rc}^{(1)}
\end{bmatrix}
\begin{bmatrix}
H_{rc}^{(1)} \\
H_{im}^{(1)}
\end{bmatrix}
= \begin{bmatrix}
Y_{r,m,rc} \\
0
\end{bmatrix}
\]

(4.15)

We can see that \( H_{im}^{(1)} = 0 \). Roundoff errors or other inaccuracies in the simulation and Fourier transform process can cause \( W_{im}^{(1)} \) and \( Y_{r,m,im} \) to be very small nonzero numbers. Since for this case we know ahead of time by inspecting the FIPD that \( H_{im}^{(1)} = 0 \) we just set it to zero and calculate \( H_{rc}^{(1)} = Y_{r,3,rc} \div W_{rc}^{(1)} \).
Unbalanced DC FIPD

For orders greater than two there can be unbalanced arguments to the nonlinear transfer function that add to zero. In these cases the transfer function will not in general be real. Of course since the output frequency is zero the imaginary part of the measured output will be zero \(Y_{i,n,im} = 0\). Furthermore the value of \(W^{(1)}\) can be complex.

For example the third order DC response of a system with input of three sinusoids of frequencies 1 Hz, 2 Hz, and 3 Hz the unknown values of \(H_3\) are \(H_3(-2, -1, 3)\) and \(H_3(-3, 2, 1)\). The other points contributing to the DC output of the system \(H_3(-2, 1, 1) H_3(-1, -1, 2)\) can be calculated from previous runs using simpler simulations with inputs of only two frequencies 1 Hz and 2 Hz. \(H_3(-2, -1, 3)\) can be in general complex but both \(H^{(1)} = H_3(-2, -1, 3)\) and \(H^{(2)} = H_3(-3, 1, 2)\) contribute to the third order DC output. We also know that for this case \(H_3(-2, -1, 3) = H_3(-3, 1, 2)\) and \(W^{(1)} = (W^{(2)})^*\). (See appendix B for more information). Thus it works out that \(H^{(1)} W^{(1)} + H^{(2)} W^{(2)} = 2 \text{Re}\{H^{(1)} W^{(1)}\}\). Therefore (4.14) will have the form

\[
\begin{bmatrix}
W^{(1)}_{11} & -W^{(1)}_{12} & W^{(1)}_{13} \\
-W^{(1)}_{21} & W^{(1)}_{22} & -W^{(1)}_{23} \\
W^{(1)}_{31} & -W^{(1)}_{32} & W^{(1)}_{33}
\end{bmatrix}
\begin{bmatrix}
H^{(1)}_{1e} \\
H^{(1)}_{1m} \\
H^{(1)}_{im}
\end{bmatrix} =
\begin{bmatrix}
Y_{q,3,er} \\
0
\end{bmatrix}
\]

(4.16)

Which reduces to

\[
\begin{bmatrix}
2W^{(1)}_{1e} & -2W^{(1)}_{1m} \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
H^{(1)}_{1e} \\
H^{(1)}_{1m} \\
H^{(1)}_{im}
\end{bmatrix} =
\begin{bmatrix}
Y_{q,3,er} \\
0
\end{bmatrix}
\]

(4.17)

Note that we cannot solve (4.17) for \(H^{(1)}_{1e}\) and \(H^{(1)}_{1m}\). We need more information. To solve this problem we can add two additional rows to the matrix equation. These two rows would represent an additional run using the same input frequencies. The additional run however must be different from the first run. The two things we can change and still keep the same input frequencies are the relative amplitudes of the input signals and the relative phases of the input signals. Careful examination of (4.14) will show that system will still not be solvable if we change the relative amplitudes of the input signals. For example let us assume that our original run had equal amplitudes of each of the three signals 1 Hz, 2 Hz, and 3 Hz. If we double the amplitude of the 1 Hz signal and half the amplitude of the 2 Hz signal then the response associated with \(H_3(-2, -1, 3)\) and \(H_3(-3, 1, 2)\) will not change. This is because the \(W\) ‘s will be the same. Thus the two rows we added to (4.14) are identical to the first two. If we double just the 1 Hz input signal and leave the 2 Hz and 3 Hz the same then the two rows that we add will just be a copy of the first two rows multiplied by 2.

If on the other hand we change the relative phases of the input signals say shift the 1 Hz signal by 90 degrees then the value of \(W^{(1)}\) will have the same amplitude but be shifted in phase by -90 degrees and the value of \(W^{(2)}\) will have the same amplitude but be shifted in phase by 90 degrees. Thus the matrix equation to be
solved will be of the form

\[
\begin{bmatrix}
W_{re}^{(1)} & -W_{im}^{(1)} & W_{im}^{(1)} & W_{re}^{(1)} \\
W_{re}^{(1)} & W_{im}^{(1)} & -W_{im}^{(1)} & W_{re}^{(1)} \\
W_{im}^{(1)} & W_{re}^{(1)} & -W_{im}^{(1)} & W_{re}^{(1)} \\
-W_{re}^{(1)} & W_{im}^{(1)} & -W_{re}^{(1)} & -W_{im}^{(1)}
\end{bmatrix}
\begin{bmatrix}
H_{re}^{(1)} \\
H_{im}^{(1)} \\
H_{re}^{(1)} \\
-H_{im}^{(1)}
\end{bmatrix}
= \begin{bmatrix}
Y_{g,3,rr} \\
0 \\
Y_{g,3,rr} \\
0
\end{bmatrix}
\]

(4.18)

which reduces to

\[
\begin{bmatrix}
2W_{re}^{(1)} & -2W_{im}^{(1)} & 0 & 0 \\
-2W_{re}^{(1)} & 0 & -2W_{im}^{(1)} & 0 \\
0 & 0 & 2W_{re}^{(1)} & -2W_{im}^{(1)}
\end{bmatrix}
\begin{bmatrix}
H_{re}^{(1)} \\
H_{im}^{(1)} \\
H_{re}^{(1)}
\end{bmatrix}
= \begin{bmatrix}
Y_{g,3,rr} \\
0 \\
Y_{g,3,rr}
\end{bmatrix}
\]

(4.19)

or

\[
\begin{bmatrix}
2W_{re}^{(1)} & -2W_{im}^{(1)} & 0 & 0 \\
-2W_{re}^{(1)} & 0 & -2W_{im}^{(1)} & 0 \\
0 & 0 & 2W_{re}^{(1)} & -2W_{im}^{(1)}
\end{bmatrix}
\begin{bmatrix}
H_{re}^{(1)} \\
H_{im}^{(1)} \\
H_{re}^{(1)}
\end{bmatrix}
= \begin{bmatrix}
Y_{g,3,rr} \\
0 \\
Y_{g,3,rr}
\end{bmatrix}
\]

(4.20)

which can be solved.

**FIPD with Unbalanced DC FIPD subset**

The case of an FIPD which contains a subset of an unbalanced DC FIPD is very similar to the case of an unbalanced DC FIPD. Each FIPD of this type will have a corresponding mate FIPD which is the same except that the frequencies of the unbalanced subset have been multiplied by -1. Since it takes at least 3 frequencies to make an unbalanced DC FIPD these types of FIPD’s must be at least order 4. For example both \(H_{i}(-2, -1, 3, 4)\) and \(H_{i}(-3, 1, 2, 4)\) contain the unbalanced DC FIPD \((2, -1, 3)\). As before the matrix equation cannot be solved without adding an additional run. The new run should be such that the resulting \(W\) phasor of the unbalanced DC FIPD is rotated by 90 degrees with respect to the other FIPD. In this example shifting the phase of the 4 Hz input would not work since it has the same sign in both FIPD’s.

**Other unique FIPD’s with same order and frequencies**

Beginning with third order there are some circumstances where there can be more than one FIPD of the same order which contain the same frequency magnitudes and have the same output frequency (i.e. add to the same positive frequency). For example both \(H^{(1)} = H(-1, 3, 3)\) and \(H^{(2)} = H(1, 1, 3)\) contain only the frequencies 1 and 3 and both add to 5. For this example (4.14) would be of the form

\[
\begin{bmatrix}
W_{re}^{(1)} & -W_{im}^{(1)} & W_{im}^{(1)} & W_{re}^{(1)} \\
W_{re}^{(1)} & W_{im}^{(1)} & -W_{im}^{(1)} & W_{re}^{(1)} \\
W_{im}^{(1)} & W_{re}^{(1)} & -W_{im}^{(1)} & W_{re}^{(1)} \\
-W_{re}^{(1)} & W_{im}^{(1)} & -W_{re}^{(1)} & -W_{im}^{(1)}
\end{bmatrix}
\begin{bmatrix}
H_{re}^{(1)} \\
H_{im}^{(1)} \\
H_{re}^{(1)} \\
-H_{im}^{(1)}
\end{bmatrix}
= \begin{bmatrix}
Y_{g,3,rr} \\
0 \\
Y_{g,3,rr} \\
0
\end{bmatrix}
\]

(4.21)
Here all four values of \( H \) are unknown and we have no additional information. As before we must simulate an additional run in order to calculate the transfer function values. For this case we choose to vary the relative amplitudes of the input frequency components instead of the relative phases. For our example let us add an additional run with the 3 Hz signal increased in amplitude by 1/3 and the 1 Hz signal decreased in amplitude by 1/3. Assuming the first run had both signals at equal amplitude, then the peak value of the second run will be the same and the ratio of the 3 Hz to 1 Hz amplitude will be 1/2. The amplitude of \( W^{(1)} \) will increase by a factor of \( \frac{2}{3} \times \frac{3}{2} \times \frac{4}{3} = \frac{2}{3} \) and the amplitude of \( W^{(2)} \) will decrease by a factor of \( \frac{2}{3} \times \frac{3}{2} \times \frac{4}{3} = \frac{16}{27} \). Now (4.21) becomes

\[
\begin{bmatrix}
W^{(1)}_{re} & -W^{(1)}_{im} & W^{(2)}_{re} & -W^{(2)}_{im} \\
W^{(1)}_{im} & W^{(1)}_{re} & W^{(2)}_{im} & W^{(2)}_{re} \\
\frac{32}{27}W^{(1)}_{im} & -\frac{32}{27}W^{(1)}_{re} & \frac{16}{27}W^{(2)}_{im} & -\frac{16}{27}W^{(2)}_{re} \\
\frac{32}{27}W^{(1)}_{re} & \frac{32}{27}W^{(1)}_{im} & \frac{16}{27}W^{(2)}_{re} & \frac{16}{27}W^{(2)}_{im}
\end{bmatrix}
\begin{bmatrix}
H^{(1)}_{re} \\
H^{(1)}_{im} \\
H^{(2)}_{re} \\
H^{(2)}_{im}
\end{bmatrix}
= \begin{bmatrix}
Y^{3,3,re} \\
Y^{3,3,im} \\
Y^{3,3,re} \\
Y^{3,3,im}
\end{bmatrix}
\]

(4.22)

Note that if we have more runs than we have unknown FIPD's then (4.14) will be underconstrained if the number of equations is greater than the number of unknowns. In our algorithm we would actually add two more runs for the case of \( H^{(1)} = H(-1,3,3) \) and \( H^{(2)} = H(1,1,3) \). One case where the 3 Hz signal was increased and one case where the 1 Hz signal was increased. This poses no mathematical problem since we solve (4.14) using least squares techniques.

**Maximum Input Amplitude and Assumed Order of System**

Since the set of extracted nonlinear transfer functions is intended to be used to simulate the subcircuit as a black box it is important to know the maximum peak amplitude in the black box simulation. If the amplitude is very small it may be sufficient to extract only the first order transfer function. For larger input amplitudes and for increasingly nonlinear circuits the extraction needs to include the higher order transfer functions. Furthermore it is important to extract all of the transfer functions in the same procedure. In other words one should not extract the first and second order functions and then go back and start the procedure again extracting just the third and fourth order functions while keeping the same first and second order functions extracted in the first procedure. Remember that when we solve (4.6) we are essentially performing a polynomial least squares fit. In order to get the best fit and thus the most accurate behavioral model over the entire input range we need to extract all transfer functions in the same procedure.

On the other hand if one wanted to trade off accuracy in one amplitude range for accuracy in another amplitude range then one could use the two procedure method. For example suppose the behavioral model were to be used primarily in two modes (i.e. small signal (i.e. linear) simulation and large signal (i.e. nonlinear) simulation. The small signal simulation would be much faster and would only be valid for small inputs to the circuit. Whereas the large signal simulation would accurately simulate the nonlinear effects. For this case another possibility would be to use standard AC simulation to extract the first order transfer function. This would be much faster
than the running of the extraction procedure and specifying order=1. Section 5.5 will show example results of this process.

Another important aspect of the extraction procedure concerns the peak amplitude of the test signal. We want to be sure that the test signals used in the extraction procedure have a peak absolute value amplitude equal to that specified by the user at the beginning of the extraction process. This is to ensure that the extracted model is valid for the range for which it will be used. Thus for each run, the peak input amplitude absolute value will equal that specified by the user. This is easily accomplished by searching the input amplitude waveform to find the peak (absolute value) and then scaling the input waveform accordingly. But we also want to maximize the energy to the circuit to get the maximum energy output and thus obtain a more accurate extraction. For any measurement or simulation system there will be some unwanted noise injected into the measurement or simulation. Larger power input signals help to reduce the effects of this noise.

In order to maximize input energy, we adjust the relative phases of the input tones to minimize the absolute value of the input peak before we scale the signal as discussed above. In our procedure, we will use tree annealing [90] to accomplish this minimization.

