Model Order Reduction Techniques for Circuit Simulation

Luis Miguel Silveira

RLE Technical Report No. 592

February 1995

The Research Laboratory of Electronics
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE, MASSACHUSETTS 02139-4307

This work was supported in part by the Semiconductor Research Corporation under Contract SRC 93-SJ-558 and in part by the National Science Foundation/Advanced Research Projects Agency under Grant MIP 91-17724.

Model Order Reduction Techniques for Circuit Simulation

by

Luís Miguel Silveira

Submitted to the Department of Electrical Engineering and Computer Science
on May 16, 1994, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Theoretical and practical aspects of model order reduction techniques for use in the context of circuit simulation are investigated, with particular attention to problems involving clocked analog circuitry and to interconnect and packaging applications.

First, an algorithm for the efficient simulation of clocked analog circuits is described and simulation results are presented. Traditional simulation programs, which must accurately solve the associated differential equations with a time discretization method become extraordinarily computationally expensive when applied to simulating the transient behavior of clocked analog circuits. These circuits are clocked at a frequency whose period is typically orders of magnitude smaller than the time interval of interest to the designer. The nature of the calculations requires that in order to construct the solution over the time interval of interest, an accurate solution must be computed for every cycle of the high frequency clock in the interval, and this can involve thousands of cycles. The algorithm to be described substantially reduces the simulation time without compromising accuracy by exploiting the property that the behavior of such a circuits in a given high frequency clock cycle is similar, but not identical, to the behavior in the preceding and following cycles. This algorithm is in itself a model order reduction technique, since it simplifies the understanding of the problem and reduces its computational cost. Further model order reduction is possible which allows for significant speedups in circuits containing digital control circuitry.

Next, we describe an algorithm for efficient SPICE-level simulation of frequency-dependent elements, such as transmission lines with arbitrary scattering parameter descriptions, or complicated 3-D interconnect with nonlinear transistor drivers and receivers. The elements can be represented in the form of a frequency-domain model or a table of measured frequency-domain data. Our approach initially uses a forced stable decade-by-decade ℓ_2 minimization approach to construct a sum of rational functions approximation, which may have dozens of poles and zeros. This unnecessarily high-order model is then reduced using a guaranteed stable model order reduction scheme based on balanced realizations. Once the reduced-order model is derived, an impulse response can easily be generated. Finally, the impulse response can be efficiently incorporated into a circuit simulator using recursive convolution. Examples including a transmission line with skin-effect and a system with a set of package pins connecting on-chip drivers to off-chip receivers, are examined to both demonstrate the effectiveness of the approach and to show its generality.

The results from both applications are encouraging and demonstrate that model order reduction techniques can be an extremely useful tool for circuit simulation problems and can lead to substantial savings in the simulation of many types of circuits.

Thesis Supervisor: Jacob K. White
Title: Associate Professor of Electrical Engineering and Computer Science

To my wife Júlia, with all my love

Experience is the name everyone gives to their mistakes
- Oscar Wilde, *"Lady Windermere's Fan"*

Patience: a minor form of despair disguised as a virtue
- Ambroise Pierce, *"The Devil's Dictionary"*

Acknowledgments

First and foremost I must acknowledge the guidance, help, motivation and expertise of my thesis adviser Prof. Jacob K. White. Throughout my years at MIT he always impressed me with his knowledge and his ability and willingness to share it with others. Not only did he supervise my work, ensuring its progress, quality and relevance, but he also encouraged and taught me how to supervise myself and others.

I am also grateful to my thesis readers, Profs. Jonathan Allen and Steven Leeb for all the help and consideration they extended to me during the course of this work. Prof. Leeb did much of the ground work for Chapter 3 and was always available for discussions on the matter. Prof. Allen has been a constant source of encouragement since my arrival at MIT. In large part through his efforts and perseverance, the CAD group at the Research Laboratory of Electronics remains a first-rate research environment of which I had the privilege to be a member.

Several other members of my research group have contributed directly or indirectly to the work described in this thesis. Ricardo Telichevesky and Keith Nabors were more than colleagues and friends. Their company and support made the good times better and the bad times merely forgettable. I had the pleasure of working closely with Ibrahim Elfadel from whom I received extensive assistance. Abe was always a knowledgeable resource for any sort of question of a theoretical nature. Andrew Lumsdaine, while at MIT, was a friend and a companion. We collaborated on several large projects that helped shape my knowledge in the field. I would also like to acknowledge the help and friendshipness of José Carlos Monteiro, Mattan Kamon, Ignacio McQuirk, Robert Armstrong, Prof. Srinivas Devadas, and others, too many to mention, students and faculty at the CAD group at RLE and at other laboratories at MIT.

I am extremely grateful to the Serpa and Brasil families, who received me as one of them, and helped me throughout the years. Their friendship, their support and their mere presence was fundamental in keeping my peace of mind.

I would also like to acknowledge the support and consideration I have received over the years from Professors Luís Vidigal and Horácio Neto, from my research group at INESC in Portugal.

For all the support, love and understanding they have given me throughout the years, I wish to thank my parents, Fernando Augusto and Maria Luísa, my brother Eduardo and other members of my family.

Finally, completion of this thesis would not have been possible without the love and the encouragement I received from my wife Júlia. Her patience and support during the course of my graduate program have held us as a family and the joys and pains we went through together have made us stronger and closer than we thought possible. The recent birth of our daughter Ana Teresa has filled our lives with more happiness than we ever imagined.

During the course of this work I was fortunate to receive the support of several agencies at various times. I would like to thank IBM, the Fullbright Committee and the I.I.E., the Portuguese INVOTAN committee and the Portuguese "Junta Nacional de Investigação Científica e Tecnológica".

Contents

List of Figures

List of Tables

List of Acronyms

GE	Gaussian elimination
GJ	Gauss-Jacobi
GS	Gauss-Seidel
EF	Envelope-Following
KCL	Kirchoff's Current Law
KVL	Kirchoff's Voltage Law
LR	Linear Relaxation
MIMO	Multiple Input Multiple Output
MOS	Metal-Oxide-Semiconductor
MOSFET	MOS Field Effect Transistor
NLR	Nonlinear Relaxation
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
SISO	Single Input Single Output
TBR	Truncated Balanced Realization
VLSI	Very Large Scale Integration
VCCS	Voltage-Controlled Current Source
WR	Waveform Relaxation

List of Symbols

Vectors

Members of a vector space are denoted in lower-case with the bold math font, e.g., $\boldsymbol{x}$. The vector space in question may be $\mathbb{R}^n$, or a function space. Components of a multi-dimensional vector are denoted in the plain math font, with the appropriate subscript, e.g., x_i. Constant vectors are sometimes indicated with a subscript, e.g., $\boldsymbol{x}_0$. The symbols for some particular vectors used in this thesis are:

$\boldsymbol{x}$ The vector of state variables in a linear system.

$\boldsymbol{u}$ The vector of inputs to a linear system.

$\boldsymbol{y}$ The vector of outputs of a linear system.

$\boldsymbol{r}$ Residual vector

$\boldsymbol{e}$ Error vector. The elementary basis vectors for $\mathbb{R}^n$ are denoted $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$.

Matrices

Matrices are denoted in upper-case with the bold math font, e.g., $\boldsymbol{A}$. Components of a matrix are denoted in upper- or lower-case with the plain math font, using the appropriate subscript, e.g., A_{ij} or a_{ij}. The algebraic or Hermitian transpose of a matrix is denoted as $\boldsymbol{A}^T$. The identity matrix is denoted by $\boldsymbol{I}$ or by $\boldsymbol{I}_n$ indicating the matrix size.

Operators

Operators are denoted in upper-case with the script font, e.g., $\mathcal{F}$.

Spaces

Spaces are denoted with the blackboard bold math font, e.g., $\mathbb{R}^n$. The symbols for some particular spaces used in this thesis are:

$\mathbb{L}_1([0,T],\mathbb{R}^n)$ Function space of modulo-integrable functions (in the Lebesgue sense) mapping from the interval $[0,T]$ to $\mathbb{R}^n$.

$\mathbb{L}_2([0,T],\mathbb{R}^n)$ Function space of square-integrable functions (in the Lebesgue sense) mapping from the interval $[0,T]$ to $\mathbb{R}^n$.

$\mathbb{C}^1([0,T],\mathbb{R}^n)$ Function space of continuously differentiable functions mapping from the interval $[0,T]$ to $\mathbb{R}^n$.

$\mathbb{N}$ Set of natural numbers, i.e., $\{1,2,\ldots\}$.

$\mathbb{R}^n$ n-dimensional Euclidean space.

$\mathbb{C}^n$ n-dimensional complex space.

Complex Functions of Complex Variables

Complex functions of complex variables are denoted in upper-case with the bold-math font, e.g. $\boldsymbol{H}(s)$. If the function is a rational function, occasionally a subscript can be used to indicate the order of the denominator polynomial, e.g. $\boldsymbol{H}_k(s)$

Miscellaneous

The following is a brief description of other miscellaneous nomenclature:

$()^k$ The superscript k denotes the k^{th} iterate of the associated quantity, e.g., $\boldsymbol{x}^k$ is the k^{th} iterate of $\boldsymbol{x}$. Indexing generally begins from $k = 0$ but sometimes also from $k = 1$. Which is the case should be clear from the context.

$()_k$ The subscript k denotes the k^{th} component of the associated quantity, as mentioned above. Component indexing generally begins from $k = 1$. Occasionally, the subscript is also used in reference to the order of an approximation, but those cases will be clear from the context.

$\boldsymbol{j}$ Denotes the imaginary unit, $\boldsymbol{j} = \sqrt{-1}$. The lower-case bold math font is used to distinguish it from the the letter j.

$\mathbf{o}$ Denotes the null vector of appropriate size, that is the vector with all zero components.

$\Omega(\cdot)$ Lower bound complexity.

$\mathcal{O}(\cdot)$ Upper bound complexity.

1

Introduction

The ever increasing size and complexity of today's integrated circuit designs and the continuous development of processing technologies has led to an increase in the fabrication costs and development time of such designs. In order to minimize these growing costs, accurate and reliable verification of each project before fabrication becomes imperative, so that fabricated circuits function correctly on "first silicon".

As circuits become larger and more complex, design techniques become more elaborate and innovative in order to fully exploit the available space. Also, the continuing trend for increasing speeds of operation makes it impossible to discount previously ignored effects of the interaction between circuits and their surrounding environment. Therefore, it becomes necessary that accurate circuit simulation methods be applied instead of other more intuitive but less detailed approaches to verification, such as timing simulation or functional simulation. For all practical purposes, the general circuit simulation problem is considered as having been solved. Conventional circuit simulators, developed during the 1970's, like SPICE [1] or ASTAP [2] are capable of simulating virtually any circuit given enough resources in terms of time and computational power. However, full verification of the functionality of a system must not only be reliable, but also fast enough to provide results in an acceptable time interval. This implies that the computational resources that must be allocated for the simulation task also increase dramatically and in some cases the amount of resources needed becomes inadmissibly expensive.

One way to address this problem is through the usage of faster computer hardware. While this may be as simple as using a faster mainframe computer or a new generation workstation, it can also be viewed in the context of parallel computation. Parallel circuit simulation has been successfully applied to specific classes of circuits with very encouraging results [3]. With some degree of confidence it is believed that this is true whenever the sheer size of the circuit or the nature of the computation it performs make it amenable to parallel simulation. However, it is also recognized that for the most part, the circuit simulation paradigm does not contain enough structure for efficient parallelization [4]. In fact in many cases the nature of the difficulty may not be related to the size of the circuit itself but merely to the length of the simulation interval, or to the complexity of the models of the elements involved. In such cases it is believed that improving the efficiency of the simulation process can only be accomplished by devising specific algorithms that exploit the characteristic features of the particular problem to be solved.

Transient circuit simulation involves forming the system of nonlinear ordinary differential equations (ODE) that describes the dynamics of a circuit, and solving them numerically. The typical approach is to first discretize the set of ODE's in time with an integration method. Since stiff ODE systems require the use of implicit integration schemes, the time discretization generates a sequence of nonlinear algebraic problems which are solved with an iterative method, usually some variant of the Newton-Raphson algorithm. The sequence of linear algebraic systems generated at each iteration of the nonlinear solution method are then solved via Gaussian elimination or possibly some iterative linear solution algorithm.

Two possible computational bottlenecks are usually identified in standard circuit simulation. One is the function evaluations necessary to compute the components of the Jacobian and right hand side vector used at each Newton-Raphson iteration. These function evaluations correspond to the computation of the device's constitutive relations and their derivatives. The cost of these evaluations is general dominated by the Jacobian computations. For a circuit with n nodes, that cost can be as low as $\mathcal{O}(n)$ or as high as $\mathcal{O}(n^2)$ depending on the degree of connectivity of the circuit. The other usual bottleneck is the linear system solution performed at each Newton-Raphson iteration. The complexity of direct elimination methods for solving systems of equations is polynomial in n, typically from $\mathcal{O}(n^{1.5})$ for sparse problems to $\mathcal{O}(n^3)$ for dense problems.

Which of these bottlenecks will dominate depends on the particular problem. If the circuit is very large, then eventually the linear solution task will dominate the total complexity. On the other hand, if the circuit is small or only moderately large, then the device evaluation task will in general dominate. However, one should be very careful when weighting the costs associated with the circuit simulation process using complexity issues alone as this can sometimes lead to erroneous conclusions. In fact, even assuming the number of circuit nodes to be small, the real cost of each Newton iteration might still be extremely large if the evaluation of some or all of the devices is computationally very expensive. An example of this is the case where the device might be described via its frequency response, thus requiring at each timepoint a convolution with the derived impulse response. Furthermore, even if the device evaluation task has negligible cost, it may happen that the integration interval is so large that the sheer number of timepoints necessary for accurate computation of the ODE solution makes the total simulation cost overwhelming. A simple example of this situation is the case of clocked analog circuits such as power converters or phase-locked loops. It seems therefore hopeless to expect that standard circuit simulation algorithms be able to handle every conceivable scenario with acceptable efficiency. For these reasons, a pertinent question is whether it is possible to achieve efficient simulation of certain classes of circuits by using algorithms or techniques specifically tuned to exploit the characteristic features of the particular problem to be solved and make them more tractable.

One possible answer to this quest is through **Model Order Reduction**. Model order reduction is a process by which, given a complex dynamic system, one obtains a simpler model that approximates the full system in some adequate way. The main reasons for obtaining lower-order models are twofold: to facilitate an understanding of the system, or to reduce computational efforts in the simulation process. In this light, model order reduction is an approximation technique at the system level, and any approximation or modeling technique leads to the classical dilemma between accuracy and simplicity. Only knowledge of the specific problem enables the designer to establish reasonable accuracy criteria for the approximation and allows him to fully understand the nature of the tradeoff involved. In the context of circuit simulation one can envision the application of model order reduction techniques in many ways, depending on the level at which it is applied. Macromodeling of a circuit element with a functionally equivalent but simpler circuit or function is a model order reduction technique. Similarly extracting the important characteristics of an otherwise complex circuit behavior can be viewed as a model order reduction technique.

In this thesis we will study some theoretical and practical aspects of model order reduction techniques for use in the context of circuit simulation. Two different types of approaches will be considered. One is to look at the model order reduction technique as a way to simplify the complexity of the computations in an otherwise computationally expensive problem. In this context, the example studied is that of clocked analog circuits. Another approach taken is to consider model order reduction as a macromodeling technique applied in this case to the modeling of frequency-dependent interconnect and packaging structures.

A review of the circuit simulation problem and the standard techniques and algorithms necessary for its solution are given in Chapter 2. In Chapter 3, algorithms are developed for handling the case of clocked analog circuits with high frequency clocks. These algorithms focus on the slow underlying behavior of the circuit nodes which is generally what is of interest to the designer. Frequency-dependent elements, either described by a functional relationship or by tabulated or measured data are studied in Chapter 4, where a guaranteed stable algorithm is presented that allows easy incorporation of the device models into an electrical level circuit simulator. Finally in Chapter 5 some conclusions are drawn and suggestions for further work are offered.

References

[1] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Tech. Rep. ERL M520, Electronics Research Laboratory Report, University of California at Berkeley, Berkeley, California, May 1975.

[2] W. T. Weeks, A. J. Jimenez, G. W. Mahoney, D. Mehta, H. Quasemzadeh, and T. R. Scott, "Algorithms for ASTAP - A network analysis program," *IEEE Transactions on Circuit Theory*, pp. 628–634, November 1973.

[3] A. Lumsdaine, *Theoretical and Practical Aspects of Parallel Numerical Algorithms for Initial Value Problems, with Applications.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1992.

[4] L. M. Silveira, "Circuit simulation algorithms for massively parallel processors," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1990.

2

Review of Numerical Techniques for Circuit Simulation

2.1 Introduction

In this chapter the standard numerical techniques used in the solution of the circuit simulation problem are reviewed. The presentation is not exhaustive in any conceivable way, nor is that the intention. The function of this chapter is merely to provide the reader with enough background to facilitate the understanding of the material in subsequent chapters.

Electrical circuit simulation begins with the formulation of the system of nonlinear ordinary differential equations which govern its behavior. These equations are introduced in Section 2.2 and the standard algorithms used to compute their solution are also described. Direct methods based on some form of sparse Gaussian elimination for the solution of the linear systems obtained at each timepoint are reviewed in Section 2.3, and relaxation methods are briefly reviewed in Section 2.4. Finally Section 2.5 debates some issues concerning the evaluation of circuit devices.

2.2 Formulation of System Equations

The electrical behavior of a circuit in the time domain can be described by a system of equations involving node voltages, resistive currents, capacitor charges and inductor fluxes. This system of equations can be constructed from a circuit description using the Kirchoff current and voltage laws (KCL and KVL) and the constitutive or branch equations of each circuit element. For simplicity in this chapter we will restrict ourselves to noninductive circuits, and present a technique referred to as nodal analysis [1, 2, 3] used to formulate those equations. It is possible to extend the nodal analysis technique to include circuits with inductors and voltage sources by using modified nodal analysis [4], while still preserving the form of many of the equations and algorithms that will follow. For the most part that would only involve changing the physical meaning of some of the unknowns to be used. Therefore, almost all the comments made in the following discussion about these circuits will apply to circuits with inductive elements. Whenever appropriate, comments will be made regarding the necessary changes to handle more generic types of circuits.

Let us then assume for simplicity that in the circuits to be studied all resistive elements, including active devices, are characterized by constitutive equations where voltages are the controlling variables and currents are the controlled variables, that energy storage elements are represented by voltage-controlled charge devices and that all voltage sources have one terminal connected to ground or can be transformed into current sources with the use of the Norton transformation. Then nodal analysis yields a system of equations of the form:

$$\frac{d}{dt}\boldsymbol{p}\left(\boldsymbol{v}(t),\boldsymbol{u}(t)\right)+\boldsymbol{f}\left(\boldsymbol{v}(t),\boldsymbol{u}(t)\right)=\mathbf{o} \qquad \boldsymbol{v}(0)=\boldsymbol{v}_0. \tag{2.1}$$

Here $\boldsymbol{p}\left(\boldsymbol{v}(t),\boldsymbol{u}(t)\right) \in \mathbb{R}^n$ is the vector of sums of capacitive charges at each node, $\boldsymbol{f}\left(\boldsymbol{v}(t),\boldsymbol{u}(t)\right) \in \mathbb{R}^n$ is the vector of sums of resistive currents at each node, $\boldsymbol{u}(t) \in \mathbb{R}^n$ is the vector of input voltages, and $\boldsymbol{v}(t) \in \mathbb{R}^n$ is the vector of node voltages with initial values $\boldsymbol{v}(0) = \boldsymbol{v}_0$ and n is the number of circuit nodes excluding the reference. Each equation in this system corresponds to applying KCL to each node in the circuit and using the constitutive equations of the branch elements to express their currents in terms of the node voltages. If the circuit contains inductive elements, some of our unknowns would now be branch currents and some of our equations would be derived from applying KVL to the branches containing such elements. Nevertheless, the basic form of Eqn. (2.1) would still be preserved.

2.3 Standard Circuit Simulation Solution

To solve the systems of nonlinear ordinary differential equations (ODE's) in Eqn. (2.1) in the interval $t \in [0, T]$, the interval is discretized and a numerical solution is computed at each point of the discretization. This is usually accomplished using Linear Multistep

Formulas [5, 6] which when applied to the first order ordinary differential equation

$$\frac{d}{dt}\boldsymbol{x}(t) = \boldsymbol{g}(\boldsymbol{x}(t), t) \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0 \qquad t \in [0, T], \tag{2.2}$$

have the general form

$$\sum_{i=0}^{s} \alpha_i \boldsymbol{x}(t_{m+1-i}) = h_m \sum_{i=0}^{s} \beta_i \boldsymbol{g}(\boldsymbol{x}((t_{m+1-i}), t_{m+1-i}) \tag{2.3}$$

where $h_m = t_{m+1} - t_m$ is the discretization timestep, $\boldsymbol{x}(t_m)$ is the estimated value of the solution at time $t = t_m$, $\boldsymbol{g}(\boldsymbol{x}((t_{m+1-i}), t_{m+1-i})$ represents the dynamics of the equation at the given timepoint using the estimated value, the parameters α_i and β_i are chosen for accuracy within stability limits, and s is the order of the formula [5, 7].

If $\beta_0 = 0$ in Eqn. (2.3) $\boldsymbol{x}(t_{m+1})$ can be found explicitly as a function of previous values of $\boldsymbol{x}$ and $\boldsymbol{g}(\cdot)$ evaluated at previous values of $\boldsymbol{x}$. For $\beta_0 \neq 0$, an implicit equation in terms of $\boldsymbol{x}(t_{m+1})$ and $\boldsymbol{g}(\boldsymbol{x}(t_{m+1}), t_{m+1})$ must be solved for $\boldsymbol{x}(t_{m+1})$. Hence, the method is called an "explicit method" for $\beta_0 = 0$ and an "implicit method" for $\beta_0 \neq 0$.

For problems such as (2.2), explicit methods have a potentially significant advantage over implicit methods because there is no need to perform a nonlinear system solution. This results in in a substantial reduction in computation as well as storage. However, explicit methods are far less stable than their implicit counterparts. For problems having eigenvalues that differ by several orders of magnitude (i.e., stiff problems), implicit methods are computationally superior to explicit methods. The stability of the implicit methods allows for substantially larger timesteps, resulting in lower overall computational work for the simulation. Explicit integration methods also lose their advantages when the differential portion of the initial value problem is itself implicit, as is the case with generic circuit simulation as denoted by Eqn.(2.1). A linear multistep integration formula applied to solving (2.1) results in:

$$\sum_{i=0}^{s} \alpha_i \boldsymbol{p}\left(\boldsymbol{v}(t_{m+1-i})\right) + h_m \sum_{i=0}^{s} \beta_i \boldsymbol{f}\left(\boldsymbol{v}(t_{m+1-i}), t_{m+1-i}\right) = \mathbf{o}. \tag{2.4}$$

Even if an explicit integration method is used, i.e., by choosing $\beta_0 = 0$, (2.4) is still implicit in $\boldsymbol{v}(t_{m+1})$. Therefore, a nonlinear solution step would be required, but without gaining the stability inherent to an implicit integration method. In special cases, some advantage can be gained because $\boldsymbol{p}$ may be easier to invert than $\boldsymbol{f}$, but in general this advantage is not enough to compensate for the lack of the strong stability properties of implicit integration methods.

In circuit simulation, the trapezoidal formula, a 2^{nd} order implicit linear multistep formula, is frequently used to approximate the system of Eqn. (2.1) by a sequence of implicit algebraic equations. Given a timestep, h, the trapezoidal integration algorithm is defined as:

$$\boldsymbol{x}(t_{m+1}) = \boldsymbol{x}(t_m) + \frac{h}{2}\left(\boldsymbol{g}(\boldsymbol{x}(t_{m+1}), t_{m+1}) + \boldsymbol{g}(\boldsymbol{x}(t_m), t_m)\right) \tag{2.5}$$

When applied to Eqn. (2.1) the trapezoidal formula yields:

$$\boldsymbol{p}\left(\boldsymbol{v}(t+h), \boldsymbol{u}(t+h)\right) - \boldsymbol{p}\left(\boldsymbol{v}(t), \boldsymbol{u}(t)\right) + \frac{h}{2}\left[\boldsymbol{f}\left(\boldsymbol{v}(t+h), \boldsymbol{u}(t+h)\right) + \boldsymbol{f}\left(\boldsymbol{v}(t), \boldsymbol{u}(t)\right)\right] = \mathbf{o} \tag{2.6}$$

where $\boldsymbol{v}(t)$ is known, and the equation must be solved to compute $\boldsymbol{v}(t+h)$.

Using a linear multistep formula to discretize the system ODE's has replaced the problem of solving a system of first-order differential equation with that of solving a system of nonlinear algebraic equations at every timepoint in the discretization interval. The usual way to solve Eqn. (2.6) is to use an iterative nonlinear solver. The most popular of such methods is Newton's method or one of its variants. If the standard multidimensional Newton method is used to solve Eqn. (2.6), the new iterate $\boldsymbol{v}^{k+1}(t+h)$ is derived from $\boldsymbol{v}^k(t+h)$ by solving the linear system of equations

$$\boldsymbol{J_F}\left(\boldsymbol{v}^k(t+h)\right)\boldsymbol{\Delta v}^k(t+h) = -\boldsymbol{F}(\boldsymbol{v}^k(t+h)) \tag{2.7}$$

where

$$\boldsymbol{\Delta v}^k(t+h) = \boldsymbol{v}^{k+1}(t+h) - \boldsymbol{v}^k(t+h). \tag{2.8}$$

Solution of Eqn. (2.7) involvs the computation of the residue at the k^{th} step, $\boldsymbol{F}\left(\boldsymbol{v}^k(t+h)\right)$, defined as

$$\begin{aligned}\boldsymbol{F}\left(\boldsymbol{v}^k(t+h)\right) = \boldsymbol{p}\left(\boldsymbol{v}^k(t+h), \boldsymbol{u}(t+h)\right) - \boldsymbol{p}\left(\boldsymbol{v}(t), \boldsymbol{u}(t)\right) - \\ \frac{h}{2}\left[\boldsymbol{f}\left(\boldsymbol{v}^k(t+h), \boldsymbol{u}(t+h)\right) + \boldsymbol{f}\left(\boldsymbol{v}(t), \boldsymbol{u}(t)\right)\right]\end{aligned} \tag{2.9}$$

and the Jacobian of $\boldsymbol{F}\left(\boldsymbol{v}^k(t+h)\right)$, $\boldsymbol{J_F}\left(\boldsymbol{v}^k(t+h)\right)$ which is given by

$$\boldsymbol{J_F}\left(\boldsymbol{v}^k(t+h)\right) = \frac{\partial \boldsymbol{p}\left(\boldsymbol{v}^k(t+h), \boldsymbol{u}(t+h)\right)}{\partial \boldsymbol{v}} - \frac{h}{2}\frac{\partial \boldsymbol{f}\left(\boldsymbol{v}^k(t+h), \boldsymbol{u}(t+h)\right)}{\partial \boldsymbol{v}}. \tag{2.10}$$

The Newton iteration is continued until $\left|\left|\boldsymbol{v}^{k+1}(t+h) - \boldsymbol{v}^k(t+h)\right|\right| < \epsilon$, where $\epsilon > 0$ is a small number and $\boldsymbol{F}\left(\boldsymbol{v}^k(t+h)\right)$ is close enough to zero. Newton's method converges quadratically, provided the initial guess, $\boldsymbol{x}^0$, is sufficiently close to the exact solution [8]. Eqn. (2.7) is now a system of linear equations of the form

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.11}$$

which must be solved at each iteration of the Newton method.

The standard Newton method has some potential drawbacks. First, a linear system solution step is required at each iteration, which can be expensive in terms of computation and in terms of storage, especially if a direct factorization method is used. Second, global convergence can be problematic if the initial guess is not close enough to the exact solution. Both of these difficulties have been addressed extensively in the literature and modified versions of Newton's method as well as other algorithmic solution methods are known to exist [8, 9, 10, 11, 12, 13].

The standard algorithm for solving the linear system of equations derived from Eqn. (2.7) is sparse Gaussian elimination (GE), typically implemented with standard LU-factorization techniques. This algorithm decomposes the matrix $\boldsymbol{A}$ into lower and upper triangular factors $\boldsymbol{L}$ and $\boldsymbol{U}$, respectively, such that $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$. The solution $\boldsymbol{x}$ to Eqn. (2.11) is computed by first solving $\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}$ with a forward elimination process and then solving $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{y}$ with backward substitution.

A discussion of direct methods can be found in most linear algebra or numerical methods texts [14, 15, 16]. The main advantage of direct methods is reliability. With exact arithmetic, the solution $\boldsymbol{x}$ can be computed exactly in a fixed number of steps. However, direct methods present two major drawbacks: computational complexity and storage. The complexity of direct elimination methods for solving linear systems of equations is polynomial in n, the size of the matrix, typically from $\mathcal{O}(n^{1.5})$ for sparse problems to $\mathcal{O}(n^3)$ for dense problems (it is actually known that the commonly cited complexity of $\mathcal{O}(n^3)$ for dense matrix problems is not optimal; algorithms are known that have better complexity, as low as $\mathcal{O}(n^{2.8})$ [17]).

Since the matrices that describe the linear systems in circuit simulation problems are usually sparse as a consequence of the sparse connectivity pattern of most circuits, most of the matrix entries are zero. It is therefore possible to obtain substantial memory savings by storing only those nonzero matrix entries. However, direct methods also require storage for the fill-in elements, i.e., matrix zero locations which become non-zero as the elimination process proceeds. Clever data structures and careful reordering algorithms have been used to reduce the complexity of Gaussian elimination for sparse matrices associated with circuit problems to a range of from $\mathcal{O}(n)$ to $\mathcal{O}(n^{1.5})$ [18]. However, sparse matrix techniques based on Gaussian elimination still grow superlinearly with the

Algorithm 2.3.1 (Direct Method Circuit Simulation).

```
simulate()
{
  $t = 0$ ;
  formulate the circuit equations ;
  while (t < Simulation_Time) {
    select $h$ ;
    repeat {
      use Newton's method to solve the nonlinear algebraic
        equations;
      solve the resulting linear algebraic system with sparse
        Gaussian elimination
    } until (converged) ;
    $t = t + h$ ;
  }
}
```

number of unknowns, and therefore other solution methods have been attempted which address this problem. In the next section we will mention briefly alternatives to some of these methods.

The standard circuit simulation outlined in this section is known as the *Direct Method* for solving Eqn. (2.1) and is summarized in Algorithm 2.3.1. As shown, the algorithm indicates that at every timepoint a new timestep is chosen. The timestep selection is in general done automatically based on a number of parameters which may include the number of iterations required for Newton convergence, the local truncation error estimate and other accuracy-related parameters. It may happen that at some timepoint, the Newton will either fail to converge or produce a solution which does not satisfy the users' accuracy requirements. In this case the timestep must be rejected and a new, smaller timestep chosen. Automatic timestep selection is a fuzzy area, generally implementation dependent, in which considerable work has been done [19, 20].

2.4 Relaxation-Based Circuit Simulation

Relaxation methods comprise one class of iterative solution methods for systems of equations which grow more slowly with problem size than the standard direct methods, that is Newton's method followed by sparse Gaussian elimination. In particular, the computation per iteration grows linearly with the number of nonzero jacobian matrix entries. The disadvantage of relaxation is that each iteration only produces an *approximate* solution to a linear system, and repeated relaxation iterations do not necessarily improve that approximation. For some circuit simulation problems, relaxation methods work well because repeated relaxation iterations converge rapidly to the exact solution [21].

Relaxation methods can be used to solve Eqn. (2.1) in a number of ways. Applying relaxation to the solution of Eqn. (2.1) has in general a simple circuit interpretation. For instance, when updating the voltage at a particular node, one could consider all the other nodes in the circuit as constant voltages sources. One could then solve the corresponding scalar equation for that node voltage and afterwards use the same procedure for the other nodes. If this is performed in an iterative fashion, then one could hope that after a certain number of iterations a convergent set of node voltages would be obtained. Viewed in this way, relaxation methods perform a decoupling of the circuit equations or equivalently a partitioning of the circuit nodes. The simplest relaxation method is the Richardson iteration [22, 23], but by far the two most common relaxation methods used in electrical circuit simulation are the Gauss-Jacobi method and the Gauss-Seidel Method [8]. The interpretation above describes both of these methods fairly well. They differ only in the fact that at each relaxation iteration the Gauss-Jacobi method only uses values from previous iterations, while the Gauss-Seidel method uses the most recent updates for each node's voltage, which is the reason why Gauss-Seidel has a faster convergence than Gauss-Jacobi for most problems.

Relaxation methods can be applied at different stages of the solution process. If applied to the linear equation (2.7) these methods are known as Linear Relaxation (LR) [23]. If applied to the nonlinear equation (2.6) they are called Nonlinear Relaxation (NLR) [8] and if applied to the differential equation (2.1) directly they are known as Waveform Relaxation (WR) [24, 25]. Under very mild conditions, relaxations algorithms can be proven to be convergent[23, 25, 26, 27].

For completeness, and also to give the reader a flavor for iterative methods and relaxation methods in particular we include more detailed descriptions for the Gauss-Jacobi versions of the linear and nonlinear relaxation methods mentioned.

Linear Relaxation can be used in lieu of sparse Gaussian elimination in the solution of the linear problem of Eqn. (2.7). More generally, when applied to a linear system of the form $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = 0$ where $\boldsymbol{x} = (x_1, ..., x_n)^T, \boldsymbol{b} = (b_1, ..., b_n)^T, x_i, b_i \in \mathbb{R}$, and $\boldsymbol{A} = (a_{ij})$, $A \in \mathbb{R}^{n \times n}$, the Gauss-Jacobi relaxation method attempts to solve it by solving the equations one row at time, using for the other variables the values from previous iterations. This

Algorithm 2.4.1 (Linear Gauss-Jacobi Relaxation for solving $\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{b} = 0$).

```
linear_Gauss_Jacobi_relaxation()
{
    $k = 0$;
    guess some $\boldsymbol{x}^0$
    repeat {
        $k = k + 1$;
        for $(i = 1;\ 1 \leq n;\ i++)$ {
            $x_i^k = \frac{1}{a_{ii}} \left[ b_i - \sum_{j=1}^{i-1} a_{ij} x_i^k + \sum_{j=i+1}^{n} a_{ij} x_i^{k-1} \right]$;
        }
    } until $\left( \left\| \boldsymbol{x}^k - \boldsymbol{x}^{k-1} \right\|_1 \leq \epsilon \right)$
}
```

is shown in Algorithm 2.4.1, where the superscript k is the iteration count and ϵ is a small positive number. The Gauss-Jacobi and Gauss-Seidel iterations can be described compactly in matrix form. Let $\boldsymbol{L}$, $\boldsymbol{U}$, and $\boldsymbol{D}$ be the strictly lower triangular, strictly upper triangular, and diagonal of $\boldsymbol{A}$, respectively. Then, the Gauss-Jacobi and Gauss-Seidel algorithms can be expressed as

$$\boldsymbol{D}\boldsymbol{x}^{k+1} = \boldsymbol{b} - (\boldsymbol{L} + \boldsymbol{U})\boldsymbol{x}^k,$$

and

$$(\boldsymbol{L} + \boldsymbol{D})\boldsymbol{x}^{k+1} = \boldsymbol{b} - \boldsymbol{U}\boldsymbol{x}^k,$$

respectively [15].

For a result on the necessary conditions for convergence regardless of the initial guess see [15, 25]. In terms of circuit topology, sufficient conditions for convergence are met if there is a capacitor to ground at each node, an assumption that is reasonable in VLSI designs. For an equivalent result in the case of Nonlinear Relaxation see [8, 25].

The point Gauss-Jacobi Nonlinear Relaxation method applied at the nonlinear equation (2.6) is shown in Algorithm 2.4.2. The superscript k represents the iteration counter, the subscript i represents the node index in the vector of voltages, and ϵ is a small positive number. The function $F_i(\cdot)$ is the i^{th} component of equation (2.6), that is

$$F_i\left(\boldsymbol{v}_{(i)}^k(t+h)\right) = p_i\left(\boldsymbol{v}_{(i)}^k(t+h)\right) - p_i\left(\boldsymbol{v}(t)\right) + \frac{h}{2}\left[f_i\left(\boldsymbol{v}_{(i)}^k(t+h)\right) + f_i\left(\boldsymbol{v}(t)\right)\right] = \mathbf{0}$$

Algorithm 2.4.2 (Nonlinear Gauss-Jacobi Relaxation).

```
nonlinear_Gauss_Jacobi_relaxation()
{
    $t = 0$;
    while (t < Simulation_Time) {
        select $h$ ;
        $k = 0$;
        guess $\boldsymbol{v}^0$
        repeat {
            $k = k + 1$;
            for ($i = 1$; $1 \leq n$; $i++$) {
                solve $F_i\left(v_1^{k-1}, \ldots, v_{i-1}^{k-1}, v_i^k, v_{i+1}^{k-1}, \ldots, v_n^{k-1}\right) = 0$ for $v_i^k$,
                    given $v_j^{k-1}$ $j \neq i$
            }
        } until $\left(\left\| v_i^k - v_i^{k-1} \right\|_1 \leq \epsilon\right)$
        update waveforms ;
        $t = t + h$;
    }
}
```

using the notation

$$\boldsymbol{v}_{(i)}^k = \left(v_1^{k-1}, \ldots, v_{i-1}^{k-1}, v_i^k, v_{i+1}^{k-1}, \ldots, v_n^{k-1}\right)$$

where, v_i^k is the i^{th} component of $\boldsymbol{v}(t+h)$ at the k^{th} relaxation iteration, and dependence on $\boldsymbol{u}(\cdot)$ was dropped for clarity.

It is possible to use the Newton-Raphson algorithm to solve the algebraic equation

$$F_i\left(\boldsymbol{v}_{(i)}^k\right) = F_i\left(v_1^{k-1}, \ldots, v_{i-1}^{k-1}, v_i^k, v_{i+1}^{k-1}, \ldots, v_n^{k-1}\right) = 0$$

accurately at each step, but this is not necessary. In fact it has been proved that only one iteration of the Newton-Raphson method is required to maintain the convergence of the relaxation iteration due to the quadratic convergence of the Newton-Raphson method [21]. After the relaxation iteration has converged, $\boldsymbol{v}^k$ becomes the new voltage vector at time $t + h$, and one advances to the next timepoint.

2.5 Device Evaluation Issues

As mentioned in chapter 1, function or device evaluation is one of the bottlenecks in standard electrical circuit simulation. Evaluation of a device's constitutive relation is necessary for the computation of KCL or KVL. If the device has a nonlinear characteristic, then it will also be necessary to compute its derivatives in order to obtain the jacobian entries for the underlying Newton method. Even if the device is linear, computing its characteristic can be a computationally expensive task. Furthermore, these computations have to be carried out at every Newton iteration for every timestep simulated. In some cases, however, it is possible to bypass some the computations of certain elements at certain times, thus saving computation time [19, 20].

Obtaining accurate models that describe the behavior of nonlinear devices is in itself a difficult problem. The models must be accurate and general enough for application in a wide variety of situation. Traditionally, modeling of semi-conductor devices has received much attention, and current simulators such as SPICE contain fairly accurate and general-purpose device models for almost every conceivable device type that one would find in traditional discrete system or VLSI circuits. In recent years, as a result of increasing signal and clock speeds, interconnections are rapidly becoming a dominant factor in electronic circuit and system performance. It is now widely accepted that for accurate circuit simulation the combined electrical and magnetic effects of the packaging and interconnect and the nonlinear drivers and loads must be simulated during some stage of the verification process. However, accurate modeling of packaging and interconnects results in general in large, linear circuit models that can dramatically slow down a circuit simulation [28, 29]. Considerable research has therefore been dedicated to the study of efficient and accurate simulation techniques to analyze circuits dominated by packaging and interconnect effects. The fundamental difficulty encountered in integrating interconnect and packaging effects into a transient circuit simulator has to do with the fact that circuits containing nonlinear devices or time-dependent characteristics must be characterized in the time domain, while most of the important effects due to interconnect and packaging are best characterized in the frequency domain. System-level descriptions of interconnect have been introduced in recent years [30, 31, 32] and incorporated into popular circuit simulators [33, 34, 35].