### 4.2.3 Algorithm Flow Description

First we will examine the box labeled “Initialize” in figure 4.1. Figure 4.2 gives a more detailed description of this block. First the data from a file created by the user is read. A sample of this file is given in figure 4.3. The first line of the file is a description that is used to annotate other files. The second line specifies the maximum input value to the circuit. The value is the peak absolute voltage of the input. The maximum order specified on the next line is the assumed order of the system. The next line specifies the number of additional simulations to be run for each set of relative input phasors. As explained in section 4.2, the given set of input frequencies will be simulated \( n \) times; each with a different input amplitude but with the same relative input phasors. The output is decomposed into the various \( m \)th order responses. The decomposition is basically a series of polynomial curve fits where the polynomial has the order equal to the assumed order of the system. The number of different amplitudes \( m \) must therefore be equal to or greater than the assumed order of the system. The number of extra simulations is just \( m - n \) which must be a non-negative integer. A larger number will require more simulations and more extraction calculations. If no extra simulations are used, then the number of points for the polynomial would equal the order of the polynomial (the zeroth order component being known) and thus the polynomial would always pass through all of the points given. If the number of different input amplitude simulations is greater than the order of the system then there are two advantages:

1. The measurement noise will have a reduced effect on the calculation. Even though the circuit output voltage used is from a simulation as opposed to a lab measurement, there is still noise present in the values.

2. The value of the assumed order can be verified. Though this is not a rigorous verification, the goodness of fit of the oversampled polynomial fit is an
Figure 4.2: High level flow chart for the procedure to initialize the extraction procedure. This flow chart represents the box labeled “Initialize” in figure 4.1
low pass filter with pole at 3MHz
max_input: 1.0
max_order: 3
xtra_runs: 1
fn: lp1
test_model_lib: /u/lunsford/asxlib/lowpass
runcontrol: /u/lunsford/asxlib/stdrun
acruncontrol: /u/lunsford/asxlib/acrun
loadlib: /u/lunsford/asxlib/libspi31.o
num_extract_runs: -1
tot_runs: 18
run_num: -1
calc_dc: yes
out_dc_offset: 0
num_freq: 4
freq[0]: 0
freq[1]: 1000
freq[2]: 2000
freq[3]: 3000

Figure 4.3: The input file for low-pass RC filter.
indication of the validity of the assumed order.

The next five lines of figure 4.3 are related to the simulator used ASTAP [113]. The first is the file name containing the circuit description, the second is the name of the directory that contains this file and other files that contain needed subcircuit models. The fourth and fifth are file names that contain run control statements for the simulations. And the last of the five lines gives the name of a file that contains compiled fortran used to model the nonlinear characteristics of the circuit elements.

The next three lines, num_extract_runs, tot_runs, and run_num, are set by the program as it goes through the various steps and are not set by the user. These values are written to the file during the initialization phase of the program. The following two lines, calc dc and out dc_offset, are used to calculate the value of $H_0$, which is just DC output voltage for the circuit when the input value is a constant value of zero. If calc dc is set to “yes” then the value of out dc_offset will be calculated by the program by running a simulation with an input voltage of zero volts. If the value of calc dc is set to “no” then the program will take the value entered for out dc_offset and use it as the value of $H_0$.

Lastly, the discrete values of the assumed input frequencies are given, first the number of frequencies then each frequency in hertz.

The third block in figure 4.2 is for determining how many simulations are needed and what the input conditions will be for each simulation. This is a complicated procedure and is shown in more detail in figure 4.4. The list of basic runs created in the first block is just all the different possible combinations of the input frequencies in groups of 1, 2, …, $n$ where $n$ is the assumed order of the system. Table 4.3 gives an example of the basic runs that correspond to figure 4.3. Run 0 is used to determine the DC operation point for $H_0$. For table 4.3 we have chosen the amplitudes of each input phasor to be the maximum allowed amplitude divided by the number of input phasors, thus guaranteeing that the peak amplitude will not be greater than the maximum specified in the “max input:” field in the input file. Table 4.3 gives just a starting point for determining all of the inputs for the simulations. There are four adjustments that are made to this table. They are (a) adjusting the relative phases of the input phase for the minimum case for value of $H$; (b) adding extra runs; (c) shifting the input in time such that the average value of the signal occurs at time equal to zero and (d) adjusting input amplitudes for exact peak value.

Table 4.4 is table 4.3 with the addition of the extra runs needed to calculate all of the nonlinear transfer functions. Run 8 has been duplicated with a 90 degree phase change in the 1 MHz signal because run 8 determines the $H$ value at an unbalanced DC FIPD. Run 9 is duplicated with different input phasor amplitudes because $H_3(1, 1, 3)$ and $H_3(-1, 3, 3)$ are uniquely determined only with two runs as they both have the same output frequency. This type of situation was discussed above on page 60 under the heading “Other unique FIPD’s with same order and frequencies”. Run 14 determines an $H$ value at an unbalanced DC FIPD $(-2, -1, 3)$ thus a second simulation was added that shifted one of the input phasors by 90 degrees. The actual algorithm used to determine the cases where additional runs are required is shown in figures 4.5 and 4.6.
Figure 4.4: Flow chart for determining simulation conditions. This represents the block labeled “Determine Simulation Conditions” in figure 4.2.
<table>
<thead>
<tr>
<th>Run Number</th>
<th>Input Amplitudes</th>
<th>Run Number</th>
<th>Input Amplitudes</th>
<th>Run Number</th>
<th>Input Amplitudes</th>
<th>Run Number</th>
<th>Input Amplitudes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 Hz</td>
<td>1 MHz</td>
<td>2 MHz</td>
<td>3 MHz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0 0</td>
<td>0 1</td>
<td>0 0</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0 0</td>
<td>0 0</td>
<td>0 1</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.5 0</td>
<td>0 0.5</td>
<td>0 0</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.5 0</td>
<td>0 0</td>
<td>0 0.5</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.5 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0 0</td>
<td>0 0.5</td>
<td>0 0</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0 0</td>
<td>0 0.5</td>
<td>0 0</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0.5</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.3333 0</td>
<td>0 0.3333</td>
<td>0 0.3333</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.3333 0</td>
<td>0 0.3333</td>
<td>0 0.3333</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.3333 0</td>
<td>0 0</td>
<td>0 0.3333</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0 0</td>
<td>0 0.3333</td>
<td>0 0.3333</td>
<td>0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3: Basic list of runs for third order and $f = 0\text{MHz} \Gamma 2\text{MHz} \Gamma 3\text{MHz}$ without peak minimization (without additional needed runs), without time shift and without amplitude adjustment.
<table>
<thead>
<tr>
<th>Run Number</th>
<th>Input Amplitudes</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 Hz</td>
<td>1 MHz</td>
<td>2 MHz</td>
<td>3 MHz</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0.0000</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>-0.4330</td>
<td>-0.2501</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0001</td>
</tr>
<tr>
<td>11</td>
<td>0.3333</td>
<td>0</td>
<td>0.2357</td>
<td>-0.2357</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0.3333</td>
<td>0</td>
<td>-0.2886</td>
<td>-0.1667</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0.3333</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-0.1667</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0.2232</td>
<td>-0.2476</td>
<td>0.0525</td>
</tr>
</tbody>
</table>

Table 4.4: Basic list of runs for third order and $f = 0\gamma 1 MHz \gamma 2 MHz \gamma 3 MHz$, with peak minimization, without additional needed runs, without time shift, and without amplitude adjustment. A value of 0.0000 indicates a nonzero number whose absolute value is less than 0.0001.
Figure 4.5: Flow chart for determining indeterminant conditions. This represents the block labeled “Determine the indeterminent conditions” in figure 4.4.
## Table 4.5: List of runs for third order and $f = 0/1MHz/2MHz/3MHz$. with peak minimization/, with additional needed runs/, without time shift/ and without amplitude adjustment. A value of 0/0000 indicates a nonzero number whose absolute value is less than 0/0001.

<table>
<thead>
<tr>
<th>Run Number</th>
<th>0 Hz Real</th>
<th>0 Hz Imag.</th>
<th>1 MHz Real</th>
<th>1 MHz Imag.</th>
<th>2 MHz Real</th>
<th>2 MHz Imag.</th>
<th>3 MHz Real</th>
<th>3 MHz Imag.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>8a</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0.0000</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8b</td>
<td>0</td>
<td>0</td>
<td>0.3535</td>
<td>0.3535</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9a</td>
<td>0</td>
<td>0</td>
<td>-0.4330</td>
<td>-0.2501</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>9b</td>
<td>0</td>
<td>0</td>
<td>-0.5773</td>
<td>-0.3335</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.3333</td>
</tr>
<tr>
<td>9c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-0.2886</td>
<td>-0.1667</td>
<td>0</td>
<td>0</td>
<td>0.6666</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0001</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0.3333</td>
<td>0</td>
<td>0.2357</td>
<td>-0.2357</td>
<td>0</td>
<td>0.3333</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0.3333</td>
<td>0</td>
<td>-0.2886</td>
<td>-0.1667</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.3333</td>
</tr>
<tr>
<td>13</td>
<td>0.3333</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-0.1667</td>
<td>-0.2886</td>
<td>0</td>
<td>0.3333</td>
</tr>
<tr>
<td>14a</td>
<td>0</td>
<td>0</td>
<td>0.2232</td>
<td>-0.2476</td>
<td>0.0525</td>
<td>-0.3292</td>
<td>0</td>
<td>0.3333</td>
</tr>
<tr>
<td>14b</td>
<td>0</td>
<td>0</td>
<td>0.2232</td>
<td>-0.2476</td>
<td>0.3292</td>
<td>0.0525</td>
<td>0</td>
<td>0.3333</td>
</tr>
</tbody>
</table>

Table 4.5 gives the values of the input phasors of table 4.4 after the phase has been adjusted to maximize the input power relative to the maximum peak amplitude. Since we will later adjust the overall amplitude of the step here is just a minimization of the peak voltage as a function of the relative phases of the input vectors. Tree annealing [90] is well suited for the phase adjustment and is used in this procedure. Note that the runs added in table 4.4 required a certain phase relationship to another run for example the phase of the 1 MHz phasor in run 8b is the phase of the 1 MHz phasor in run 8a shifted by 90 degrees. Thus the same relative phase adjustment for maximizing power calculated for 8a is used in 8b. Table 4.6 gives table 4.5 with the input delayed such that the value at time equal zero is the average value of the signal. This is equivalent to shifting the time such that the value of all of the non-DC input phasors add to zero. This adjustment can help to speed the
Figure 4.6: Flow chart for determining additional required runs. This represents the block labeled “Determine additional required runs” in figure 4.5.
<table>
<thead>
<tr>
<th>Run Number</th>
<th>Input Amplitudes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 Hz</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
</tr>
<tr>
<td>8a</td>
<td>0</td>
</tr>
<tr>
<td>8b</td>
<td>0</td>
</tr>
<tr>
<td>9a</td>
<td>0</td>
</tr>
<tr>
<td>9b</td>
<td>0</td>
</tr>
<tr>
<td>9c</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0.3333</td>
</tr>
<tr>
<td>12</td>
<td>0.3333</td>
</tr>
<tr>
<td>13</td>
<td>0.3333</td>
</tr>
<tr>
<td>14a</td>
<td>0</td>
</tr>
<tr>
<td>14b</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.6: List of runs for third order and $f = 0 \Gamma 1 MHz \Gamma 2 MHz \Gamma 3 MHz$. With peak minimization with additional needed runs with time shift and without amplitude adjustment. A value of 0.0000 indicates a nonzero number whose absolute value is less than 0.0001.
simulations if a shooting method simulator is used. If the circuit being simulated has widely varying time constants and if the input frequencies are high compared to the inverse of the smallest time constant of the circuit, then this adjustment can significantly reduce the simulation time. This reduction in simulation time can be understood by looking at the example of a series resistor–capacitor pair in the circuit. Given that the pole associated with the RC pair is much lower in frequency than the input frequencies, then the voltage across that capacitor will remain relatively steady through the simulation. Since shooting methods first solve for the DC solution to the circuit using the input conditions at time equal zero, the value of the voltage across our example capacitor will often be the same for the DC solution as for the steady-state solution. Thus if we adjust the time such that the DC solution at time equal zero is the average, we can reduce the simulation time for some circuits if a shooting method simulator is used. This will be true for the case of a linear circuit where the input frequencies are much higher than the pole associated with the example resistor–capacitor pair but for the general case of nonlinear circuits, this is not guaranteed. For memoryless circuits, this time shift has no effect on the simulation time.

Table 4.7 gives table 4.6 with the input phasor amplitudes adjusted so that the peak absolute value matches the value of the "max_input:" field in the input file. This adjustment is accomplished by calculating all of the time-domain values of the input at a sufficiently small time step and taking the largest absolute value. This value is then compared to the value given in the input file and the phasor amplitudes are adjusted accordingly.

Now that the “Initialize” block in figure 4.1 has been completed, the next step is to create the simulation files. These files depend on the simulator to be used and have the input conditions as described by the runs calculated in the previous step. Note that for each run, several simulations are performed all with the same relative phases but with different input amplitudes. The last simulation for each run is the one with largest input amplitude and the peak absolute value of the input for this simulation equals that specified in the input file.

After the simulation files are created, all of the simulation files are simulated. This section uses the majority of the computer resources and can take days or weeks depending on several factors. The biggest factor is the speed of the circuit simulator being used relative to the complexity of the circuit being modeled. The more complex the subcircuit, the longer it takes to simulate the circuit. A second factor is the value chosen for “xtra_runs” in the input file. This is a relatively small factor since the total number of runs varies linearly with the sum of the “xtra_runs” parameter summed with the “max_order” parameter. Thirdly, both the “max_order” and the “num_freq” parameters in the input file together determine how many points of $H$ values will be calculated. This is an extremely nonlinear relationship and is the biggest factor in determining how many runs must be simulated. Lastly, for a shooting method type of circuit simulator the actual values of the input frequencies can drastically effect the simulation time. Since shooting methods require that the input be periodic, the smallest period for two almost incommensurable frequencies can be very large. For example, if the input frequencies were chosen to be 2 Hz and 4 Hz then the input signal is periodic in a time of 0.5 s. But if the input frequencies were chosen to be 2.01 and 4.01 then the input signal is periodic in a time of 100 s.
<table>
<thead>
<tr>
<th>Run Number</th>
<th>Input Amplitudes</th>
<th>0 Hz</th>
<th>1 MHz</th>
<th>2 MHz</th>
<th>3 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8a</td>
<td>0</td>
<td>0</td>
<td>0.5697</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>8b</td>
<td>0</td>
<td>0</td>
<td>0.3536</td>
<td>0.3536</td>
<td>0.0000</td>
</tr>
<tr>
<td>9a</td>
<td>0</td>
<td>0</td>
<td>-0.5636</td>
<td>-0.3255</td>
<td>0</td>
</tr>
<tr>
<td>9b</td>
<td>0</td>
<td>0</td>
<td>-0.8053</td>
<td>-0.4652</td>
<td>0</td>
</tr>
<tr>
<td>9c</td>
<td>0</td>
<td>0</td>
<td>-0.3440</td>
<td>-0.1987</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0001</td>
</tr>
<tr>
<td>11</td>
<td>0.4713</td>
<td>0</td>
<td>0.3333</td>
<td>-0.3333</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0.3942</td>
<td>0</td>
<td>-0.3414</td>
<td>-0.1972</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0.3801</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-0.1901</td>
</tr>
<tr>
<td>14a</td>
<td>0</td>
<td>0</td>
<td>0.3373</td>
<td>-0.3742</td>
<td>0.0794</td>
</tr>
<tr>
<td>14b</td>
<td>0</td>
<td>0</td>
<td>0.2421</td>
<td>-0.2686</td>
<td>0.3571</td>
</tr>
</tbody>
</table>

Table 4.7: Final list of runs for third order and $f = 0 \Gamma 1 \text{MHz} \Gamma 2 \text{MHz} \Gamma 3 \text{MHz}$. Peak minimization/additional needed runs/time shift/and amplitude adjustment are all included. A value of 0.0000 indicates a nonzero number whose absolute value is less than 0.0001.
After the simulations are complete, the values of $H$ can finally be calculated. This process is complicated but very similar to the algorithm used to determine the needed additional runs in the initialization phase. Figure 4.7 gives the overall flow of the calculation procedure. The first step deals with multiple runs with the same input frequencies. This step will be discussed later. For each run, all of the FIPD’s and their corresponding output frequencies are calculated. This calculation is the same as performed in the initialization procedure. The next step is to take the time-domain output waveform and convert it to frequency-domain phasors using the FFT algorithm. Then each output phasor $Y_q$ in a run is decomposed into the order responses $Y_{q,1}Y_{q,2}...Y_{q,n}$. This procedure is just a polynomial least squares fit and is done with the SVD algorithm [114]. Then for each order response the $H$ values are calculated if possible. The details of this block are shown in figure 4.8. For each FIPD that corresponds to a particular output frequency and order, first the previously calculated values of $H$ are checked to see if the value of $H$ at this FIPD is known. If it is known, then the response due to this FIPD is calculated and subtracted from $Y_q$. If not, then the unknown FIPD is added to a list. After all of the FIPD’s have been checked, the length of the list is checked. If the length is zero, then all of the values of $H$ contributing to the response at this order and at this output frequency have already been calculated. If there is only one FIPD in the list, then the value of $H$ at this point can be found by solving (4.14). If there is more than one FIPD in the list, then there are more than two columns in the corresponding decomposition matrix equation (4.14) and we need information of more than one run. Therefore we save the current value of $Y_{q,i}$ and the set of unknown FIPD’s to a linked list. This linked list is used in the box marked “Deal with multiple runs with the same input frequencies” in figure 4.7. Figure 4.9 gives the detailed algorithm flow for this procedure. Remember that the order of runs is important. We first simulated all of the runs that had a single frequency input, then a two frequency input, and so on. The additional run(s) that are added to the basic list during the “Add needed runs” block of figure 4.5 have the same input frequencies as one of the runs in the basic list. These additional run(s) are added adjacent to that run. Thus when a list of FIPD’s is added to the linked list in figure 4.8 it is because there is more than one run with the same input frequencies. Thus the matrix in (4.14) has more than two columns. The information needed to formulate and solve (4.14) is contained in the linked list.

4.2.4 Summary

The extraction procedure consists of three major steps: initialization, simulation, and calculation. First the number of simulations and the conditions for these simulations must be determined. This determination is done by first determining all the different possible combinations of the input frequencies taken at most $n$ at a time where $n$ is the assumed order of the system. For each combination of input frequencies, at least $n$ different simulations are required. Each with different input amplitudes, keeping constant the relative phases and relative amplitudes of the components of the input frequencies. Furthermore, for systems of order three and higher, additional simulations with different relative phases and different relative amplitudes of the components of the input frequencies may be required. Secondly, the actual
Figure 4.7: High level flow chart for the procedure to go through all of the runs and calculate the transfer function once the simulations are finished.
Figure 4.8: Flow chart for the procedure to calculate the nonlinear transfer functions. This represents the block labeled “Calculate $H$ values” in figure 4.7.
Figure 4.9: Flow chart for the procedure used to deal with multiple runs with the same input frequencies.
simulations are run and the output values saved in files. And lastly, the results from the simulations are used to calculate the Volterra nonlinear transfer functions.

4.3 Conclusion

The extraction procedure described in this chapter provides an automatic way to first determine the needed simulations, perform the needed simulations, and then extract the Volterra nonlinear transfer function values from the simulation results.

The compute time needed for the entire process is mainly dependent on the number of input frequencies and the assumed order of the system. The complexity of the system and the efficiency of the circuit simulator being used also affect the amount of required compute resource. Typically, the actual simulation is the part of the procedure that uses the majority of the needed compute resource.

The resulting calculated points of the Volterra nonlinear transfer function are stored in files and can be used to calculate the steady-state response of the system as long as two conditions are satisfied. First, the input amplitude must not be greater than the maximum amplitude specified for the extraction. And secondly, the input spectrum must be comprised solely of discrete frequency values that belong to the list of input frequencies specified for the extraction.
Chapter 5
Verification

5.1 Introduction

This chapter presents examples and measurements of the techniques proposed in chapters 3 and 4. Various examples verify the individual techniques developed in this thesis. A final example provides verification of the integration of the techniques in the analysis of a subsystem using a behavioral model developed in a systematic manner from circuit level simulations. First an example of bivariate generalized power series analysis as proposed in chapter 3 is given. The first example is a ring diode mixer that is modeled as a two input voltage controlled voltage source. Results of the bivariate power series analysis are compared to results from a time domain circuit simulator. Secondly a MESFET amplifier with a bivariate series characterization of the current source in the equivalent circuit model is examined. By using the relationship between generalized power series and bivariate Volterra series the circuit is analyzed using signal flow analysis for bivariate Volterra series. Comparison to lab measurements show good results.

Finally examples of the extraction technique proposed in chapter 4 are given. Circuits with known Volterra nonlinear transfer functions are used to quantify the accuracy of the extraction technique. Lastly a nonlinear equalization circuit is used as an example of a circuit that does not have a known Volterra series representation. The output predicted by the extracted nonlinear transfer functions is compared to that of the original circuit level simulation.

5.2 Bivariate Generalized Power Series – Ring Diode Mixer

An example of a circuit that can be modeled using bivariate generalized power series analysis is the diode ring mixer shown in Fig. 5.1. In order to calculate a behavioral model for this entire circuit we need to find an approximation to the output voltage $V_{IF}$ as a function of the two input voltages $V_{LO}$ and $V_{RF}$. Moreover this approximation needs to be in the form of (3.3). Thus the entire circuit is modeled as a two-input voltage controlled voltage source.

The current-voltage relationship for the individual diodes are modeled by the Shockley diode equation:

$$I_j = I_{0j} (e^{V_{jV_i}/V_t} - 1)$$

(5.1)

Where $I_j$ is the current through diode $D_j$, $V_{jV_i}$ is the voltage across $D_j$, $I_{0j}$ is the saturation current for diode $j$, $V_t = kT/q$ is Boltzmann's constant and $q$ is the ideality factor.
Assuming that the transformers are ideal, the KCL and KVL equations for this circuit reduce to:

\[
V_1(t) = \frac{1}{2}V_{LO}(t) - \frac{1}{2}V_{RF}(t) - V_{IF}(t) \\
V_2(t) = \frac{1}{2}V_{LO}(t) + \frac{1}{2}V_{RF}(t) + V_{IF}(t) \\
V_3(t) = \frac{1}{2}V_{RF}(t) - \frac{1}{2}V_{LO}(t) - V_{IF}(t) \\
V_4(t) = V_{IF}(t) - \frac{1}{2}V_{LO}(t) - \frac{1}{2}V_{RF}(t) \\
I_1(t) - I_2(t) + I_3(t) - I_4(t) - \frac{V_{IF}(t)}{R_{IF}} = 0
\] (5.6)

The explicit notation of time \( t \) will be dropped for simplification.

Equations (5.2) - (5.6) were solved for \( V_{IF} \) by means of the computer-algebra program MAPLE [115] to perform the algebra and calculus operations. The MAPLE code used is included in appendix C. Although all of the operations are straightforward, some of the equations are so large that manual calculations are impractical. As an intermediate step, a function \( f \) having the desired solution at \( f = 0 \) was obtained:

\[
f = I_{01} \left( e \left( \frac{1}{2}V_{LO} - \frac{1}{2}V_{RF} - V_{IF} \right)/V_i - 1 \right) \\
- I_{02} \left( e \left( \frac{1}{2}V_{LO} + \frac{1}{2}V_{RF} + V_{IF} \right)/V_i - 1 \right)
\]
Since we want a closed form solution for $V_{IF}$ we now estimate $f$ by a third degree Taylor series in $V_{IF}$.

$$f \approx f(V_{IF} = 0) + f'(V_{IF} = 0) V_{IF} + \frac{1}{2} f''(V_{IF} = 0) V_{IF}^2 + \frac{1}{6} f'''(V_{IF} = 0) V_{IF}^3 \quad (5.8)$$

This third order polynomial in $V_{IF}$ can be solved in closed form [111]. In general all of the three solutions to this third order equation two are complex and one is real. Using the real solution we get a very large equation which we will call $g$. The full solution of $g$ is too large to be included here but is given in appendix C for completeness. The function $g$ has the form $V_{IF} = g(V_{RF}, V_{LO}, I_{01}, I_{02}, I_{03}, I_{04}, V_t)$ to which a bivariate power series of the form

$$g(V_{RF}, V_{LO}) = \sum_{\sigma=0}^{9} \sum_{\rho=0}^{9} a_{\sigma, \rho} V_{RF}^\sigma V_{LO}^\rho \quad (5.9)$$

can be fitted using least square techniques over a given range. Thus (5.9) has the form of (3.3) with $b_k = 1 \Gamma d_k = 1 \Gamma \lambda_k = 0$. For our investigation we used the voltage ranges $-0.6 \leq V_{LO} \leq 0.6 \Gamma$ and $-0.2 \leq V_{RF} \leq 0.2$. and the RMS fit error was less than 1%. For the ring mixer of Fig. 11 $\eta = 1 \Gamma I_{01} = 1.00 \ pA \ I_{02} = 1.01 \ pA \ I_{03} = 1.03 \ pA \ I_{04} = 1.06 \ pA \ V_t = 0.02569 \ \text{Volts}$. and the transformers are ideal and have 1:1 turns ratios. The solid lines in Fig. 5.2 give the steady-state conversion power gain of $P_{IF}/P_{RF}$ for several LO powers. Since the power series (5.9) is valid only for a given range of the inputs the macromodel is only valid within that range. The circles are simulation results using the time domain simulator ASTAP [113] and show good agreement.

The dashed lines in Fig. 5.2 demonstrate an important property of bivariate GPSA. Since the summation given by (3.114) contains powers of the phasor components as given by (3.16) any arbitrary input-output transfer characteristic is of the form

$$Y_q = \sum_i H_{i,j}(X_j) \quad (5.10)$$

Where $H_{i,j}$ is an $i$th order nonlinear transfer function for the input phasor $X_j$ (or set of $X_j$’s). In general $H_{i,j}$ is a function of the other input phasors. The dashed lines in Fig. 5.2 give (a) first order $H_1 \Gamma$ and (b) third order $H_3 \Gamma$ transfer characteristics for the IF phasor when $P_{LO} = 2.0 \ \text{dBm}$. In our example the even-order transfer functions are zero. Note that in Fig. 5.2 the vertical axis is gain not output amplitude; thus the lines represent the 0th order and 2nd order transfer characteristics for the gain corresponding to the 1st and 3rd order transfer characteristics for output amplitude. For the range modeled all higher order transfer functions are negligible. Adding (a) and (b) yields a value within 1% of the total characteristic (solid line) shown for $P_{LO} = 2 \ \text{dBm}$. Thus a simple behavioral model is obtained by using only lower order powers of the input. Here the nonlinear RF to IF characteristics can be described by the sum of two components. Each component can be represented as a linear function when expressed in log-log form.
Figure 5.2: Conversion Gain for different LO power levels. Solid lines are bivariate GPSA calculations. Circles are ASTAP (time domain simulation) calculations. Dashed lines are the first two terms of the decomposition of the solid line for $P_{LO} = 2.0$ dBm. (a) is the constant term: $RF^0$. (b) is the second order term: $RF^2$. 
One form of Volterra series analysis of nonlinear circuits casts the nonlinear circuit into block diagram form each block being described by linear transfer functions for linear subcircuits and by Volterra nonlinear transfer functions for nonlinear subcircuits [116]. Here we apply this technique to the Volterra series–based analysis of a MESFET amplifier and compare the simulated results to experimental results.