To understand how such models can be incorporated into a circuit simulator, consider for example the case of a transmission line with skin effect which may be used to model interconnect effects at the chip level as well as at the package, multi-chip module, board and backplane level. Two systematic approaches have been succesfully used to incorporate transmission line models into circuit simulators. The first one is the analytical approach. SPICE3, for instance has an analytical model of a lossless as well as a lossy transmission line [19, 33]. While this analytical model is quite accurate, it is not very general. For instance, the model used does not consider skin-effects or any other frequency dependent characteristic whose effects the designer might wish to verify. The transmission-line equations are hard-coded into the simulator and even though it is possible to modify the characteristics of the line by appropriately chosing the values of some of the model parameters, it is impossible to go beyond the changes allowed by those parameters.

Another approach, pioneered in [34, 35] is based on directly incorporating a frequency dependent model into a simulator. In general, a transmission line can be described in the frequency domain using scattering parameters, in which case

$$\begin{bmatrix} \boldsymbol{Y}_o(j\omega)\boldsymbol{V}_a(j\omega) + \boldsymbol{I}_a(j\omega) \\ \boldsymbol{Y}_o(j\omega)\boldsymbol{V}_b(j\omega) + \boldsymbol{I}_b(j\omega) \end{bmatrix} = \begin{bmatrix} 0 & \boldsymbol{S}_{12}(j\omega) \\ \boldsymbol{S}_{12}(j\omega) & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{Y}_o(j\omega)\boldsymbol{V}_a(j\omega) - \boldsymbol{I}_a(j\omega) \\ \boldsymbol{Y}_o(j\omega)\boldsymbol{V}_b(j\omega) - \boldsymbol{I}_b(j\omega) \end{bmatrix} \tag{2.12}$$

where $\boldsymbol{V}_a(j\omega), \boldsymbol{I}_a(j\omega)$ and $\boldsymbol{V}_b(j\omega), \boldsymbol{I}_b(j\omega)$ are the voltages and currents at terminals a and b of the transmission line, $\boldsymbol{Y}_o(j\omega)$ is its characteristic admittance, and $\boldsymbol{S}_{12}(j\omega)$ is the relation between the incident and reflected waves on opposite ends of the transmission line. Note, the nonstandard choice of $\boldsymbol{Y}_o(j\omega)$ instead of $\boldsymbol{Z}_o(j\omega) = 1/\boldsymbol{Y}_o(j\omega)$ is that for a line with no shunt loss, $\boldsymbol{Z}_o(0) = \infty$, which may cause numerical difficulties in many situations. It should be noted that any ideal delay resulting from propagation along the transmission line and which reflects itself on $\boldsymbol{S}_{12}(j\omega)$ or $(\boldsymbol{Y}_o\boldsymbol{S}_{12})(j\omega)$ is usually handled separately and cancelled from the above frequency dependent measurements or model before they are incorporated into the simulator. This is in general easily accomplished by multiplying by the associated exponentials [31, 36].

To incorporate such a general transmission line representation into a circuit simulator, it is necessary to compute the inverse transforms of $\boldsymbol{S}_{12}(j\omega)$, $\boldsymbol{Y}_o(j\omega)$, and $(\boldsymbol{Y}_o\boldsymbol{S}_{12})(j\omega)$ so as to determine the impulse responses $s_{12}(t), y_o(t)$, and $(y_o s_{12})(t)$. Then (2.12) becomes

$$\begin{aligned} (y_o \star v_a)(t) + i_a(t) &= ((y_o s_{12}) \star v_b)(t - t_d) - (s_{12} \star i_b)(t - t_d) \\ (y_o \star v_b)(t) + i_b(t) &= ((y_o s_{12}) \star v_a)(t - t_d) - (s_{12} \star i_a)(t - t_d) \end{aligned}$$

where $\star$ is used to denote convolution and t_d is the propagation delay which was extracted from the frequency dependent model and is now explicitely introduced into the time-domain equations. If for instance frequency data is available either as measured or tabulated data then we can derive $s_{12}(t), y_o(t)$ and$(y_o s_{12})(t)$ by applying the inverse Fast Fourier Transform to $\boldsymbol{S}_{12}(j\omega)$, $\boldsymbol{Y}_o(j\omega)$, and $(\boldsymbol{Y}_o\boldsymbol{S}_{12})(j\omega)$. Once the impulse response is known, then the circuit equation can be solved at any timepoint t by numerically computing the necessary convolution integrals,which involves an impulse response and some voltage or current waveform.

The ability to incorporate complex models such as the one described makes a simulator more general and therefore increases its usefulness.

References

[1] C. A. Desoer and E. S. Kuh, *Basic Circuit Theory.* New York: McGraw-Hill, 1969.

[2] L. O. Chua, C. A. Desoer, and E. S. Kuh, *Linear and Nonlinear Circuits.* Circuits and Systems, New York: McGraw-Hill, 1987.

[3] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design.* Berkshire, England: Van Nostrand Reinhold, 1983.

[4] C. W. Ho, A. E. Ruehli, and P. A. Brennan, "The modified nodal approach to network analysis," *IEEE Transactions on Circuits and Systems*, vol. CAS-22, pp. 504–509, June 1975.

[5] G. Dahlquist and Å. Björck, *Numerical Methods.* Automatic Computation, Englewood Cliffs, New Jersey: Prentice-Hall, 1974.

[6] L. N. Trefethen, "Finite-difference and spectral methods." Lecture notes, Massachusetts Institute of Technology. Unpublished, 1989.

[7] C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations.* Automatic Computation, Englewood Cliffs, New Jersey: Prentice-Hall, 1971.

[8] J. M. Ortega and W. C. Rheinbolt, *Iterative Solution of Nonlinear Equations in Several Variables.* Computer Science and Applied Mathematics, New York: Academic Press, 1970.

[9] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, "Inexact Newton methods," *SIAM J. Numer. Anal.*, vol. 19, pp. 400–408, April 1982.

[10] C. W. Gear and Y. Saad, "Iterative solution of linear equations in ODE codes," *SIAM J. Sci. Statist. Comput.*, vol. 4, pp. 583–601, December 1983.

[11] P. N. Brown and A. C. Hindmarsh, "Matrix-free methods for stiff systems of ODE's," *SIAM J. Numer. Anal.*, vol. 23, pp. 610–638, June 1986.

[12] P. N. Brown and A. C. Hindmarsh, "Reduced storage methods in stiff ODE systems," *J. Appl. Math. Comput.*, vol. 31, pp. 40–91, 1989.

[13] P. Brown and Y. Saad, "Hybrid Krylov methods for nonlinear systems of equations," *SIAM J. Sci. Statist. Comput.*, vol. 11, pp. 450–481, May 1990.

[14] G. Strang, *Linear Algebra and Its Applications.* New York: Academic Press, 1980.

[15] G. H. Golub and C. F. Van Loan, *Matrix Computations.* Baltimore, Maryland: The John Hopkins University Press, 1983.

[16] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices.* Oxford: Clarendon Press, 1986.

[17] V. Strassen, "Gaussian elimination is not optimal," *Numer. Math.*, vol. 13, pp. 354–356, 1968.

[18] K. S. Kundert, "Sparse matrix techniques," in *Circuit Analysis, Simulation and Design* (A. E. Ruehli, ed.), pp. 281–324, North-Holland, 1986.

[19] T. L. Quarles, "The SPICE3 Implementation Guide," Tech. Rep. ERL M89/44, Electronics Research Laboratory Report, University of California at Berkeley, Berkeley, California, April 1989.

[20] A. L. Sangiovanni-Vincentelli, "Circuit simulation," in *Computer Design Aids for VLSI Circuits* (P. Antognetti, D. O. Pederson, and H. de Man, eds.), NATO ASI Series, pp. 19–112, Dordrecht, Germany: Martinus Nijhoff Publishers, 1986.

[21] A. R. Newton and A. L. Sangiovanni-Vincentelli, "Relaxation - Based circuit simulation," *IEEE Transactions on Electron Devices*, vol. ED-30, pp. 1184–1207, September 1983.

[22] L. F. Richardson, "The approximate arithmetical solution by finite differences of physical problems involving differential equations,with applications to the stress in a masonry dam," *Philos. Trans. Roy. Soc. London*, vol. Ser. A210, pp. 307–357, 1910.

[23] R. S. Varga, *Matrix Iterative Analysis.* Automatic Computation Series, Englewood Cliffs, New Jersey: Prentice-Hall Inc, 1962.

[24] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli, "The waveform relaxation method for time domain analysis of large scale integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 1, pp. 131–145, July 1982.

[25] J. K. White and A. Sangiovanni-Vincentelli, *Relaxation Techniques for the Simulation of VLSI Circuits.* Engineering and Computer Science Series, Norwell, Massachusetts: Kluwer Academic Publishers, 1986.

[26] A. R. Newton and S. L. Sangiovanni-Vincentelli, "Relaxation-based electrical simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 3, March 1984.

[27] D. Dumlugol, *The Segmented Waveform Relaxation Method for Mixed-Mode Simulation of Digital MOS Circuits.* PhD thesis, Katholieke Universiteit Leuven, October 1986.

[28] A. E. Ruehli and P. A. Brennan, "Efficient Capacitance Calculations for Three-Dimensional Multiconductor Systems," *IEEE Transactions on Microwave Theory and Techniques*, vol. 21, pp. 76–82, February 1973.

[29] A. E. Ruehli, "Survey of Computer-Aided Electrical Analysis of Integrated Circuit Interconnections," *IBM Journal of Research and Development*, vol. 23, pp. 626–639, November 1979.

[30] J. S. Roychowdhury, A. R. Newton, and D. O. Pederson, "An Impulse-Response based Linear Time-Complexity Algorithm for Lossy Interconnect Simulation," in *International Conference on Computer Aided-Design*, pp. 62–65, November 1991.

[31] J. E. Bracken, V. Raghavan, and R. A. Rohrer, "Interconnect Simulation with Asymptotic Waveform Evaluation," *IEEE Trans. Circuits Syst.*, vol. 39, pp. 869–878, November 1992.

[32] J. R. Griffith and M. S. Nakhla, "Time-Domain Analysis of Lossy Coupled Transmission Lines," *IEEE Transactions on Microwave Theory and Techniques*, vol. 38, pp. 1480–1487, October 1990.

[33] J. S. Roychowdhury and D. O. Pederson, "Efficient Transient Simulation of Lossy Interconnect," in 28^{th} *ACM/IEEE Design Automation Conference*, pp. 740–745, June 1991.

[34] V. Raghavan, J. E. Bracken, and R. A. Rohrer, "AWESpice: A General Tool for the Accurate and Efficient Simulation of Interconnect Problems," in 29^{th} *ACM/IEEE Design Automation Conference*, (Anaheim, California), pp. 87–92, June 1992.

[35] T. K. Tang and M. S. Nakhla, "Analysis of High-Speed VLSI Interconnects using the Asymptotic Waveform Evaluation Technique," in *International Conference on Computer Aided-Design*, (Santa Clara, California), pp. 542–544, November 1990.

[36] S. Lin and E. S. Kuh, "Transient Simulation of Lossy Interconnects Based on the Recursive Convolution Formulation," *IEEE Trans. Circuits Syst.*, vol. 39, pp. 879–892, November 1992.

3

Clocked Analog Circuit Simulation

3.1 Introduction

In general, clocked analog circuit designers rely heavily on circuit simulation programs like SPICE [1] or ASTAP [2] to verify the correctness and to determine the performance of their designs. As we saw in chapter 2, these programs simulate a circuit by first constructing the system of differential equations that describes the circuit, and then solving that system numerically with a time discretization method such as backward-Euler. When applied to clocked analog circuits such as switching power converters, switched-capacitor circuits or phase-locked loops, such classical circuit simulation algorithms become extraordinarily computationally expensive. This is because these circuits use high-frequency clocks whose periods are typically orders of magnitude smaller than the time intervals of interest to a designer. The nature of the calculations used in a circuit simulator implies that to construct the solution over the time interval of interest, an accurate solution must be computed for every cycle of the high-frequency clock in the interval and therefore the circuit simulation timesteps are constrained to be much smaller than a clock period. This implies that to compute the solution over the time interval of interest to a designer may require hundreds of thousands of timepoints.

The infeasibility of simulating such circuits with classical techniques has led designers to explore a variety of simulation alternatives, approaching each problem differently and producing solution methods that are usually tuned to specific problems and can only be used for a specific class of problems. For example, to simulate switching power converters the alternatives exploited include the development of specialized dynamically reconfigurable analog computers [3]. This approach has the usual drawbacks associated with breadboarding, that is parasitics are not well controlled, arbitrary nonlinearities are hard to add, and performance sensitivities can not be easily examined. More popular are fast approximate simulation techniques, based on assuming some ideal characteristic of the switching, such that shortly after the beginning of each clock cycle, the equilibrium point is achieved. A system of difference equations that represent the switching power converter as changing from one equilibrium point to the next is then constructed and solved. This approach treats the switching converter's switches as ideal, and the remaining circuitry as linear [4, 5]. In addition, it is sometimes possible to further simplify the converter circuit by eliminating certain state variables that do not contribute significantly to the output of interest [6]. Approximate techniques such as these can reduce the cost of computing the behavior of a switching converter circuit over one high-frequency clock cycle to the point where it becomes computationally feasible to simulate the circuit for the hundreds of cycles needed to construct a complete transient.

The most common approach to simulating switched-capacitor filters is to break the circuit up into functional blocks such as operational amplifiers and switches. Each functional bock is simulated, using a traditional circuit simulator, for some short period. The simulations of the functional blocks are used to construct extremely simple macromodels, which replace the functional blocks in the circuit. The result is a much simplified circuit that can be simulated easily. It is then assumed that after each clock transition, every node in the circuit reaches its equilibrium point before another transition occurs. This assumption, known as the "slow-clock" approximation, along with the use of algebraic macromodels, allows the filter to be treated as a discrete-time system with one time point per clock transition. A set of difference equations is then used to describe the filter, whose solution for hundreds of clock cycles can be computed quickly. Simulators of this macromodeling sort have been formalized in programs like DIANA [7] and SWITCAP [8]. Although these programs have served designers well, a macromodeling approach is only as good as the macromodel. If a second order effect in a functional block changes overall performance, but this effect is not included in the macromodel, the effect will never be seen in the simulation.

Although programs based on the above techniques are reasonably efficient, they are based on idealizations of the circuits involved which may eliminate behavior that is important to a designer. Another approach to simulating clocked analog circuits which does not involve any idealization of the behavior is the method referred to as **Envelope-Following** [9]. This method exploits the fact that a designer typically is not interested in the details of the node voltage behavior in every clock cycle, but rather is interested in the "envelope" of that behavior. The method also exploits the property of such circuits that the node voltage waveforms over a given high frequency clock cycle are similar, but not exact duplicates, of the node voltages waveforms in proceeding or following cycles. This suggests that it is possible to construct a solution accurate over many high frequency clock cycles by calculating the solution accurately for a few selected cycles. The envelope-following method can be therefore considered as an order reduction technique in itself, given that it provides to the designer the ability to obtain accurate information regarding the "enveloping" behavior of the node voltages while hiding from him/her the unnecessary details of the voltage waveforms evolution. The envelope-following approach, has been found to be extremely efficient for the simulating the power-up transient of open-loop switching power-converters [10]. However the method was shown to perform poorly for closed-loop switching power converters [11].

In the next section the Envelope-Following algorithm is introduced. We will make the definition of envelope more precise and derive a simple method for computing envelopes which involves solving a sequence of two-point boundary value problems. Stability issues regarding the formulation of these equations are studied in section 3.2.2. The two-point boundary value problems are solved with a "shooting" or Newton method, as described in section 3.2.3. The computations involved are explained in detail in section 3.2.4. In section 3.3 we will show that the standard algorithm has problems dealing with circuits for which there are states in the system which change rapidly due to small changes in other slowly changing states. Given that, as mentioned, the designer is not in general interested in these fast transients but is in fact more concerned with those slower characteristics of the circuit, it would seem reasonable to obtain a reduced-order model of the circuit that would encapsulate the important enveloping behavior of the response, without need to consider the faster transients. In section 3.4.1 we start by showing how it is possible to modify the standard envelope-following algorithm to circumvent the above problems. In the remainder of section 3.4 we will outline the proposed solution to the problems described and and present computational results we have obtained with this modified technique. In section 3.5 we discuss our experience and ideas regarding issues related to extending the envelope-following techniques described to other types of circuits, including autonomous circuit and circuits containing multiple frequencies. Finally, conclusions and suggestions for further work are contained in section 3.6.

3.2 The Envelope-Following Method

The type of clocked analog circuits with which we are concerned have the property that the circuit to be simulated has as an input clock with period, T, that is much smaller than the simulation interval. For example, consider the simplified "buck" DC-DC

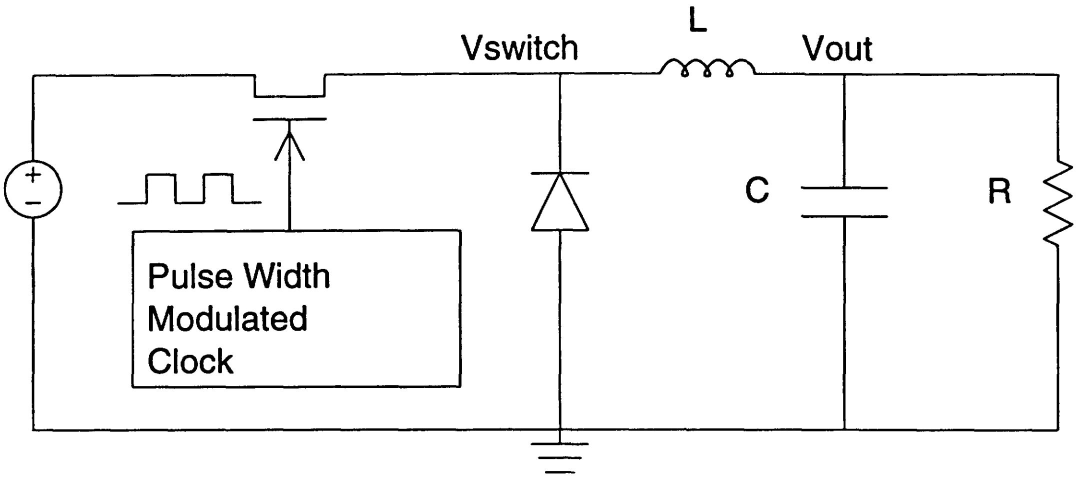

Figure 3-1: Open-loop buck converter.

converter circuit in Fig. 3-1 [4]. This circuit's behavior in steady-state is roughly that of a modulator followed by a low-pass filter. The modulator converts the input DC source into a periodic pulse waveform and the low-pass filter time-averages the pulse waveform to produce a DC voltage at the output. In the circuit in Fig. 3-1, the N-channel MOS transistor combined with a diode act as the modulator, and are controlled by the input clock connected to the MOS transistor's gate. The DC output voltage of the converter is given approximately by DV_{in}, where D is the duty-cycle of the input clock waveform and V_{in} is the input voltage.

The voltage waveforms in steady state for the switch and output nodes of the "buck"-converter of Fig. 3-1 were computed numerically using a standard backward-Euler integration scheme; the computed timepoints for a portion of the simulation interval are plotted in Fig. 3-2 (for the simulation, the DC input was 10 volts, the clock was a 100 kHz square wave and $R = 140\Omega$, $L = 420\mu h$ and $C = 38\mu f$). The backward-Euler integration scheme used for this simulation required more than twenty timepoints for each simulated clock cycle, because of the rapid variation of the voltage at the switch node. This implies that simulating a power converter transient, which can span hundreds of clock cycles because of the low-pass filtering, will be computationally very expensive. For example, the plot in Fig. 3-3 is of the output voltage waveform for the power-up transient of the buck converter in Fig. 3-1. In this case the power-up transient waveforms is made up of more than 1000 cycles of the input clock, and the total simulation used about 55,000 timepoints. The reasons for this large amount of computations lie mainly with the fact that the input clock period T is much smaller than the simulation interval.

As mentioned before, the number of timepoints computed during a switching converter

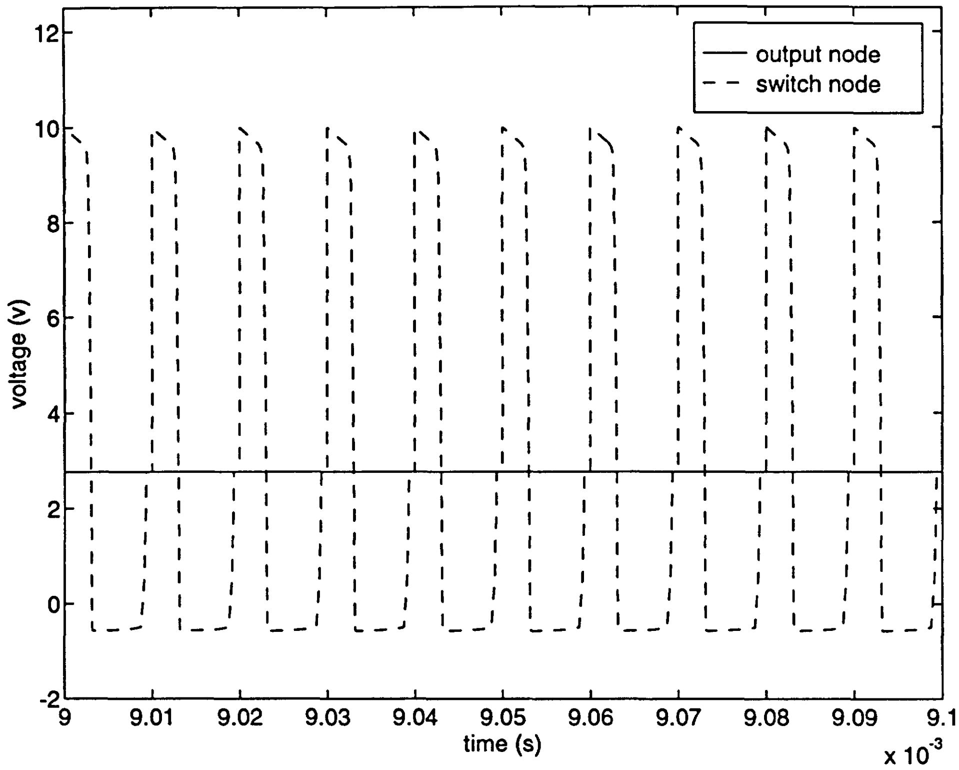

Figure 3-2: Buck converter switch and output nodes in steady-state.

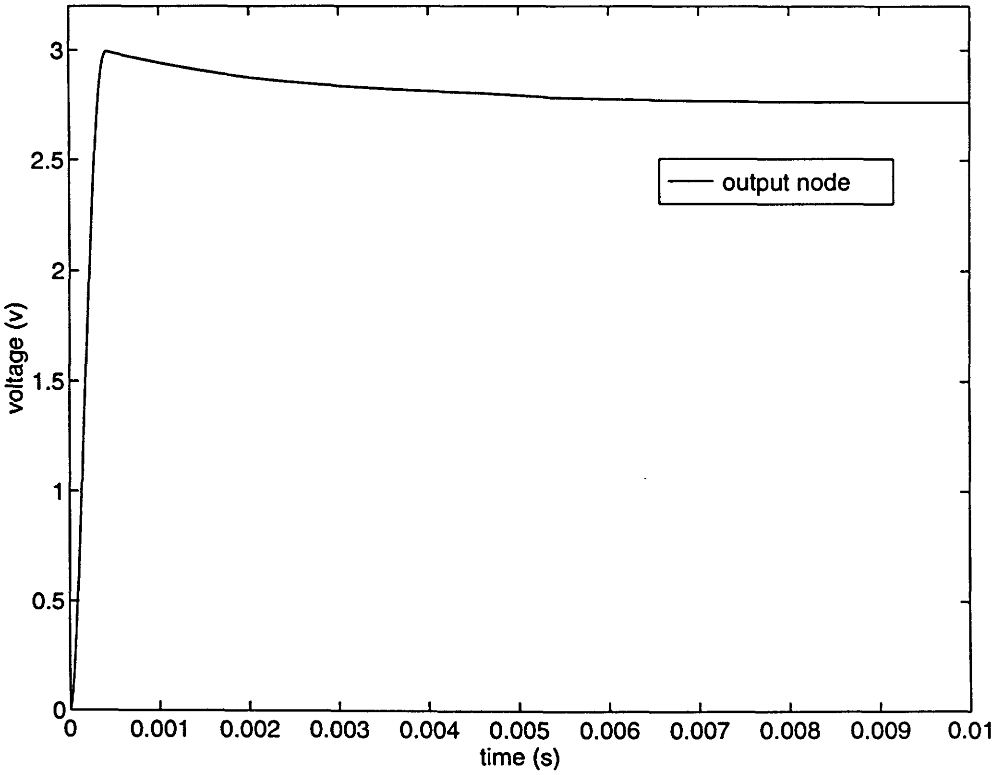

Figure 3-3: Buck converter output voltage during power-up transient.

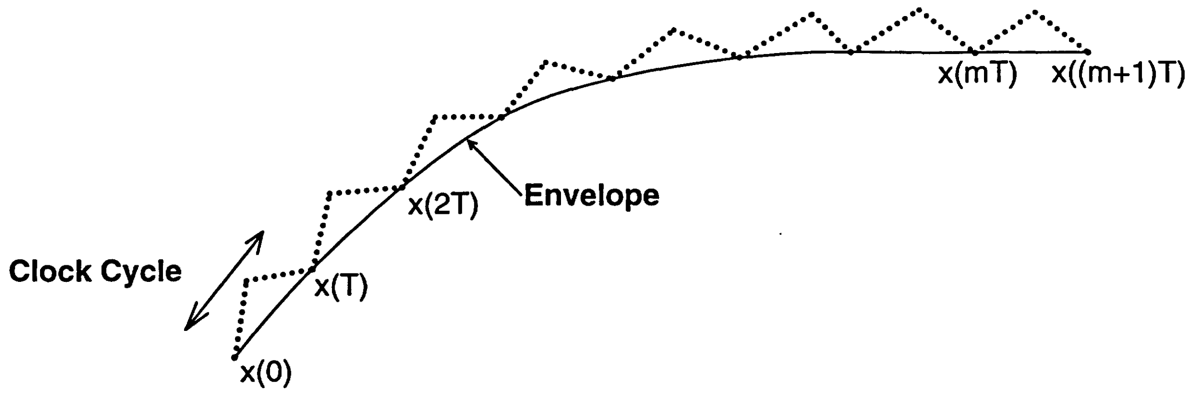

Figure 3-4: Envelope definition.

transient simulation can be reduced by exploiting the fact that a designer typically is not interested in the details of the node voltage behavior in every clock cycle, but rather is interested in the "envelope" of that behavior. Specifically, we define the "envelope" to be a continuous function derived by interpolating the sequence formed by sampling the state every clock period T (See Fig. 3-4). Note our use of "envelope" is not standard. Here, the envelope is not unique given $\boldsymbol{x}(t)$; the envelope generated by interpolating the sequence $\boldsymbol{x}(0+\tau), \boldsymbol{x}(T+\tau), \boldsymbol{x}(2T+\tau), \ldots$ depends on τ. The key advantage of considering just the envelope is that if the sequence formed by sampling the state at the beginning of each clock cycle, $\boldsymbol{x}(0), \boldsymbol{x}(T), \boldsymbol{x}(2T), \ldots, \boldsymbol{x}(mT), \ldots$, changes slowly as a function of m, the clock cycle number, then the envelope can be accurately computed by detailed simulation of only every l^{th} clock cycle, where l, referred to as the cycle-step, is large.

3.2.1 Computing the Envelope

As described in Chapter 2, most circuits can be described by a system of differential equations of the form

$$\frac{d}{dt}\boldsymbol{p}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) + \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) = 0, \tag{3.1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^N$, the state, is the vector of capacitor voltages and inductor currents, $\boldsymbol{u}(t) \in \mathbb{R}^M$ is the vector of input sources, $\boldsymbol{p}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) \in \mathbb{R}^N$ is the vector of capacitor charges and inductor fluxes, and $\boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) \in \mathbb{R}^N$ is the vector of resistive currents and inductor voltages.

If the state $\boldsymbol{x}$ is known at some time t_0, it is possible to solve Eqn. (3.1) and compute the state at some later time t_1. In general, one can write

$$\boldsymbol{x}(t_1) = \boldsymbol{\phi}(\boldsymbol{x}(t_0), t_0, t_1) \tag{3.2}$$

where $\phi : \mathbb{R}^N \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^N$ is a state transition function for the differential equation. However, if $t_1 \gg t_0$ this may be computationally very expensive. Consider an alternate situation where the state is know not only at a single time but at a collection of timepoints $0, T, 2T, \ldots, mT, \ldots$. In this case, the value of the state at any other timepoint $\boldsymbol{x}(t_s)$, can again by obtained by solving Eqn. (3.1). However, since we can now determine m such that $mT < t_s < (m+1)T$, we can use $\boldsymbol{v}_(mT)$ as an initial condition to the differential equation. This in turn implies that to obtain the state at any timepoint it is enough to integrate the differential equation for an interval no longer than T. In other words, the knowledge of the state at the given set of timepoints, reduces the problem of computation of the state at any timepoint to that of an integration in an interval no longer than T, which has obvious computational advantages. Then question we now address is how to compute the state at that particular set of timepoints, or in other words, how to compute the envelope of the state $\boldsymbol{x}$.

The straight-forward approach to computing the envelope of the solution to Eqn. (3.1) is to numerically compute $\boldsymbol{x}(t)$ for all t and then to sample this computed solution at $\boldsymbol{x}(0)$, $\boldsymbol{x}(T)$, $\boldsymbol{x}(2T), \ldots$ to construct the envelope. If the envelope is *smooth enough*, then it will be possible to approximately represent an interval of sample points, $\boldsymbol{x}\left((m-1)T\right), \boldsymbol{x}(mT)$, $\ldots, \boldsymbol{x}\left((m+l)T\right)$ with a low order polynomial in the cycle number. For example, if over $l+1$ cycles the envelope is well approximated by a straight line, then

$$\boldsymbol{x}\left((m+l)T\right) - \boldsymbol{x}\left(mT\right) \approx l \; \left[\boldsymbol{x}\left((m+1)T\right) - \boldsymbol{x}\left(mT\right)\right]. \tag{3.3}$$

The term $[\boldsymbol{x}(mT) - \boldsymbol{x}\left((m-1)T\right)]$ can be thought of, imprecisely, as the derivative of the envelope at $\boldsymbol{x}(mT)$, in which case Eqn. (3.3) is loosely analogous to solving a differential equation by forward-Euler. Following that analogy, l is then the *cycle-step* for the integration method. Graphically this approximation corresponds to aligning the values of the state at the beginning and end of a cycle and then extrapolating l cycles ahead to obtain the new starting point of a cycle. This is shown in Fig. 3-5.

To compute the envelope of a system with period T using a fixed cycle-step version of the above forward-Euler style envelope-following algorithm, a simple repetitive two-step process can be used. The first step is to compute $\boldsymbol{x}(T)$, given $\boldsymbol{x}(0)$, by solving Eqn. (3.1) over the interval $[0, T]$, as shown graphically in Fig. 3-6. Then the second step is to set $\boldsymbol{x}\left((1+l)T\right) = \boldsymbol{x}(T) + l\left[\boldsymbol{x}(T) - \boldsymbol{x}(0)\right]$. This process can be repeated to compute $\boldsymbol{x}\left((2+2l)T\right), \boldsymbol{x}\left((3+3l)T\right)$, et cetera. Note that calculating the envelope over a long interval then requires solving Eqn. (3.1) for one out of every l cycles.

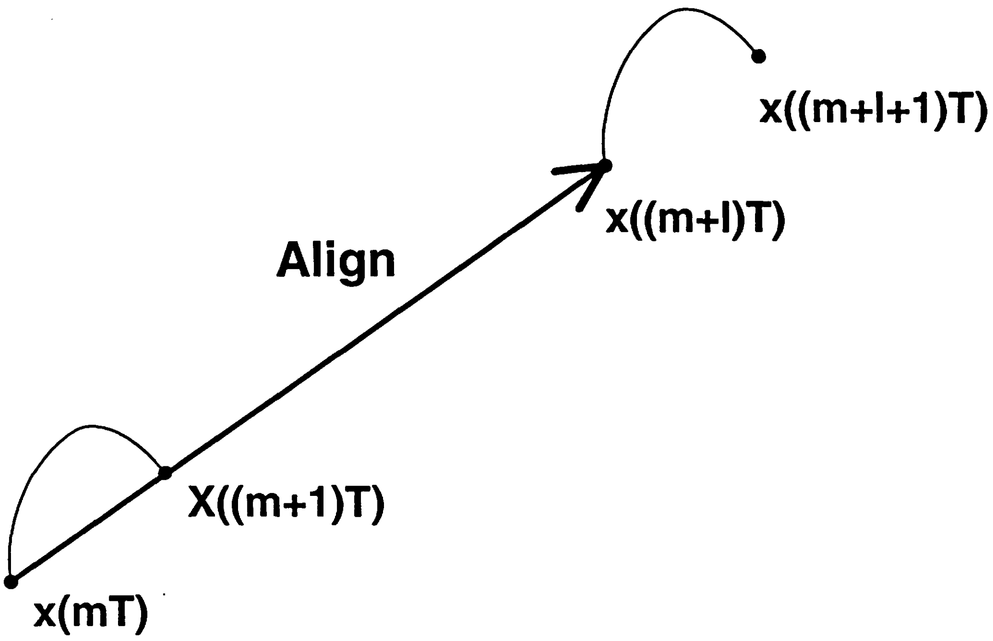

Figure 3-5: Forward-Euler -like extrapolation for the Envelope-Following algorithm indicating alignment necessary to obtain starting values at the beginning of a cycle l cycles away.

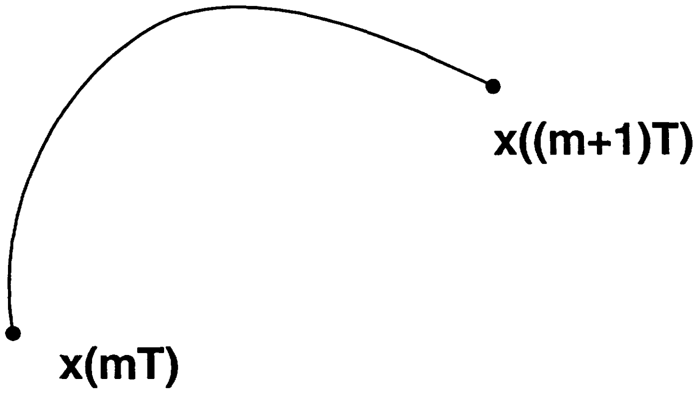

Figure 3-6: Integrating the circuit equations for one high-frequency cycle takes us from the beginning point of a cycle to the endpoint of the same cycle.

3.2.2 Stability Issues in Envelope-Following Formulation

Although simple to describe, a forward-Euler based approach to computing envelopes is inefficient for simulating switching converter circuits. Maintaining stability severely limits the size of the cycle-step l, just as in the standard forward-Euler algorithm. To see that this is indeed the case we analyze a simple linear system of the form

$$\dot{\boldsymbol{x}}= \boldsymbol{A}\boldsymbol{x}. \tag{3.4}$$

for which the state transition function is readily obtained as

$$\phi\left(\boldsymbol{x}\left(mT\right), mT, (m+1)T\right) = e^{\boldsymbol{A}T}\boldsymbol{x}\left(mT\right). \tag{3.5}$$

Applying the forward-Euler style envelope-following algorithm in Eqn. (3.3) to Eqn. (3.4) leads to

$$\boldsymbol{x}\left((m+l)T\right) - \boldsymbol{x}\left(mT\right) = l\ \left[e^{\boldsymbol{A}T}\boldsymbol{x}(mT) - \boldsymbol{x}(mT)\right].$$

With some algebraic manipulation one obtains

$$\boldsymbol{x}\left((m+l)T\right) = \left[\boldsymbol{I} + l\ \left(e^{\boldsymbol{A}T} - \boldsymbol{I}\right)\right] \boldsymbol{x}(mT). \tag{3.6}$$

If we now consider the repeated application of Eqn (3.6) using a fixed cycle-step of l we obtain

$$\boldsymbol{x}\left((m+nl)T\right) = \left[\boldsymbol{I} + l\ \left(e^{\boldsymbol{A}T} - \boldsymbol{I}\right)\right]^n \boldsymbol{x}(mT). \tag{3.7}$$

The forward-Euler Envelope-Following algorithm will be considered stable if the solutions to Eqn. (3.7) remain bounded as n goes to infinity. In this case, this implies that the spectral radius of the iteration matrix $\boldsymbol{M} = \boldsymbol{I} + l\ \left(e^{\boldsymbol{A}T} - \boldsymbol{I}\right)$ is smaller than one, that is

$$\rho(\boldsymbol{M}) = \rho\left(\boldsymbol{I} + l\ \left(e^{\boldsymbol{A}T} - \boldsymbol{I}\right)\right) < 1.$$

If for simplicity $\boldsymbol{A}$ is assumed to be diagonalizable, i.e. it has a full set of linearly independent eigenvalues it is possible to perform a spectral decomposition on $\boldsymbol{A}$. One obtains $\boldsymbol{A} = \boldsymbol{S}^{-1}\boldsymbol{\Lambda}\boldsymbol{S}$, where $\boldsymbol{S}$ is a matrix whose columns are the eigenvectors of $\boldsymbol{A}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix whose elements are the eigenvalues of $\boldsymbol{A}$, that is $\boldsymbol{\Lambda}_{ii} = \lambda_i\left(\boldsymbol{A}\right)$. Furthermore, from the definition of matrix exponential

$$e^{\boldsymbol{A}T} = e^{\boldsymbol{S}^{-1}\boldsymbol{\Lambda}\boldsymbol{S}T} = \boldsymbol{S}^{-1}e^{\boldsymbol{\Lambda}T}\boldsymbol{S}.$$

Therefore

$$\rho\left(\boldsymbol{M}\right) = \max_{\lambda_i}\left|1 + l\ \left(e^{\lambda_i T} - 1\right)\right| < 1. \tag{3.8}$$

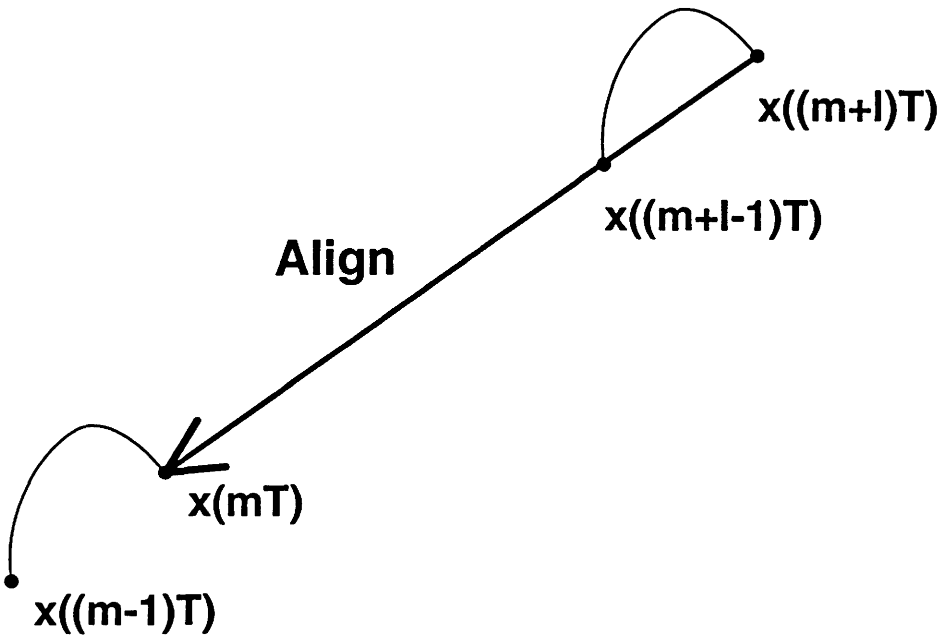

Figure 3-7: Backward-Euler -like extrapolation for the Envelope-Following algorithm indicating alignment necessary to obtain starting values at the beginning of a cycle l cycles away.

If we assume that the system in Eqn. (3.4) is stable, thus implying that all eigenvalues of $\boldsymbol{A}$ are in the left half complex, that is $Real[\lambda_i] < 0$, $\forall i$ then $e^{\lambda_i T} < 1$ which in turn implies that

$$|1 + l\ (e^{\lambda_i T} - 1)| = |1 - l\ \underbrace{(1 - e^{\lambda_i T})}_{>0}|.$$

For stability we must therefore ensure that $\forall i$, $|1 - l\ (1 - e^{\lambda_i T})| < 1$ or equivalently $1 - l\ (1 - e^{\lambda_i T}) > -1$ which leads to

$$l < \frac{2}{1 - e^{\lambda_{min} T}} \tag{3.9}$$

This overly limiting condition implies that if a forward-Euler style envelope-following algorithm is used, in many cases it will be impossible to use large cycle-steps, which means that the efficiency increase with this algorithm will be minimal if any at all.