The MESFET amplifier used here was previously analyzed by Chang et al. [20].

The model schematic is shown in Fig. 5.3 and the measured parameter values are given in table 5.1. Using the substitution theorem [79] Section 2.2.1] the equivalent signal flow graph of the circuit is shown in Fig. 5.4. The input is the value of the source voltage $V_{IN}$. This signal passes through the linear systems $G_{IN-GS}$ and $G_{IN-DS}$. Each of these systems is characterized by all of the linear components of the circuit. $V_{GS}$ and $V_{DS}$ are the inputs to the two-input nonlinear block $B$ characterized by $\tau$ and table 5.2. Using (3.106) we can directly calculate the bivariate Volterra
### Table 5.1: Element values for MESFET circuit

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{g1}$</td>
<td>0.1386 pF</td>
</tr>
<tr>
<td>$L_g$</td>
<td>0.69414 nH</td>
</tr>
<tr>
<td>$C_{g2}$</td>
<td>0.30707 pF</td>
</tr>
<tr>
<td>$R_g$</td>
<td>2.9Ω</td>
</tr>
<tr>
<td>$R_s$</td>
<td>2.4Ω</td>
</tr>
<tr>
<td>$L_s$</td>
<td>0.00323 nH</td>
</tr>
<tr>
<td>$R_{d}$</td>
<td>5.3Ω</td>
</tr>
<tr>
<td>$L_d$</td>
<td>0.41143 nH</td>
</tr>
<tr>
<td>$C_{d2}$</td>
<td>0.09012 pF</td>
</tr>
<tr>
<td>$C_{d1}$</td>
<td>0.00341 pF</td>
</tr>
<tr>
<td>$R_1$</td>
<td>10Ω</td>
</tr>
<tr>
<td>$\tau$</td>
<td>6.56 ps</td>
</tr>
<tr>
<td>$C_{ds}$</td>
<td>0.25050 pF</td>
</tr>
<tr>
<td>$C_{gs}$</td>
<td>0.50150 pF</td>
</tr>
<tr>
<td>$C_{gd}$</td>
<td>0.08637 pF</td>
</tr>
</tbody>
</table>

### Table 5.2: Coefficients for bivariate power series of $I_{ds}$ about DC operating point

<table>
<thead>
<tr>
<th>Order of $V_{gs}$ Term</th>
<th>Order of $V_{ds}$ Term</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0211092</td>
<td>0.0467136</td>
<td>0.000004985</td>
<td>0.00026358</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.0689287</td>
<td>0.00307239</td>
<td>-0.000616581</td>
<td>0.00244648</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.0552606</td>
<td>-0.0158083</td>
<td>0.000336844</td>
<td>0.00931433</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-0.0241834</td>
<td>-0.0132546</td>
<td>0.00541917</td>
<td>0.0127004</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-0.0328935</td>
<td>0.0236952</td>
<td>-0.00941775</td>
<td>-0.0136877</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.00865757</td>
<td>0.0340615</td>
<td>-0.0131895</td>
<td>-0.0366089</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.00689521</td>
<td>-0.00467676</td>
<td>0.0034243</td>
<td>-0.00190339</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>-0.00616748</td>
<td>-0.0241809</td>
<td>0.00586464</td>
<td>0.0253214</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>-0.0029458</td>
<td>-0.00895554</td>
<td>0.00252446</td>
<td>0.0105268</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Order of $V_{gs}$ Term</th>
<th>Order of $V_{ds}$ Term</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-0.0006759</td>
<td>-0.000138752</td>
<td>-0.000008659</td>
<td>0.0000154853</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-0.000132799</td>
<td>-0.000942032</td>
<td>0.0000306236</td>
<td>0.000105969</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-0.00024949</td>
<td>-0.00326047</td>
<td>0.000315678</td>
<td>0.000336661</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-0.00146277</td>
<td>-0.00362775</td>
<td>0.000297059</td>
<td>0.000290322</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.00546728</td>
<td>0.00528516</td>
<td>-0.000791546</td>
<td>-0.000575213</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.00563955</td>
<td>0.0110368</td>
<td>-0.000849452</td>
<td>-0.000954538</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-0.00231212</td>
<td>-0.000264063</td>
<td>0.000324688</td>
<td>0.00010378</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>-0.00426292</td>
<td>-0.00775008</td>
<td>0.000655445</td>
<td>0.000683864</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>-0.001237</td>
<td>-0.00305257</td>
<td>0.00019882</td>
<td>0.000254013</td>
<td></td>
</tr>
</tbody>
</table>
Figure 5.4: Nonlinear signal flow graph for MESFET amplifier. $G_{IN-GS} \Gamma G_{IN-DS} \Gamma G_{FB-GS} \Gamma G_{FG-DS} \Gamma$ and $G_{OUT}$ are linear systems. $B$ is a bivariate nonlinear system.
nonlinear transfer function and thus can calculate the value $I_{DS}$. The value of the output voltage $V_L$ is the output of the linear system $G_{OUT}$ which has as its input $I_{DS}$. The two linear systems $G_{FB-DS}$ and $G_{FB-GS}$ provide the feedback to the two inputs of the bivariate block.

The aim now is to solve for the steady-state output of the system given a steady-state input signal $V_{IN}$. The two feedback paths do not allow a straightforward closed form solution. However, the steady-state output of the system can be solved relatively easily by iteration for low input powers. For the first iteration, we assume that the feedback contributions are negligible and calculate the system response. Then using the new calculated value of $I_{DS}$ we calculate a new $V_{DS}$ and $V_{GS}$ and use the new values for the next iteration. This process is continued until there is no appreciable change of any of the values from one iteration to the next.

Figure 5.5 gives the results of a two-tone test for the circuit using two equal-amplitude signals at input frequencies of 2.35 GHz and 2.4 GHz. The horizontal axis is the input power for one of the tones and the vertical axis is the output power at the fundamental frequency of 2.35 GHz and the image frequency of 2.3 GHz. The points are experimentally measured values and the solid lines are the simulated values and show good agreement. These simulated values also agree closely with those given by Chang et al. [20]. Note that a linear simulation would predict no power at 2.3 GHz.

Another technique for solving nonlinear circuits that is based on the work of Volterra is the method of nonlinear currents [116]. This technique has the advantage of a closed form solution to a nonlinear system which consists of (a) linear subsystems and (b) simple nonlinear systems described by univariate power series. Strictly speaking the generalized form of the power series (which includes the time delays $\tau$) cannot be used with this analysis. However, a pure delay can be lumped into the linear systems and thus a generalized power series can be adapted to this analysis. This method has been used to analyze MESFET amplifiers [81] but since the nonlinear blocks are characterized by univariate power series the $V_{DS}$-$V_{GS}$ cross product terms of the bivariate power series are ignored. Thus the coefficients in table 5.2 that are neither in the first row nor the first column are assumed to be negligible. The work presented here can be used to extend the method of nonlinear currents to include more general nonlinearities.

### 5.4 New Volterra Extraction

#### 5.4.1 Linear RC Filter

The single-pole low pass filter circuit shown in figure 5.6 was used to test the extraction procedure described in chapter 5. The circuit was simulated using the ASTAP circuit simulator [113]. This circuit is linear and thus the nonlinear transfer function is equal to zero for orders higher than 1. The input file used was that shown in figure 4.3. Note that we assumed an order of 3 and thus we expect to extract values equal to zero for $H_2$ and $H_3$. Figures 5.7 and 5.8 show the tables that were extracted using the procedure. These tables show the general form of the output files. This first line contains two numbers, the number of inputs and the number of outputs. Since the functions $H_n$ are complex, the number of outputs is always two, one for
Figure 5.5: The results of a two tone test for the circuit using two equal-amplitude signals at input frequencies of 2.35 GHz and 2.4 GHz. The horizontal axis is the input power for one of the tones and the vertical axis is the output power at the fundamental frequency of 2.35 GHz and the image frequency of 2.3 GHz. The solid lines are the simulated values and the points are experimentally measured values.
the real part and one for the imaginary part. The number of inputs corresponds to the order of the nonlinear transfer function. The general form of the tables is shown in table 5.3.

In figure 5.7 the value of $H_0$ is equal to 0 as expected. The known linear characteristics are shown by the values of $H_1$ and show the expected roll off of a single pole at 3MHz. The point for $H_2$ in figure 5.7 and the points for $H_3$ in figure 5.8 are close to zero as expected. These values are not exactly equal to zero due to computational noise present in the simulation results.

5.4.2 Power Series Device, 7th order

Now that a linear circuit with memory has been used to test the extraction algorithm we move onto the case of a nonlinear system with \textit{apriori} known values of $H$. In this example we will use a memoryless sytem that can be described by the equation

$$y = x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x^1 + 1$$

(5.11)
Figure 5.7: The results for $H_0 \Gamma$, $H_1 \Gamma$, and $H_2$ of a low-pass RC filter. Pole is at 3MHz. The last two columns are the real and imaginary part of the extracted transfer function. The beginning columns are the arguments in hertz to the transfer function.
### 3.2

**Format:** ascii

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-3e+06</td>
<td>+0</td>
<td>+3e+06</td>
<td>-0.000170968</td>
<td>+0</td>
</tr>
<tr>
<td>-3e+06</td>
<td>+1e+06</td>
<td>+3e+06</td>
<td>+2.95559e-06</td>
<td>+5.03615e-06</td>
</tr>
<tr>
<td>-3e+06</td>
<td>+2e+06</td>
<td>+2e+06</td>
<td>+1.60853e-05</td>
<td>-2.08472e-05</td>
</tr>
<tr>
<td>-3e+06</td>
<td>+2e+06</td>
<td>+3e+06</td>
<td>+2.08938e-06</td>
<td>+3.12744e-06</td>
</tr>
<tr>
<td>-3e+06</td>
<td>+3e+06</td>
<td>+3e+06</td>
<td>+1.19571e-06</td>
<td>+1.75821e-07</td>
</tr>
<tr>
<td>-2e+06</td>
<td>-1e+06</td>
<td>+3e+06</td>
<td>-0</td>
<td>+1.3751e-05</td>
</tr>
<tr>
<td>-2e+06</td>
<td>+0</td>
<td>+2e+06</td>
<td>-0.000168853</td>
<td>+0</td>
</tr>
<tr>
<td>-2e+06</td>
<td>+0</td>
<td>+3e+06</td>
<td>-1.95233e-05</td>
<td>+9.20555e-06</td>
</tr>
<tr>
<td>-2e+06</td>
<td>+1e+06</td>
<td>+2e+06</td>
<td>-0.00010614</td>
<td>-3.1055e-05</td>
</tr>
<tr>
<td>-2e+06</td>
<td>+1e+06</td>
<td>+3e+06</td>
<td>+0.000408738</td>
<td>+0.0002841</td>
</tr>
<tr>
<td>-2e+06</td>
<td>+2e+06</td>
<td>+2e+06</td>
<td>+1.31617e-06</td>
<td>-3.43648e-07</td>
</tr>
<tr>
<td>-2e+06</td>
<td>+2e+06</td>
<td>+3e+06</td>
<td>+9.4762e-06</td>
<td>+4.53258e-07</td>
</tr>
<tr>
<td>-2e+06</td>
<td>+3e+06</td>
<td>+3e+06</td>
<td>+2.57414e-05</td>
<td>+2.58141e-05</td>
</tr>
<tr>
<td>-1e+06</td>
<td>-1e+06</td>
<td>+2e+06</td>
<td>-0</td>
<td>+7.18017e-06</td>
</tr>
<tr>
<td>-1e+06</td>
<td>-1e+06</td>
<td>+3e+06</td>
<td>-5.91117e-06</td>
<td>-1.00723e-05</td>
</tr>
<tr>
<td>-1e+06</td>
<td>+0</td>
<td>+1e+06</td>
<td>-0.000169207</td>
<td>+0</td>
</tr>
<tr>
<td>-1e+06</td>
<td>+0</td>
<td>+2e+06</td>
<td>-0.00019892</td>
<td>+0.000616142</td>
</tr>
<tr>
<td>-1e+06</td>
<td>+0</td>
<td>+3e+06</td>
<td>-3.37367e-05</td>
<td>+2.5736e-05</td>
</tr>
<tr>
<td>-1e+06</td>
<td>+1e+06</td>
<td>+1e+06</td>
<td>+4.16386e-07</td>
<td>-3.29523e-07</td>
</tr>
<tr>
<td>-1e+06</td>
<td>+1e+06</td>
<td>+2e+06</td>
<td>-7.65668e-05</td>
<td>-5.72107e-05</td>
</tr>
<tr>
<td>-1e+06</td>
<td>+1e+06</td>
<td>+3e+06</td>
<td>+1.01919e-05</td>
<td>+1.59415e-06</td>
</tr>
<tr>
<td>-1e+06</td>
<td>+2e+06</td>
<td>+2e+06</td>
<td>+7.87875e-06</td>
<td>+6.11655e-06</td>
</tr>
<tr>
<td>-1e+06</td>
<td>+2e+06</td>
<td>+3e+06</td>
<td>+1.23486e-05</td>
<td>+4.00424e-06</td>
</tr>
<tr>
<td>-1e+06</td>
<td>+3e+06</td>
<td>+3e+06</td>
<td>+5.42847e-06</td>
<td>+9.70645e-06</td>
</tr>
<tr>
<td>+0</td>
<td>+0</td>
<td>+0</td>
<td>+1.23999e-05</td>
<td>+0</td>
</tr>
<tr>
<td>+0</td>
<td>+0</td>
<td>+1e+06</td>
<td>-0.000106029</td>
<td>-3.70775e-05</td>
</tr>
<tr>
<td>+0</td>
<td>+0</td>
<td>+2e+06</td>
<td>-8.27047e-05</td>
<td>-5.50806e-05</td>
</tr>
<tr>
<td>+0</td>
<td>+0</td>
<td>+3e+06</td>
<td>-5.96514e-05</td>
<td>-5.90868e-05</td>
</tr>
<tr>
<td>+0</td>
<td>+1e+06</td>
<td>+1e+06</td>
<td>+9.42825e-08</td>
<td>+2.04803e-06</td>
</tr>
<tr>
<td>+0</td>
<td>+1e+06</td>
<td>+2e+06</td>
<td>-7.71757e-06</td>
<td>-1.48022e-05</td>
</tr>
<tr>
<td>+0</td>
<td>+1e+06</td>
<td>+3e+06</td>
<td>-1.73463e-05</td>
<td>-3.431e-05</td>
</tr>
<tr>
<td>+0</td>
<td>+2e+06</td>
<td>+2e+06</td>
<td>-2.02255e-06</td>
<td>+1.45229e-06</td>
</tr>
<tr>
<td>+0</td>
<td>+2e+06</td>
<td>+3e+06</td>
<td>-2.10179e-05</td>
<td>-1.00984e-06</td>
</tr>
<tr>
<td>+0</td>
<td>+3e+06</td>
<td>+3e+06</td>
<td>-4.61831e-06</td>
<td>+9.56671e-07</td>
</tr>
<tr>
<td>+1e+06</td>
<td>+1e+06</td>
<td>+1e+06</td>
<td>+2.79563e-06</td>
<td>+1.35041e-06</td>
</tr>
</tbody>
</table>

Figure 5.8: Sample of results for $H_3$ of low-pass RC filter. Pole is at 3MHz. The first three columns represent the argument to $H_3$. The last two columns are the real and imaginary part of the value of the function.
<table>
<thead>
<tr>
<th>Transfer Function indicating order</th>
<th>Number of Calculated points</th>
</tr>
</thead>
<tbody>
<tr>
<td>( H_0 )</td>
<td>1</td>
</tr>
<tr>
<td>( H_1 )</td>
<td>4</td>
</tr>
<tr>
<td>( H_2 )</td>
<td>16</td>
</tr>
<tr>
<td>( H_3 )</td>
<td>32</td>
</tr>
<tr>
<td>( H_4 )</td>
<td>60</td>
</tr>
<tr>
<td>( H_5 )</td>
<td>87</td>
</tr>
<tr>
<td>( H_6 )</td>
<td>131</td>
</tr>
<tr>
<td>( H_7 )</td>
<td>179</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>510</strong></td>
</tr>
</tbody>
</table>

Table 5.4: Number of points calculated for each \( H \) for 7th order system

From (3.69) we know that the nonlinear transfer function for a memoryless system described by a polynomial is just the constant coefficient of that order. Thus for this system we know that

\[
\begin{align*}
H_0 &= 1 \\
H_1(f_a) &= 1 \\
H_2(f_a, f_b) &= 1 \\
H_3(f_a, f_b, f_c) &= 1 \\
H_4(f_a, f_b, f_c, f_e) &= 1 \\
H_5(f_a, f_b, f_c, f_e, f_f) &= 1 \\
H_6(f_a, f_b, f_c, f_e, f_f, f_g) &= 1 \\
H_7(f_a, f_b, f_c, f_e, f_f, f_g, f_h) &= 1
\end{align*}
\]

(5.12) \( \text{to} \) (5.19)

This polynomial characteristic was simulated using ASTAP with a voltage controlled voltage source with the specified polynomial characteristics. The number of transfer function points extracted for each order is indicated in table 5.4. Figure 5.9 is a scattergram of the extracted values of the transfer functions as functions of transfer function order.

5.5 Four Megabit Token Ring Equalizer Circuit

In order to test the extraction procedure an existing circuit was chosen that exhibits linear behavior for small inputs and nonlinear behavior for large inputs. This circuit will be used to test the theory and practicality of the development presented in this thesis. The circuit shown in figure 5.10 is an active equalizer circuit for the IBM 4M bit token ring local area network. Terminals IN_HI and IN_LO are the high and low inputs and the output is the voltage across OUT_HI and OUT_LO. For long cable lengths and thus low power input signals both input transistors remain in the active region and the small signal or linear model of the circuit accurately predicts
Figure 5.9: Extracted value of the nonlinear transfer function for a memoryless 7th order system with known characteristics. All values should be equal to (110). Different symbols represent the order of the nonlinear transfer function.
the performance of the circuit. But for short cable lengths and thus large input signals, the input signal is strong enough to alternately turn off both transistors and the output signal is clipped. This behavior is shown in figures 5.11 and 5.12 for various input signal levels and at two different frequencies. Note also the frequency-dependent nonlinear effects of the zero crossing shift. The phase of the input signal is the same for all signals.

Since we will use the extraction algorithm described in section 4.2 to develop the transfer functions used in simulating a steady-state system, we must know the frequencies which will be used. Furthermore, we must be able to make the tradeoff between accuracy and speed of both the simulation and the extraction procedure. Let us examine a lower bound for the number of simulations needed for our extraction routine. If \( F \) is the number of input frequencies and \( N \) is the assumed order of the system, then the number of simulations needed \( S \) is

\[
S \geq 1 + N \sum_{k=1}^{\min(N,F)} \binom{F}{k}
\]  

(5.20)

Here the 1 represents the single simulation that is needed to determine \( H_0 \). The value \( \sum_{k=1}^{\min(N,K)} \binom{F}{k} \) is the sum of all of the possible combinations of the input frequencies starting with a single input frequency up to the maximum number of input frequencies. This maximum is the lesser of either the assumed order of the system \( N \) or the number of input frequencies. Since each run represents at least \( N \) simulations this sum is multiplied by \( N \). Table 5.5 shows the relationship between this lower bound for the number of required simulations as a function of the number of frequencies \( N \) and the assumed order of the system. This table represents a lower bound in the sense that it will be exactly the minimum number of simulations needed if and only if there are no unique FIPD's with the same order and frequency as discussed in section 4.2.2. Additional runs must be added either because of (a) unbalanced DC FIPD's, (b) FIPD's with unbalanced DC FIPD subsets or (c) other unique FIPD's with the same order and frequencies. The actual number of runs used can be greater than the minimum required as discussed here. This is denoted by the value of \( \text{extra} \) runs as described in figure 4.3 being greater than zero. The term \( \text{extra} \) runs is somewhat misleading because it really refers to the number of extra simulations that are done for each run. For a \( n \)th order system, each run must have at least \( n \) different simulations at different input amplitudes. The equivalent of a polynomial fit is performed on the varying output amplitudes in order to decompose the output into the 1st, 2nd, etc. order response. Increasing \( \text{extra} \) runs only allows for extra points on the polynomial fit for each output phasor. Additional runs added because (a) \( \Gamma \), (b) and (c) listed above are required to separate out different components of an output frequency that are caused by the same set of input frequencies.

Table 5.6 shows the same relationship of table 5.5 for the special case of all input frequencies being the lowest possible harmonics of a fundamental. So, for instance for 3 frequencies the frequencies used would be the first harmonic (the fundamental), the second harmonic and the third harmonic. The value of the fundamental does
Figure 5.10: Schematic of equalizer circuit
Figure 5.11: Output of the equalizer circuit shown in figure 5.10 with a single sinusoidal 4 MHz input signal of 100mV, 200mV, 300mV, 400mV, and 500mV. Note the nonlinear affect of the clipping and the zero crossing. The phase of the input is the same for all signals.
Figure 5.12: Output of the equalizer circuit shown in figure 5.10 with a single sinusoidal 8 MHz input signal of 100mV, 200mV, 300mV, 400 mV and 500mV. Note the nonlinear affect of the clipping and the zero crossing. The phase of the input is the same for all signals.
Table 5.5: The lower bound of the minimum number of simulations needed to extract the nonlinear transfer functions as a function of the number of frequencies and the order of the system. No assumption is made on the relationship between the input frequencies. This lower bound will be exactly the same as the minimum if there are no additional runs required because of multiple unique FIPD's with the same order and frequencies.

<table>
<thead>
<tr>
<th>Number of Frequencies</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>7</td>
<td>10</td>
<td>13</td>
<td>16</td>
<td>19</td>
<td>22</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>13</td>
<td>22</td>
<td>29</td>
<td>36</td>
<td>43</td>
<td>50</td>
<td>57</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>21</td>
<td>43</td>
<td>61</td>
<td>76</td>
<td>91</td>
<td>106</td>
<td>121</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>31</td>
<td>76</td>
<td>121</td>
<td>156</td>
<td>187</td>
<td>218</td>
<td>249</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>43</td>
<td>124</td>
<td>225</td>
<td>311</td>
<td>379</td>
<td>442</td>
<td>505</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>57</td>
<td>190</td>
<td>393</td>
<td>596</td>
<td>757</td>
<td>890</td>
<td>1017</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>73</td>
<td>277</td>
<td>649</td>
<td>1091</td>
<td>1477</td>
<td>1779</td>
<td>2041</td>
</tr>
</tbody>
</table>

not affect this table as long as DC (zero hertz) is not considered an input frequency. Note the striking difference between table 5.6 and table 5.5. For example: the lower bound for the number of simulations needed for a third order system with four input frequencies is 43. By looking at the similar input shown in table 4.3 one can derive this number of simulations. Run 0 in table 4.3 represents a single simulation and runs 1 through 14 represent 3 (because the system is 3rd order) runs each. Thus a total of $1 + 3 \times 14 = 43$ simulations are needed. Note that tables 4.3 through 4.7 correspond to a system of 4 input frequencies that include DC (zero hertz). This is slightly different from table 5.5 which represents input frequencies that are harmonics and do not include DC. When DC is an input frequency fewer additional runs need to be added because DC mixed with any other frequency can produce only that other frequency whereas a non-DC frequency say $f_1$ mixed with another frequency say $f_2$ produces two frequencies $f_1 + f_2$ and $f_1 - f_2$. Thus the number of simulations represented in tables 4.5 through 4.7 $(1 + 3 \times (14 + 4) = 55)$ is slightly different from the number of simulations shown for third order and 4 input frequencies in table 5.5 (61). The additional simulations over the lower bound given in table 5.6 are due to: (a) unbalanced DC FIPD’s, (b) FIPD's with unbalanced DC FIPD subsets or (c) other unique FIPD’s with the same order and frequencies. As the assumed order of the system increases and as the number of input frequencies increases the number of additional runs needed increases dramatically.

Since the number of additional runs is related to the number of unique FIPD’s that add to the same number it can be conjectured that table 5.5 represents the worst case condition for the upper bound for the minimum number of simulations needed. Of course the number of actual simulations can be made arbitrarily high by choosing a large value for xtra runs as described in figure 4.3.

The circuit shown in figure 5.10 was extracted assuming a 5th order system and
a maximum input amplitude of 0.5 Volts. The number of extra simulations per run \( \text{xtra\_runs} \) as described in figure 4.3 was set to 1. The input frequencies to be extracted were 2MHz, 4MHz, 6MHz, 8MHz, and 10MHz. A total of 509 simulation runs each consisting of 6 simulations resulted in a total of 3054 individual simulations. The total number of points extracted for each transfer function is shown in table 5.7. Figures 5.13 and 5.14 show the response of the equalizer circuit at 4MHz and 8MHz respectively when the input amplitude is 0.5 volts. In addition to the simulated response, the response as predicted by the extracted Volterra model is shown. The Volterra model is truncated to various orders to show the significance of the various order contributions. For the 4MHz sine waves in figure 5.13 the full fifth

### Table 5.6: The minimum number of simulations needed to extract the nonlinear transfer functions as a function of the number of frequencies and the order of the system. The frequencies consist of a fundamental and the lowest harmonics. A value of "-" indicates that the run time to calculate the number of simulations was over 24 hours on an IBM RS/6000 model 530.

<table>
<thead>
<tr>
<th>Number of Frequencies</th>
<th>Assumed Order of the System</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>3</td>
<td>7</td>
<td>13</td>
<td>81</td>
<td>101</td>
<td>175</td>
<td>302</td>
<td>665</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>4</td>
<td>13</td>
<td>34</td>
<td>165</td>
<td>355</td>
<td>625</td>
<td>1261</td>
<td>3169</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>5</td>
<td>21</td>
<td>61</td>
<td>393</td>
<td>891</td>
<td>2131</td>
<td>4383</td>
<td>10209</td>
</tr>
</tbody>
</table>
| 5                     |                             | 6 | 31|100|825|2546|5587|12776|-
| 6                     |                             | 7 | 43|163|1445|5436|13273|30864|-
| 7                     |                             | 8 | 57|238|2345|11241|30787|-
| 8                     |                             | 9 | 73|337|3529|20556|65131|-

### Table 5.7: Number of points calculated for each nonlinear transfer function. The assumed order of the system equals 5 and the number of frequencies equals 5.