A more stable algorithm is to approximate the value of $\boldsymbol{x}\,((m+l)T)$ by

$$\boldsymbol{x}\,((m+l)T) - \boldsymbol{x}(mT) \approx l\ \left[\boldsymbol{x}\,((m+l)T) - \boldsymbol{x}\,((m+l-1)T)\right] \tag{3.10}$$

which is analogous to the backward-Euler algorithm. Graphically this corresponds to aligning the end-point of a cycle with the beginning and end point of a cycle l cycles into the future, as shown in Fig. 3-7.

This approach allows for larger cycle-steps than the forward-Euler based approach, just as in the case of numerical integration. To see that this is indeed the case, we apply this formula to our test example from Eqn. (3.4). Recalling the semigroup property of the state-transition function [12] which establishes that

$$\boldsymbol{\phi}\left(\boldsymbol{x}(t_1), t_1, t_2\right) = \boldsymbol{\phi}\left(\boldsymbol{x}(t_1), t_1, t_3\right) = \boldsymbol{\phi}\left(\boldsymbol{x}(t_3), t_3, t_2\right)$$

a special case of which is

$$\boldsymbol{\phi}^{-1}\left(\boldsymbol{x}(t_1), t_1, t_2\right) = \boldsymbol{\phi}\left(\boldsymbol{x}(t_2), t_2, t_1\right)$$

and noting that

$$\boldsymbol{x}\left((m+l-1)T\right) = \boldsymbol{\phi}^{-1}\left(\boldsymbol{x}\left((m+l)T\right), (m+l)T, (m+l-1)T\right) = e^{-\boldsymbol{A}T}\boldsymbol{x}\left((m+l)T\right),$$

we can rewrite Eqn.(3.10) as follows

$$\boldsymbol{x}\left((m+l)T\right) - \boldsymbol{x}(mT) = l\ \left[\boldsymbol{x}\left((m+l)T\right) - e^{-\boldsymbol{A}T}\boldsymbol{x}\left((m+l)T\right)\right].$$

With some algebraic manipulation one obtains

$$\left[\boldsymbol{I} - l\ \left(\boldsymbol{I} - e^{-\boldsymbol{A}T}\right)\right]\boldsymbol{x}\left((m+l)T\right) = \boldsymbol{x}(mT)$$

and therefore

$$\boldsymbol{x}\left((m+l)T\right) = \left[\boldsymbol{I} - l\ \left(\boldsymbol{I} - e^{-\boldsymbol{A}T}\right)\right]^{-1}\boldsymbol{x}(mT).$$

In this case the iteration matrix for a fixed cycle-step scheme would be

$$\boldsymbol{M} = \left[\boldsymbol{I} - l\ \left(\boldsymbol{I} - e^{-\boldsymbol{A}T}\right)\right]^{-1}$$

whose spectral radius equals,

$$\rho(\boldsymbol{M}) = \rho\left(\left[\boldsymbol{I} - l\ \left(\boldsymbol{I} - e^{-\boldsymbol{A}T}\right)\right]^{-1}\right) = \max_{\lambda_i}\ |1 - l(1 - e^{-\lambda_i})|^{-1}.$$

Enforcing, for stability, that $\rho(\boldsymbol{M}) < 1$ then leads to the condition

$$\min_{\lambda_i}\ |1 - l\ \left(1 - e^{-\lambda_i}\right)| > 1.$$

Since by hypothesis $\forall i,\ Real[\lambda_i] < 0\ \Rightarrow\ e^{-\lambda_i} > 1\ \Rightarrow\ 1 - e^{-\lambda_i} < 0$. Therefore

$$|1 - l\,(1 - e^{\lambda_i})| = 1 + l\ |e^{\lambda_i} - 1| > 1,\ \forall l.$$

which shows that there are no limitations on the value of the cycle-step l if the backward-Euler style of the Envelope-Following algorithm is used. This property can be shown to be shared by any "implicit"-like formula.

A simple example circuit will show how dramatic the issue of stability can become. Consider the switched capacitor filter in Fig. 3-8, where the circuit input is a sine wave of frequency $10KHz$, and is being clocked by two non-overlapping clocks ϕ_1 and ϕ_2 at a frequency of $1MHz$. Figure 3-9 shows the transient start-up of this circuit when computed with the forward-Euler and backward-Euler styles of Envelope Following. In the figure, the waveform denoted as FE(2) shows the transient of the output node when computed with the forward-Euler equation using a fixed cycle-step of $l = 2$. It should be obvious that even with such a small cycle-step the computed waveform is completely inaccurate and furthermore, instability is beginning to develop at the later stages of simulation interval. We point out that $l = 2$ would be the smallest cycle-step for which the envelope-following algorithm could provide any speedup, which clearly implies that this formulation is not usable when simulating real circuits. While using a fixed cycle-step is unlikely to be a good strategy in many situations, it is nevertheless a good indication of the behavior of the underlying algorithm. In the same figure, two other waveforms are shown for comparison purposes. The waveform denoted as BE(5) indicates the transient start-up obtained using using the backward-Euler formulation and a fixed cycle-step of $l = 5$. Clearly this waveform indicates a stable algorithm and it is in fact quite accurate. The third waveform, denoted as BE(20) is the output transient computed again using the backward-Euler style of formulation, but now with a fixed cycle-step of $l = 20$. While this waveform is not very accurate (compare with BE(5)), and would probably not satisfy any reasonable accuracy criteria, it nevertheless shows that this formulation is indeed stable and can withstand the usage of large cycle-steps, accuracy permitting.

As we have shown, the backward-Euler style algorithm allows for larger cycle-steps than the forward-Euler based approach. However, it leads to a more difficult to solve equation for each cycle-step. To see this, consider computing $\boldsymbol{x}(lT)$ given $\boldsymbol{x}(0)$ based on Eqn. (3.10). An $\boldsymbol{x}((l-1)T)$ must be determined such that when used as an initial condition for Eqn. (3.1), the $\boldsymbol{x}(lT)$ computed with standard discretization techniques satisfies $\boldsymbol{x}(lT) - \boldsymbol{x}(0) = l\ [\boldsymbol{x}(lT) - \boldsymbol{x}((l-1)T)]$. This is a boundary value problem, and is in general difficult to solve. For the case of switching power or filter circuits, the above boundary value problem can be solved efficiently using a Newton or shooting method [10], and this is presented in the next section.

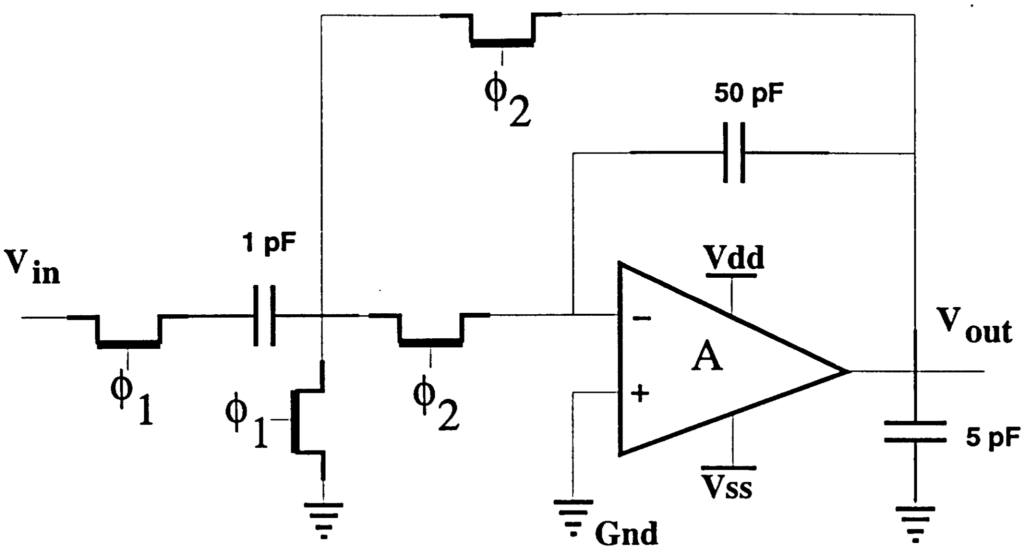

Figure 3-8: Switched-capacitor filter circuit with input sine wave at $10KHz$ being clocked by two non-overlapping clock phases of frequency $1MHz$.

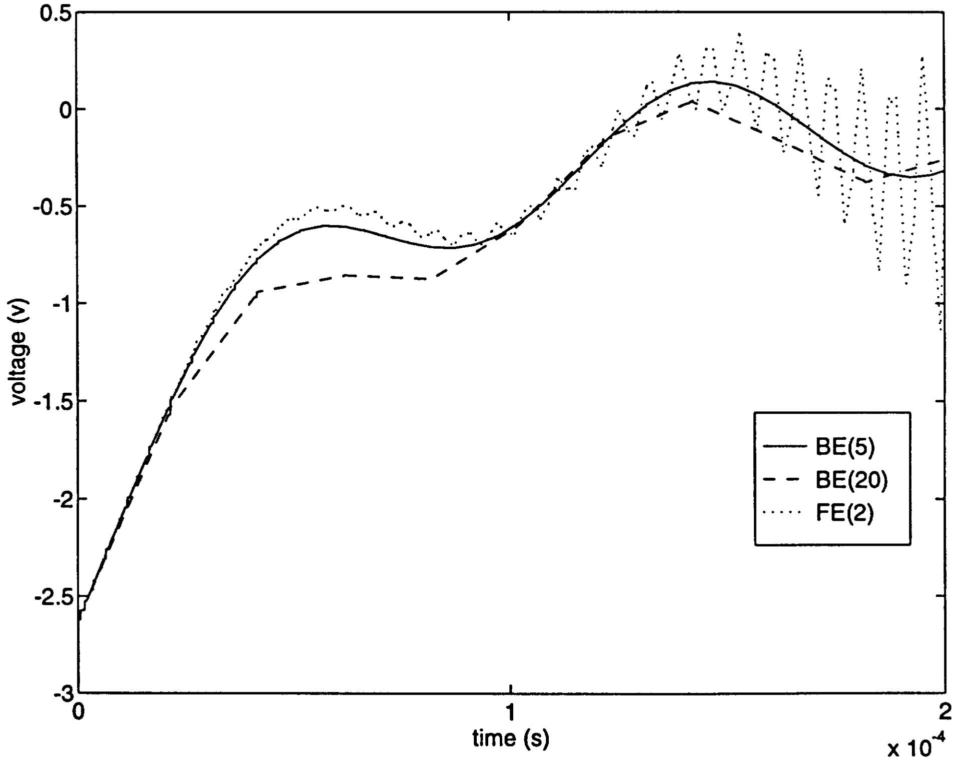

Figure 3-9: Comparison of the transient start-up Envelope-Following simulation of a switched-capacitor circuit for Forward- and Backward-Euler alignment schemes. Shown in the plot are three waveforms, obtained by using the forward-Euler formulation with a fixed cycle-step of 2 (FE(2)), and the backward-Euler formulation with fixed cycle-steps of 5 (BE(5)) and 20 (BE(20)).

3.2.3 Solution by Newton

As mentioned in the previous section, each cycle-step of a backward-Euler envelope-following algorithm applied to Eqn. (3.1) involves finding an $\boldsymbol{x}\left((m+l-1)T\right)$ which satisfies

$$\boldsymbol{x}\left((m+l)T\right)-\boldsymbol{x}(mT)=l\ \left[\boldsymbol{x}\left((m+l)T\right)-\boldsymbol{x}\left((m+l-1)T\right)\right] \tag{3.11}$$

where $\boldsymbol{x}(mT)$ is known from the previous cycle-step and $\boldsymbol{x}\left((m+l)T\right)$ is determined from $\boldsymbol{x}\left((m+l-1)T\right)$ by solving Eqn. (3.1) over one cycle. Using the state transition function defined in Eqn. (3.2), the relation between $\boldsymbol{x}\left((m+l)T\right)$ and $\boldsymbol{x}\left((m+l-1)T\right)$ can be written as

$$\boldsymbol{x}\left((m+l)T\right)=\boldsymbol{\phi}\left(\boldsymbol{x}\left((m+l-1)T\right),(m+l-1)T,(m+l)T\right). \tag{3.12}$$

Using this relation in Eqn. (3.11) yields the nonlinear algebraic equation

$$\begin{aligned}&\boldsymbol{\phi}\left(\boldsymbol{x}((m+l-1)T),(m+l-1)T,(m+l)T\right)-\boldsymbol{x}(mT)=\\&\quad l\ \left[\boldsymbol{\phi}\left(\boldsymbol{x}((m+l-1)T),(m+l-1)T,(m+l)T\right)-\boldsymbol{x}\left((m+l-1)T\right)\right]\end{aligned} \tag{3.13}$$

from which $\boldsymbol{x}\left((m+l-1)T\right)$ can be determined given $\boldsymbol{x}(mT)$.

An iterative Newton's method can be applied to solving Eqn. (3.13) for $\boldsymbol{x}\left((m+l-1)T\right)$. In general, the Newton method applied to the problem of finding an $\boldsymbol{x}\in\mathbb{R}^N$ such that $\boldsymbol{F}(\boldsymbol{x})=0$, $\boldsymbol{F}:\mathbb{R}^n\rightarrow\mathbb{R}^n$, yields the iteration equation

$$\boldsymbol{J_F}(\boldsymbol{x}^k)\left[\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\right]=-\boldsymbol{F}(\boldsymbol{x}^k), \tag{3.14}$$

where k is the Newton iteration count and $\boldsymbol{J_F}\in\mathbb{R}^{N\times N}$ is the Jacobian of $\boldsymbol{F}$. Reorganizing Eqn. (3.13) into a form to apply Newton's method leads to

$$\begin{aligned}\boldsymbol{F}\left(\boldsymbol{x}\left((m+l-1)T\right)\right)=&\ \boldsymbol{\phi}\left(\boldsymbol{x}\left((m+l-1)T\right),(m+l-1)T,(m+l)T\right)\\&-\frac{l}{l-1}\boldsymbol{x}\left((m+l-1)T\right)+\frac{1}{l-1}\boldsymbol{x}(mT)=\mathbf{o}.\end{aligned} \tag{3.15}$$

In this case, $\boldsymbol{J_F}$ is given by

$$\boldsymbol{J_F}\left(\boldsymbol{x}\left((m+l-1)T\right)\right)=\frac{\partial\boldsymbol{\phi}\left(\boldsymbol{x}\left((m+l-1)T\right),(m+l-1)T,(m+l)T\right)}{\partial\boldsymbol{x}\left((m+l-1)T\right)}-\frac{l}{l-1}\boldsymbol{I}_N \tag{3.16}$$

where $\boldsymbol{I}_N$ is the identity matrix of size N.

The most time-consuming computation in this Newton iteration is evaluating $\boldsymbol{J_F}$ and $\boldsymbol{F}()$, which involves computing the state transition function and its derivative. The state transition function can be evaluated by numerically integrating Eqn. (3.1) from

$(m+l-1)T$ to $(m+l)T$ given $\boldsymbol{x}((m+l-1)T)$. The derivative of the state transition function, referred to as the sensitivity matrix, represents the sensitivity of $\boldsymbol{x}((m+l)T)$ to perturbations in $\boldsymbol{x}((m+l-1)T)$ and can be computed with a small amount of additional work during the numerical integration, as is described in the following section.

3.2.4 Sensitivity Computation

To see how the computation of the state transition function and its derivative fit together, consider numerically integrating Eqn. (3.1) with backward-Euler, which we chose for its simplicity and because it is effective for problems with rapidly varying inputs, like clocks. Given some initial time t_0 and some initial condition $\boldsymbol{x}(t_0)$, applying backward-Euler to Eqn. (3.1) results in the algebraic equation

$$\boldsymbol{g}(\boldsymbol{x}(t_0+h)) = \frac{1}{h}\left(\boldsymbol{p}(\boldsymbol{x}(t_0+h)) - \boldsymbol{p}(\boldsymbol{x}(t_0))\right) + \boldsymbol{f}(\boldsymbol{x}(t_0+h)) = 0, \tag{3.17}$$

where $h \in \mathbb{R}$ is the timestep. Note we have dropped explicitly denoting the dependence of $\boldsymbol{p}$ and $\boldsymbol{f}$ on the input $\boldsymbol{u}$ for simplicity.

Equation (3.17) is usually solved with Newton's method, for which the iteration equation is

$$\begin{aligned} J_{\boldsymbol{g}}\left(\boldsymbol{x}^{(k)}(t_0+h)\right)\left(\boldsymbol{x}^{(k+1)}(t_0+h) - \boldsymbol{x}^{(k)}(t_0+h)\right) &= \\ -\boldsymbol{g}\left(\boldsymbol{x}^{(k)}(t_0+h), \boldsymbol{x}(t_0)\right) \end{aligned} \tag{3.18}$$

where k is the Newton iteration index, and $J_{\boldsymbol{g}}(\boldsymbol{x}(t)) \in \mathbb{R}^{N\times N}$ is the Frechet derivative of the nonlinear equation (3.17) and is given by

$$J_{\boldsymbol{g}}(\boldsymbol{x}(t)) = \frac{\partial \boldsymbol{g}(\boldsymbol{x}(t))}{\partial \boldsymbol{x}(t)} = \frac{1}{h}\frac{\partial \boldsymbol{p}(\boldsymbol{x}(t))}{\partial \boldsymbol{x}(t)} + \frac{\partial \boldsymbol{f}(\boldsymbol{x}(t))}{\partial \boldsymbol{x}(t)}. \tag{3.19}$$

Solving Eqn. (3.17) yields an approximation to $\boldsymbol{x}(t_0+h) = \boldsymbol{\phi}(\boldsymbol{x}(t_0), t_0, t_0+h)$. Implicitly differentiating Eqn. (3.17) for $\boldsymbol{x}(t_0+h)$ with respect to $\boldsymbol{x}(t_0)$ yields

$$J_{\boldsymbol{g}}(\boldsymbol{x}(t_0+h))\frac{\partial \boldsymbol{x}(t_0+h)}{\partial \boldsymbol{x}(t_0)} = \frac{1}{h}\frac{\partial \boldsymbol{p}(\boldsymbol{x}(t_0))}{\partial \boldsymbol{x}(t_0)}. \tag{3.20}$$

Given an $\boldsymbol{x}(t_0)$, Eqn. (3.17) can be repeatedly applied to approximately compute $\boldsymbol{x}(t_0+T) = \boldsymbol{\phi}(\boldsymbol{x}(t_0), t_0, t_0+T)$, using Newton's method and Eqn. (3.18) at every timepoint in the interval $[t_0, t_0+T]$.

Similarly Eqn. (3.20) can be repeatedly applied to approximately compute the sensitivity matrix

$$\frac{\partial \boldsymbol{x}(t_0+T)}{\partial \boldsymbol{x}(t_0)} = \frac{\partial \boldsymbol{\phi}(\boldsymbol{x}(t_0), t_0, t_0+T)}{\partial \boldsymbol{x}(t_0)}.$$

As mentioned previously, this can be accomplished with only a small amount of additional work during the numerical integration [13]. To understand the process, note that direct application of Eqn. 3.20 provides the sensitivity matrix

$$\frac{\partial x(t_0+h)}{\partial x(t_0)} = \phi(x(t_0), t_0, t_0+h).$$

Assuming then that $x(t_1) = x(t_0+h)$ has been computed by application of Eqn. (3.17) the sensitivity matrix can be readily updated by noting that

$$\phi(x(t_0), t_0, t_1+h) = \frac{\partial x(t_1+h)}{\partial x(t_0)} = \frac{\partial x(t_1+h)}{\partial x(t_1)} \frac{\partial x(t_1)}{\partial x(t_0)} \tag{3.21}$$

and therefore

$$\frac{\partial x(t_1+h)}{\partial x(t_1)} = \frac{\partial x(t_1+h)}{\partial x(t_0)} \left[\frac{\partial x(t_1)}{\partial x(t_0)}\right]^{-1} \tag{3.22}$$

which when plugged into Eqn. (3.20) finally leads to

$$J_g\left(x(t_1+h)\right) \frac{\partial x(t_1+h)}{\partial x(t_0)} = \frac{1}{h} \frac{\partial p\left(x(t_1)\right)}{\partial x(t_1)} \; \frac{\partial x(t_1)}{\partial x(t_0)} \tag{3.23}$$

Therefore repeated application of Eqn. (3.23) leads to an approximate computation of the sensitivity matrix

$$\frac{\partial x(t_0+T)}{\partial x(t_0)} = \frac{\partial \phi(x(t_0), t_0, t_0+T)}{\partial x(t_0)}.$$

Furthermore, note that at any timepoint J_g is required in both Eqn. (3.18) and Eqn. (3.20), and thus the sensitivity matrix update can be made very efficient by factoring J_g once and using it for both computations. For large problems, though, computing the dense $N \times N$ sensitivity matrix can become expensive since its update requires N back-solves, each of which cost at least N effort.

The standard envelope-following algorithm as described is shown in Algorithm 3.2.1. Only two of the steps mentioned in the algorithm were not already discussed. These are the selection of the cycle step and the prediction of a first guess for the state l cycle-steps ahead. These steps are very similar to equivalent steps one takes when performing the timepoint per timepoint simulation. The guess used for the state is obtained with an explicit low-order predictor using previously computed values. The selection of the cycle-step l is done in a manner very similar to that of the selection of a new timestep at every timepoint, using some approximation to what now should be seen as the local envelope truncation error. Both of these techniques will be further discussed in section 3.3 ahead. Graphically Algorithm 3.2.1 can be described as shown in Fig. 3-10.

Computational procedures based on this algorithm are particularly efficient when used to simulate simplified switching power converters [10, 11]. The method is not effective, *Algorithm 3.2.1 (Standard Envelope-Following Algorithm).*

```
$m = 0$
While $mT <$ STOPTIME {
    Select the cycle-step $l$
    Predict a first guess, $x^0((m+l-1)T)$
    Numerically integrate Eqn. (3.1) from $(m+l-1)T$ to $(m+l)T$ to
      compute $x^0((m+l)T)$ and $\frac{\partial x^0((m+l)T)}{\partial x((m+l-1)T)}$
    Compute $J_F(x^0((m+l-1)T)$ as in Eqn. (3.16)
    Set k=0
    Until Newton Converges {
        Solve the Newton update equation for $x^{k+1}((m+l-1)T)$.
        Numerically integrate Eqn. (3.1) from $(m+l-1)T$ to $(m+l)T$
          to compute $x^{k+1}((m+l)T)$
    }
    $m = m + l$
}
```

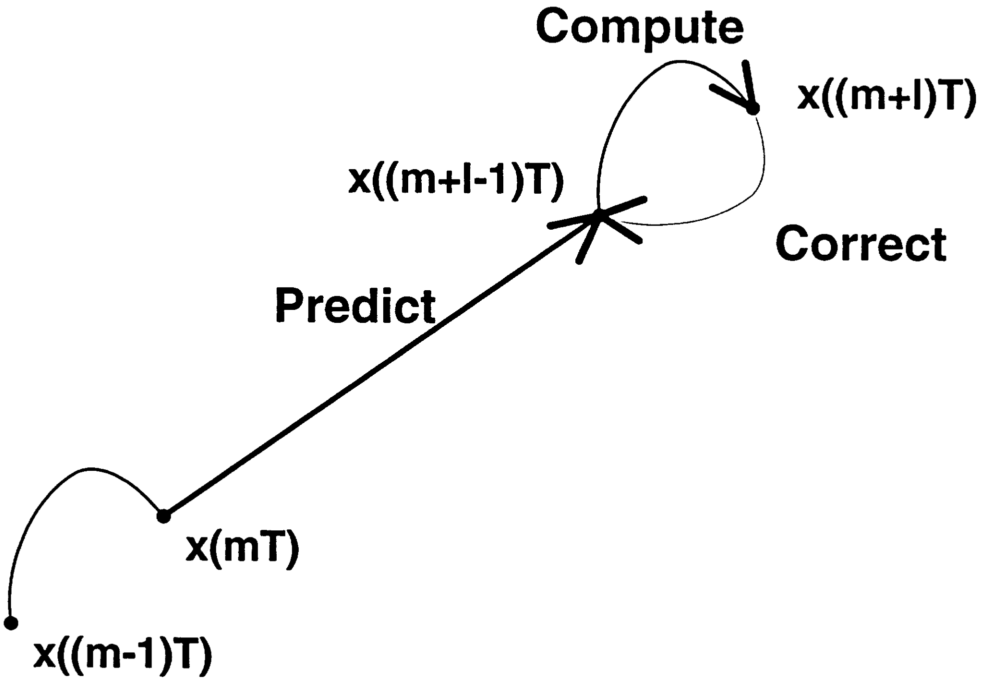

Figure 3-10: Graphical description of the implicit scheme standard Envelope-Following Algorithm, showing how the cycle integration and the alignment equation are fit together to provide a consistent extrapolation of the behavior of the envelope l cycles ahead.

however, if there are states in the system which change rapidly and dramatically due to small changes in much more slowly changing states [11], as we shall see in the next section.

3.3 Envelope-Following Modeling of Clocked Analog Circuits

An envelope-following method has been implemented in the NITSWIT [10] simulation program. The program is written in "C", and runs under the UNIX operating system. The program uses a trapezoidal-rule based envelope-following algorithm for which the cycle-step update equation is

$$\frac{\boldsymbol{x}\left((m+l-1)T\right)-\boldsymbol{x}(mT)}{l-1}= \\ \tfrac{1}{2}\left[\boldsymbol{x}\left((m+l)T\right)-\boldsymbol{x}\left((m+l-1)T\right)\right]+\tfrac{1}{2}\left[\boldsymbol{x}(mT)-\boldsymbol{x}\left((m-1)T\right)\right] \qquad (3.24)$$

The terms $[\boldsymbol{x}((m+l)T)-\boldsymbol{x}((m+l-1)T)]$ and $[\boldsymbol{x}(mT)-\boldsymbol{x}((m-1)T)]$ in Eqn. (3.24) can be thought of as envelope derivatives at $\boldsymbol{x}((m+l)T)$ and $\boldsymbol{x}(mT)$ respectively. Just as in the classical trapezoidal-rule, the average of these two "derivatives" is used in the cycle-step update equation.

The Newton method described above is used to solve for the cycle-step update and as in standard integrators, the cycle-steps for the follower are selected automatically, based on examining both the envelope truncation error and the iteration count for the Newton method. The automatic cycle-step algorithm is shown in pseudo-code in Algorithm 3.3.1. It takes the last few cycle-steps and two input vectors, namely the guess or predictor used for the state and the newly computed state value and returns a new cycle-step. The predictor for the value of the state at the beginning of a cycle is obtained with an explicit low-order polynomial using values of the states computed at previous cycle boundaries. The computation of the envelope truncation error reflects the fact that a trapezoidal-rule based envelope-following algorithm was used. Also, as you can see from the algorithm there are several parameters and fudge factors which either must be set based on the desired accuracy or are dependent upon the integration-like method used for formulating the alignment equation. For our computations, the values shown in Table 3.3 were found to be fairly adequate in most circumstances. At the end of each Newton iteration if convergence was not achieved, the cyclestep is immediately halved and a new integration is performed. If the Newton iteration was convergent, then the the algorithm is invoked and an estimate of the envelope truncation error is computed. If this error satisfies the desired envelope computation accuracy, the newly computed values are accepted and a new cycle-step is selected automatically. If on the other hand the envelope truncation error does not satisfy the accuracy criteria, the step is rejected, a smaller cycle-step is computed and a new integration is performed.

Parameter		value
ETE_{rel}	=	0.01
ETE_{abs}	=	0.005
k_1	=	10
k_2	=	64
k_3	=	0.75
k_4	=	8
k_5	=	1.2
k_6	=	0.9

Table 3-1: Parameters used for the computation of the Envelope truncation error and the selection of new cycle-steps by the Envelope-Following Algorithm.

3.3.1 Near-Linearity of the State Transition Function

For the envelope-following approach to be more efficient than classical methods for a given problem, it must be possible to accurately represent the envelope of interest with a small fraction of the clock cycles. In addition, the Newton method used to solve the envelope update equations must converge rapidly, as each Newton iteration involves numerically simulating an entire converter clock cycle. If the problem is simulating a switched capacitor filter or an "open-loop" converter, that is a circuit in which the frequency and duty cycle of the input clock are not functions of the filter or circuit state, the Newton method does converge very rapidly.

That the shooting Newton method should converge rapidly is clear for the case where an open-loop converter is constructed from clock-controlled ideal switches and other linear circuit elements. For such a converter the state transition function is affine (linear plus a constant) [14] and $\boldsymbol{J_F}$ in Eqn. (3.16) is a constant. This implies that the Newton method will *always converge in one iteration.*

A simple example will make this point more clear. Consider the simple switched-capacitor circuit in Fig. 3-11, which is clocked by two non-overlapping clocks with period T, as indicated. Assume that the MOS switches are ideal. It is relatively easy to determine the value of the output at end of a cycle. During the ϕ_1 phase the capacitor

Algorithm 3.3.1 (Automatic Cycle-Step Selection).

```
$check\_select\_cyclestep(l_{m-2}, l_{m-1}, l_m, \boldsymbol{predictor}, \boldsymbol{state})$
{
    Compute the desired accuracy for envelope truncation error:
      $MAX_{ETE} = (ETE_{rel}\ |\boldsymbol{state}|_1 + ETE_{abs})\ k_1$
    Compute the predictor error:  $PTE = |\boldsymbol{predictor} - \boldsymbol{state}|_1$
    /* Envelope truncation error is $ETE \approx PTE$ */
    Compute ratio of envelope truncation error to desired
      accuracy:  $ETE_{ratio} = \frac{PTE}{MAX_{ETE}} |\frac{l_m - l_{m-1}}{2(l_m - l_{m-2})}|$
    $if$ $(ETE_{ratio} > 1)$ {      /* reject cycle-step */
        $if$ $(ETE_{ratio} > k_2)$
            recompute with $l_m = l_m / k_2$
        $else$
            recompute with $l_m = k_3\ l_m / \sqrt[3]{ETE_{ratio}}$
        $l_m = \lfloor l_m \rfloor$     /* cycle-step is an integer */
        $l_m = \max(l_m, 1)$
        return($REJECT$)
    } $else$ { /* accept cycle-step; select new cycle-step */
        $if$ $(k_4\ ETE_{ratio} < 1)$
            recompute with $l_m = 2\ l_m$
        $else\ if$ $(k_5\ \sqrt[3]{l_m} < 1)$
            use $l_m = (l_m - l_{m-1}) \left[1 + k_6 \left((\sqrt[3]{ETE_{ratio}})^{-1} - k_5\right)\right]$
        $else$
            use the same $l_m$
        $l_m = \lfloor l_m \rfloor$     /* cycle-step is an integer */
        $l_m = \min(l_m, MAX_{step})$
        $l_m = \max(l_m, 1)$
        return($ACCEPT$)
    }
}
```

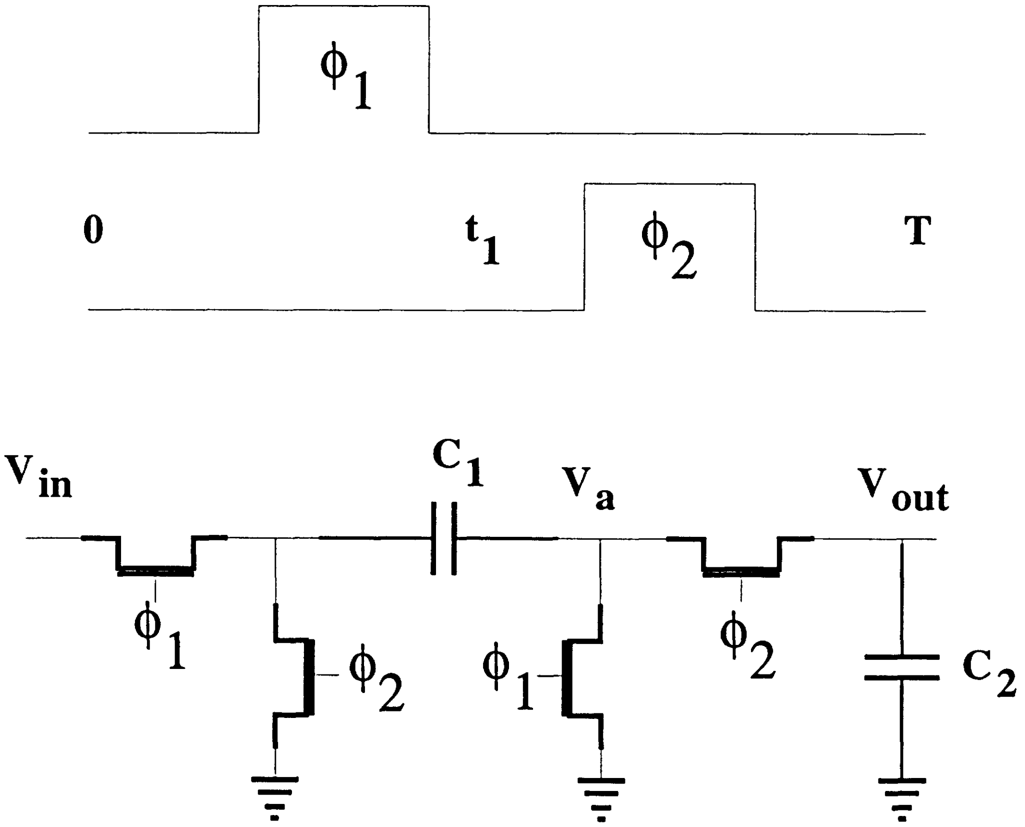

Figure 3-11: Switched-capacitor circuit being clocked by two non-overlapping clock phases of frequency $1/T$. The MOS switches are assumed to be ideal.

C_1 is charged to the value of v_{in}, while the capacitor C_2 is disconnected from the circuit. At the end of the ϕ_1 the switches controlled by that phase open. At this point the charge in C_1 equals $Q_1(t_1) = C_1 \; v_{in}(t_1)$ where t_1 is the time at which the switches opened. Meanwhile the charge in C_2 has not changed and it is equal to $Q_2(t_1) = C_2 \; v_{out}(0)$. When ϕ_2 is high, the switches ϕ_2 controls are closed and the two capacitors are placed in parallel. At this point charge will be shared between C_1 and C_2 such that the total charge equals $Q(T) = Q_1(T) + Q_2(T) = (C_1 + C_2) \; v_{out}(T)$ at the end of the cycle when ϕ_2 ends and node 2 is again separated from the rest of the circuit. Since the charge is conserved, we can write

$$\begin{aligned} Q_1(t_1) + Q_2(t_1) &= Q_1(T) + Q_2(T) \\ C_1 \; v_{in}(t_1) + C_2 \; v_{out}(0) &= (C_1 + C_2) \; v_{out}(T) \end{aligned}$$

Solving for the value of the output node at the end of the cycle, we get:

$$v_{out}(T) = \frac{C_2}{C_1 + C_2} \; v_{out}(0) + \frac{C_1}{C_1 + C_2} \; v_{in}(t_1)$$

or in other words the state transition matrix

$$\phi\,(0, T, v_{out}(0)) = k_1 \; v_{out}(0) \; + \; k_2$$

is affine as stated. Therefore, as previously mentioned, the Newton method applied to the alignment equation will converge in one iteration.

For realistic circuits in which switches are implemented by transistors and diodes, the state transition function *over one cycle* will still be nearly affine, and in our experience, the Newton method typically converges in three or fewer iterations at each cycle-step.

It is possible to further exploit the nearly affine property of the open-loop converter state transition function by only computing $\boldsymbol{J_F}$ for the first Newton iteration in each cycle-step. This is a significant savings, as it avoids recomputing the sensitivity matrix and usually doesn't slow the Newton method's convergence.

3.3.2 Discussion of Experimental Results

Exactly how the envelope-following method behaves can be seen by examining Figures 3-12 and 3-13. Figure 3-12 shows the output voltage of the buck converter introduced in Fig. 3-1, during a power-up transient. This simulation was obtained using the Envelope-Following algorithm as described in section 3.2, and the thicker sections indicate where cycles were in fact computed. As can be seen in the figure, the envelope-following method computes only occasional cycles, shown in the figure with marks, but the output

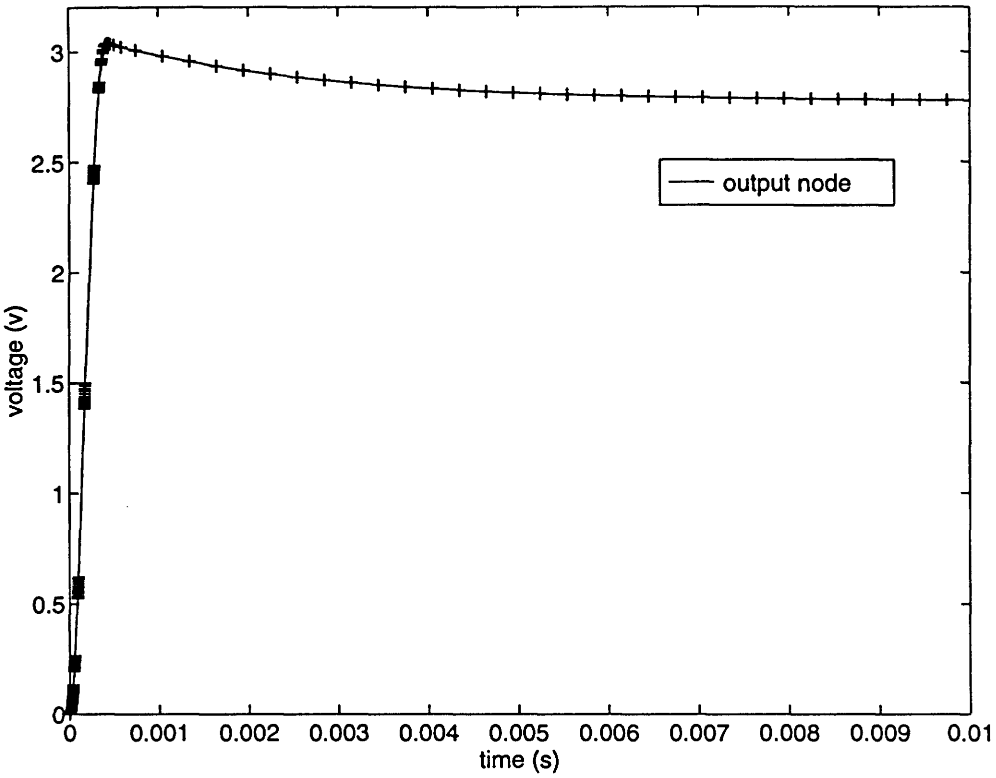

Figure 3-12: Envelope-Following simulation of the buck converter output voltage during power-up transient.

voltage for the computed cycles are within a few percent of those computed with the classical method. Similarly, Fig. 3-13 shows the start-up transient of the switched capacitor filter introduced in Fig. 3-8, where again the circuit input is a sine wave of frequency $10KHz$, and is being clocked by two non-overlapping clocks ϕ_1 and ϕ_2 at a frequency of $1MHz$. The CPU times obtained simulating these circuits with the envelope following approach are quite encouraging. The results are shown in Table 3-2 where the open-loop buck converter is called *dbuck*, and the switched capacitor filter is called *scop*.

Circuit	N	Total Cycles	Classical (sec)	EF (sec)	EF (cycles)
scop	13	500	390	162.5	75
dbuck	4	1000	359	34.5	50

Table 3-2: CPU time comparisons for Classical versus Envelope-Following Simulation algorithms. The number of cycles shown corresponds to the simulation interval divided by the clock period. For the envelope-following approach, the number of effectively simulated cycles is also shown. Times are in seconds on a SUN4/260.