<table>
<thead>
<tr>
<th>Transfer Function</th>
<th>Number of Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>( H_0 )</td>
<td>1</td>
</tr>
<tr>
<td>( H_1 )</td>
<td>5</td>
</tr>
<tr>
<td>( H_2 )</td>
<td>30</td>
</tr>
<tr>
<td>( H_3 )</td>
<td>110</td>
</tr>
<tr>
<td>( H_4 )</td>
<td>365</td>
</tr>
<tr>
<td>( H_5 )</td>
<td>1001</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>Input</td>
</tr>
<tr>
<td>----------------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td>Real</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>-0.225079</td>
</tr>
<tr>
<td>4</td>
<td>0.15915494</td>
</tr>
<tr>
<td>6</td>
<td>-0.07502636</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0.04501582</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>42</td>
<td>0</td>
</tr>
<tr>
<td>44</td>
<td>0</td>
</tr>
<tr>
<td>46</td>
<td>0</td>
</tr>
<tr>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.8: Input and output phasors as an example of the response predicted by the extracted Volterra model. The input is derived from a pulse train with truncated harmonics.
order Volterra model prediction is indistinguishable from the simulated response. The results in figure 5.14 are similar but because of the more pronounced clipping, a slight difference can be distinguished between the fifth order Volterra model response and the simulated response. Figures 5.15 and 5.16 show the same curves as figures 5.13 and 5.14 but with an input amplitude of 0.1 volts. With this lower input amplitude, the circuit exhibits a linear response and thus the various truncations of the Volterra model are indistinguishable i.e., only the first order response ($H_1$) is significant. Figure 5.17 shows the various truncated Volterra models for an arbitrary input with non-zero amplitudes for all five valid input frequencies (2MHz, 4MHz, 6MHz, 8MHz, and 20MHz). The output phasors for the full fifth order Volterra model and the input phasors are given in table 5.9. Note that the output components are negligible at high frequencies. As in figures 5.13 and 5.14, the higher order Volterra model yields better results and the full fifth order model is almost indistinguishable from the simulated result. Figure 5.18 shows the effect of increasing the input amplitude 10 percent over the maximum valid range for the model. Although the fifth order model closely matches the simulated result for most of the waveform, the area where the circuit clips the output (and is thus highly nonlinear) the Volterra model differs from the simulated result.
Figure 5.13: Output of the equalizer circuit shown in figure 5.10 with a single sinusoid 4 MHz input signal of 500mV. The Volterra model is truncated at various orders. The even order contributions are negligible, thus model order = 1 is indistinguishable from order = 2, and model order = 3 is indistinguishable from order = 4.
Figure 5.14: Output of the equalizer circuit shown in figure 5.10 with a single sinusoidal 8 MHz input signal of 500mV. The Volterra model is truncated at various orders. The even-order contributions are negligible; thus model order = 1 is indistinguishable from order = 2, and model order = 3 is indistinguishable from order = 4.
Figure 5.15: Output of the equalizer circuit shown in figure 5.10 with a single sinusoid 4 MHz input signal of 100mV. The Volterra model is truncated at various orders. Since the input signal is relatively low, only the first order contribution is significant and thus the higher order models are indistinguishable from order=1.
Figure 5.16: Output of the equalizer circuit shown in figure 5.10 with a single sinusoid 8 MHz input signal of 100mV. The Volterra model is truncated at various orders. Since the input signal is relatively low, only the first order contribution is significant and thus the higher order models are indistinguishable from order=1.
<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Real</td>
<td>Imag.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.076699</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.092379</td>
<td>0.056737</td>
</tr>
<tr>
<td>6</td>
<td>0.060898</td>
<td>0.13464</td>
</tr>
<tr>
<td>8</td>
<td>0.037567</td>
<td>0.056706</td>
</tr>
<tr>
<td>10</td>
<td>0.073445</td>
<td>0.094801</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>38</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>42</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>44</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>46</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>48</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.9: Input and output phasors as an example of the response predicted by the extracted Volterra model. The corresponding waveforms are shown in figure 5.17.
5.5.1 Problems

The DC or zero Hz output component is in general the most difficult to predict because in there are more distinct contributions to DC than to any other frequency. One problem with the extraction algorithm presented here is that previously calculated values of the Volterra nonlinear transfer function are used to subtract “known” contributions of the outputs of later simulations. Since there are so many different FIPD’s and thus points on the Volterra nonlinear transfer functions that contribute to DC, the “known” contributions that are subtracted out are more sensitive to error. This error in the “known” contributions leads to error in the calculated values of the Volterra nonlinear transfer function and these errors lead to larger errors later on in the extraction process. This effect can be seen in figure 5.19. The input in this case is a truncated Fourier series of a pulse train. The DC error of the fifth order Volterra model is about 0.15 Volts. If the DC component is removed from the output of the Volterra model and the simulated output the higher order Volterra models give very good agreement as shown in figure 5.20. The output phasors for the fifth order Volterra model and the input phasors as shown in figure 5.19 are given in table 5.8.

5.5.2 Discussion

The computation needed in the extraction process is dominated by the simulations. The resource needed for simulations is determined by the number of simulations needed and the individual simulation times. The number of simulations needed is a function of the number of input frequencies and the assumed order of the system. On the other hand, the individual simulation times depend on the inputs to the simulator, the simulator itself, and the complexity of the circuit. In the equalizer example, the input frequencies were required to be commensurable because the simulator employed shooting methods and thus required a periodic input. Large differences in the values of the input frequencies would result in small time steps relative to the input period and thus long run times and large output files. For our case, only simple harmonics were used as input frequencies and thus relative to the highest input frequency the simulation period is kept to a minimum. If a frequency domain simulator were used in the extraction process these would not be the dominating factors.

The extracted fifth order Volterra nonlinear transfer function of the equalizer circuit agrees well with simulated data with the exception of the DC component for some sets of input frequencies. This disagreement is due to the error accumulated during the extraction process.
Figure 5.17: Sample output predicted by extracted Volterra model for arbitrary input with maximum allowable input amplitude. The peak value of the input is 0.5 volts. Higher order Volterra models more closely match the output as predicted by simulation. The full 5th order Volterra model output is almost indistinguishable from the simulated output.
Figure 5.18: Sample output predicted by extracted Volterra model for arbitrary input with input amplitude 10 percent greater than maximum valid level. The peak value of the input is 0.55 volts. Higher order Volterra models more closely match the output as predicted by simulation.
Figure 5.19: The characteristics of the equalizer with a pulse with truncated harmonics. The output predicted by the Volterra model closely matches the output predicted by time domain simulation if the Volterra model curve is shifted up by 0.15 volts. The phasors for the Volterra output are given in table 5.8.
Figure 5.20: The characteristics of the equalizer with a pulse with truncated harmonics. This is figure 5.19 with the DC component removed for all signals. The higher order Volterra model more closely matches the simulated results.
Chapter 6

Conclusion

6.1 Summary

In this work, a new technique for the characterization of nonlinear analog circuits was developed. As a preliminary, the analysis of univariate and bivariate generalized power series and their relationship to univariate and bivariate Volterra series was examined both mathematically and experimentally.

The relationship between Volterra series analysis and generalized power series analysis was established for both the single input (univariate) and two input (bivariate) system representations. A general power series is equivalent to a Volterra system if the form of the Volterra nonlinear transfer function is restricted to a linear system followed by an ideal integer exponentiation $(\cdot)^n$ followed by a second linear system. A similar restriction applies to the bivariate case.

The experimentally derived bivariate power series characteristics of the drain current of a MESFET amplifier were used to calculate the bivariate Volterra nonlinear transfer function and thus the output of a MESFET circuit can be represented by a nonlinear signal flow graph. If feedback is present in the flow graph, iterative techniques can be used to solve for the steady-state response entirely in the frequency domain. The predicted output calculated in this way matches experimental data closely.

A new general method to measure nonlinear transfer functions was presented. The method is applicable to extraction from circuit level simulation and the results could be used in a table-based frequency-domain simulator. As an example of this extraction technique, several test circuits were examined including a simple linear circuit, a memoryless seventh order system, and a practical circuit whose transfer functions were unknown. Predicted values based on the extracted table show good agreement with the behavior predicted by circuit level simulations with the exception of the DC output value for some input conditions.

This new method of extracting Volterra nonlinear transfer functions lends to applications for large analog system simulation. The system is broken up into small subsystems, especially subsystems that are linear with some distortion. The extraction process is then applied to each subsystem. A harmonic balance type simulator could then be used to efficiently predict the behavior of the entire system. If a subsystem was used in different systems or more than once in the same system, then the extraction procedure would only be performed once on the subsystem.

Furthermore, with a harmonic balance simulator, these black box characterizations of the subsystems could be mixed with device level elements. Thus a simulation would not necessarily be at the system level or the device level but rather something in between. This ability greatly increases the flexibility of the simulator and allows the designer to seamlessly choose the level and detail of the simulation.
6.2 Suggestions for Further Work

Several aspects of this study can possibly be extended. Firstly, several mathematical treatments need to be further studied. The indexing scheme given in appendix A for determining the IPD’s of a bivariate simulation is much faster than a brute force approach but could possibly still be improved with further work. Secondly, a rigorous mathematical treatment of the best and necessary choices of the phase and amplitude for the test signals used in extraction could provide more insight and better extraction techniques.

Furthermore, practical applications using the extraction work could be investigated. The black-box extracted model can be integrated into a harmonic balance simulator as proposed in section 3.4. Ways to reduce the amount of data stored need to be investigated. Possibilities include:

1. Extract complex polynomial coefficient form for Volterra nonlinear transfer functions instead of table of values.

2. Investigate restricted forms of the Volterra nonlinear transfer functions and develop optimum ways to extract the necessary parameters.

The extraction examples given were based on a time-domain shooting method simulator. Integrating a harmonic balance simulator into the extraction system could improve the accuracy of the model and speed up the extraction process. This would also allow for incommensurable input frequencies. A harmonic balance simulator based approach could also be extended to perform the extraction process on a system consisting of subsystems that were previously characterized by the extraction process. Thus a complicated system described only by black box elements characterized by several sets of Volterra nonlinear transfer functions could be reduced to a single black box characterized by a single set of Volterra nonlinear transfer functions.

Solutions to the DC error problems could be investigated. One possible approach would be to form the problem as a large error minimization system. The existing extraction process would first be run to extract best guesses for all of the input values. Then these values (all calculated points on the Volterra nonlinear transfer functions) would be used as starting points in a many-input simulated annealing process. The existing simulation data would be used as the target behavior and thus be used to calculate the error function. Thus no new simulations would be needed. Different combinations of input sets, error functions, and annealing techniques could be investigated.

Lastly, an investigation of application of the algorithm used to choose the input conditions could be applied to analog test vector generation. As analog systems become more complex, the cost of testing in the manufacturing environment becomes more and more important.
References


**Other Useful References Not Cited in the Text**


128


Appendix A
Algorithm for Bivariate GPSA Indexing

In order to calculate \( T \) in (3.22) the set of conditions on the second summation give restrictions that are less than obvious to implement:

\[
\sigma = \sum_{i=0}^{N} p_i + r_i
\]  
\hspace{1cm} (A.1)

\[
\rho = \sum_{i=0}^{N} q_i + s_i
\]  
\hspace{1cm} (A.2)

\[
\text{for } 1 \leq i \leq N : p_i + q_i - r_i - s_i = |n_i|
\]  
\hspace{1cm} (A.3)

\[
\sigma + \rho = n + 2\alpha
\]  
\hspace{1cm} (A.4)

At this point we know the value of \( n\Gamma \alpha \Gamma \) and the IPD vector \( \overline{n} \) which consists of integers such that \( \sum_{i=1}^{N} |n_i| = n \). We also know that \( \sigma \Gamma \rho \Gamma p_i \Gamma q_i \Gamma r_i \Gamma s_i \Gamma \) are all non-negative integers. From (A.4) it follows that \( 0 \leq \sigma \leq (n + 2\alpha) \) and \( 0 \leq \rho \leq (n + 2\alpha) \) since both \( \sigma \) and \( \rho \) are nonnegative. We can therefore step through all possible \( \sigma \) and \( \rho \) by the algorithm given in figure A.1. Thus the only remaining task is to identify all sets of vectors for a given \( \overline{n} \Gamma \sigma \Gamma \) and \( \rho \) such that that all equations are satisfied.

In order to facilitate further explanation a special ripple algorithm is defined. The purpose of the algorithm is to sequence through all possible sets of vectors of length \( N \) such that the sum of the components is a given constant \( S \), i.e. \( \sum_{i=1}^{N} v_i = S \). We further restrict all of the components of the vectors to be non-negative integers, i.e. \( v_i \geq 0 \). Since all the components of \( \overline{v} \) are non-negative the sum is equivalent to the \( L_1 \) norm \( ||\overline{v}||_1 = \sum_i |v_i| \). It follows that \( \max_i v_i \leq S \). Our algorithm first initializes \( v_1 = S \) and all other \( v_i = 0 \). Since \( \sum_{i=1}^{N} v_i = S \) then this initial vector is the only vector with \( v_1 = S \). We then find all the possible vectors with \( v_1 = S - 1 \) then all possible vectors with \( v_1 = S - 2 \) and so on. The key to this algorithm is in recognizing that the problem of finding all valid vectors with \( v_1 = S - j \) is equivalent to the problem of finding all possible vectors of length \( N - 1 \) with \( ||\overline{v}||_1 = j \). Thus our algorithm is recursive. Let us define the number of vectors to be \( F_N(S) \). From

- initialize \( \sigma = 0 \) and \( \rho = n + 2\alpha \).
- Increment \( \sigma \) and decrement \( \rho \) until \( \sigma = n + 2\alpha \) and \( \rho = 0 \).

Figure A.1: Algorithm for stepping through \( \rho \) and \( \sigma \)

133
Let $v_1 = S, S - 1, S - 2 \ldots 0$.

- for each value of $v_1$, define a new vector $v''$ of length $N - 1$ defined by $v'' = v''_1 = v_{i+1}$ and a new constant value $S'$ defined by $S' = S - v''_1$.
  - let $v'_1 = S', S' - 1, S' - 2 \ldots 0$.
  - for each value of $v'_1$, define a new vector $v'''$ of length $N - 2$ defined by $v''' = v'''_1 = v'_{i+1}$ and a new constant value $S''$ defined by $S'' = S' - v''_1$.
  - Continue the process until the vector defined is of length $1$ and thus the vector has only one possible value for a given $L_1$ norm.