Such encouraging results are not always obtained however. Consider a similar "buck"

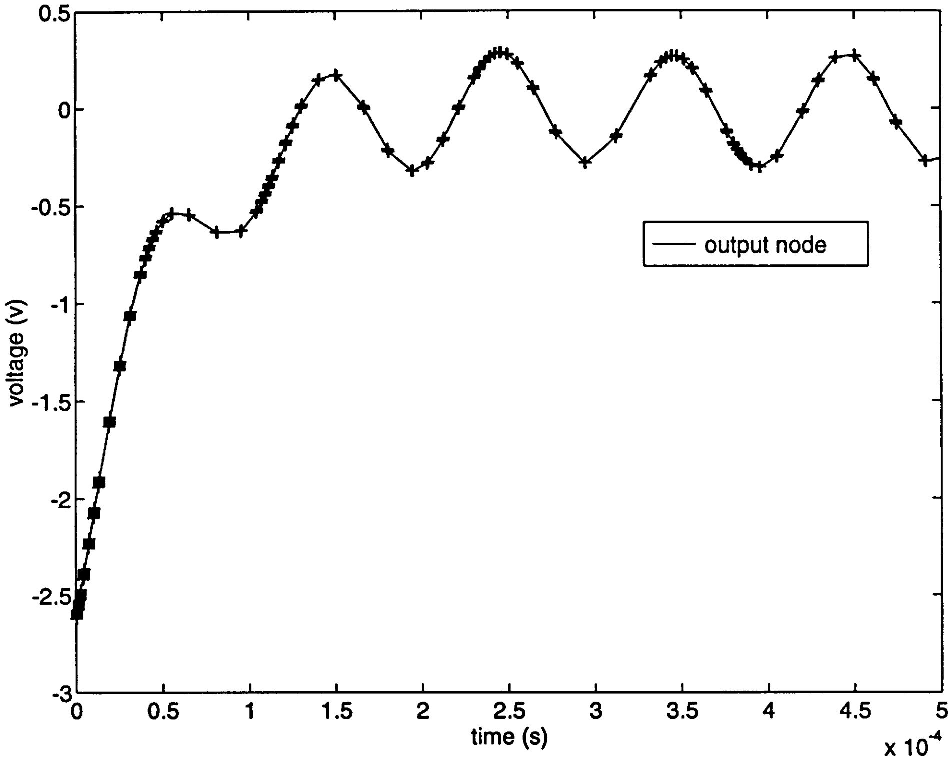

Figure 3-13: Envelope-Following simulation of the switched-capacitor filter output voltage during transient start-up.

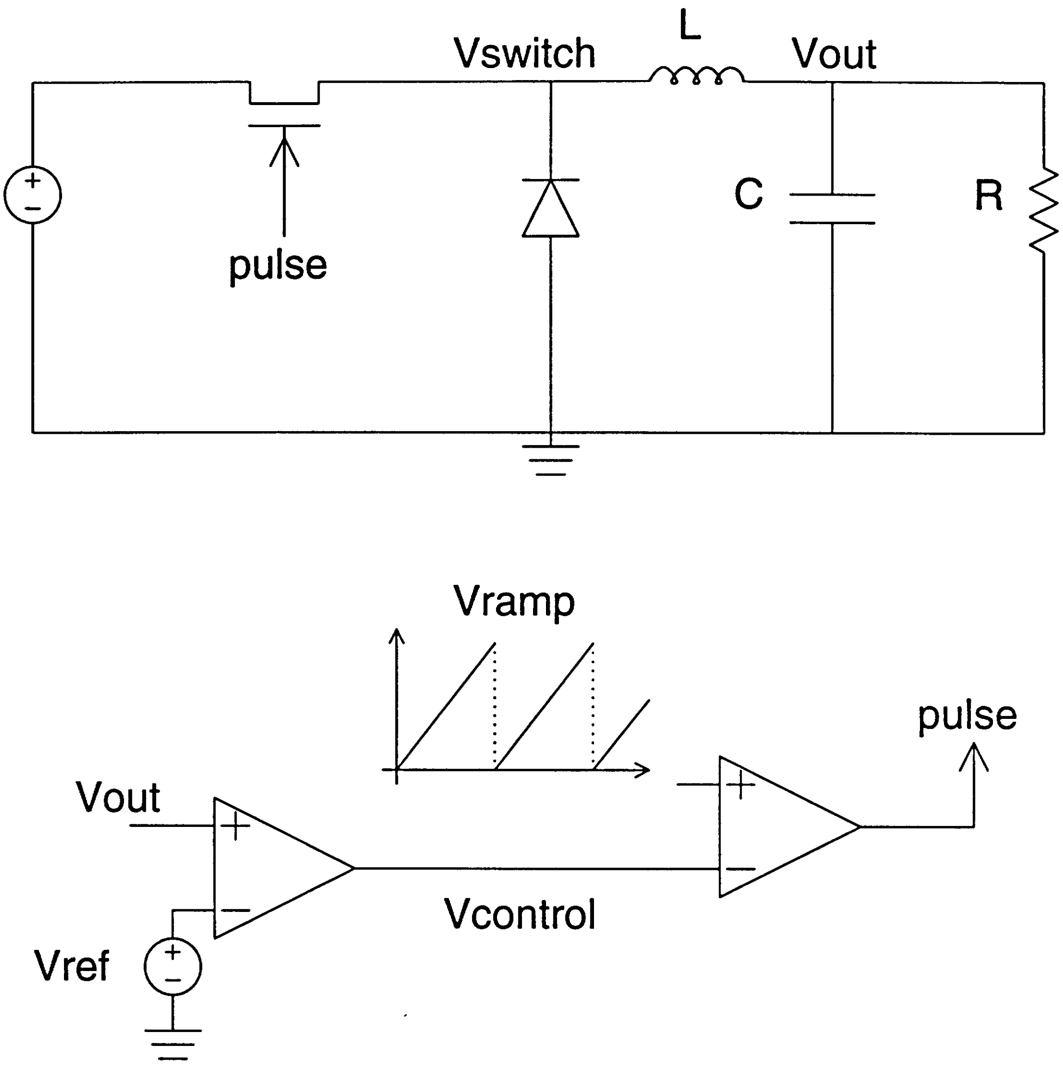

Figure 3-14: Closed-loop buck converter.

DC-DC converter circuit in a closed loop configuration as shown in Fig. 3-14. In the closed-loop configuration, the input clock duty cycle is a function of the converter's output voltage, which is fed back and compared to a reference voltage. In principle, this converter should be able to be simulated with the envelope-following method as implemented in NITSWIT, and the same type of quantitative results was expected. However, the results obtained (again, shown in Table 3-3) are not that encouraging. Experiments with NITSWIT indicate that the obvious explanation for the poorer efficiency, that closed-loop converters have state transition functions which are more nonlinear, is *not* the dominant problem [11].

Figure 3-15 show the output of the closed-loop buck converter during a power-up transient as obtained with the standard envelope-following algorithm. As expected, dur-

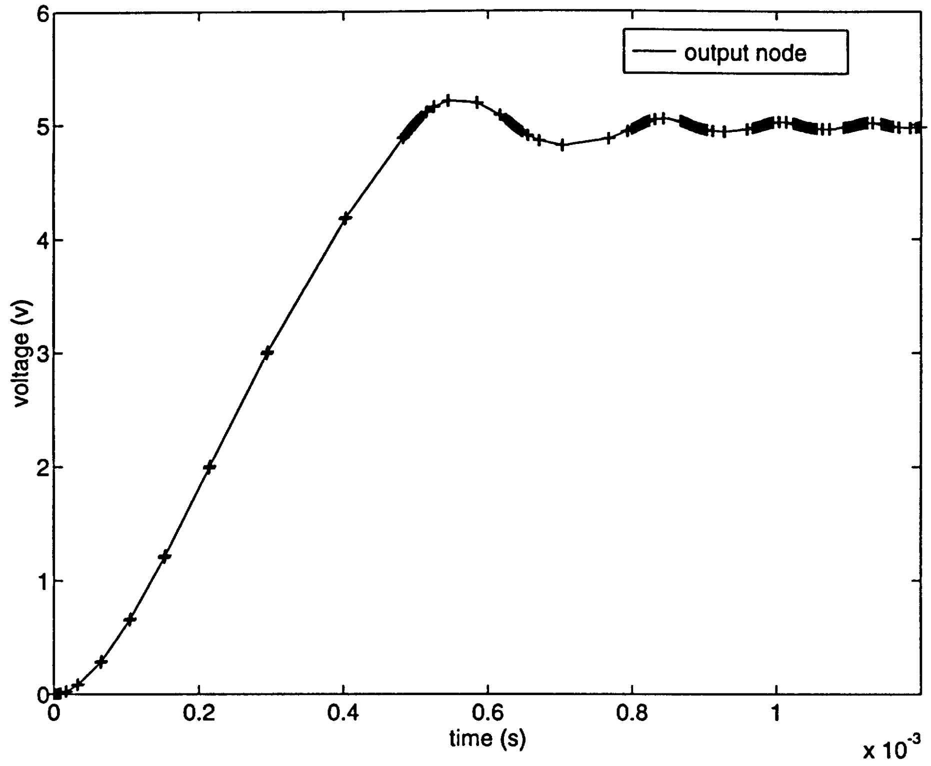

Figure 3-15: Closed-loop buck converter output node during power-up transient obtained with the standard envelope-following technique.

ing the initial phase of the transient power-up the algorithm computes only occasional cycles (again shown with plus marks on the figure). As the converter output approaches steady-state, one would expect the number of computed cycles to decrease, as the envelope of the solution changes only slightly. However, as shown, a surprising number of cycles are in fact being computed when the converter output is close to steady-state. A more detailed look into what is happening during the simulation process reveals that even though a few instances of non-convergence of the Newton algorithm are seen, for the most part it is the local truncation error control mechanism, which is not allowing the cycle step to become very large.

The difficulty simulating closed-loop loop converters, such as the one presented, is that they typically include control circuitry which produce large, very rapid responses to small changes in the converter output. In this case, a closer look at the behavior of some of the control nodes shows that as we approach steady state, these nodes still have wildly varying behavior. Figure. 3-16 shows the behavior of the intermediate control node (denoted as $V_{control}$ in Fig 3-14) during the power-up transient interval. As shown, this controlling node changes rapidly as the converter approaches steady-state and, if envelope-followed, will restrict the length of the cycle step, or equivalently allow few

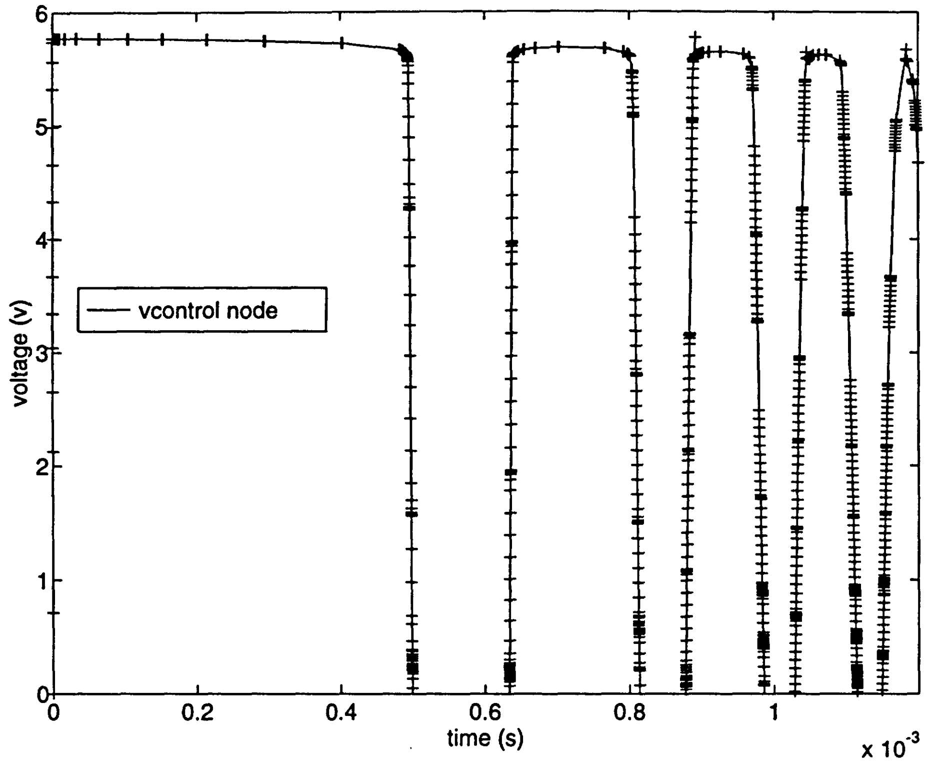

Figure 3-16: Closed-loop buck converter $V_{control}$ node during power-up transient obtained with the standard envelope-following technique.

cycles to be skipped. This undesirable behavior strongly limits the applicability of the standard envelope-following algorithm.

3.4 Model Order Reduction for Envelope-Following Algorithms

If the envelope-following algorithm, as described in section 3.2, is used unmodified, the cycle-step will be constrained by the component of x with the fastest changing envelope. This can be unnecessarily conservative, as components of x which have rapidly changing envelopes are likely to be nearly algebraic functions of other, more slowly changing components, at least over the time scale of one clock period. That is, these nearly algebraic, or quasi-algebraic components of x can be computed from a subset of the present state, and therefore envelopes associated with quasi-algebraic nodes need not be computed with a formula like Eqn. (3.11).

To clarify the above claim consider, for example, the circuit in Figure 3-17 where the driving source is assumed to be periodic with period T. Further assume that $\tau_s =$

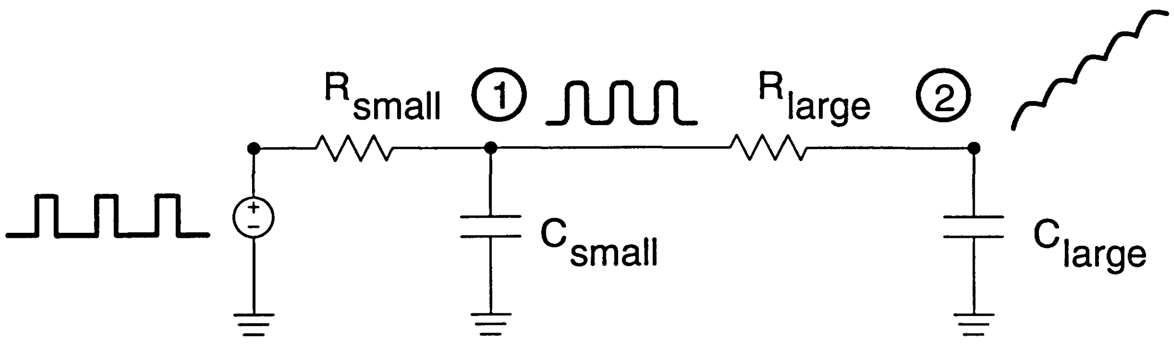

Figure 3-17: Example circuit showing the difference between state-variables and quasi-algebraic variables.

$R_{small}C_{small} \ll T$ and that $\tau_l = R_{large}C_{large} > T$. Let $v_1(0), v_1(T), \ldots, v_1(nT)$ and $v_2(0), v_2(T), \ldots, v_2(nT)$ be the envelope of, respectively, nodes 1 and 2. These envelopes are obtained by sampling the voltage at those nodes at intervals of T, the clock period.

Clearly in this circuit, the value of $v_1(t)$ is almost independent of the value of $v_1(t-T)$. However, the value of $v_2(t)$ is very dependent upon the value of $v_2(t-T)$. This is due to the time-constants associated with the behavior of each node. In other words, to be able to know the state of node 2 at the end of a cycle, we need to know its value at the beginning of the cycle. To indicate this fact, we say that v_2 is a *state* variable. On the other hand, to know the voltage at node 1 at any point during a cycle, it is enough to know the current values of node 2 and the value of the source. For that reason, we call v_1 a *quasi-algebraic* variable. Therefore we can claim that the value of a quasi-algebraic variable at the end of a cycle is nearly independent of it's values at the beginning of a cycle, while the same does not hold for state variables.

Returning to the closed-loop buck-converter in Figure 3-14, we saw that some of the variables in the controlling feedback loop experience rapid changes as the converter's output approaches steady-state. That these controller variables are nearly algebraic functions of other system states implies that they are independent of their own past, and need not be envelope-followed. Eliminating these variables from the envelope computation will allow larger cycle-steps and provide for a faster simulation.

3.4.1 Removing Quasi-Algebraic Variables

We now precisely define a Quasi-Algebraic variable.

Definition 3.4.1. **(Quasi-Algebraic Variable)** We say that a component x_a of a vector $x \in \mathbb{R}^N$ is a quasi-algebraic variable if all the components of

$$\frac{\partial \boldsymbol{x}(T)}{\partial x_a(0)} = \left[\frac{\partial x_1(T)}{\partial x_a(0)}, \cdots \frac{\partial x_N(T)}{\partial x_a(0)}\right].$$

are *negligible.* The set of all quasi-algebraic variables is denoted as $\boldsymbol{x}_a$. Those components of $\boldsymbol{x}$ which are not quasi-algebraic are called state variables, and denoted as $\boldsymbol{x}_s$.

In others words, the value of any quasi-algebraic component x_a of $\boldsymbol{x}$ at the end of cycle is nearly independent of its value at the beginning of the cycle, and it is also independent of the value of any other quasi-algebraic variable at the beginning of the cycle. Similarly the value of any state variable x_s at the end of a cycle is also nearly independent of the value of any quasi-algebraic variable at the beginning of the cycle.

Determining which components of $\boldsymbol{x}$ are quasi-algebraic can be accomplished using the data in the sensitivity matrix already required to solve Eqn. (3.13) with Newton's method. To see this, note that a component x_i in $\boldsymbol{x}$ is quasi-algebraic in one clock period if all components of $\boldsymbol{x}$ are insensitive to x_i's value at the beginning of a period. By definition, entry (i,j) in the sensitivity matrix represents the sensitivity of $x_i((m+l)T)$ to perturbations in $x_j((m+l-1)T)$. Therefore, x_i is a quasi-algebraic component if every element in i^{th} column of the sensitivity matrix is nearly zero.

Now let the components of $\boldsymbol{x}$ be divided into two vectors, as outlined above: $\boldsymbol{x}_s$, the vector of true states, and $\boldsymbol{x}_a$, the quasi-algebraic vector. Then the sensitivity matrix can be reordered so that

$$\frac{\partial \phi(\boldsymbol{x}_s(T), \boldsymbol{x}_a(T))}{\partial(\boldsymbol{x}_s(0), \boldsymbol{x}_a(0))} = \begin{bmatrix} \frac{\partial \boldsymbol{x}_s(T)}{\partial \boldsymbol{x}_s(0)} & \frac{\partial \boldsymbol{x}_s(T)}{\partial \boldsymbol{x}_a(0)} \\ \frac{\partial \boldsymbol{x}_a(T)}{\partial \boldsymbol{x}_s(0)} & \frac{\partial \boldsymbol{x}_a(T)}{\partial \boldsymbol{x}_a(0)} \end{bmatrix} \tag{3.25}$$

Therefore, by the definition of a quasi-algebraic component, every element of the second block column of the sensitivity matrix, $\frac{\partial \phi(\boldsymbol{x}_s(T), \boldsymbol{x}_a(T))}{\partial(\boldsymbol{x}_s(0), \boldsymbol{x}_a(0))}$ is nearly zero.

This implies that the standard envelope-following algorithm can then be applied to a subset of the circuit variables, using as the sensitivity matrix the block diagonal submatrix corresponding to the state variable,

$$\frac{\partial \boldsymbol{x}_s(T)}{\partial \boldsymbol{x}_s(0)}.$$

As the sensitivity matrix is updated every cycle, that

$$\frac{\partial \boldsymbol{x}_a(T)}{\partial \boldsymbol{x}_a(0)} \qquad \textit{and} \qquad \frac{\partial \boldsymbol{x}_s(T)}{\partial \boldsymbol{x}_a(0)}$$

remain small can be verified, and a decision can be made about which variables should be considered quasi-algebraic for subsequent cycle computations. This provides an automatic algorithm for determining quasi-algebraic components. However this is somewhat inefficient because variables which are consistently quasi-algebraic should not have their sensitivities computed at all. A simple improvement would be to recompute the full sensitivity matrix only every few cycles in order to check that any node currently in the set of quasi-algebraic variables remains in that set, and equivalently that every node considered as a state also remains a state. Experience seems to indicate that once a node is considered either a state or a quasi-algebraic variable, it is unlikely to be switched to the other set.

Summarizing, at the beginning of each computed cycle we have computed the sensitivity matrix and symbolically divided the system variables into state variables, $\boldsymbol{x}_s$, and quasi-algebraic variables, $\boldsymbol{x}_a$. We compute the quasi-algebraic variables from the state variables and then integrate for a cycle and use the newton iteration update to ensure that the state variables satisfy the alignment equation

$$\begin{aligned}&\boldsymbol{\phi}\left(\boldsymbol{x}_s((m+l-1)T), \boldsymbol{x}_a((m+l-1)T)\right) - \boldsymbol{x}_s(mT) = \\ &\qquad\qquad l\ \left[\boldsymbol{\phi}\left(\boldsymbol{x}_s((m+l-1)T), \boldsymbol{x}_a((m+l-1)T)\right) - \boldsymbol{x}_s((m+l-1)T)\right]\end{aligned}$$

where some of the arguments for $\boldsymbol{\phi}(\cdot)$ are omitted for brevity. In Algorithm 3.4.1, we give the complete modified envelope-following algorithm.

Note that in Algorithm 3.4.1, at the beginning of every cycle-step, the quasi-algebraic components, $\boldsymbol{x}_a$, are computed from the state components, $\boldsymbol{x}_s$. This can be done by ensuring that KCL is satisfied at the algebraic nodes. This corresponds to solving the circuit shown in Figure 3-18, where the state variables are replaced with voltage sources and the unknowns are the voltages at the nodes corresponding to the quasi-algebraic variables. In other words $\boldsymbol{x}_a$ is computed by solving

$$\boldsymbol{G}\left(\boldsymbol{x}_a((m+l-1)T), \boldsymbol{x}_s((m+l-1)T)\right) = 0. \qquad (3.26)$$

where $\boldsymbol{G}(\cdot)$ corresponds to the KCL equations for quasi-algebraic nodes.

Another approach to solving Eqn. (3.26) coupled with the alignment equation is to use a nonlinear Gauss-Seidel Relaxation loop such as shown in Algorithm 3.4.2. This nonlinear-relaxation algorithm converges quickly if

$$\frac{\partial \boldsymbol{\phi}\left(\boldsymbol{x}_s, \boldsymbol{x}_a, T\right)}{\partial \boldsymbol{x}_a} \approx 0.$$

Algorithm 3.4.1 (Nitswit Modified Envelope-Follower).

```
Divide $x$ into $x_s$ (States) and $x_a$ (Quasi-Algebraic) using the
  latest Sensitivity matrix.
$m = 0$

While $mT <$ STOPTIME {
   Select the cycle-step $l$
   Predict a first guess, $x_s^0((m+l-1)T)$
   Compute $x_a^0((m+l-1)T)$ from $x_s^0((m+l-1)T)$.
   Numerically integrate Eqn. (3.1) from $(m+l-1)T$ to $(m+l)T$ to
     compute $x^0((m+l)T)$ and $\frac{\partial x^0((m+l)T)}{\partial x((m+l-1)T)}$
   Compute $J_F(x^0((m+l-1)T)$ as in Eqn. (3.16)
   Redivide $x$ into $x_s$ (states) and $x_a$ (quasi-algebraic) using
     the newly computed Sensitivity matrix.
   Set k=0
   Until Newton Converges {
      Compute $x_a^{k+1}((m+l-1)T)$ from $x_s^{k+1}((m+l-1)T)$.
      Solve the Newton update equation for $x_s^{k+1}((m+l-1)T)$.
      Numerically integrate Eqn. (3.1) from $(m+l-1)T$ to $(m+l)T$
        to compute $x^{k+1}((m+l)T)$
   }
   $m = m + l$
}
```

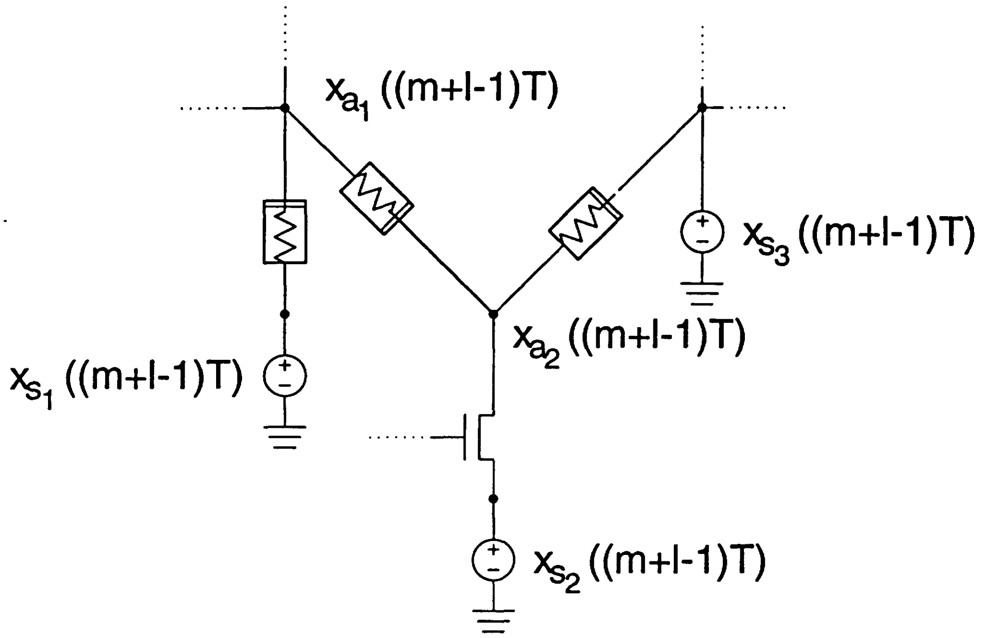

Figure 3-18: Computing the values of the quasi-algebraic variables, assuming the state variables have fixed values, and enforcing KCL at the algebraic nodes.

```
Algorithm 3.4.2 (Gauss-Seidel Nonlinear Relaxation Envelope-Following).

For $k = 1, 2, \ldots$

    Compute $\boldsymbol{x}_a^{k+1}((m+l-1)T)$ from

        $\boldsymbol{G}\left(\boldsymbol{x}_a^{k+1}((m+l-1)T), \boldsymbol{x}_s^{k}((m+l-1)T)\right) = 0.$

    Compute $\boldsymbol{x}_s^{k+1}((m+l-1)T)$ from the alignment equation

        $\boldsymbol{\phi}\left(\boldsymbol{x}_s((m+l-1)T), \boldsymbol{x}_a((m+l-1)T)\right) - \boldsymbol{x}_s(mT) =$
            $l\ \left[\boldsymbol{\phi}\left(\boldsymbol{x}_s((m+l-1)T), \boldsymbol{x}_a((m+l-1)T)\right) - \boldsymbol{x}_s((m+l-1)T)\right]$
```

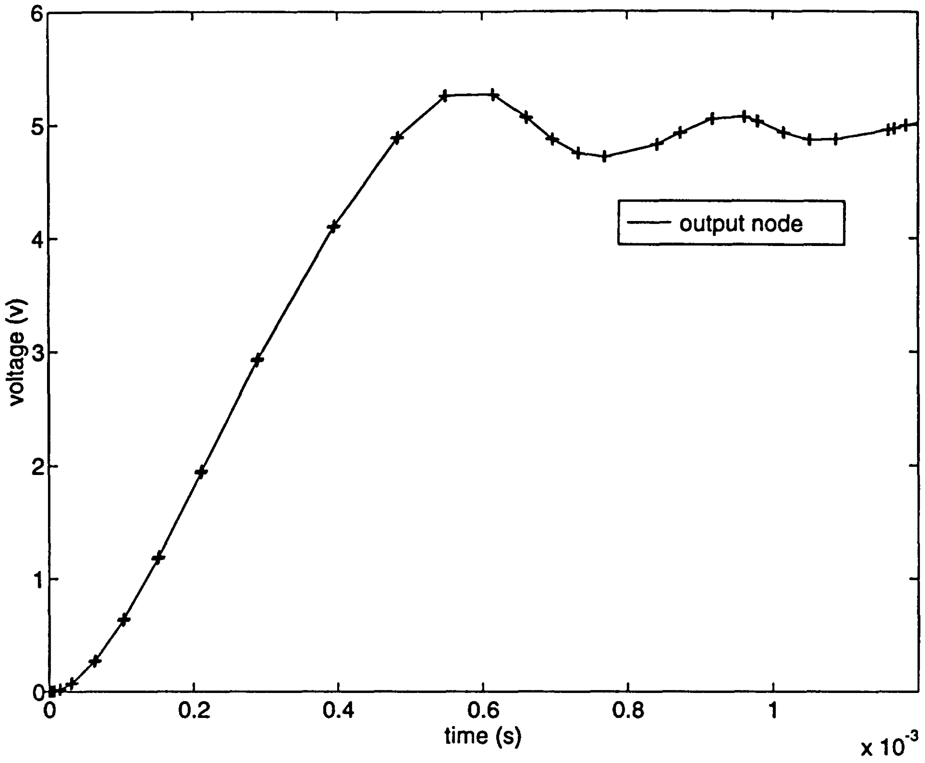

Figure 3-19: Closed-Loop buck converter output node during power-up transient obtained with the modified envelope-following technique.

Figure 3-19 shows the output of the closed-loop buck converter during a power-up transient as obtained with the modified envelope-following algorithm. As with the standard formulation, we can see that during the initial phase of the transient power-up the algorithm computes only occasional cycles (again shown as thicker sections). As the converter output approaches steady-state, only a few cycles are now being simulated, unlike the Standard Envelope-Following algorithm. The introduction of the concept of quasi-algebraic variables allowed us to isolate the variable which contain state information and envelope-follow only those, providing for a much more efficient simulation.

This modified envelope-following algorithm is an example of a model order reduction technique applied to circuit simulation. Basically the states associated with the fastest dynamics are being neglected in favor of the states that contain the low-pass frequency information, i.e., the envelope. The technique described presents us with a simple way to perform this automatic dynamic splitting of the circuit states, based on their relevant frequency contents.

3.4.2 Experimental Results

Both the standard and a modified version of the envelope-following method have been implemented in the NITSWIT simulation program. As mentioned, the program uses a trapezoidal-rule based envelope-following algorithm with local-truncation error cycle-step control. In Table 3-3, we compare the CPU times required to simulate the start-up transient from four different circuits by classical direct methods, the standard envelope-following algorithm and our modified algorithm. The circuits presented are: a resonant converter *quasi* [15], an open-loop buck converter circuits, *dbuck*, a closed loop converter, *closed* and a switched capacitor filter, *scop*. In each case, the clocking is provided by a user-defined source. As can be seen from the table, the envelope-following method can be very efficient, particularly when the simulation interval is long compared to the clock period. In particular, from the results presented it is clear that the standard envelope-following algorithm is very efficient when simulating open-loop circuits.

Circuit	N	Cycles	Classical	Std EF	Mod EF
quasi	7	200	188	69.4 (33)	16.5 (13)
scop	13	200	156	65 (30)	27.6 (15)
dbuck	4	1000	359	34.5 (50)	29.0 (48)
closed	5	600	79	47 (124)	10.8 (31)

Table 3-3: CPU time comparisons for Classical and Standard Envelope-Following Simulation versus Modified Envelope-Following. The number of cycles shown corresponds to the simulation interval divided by the clock period. For the envelope-following approaches, the number of effectively simulated cycles is also shown. Times are in seconds on a SUN4/260.

The results obtained when comparing envelope-following to classical methods for a closed-loop buck converter *closed* does not produce equally encouraging results. As discussed, the difficulty simulating the closed-loop converter is that it includes control circuitry which rapidly responds to small changes in the converter output. However, variables associated with the controller are quasi-algebraic, and therefore the modified algorithm performs substantially better. Note also that the results in Table 3-3 show that modified envelope-following is always faster than the standard envelope-following, due to the reduction in the number of computed cycles. Most noticeably, for the most difficult example, namely the closed loop converter, a speedup of a factor of over four is obtained over standard envelope-following, and this makes the modified envelope following algorithm almost eight times faster than the classical direct approach.

3.5 Autonomous Circuits and Multiple Frequency Circuits

In this section we will detail our efforts in extending the applicability of the envelope-following algorithm to handle circuits that contain either a clock of unknown frequency, or multiple clocks or frequencies. Each of these types of circuits are important on their own and each present new challenges in terms of efficient simulation. However, they share the important characteristics of most of the circuits that the envelope-following algorithm is already able to deal with, namely the fact that the period of their high-frequency clocks are typically orders of magnitude smaller than the time intervals of interest to a designer. It would seem therefore plausible that the envelope-following algorithm, possibly with slight modifications, would be able to simulate them in an efficient manner, similar to what was accomplished with the single frequency, externally clocked circuit studied in the previous sections.

3.5.1 Autonomous Circuits: Period Detection and Extraction

Autonomous circuits are circuits that contain internal clock generators, i.e. circuits for which the frequency of operation is generated internally instead of being supplied externally. Typical examples of such circuits are oscillators. The difficulty with simulating autonomous circuits in terms of applying the envelope-following algorithm directly, lies in the fact that the clock period is unknown a-priori, which makes it hard to determine the cycle boundaries and furthermore, the cycle length. Furthermore, some autonomous circuits may have clock frequencies that change with time, which implies that an adaptive detection method would have to be implemented. There has been some work done in this area and the results reported were extremely encouraging [16, 17]. In [17] heuristic methods are developed that allow one to detect and extract the period of a highly oscillatory differential equation. When applied in our setting, these involve the automatic and adaptive detection of cycle boundaries. Some preliminary studies that we have conducted have shown that it is possible to adapt the algorithm to work with such autonomous circuits. In fact we have tried a modified version of our algorithm with a simple voltage-controlled oscillator and were able to detect its period based on simple heuristic reasonings. This modified algorithm was based upon the detection of either changes in the sign of the derivative of certain nodes voltages or zero crossovers and use that information to guess at the clock period. These preliminary results indicate that a robust detection and adaptive method could be incorporated into the envelope-following framework. This is however beyond the scope of this thesis. One should note nonetheless, that for such circuits the efficiency of the envelope-following algorithms will expectedly be lower, given that initially some complete cycles will have to be simulated so that the detection algorithm can extract the circuit natural clock frequency. The situation will worsen if this natural clock frequency changes with time, as in oscillator start-up or voltage-controlled oscillators. In that situation the number of cycles to be skipped will depend upon the rate at which the frequency changes. Further research and work into devising a robust detection algorithm would certainly extend the importance and applicability of such algorithms.

3.5.2 Multiple Frequency Circuits

Phase-locked loops (PLL's) are becoming more commonly used in analog and digital applications. Phase-locked loops are extremely challenging to simulate because their capture transient may be extremely slow compared to PLL free-running period. Therefore, the existence of an algorithm that could efficiently and accurately simulate the behavior of a PLL while at the same time giving the designer the high degree of accuracy he expects from an electrical-level simulation would be very desirable.

In terms of simulation, PLL's share all the characteristics of the clocked analog circuits we dealt with in previous sections but they also present new problems. To better understand the source of difficulties that are involved, consider the simplified PLL model in Fig. 3-20. This circuit is composed of three main building blocks: a frequency/phase discriminator, a low-pass filter and a voltage controlled oscillator.

The output signal of a voltage-controlled oscillator (VCO) has a frequency, f_{vco}, which is proportional to the input signal voltage v_{out}. Every VCO has associated with it a range of voltages that it can accept as input and for which the corresponding output frequency changes proportionally. Changes in the input voltage outside this limits cause the VCO output frequency to saturate and the device will not behave as expected. Also associated with a VCO is what is called the central or natural frequency, which corresponds to a zero input. The phase detector compares two inputs and generates an output that is a measure of their phase difference. If v_{in} and v_{vco} are in phase, then $v_e = 0$, and f_{vco} will be equal to the center frequency. If on the other hand f_{in} does not equal f_{vco}, there will be a nonzero phase-error which, after being filtered and amplified, causes the VCO frequency to deviate in the direction of f_{in} in order to reduce the phase difference. The PLL structure suggests therefore that the relevant phase error must be contained in the low-frequency content of the phase-error signal v_e. With generality the functionality of the phase detector implies

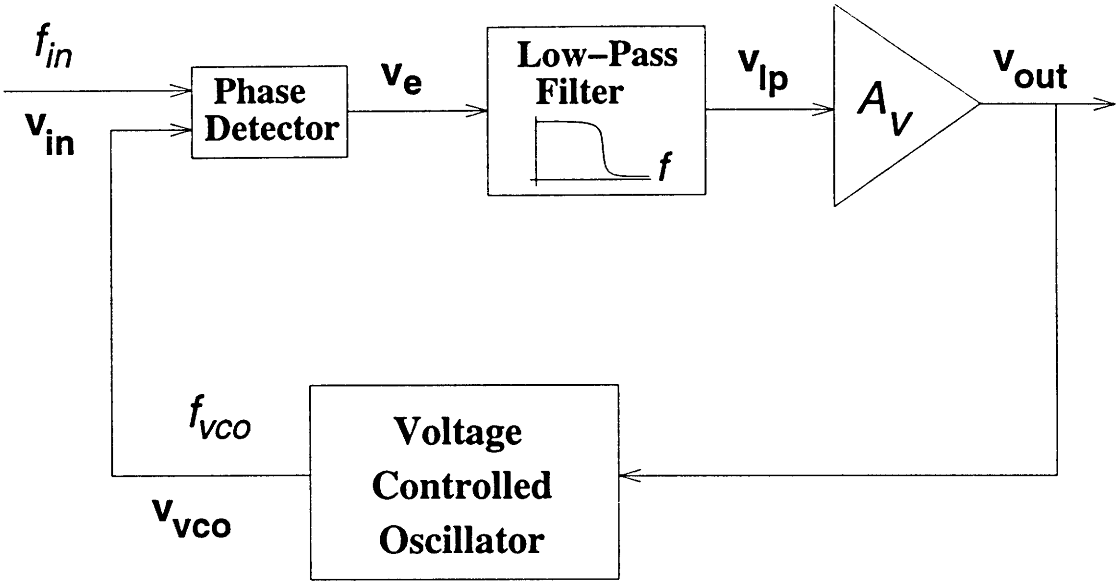

Figure 3-20: Phase-locked loop.

some sort of multiplicative action between its two inputs in order to obtain the phase difference as part of its low-frequency content. This multiplicative action implies that the the phase error signal signal v_e will have frequency components at $|f_{in} - f_{vco}|$ and $f_{in} + f_{vco}$. If the difference frequency is small, then the VCO is close to locking into the incoming frequency. The high-frequency component will get filtered out by the low-pass filter ahead and therefore the filtered signal, before or after amplification will contain only the low-frequency part of that error. If proper conditions are met, the VCO will eventually "lock" to f_{in}, meaning that its frequency f_{vco} will become such as to maintain a fixed phase relationship with the input signal. The process by which this happens is as follows: as the phase error signal brings the VCO frequency closer to the reference input frequency, the error waveform varies more slowly. So the error signal is asymmetric, varying more slowly over the part of the cycle during which the two frequencies are closer. The net result is a nonzero average, i.e. a DC component that brings the PLL into lock. That is, the filtered output of the phase detector will basically consist of a DC signal, and therefore the control input to the VCO will a measure of the input frequency. In other words, the VCO output is a locally generated, clean replica of f_{in}.

There are several fundamental characteristics associated with a PLL: its free-running or fundamental frequency, its lock-up time, its capture range and lock ranges, etc. Consider for instance a simulation run that would enable a designer to determine the lock range of a PLL, defined as the range of frequencies over which the loop will remain in lock. Typically in such a simulation the input frequency to the PLL would be either the same, if known, or very close to the PLL fundamental frequency, that is $f_{in} \approx f_{vco}$. Presumably the input frequency would then be varied slowly such that lock is not lost. If the input frequency wanders outside the range of frequencies that the VCO was designed to handle, then lock will be lost and the lock range can be effectively determined. In terms of the envelope following algorithm, this type of simulation presents only one problem which is the fact that the input frequency is known but not fixed. In fact in this particular type of simulation, the beat frequency $f_{in} - f_{vco}$ is by definition zero, since the loop is assumed to be in lock, and the phase error signal consists of a DC component plus a high-frequency signal which is filtered away by the low-pass filter. Therefore the question of which sampling frequency to use is immediately answered since assuming that the sum frequency gets filtered out, there really is only one frequency in the circuit. Therefore simulation of the lock range of a PLL is a process analogous to the simulation of an autonomous circuit with a varying frequency. Furthermore, in this case the frequency that is being varied is the input frequency and its rate of change will be known beforehand, which may used to help the simulator keep its synchronism. If the rate of change of the input frequency is small, then it is likely that the envelope-following algorithm will be able to perform efficiently and therefore we believe that determination of the lock range of a PLL can be accomplished by efficient and accurate simulation.

Consider now a dual situation in which one is interested in determining the PLL capture range, defined as the range of input frequencies for which the PLL is able to acquire lock, and the lock-up time defined as the transient time required for the lock to be established. To accomplish this goal one might consider a simulation run where a signal is input to the PLL such that its frequency is different from the loop's free-running frequency. Note that in general the capture range is smaller than the lock range because of the selectivity afforded by the loop filter. Still, a PLL should typically be able to lock onto frequencies that are within 20% of its free-running frequency, and therefore the capture range is in general a fairly large frequency range around the free-running frequency. Unfortunately in this case such a simulation will become a source of difficulty for an algorithm such as the envelope-following algorithm.