Figure A.2: Recursive ripple algorithm

Our recursive algorithm we know that

$$F_N(S) = \sum_{K=0}^{S} F_{(N-1)}(K) \quad (A.5)$$

and

$$F_1(S) = 1 \quad (A.6)$$

Thus

$$F_N(S) = \sum_{K_N=0}^{S} K_N \sum_{K_{(N-1)}=0}^{K_N} K_{(N-1)} \sum_{K_{(N-2)}=0}^{K_{(N-1)}} K_{(N-2)} \ldots \sum_{K_3=0}^{K_4} K_3 \sum_{K_2=0}^{K_3} 1 \quad (A.7)$$

It follows that the number of vector combinations $F_N(S)$ is of order $\frac{S^{N-1}}{(N-1)!}$. In order to better illustrate this sequence, let us look at the example of $N = 4$ and $S = 3$. Here our algorithm produces the vectors shown in figure A.3. In order to more easily implement this ripple algorithm we will use the non-recursive procedure shown in figure A.4 that takes a vector of length $N$ and finds the next vector in the sequence. If the input vector is the last in the sequence then the overflow flag is set and the first vector of the sequence is returned. The following algorithm will take any of the above vectors and produce the next vector.

The problem now is to find all sets of vectors $\vec{v}, \vec{q}, \vec{r}$ for a given $\Gamma \sigma \Gamma$ and $\vec{n}$. We now define a new vector $\vec{u}$ of length $2N$ that consists of the concatenation of $\vec{p}$ and $\vec{q}$ i.e. $u_i = p_i$ for $i \leq N$ and $u_i = r_{i-N}$ for $i > N$. From (A.1) and (A.2) we know that the $L_1$ norm of $u$ is equal to $\rho \Gamma$ thus we can step through all possible values of $\vec{p}$ and $\vec{q}$ using our ripple algorithm on $\vec{u}$ with $\|\vec{u}\|_1 = \rho$.

Our only remaining task is to find all valid values of $\vec{q}$ and $\vec{s}$ for a given $\vec{n} \Gamma \rho \Gamma$ and $\vec{p}$. By solving (A.3) for $q_i$ and substituting into (A.2) we get:

$$\rho = \sum_{i=1}^{N} |n_i| + r_i + 2s_i - p_i \quad (A.8)$$
Figure A.3: Example of ripple sequence for $S = 3$ and $N = 4$

$$(3, 0, 0, 0)$$
$$(2, 1, 0, 0)$$
$$(2, 0, 1, 0)$$
$$(2, 0, 0, 1)$$
$$(1, 2, 0, 0)$$
$$(1, 1, 1, 0)$$
$$(1, 1, 0, 1)$$
$$(1, 0, 2, 0)$$
$$(1, 0, 1, 1)$$
$$(1, 0, 0, 2)$$
$$(0, 3, 0, 0)$$
$$(0, 2, 1, 0)$$
$$(0, 2, 0, 1)$$
$$(0, 1, 2, 0)$$
$$(0, 1, 1, 1)$$
$$(0, 1, 0, 2)$$
$$(0, 0, 3, 0)$$
$$(0, 0, 2, 1)$$
$$(0, 0, 1, 2)$$
$$(0, 0, 0, 3)$$
• Calculate $L_1$ norm $= S$.
• If $S = 0$ then
  - set the overflow flag
  - end
• Find $i_{\text{max}} = \max_{vi \neq 0} i$.
• If $i_{\text{max}} = N$ then
  - Find $i_{\text{next}} = \max_{vi \neq 0} i < N$
  - If $i_{\text{next}}$ does not exist
    * $v_1 = v_N$
    * If $N \neq 1$ then $v_N = 0$
    * set overflow flag
    * end
  - If $i_{\text{next}} = N - 1$ then
    * $v_{N-1} = v_{N-1} - 1$
    * $v_N = v_N + 1$
    * end
  - If $i_{\text{next}} \neq N - 1$ then
    * $v_{i_{\text{next}}} = v_{i_{\text{next}}} - 1$
    * $v_{i_{\text{next}}+1} = 1 + v_N$
    * $v_N = 0$
    * end
• If $i_{\text{max}} \neq N$ then
  - $v_{i_{\text{max}}} = v_{i_{\text{max}}} + 1$
  - $v_{i_{\text{max}}+1} = 1$
  - end

Figure A.4: Nonrecursive ripple algorithm
Rearranging for $||\bar{z}||_1$ yields

$$
\sum_{i=1}^{N} s_i = \frac{\rho + \sum_{i=1}^{N} p_i - r_i - |n_i|}{2}
$$

(A.9)

Thus using (A.9) we can determine $||\bar{z}||_1$ and ripple through all possible combinations of $\bar{z}$. If the right side of (A.9) yields a negative or a noninteger value then we know that no solutions exist for the current set of $\bar{n}\Gamma\rho\Gamma\bar{\sigma}\Gamma\bar{p}\Gamma\bar{r}$ and $\bar{\tau}$.

Rearranging (A.3) for $q_i$ yields

$$
q_i = |n_i| + r_i + s_i - p_i
$$

(A.10)

Thus for a given set of $\bar{n}\Gamma\rho\Gamma\sigma\Gamma\bar{p}\Gamma\bar{r}\Gamma\bar{q}$ we can calculate $\bar{q}$. If any $q_i$ is negative then no solution exists for the current set of $\bar{n}\Gamma\rho\Gamma\sigma\Gamma\bar{p}\Gamma\bar{r}\Gamma\bar{q}$ and $\bar{\tau}$.

The alternate way of finding all of the possible values $\bar{p}\Gamma\bar{r}\Gamma\bar{q}\Gamma\bar{z}$ that fit our criterion is to go through all possible combinations with components less than or equal to $n + 2\alpha$. The number of possibilities is $(n + 2\alpha + 1)^N$. 

137
Appendix B
Properties of Volterra Nonlinear Transfer Functions

B.1 Introduction

In order to gain a better understanding of Volterra nonlinear transfer functions, we will look at the basic formula and the implications of its form. Specifically, we will look at the basic formula, the idea of kernel symmetry, and the relationship of transfer functions whose arguments are related.

B.2 Basic Formula

The output $y(t)$ of a system described by Volterra kernels $h_n(\tau_1, \tau_2 \ldots \tau_n)$ stimulated by an input $x(t)$ is given by

$$y(t) = \sum_{n=0}^{\infty} y_n(t)$$  \hspace{1cm} (B.11)

where

$$y_n(t) = \int_{\tau_n=-\infty}^{+\infty} \int_{\tau_{n-1}=-\infty}^{+\infty} \ldots \int_{\tau_1=-\infty}^{+\infty} h_n(\tau_1, \tau_2 \ldots \tau_n) \prod_{i=1}^{n} x(t - \tau_i)$$

$$d\tau_1 d\tau_2 \ldots d\tau_n$$  \hspace{1cm} (B.12)

$$y_0(t) = h_0$$  \hspace{1cm} (B.13)

The $n$th order nonlinear transfer function $H_n(f_1, f_2 \ldots f_n)$ is defined to be the multi-dimensional Fourier transform of the $n$th order kernel $h_n(\tau_1, \tau_2 \ldots \tau_n)$.

$$H_n(f_1, f_2 \ldots f_n) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \ldots \int_{-\infty}^{+\infty} h_n(\tau_1, \tau_2 \ldots \tau_n)$$

$$e^{-j2\pi(f_1\tau_1 + f_2\tau_2 + \ldots + f_n\tau_n)} d\tau_1 d\tau_2 \ldots d\tau_n$$  \hspace{1cm} (B.14)

$$H_0 = h_0$$  \hspace{1cm} (B.15)

and

$$h_n(\tau_1, \tau_2 \ldots \tau_n) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \ldots \int_{-\infty}^{+\infty} H_n(f_1, f_2 \ldots f_n)$$

$$e^{+j2\pi(f_1\tau_1 + f_2\tau_2 + \ldots + f_n\tau_n)} df_1 df_2 \ldots df_n$$  \hspace{1cm} (B.16)
B.3 Symmetry of Kernels and Transfer Functions

The Volterra kernels $h_n$ are used to completely characterize the system. But there can be more than one set of kernels that are equivalent and predict the exact same system response. In order to simplify the output calculations, we can require that all of the kernels be symmetric with respect to their input arguments.

From (B.12)

$$y_n(t) = \int^{+\infty}_{\tau_n=-\infty} \int^{+\infty}_{\tau_{n-1}=-\infty} \cdots \int^{+\infty}_{\tau_2=-\infty} \int^{+\infty}_{\tau_1=-\infty} h_n(\tau_1, \tau_2 \ldots \tau_n) x(t - \tau_1) x(t - \tau_2) \prod_{i=3}^{n} x(t - \tau_i) \, d\tau_1 \, d\tau_2 \ldots d\tau_n$$

(B.17)

Or by interchanging the dummy integration variables $\tau_1$ and $\tau_2$, i.e., $\tau_1 \rightarrow \tau_2$ and $\tau_2 \rightarrow \tau_3$

$$y_n(t) = \int^{+\infty}_{\tau_n=-\infty} \int^{+\infty}_{\tau_{n-1}=-\infty} \cdots \int^{+\infty}_{\tau_3=-\infty} \int^{+\infty}_{\tau_2=-\infty} h_n(\tau_2, \tau_1 \ldots \tau_n) x(t - \tau_2) x(t - \tau_1) \prod_{i=3}^{n} x(t - \tau_i) \, d\tau_2 \, d\tau_1 \ldots d\tau_n$$

(B.18)

and changing the order of integration

$$y_n(t) = \int^{+\infty}_{\tau_n=-\infty} \int^{+\infty}_{\tau_{n-1}=-\infty} \cdots \int^{+\infty}_{\tau_3=-\infty} \int^{+\infty}_{\tau_2=-\infty} h_n(\tau_2, \tau_1 \ldots \tau_n) x(t - \tau_2) x(t - \tau_1) \prod_{i=3}^{n} x(t - \tau_i) \, d\tau_1 \, d\tau_2 \ldots d\tau_n$$

(B.19)

Or

$$y_n(t) = \int^{+\infty}_{\tau_n=-\infty} \int^{+\infty}_{\tau_{n-1}=-\infty} \cdots \int^{+\infty}_{\tau_3=-\infty} \int^{+\infty}_{\tau_2=-\infty} h_n(\tau_2, \tau_1 \ldots \tau_n) \prod_{i=1}^{n} x(t - \tau_i) \, d\tau_1 \, d\tau_2 \ldots d\tau_n$$

(B.20)

Therefore we see that changing the order of the arguments of $h_n$ yields the same output. Thus we can restrict the kernels to be symmetric $\Gamma(h_n(\tau_1 \ldots \tau_i, \tau_{i+1} \ldots \tau_j, \tau_{j+1} \ldots \tau_n) = h_n(\tau_1 \ldots \tau_i, \tau_{i+1} \ldots \tau_j, \tau_{j+1} \ldots \tau_n)$ and thus uniquely specify the kernels. Note that symmetry implies that any two arguments of the function can be interchanged and the resulting value will be the same.

From (B.14) we can see that if we restrict $h_n$ to be symmetric then the nonlinear transfer functions $H_n(f_1, f_2 \ldots f_n)$ are also symmetric. This property simplifies the calculation of determining the steady-state frequency domain response of the system described by a set of nonlinear transfer functions.
B.4 Negative Argument of a Transfer Function

In extracting symmetric nonlinear transfer functions, it is useful to understand the relationship between the value of \( H_n(f_1, f_2 \ldots f_n) \) and \( H_n(-f_1, -f_2 \ldots - f_n) \). From (B.14)

\[
H_n(-f_1, -f_2 \ldots - f_n) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} h_n(\tau_1, \tau_2 \ldots \tau_n) e^{-j2\pi(-f_1 \tau_1 - f_2 \tau_2 \ldots - f_n \tau_n)} d\tau_1 d\tau_2 \ldots d\tau_n
\]

(B.21)

Multiplying out the \(-1\) in the exponent yields

\[
H_n(-f_1, -f_2 \ldots - f_n) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} h_n(\tau_1, \tau_2 \ldots \tau_n) e^{+j2\pi(f_1 \tau_1 + f_2 \tau_2 + \ldots + f_n \tau_n)} d\tau_1 d\tau_2 \ldots d\tau_n
\]

(B.22)

Taking the complex conjugate twice of the right hand side results in

\[
H_n(-f_1, -f_2 \ldots - f_n) = \left( \left[ \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} h_n(\tau_1, \tau_2 \ldots \tau_n) e^{+j2\pi(f_1 \tau_1 + f_2 \tau_2 + \ldots + f_n \tau_n)} d\tau_1 d\tau_2 \ldots d\tau_n \right]^* \right)^*
\]

(B.23)

Since \( h_n \) is a real function and \( f \) and \( \tau \) are real variables, we can take the complex conjugate of the integrand.

\[
H_n(-f_1, -f_2 \ldots - f_n) = \left( \left[ \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} h_n(\tau_1, \tau_2 \ldots \tau_n) e^{-j2\pi(f_1 \tau_1 + f_2 \tau_2 + \ldots + f_n \tau_n)} d\tau_1 d\tau_2 \ldots d\tau_n \right]^* \right)^*
\]

(B.24)

By using (B.14) we show

\[
H_n(-f_1, -f_2 \ldots - f_n) = (H_n(f_1, f_2 \ldots f_n))^* = H_n^*(f_1, f_2 \ldots f_n)
\]

(B.25)

Since the value of a transfer function with all of the arguments multiplied by \(-1\) is just the complex conjugate of the transfer function at the original point, we can reduce the number of transfer function points that need to be stored in order to characterize a system. For our case, we choose to store only those transfer function points where the sum of the frequencies is non-negative.