From the point of view of the envelope-following algorithm, PLL's as described above may violate many of the standard requirements of the algorithm. One problem is that the VCO output signal, v_{vco} has a frequency f_{vco} which is not fixed. However we have already mentioned that with slight modification in the algorithm, envelope-following can efficiently simulate such circuits. The main problem in fact lies elsewhere, and is related to the fact that in problems such as the determination of the capture range, the PLL is a multiple frequency circuit. We recall that the main reason for the efficiency of the envelope-following algorithm lies in its ability to exploit the property of clocked analog circuits that the node voltage waveforms over a given high frequency clock cycle are similar, but not exact duplicates, of the node voltages waveforms in proceeding or following cycles. This property is based on the fact that the envelope of the solution obtained by sampling the node voltage waveforms at multiples of some high-frequency clock is a smooth and slow function of time. This fact was essential is devising an algorithm that enabled us to construct a solution accurate over many high frequency clock cycles by computing the solution accurately for a few selected cycles. In the case of a circuit with multiple frequencies, the first difficulty that one is faced with is in fact how to define a cycle.

To understand this situation consider the functionality of the phase detector. As mentioned before, in order to obtain the phase error, the phase detector generates a signal v_e which has frequency components at $|f_{in} - f_{vco}|$ and $f_{in} + f_{vco}$. The high-frequency component will get filtered out by the low-pass filter ahead so we will discard it as it is not relevant for this analysis. The first question to address in this situation is which sampling frequency to use, that is how to define the duration of a cycle. The following periods/frequencies are observable at some of the circuit nodes:

$$\begin{aligned} T_{in} &= 1/f_{in} \\ T_{vco} &= 1/f_{vco} \\ T_s &= 1/(f_{in} + f_{vco}) \\ T_d &= 1/|f_{in} - f_{vco}|. \end{aligned}$$

The first of these choices would be the natural candidate, that is, to choose the period of the external excitation as the underlying frequency of the envelope to be computed. One of the reasons for making that choice is the fact that is the only known value of the set above. In the lock range problem the beat frequency $|f_{in} - f_{vco}|$ is essentially zero and therefore approximately constant if sampled at multiples of $T_{in} = T_{vco}$. So, in this situation using the input frequency in order to define the duration of a cycle seems an acceptable choice. However, in the capture range determination problem, this beat frequency may be as large as 20% of f_{in} which is by no means a smooth frequency in comparison with the high frequency input. In this case nodes v_e and v_{out} are not necessarily periodic with period T_{in} and it is likely that the alignment equation will at multiples of T_{in} fail for these nodes. Similar objections can be raised if any of the remaining frequencies are chosen. Consider for instance Figure 3-21 which could be obtained by simulating a PLL with a free-running frequency of $100KHz$ using a $110KHz$ input signal. Shown in the figure is the beat frequency signal $|f_{in} - f_{vco}| = 10KHZ$

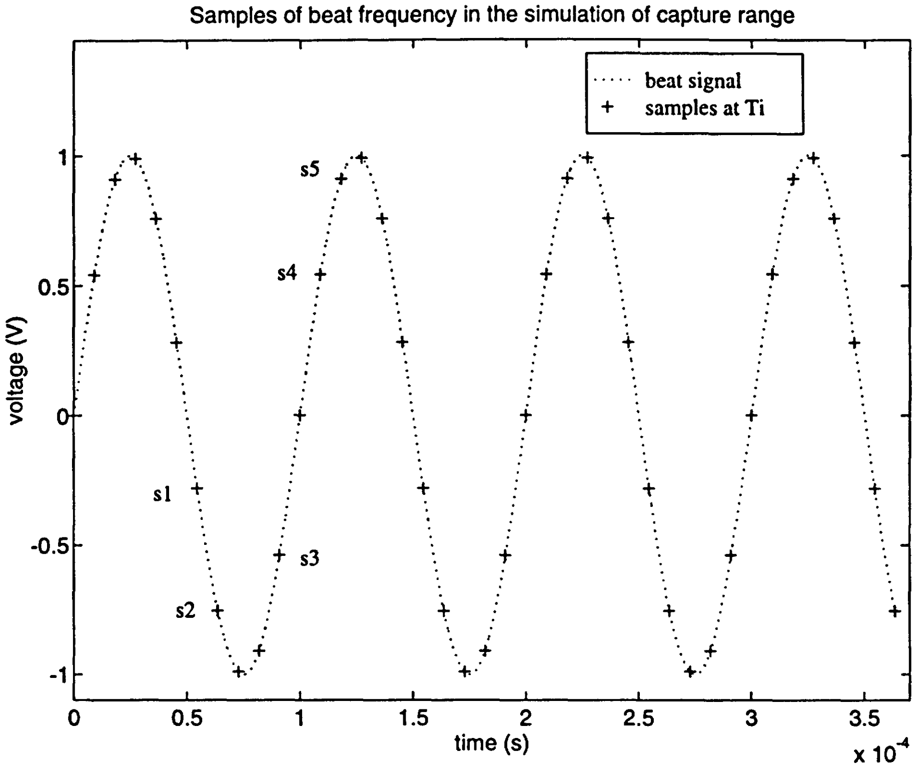

Figure 3-21: Signal at PLL beat frequency and its samples obtained using a clock frequency $f_{in} = 100KHz$.

and its samples obtained at intervals of $T_{in} = 10\mu s$. Clearly these samples do NOT constitute a smooth function. In fact consider for instance that the samples on the bottom left corner of the figure marked as "sl" and "s2" constitute the beginning and end of some high-frequency cycle. It is obvious from the figure that it will be impossible to align the beginning and end points of this cycle with those of *any* cycle in the future. On the other hand it might be possible to align samples "s3" with "s4" and "s5", but this avoids simulating just one or two cycles. Therefore applying the envelope-following algorithm to the simulation of this circuit will result in numerous failures of the Newton envelope iteration and, as a consequence, almost every other cycle will be simulated. This immediately translates into poor performance of the algorithm.

Considerable effort was spent in trying to redefine the envelope of every node in a circuit in terms of a single underlying period, but it does not seem likely that such method will meet with success. It quickly becomes clear that it would be impossible to impose an alignment on all variables using a single period, T. Using the modified envelope-following algorithm described in section 3.4 on a simple implementation of a PLL, it was possible to determine that very few nodes in the circuit are what we have called states. Namely, there is a state associated with the energy conserving element in the low-pass filter, and there is some state associated with the VCO. Every other node in the circuit can be considered a quasi-algebraic node for most purposes, because its value can be computed given the value of those few other states. Therefore, for the most part the problem did not reside in the determination of the quasi-algebraic constraint or in rapid changes occuring at those nodes. However, the both the vco output and the filter output nodes have voltage waveforms that over a given high frequency clock cycle are similar to the node voltages waveforms in proceeding or following cycles. The length of the cycle in each case is however very different. The VCO state node has most of its frequency contents around f_{vco}, while the low-pass filter state node has most of its energy around the slower beat frequency. And neither node will satisfy alignment equations if any other period T is chosen.

At this point it is unclear whether the envelope-following algorithm can be modified to handle circuits with multiple frequencies in a straightforward and efficient manner. This topic deserves further research which is nonetheless outside the scope of this thesis.

3.6 Conclusions

Simulating the transient behavior of Clocked Analog Circuits like switching power converters, switched-capacitor filters and phase-locked loops, is computationally expensive because these circuits are clocked at a frequency whose period is orders of magnitude smaller than the time interval of interest to the designer. We have seen that it is possible to reduce the simulation time without compromising much accuracy by exploiting the property that the behavior of switching converters in a given high-frequency clock cycle is similar, but not identical, to the behavior in the preceding and following cycles. In particular, the "envelope" of the high-frequency clock can be followed by accurately computing the circuit behavior over occasional cycles. Faster simulation can be achieved if the state variables of the circuit are isolated and those variables which contain no state information are not envelope-followed.

Several aspects of the modified method could be improved and further investigated. Of particular importance is finding faster techniques for updating the variables when leaping over some cycles. It has been verified that the extrapolation of the state variables when skipping cycles can be quite inaccurate. This is troublesome as it may lead the simulator to believe the circuit is in a totally different state. Ways of improving the extrapolation procedure are of great importance. For the quasi-algebraic variables, this is currently done by computing a DC solution with the state components held fixed, but other, more efficient possibilities may lead to more accurate results. Also, faster ways of identifying quasi-algebraic variables could lead to substantial speed improvements, particularly if they would circumvent the need to compute the full sensitivity matrix. Variables which are consistently quasi-algebraic should not have their sensitivities computed at all. A simple improvement would be to recompute the full sensitivity matrix only every few cycles in order to check that any node currently in the set of quasi-algebraic variables remains in that set, and equivalently that every node considered as a state also remains a state. Experience seems to indicate that once a node is considered either a state or a quasi-algebraic variable, it is unlikely to be switched to the other set. It has also been observed that most of the entries in the sensitivity matrix remain close to zero, and ways to exploit this should be considered. Faster and more efficient ways of computing and updating the sensitivity matrix would be welcome and deserve further study.

Modifying the method to be independent of the placement of cycle boundaries appears to be possible. This could provide for substantial improvement, as the effectiveness of the envelope-following is somewhat dependent on where the cycle boundaries are placed, and an automatic selection method is desirable. A simple technique to achieve the desired result would be to move the cycle boundary over, if it is found that large circuit activity is centered around the current boundaries. This can happen for instance if the current boundary coincides or is close to a quick change in the clock variable.

Finally, our difficulties in handling multiple frequency circuits should be addressed given that efficient simulators for such class of circuits are so much sought after. It is unclear at this point whether the Envelope-Following methods can be extended to handle circuits with more than one frequency.

References

[1] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Tech. Rep. ERL M520, Electronics Research Laboratory Report, University of California at Berkeley, Berkeley, California, May 1975.

[2] W. T. Weeks, A. J. Jimenez, G. W. Mahoney, D. Mehta, H. Quasemzadeh, and T. R. Scott, "Algorithms for ASTAP - A network analysis program," *IEEE Transactions on Circuit Theory*, pp. 628–634, November 1973.

[3] J. Kassakian, "Simulating power electronic systems – a new approach," *Proceedings of the IEEE*, vol. 67, October 1979.

[4] C. J. Hsiao, R. B. Ridley, H. Naitoh, and F. C. Lee, "Circuit-oriented discrete-time modeling and simulation for switching converters," *IEEE Power Electronics Specialists' Conf. Rec.*, 1987.

[5] S. C. Fang, *et al*, "SWITCAP: A switched-capacitor network analysis program – Part 1: Basic Features," *IEEE Circuits Syst. Mag.*, vol. 5, September 1983.

[6] S. Leeb, "Recognition of dynamic patterns in high frequency dc-dc switching converters," Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Sciences, February 1989.

[7] H. D. Man, J. Rabaey, G. Arnout, and J. Vandewalle, "Practical implementation of a general computer aided design technique for switched capacitor circuits," *IEEE Journal Solid-State Circuits*, vol. SC-15, pp. 190–200, April 1980.

[8] Y. P. Tsividis, "Analysis of switched capacitor networks," *IEEE Trans. Circuits Syst.*, vol. CAS-26, pp. 935–946, November 1979.

[9] L. Petzold, "An efficient numerical method for highly oscillatory ordinary differential equations," *SIAM J. Numer. Anal.*, vol. 18, June 1981.

[10] K. Kundert, J. White, and A. Sangiovanni-Vincentelli, "An envelope-following method for the efficient transient simulation of switching power and filter circuits," in *International Conference on Computer Aided-Design*, (Santa Clara, California), pp. 446–449, October 1988.

[11] J. White and S. Leeb, "An envelope-following approach to switching power converter simulation," *IEEE Trans. on Power Electronics*, vol. 6, April 1991.

[12] T. Kailath, *Linear Systems*. Information and System Science Series, Englewood Cliffs, New Jersey: Prentice-Hall, First ed., 1980.

[13] T. Aprille and T. Trick, "Steady-state analysis of nonlinear circuits with periodic inputs," *Proceedings of the IEEE*, January 1972.

[14] G. C. Verghese, M. E. Elbulukand, and J. G. Kassakian, "A general approach to sampled-data modeling for power electronic circuits," *IEEE Trans. on Power Electronics*, vol. PE-1, April 1986.

[15] L. F. Casey and M. F. Schlecht, "A high frequency, low volume, point-of-load power supply for distributed power systems," *IEEE Power Electronics Specialists' Conf. Rec.*, 1987.

[16] L. R. Petzold, *An Efficient Numerical Method for Highly Oscillatory Ordinary Diferential Equations*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Chapaign, August 1978.

[17] K. A. Gallivan, *An Algorithm for the Detection and Integration of Highly Oscillatory Ordinary Differential Equations using a Generalized Unified Modified Divided Difference Representation*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Chapaign, June 1993.

4

Frequency-Domain Model Order Reduction

4.1 Introduction

As a result of the ever-lasting trend for increasing system complexity, shrinking device sizes, and faster signal and clock speeds, interconnections are rapidly becoming the dominant factor in electronic circuit and system performance. In many of today's high performance digital electronic system designs, interconnect delay either dominates or is a non-negligible component of the system delay. Furthermore, previously ignored effects such as ringing, coupling and transmission line effects in interconnections have begun to affect signal integrity. These problems manifest themselves at the chip level as well as at the package, multi-chip module, board and backplane level. Magnetic interactions such as those produced by the dense three-dimensional packaging now commonly used in compact electronic systems will also interfere with system performance. Such effects are difficult to simulate because they occur only as a result of an interaction between the field distribution in a complicated geometry of conductors, and the circuitry connected to those conductors.

Irrespective of the design style or methodology, the combined electrical and magnetic effects of the packaging and interconnect and the nonlinear drivers and loads must be simulated during some stage of the verification process. More accurate modeling of packaging and interconnects results in large, linear circuit models that can dramatically slow down a circuit simulation. Thus, there exists a need for efficient and accurate simulation techniques to analyze circuits dominated by packaging and interconnect effects. This demand is particularly evident all the way from the extraction to the system-level verification of high-speed physical interconnect. It is believed that even for integrated circuits, interconnect behavior will present itself as the the major limitation on the performances of future high-speed, sub-micron technologies.

The fundamental difficulty encountered in integrating interconnect and packaging effects into a transient circuit simulator has to do with the fact that circuits containing nonlinear devices or time-dependent characteristics must be characterized in the time domain, while most of the important effects due to interconnect and packaging are best characterized in the frequency domain. In fact, stripline and microstrip printed circuit board traces, interchip connections on multi-chip modules, and coaxial cable connections all have nonidealities in their frequency response, many of which cannot be represented using a frequency-independent model. Therefore in order to accurately model the effects of packaging and interconnect in the signal integrity and performance, it becomes necessary to develop frequency dependent models that can be efficiently incorporated into circuit simulation programs.

The most straight-forward approach to including general frequency-domain models in a circuit simulator is to calculate the associated impulse response using for instance an inverse fast Fourier transform [1]. Then, the response of the device at any given time can be determined by convolving the impulse response with an excitation waveform. Such an approach is too computationally inefficient for use in general circuit simulation, as it requires that at every simulator timestep, the impulse response be convolved with the entire computed excitation waveform as discussed in Chapter 2. An alternative approach is to approximate the frequency-domain representation with a rational function, in which case the associated convolution can be accelerated using a recursive algorithm [2].

In this chapter algorithms are presented that allow efficient and accurate SPICE-level simulation of interconnect and packaging effects described by arbitrary frequency-domain representations, including measured and tabulated data. The emphasis is placed not only on efficiency but also on accuracy and generality. The approach to be presented is a combination of several reasonably well-known techniques that together form a robust algorithm that guarantees some fundamental numerical properties. The algorithms are based on a two step procedure: first, a decade-by-decade ℓ_2 minimization approach is used to construct a collection of forced stable rational functions whose sum, after a final global ℓ_2 minimization, approximates the original frequency-domain data. This algorithm is described in section 4.3. In many cases the resulting approximation, though extremely accurate, can be of too high an order for efficient computation. Therefore, as described in section 4.5, a second step is performed whereby the unnecessarily high-order model is reduced using a guaranteed stable model order reduction scheme based on balanced realizations [3, 4]. Finally an impulse response can easily be generated from this reduced-order model and efficiently incorporated into the circuit simulator SPICE using recursive convolution. In Section 4.7, we present results derived from the application of this algorithm to a number of circuits and packaging structures. The examples are meant to demonstrate the efficiency of the approach as well as its generality.

4.2 Background

Simulating the effects of interconnect and packaging on systems performance and signal integrity requires the modeling of frequency response nonidealities. In order to accurately model such effects it is necessary to develop algorithms that can handle devices described in the frequency domain. These algorithms must then be efficiently incorporated into circuit simulation programs.

In order to cope with this problem, four generic types of approaches have been proposed in the past. One is to use networks of linear elements and ideal elements that approximate the frequency response of the device in question [5]. While these models are suited to existing circuit simulator programs, their addition to the circuits implies that the amount of computations will increase because a large number of extra nodes and elements may be introduced [6]. Since the cost of simulation grows superlinearly with the number of nodes, this method can quickly become too expensive and is therefore not regarded as very general. Another type of approach involves the use of convolution techniques. Using inverse Fourier transformation of frequency parameters [1] or numerical inversion of the Laplace transform [7], or even explicit analytical expressions [6] an impulse response can be computed. If the device is described by tabulated or measured frequency domain data, a straight-forward approach to accomplish this task is to apply an inverse fast Fourier (FFT) transform and compute the associated impulse response [8]. Once the impulse response is known, the device's response at any given time can be determined by convolving that impulse response with an excitation waveform. Unfortunately for many applications such an approach becomes computationally too inefficient for use in general circuit simulation. The reason for this difficulty was already reviewed in Chapter 2 and is directly related to the fact that at every simulator timestep, the impulse response must be convolved with the entire computed excitation waveform. The computation required at the N^{th} time point is then proportional to N, that is $\mathcal{O}(N)$; therefore the convolutions at large values of N will become very time-consuming. Furthermore, the Fourier techniques require special attention in order to avoid aliasing effects which may affect the accuracy of the computed impulse response. Also, in many cases accuracy considerations alone make it necessary to use many frequency points which makes the transformations costly.

Two other approaches of note have been proposed which try to avoid the time-consuming convolution integrations. One is the state-based approach [9] which uses information about the internal states of a device at a given timepoint in order to solve for the next time point. The necessary voltages and currents are considered as the state variables and their values kept as the states of the system. Simple assumptions such as piecewise-linearity between sample points are made, which allow computation of the state with simple integrations in space, thus avoiding convolutions. However the efficiency of this approach is questionable in situations where sharp edges are present in the state's waveforms, since accurate simulation of these edges will require dense placing of sample points in the regions where the waveforms are fast-varying. This same problem is seen in implementation of the waveform relaxation based approach [5, 10]. The waveform relaxation based approach works in the frequency domain and uses the FFT to transform the results back and forth between the frequency and time domains. Hence, time-domain convolutions are avoided by performing frequency domain multiplications. Since this FFT takes into account the whole waveform, the amount of data involved in the FFT can clearly become overwhelming if fast-rising signals are involved. For that reason this approach has also met with limited success.

An alternative approach to the ones presented above is to approximate the frequency-domain representation with a rational function, hopefully of low order. The advantage of this approach has to do with the fact that convolution with the associated impulse response can be performed very efficiently. This will be discussed in section 4.6.

Rational function approximation is not without its shortcomings. However attractive in terms of computational efficiency, the approximations must also be accurate not only in the frequency domain but also in the time domain. Unfortunately accuracy in the frequency domain does not necessarily translate into accuracy in the time domain. A simple example will illustrate this point in dramatic fashion. Consider the following transfer function,

$$\boldsymbol{H}(s) = \frac{s + 1.1 \times 10^5}{(s + 10^5)(s + 2 \times 10^5)}. \tag{4.1}$$

For the sake of argument let us assume that an approximation to this function is computed as

$$\boldsymbol{G}(s) = \frac{s - 1.1 \times 10^5}{(s - 10^5)(s + 2 \times 10^5)}. \tag{4.2}$$

The Bode plots for both the function and its approximation are shown in Fig. 4-1.

Other than a small phase error in the vicinity of $w \approx 10^5 \jmath$, this frequency domain

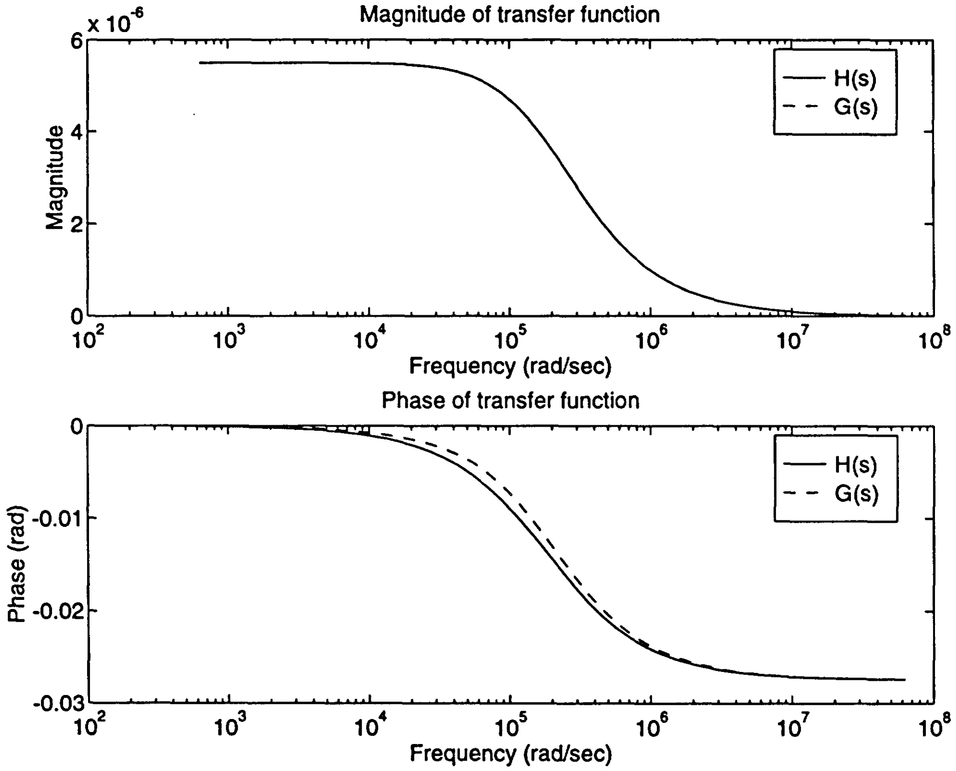

Figure 4-1: Bode plots for the function $\boldsymbol{H}(s) = \frac{s+1.1\times10^5}{(s+10^5)(s+2\times10^5)}$ and its approximant $\boldsymbol{G}(s) = \frac{s-1.1\times10^5}{(s-10^5)(s+2\times10^5)}$.

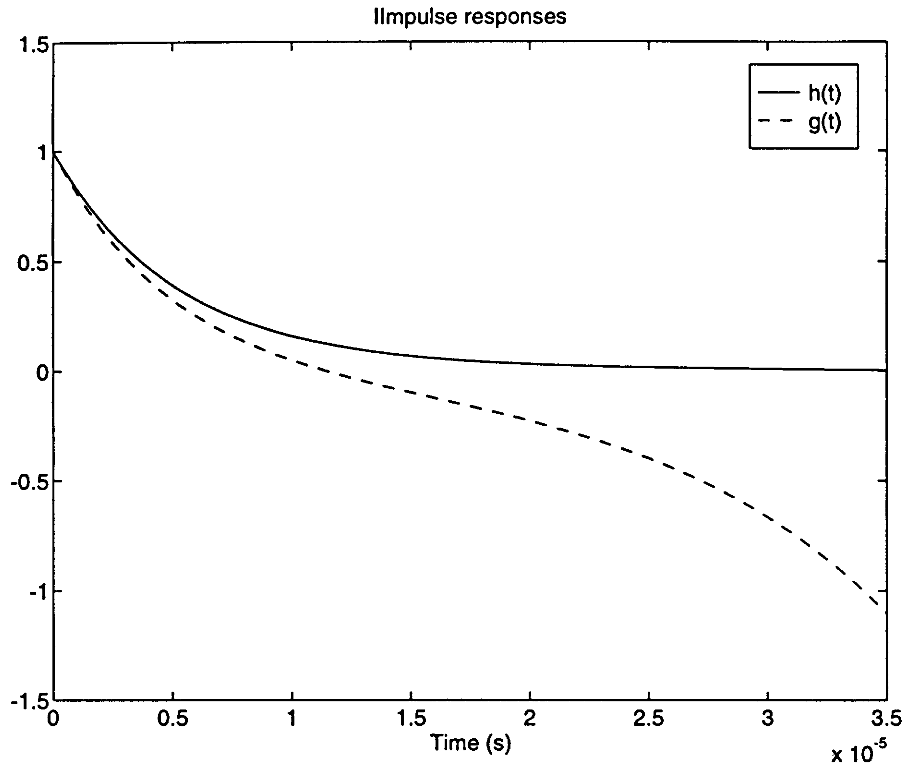

Figure 4-2: Impulse responses for the transfer functions $\boldsymbol{H}(s) = \frac{s+1.1\times10^5}{(s+10^5)(s+2\times10^5)}$ and its approximant $\boldsymbol{G}(s) = \frac{s-1.1\times10^5}{(s-10^5)(s+2\times10^5)}$.

approximation would probably be considered sufficiently accurate. At first sight, it may seem strange that a second order function is being approximated by a function of the same order. Note however that the function to be approximated might be given by tabulated data, in which case the exact form is not known. Either way, the example demonstrates that accuracy in the frequency-domain does not necessarily translate to accuracy in the time-domain. This is easily seen by computing the associated impulse responses, which in this case are:

$$\begin{array}{rlll} \text{exact:} & h(t) = k_1\, e^{-10^5 t} & + & k_2\, e^{-2\times10^5 t} \\ \text{approximation} & g(t) = k_3\, e^{10^5 t} & + & k_4\, e^{-2\times10^5 t} \end{array} \tag{4.3}$$

where the constants k_1, k_2, k_3 and k_4 are a function of the respective poles and zeros and are relatively unimportant for our discussion. The impulse responses for both the function and its approximation are shown in Fig. 4-2.

The term $k_3\, e^{10^5 t}$ in $g(t)$ grows with t, and therefore

$$\lim_{t\to\infty} g(t) = \infty \tag{4.4}$$

whereas

$$\lim_{t\to\infty} h(t) = 0. \tag{4.5}$$

Eventually this term will dominate the response which implies that for large t the error will increase very quickly. Clearly this demonstrates that even though the approximation is very accurate in the frequency domain, it is totally unacceptable as an approximation in the time-domain. We should note however that while accuracy in the frequency domain does not translate necessarily into accuracy or even stability in the time-domain, it is true that if the frequency-domain approximation has all its poles in the left half-plane, then the corresponding impulse response will not only be stable but will also approach the exact impulse response whenever the frequency-domain error tends to zero, that is:

$$\|\boldsymbol{F}(s) - \boldsymbol{H}(s)\| \to 0 \qquad \Longrightarrow \qquad |f(t) - h(t)| \to 0. \tag{4.6}$$

The downside of this result is that it being a limiting result, provides us with no time-domain error estimation even in the face of a given frequency-domain error. The upside, however is that it indicates that if the frequency error is small *and* the approximation is stable, it is likely that the time-domain error will also be small, and moreover decreasing the frequency-domain error will also decrease the time-domain error.

Unfortunately many of the commonly used techniques, have no guarantee of stability and in practice do generate unstable approximations. Due to their simplicity they are nonetheless widely popular. In the following section we review some of these techniques.

4.2.1 Padé Approximations

Moment-matching techniques have become widely used and accepted for the modeling and simulation of VLSI interconnections and packaging [11]. These techniques, known and studied for some time [12, 13, 14, 15, 16] were made popular in recent years with the appearance of Asymptotic Waveform Relaxation (AWE) [17], and the family of simulators derived from it. Moment-matching techniques are known to be a form of Padé approximation.

Padé approximations are approximations to complex function by the usage of general rational fractions expansions. In Padé approximations, a given function, $\boldsymbol{F}(s)$ is approximated by a low-order rational fraction expansion such as

$$\boldsymbol{H}(s) = \frac{U(s)}{V(s)} = \frac{u_p s^p + \cdots + u_1 s + u_0}{s^q + v_{q-1}s^{q-1} + \cdots + v_1 s + v_0} \tag{4.7}$$

by matching the various order Taylor series coefficients of that function. The reason why this procedure is also known as moment-matching can by seen by noting that by definition, we have

$$\begin{aligned} \boldsymbol{F}(s) &= \int_0^\infty f(t)e^{-st}dt = \\ &= \int_0^\infty f(t)\left[1 - st + \frac{1}{2}s^2t^2 - \frac{1}{6}s^3t^3 + \cdots\right]dt = \\ &= \int_0^\infty f(t)dt - s\int_0^\infty tf(t)dt + \frac{1}{2}s^2\int_0^\infty t^2 f(t)dt - \cdots . \end{aligned} \tag{4.8}$$

The coefficients multiplying the various powers of s in Eqn. (4.9) are analogous to the moments of a probability density function. They are referred to as the time-moments of the approximation. Moment-matching approximation is therefore equivalent to using our rational fraction expansion $\boldsymbol{H}(s)$ to match the values of the function $\boldsymbol{F}(s)$ and its derivatives at $s = 0$, that is:

$$\frac{u_p s^p + \cdots + u_1 s + u_0}{s^q + v_{q-1}s^{q-1} + \cdots + v_1 s + v_0} = m_0 + m_1 s + m_2 s^2 + \cdots + m_{p+q}s^{p+q} \tag{4.9}$$

where

$$m_0 = \boldsymbol{F}(s)|_{s=0} = \boldsymbol{F}(0) \qquad \text{and} \qquad m_j = \frac{d^j}{ds^j}\boldsymbol{F}(s)\bigg|_{s=0} . \tag{4.10}$$

As such, Padé approximations are clearly local approximations capturing the behavior of $\boldsymbol{F}(s)$ around $s = 0$.

It is possible to obtain the moments directly from a state realization of the system by simple solution of a number of systems with the same matrix and different right-hand sides and a matrix vector product. That is, assuming that $[\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}]$ is the state space representation of $\boldsymbol{F}(s)$, that is $\boldsymbol{F}(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D}$, then the moments are defined by as the coefficients of the following expansion, where the direct term $\boldsymbol{D}$ is neglected because it can always be computed separately:

$$\begin{aligned} \boldsymbol{F}_0(s) &= -\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B} - \boldsymbol{C}\boldsymbol{A}^{-2}\boldsymbol{B}s - \cdots \\ &= \sum_{j=0}^{\infty} m_j s^j \end{aligned} \tag{4.11}$$

where

$$m_j = -\boldsymbol{C}\boldsymbol{A}^{-j-1}\boldsymbol{B}, \qquad j \geq 0. \tag{4.12}$$

Once the moments are known, enforcing Eqn. (4.9) leads, for the case $p = q - 1$, to the following system of equations:

$$\left\{\begin{array}{lcl} u_0 & = & m_0 v_0 \\ u_1 & = & m_1 v_0 + m_0 v_1 \\ u_2 & = & m_2 v_0 + m_1 v_1 + m_0 v_2 \\ \cdots & \cdots & \cdots \\ u_{q-1} & = & m_{q-1} v_0 + m_{q-2} v_1 + \cdots m_1 v_{q-2} + m_0 v_{q-1} \\ 0 & = & m_q v_0 + m_{q-1} v_1 + \cdots + m_2 v_{q-2} + m_1 v_{q-1} + m_0 \\ 0 & = & m_{q+1} v_0 + m_q v_1 + \cdots + m_3 v_{q-2} + m_2 v_{q-1} + m_1 \\ \cdots & \cdots & \cdots \\ 0 & = & m_{2q-1} v_0 + m_{2q-2} v_1 + \cdots + m_{q+1} v_{q-2} + m_q v_{q-1} + m_{q-1} \end{array}\right. \tag{4.13}$$

The standard way of obtaining the coefficients of the approximation is to solve the last q equations of (4.13) to obtain the denominator coefficients and then use these values to obtain the numerator coefficients from the first q equations of (4.13). Rewriting the last q equations in matrix form leads to the following Hankel matrix:

$$\begin{bmatrix} m_1 & m_2 & m_3 & \cdots & m_q \\ m_2 & \iddots & & \iddots & \vdots \\ m_3 & & \iddots & & \vdots \\ \vdots & \iddots & & & m_{2q-2} \\ m_q & \cdots & \cdots & m_{2q-2} & m_{2q-1} \end{bmatrix} \begin{bmatrix} v_{q-1} \\ v_{q-2} \\ \vdots \\ v_1 \\ v_0 \end{bmatrix} = - \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{q-2} \\ m_{q-1} \end{bmatrix} \tag{4.14}$$

Once the coefficients of the denominator polynomial are obtained by solving Eqn. (4.14) a partial fraction expansion is computed leading to:

$$\boldsymbol{H}(s) = \frac{\boldsymbol{U}(s)}{\boldsymbol{V}(s)} = \frac{u_{q-1}s^{q-1} + \cdots + u_1 s + u_0}{s^q + v_{q-1}s^{q-1} + \cdots + v_1 s + v_0} = \sum_{i=1}^{q} \frac{r_i}{s - p_i} \tag{4.15}$$

where p_i, the poles of the approximation, are the roots of the denominator polynomial. The residues, r_i are computed directly by noting that the moments of the Padé approximations (4.15) match the first $2p - 1$ moments of the function $\boldsymbol{F}(s)$. Using Eqn. (4.10) to compute the moments of the approximation, we get that:

$$\left.\frac{d^j}{ds^j}\boldsymbol{H}(s)\right|_{s=0} = \sum_{i=1}^{q} r_i \left.\frac{1}{(s - p_i)^j}\right|_{s=0} = m_j \tag{4.16}$$

Matching the first q moments leads to

$$\sum_{i=1}^{q} r_i \, p_i^{-(j+1)} = -m_j \qquad\qquad j \in [0, q-1], \tag{4.17}$$

which in matrix form looks like:

$$\begin{bmatrix} p_1^{-1} & p_2^{-1} & \cdots & p_q^{-1} \\ p_1^{-2} & p_2^{-2} & \cdots & p_q^{-2} \\ \cdots & \cdots & \cdots & \cdots \\ p_1^{-q} & p_2^{-q} & \cdots & p_q^{-1} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_q \end{bmatrix} = - \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_q \end{bmatrix} \tag{4.18}$$

and therefore the residues can be obtained by inverting a $q \times q$ Vandermonde matrix.

Algorithm 4.2.1 summarizes the operations performed to obtain a Padé approximation using the standard moment-matching procedure.

In recent years the convergence of rational interpolation with free poles, that is poles determined by interpolation conditions, has received much attention [18, 19, 20, 21, 22]. One of the most vexing features of such interpolation is the appearance of spurious poles, that is, some poles of the interpolating rational function need not reflect the analytical behavior of the approximated functions [23]. It has been shown that in the presence of branch points, most poles have an asymptotic behavior determined by the approximated function [24, 25]. However, in many cases such results are not possible and spurious unstable poles do appear [26].

The moment-matching techniques that have become very popular recently, while computationally very efficient and accurate for low-order expansions, suffer from stability problems whenever high-order approximations are attempted. Consider for instance the circuit in Figure 4-3 which can regarded as a uniform lumped model for an interconnect segment. The state-space representation of this system can be constructed directly from inspection of the circuit. Once a state-space representation is obtained the moments of the system can be computed and a Padé approximation of any order can easily derived, using Algorithm 4.2.1.

Figure 4-4 compares the exact step response of this circuit with that obtained with Padé approximations of order 2 and 8.

As can be seen from Figure 4-4 the Padé approximation of 2^{nd} order is stable but not very accurate. Increasing the order of the approximation will, in many cases, increase the accuracy, but for certain orders, unstable poles may appear, and instability may develop as is the case with the 8^{th} order approximation shown. Unfortunately there is no way to know a-priori which approximation orders will lead to stable solutions. Furthermore there is also no a-priori measure of accuracy for the approximation at any order.

For systems that are very large with on the order of thousands of nodes the simplicity of such a method is hard to ignore and in such cases most people are willing to trade some of that instability for the extra effort necessary to in some sense "stabilize" the

Algorithm 4.2.1 (Moment-Matching Approximations).

```
pade($\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, q$)
{
    for ($i = 1; i <= q; i++$)
        compute the i$^{th}$ moment as $m_i = -\boldsymbol{C}\boldsymbol{A}^{-i-1}\boldsymbol{B}$

    compute the coefficients of the denominator polynomial by
      solving

      $\begin{bmatrix} m_1 & m_2 & m_3 & \cdots & m_q \\ m_2 & \iddots & & \iddots & \vdots \\ m_3 & & \iddots & & \vdots \\ \vdots & \iddots & & & m_{2q-2} \\ m_q & \cdots & \cdots & m_{2q-2} & m_{2q-1} \end{bmatrix} \begin{bmatrix} v_{q-1} \\ v_{q-2} \\ \vdots \\ v_1 \\ v_0 \end{bmatrix} = - \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{q-2} \\ m_{q-1} \end{bmatrix}$

    compute the roots $p_i, i = 1, \cdots, q$ of the denominator polynomial

      $s^q + v_{q-1}s^{q-1} + \cdots + v_1 s + v_0 = 0$

    compute the residues of the partial fraction expansion by
      solving

      $\begin{bmatrix} p_1^{-1} & p_2^{-1} & \cdots & p_q^{-1} \\ p_1^{-2} & p_2^{-2} & \cdots & p_q^{-2} \\ \cdots & \cdots & \cdots & \cdots \\ p_1^{-q} & p_2^{-q} & \cdots & p_q^{-q} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_q \end{bmatrix} = - \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_q \end{bmatrix}$

    return the Padé approximation

      $\boldsymbol{H}(s) = \sum_{i=1}^{q} \frac{r_i}{s - p_i}$

}
```

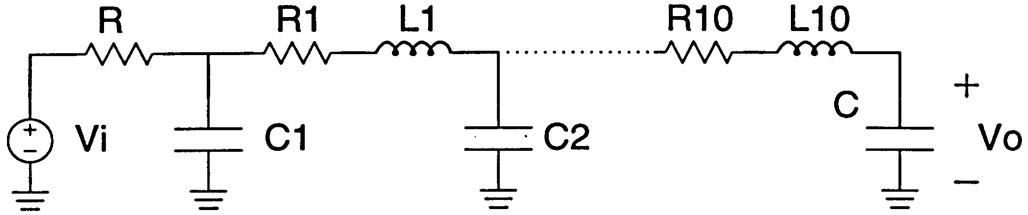

Figure 4-3: Uniform lumped model of interconnection segment; component values are: $R_j = 0.02\Omega, C_j = 0.21pF, L_j = 1nH, R = 25\Omega, C = 0.71pF$.

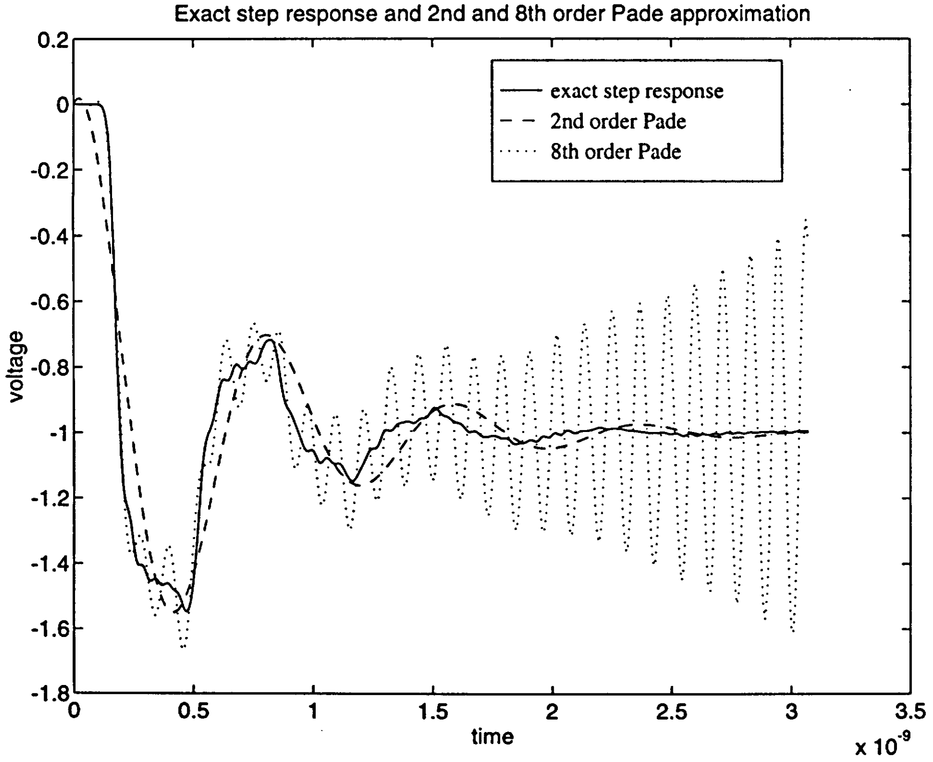

Figure 4-4: Comparing the exact step response with those obtained using respectively a 2^{nd} and an 8^{th} order Padé approximation.

approximation. Therefore, at least for large systems, Padé-type approximations are the most popular choice and are considered the state of the art. It is only for smaller circuits that it becomes interesting whether other techniques, possibly less computationally efficient, exist that would get around the stability problem.

It is known that for many examples the widespread range of frequencies involved causes the moments to diverge very quickly thus leading to ill-conditioned Hankel matrices and to numerical instability in the coefficients computation. This is due to the recursive nature of the computation of the moments which involves powers of the system matrix $\boldsymbol{A}$. This problem was noticed early on, and as a remedy it was proposed to use scaling [27], that is to replace the original moments m_j by $\zeta^j m_j$, where ζ is a suitably chosen scaling factor. Unfortunately, while scaling reduces the ill-conditioning somewhat, it does not solve the problem, and for increasing values of q, numerical difficulties still occur. Is has also been conjectured that some pole-zero configurations cannot be modeled at certain moment-matching orders, which is thought to be caused by the effects of very high frequency poles that are not strongly stimulated, or due to the close presence of masking high-frequency zeros [28]. However the matter of the fact is that these approximations may in certain cases become inadequate and other approaches must be pursued. Mathematicians have also explored other types of alternatives, such as Padé-Chebyshev approximations [29], which are supposed to have better convergence properties, or Carathéodory-Fejér approximations [30], which are $\mathbb{H}_\infty$ approximation with important stability guarantees on the poles.

If one considers that moment-matching techniques are based exclusively on information obtained at $s = 0$ it is not surprising that these stability problems reveal themselves, and furthermore it is indeed remarkable that in many cases so much information can in fact be obtained from a single expansion. In general it is known that the larger the frequency range the more difficult it is to obtain stable and accurate approximations. However it is also known that the order of such approximations must be small for stability to be kept. The balance between these contradicting issues is difficult and at the moment there does not seem to be a systematic way to solve this problem. Interestingly, it has been noticed by many researchers, that even when unstable poles are obtained, the approximation minus these instabilities is generally still quite accurate. In certain cases even, the absolute values of these unstable poles is such that the magnitude error is kept extremely small, which has led to the conjecture that simply moving the unstable poles over to the right half plane leads to an accurate and stable approximation.

Based on these observations, it would seem necessary and desirable to look for approximations built on a set in the complex plane rather than just at a single point.

4.2.2 Shifted Padé Approximations

A natural extension to the moment-matching methods is to compute the coefficients of the Padé approximant using a collocation procedure. This corresponds to computing multiple expansions at various frequency points and matching the values of the function and maybe its derivatives at those points. This approach extends the line of thought that has led to the development of moment-matching methods, in that it tries to capture information regarding the behavior of the function by using expansions around several values of s. Such techniques have in fact been studied and recent results indicate that this approach can be very valuable [31].

By computing expansions around $s_1, s_2, \cdots, s_l$ one can obtain a set of poles which is likely to provide accurate information about the local behavior of the function we are trying to approximate. However questions still remain on how one effectively incorporates the information gathered from all these matching expansions into a single accurate low order approximation.

In [31] a rational fraction is matched to the values of $F(s)$ and its derivatives computed at $s_1, s_2, \cdots, s_l$, where the number of expansion points and the points themselves are determined automatically with some heuristic algorithm. At each point a shifted Padé approximation is computed and some of the dynamics of this approximation are kept, according to a criteria which involves an a-priori guess regarding the accuracy of the approximation. Preference is given also to poles which appear in multiple expansions thus indicating that they are poles of the function and not artifacts of the matching process. The algorithm uses as many expansions as necessary to obtain an approximation of the desired order [31]. A problem with this method is that if an approximation of low order is initially required, then the number of expansions attempted will be equally small. Since the information related by each expansion is local in essence, that means that the expansion points must be carefully chosen. Unfortunately, correct placement of the expansion points is a case-dependent procedure and any systematic way to choose the frequency points is not likely to be very general. In some cases important detailed information about some frequency range, may be overlooked thus leading one to neglect important frequency behavior.

A simple extension to this method would be to use more expansions than necessary. The basic idea is that one would like to use as many expansions as necessary in order to capture information regarding the overall behavior of $\boldsymbol{F}(s)$, the system transfer function, over the whole frequency range. Given our low-order rational function approximant and a significant set of frequency values $\{s_1, s_2, \cdots, s_m\}$, we would like to satisfy, as accurately as possible, the following set of equations:

$$\boldsymbol{H}(s_j) = \boldsymbol{F}(s_j) \qquad j = 1, 2, \cdots, m \tag{4.19}$$

where

$$\boldsymbol{H}(s) = \frac{\boldsymbol{U}(s)}{\boldsymbol{V}(s)} = \frac{u_p s^p + \cdots + u_1 s + u_0}{s^q + \cdots + v_1 s + v_0} \tag{4.20}$$

is the low-order approximation. This approach may in fact be necessary in the case that only measured or tabulated data is available. In that situation moment-matching methods are not applicable since it is not easy to obtain or compute the values of the various moments or the expansions at the various points.

Clearly, the system in (4.19) will be over-determined if the number of frequency points exceeds the number of unknown coefficients in the approximation (4.20), that is if $m > p+q+1$, which is the usual case. In that situation there will be, in general, no exact solution, and the best that we can expect is that the approximation error be minimal in some sense. That is, we can perform some minimization on the over-determined system that will make sure that our approximation minimizes the global error in some normed sense, instead of just being very accurate around $s = 0$. For instance we can force the 2-norm of the error to be minimized, that is, make sure that the coefficients of the polynomials $\boldsymbol{U}(s)$ and $\boldsymbol{V}(s)$ are chosen such that

$$\|\boldsymbol{H}(s) - \boldsymbol{F}(s)\|_2 = \left\| \frac{\boldsymbol{U}(s)}{\boldsymbol{V}(s)} - \boldsymbol{F}(s) \right\|_2 \tag{4.21}$$

is minimized for all $s \in \{s_1, s_2, \cdots, s_m\}$. In general, the solution of the nonlinear minimization problem in Eqn. (4.21) is too expensive, and instead the cost function is reformulated and the solution of the derived linear system

$$\min_{\boldsymbol{U},\boldsymbol{V}} \|\boldsymbol{U}(s_j) - \boldsymbol{V}(s_j)\boldsymbol{F}(s_j)\|_2 \qquad j = 1, \cdots, m. \tag{4.22}$$

is computed instead. Note that the solution to (4.22) is not equal to the solution of (4.21), but is instead a weighted ℓ_2 minimization.

This ℓ_2 solution of the over-determined system will make sure that our approximation minimizes the *global* error in the ℓ_2 sense, instead of just being very accurate around $s = 0$ or at any of the various expansion points. However in general that may be problematic given the intended application of our model. While the ℓ_2 minimization does indeed minimize the global error, it does not guarantee any specific bound on the error at any particular frequency. In many instances this could be unacceptable because the reduced-order approximation might be used as a model for interconnect or packaging delay inside a circuit simulator for time-domain computations. In that case the model should certainly be accurate along the whole frequency range but in particular one wants it to be very accurate for instance when computing the DC operating point and the steady state. A solution to this problem is to constrain the minimization problem. For instance, one can ensure that the steady-state value is satisfied by introducing the constraint that $\boldsymbol{U}(0) = \boldsymbol{V}(0)\boldsymbol{F}(0)$. Similar constraints can be imposed at high frequencies if necessary.

The solution of this constrained minimization will then allow us to obtain the coefficients of our low-order rational function approximation,

$$\begin{cases} \dfrac{\boldsymbol{U}(0)}{\boldsymbol{V}(0)} = \boldsymbol{F}(0) & \\ \min\limits_{\boldsymbol{U},\boldsymbol{V}} \|\boldsymbol{U}(s_j) - \boldsymbol{V}(s_j)\boldsymbol{F}(s_j)\|_2 & j = 1, \cdots, m \\ \lim\limits_{s\to\infty} \dfrac{\boldsymbol{U}(s)}{\boldsymbol{V}(s)} = \lim\limits_{s\to\infty} \boldsymbol{F}(s) & \end{cases} \tag{4.23}$$

This constrained linear $\boldsymbol{\ell}_2$-minimization has two major drawbacks, namely the large dynamic range of the numbers involved and the over-emphasizing of high-frequency errors. The dynamic range of the number in the equation presents a difficulty especially in the case when the natural frequencies of the problem span a wide range, as is usual in interconnect problems. In that situation this minimization can easily become extremely ill-conditioned. It is easy to see this by looking at the structure of the matrix one obtains from the minimization portion of (4.23), which can be written as:

$$\begin{bmatrix} s_1^p & \cdots & s_1 & 1 & -F_1 s_1^{q-1} & \cdots & -F_1 s_1 & -F_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \\ s_j^p & \cdots & s_j & 1 & -F_j s_j^{q-1} & \cdots & -F_j s_j & -F_j \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \\ s_m^p & \cdots & s_m & 1 & -F_m s_m^{q-1} & \cdots & -F_m s_m & -F_m \end{bmatrix} \begin{bmatrix} u_p \\ \vdots \\ u_1 \\ u_0 \\ v_{q-1} \\ \vdots \\ v_1 \\ v_0 \end{bmatrix} = \begin{bmatrix} F_1 s_1^q \\ \vdots \\ F_j s_j^q \\ \vdots \\ F_m s_m^q \end{bmatrix} \tag{4.24}$$

Each row of this matrix corresponds to computing $\boldsymbol{U}(s_j) - \boldsymbol{V}(s_j)\boldsymbol{F}(s_j)$ at some frequency value s_j. The matrix is therefore a transposed Vandermonde-like matrix in the sense that the entries along each row are simple powers of the corresponding frequency value. If the span of frequencies being considered is large, then the magnitude of the entries on those rows will be much larger than those in rows corresponding to low frequency values. Not only may this cause overflow in the computation, but in general it may lead to an ill-conditioned matrix.

Furthermore even if the conditioning of the matrix is tolerable, another important issue comes into place. Since the $\boldsymbol{\ell}_2$ minimization will try to minimize the sum of the squares of the errors, it is likely to weight more heavily those values of frequency that produce larger entries in the matrix because the sensitivity of the solution to those entries is larger in the $\boldsymbol{\ell}_2$ sense. To understand this problem consider the case $p = q - 1$, and recall that an $\boldsymbol{\ell}_2$ minimization attempts to minimize the sums of the squares of the error at each point, that is:

$$e = \sum_{i=1}^{m} \sqrt{e_1^2 + e_2^2 + \cdots + e_m^2}, \tag{4.25}$$

where

$$\begin{aligned} e_j &= \|\boldsymbol{U}(s_j) - \boldsymbol{V}(s_j)\boldsymbol{F}(s_j)\|_2 = \\ &= \left|u_{q-1}s_j^{q-1} + \cdots + u_1 s_j + u_0 - s_j^q F_j - \cdots + v_1 s_j F_j - v_0 F_j\right| = \\ &= \left|-s_j^q F_j + (u_{q-1} - v_{q-1}F_j)\, s_j^{q-1} + \cdots + (u_1 - v_1 F_j)\, s_j + (u_0 - v_0 F_j)\right| \end{aligned} \tag{4.26}$$

is the error for the j^{th} equation, corresponding to the frequency value s_j.

From Eqn. (4.27) one can immediately see that the sensitivity of the error for the j^{th} equation, e_j with respect to any coefficient is a polynomial in s_j. Hence, the contribution of an error at s_j to the global cost function is a polynomial in s_j. This implies that, for a high frequency value s_j, small changes in the values of the coefficients translate into large errors and e_j will be large. Therefore minimizing the total error requires that the error components e_j corresponding to higher frequencies be carefully minimized, while those corresponding to lower frequencies, which have less impact on the global error, will not deserve so much attention. The end result of this situation is that the high frequency content of our function tends to be weighted more heavily than the low frequency content and the resulting rational function approximant is usually inaccurate in the low frequency range.

It is possible to introduce constraints in the minimization process such as using a weighting function that minimizes the high-frequency predominance effect. For instance one could multiply both sides of 4.22 with a low-pass transfer function in an attempt to decrease the importance of the high-frequency values. Care must be taken to make sure that the high-frequency content is not completely ignored, otherwise the solution one obtains may end up being accurate only at low frequency. Although substantial research has been conducted on this problem by many researchers from different fields, such weighting techniques remain very much case-dependent, and there is still no systematic way for finding an appropriate weighting function. One should point out however that the weighting problems are significant only when the range of frequencies used is large.

4.3 Interpolation of Frequency-Domain Data

In this section we describe a sectioned approach to the problem of approximating the transfer function of a system by a forced stable rational function. With this approach, we replace the problem of directly computing a low order rational function that is an accurate approximation over a wide frequency range with that of repeatedly computing local approximations over narrower ranges. These local approximations can then be summed to create an accurate approximation over the whole frequency range. This approach avoids, or at least minimizes the ill-conditioning of the global approximation problem. If the data available is a table of values measured at certain frequencies, any global approach of the Padé kind will clearly be ineffective, as derivative information at $s = 0$ or at any other point is not known. The approach described in this section is similar in spirit to the moment methods generalization that uses multiple expansions around other values of s to gather more global information [31].

The observation that the computation of global approximants leads in general to numerical difficulties, seems to indicate that instead of computing a low order rational function expansion that is an accurate approximation over the whole range of frequencies one would fare better by limiting oneselve to a smaller range. On one hand we can expect that a Padé approximation of a function over a smaller range will likely be easier to obtain. Furthermore, if unstable poles are obtained from the approximation algorithm, it is likely that discarding them will not have a profound effect on the accuracy over that small range and even less of an effect over far away frequencies. In other words, it is possible that a very low-order approximation is accurate enough to capture the local behavior of $\boldsymbol{F}(s)$ without instability, numerical or otherwise, playing a significant role. There seems to be two obvious ways to compute these local approximations: either one could use a shifted moment-matching or one could use an $\boldsymbol{\ell}_2$ minimization over a set of frequencies in the small range in question. Since with generality the function we are trying to approximate may not be known in closed form and the information one may have about it may be just a set of measurements, we chose to concentrate on the minimization approach. For this case we note that the problems regarding the ill-conditioning of the $\boldsymbol{\ell}_2$ minimization or those derived from any undesirable weighting are minimal, given the small range of frequencies in question. The local or **sectioning** approach to be described leads in general to extremely simple and accurate local approximations but it also raises the additional question of how to incorporate all the local information resulting from the various expansions into a global approximant which is still desirably of low order. This question will be studied in later sections.

4.3.1 Computing Section-by-Section Approximants

In order to avoid the numerical ill-conditioning and the uneven frequency weighting, it is desirable to limit the frequency range for the ℓ_2 minimization. Computing a low-order local approximation has the added advantage that the orders of the polynomials in the rational function approximation may in general be chosen small without compromising the accuracy of the approximation for that small frequency range. Moreover, if unstable poles are obtained from the local minimization procedure it is likely that using some simple heuristic, such as simply discarding them, will not have a profound effect on the accuracy over the small range of frequencies involved. In other words, it is possible that a very low-order approximation is accurate enough to capture the local behavior of $\boldsymbol{F}(s)$ without instability, numerical or otherwise, playing a significant role.

The idea of computing local approximations leads to a sectioning algorithm in which only accurate local approximations are computed. The remaining problem is how to incorporate all the local information resulting from the various approximations into a global approximant.

Our proposed solution is to perform the local approximations in a repeated fashion using a collocation procedure. Initially, the frequency range of interest, $\boldsymbol{\Omega} = [\omega_{min}, \omega_{max}]$, is partitioned into small sections, $\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2, \cdots, \boldsymbol{\Omega}_M$, such that $\boldsymbol{\Omega} = \bigcup_{i=1}^{M} \boldsymbol{\Omega}_i$, where each $\boldsymbol{\Omega}_i = [\omega_{i1}, \omega_{im_i}]$ is a decade or two long. Then, starting with the lowest frequency range $\boldsymbol{\Omega}_1$, with frequency values $\boldsymbol{F}(\omega_{11}), \boldsymbol{F}(\omega_{12}), \cdots, \boldsymbol{F}(\omega_{1m_1})$, a constrained ℓ_2 minimization is performed and a local approximant is computed. Once the first local approximation, $\boldsymbol{L}_1(s)$, is obtained in the form of a collection of poles and their corresponding residues, it is examined and the stable poles are retained while the unstable ones are discarded, leaving us with a *forced stable* approximation, $\boldsymbol{H}_1(s)$. Since the fit at the lower frequencies has captured the low frequency dynamics, $\boldsymbol{F}(s) - \boldsymbol{H}_1(s)$ will contain primarily the higher-frequency error information and is then approximated. To this end, frequency values in the second section $\boldsymbol{\Omega}_2$ are approximated. The value of $\boldsymbol{H}_1(s)$ at every point $\omega_{21}, \omega_{22}, \cdots, \omega_{2m_2}$ is computed, subtracted from the corresponding values $\boldsymbol{F}(\omega_{21}), \boldsymbol{F}(\omega_{22}), \cdots, \boldsymbol{F}(\omega_{2m_2})$ and the resulting data is again then fit using a constrained weighted ℓ_2 minimization. This results in a new local approximant $\boldsymbol{L}_2(s)$, from which a stable approximation, $\boldsymbol{H}_2(s)$ can be obtained. $\boldsymbol{H}_2(s)$ is then a new approximant to

Algorithm 4.3.1 (Section-by-Section Approximations).

```
sectioned($\omega_{min}, \omega_{max}, \boldsymbol{F}$)
{
  partition the frequency range into sections $\boldsymbol{\Omega}_1, \cdots, \boldsymbol{\Omega}_M$
    with associated frequencies $\{\omega_{i1}, \cdots, \omega_{im_i}\}$, $i = 1, \cdots, M$, and
    function values
  for ($k = 1$; $k <= M$; $k++$) {
    if ($k$ > 1) {
      subtract previous approximants from exact data:
        $$\boldsymbol{F}_k(s_{kj}) = \boldsymbol{F}(s_{kj}) - \sum_{l=1}^{k-1} \boldsymbol{H}_l(s_{kj}) = \boldsymbol{F}(s_{kj}) - \boldsymbol{H}(s_{kj}),$$
      $j = 1, \cdots, m_k$,  $s_{kj} = j\omega_{kj}$
    } else {
      $\boldsymbol{F}_1(s_{1,j}) = \boldsymbol{F}(s_{1,j})$
    }
    compute local approximant at the $k$-th section, $\boldsymbol{L}_k(s)$ using
      the corrected data $\boldsymbol{F}_k(s_{i,j})$
    examine the approximation and keep the stable poles and
      residues of $\boldsymbol{L}_k(s)$ in $\boldsymbol{H}_k(s)$
    add the new stable approximation to the current global
      approximant $\boldsymbol{H}(s) \leftarrow \boldsymbol{H}(s) + \boldsymbol{H}_k(s)$
  }
  while keeping the locally computed dynamics, perform a final
    global constrained $\ell_2$ minimization over the whole frequency
    range to recompute the residues
}
```

$\boldsymbol{F}(s) - \boldsymbol{H}_1(s)$ on $\boldsymbol{\Omega}_1 \cup \boldsymbol{\Omega}_2$, and therefore $\boldsymbol{F}(s) \approx \boldsymbol{H}_1(s) + \boldsymbol{H}_2(s)$ on that frequency interval. The procedure is repeated until data in the last frequency section, $\boldsymbol{\Omega}_M$, is approximated. A simplified form of this sectioning algorithm is shown in pseudo-code form as Algorithm 4.3.1. Figure 4-5 tries to convey visually how the algorithm works.

When the procedure terminates we are left with a forced stable global approximation which consists of all the stable poles and their corresponding residues obtained in all the iterated local minimizations. We should point out that our iterative sectioning algorithm is aimed at computing approximations which match successively higher frequency ranges. However, while subtracting the already computed approximations from the exact data,

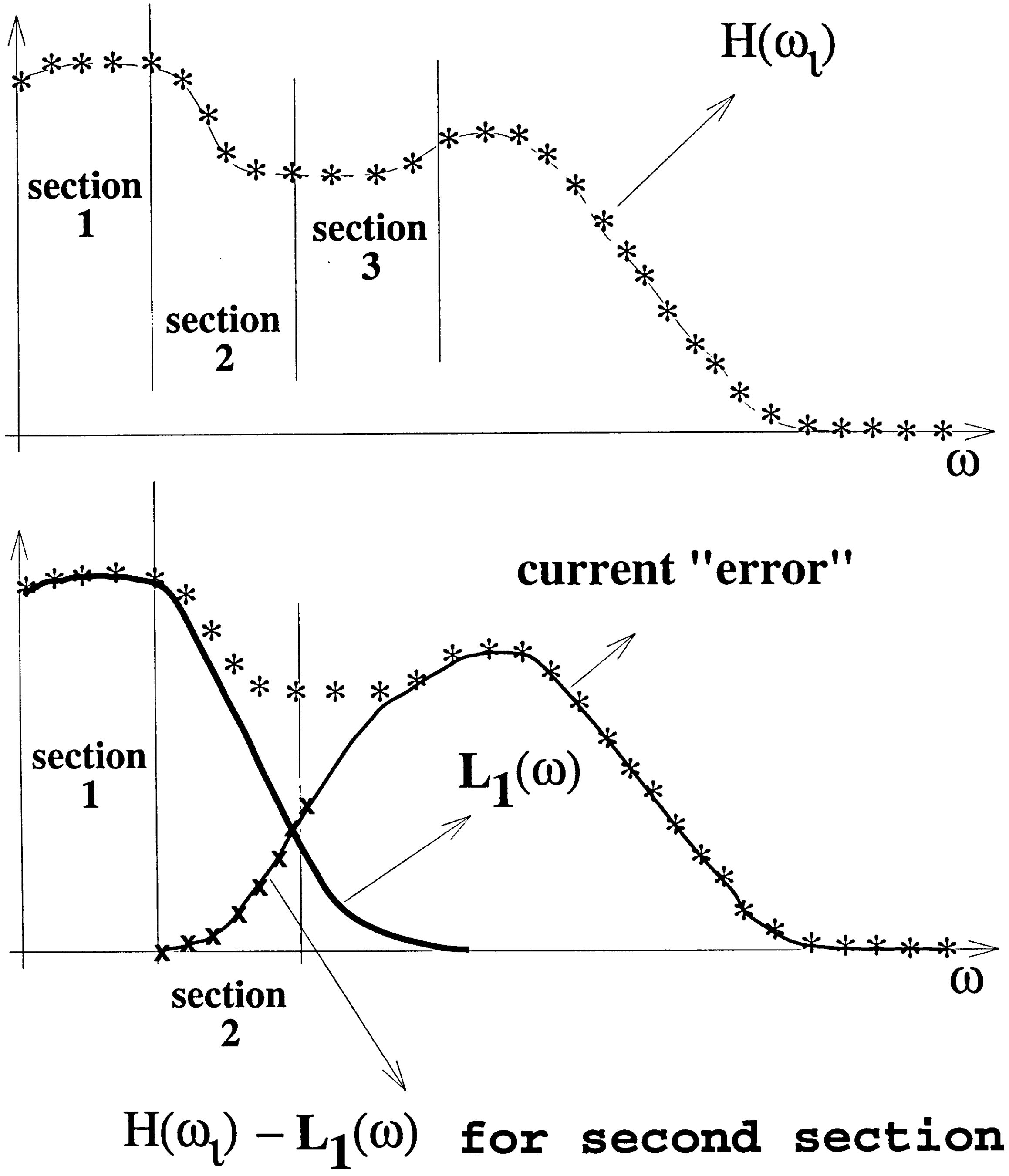

Figure 4-5: Applying the sectioning algorithm to measured or tabulated frequency data. The example tries to illustrate the sequence of operations that are performed to compute a global approximation, add it to the current global approximation and recompute the current error function.

some erroneous dynamics may be introduced at low frequency. To solve this problem, a final constrained global ℓ_2 minimization is performed in which the computed poles are used to recalculate their residues in order to match the exact data points. This final step does not in general suffer from the numerical problems mentioned in Section 4.2.1 regarding the global ℓ_2 minimization. In fact, the matrix one obtains in this case is better behaved because its (i, j) entries are of the form $(s_i - p_j)^{-1}$ and therefore the matrix is not necessarily ill-conditioned.

The algorithm just described reliably obtains a stable collection of pole-residue pairs which form an accurate approximation to $\boldsymbol{F}(s)$. Unfortunately since $\boldsymbol{H}(s)$ is represented as a sum of local approximations the approach introduces redundancies resulting in many more poles than necessary. With such a large number of terms, even fast recursive convolution, to be described in section 4.6, may prove to be inefficient. However it is possible to further reduce the order of the approximation using robust model order reduction techniques, which are described in section 4.5.

4.3.2 Section-by-Section Approximant: numerical example

In order to test the accuracy of the approximant obtained with our section-by-section algorithm, consider the example of a transmission line where skin effects are significant. The transmission lines equations were introduced as an example in section 2.5. The section-by-section algorithm was then applied to the frequency data obtained from the transmission line model after removing the ideal delay. These approximations have respectively 21 and 24 poles. In Figures 4-6 and 4-7 we compare the magnitude plots of the transfer functions of, respectively, the section-by-section approximations to $\boldsymbol{S}_{12}(j\omega)$ and $\boldsymbol{Y}_o(j\omega)$, with the transmission line data points for the same functions.

As one can see, the match is almost perfect, and the error is smaller than 0.5%. Moreover the low-frequency error is nearly zero.

4.4 State-Space Systems Realization

The frequency-domain data fitting method described in the previous section resulted in a stable transfer function $\boldsymbol{H}(s)$ with a large number of poles. Incorporating such a model (or equivalently its impulse response) directly in a circuit simulator will be computationally expensive. Instead, we will use a model order reduction approach which has three main steps. First, we find a well-conditioned state-space realization of the full frequency-domain model. Second, we use a state-space transformation to *balance* the

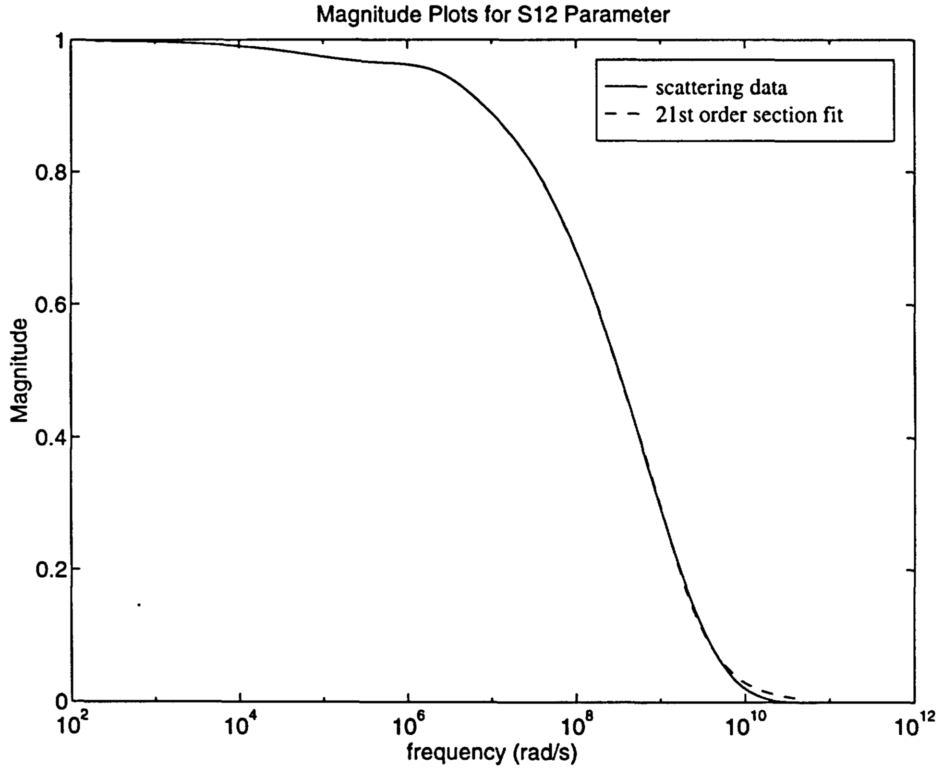

Figure 4-6: Accuracy of the section-by-section fit for the magnitude of the S_{12} transfer function with respect to the transmission line data points. The two curves are almost indistinguishable.

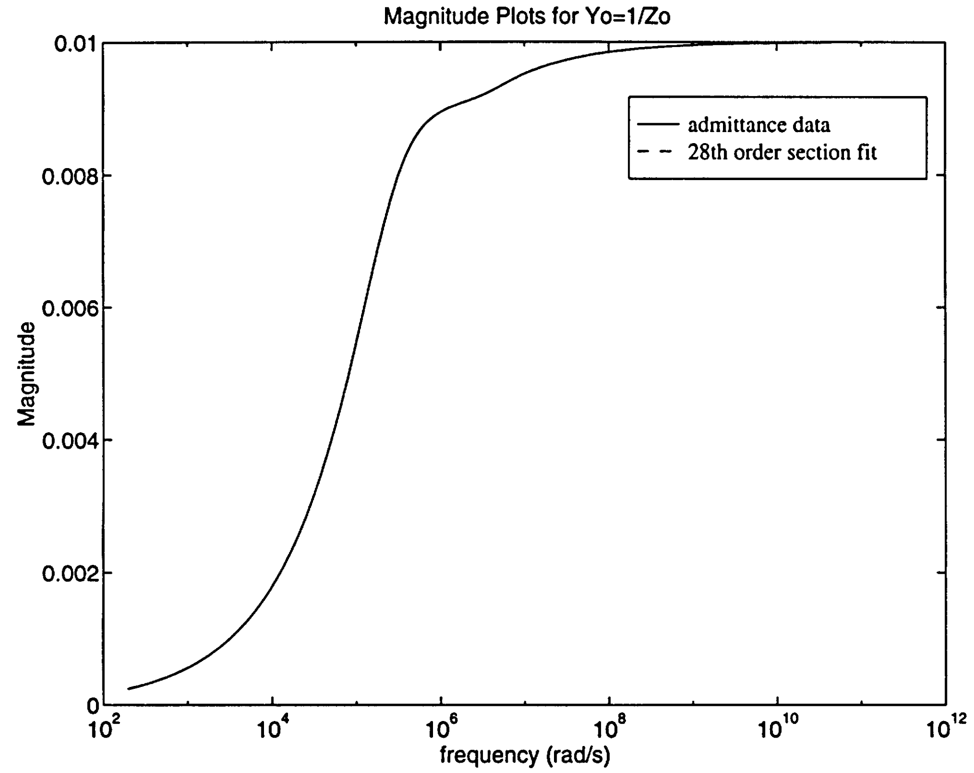

Figure 4-7: Accuracy of the section-by-section fit for the magnitude of the Y_o transfer function with respect to the transmission line data points. The two curves are almost indistinguishable.

state-space realization. Third, we truncate the balanced realization in a way consistent with the achievable accuracy at each reduced model order. In this section we discuss the issue of obtaining a well-conditioned state-space realization of the frequency-domain model.

The task of determining a state-state realization for a given transfer function is one that has been the target of extensive study and is now considered fairly standard. The transfer function that one is given can be in a pole-residue form, pole-zero form, or as a set of numerator and denominator polynomial coefficients. This can be written as

$$\boldsymbol{H}(s)=\sum_{i=1}^{n}\frac{r_i}{s-p_i}=K\frac{(s-z_i)\cdots(s-z_n)}{(s-p_1)\cdots(s-p_n)}=\frac{b_{n-1}s^{n-1}+\cdots+b_1s+b_0}{s^n+a_{n-1}s^{n-1}+\cdots+a_1s+a_0} \qquad (4.27)$$

whose state-space representation can be written as

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}u, && \boldsymbol{x}\in\mathbb{R}^n,\ u\in\mathbb{R},\ \boldsymbol{A}\in\mathbb{R}^{n\times n},\ \boldsymbol{B}\in\mathbb{R}^n \\ y &= \boldsymbol{C}\boldsymbol{x}, && y\in\mathbb{R},\ \boldsymbol{C}\in\mathbb{R}^n \end{aligned} \qquad (4.28)$$

such that $\boldsymbol{H}(s)=\boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}$.

Converting $\boldsymbol{H}(s)$ in a pole-residual form to a state-space form is a standard problem [32], and it is tempting to use one of the common techniques (canonical controllability realization, canonical observability realization, etc.) to find the matrices $\boldsymbol{A}, \boldsymbol{B}$, and $\boldsymbol{C}$. However, these approaches can result in a system matrix $\boldsymbol{A}$ which is poorly scaled and therefore unsuitable for computations.

An example will illustrate this problem. Consider the a controller canonical realization for $\boldsymbol{H}(s)$ which is shown in Fig. 4-8. For this canonical realization the system matrix has the following pattern

$$\boldsymbol{A}=\begin{bmatrix} -a_{n-1} & -a_{n-2} & \cdots & -a_1 & -a_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \qquad (4.29)$$

For medium to large n this matrix can quickly become ill-conditioned. To see this note from Eqn. (4.27) that $a_0=\prod_{i=1}^{n}p_i$. For systems with even a moderate number of high-frequency poles this may be beyond the largest representable number and even if that is not the case, the matrix thus formed is easily ill-conditioned. Similar situations

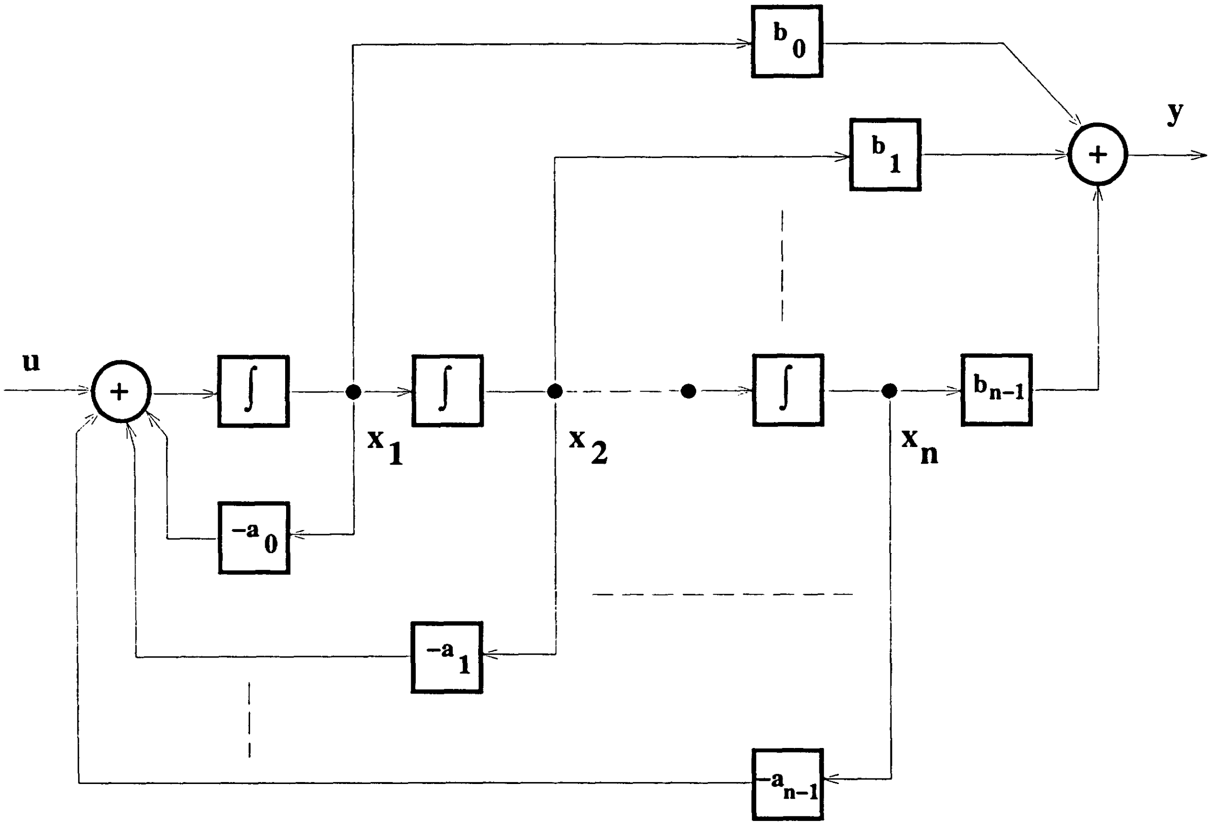

Figure 4-8: Controller Canonical form realization for $\boldsymbol{H}(s) = \frac{b_{n-1}s^{n-1}+\cdots+b_1s+b_0}{s^n+a_{n-1}s^{n-1}+\cdots+a_1s+a_0}$.

happen if other canonical realizations, such as observer, controllability or observability are used.

Instead of using a standard realization scheme, when all the poles are simple and real, the matrix $\boldsymbol{A}$ can be chosen equal to a diagonal matrix having the real poles as diagonal coefficients [32]. The control and observation matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ can then be chosen based on the residues of the poles. More explicitly, given

$$\boldsymbol{H}(s) = \sum_{i=1}^{n} \frac{r_i}{s - p_i} \tag{4.30}$$

where all the poles are negative reals and all the residues are real,

$$\begin{aligned} \boldsymbol{A} &= diag(p_1, \ldots, p_n) \\ \boldsymbol{B} &= (\sqrt{|r_1|}, \ldots, \sqrt{|r_n|})^T \\ \boldsymbol{C} &= (\mathrm{sign}(r_1)\sqrt{|r_1|}, \ldots, \mathrm{sign}(r_n)\sqrt{|r_n|}) \end{aligned} \tag{4.31}$$

This realization can be seen in Fig. 4-9.

When $\boldsymbol{H}(s)$ has pairs of complex conjugate poles, a block diagonal matrix $\boldsymbol{A}$ can be constructed where the blocks are all either of size 1×1, corresponding to real poles, or 2×2 corresponding to pairing the complex conjugate poles in state-space realizations of order 2. It is also possible to find suitable state-space realizations when some of the poles are repeated. However, if the circuit simulator one is using or writing can handle complex numbers representation and arithmetic, then the realization show in Eqn. (4.32) is absolutely general.

4.5 System Balancing and Model Order Reduction

In the previous section we described a numerically sound realization for the frequency-domain data fitting approximation obtained in section 4.3. It was pointed out at the time that this approximation may have a large number of poles. Incorporating such a model, or equivalently its impulse response, directly in a circuit simulator can become computationally expensive. A model order reduction approach is then sought which can provide a lower-order stable plant while maintaining the overall accuracy within acceptable bounds.