B.5 Balanced Transfer Function Arguments that Add to Zero

We know that the transfer functions are symmetric and also that the value of the arguments to a transfer function multiplied by \(-1\) is just the complex conjugate of
the original transfer function. Using these two properties we can show that the value of any transfer function whose arguments are a symmetric DC FIPD is a real number. A balanced argument to a transfer function is one where every frequency value has a corresponding negative pair e.g. \( H(f_a, f_b, \ldots, f_b, -f_a) \). For an odd order balanced argument there of course will be a single frequency input that does not have a matching negative pair. For a DC FIPD the values of the frequencies sum to zero thus all even balanced symmetric FIPD’s are DC FIPD’s. An odd symmetric FIPD is DC if and only if the frequency with the unmatching pair is equal to zero. Thus for a symmetric DC FIPD as the input to a transfer function

\[
H(f_a, f_b, \ldots, -f_b, -f_a) = H^*(-f_a, -f_b, \ldots f_b, f_a) \\
H^*(f_a, f_b, \ldots, -f_b, -f_a)
\]  

(B.26)

Therefore for any balanced DC FIPD

\[
\text{Im} \{ H(f_a, f_b, \ldots, -f_b, -f_a) \} = 0 
\]  

(B.27)

### B.6 Conclusion

We have presented the basic equations of Volterra series and shown several properties that can be derived from its form. Without loss of generality we can restrict all kernels and thus all Volterra nonlinear transfer functions to be symmetric with respect to their arguments. This simplifies some of the output calculations and guarantees a unique description of a system. We also have looked at properties relating to the signs (+ or −) of the nonlinear transfer functions. These properties help us in developing a process to characterize an unknown system by extracting their nonlinear transfer functions.
printlevel:=-1;
Digits:=20;
rIF:=1000;
i01:=0.0000000000100;
i02:=0.00000000101;
i03:=0.00000000103;
i04:=0.000000000106;
vt:=(0.00008617349*(25+273.15));
e1:= i1 = i01*(exp(v1/vt)-1);
e2:= i2 = i02*(exp(v2/vt)-1);
e3:= i3 = i03*(exp(v3/vt)-1);
e4:= i4 = i04*(exp(v4/vt)-1);
e5:= v1 = (1/2)*vL0 - (1/2)*vRF - vIF ;
e6:= v2 = (1/2)*vL0 + (1/2)*vRF + vIF ;
e7:= v3 = (1/2)*vRF - (1/2)*vL0 - vIF ;
e8:= v4 = vIF - (1/2)*vL0 - (1/2)*vRF ;
e9:= iIF = vIF/rIF ;
f1:= i1 - i2 + i3 - i4 - iIF ;
f2:= subs(e5,e6,e7,e8,subs(e1,e2,e3,e4,e9,f1));
vIF1:= taylor(f2,vIF,4);
save rIF,i01,i02,i03,i04,vt,vIF1,\'VIF\' ;
quit;

Figure C.5: MAPLE code for creating taylor series to solve KCL and KVL of a ring diode mixer.

Appendix C
MAPLE Code for Ring Diode Mixer

C.1 Formulation

The KVL and LCI equations for the circuit in figure 5.1 are given by (5.1)–(5.6). In order to solve these equations for VIF we use the computer–algebra program MAPLE [115] . The MAPLE code given in figure C.5 defines the relationships given in (5.1)–(5.6) and then tells MAPLE to solve for function f2. The relationships of (5.1)–(5.6) are met when f2=0. A taylor series of f2 is then taken with respect to the MAPLE variable vIF (VIF in (5.2)–(5.6)). This taylor series is saved in the file VIF. The resulting file VIF is shown in figure C.6. The file VIF is then copied to VIFt and the “O(vIF**4)” is removed from the last line. Thus when the code in figure C.7 reads the file VIFt the higher order terms (4th order and higher) are
rIF := 1000;
101 := Float(100,-11);
i02 := Float(101,-11);
i03 := Float(103,-11);
i04 := Float(106,-11);
vt := Float(266926260435,-13);

vIF1 := (Float(100,-11)*exp(19.460836706744324355*vL0-19.460836706744324355*vRF
+Float(4,-11)+Float(-101,-11)*exp(19.460836706744324355*vL0+19.460836706744324355*vRF
+Float(-106,-11)*exp(-19.460836706744324355*vL0-19.460836706744324355*vRF
+Float(103,-11)*exp(19.460836706744324355*vRF
-19.460836706744324355*vRF
+Float(-100000000000000000000000000000,-22)*exp(19.460836706744324355*vL0
-19.460836706744324355*vRF
+Float(-38921673413488648709,-27)*exp(19.460836706744324355*vL0
-19.460836706744324355*vRF
+Float(-415697381297967632,-27)*exp(-19.460836706744324355*vL0
-19.460836706744324355*vRF
+Float(-100000000000000000000000000000,-22)*exp(-39310890147623535196,-27)*exp(19.460836706744324355*vL0
+19.460836706744324355*vRF
+Float(-40089323615893308170,-27)*exp(19.460836706744324355*vRF
-19.460836706744324355*vL0)))*vIF+(Float(75744833065313455975,-26)*exp(19.460836706744324355*vL0
-19.460836706744324355*vRF
+Float(78017178057272859654,-26)*exp(19.460836706744324355*vRF
-19.460836706744324355*vRF
+Float(80289523049232263334,-26)*exp(-19.460836706744324355*vRF
+Float(-76502281395966590535,-26)*exp(19.460836706744324355*vL0+19.460836706744324355*vRF
+Float(-10121863749957223487,-24)*exp(19.460836706744324355*vRF
-19.460836706744324355*vL0)*Float(-10416675315489958152,-24)*exp(-19.460836706744324355*vL0
-19.460836706744324355*vRF
+Float(-98270521844244888226,-25)*exp(19.460836706744324355*vL0
-19.460836706744324355*vRF
+Float(-9925322706268737108,-25)*exp(19.460836706744324355*vL0+19.460836706744324355*vRF)))*vIF**3+0(vIF**4);

Figure C.6: MAPLE output of taylor series expansion.
printlevel:=-1;
Digits:=20;
gc(1000);
maxm:=1;
maxn:=1;
read 'VIFt';
VIF:= solve(vIF1,vIF)[1];
save rIF,i01,i02,i03,i04,vt,VIF,'VIFclosed';
quit;

Figure C.7: MAPLE code to create closed form equation of \( V_{IF} \).

effectively truncated.

C.2 Solution

Since the MAPLE equation \( vIF1 \) is now a third order polynomial it can be solved in closed form for the MAPLE variable \( vIF \). This is easily accomplished by line 7 in figure C.7 and the result is saved in the file VIFclosed shown here for completeness. This file can be read into MAPLE and any values of \( vIF \) can be calculated for any given \( vLO \) and \( vRF \). These calculated points are then used to fit a bivariate power series of the form given in (5.9).
\[
\begin{align*}
\text{rIF} & := 1000; \\
i01 & := \text{Float}(100, -11); \\
i02 & := \text{Float}(101, -11); \\
i03 & := \text{Float}(103, -11); \\
i04 & := \text{Float}(106, -11); \\
vt & := \text{Float}(256926260435, -13); \\
\text{VIF} & := \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \langle \angle 

--------- end file VIFclosed ---------------
Appendix D
Instructions for Running Extraction Code

D.1 Overview

The source code for the “measure” program is in ecmw6:/u0/users/MBS/pjl/c/measure. The macro definitions that affect how the program is compiled are documented in the comments of the makefile. I would recommend to always define “SAFE” (e.g. -DSAFe). If this is defined then the program checks for valid ranges of index variables and other “assertions”. This can aid in finding bugs in the code. There are several stages for running the code:

- Edit the source code to use your particular simulator. The code currently uses the IBM internal simulator AS/X (follow on of ASTAP) and expects the output of the simulator to be in rawspice format.
- Create *.status file that contains all of the information that the program needs.
- Run initialization to create directories and calculate the required simulation input.
- Create the input files for the simulations
- Run the simulations
- Extract the H values based on the simulation

D.2 Customizing for a Simulator

In order to customize the extraction code to use a particular simulator there are files that might have to be modified.

- mk.c
  This code creates the input files for the simulations.
- run.c
  This code initiates the simulations.
- get_spect.c
  This code extracts a spectrum of a single simulation.
- extr.c
  This code does the actual extraction and calls the top function in get_spect.c. The file extr.c also calls get_vect.c if the “ffextr” option is used (see section D.7). The file get_vect.c is currently used by get_spect.c and is useful only for rawspice format output files.
D.3 Creating *.status file

All of the simulation steps are related to a single root name. For our examples we will use the name “tmp”. So for our example we need to create the file “tmp.status” in the correct format. The easiest way to do this is to first use the dflt option to create a sample file e.g.

measure dflt tmp

which will return the prompt

This will over write file tmp.status if it exists
Do you want to continue?
YOU MAY LOOSE DATA IF YOU ANSWER y
(y or n): y

The macro NO_ASK can be used when compiling to suppress this prompt. The resulting tmp.status file should look like:

This is a dummy sample description
max_input: 1.1
max_order: 3
xtra_runs: 0
fn: the_name_of_this_file(without the .status)
test_model_lib: dummy_test_model_lib
runcontrol: dummy_runcontrol
acruncontrol: dummy_acruncontrol
loadlib: dummy_loadlib
num_extract_runs: 0
tot_runs: 0
run_num: -1
calc_dc: yes
out_dc_offset: 0
num_freq: 3
freq[0]: 0
freq[1]: 1
freq[2]: 3

The first line is just a comment line and contain anything. The fourth line (fn:) must be changed to

fn: tmp

The first line sets the maximum absolute value of the input signal that is to be used in the extraction. The resulting extracted Volterra nonlinear transfer functions should be used to predict output only when the peak input is less than or equal to this value. The next line is the assumed maximum order of nonlinearity and the third line is the number of extra simulations per run. This must be greater than or equal to zero. A value greater than zero will allow a least square fit where the number of points is greater than the order of the fitting polynomial. For instance
with max order set to 3 and xtra runs set to 1/4 points will be used to fit a third order polynomial. For these curve fits the 0th order coefficient \(a_0\) is known to be zero.

The next lines test_model, lib runcontrol, acruncontrol, and loadlib are specific to the AS/X simulator. The line num_extract runs should be left at 0. This line is updated by the program during the extraction process to allow for multi-step extraction. The line tot runs is updated during the init option and should not be manually changed in most circumstances. The run_num line is updated during the run option (simulations) and can be used to re-start this process at a particular run if the option was interrupted for some reason (e.g., network outage, disk space problems). The starting value of -1 corresponds to the AC simulation used with the extr option.

The next two lines calc_dc and out_dc_offset are used to set the value of \(H_0\). The calc_dc line should be set to either “yes” or “no”. If it is set to “yes” then the value of out_dc_offset will be calculated with a simulation. If set to “no” the desired value of \(H_0\) should be entered by the user. The remaining lines give the input frequencies to be used. If you have more than three input frequencies just add more lines.

### D.4 Initializing

The init option is used for initialization. e.g.,

```
measure init tmp
```

The output to the screen from this step and all other steps is highly dependent on if the following macros are defined at compilation time: VVERBOSE, VBOSET, VQUIET, VQUIET. For more information on these see the comments in the makefile. In our example, the following directories are created: tmp.xfrdir/tmp.inpdir and tmp.rundir. Additionally, the following files are created: tmp.runs/tmp.xfrdir/tmp.H0, tmp.xfrdir/tmp.H1, tmp.xfrdir/tmp.H2, and tmp.xfrdir/tmp.H3. The file tmp.runs contains the input conditions for the simulations and the tmp.xfrdir/tmp.H* files are the initial files where the extracted values will be stored.

### D.5 Creating Simulation Files

The creation of simulation files is performed with the mk (stands for make) option. e.g.,

```
measure mk tmp
```

Files are added to the tmp.inpdir and tmp.rundir directories. To create a file specific to a single run the smk option can be used. For example, to create the files associated with run 2, the following command can be issued:

```
measure smk tmp 2
```
D.6 Running Simulations

The actual simulation requires the most computer resource. To run all of the simulations the following command is issued:

```
measure run tmp
```

To perform the simulations for a single run use the srun option. For instance, to perform the simulations created with the above smk command:

```
measure srun tmp 3
```

The run command can be used to start at a particular run number and continue to the end by editing the `.status` file and changing the `run_num` line.

D.7 Extracting Values

To read the simulation results and calculate the Volterra nonlinear transfer functions enter:

```
measure extr tmp
```

The extr command can also be used to restart the extraction at a particular run number and continue to the end by editing the `.status` file and changing the `num_extract_runs` line. Care must be taken when doing this because multiple runs with the same input frequency should not be broken up with this technique. The format of the `.xfrdir/*.H` files can be changed from ascii to binary by editing these files before the extraction procedure is run. To make this change just change the word “ascii” to “binary”.

The `fextr` option can be used in place of the `extr` option if the linear response \( H_1 \) is to be taken from a linear AC simulation.

D.8 Calculating from Tables

The extracted model output can be calculated with the calc option. The format for our example is:

```
measure calc tmp
```

The file `tmp.calcin` must be created to give the input for the model calculation. Below is an example:

```
num_freq: 2
period: 5e-7
  freq[0]: 1 real[0]: -0.27 imag[0]: 0.123
  freq[1]: 3 real[1]: 0.19 imag[1]: 0.456
```

The output will be written to the two files `tmp.calcout` and `tmp.rawout`. The file `*.calcout` lists the output phasors in the form similar to the `*.calcin` file and the `*.rawout` is the time domain representation in rawspice format. The spice waveform display program “nutmeg” can be used to display the waveforms.