Given a linear state-space model for a multiple input multiple output (MIMO) system

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}, & \boldsymbol{x} \in \mathbb{R}^n,\ \boldsymbol{u} \in \mathbb{R}^m \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x}, & \boldsymbol{y} \in \mathbb{R}^p \end{aligned} \tag{4.32}$$

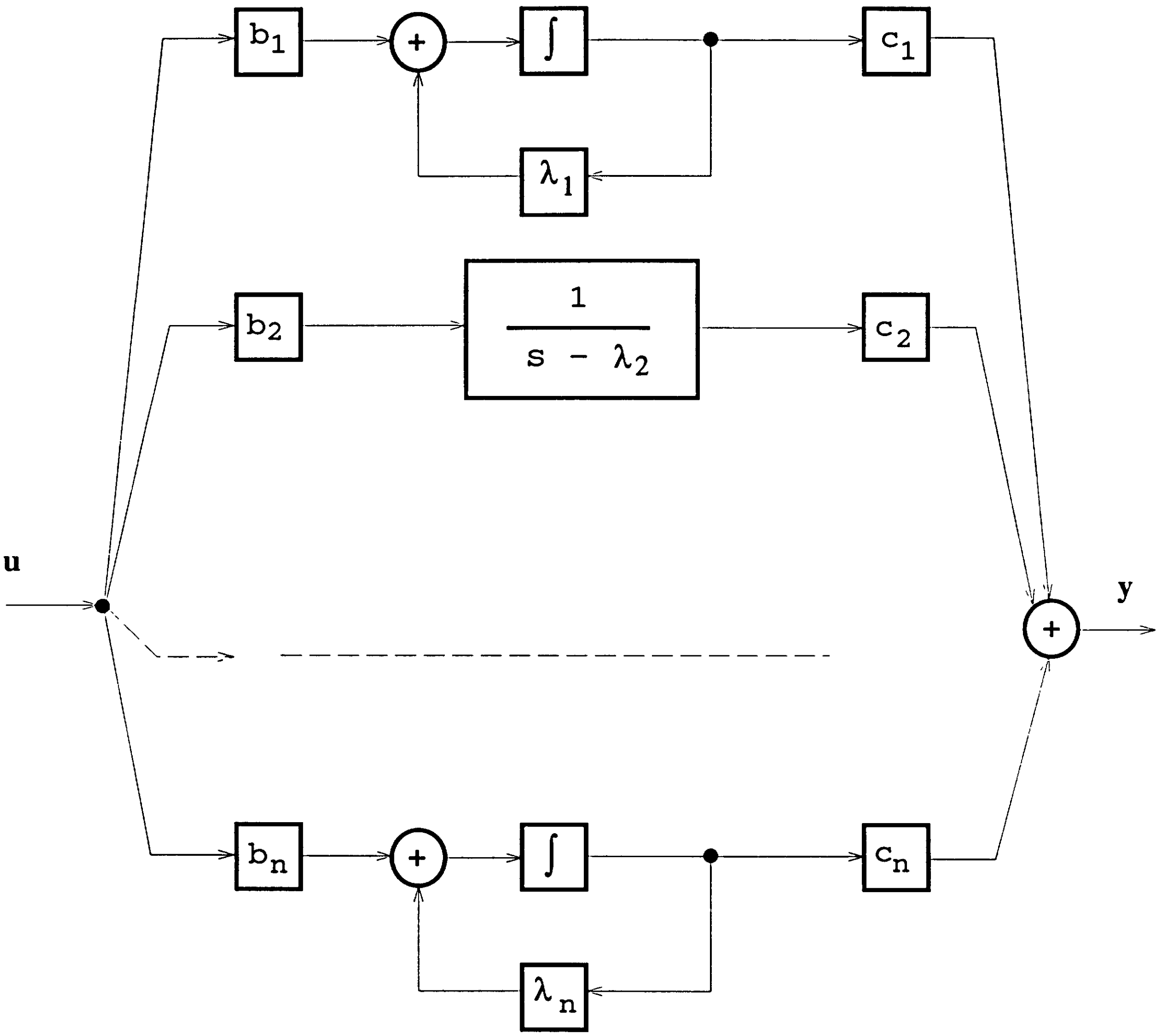

Figure 4-9: Sum, or Parallel or Diagonal realization for $H(s) = \sum_{i=1}^{n} \frac{r_i}{s-p_i}$.

with corresponding transfer function

$$H(s) = C(sI - A)^{-1}B,$$

a reduced-order model of this system is a system of the form

$$\begin{aligned} \dot{x}_r &= A_r x_r + B_r u, & x_r \in \mathbb{R}^r,\ u \in \mathbb{R}^m \\ y &= C_r x_r, & y \in \mathbb{R}^p \end{aligned} \tag{4.33}$$

where $r < n$. Similarly, the transfer function of the reduced-order model is given by

$$H_r(s) = C_r(sI_r - A_r)^{-1}B_r.$$

which therefore contains fewer poles than the original model.

Given a system realization $[A, B, C, D]$, the simplest technique for order reduction would be to truncate and discard some of the system dynamics. The simplest of such operations can be accomplished by what is referred to as spectral truncation. Spectral truncation corresponds to discarding a portion of the system dynamics by truncating a part of the system A matrix and removing the corresponding columns and rows of the B, C and D matrix. In general, this is preceded by a reordering of modes of the A matrix such that truncation will effectively discard the modes that do not significantely affect the system response. Typically this truncation is based on neglecting the "fast" modes, which implies that the reordering performed transforms the A matrix into a block diagonal form where blocks correspond to eigenvalues of increasing magnitude. There are two main reasons why this truncation can result in an innacurate reduced-order model. The first one is that by truncating the fast modes, the resulting model will only be accurate at low frequency. As a result, the time response may be too slow and the initial dynamics of the system might be lost. The second reason has to do with the fact that spectral truncation looks only at the state matrix A without taking into account how controllable or observable the neglected modes are. In other words, it might be possible that through pole-zero cancelation the modes that were left were only minimally, or not at all, present in the output, while the truncated modes might have been responsible for most of the output energy. In this case the modes with the most energy are left out of the reduced-order model and the response will most likely be very inaccurate.

More advanced model reduction techniques for stable systems have been studied in the systems control field. They lead to realizations where the controllability and observability properties of the modes are taken into account, and therefore reflected in the truncation.

4.5.1 Balanced Realization

In this section we discuss the theory of balanced realizations. We start by reviewing some important concepts that are necessary in order to fully understand the model order reduction algorithm that we will be using.

Consider the map $\boldsymbol{F}(t)$ defined in $\mathbb{R}^{n\times m}$ which we assume to be piecewise continuous. The Gramian matrix is defined as

$$\boldsymbol{W}^2 = \int_0^\infty \boldsymbol{F}(t)\,[\boldsymbol{F}(t)]^T\,dt.$$

The gramian matrix is a semi-definite positive matrix whose eigenvalues $\xi_1, \xi_2, \cdots, \xi_n$ are non-negative real quantities, and the corresponding unitary eigenvectors $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \cdots, \boldsymbol{\nu}_n$ are mutually orthogonal. Then, the map $\boldsymbol{F}(t)$ can be written as

$$\boldsymbol{F}(t) = \boldsymbol{\nu}_1 \boldsymbol{f}_1^T(t) + \boldsymbol{\nu}_2 \boldsymbol{f}_2^T(t) + \cdots + \boldsymbol{\nu}_n \boldsymbol{f}_n^T(t)$$

where $\boldsymbol{f}_i^T(t) = \boldsymbol{\nu}_i^T \boldsymbol{F}(t),\ i = 1, \cdots, n$. The vector $\boldsymbol{\nu}_i$ is called the component vector, ξ_i is the component magnitude, $\boldsymbol{f}_i$ is the component function vector and finally $\boldsymbol{\nu}_i \boldsymbol{f}_i(t)$ is called the principal component. The following relations are known to hold

$$\int_0^\infty \boldsymbol{f}_i^T(t)\boldsymbol{f}_j(t)\,dt = 0,\ i \neq j \qquad \int_0^\infty \|\boldsymbol{f}_i(t)\|^2\,dt = \xi_i^2,\ i = 1, \cdots n. \tag{4.34}$$

Consider the system of the form (4.32) to be a minimal state-space realization. We will restrict ourselves to systems that are stable because all the systems that we are interested in fall under that category. The controllability gramian $\boldsymbol{W}_c$ and observability gramian $\boldsymbol{W}_o$ are defined as the solutions of the Lyapunov equations

$$\begin{array}{rcrcl} \boldsymbol{A}\boldsymbol{W}_c & + & \boldsymbol{W}_c\boldsymbol{A}^T & = & -\boldsymbol{B}\boldsymbol{B}^T \\ \boldsymbol{A}^T\boldsymbol{W}_o & + & \boldsymbol{W}_o\boldsymbol{A} & = & -\boldsymbol{C}^T\boldsymbol{C}. \end{array} \tag{4.35}$$

Since the system is stable, the gramians can also be computed directly as

$$\boldsymbol{W}_c = \int_0^\infty e^{\boldsymbol{A}t}\boldsymbol{B}\boldsymbol{B}^T e^{\boldsymbol{A}^T t} dt \tag{4.36}$$

$$\boldsymbol{W}_o = \int_0^\infty e^{\boldsymbol{A}^T t}\boldsymbol{C}^T\boldsymbol{C} e^{\boldsymbol{A}t} dt. \tag{4.37}$$

It is known that while the eigenvalues of the gramian matrices $\boldsymbol{W}_c$ and $\boldsymbol{W}_o$ are dependent upon the state-space coordinates, those of the product matrix $\boldsymbol{W}_c.\boldsymbol{W}_o$ are invariant quantities under any state coordinate transformation. The Hankel singular values of the system transfer function $\boldsymbol{H}(s)$ are directly defined from these quantities as

$$\sigma_v(\boldsymbol{H}(s)) = \sqrt{\lambda_i\left(\boldsymbol{W}_c\boldsymbol{W}_o\right)} \qquad\qquad i = 1, \cdots, n. \tag{4.38}$$

For single input single output (SISO) systems, the single equation

$$\boldsymbol{W}_{co}\boldsymbol{A} + \boldsymbol{A}\boldsymbol{W}_{co} = -\boldsymbol{B}\boldsymbol{C}$$

has the unique solution $\boldsymbol{W}_{co}$ defined as

$$\boldsymbol{W}_{co} = \int_0^\infty \left(e^{\boldsymbol{A}t}\boldsymbol{B}\right)\left(e^{\boldsymbol{A}^Tt}\boldsymbol{C}^T\right)dt$$

In this case the matrix $\boldsymbol{W}_{co} = \boldsymbol{W}_c\boldsymbol{W}_o$ contains double information as regards the degree of controllability and observability, and its eigenvalues are still invariant quantities, not dependent upon the state variable representation.

Consider now the controllability problem, that is the problem of choosing a finite energy control signal $\boldsymbol{u}(t)$ in order to bring the state $\boldsymbol{x}(t)$ to $\mathbf{o}$ at $t = t_f$. Assuming that $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ we get

$$\boldsymbol{x}(t) = e^{\boldsymbol{A}(t-t_0)}\boldsymbol{x}_0 + \int_{t_0}^{t} e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau.$$

At $t = t_f$ we want to have

$$\mathbf{o} = e^{\boldsymbol{A}(t_f-t_0)}\boldsymbol{x}_0 + \int_{t_0}^{t_f} e^{\boldsymbol{A}(t_f-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau,$$

which after some algebra leads to

$$-\boldsymbol{x}_0 = \int_{t_0}^{t_f} e^{\boldsymbol{A}(t_0-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau. \tag{4.39}$$

This is an integral equation in $\boldsymbol{u}(t)$. Defining the operator

$$\boldsymbol{K}_c : \mathbb{L}_2([t_0, t_f], \mathbb{R}^m) \longrightarrow \mathbb{R}^n, \;\; \boldsymbol{u} \longrightarrow \int_{t_0}^{t_f} e^{\boldsymbol{A}(t_0-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau$$

we can write Eqn. (4.39) as

$$\boldsymbol{K}_c\boldsymbol{u} = -\boldsymbol{x}_0.$$

This is a system with more unknowns than equations. Therefore there will only be a solution if $\boldsymbol{x}_0$ is in the range of the columns of the operator $\boldsymbol{K}_c$. Given that $\boldsymbol{x}_0$ is arbitrary, for the system to be controllable over $[t_0, t_f]$, $\boldsymbol{K}_c$ must be subjective, or in other words, the rows of

$$\boldsymbol{F}(t) : [t_0, t_f] \longrightarrow \mathbb{R}^{n \times m}, \ \boldsymbol{F}(t) = e^{\boldsymbol{A}(t_0 - t)}\boldsymbol{B}$$

must be linearly independent. Equivalently, the controllability gramian must be invertible. In this case, and after some algebra, the control can be obtained as

$$\boldsymbol{u}(\tau) = -\boldsymbol{B}^T e^{\boldsymbol{A}(t_0 - \tau)} \boldsymbol{W}_c^{-1} \boldsymbol{x}_0. \tag{4.40}$$

For a proof see [32] or any other advanced linear systems textbook. The relevance of Eqn. (4.40) is that it shows that obtaining $\boldsymbol{u}(\tau)$ requires the inversion of the controllability gramian. Clearly, the better conditioned the gramian is, the more numerically stable and accurate the computation will be.

Now consider the observability problem. A realization is said to be observable on $[t_0, t_f]$ if $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ can be obtained from $\boldsymbol{y}(t)$, $t_0 \leq t \leq t_f$. Since the control is assumed known, its effects can be eliminated, so without loss of generality consider the case where $\boldsymbol{u}(t) = \mathbf{o}$. Then

$$\boldsymbol{y}(t) = \boldsymbol{C} e^{\boldsymbol{A}(t - t_0)} \boldsymbol{x}_0,$$

which is a linear equation in $\boldsymbol{x}_0$. Therefore the initial state can be uniquely recovered if the operator

$$\boldsymbol{K}_o : \mathbb{R}^n \longrightarrow \mathbb{L}_2([t_0, t_f], \mathbb{R}^p) : \ \boldsymbol{x}_0 \longrightarrow \boldsymbol{y}(0) = \boldsymbol{C} e^{\boldsymbol{A}(0 - t_0)} \boldsymbol{x}_0,$$

is injective. This requires that the columns of

$$\boldsymbol{G}(t) : [t_0, t_f] \longrightarrow \mathbb{R}^{p \times n}, \ \boldsymbol{G}(t) = \boldsymbol{C} e^{\boldsymbol{A}(t - t_0)}$$

must be linearly independent. In this case, and after some algebra, the initial state can be obtained as

$$\boldsymbol{x}_0 = \boldsymbol{W}_o^{-1} \int_{t_0}^{t_f} e^{\boldsymbol{A}^T(\tau - t_0)} \boldsymbol{C}^T \boldsymbol{y}(\tau) d\tau. \tag{4.41}$$

Therefore the observability gramian must be non-singular and in order to obtain $\boldsymbol{x}_0$ one is required to invert the observability gramian. Again, the better conditioned the gramian is, the more numerically stable and accurate the computation will be.

Summarizing, the solution of the controllability problem requires the inversion of the controllability gramian, while the solution of the observability problem requires inverting the observability gramian. The question one might ask is how to simultaneously precondition both gramians so as to make the solution of both problems as easy as possible from a numerical standpoint. Clearly a possible transformation would be to precondition one of the gramians to make its condition number as small as possible, since that would facilitate the process of inversion of the matrix. However, preconditioning only one of the gramian merely helps the problem for which the inversion of that gramian is required, that is, one would be in a situation where one would have to trade controllability for observability or vice-versa. Rather, it would be better if both gramians could be preconditioned simultaneously as that would allow us to make meaningful statements regarding "less controllable" *and* "less observable" states which, as we shall see also has important consequences in terms of model order reduction.

Now that we have debated the controllability and observability problems and have motivated the interest in performing a state transformation that preconditions both gramians, let us consider how such a transformation can be obtained. Consider again the state representation in Eqn. (4.32), which corresponds to a robust realization of the section-by-section approximant obtained in section 4.3. We have alluded in the previous section to the well-known fact that the choice of the triplet $[\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]$ is not unique [32]. Indeed, a linear coordinate transformation $\boldsymbol{T}$ in the state space modifies the triplet $[\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]$ to $[\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{C}}]$ *without* modifying the transfer function, that is the input-output relationship.

A particular state transformation that has important properties from the point of view or model order reduction is the so-called balanced transformation, which has been studied in control theory for some time [3, 33]. A balanced transformation leads to a state-space representation which is said to be internally balanced. The corresponding triplet $[\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{C}}]$ is called a *balanced realization* of the transfer function $\boldsymbol{H}(s)$. The word "balanced" refers to the fact that the controllability and observability gramians of the triplet $[\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{C}}]$ are both equal to the *same* diagonal matrix [3]. As we have seen this property has important numerical implications. For a balanced realization then, the following property is satisfied

$$\tilde{\boldsymbol{W}}_c = \tilde{\boldsymbol{W}}_o = \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \sigma_n \end{bmatrix}, \tag{4.42}$$

where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n > 0$ and σ_i is the i^{th} Hankel-singular value of the transfer function of the system, also known as the i^{th} second-order mode.

It is interesting to note that this transformation is equivalent to preconditioning both gramians such that their condition number equals that of the the product matrix $\boldsymbol{W}_c.\boldsymbol{W}_o$. We have stated that the eigenvalues of this matrix are invariant under all similarity

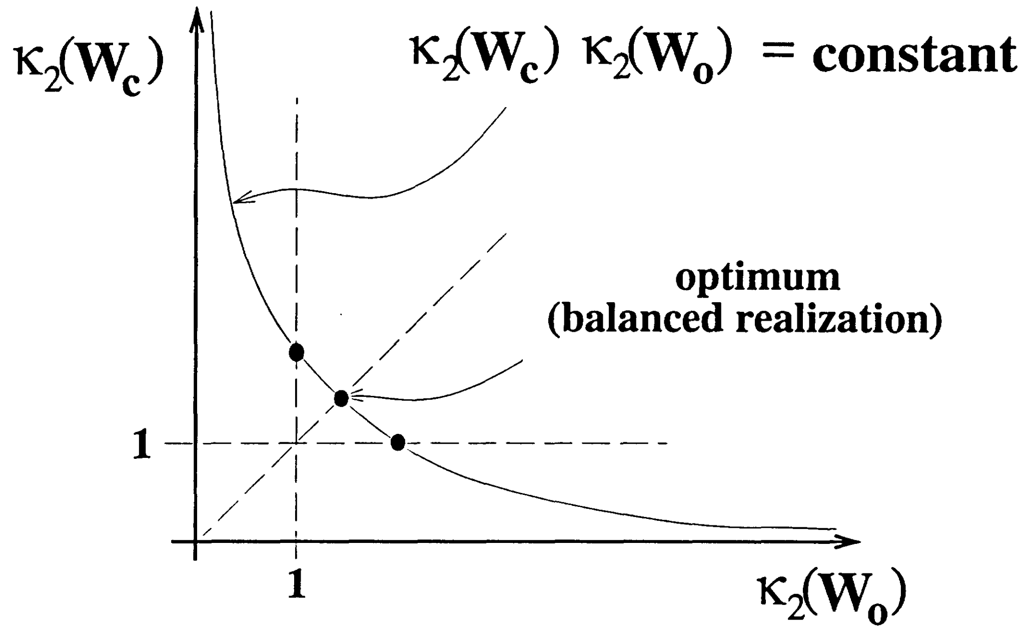

Figure 4-10: Curve of $\kappa_2(\boldsymbol{W}_c)\,\kappa_2(\boldsymbol{W}_o)$ constant in the plane $\{\kappa_2(\boldsymbol{W}_c), \kappa_2(\boldsymbol{W}_o)\}$, indicating the optimum preconditioning situation.

transformations. If one looks at its condition number, we get

$$\kappa_2(\boldsymbol{W}_c.\boldsymbol{W}_o) = \|\boldsymbol{W}_c.\boldsymbol{W}_o\|_2 \left\|(\boldsymbol{W}_c.\boldsymbol{W}_o)^{-1}\right\|_2 = \frac{\sigma^2_{max}(\boldsymbol{W}_c.\boldsymbol{W}_o)}{\sigma^2_{min}(\boldsymbol{W}_c.\boldsymbol{W}_o)}.$$

With some algebra

$$\kappa_2(\boldsymbol{W}_c.\boldsymbol{W}_o) \leq \|\boldsymbol{W}_c\|_2 \left\|\boldsymbol{W}_c^{-1}\right\|_2 \|\boldsymbol{W}_o\|_2 \left\|\boldsymbol{W}_o^{-1}\right\|_2 = \frac{\sigma^2_{max}(\boldsymbol{W}_c)\,\sigma^2_{max}(\boldsymbol{W}_o)}{\sigma^2_{min}(\boldsymbol{W}_c)\,\sigma^2_{min}(\boldsymbol{W}_o)} \tag{4.43}$$

Clearly from Eqn. (4.43) one can see that it is possible to trade-off controllability for observability by moving along the curve with $\kappa_2(\boldsymbol{W}_c)\,\kappa_2(\boldsymbol{W}_o)$ constant. Figure 4-10 shows a curve of constant $\kappa_2(\boldsymbol{W}_c)\,\kappa_2(\boldsymbol{W}_o)$ in the plane $\{\kappa_2(\boldsymbol{W}_c), \kappa_2(\boldsymbol{W}_o)\}$. Improving the solution of the controllability problem implies descending the curve for lower values of $\kappa_2(\boldsymbol{W}_c)$ which is seen to imply larger values of $\kappa_2(\boldsymbol{W}_o)$. From the figure it is immediate to see that the optimum value for the product is obtained in the case when both $\boldsymbol{W}_c$ and $\boldsymbol{W}_o$ are preconditioned to make

$$\kappa_2(\boldsymbol{W}_c) = \kappa_2(\boldsymbol{W}_o) = \frac{\sigma^2_{max}(\boldsymbol{W}_c.\boldsymbol{W}_o)}{\sigma^2_{min}(\boldsymbol{W}_c.\boldsymbol{W}_o)}.$$

The balancing transformation $\boldsymbol{T}$ that takes the state from the representation $\boldsymbol{x}$ to $\tilde{\boldsymbol{x}} = \boldsymbol{T}\boldsymbol{x}$ can be computed explicitly for any triplet $[\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]$, and in particular for the diagonal realization that we have proposed in the previous paragraph. The numerical

Algorithm 4.5.1 (Balanced Realization).

```
1 - obtain the controllability and observability gramians $W_c$
  and $W_o$ by solving the Lyapunov equations
      $A W_c + W_c A^T = -B B^T$
      $A^T W_o + W_o A = -C^T C.$
2 - perform a Cholesky factorization of $W_c$ to obtain
      $W_c = L_c L_c^T$
  where $L_c$ is lower triangular
3 - form $L_c^T W_o L_c$
4 - solve the symmetric eigenvalue/eigenvector problem
      $U^T \left( L_c^T W_o L_C \right) U = \Lambda^2$
5 - form the transformation matrix $T$ as
      $T = L_c U \Lambda^{-1/2}$
6 - obtain the internally balanced realization by applying the
  transformation $x(t) = T\tilde{x}(t)$ to get
      $\tilde{A} = T^{-1} A T$
      $\tilde{B} = T^{-1} B$
      $\tilde{C} = C T$
7 - return the internally balanced realization $[\tilde{A}, \tilde{B}, \tilde{C}]$
```

cost of such a computation is that of solving the matrix Lyapunov equations (4.35) to obtain the controllability and observability gramians and one symmetric eigenvalue problem to diagonalize their product and obtain the Hankel singular values and respective eigenvectors. Algorithm 4.5.1 details a standard procedure used to obtain a balanced realization [34, 4].

All the steps in the algorithm can be shown to have a cost of $\mathcal{O}(n^3)$ [34], and therefore the total cost of the algorithm is also $\mathcal{O}(n^3)$. Step 1 of the algorithm requires the solution of the Lyapunov equations to determine the controllability and observability gramians. In the usual situation the gramians are reliably solved by means of the Bartels-Stewart algorithm [35] at a cost of approximately $15n^3$. The remaining steps of the algorithm involve standard linear algebra procedures that can be found in standard numerical analysis textbooks [36].

Let us backtrack and consider again the problem of controlling the state $\boldsymbol{x}$ using some input $\boldsymbol{u}$. If the state is perturbed to $\boldsymbol{x}+\boldsymbol{\Delta x}$, then the new control signal can be evaluated as $\boldsymbol{u}+\boldsymbol{\Delta u}$ and the quantity

$$\sqrt{\frac{\int_0^\infty \|\boldsymbol{\Delta u}\|^2\,dt}{\int_0^\infty \|\boldsymbol{u}\|^2\,dt}}$$

may be as large as

$$\frac{\sigma_{1c}}{\sigma_{nc}}\frac{\|\boldsymbol{\Delta x}\|^2}{\|\boldsymbol{x}(t)\|^2}$$

where σ_{1c} and σ_{nc} are respectively the largest and smallest singular values of $\boldsymbol{W}_c$. The quotient of these two numbers represents a magnification coefficient that outlines the input signal necessary to drive the perturbed state. On the other hand, if we suppose that the state $\boldsymbol{x}_0$ is derived from input-output measurements, we have the observability problem. If $\boldsymbol{x}_0$ is perturbed to $\boldsymbol{x}_0+\boldsymbol{\Delta x}_0$, the effect of the state perturbation on the output, measured as

$$\sqrt{\frac{\int_0^\infty \|\boldsymbol{\Delta y}\|^2\,dt}{\int_0^\infty \|\boldsymbol{y}\|^2\,dt}}$$

may be as small as

$$\frac{\sigma_{no}}{\sigma_{1o}}\frac{\|\boldsymbol{\Delta x}_0\|^2}{\|\boldsymbol{x}_0\|^2}$$

where now σ_{1o} and σ_{no} are respectively the largest and smallest singular values of $\boldsymbol{W}_o$. For strongly observable $\boldsymbol{x}_0$, the quotient of these two numbers should not be small.

If the system is internally balanced, we know that $\sigma_{1o}=\sigma_{1c}$ and $\sigma_{no}=\sigma_{nc}$. Therefore, in the state coordinate representation of the balanced realization the contribution of each state in terms of controllability and observability can be pointed out. Accordingly, the balanced realization can be partitioned as

$$\begin{aligned}\begin{bmatrix}\dot{\tilde{\boldsymbol{x}}}_1\\ \dot{\tilde{\boldsymbol{x}}}_2\end{bmatrix} &= \begin{bmatrix}\tilde{\boldsymbol{A}}_{11} & \tilde{\boldsymbol{A}}_{12}\\ \tilde{\boldsymbol{A}}_{21} & \tilde{\boldsymbol{A}}_{22}\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{x}}_1\\ \tilde{\boldsymbol{x}}_2\end{bmatrix}+\begin{bmatrix}\tilde{\boldsymbol{B}}_1\\ \tilde{\boldsymbol{B}}_2\end{bmatrix}\boldsymbol{u}\\ \begin{bmatrix}\tilde{\boldsymbol{y}}_1\\ \tilde{\boldsymbol{y}}_2\end{bmatrix} &= \begin{bmatrix}\tilde{\boldsymbol{C}}_1 & \tilde{\boldsymbol{C}}_2\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{x}}_1\\ \tilde{\boldsymbol{x}}_2\end{bmatrix}.\end{aligned} \tag{4.44}$$

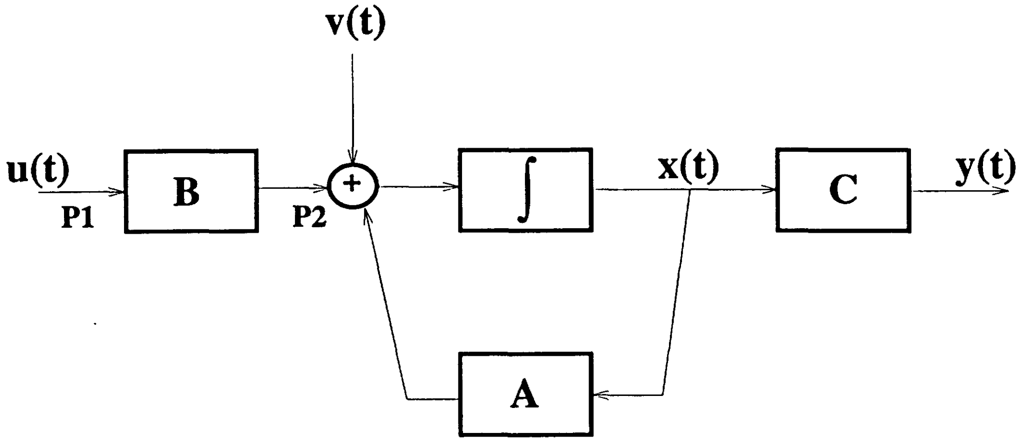

Figure 4-11: General linear system.

The contribution of the weak controllable *and* observable state variables in the input-output map is given by the state vector $\dot{\tilde{\boldsymbol{x}}}_2$, while the strongly controllable *and* observable variables are given by the state vector $\dot{\tilde{\boldsymbol{x}}}_1$.

The importance of this fact lies in the observation that for the specific purpose of extracting *stable* reduced-order models from the state-space representation, it is desirable that the new triplet $[\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{C}}]$ be in a form that allows such an extraction using some simple operation on the new state $\boldsymbol{x} = \boldsymbol{T}\tilde{\boldsymbol{x}}$. The easiest conceivable such operation would be simple state truncation. Eqn. (4.44) shows that such a truncation can be accomplished directly and provides insight into the effect of that truncation on the system.

4.5.2 Truncated Balanced Realization

The main idea underlying model reduction here, is to eliminate any *weak* subsystem which contributes little to the impulse response matrix. This implicitly defines the existence of a "dominant" susbsystem, that is, one whose impulse response matrix is in some sense very close to that of the full model. We shall now show that balanced realizations provide an explicit tool for obtaining that "dominant" reduced-order model.

To further motivate the relevance of balanced realizations and their truncations consider the general linear system scheme in Figure 4-11 and the functions $\boldsymbol{F}(t) = e^{\boldsymbol{A}t}\boldsymbol{B}$ and $\boldsymbol{G}(t) = e^{\boldsymbol{A}^T t}\boldsymbol{C}^T$ which as we know are directly related to the concepts of controllability and observability, as given by Eqns. (4.36) and (4.37). Reporting again to Figure 4-11, the function $\boldsymbol{F}(t)$ represents the state impulse response to impulses injected in point $P1$. In particular the $F_{ij}(t)$ term of $\boldsymbol{F}(t)$ represents the i^{th} state response to an impulse applied at the j^{th} input, when all the other inputs are considered zero. Equivalently, $\boldsymbol{G}(t)$ represents the output response to test impulses applied at node $P2$. Its $G_{ij}(t)$ term represents the j^{th} output response for a vector test signal $\boldsymbol{v}(t)$, of which all component are zero except for the i^{th}, which is an impulse.

If the system is balanced, then the functions $\tilde{\boldsymbol{F}}(t)$ and $\tilde{\boldsymbol{G}}(t)$ become

$$\tilde{\boldsymbol{F}}(t) = e^{\tilde{\boldsymbol{A}}t}\tilde{\boldsymbol{B}} \qquad \text{and} \qquad \tilde{\boldsymbol{G}}(t) = e^{\tilde{\boldsymbol{A}}^T t}\tilde{\boldsymbol{C}}^T.$$

Using the principal component analysis, we get

$$\tilde{\boldsymbol{F}}(t) = \tilde{\boldsymbol{\nu}}_1\tilde{\boldsymbol{f}}_1^T(t) + \tilde{\boldsymbol{\nu}}_2\tilde{\boldsymbol{f}}_2^T(t) + \cdots + \tilde{\boldsymbol{\nu}}_n\tilde{\boldsymbol{f}}_n^T(t) \tag{4.45}$$

$$\tilde{\boldsymbol{G}}(t) = \tilde{\eta}_1\tilde{g}_1^T(t) + \tilde{\eta}_2\tilde{g}_2^T(t) + \cdots + \tilde{\eta}_n\tilde{g}_n^T(t). \tag{4.46}$$

Since the system is assumed to be balanced, the gramians are equal and diagonal, and therefore the orthonormal eigenvectors $\tilde{\boldsymbol{\nu}}_1, \cdots, \tilde{\boldsymbol{\nu}}_n$ and $\tilde{\eta}_1, \cdots, \tilde{\eta}_n$ are such that $\tilde{\boldsymbol{\nu}}_i = \tilde{\eta}_i = \boldsymbol{e}_i$ $i = 1, \cdots, n,$, where $\boldsymbol{e}_i$ denotes the i^{th} elementary basis vector for $\mathbb{R}^n$. In Eqns. (4.45) and (4.46), the terms $\tilde{\boldsymbol{\nu}}_n\tilde{\boldsymbol{f}}_n^T(t)$ and $\tilde{\eta}_n\tilde{g}_n^T(t)$ will therefore make the only contributions respectively to the n^{th} row of $\tilde{\boldsymbol{F}}(t)$ and $\tilde{\boldsymbol{G}}(t)$. Recalling the physical meaning of $\tilde{\boldsymbol{F}}(t)$ and $\tilde{\boldsymbol{G}}(t)$, those terms are the contribution of the state variable $\tilde{\boldsymbol{x}}_n$. In a reduction procedure we seek to obtain a reduced-order model of the system, and thus the contribution of some of the variables will necessary be neglected. The balanced realization, therefore, allows us to choose the state variable set that is related to a significant amount of information in the external representation of the system. In fact a criteria for evaluating the possibility of eliminating $\tilde{\boldsymbol{x}}_n$ from the reduced-order model would be to look at the computed energies of $\tilde{\boldsymbol{f}}_n^T(t)$ and $\tilde{g}_n^T(t)$, which from (4.34) we know to be

$$\int_0^\infty \left\|\tilde{\boldsymbol{f}}_n(t)\right\|^2 dt = \int_0^\infty \|\tilde{g}_n(t)\|^2 dt = \xi_n^2,$$

where σ_n is the n^{th} Hankel-singular value of the transfer function of the system, and which therefore represents the energy contribution of $\boldsymbol{x}_n$ to the controllability and observability maps of the balanced system. Thus the smaller σ_n is with respect to $\sigma_1, \cdots, \sigma_{n-1}$, the least significant is the contribution of the state variable $\tilde{\boldsymbol{x}}_n$ in comparison with that of the other state variables in the balanced realization.

The previous derivation leads us to the concept of internal dominance which can be explained referring to the partitioning in Eqn. (4.44) and to Figure 4-12 [3]. The subsystem $[\tilde{\boldsymbol{A}}_{11}, \tilde{\boldsymbol{B}}_1, \tilde{\boldsymbol{C}}_1]$ is internally dominant if injecting test signals involving $\boldsymbol{u}(t)$ or $\boldsymbol{v}_1(t)$ will give stronger, i.e. with more energy, components as regards $\boldsymbol{x}_1(t)$ and $\boldsymbol{y}(t)$ than corresponding test involving $\boldsymbol{u}(t)$ and $\boldsymbol{v}_2(t)$ as regards $\boldsymbol{x}_2(t)$ and $\boldsymbol{y}(t)$. In terms of

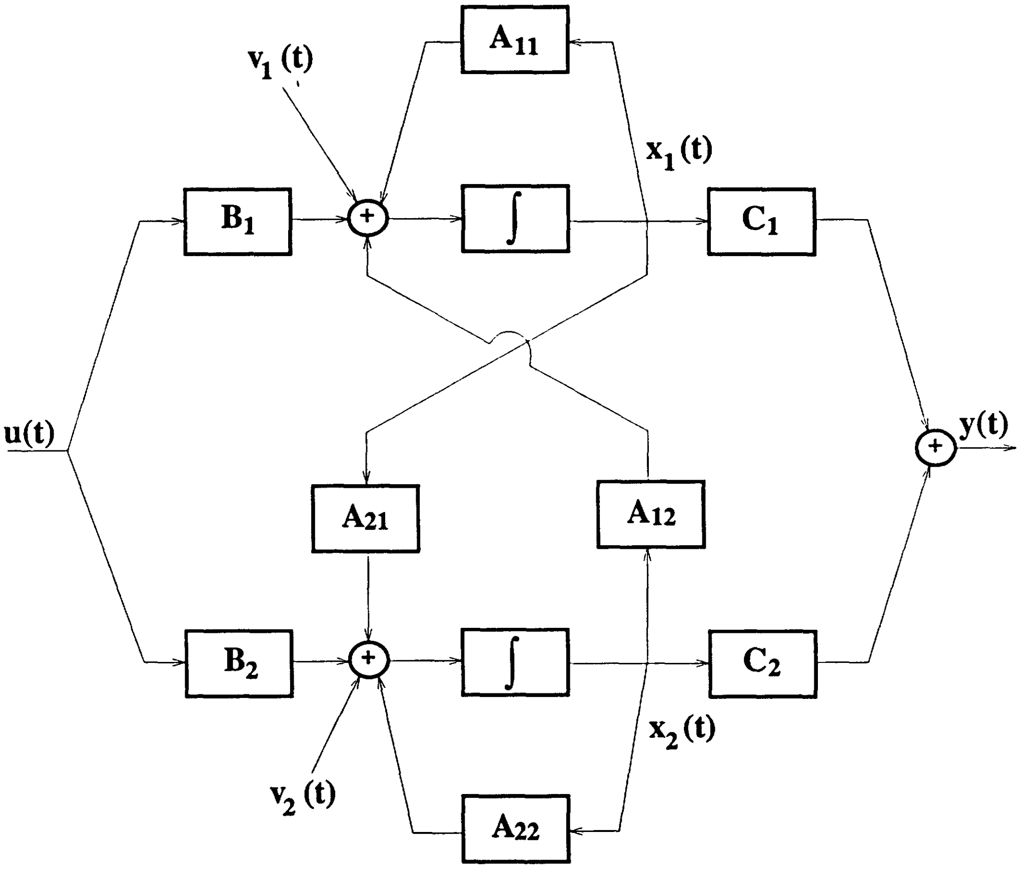

Figure 4-12: State variable partition corresponding to internal dominance of the variables in the set $x_1(t)$.

the controllability and observability maps, our partitioning leads to

$$\tilde{W}_c = \tilde{W}_o = \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_1 & 0 \\ 0 & \boldsymbol{\Sigma}_2 \end{bmatrix},$$

and internal dominance translates to saying that

$$\|\boldsymbol{\Sigma}_1\|_F^2 \gg \|\boldsymbol{\Sigma}_2\|_F^2 \tag{4.47}$$

where $\|.\|_F$ denotes the Frobenius norm, defined as

$$\|\boldsymbol{M}\|_F = \left(\sum_i \sum_j |M_{ij}|^2 \right)^{1/2}.$$

Given that the Hankel singular values of the transfer function of the balanced system are ordered by decreasing magnitude, as shown in Eqn. (4.42), and assuming that $\tilde{\boldsymbol{A}}_{11} \in \mathbb{R}^{k \times k}$, Eqn. (4.47) can also be written as

$$\sum_{i=1}^{k} \left(\sigma_i^2\right)^{1/2} \gg \sum_{i=k+1}^{n} \left(\sigma_i^2\right)^{1/2}. \tag{4.48}$$

If Eqn. (4.47) is satisfied, the subsystem $[\tilde{\boldsymbol{A}}_{11}, \tilde{\boldsymbol{B}}_1, \tilde{\boldsymbol{C}}_1]$ can be considered as an approximated model of the full balanced system $[\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{C}}]$.

It can be shown [3, 4] that if the original system in Eqn. (4.32) is a stable system, then the internally dominant subsystem corresponding to the triplet $[\tilde{\boldsymbol{A}}_{11}, \tilde{\boldsymbol{B}}_1, \tilde{\boldsymbol{C}}_1]$ is also stable and is in fact an internally balanced system itself. In other words, the triplet $[\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{C}}]$ obtained by applying the balancing transformation $\boldsymbol{T}$ to the triplet $[\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]$ has the remarkable property that truncation of the state vector $\tilde{\boldsymbol{x}}$ *necessarily* produces *stable* reduced-order models at any desirable order.

Eqn. (4.47) immediately suggests a model order reduction algorithm based on the balanced realization of a system. Given a realization, a balancing transformation is applied and the Hankel-singular values are examined. If Eqn. (4.48) is not satisfied for any k, then there is no internally dominant subsystem. If (4.48) is satisfied, then the subsystem corresponding to the first k state variables of the model is internally dominant, internally balanced and asymptotically stable. This subsystem is then the reduced-order model that we seek. Algorithm 4.5.2 details an implementation of such a procedure.

Let k be the order of the internally dominant subsystem which is our reduced-order model. Let $[\tilde{\boldsymbol{A}}_k, \tilde{\boldsymbol{B}}_k, \tilde{\boldsymbol{C}}_k]$ be the reduced-order model with a transfer function $\boldsymbol{H}_k(s)$. It

Algorithm 4.5.2 (Model Order Reduction by Truncated Balanced Realization).

```
1 - balance the realization $[\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]$ to obtain $[\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{C}}]$
2 - determine if there is a $k$, $1 \leq k \leq n$ such that
        $\sum_{i=1}^{k} \left(\sigma_i^2\right)^{1/2} \gg \sum_{i=k+1}^{n} \left(\sigma_i^2\right)^{1/2}$
3 - if no such $k$ exists, return failure
4 - if such a $k$ exists, compute the $L_\infty$ error of the truncation
    and make sure it is smaller than the desired accuracy $\epsilon$
        $E_\infty = \|\boldsymbol{E}_k(s)\|_{L_\infty} = \|\boldsymbol{H}(s) - \boldsymbol{H}_k(s)\|_{L_\infty} \leq 2\left(\sigma_{k+1} + \cdots + \sigma_n\right) \leq \epsilon$
5 - perform the truncation and discard the
    $n - k$ less controllable and less observable state to obtain the
    reduced-order model
        $[\tilde{\boldsymbol{A}}_k, \tilde{\boldsymbol{B}}_k, \tilde{\boldsymbol{C}}_k]$
6 - return the stable truncated balanced realization $[\tilde{\boldsymbol{A}}_k, \tilde{\boldsymbol{B}}_k, \tilde{\boldsymbol{C}}_k]$
    and the transfer function truncation error $E_\infty$
```

can be shown [4] that, for the full system

$$\|\boldsymbol{H}(s)\|_{L_\infty} \leq 2\left(\sigma_1 + \sigma_2 + \cdots \sigma_n\right)$$

where the L_∞ norm corresponds to the peak of the magnitude Bode plot of $\boldsymbol{H}(s)$. It can also be shown that the error transfer function for a k^{th} order truncation, $\boldsymbol{E}_k(s) = \boldsymbol{H}(s) - \boldsymbol{H}_k(s)$ has an L_∞ norm that consistently decreases to zero as k is increased to n, the order of the original model, that is

$$\lim_{k \to n} \|\boldsymbol{E}_k(s)\|_{L_\infty} = \lim_{k \to n} \|\boldsymbol{H}(s) - \boldsymbol{H}_k(s)\|_{L_\infty} = 0$$

and moreover that, for any truncation order k, the following error bound holds

$$\|\boldsymbol{E}_k(s)\|_{L_\infty} = \|\boldsymbol{H}(s) - \boldsymbol{H}_k(s)\|_{L_\infty} \leq 2\left(\sigma_{k+1} + \cdots + \sigma_n\right). \tag{4.49}$$

Eqn. (4.49) is not only important in the sense that it provides an estimate for the error induced in the truncation, but also because it allows the choice of the truncation order k to be done a-posteriori. For most of the usual approximation methods, such as Padé approximation, the order of the approximant has to be chosen a-priori before any computation is done. If after the model is obtained one realizes that the accuracy provided is not sufficient, the process must be restarted to compute a higher-order model [17]. Furthermore, such methods do not enjoy such an error reduction property because of the ample experimental evidence that the Padé model becomes unstable as the order is increased. Eqn. (4.49) on the other hand, allows us to obtain the balanced realization and then determine the necessary order for the model such that the error is within the desired accuracy.

It has been shown that the Hankel singular values alone do not reflect the full contribution of each state in terms of the L^2 magnitude of the impulse response. A new type of invariants, named *balanced gains*, have therefore been introduced which allow the impulse response to be quantified in a new representation in terms of the state variable energy. In this new representation the balanced gains, $v_1, \cdots, v_n$, $v_i \geq 0$, are used to parameterize the balanced realization triple $[\tilde{\boldsymbol{A}}_k, \tilde{\boldsymbol{B}}_k, \tilde{\boldsymbol{C}}_k]$. For the details of the algorithm, see [37, 38]. Under this representation it is then shown that

$$\|\boldsymbol{H}(t)\|_{L_2} = \left(\int_0^\infty \boldsymbol{H}(t)\left[\boldsymbol{H}(t)\right]^T dt\right)^{1/2} = \left(\sum_{i=1}^{n} \sigma_i v_i^2\right)^{1/2}$$

where $\boldsymbol{H}(t)$ is the impulse response matrix corresponding to the transfer function $\boldsymbol{H}(s)$ and the L_2 norm is defined as indicated. Furthermore, it is also shown that a k^{th} order truncation of the internally dominant system so-parameterized has an L_2 norm given by

$$\|\boldsymbol{H}_k(t)\|_{L_2} = \left(\sum_{i=1}^{k} \sigma_i v_i^2 \right)^{1/2} \tag{4.50}$$

and therefore the truncation error is given by

$$\|\boldsymbol{H}(t) - \boldsymbol{H}_k(t)\|_{L_2} = \sum_{i=1}^{n} \sigma_i v_i^2 + \sum_{i=1}^{k} \sigma_i v_i^2 - 2tr\{\tilde{\boldsymbol{C}}_k \boldsymbol{V} \tilde{\boldsymbol{C}}_k^T\} \tag{4.51}$$

where $\boldsymbol{V}$ satisfies

$$\tilde{\boldsymbol{A}}_k \boldsymbol{V} + \boldsymbol{V} \tilde{\boldsymbol{A}}^T = -\tilde{\boldsymbol{B}}_k \tilde{\boldsymbol{B}}^T.$$

From Eqn. (4.51), we can show that

$$\|\boldsymbol{H}(t) - \boldsymbol{H}_k(t)\|_{L_2} \geq \left(\sum_{i=1}^{n} \sigma_i v_i^2 \right)^{1/2} - \left(\sum_{i=1}^{k} \sigma_i v_i^2 \right)^{1/2}$$

which clearly indicates how in the parameterized balanced realization the contribution of the i^{th} state in the L_2 sense is given by a term in $\sigma_i v_i^2$. Therefore, even if a system is weakly controllable and observable, the importance in terms of impulse response energy is essentially associated to the terms $\sigma_i v_i^2$. Eqn. (4.51) also provides a criteria for a-posteriori determination of the order of the reduced model. The accuracy in this case should be given as a time-domain quantity and not as a frequency-domain parameter as is the case in Algorithm 4.5.2.

While the ability to work with time-domain accuracy parameters is perhaps more natural in the context of transient circuit simulation, the parameterization of the balanced realization requires that the balanced gains be selected beforehand. Unfortunately there does not seem to be any robust way to perform this selection automatically and in a case-independent manner. Furthermore, our experience with the truncated balanced approximations has shown that, for our examples, acceptable time-domain accuracy results from our frequency-domain approximations. As such, this approach was not followed.

Before we complete our study of state-space techniques for model-order reduction, we should state that in [4], it is shown that the truncated balanced realization is not optimal among all possible approximations of order k. It is in fact possible to obtain an approximation with smaller error, but the cost of its computation increases. Again, our experience with the truncated balanced approximations seems to indicate that for our examples such an optimality is not necessary.

4.5.3 Time-Domain Constraints

Judging the validity of the reduced-order model depends not only on meeting the L_∞ error criterion mentioned above but also on meeting the goals of the circuit simulation task for which this reduced model is used. Typically, in circuit simulations, it is essential that the reduced model match the original transfer function at $s = 0$ so that the steady-state behavior of both the reduced and full models are identical. Moreover, when the objective is to have a good match between the time-domain responses of the two models, it is essential that their transfer functions match at $s = \infty$ so that their initial behavior is the same.

In situations where the objective is to match the responses of the interconnect full and reduced models to a step input, it has been shown that it is possible to build stable reduced-order model based on balanced truncation that achieves accurate steady-state and transient behavior [39]. In other situations, where the recovery of the steady-state behavior is more important, one would apply a least-squares/collocation technique to match the reduced-order model with the full model at zero frequency [40]. The resulting reduced-order model should be stable with very few poles, almost identical to the original model at high frequencies, match it at very low frequencies, and will still meet the theoretical L_∞ error criterion. For most circuit simulation applications it will be necessary to guarantee that both the initial conditions and the steady-state values are satisfied simultaneously. In this case both constraints are introduced during the minimization process, as discussed in section 4.3.1.

4.5.4 Truncated Balanced Realization: numerical example

In order to test the accuracy of the order reduction algorithm, we consider two examples. The first one is the transmission line example whose equations were introduced as an example in section 2.5. The second example is the RLC uniform lumped model of interconnect which was introduced in section 4.2.1.

The truncated balanced realization method was applied to the transfer function obtained using the section-by-section procedure on the transmission line data as described in section 4.3.2. For that purpose a diagonal representation was obtained from the section-by-section transfer function approximant and that realization was balanced using the algorithm outlined in section 4.5.1. Truncation of this balanced realization was then performed. It was found that reduced models with seven poles each were appropriate to approximate the full transfer functions of both $S_{12}(j\omega)$ and $Y_o(j\omega)$. In Figures 4-13 and 4-14, we compare the magnitude plots of the reduced transfer functions of, respec-

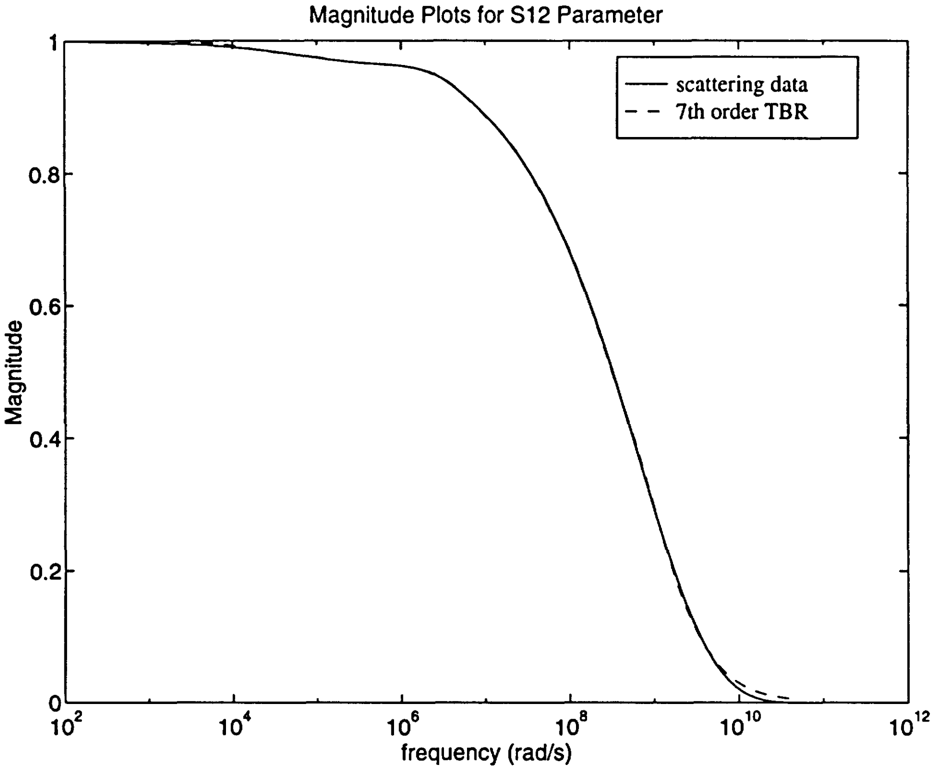

Figure 4-13: Accuracy of the reduced-order model fit for the magnitude of the S_{12} transfer function with respect to the transmission line data points.

tively, $S_{12}(j\omega)$ and $Y_o(j\omega)$ with the transmission line data points. As one can see that match is very accurate as the error is within 1%. However in contrast to the section-by-section approximation the low-frequency error can be larger. In Figures 4-15 and 4-16, the magnitude plots of the frequency dependent fitting errors from the section-by-section approximation and the reduced-order model are shown for $S_{12}(j\omega)$ and $Y_o(j\omega)$, respectively.

In order to incorporate such a transmission line model into a circuit simulator, and as outlined in section 4.5.3, a constrained minimization is performed after obtaining the truncated balanced realization. This minimization recomputes the residues of the transfer function obtained from the truncated system, while keeping its dynamics and therefore its stability properties.

The following example considers the time-domain step responses for the circuit in Figure 4-3. Given the simplicity of this circuit, the state-space representation can be constructed directly from inspection. Once a state-space representation is obtained the moments of the system can be computed and a Padé approximation of any order can easily derived, according to Algorithm 4.2.1. Similarly, once a state-space representation of the system is obtained, it can be internally balanced using Algorithm 4.5.1 and a

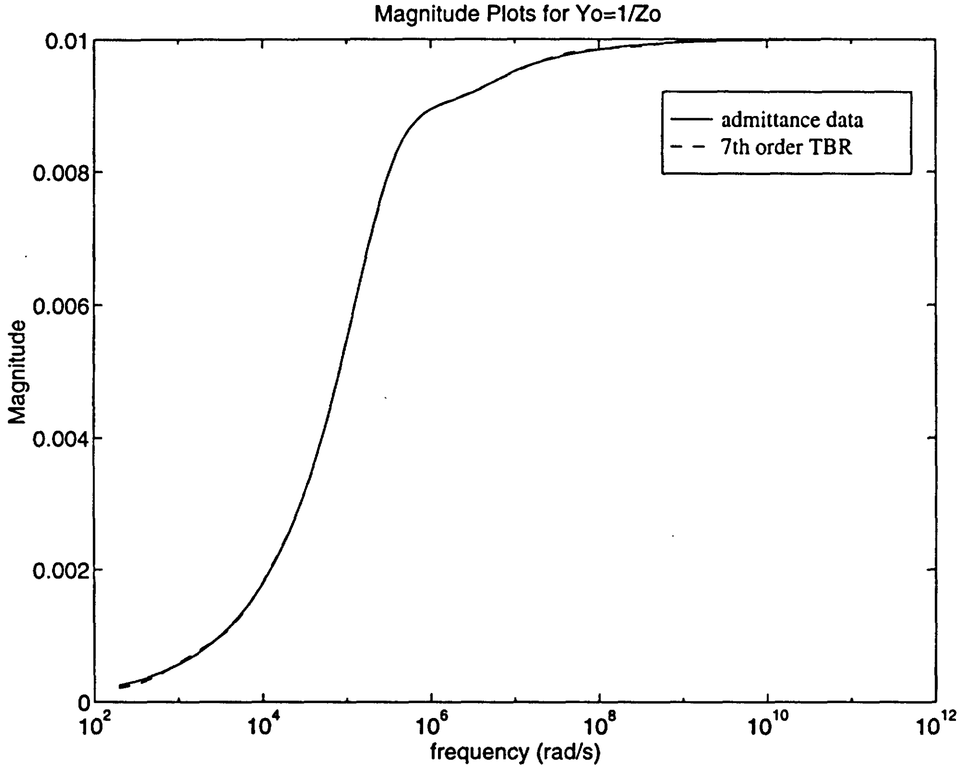

Figure 4-14: Accuracy of the reduced-order model fit for the magnitude of the $\boldsymbol{Y}_o$ transfer function with respect to the transmission line data points.

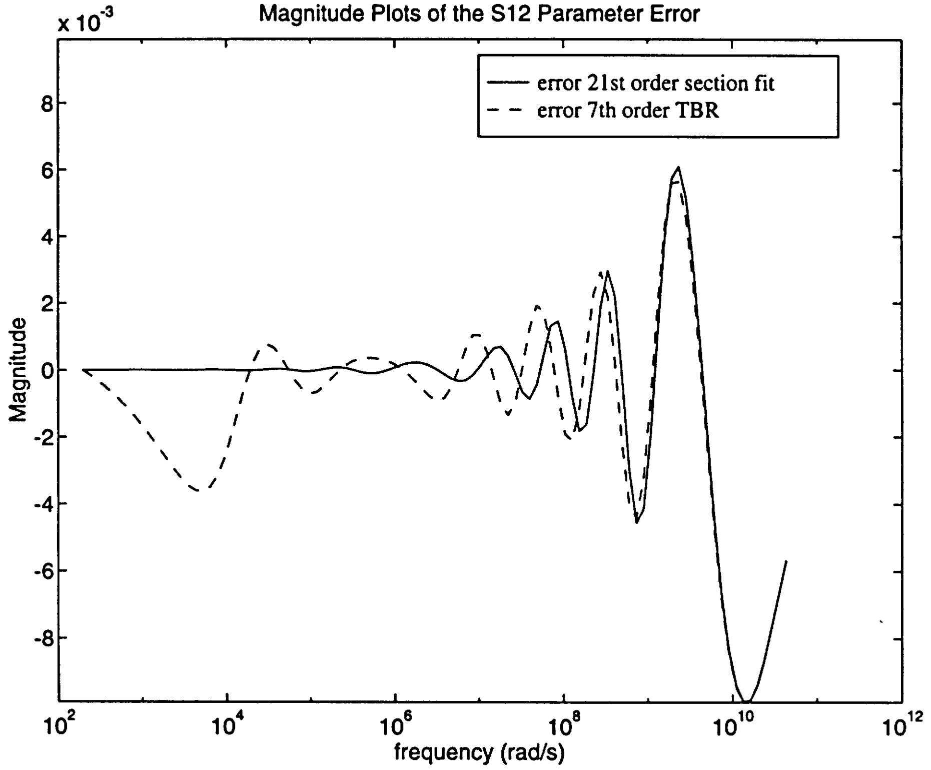

Figure 4-15: Magnitude plots of the errors with respect to the transmission line data points of the section-by-section approximant and the reduced-order transfer function for the S_{12} parameter.

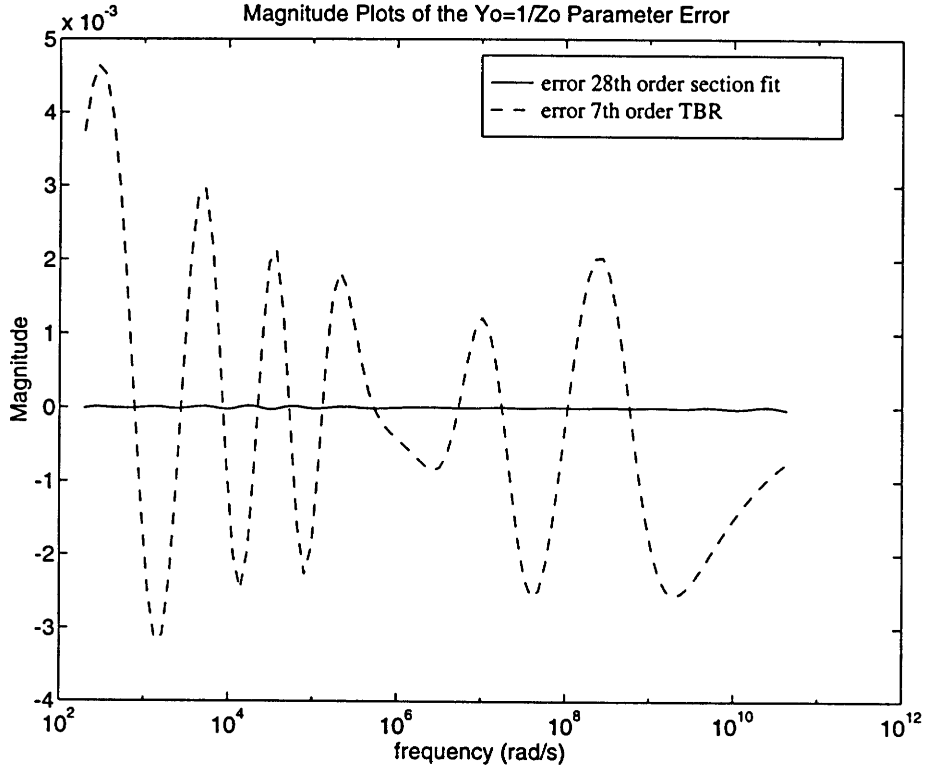

Figure 4-16: Magnitude plots of the errors with respect to the transmission line data points of the section-by-section approximant and the reduced-order transfer function for $\boldsymbol{Y}_o$.

truncation of any order obtained.

For the purpose of comparing stability and accuracy, we will also present the results obtained using Padé approximations of the same order. Since both the Padé approximations and the truncated balanced realizations are in the form of rational functions, the treatment given to each is similar. Given a k^{th} order approximation in the frequency-domain in the form of a rational function such as

$$\boldsymbol{H}_k(s) = \sum_{i=1}^{k} \frac{r_i}{s - p_i} \tag{4.52}$$

then, an inverse transform can be applied to *analytically* compute the corresponding time response as:

$$g(t) = \sum_{i=1}^{k} r_i e^{p_i t}. \tag{4.53}$$

Figures 4-17 compares the exact step response of the circuit in Figure 4-3 with those obtained with 2^{nd} order Padé and truncated balanced realization models. For this order, the Padé approximation is stable, as we had seen in section 4.2.1. The truncated balanced realization model is guaranteed stable irrespective of the order chosen. Furthermore the reduced-order model approximation is more accurate as the Padé approximation seems to have introduced some delay.

In order to improve the accuracy of the time-domain responses, the order of the approximations is increased. Figures 4-18 compares the exact step response of the circuit in Figure 4-3 with those obtained with Padé and a truncated balanced realization, both of order 8. As shown in the figure, the 8^{th} order Padé approximation is unstable, thus producing a totally inaccurate time-domain response. On the other hand, the truncated balanced realization model, which is necessarily stable, is also seen to be very accurate as it produces a response which is almost indistinguishable from the exact response.

4.6 Time Domain Simulation

The usual, though not unique, way to incorporate frequency-described devices into a circuit simulator is via a convolution process at each timepoint. If for instance some device's admittance is described in the frequency domain via some transfer function $\boldsymbol{G}(s)$ or possibly by a collection of frequency measurements $\boldsymbol{G}(j\omega_i)$, then by Laplace inversion or application of the inverse fast Fourier transform the corresponding impulse response can be computed. Then, the response of the device at any given time can be determined

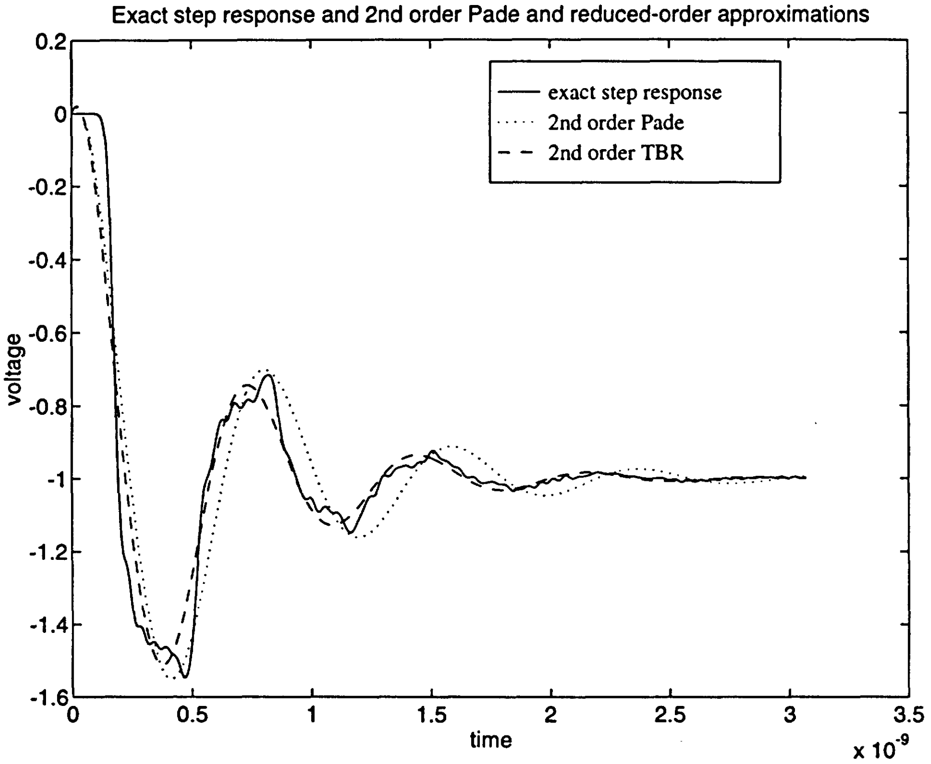

Figure 4-17: Comparing the exact step response with that obtained using respectively a 2^{nd} order Padé and a 2^{nd} order model based on truncating a balanced realization of the system.

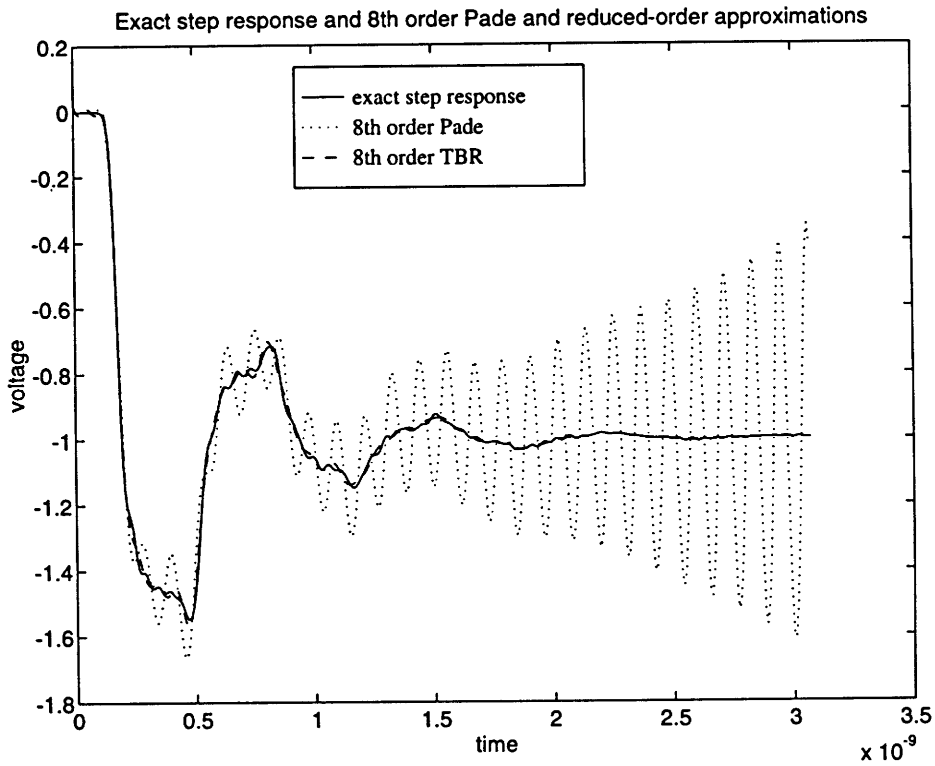

Figure 4-18: Comparing the exact step response with that obtained using respectively a 8^{th} order Padé and a 8^{th} order model based on truncating a balanced realization of the system.

by convolving the impulse response with an excitation waveform, that is, at any point t in time the current flowing through the device can be computed as:

$$i(t) = (v \star g)(t) = \int_{-\infty}^{t} v(\tau)g(t-\tau)d\tau \tag{4.54}$$

where $v(t)$ is the voltage drop along the device and $g(t) = \mathcal{L}^{-1}\left[G(s)\right](t)$ (or equivalently $g(t) = \mathcal{F}^{-1}\left[G(j\omega)\right](t)$) is the impulse response of the device. Figure 4-19 shows how this process is conducted. From the transfer function, the impulse response is obtained, possibly via inverse fast Fourier transform, and then at each time point this impulse response is convolved with some other waveform. Numerical computation of the convolution integral in Eqn. (4.54) can be accomplished with a standard quadrature formula, depending on the desired accuracy. Trapezoidal quadrature or Simpson's rule are fairly common techniques in this regard.

As can be seen in from. (4.54) or from Fig 4-19, this approach can quickly become very expensive. In fact, it requires that at every simulator timepoint, the impulse response be convolved with the entire computed excitation waveform. That is, computing the convolution integral at time $t = t_N$, where t_N is the N^{th} timepoint used in the simulation process, involves all the N previously computed values of the excitation and their impulse response counterparts and therefore has a cost of $\mathcal{O}(N)$. If a given simulation run uses a total of N timepoints, then the total cost of the convolutions required for evaluating this device will be $\mathcal{O}(N^2)$. This quadratic growth makes the cost very expensive if N is large and this fact has been a major drawback for convolution based approaches to incorporating frequency-described models into a simulator.

4.6.1 Recursive Convolution

As mentioned in previous sections, an alternative approach to describe a frequency-domain model is to approximate the frequency-domain representation with a rational function. In this case it is possible to dramatically accelerate the convolution process using a recursive convolution algorithm [2]. This process is indeed extremely simple to understand and it is worthwhile going through the necessary computations to grasp its true benefits. Assume for example that a rational function has been computed that approximates some device's admittance in the frequency domain within some predetermined accuracy. If a partial fraction expansion is computed for that rational function

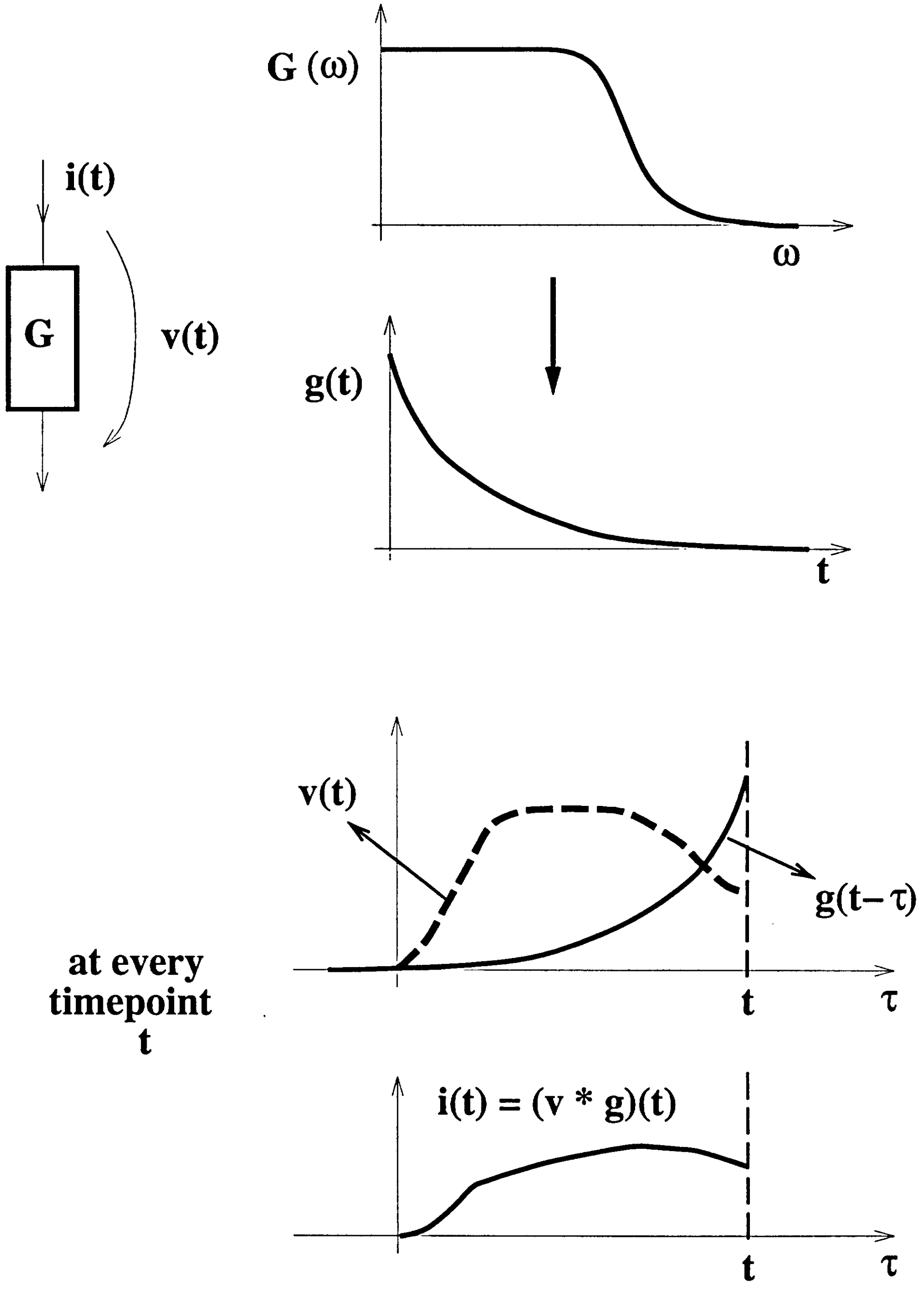

Figure 4-19: Step by step procedure for incorporating a frequency-described device into an electrical circuit simulator. Given a transfer function or an approximation to measured or tabulated frequency values, an impulse response is obtained. Then at each timepoint in the simulation process, that same impulse response is convolved with some waveform to produce a desired current or voltage. In this example it is assumed that the frequency model represents the device admittance and therefore the voltage drop along the device is convolved with the admittance impulse response to obtain the current.

one obtains

$$G(s) = \sum_{i=1}^{k} \frac{r_i}{s - p_i} \tag{4.55}$$

where the r_i, p_i may in general be real or complex numbers and $Real\{p_i\} < 0$, $\forall i$ if the transfer function represents a stable device. Clearly since the device response in time is a real function, if some r_i or p_i has a nonnegative imaginary part, its complex conjugate must also be present, that is

$$\text{If Imag}\{r_i\} \neq 0 \ \text{ or } \ Imag\{p_i\} \neq 0 \implies \exists\ j \neq i : r_j = r_i^\star \ \text{ and } \ p_j = p_i^\star \tag{4.56}$$

where $r_i^\star$ represents the complex conjugate of r_i.

From this representation, an inverse transform can be applied to *analytically* compute the corresponding impulse response as:

$$g(t) = \sum_{i=1}^{k} r_i e^{p_i t}. \tag{4.57}$$

To compute the convolution of a waveform with this impulse response, for example if $i(t) = (v * g)(t)$, then at time $t + h$:

$$\begin{aligned}
i(t+h) &= (v * g)(t+h) = \\
&= \int_{\infty}^{t+h} v(\tau) g(t+h-\tau) d\tau = \\
&= \int_{\infty}^{t+h} v(\tau) \sum_{i=1}^{k} r_i e^{p_i(t+h-\tau)} d\tau = \\
&= \sum_{i=1}^{k} \int_{\infty}^{t+h} v(\tau) r_i e^{p_i(t+h-\tau)} d\tau = \\
&= \sum_{i=1}^{k} \int_{\infty}^{t} v(\tau) r_i e^{p_i(t+h-\tau)} d\tau + \sum_{i=1}^{k} \int_{t}^{t+h} v(\tau) r_i e^{p_i(t+h-\tau)} d\tau = \\
&= \sum_{i=1}^{k} e^{p_i h} \int_{\infty}^{t} v(\tau) r_i e^{p_i(t-\tau)} d\tau + \sum_{i=1}^{k} \int_{t}^{t+h} v(\tau) r_i e^{p_i(t+h-\tau)} d\tau.
\end{aligned} \tag{4.58}$$

Recall now that by definition then

$$\begin{aligned} i(t) &= (v * g)(t+h) = \int_{\infty}^{t+h} v(\tau)g(t-\tau)d\tau = \\ &= \int_{\infty}^{t+h} v(\tau) \sum_{i=1}^{k} r_i e^{p_i(t-\tau)} d\tau = \\ &= \sum_{i=1}^{k} \int_{\infty}^{t+h} v(\tau) r_i e^{p_i(t-\tau)} d\tau. \end{aligned} \tag{4.59}$$

If we let

$$I_i(t) = \int_{\infty}^{t+h} v(\tau) r_i e^{p_i(t-\tau)} d\tau. \tag{4.60}$$

Then

$$i(t) = \sum_{i=1}^{k} I_i(t). \tag{4.61}$$

Using Eqn (4.60) on Eqn (4.58) it becomes evident that:

$$i(t+h) = (v * g)(t+h) = \sum_{i=1}^{k} e^{p_i h} I_i(t) + \sum_{i=1}^{k} \int_{t}^{t+h} v(\tau) r_i e^{p_i(t+h-\tau)} d\tau. \tag{4.62}$$

The significance of Eqn. (4.62) is that it shows that the convolution of the impulse response with the waveform $v(t)$ at time $t+h$ can be obtained by adding two terms. The first term is basically a *decayed* version of the convolution computed in the previous simulator timepoint, t. The second term is an integral that does not depend on any values of the waveform before t. In other words, if at every timepoint the convolution is computed considering each term of the form $r_i e^{p_i t}$ separately, and the values $I_i = I_i(t)$ are stored and saved, then the convolution at the following simulator timepoint becomes a recursive operation. In fact, assuming that I_i, $i = 1, \cdots k$ are known at time t, four simple operations can be performed to obtain the convolution at time $t+h$. This is described in Algorithm 4.6.1. Storing the I_i values which have to be kept from timestep to timestep where they are updated, adds a negligible storage cost, given that this merely implies storing a vector of size k, where k, the number of poles in the approximation, is in general a small number.

Algorithm 4.6.1 (Recursive Convolution).

Assuming that I_i, $i = 1, \cdots k$ are known at time t where

$$i(t) = (v \star g)(t) = \sum_{i=1}^{k} I_i(t) \qquad \text{and } I_i(t) = \int_{\infty}^{t+h} v(\tau) r_i e^{p_i(t-\tau)} d\tau.$$

1- decay each of the partial integrals: this is the recursive part

$$J_i = e^{p_i h} I_i$$

2 - compute the current timepoint contribution to the convolution considering each exponential separately

$$C_i = \int_{t}^{t+h} v(\tau) r_i e^{p_i(t+h-\tau)} d\tau$$

3 - recompute the values of I_i, now relative to time $t + h$; this sets up the recursion for the next timepoint

$$I_i = J_i + C_i \quad i = 1, \cdots k$$

4- finally add all contributions together

$$i(t+h) = (v \star g)(t+h) = \sum_{i=1}^{k} I_i$$

Clearly steps 1 and 2 require on the order of $\mathcal{O}(1)$ work per timepoint, while steps 3 and 4 require $\mathcal{O}(k)$ work where k is the number of poles in the rational function. In other words, this recursive procedure has a total cost of $\mathcal{O}(k)$ per timepoint, where k will, in general, be small. Therefore if a given simulation run uses a total of N timepoints, then the total cost of the convolutions required for evaluating this device will be $\mathcal{O}(kN)$. If $N \gg k$ this results in substantial savings when compared to the $\mathcal{O}(N^2)$ cost of the direct convolution methods. This low complexity has been an important driving force in the recent success and popularity of rational function approximations implemented via recursive convolution into circuit simulators.

In the preceding discussion it was assumed that either both the poles p_i and residuals r_i of the rational function are real, or the simulator is able to handle complex numbers. This assumption can be loosened since if this is not the case, it is also possible to derive equivalent recursive expressions for the convolution in the case of complex conjugate poles and residuals. Furthermore it is also assumed that there are no repeated poles in the rational function. In this case this assumption can with generality be accepted given that the poles are generally computed numerically and the likelihood of obtaining repeated poles if minimal. If by chance this were to happen one could modify one of the repeated poles slightly, such that the additional frequency error would be negligible and the poles would no longer be repeated.

4.6.2 Numerical Integration

While steps 1, 3 and 4 in the algorithm outlined in the previous section involve only multiplication or addition of known quantities step 2 requires the numerical computation of the integral

$$C_i = \int_t^{t+h} v(\tau) r_i e^{p_i(t+h-\tau)} d\tau.$$

Computing this integral could be accomplished using any one of the standard quadrature techniques, such as trapezoidal quadrature. In this case one would get[1]

$$\int_t^{t+h} v(\tau) r_i e^{p_i(t+h-\tau)} d\tau = r_i \frac{v_{t+h} e^{p_i h} + v_t}{2}.$$

Clearly one would like to compute this integral as accurately as possible given that any error incurred in its computation will not only affect the accuracy of the circuit solution

[1] We note that v_{t+h} is not a known quantity but is in fact being solved implicitly, as described in chapter 2, with a nonlinear solution method such as Newton's method. Therefore the most current guess would be used here.

at $t+h$ but it will also directly affect all future device evaluations due to the recursive nature of the algorithm. Unfortunately, given that $v(t)$ is not known exactly, this is not always possible and some error must be tolerated. Nevertheless using a standard quadrature formula to compute the above integral is not likely to be the best solution in this situation. Consider for instance the special case that $v(t)$ is constant throughout the integration interval. In this particular case, if $v(t) = v_{ss}$ and given the form of the integrand function, it is possible to compute the integral exactly to obtain

$$\int_t^{t+h} v(\tau) r_i e^{p_i(t+h-\tau)} d\tau = v_{ss} r_i e^{p_i(t+h)} \int_t^{t+h} e^{-p_i \tau} d\tau = v_{ss} r_i \frac{e^{p_i h} - 1}{p_i}. \tag{4.63}$$

This value may be somewhat different from the one produced by the trapezoidal quadrature, particularly if h is large which is desirable for simulation efficiency.

The advantage of the approach in (4.63) is related to an issue already discussed in previous sections. In section 4.5.3 we stated the fact that typically, in circuit simulations it is essential that the steady-state behavior be computed as accurately as possible. Failure to do so, introduces a consistent steady-state error in the waveforms and may lead to erroneous solutions which may cause other problems later in the simulation. In the frequency-domain, as stated in section 4.5.3 this implies that the reduced model must match the original transfer function at $s = 0$ so that the steady-state of both the reduced and full models are identical. In the time-domain an equivalent requirement would be to say that if the circuit approaches steady-state, the device characteristics must be computed exactly so that no error is induced in the simulation and the computed waveforms evaluate to the correct steady-state values. In our frequency-dependent admittance device this would translate into stating that if the device approaches steady-state then the convolution integral

$$(v \star g)(t) = \int_{-\infty}^{t} v(\tau) g(t-\tau) d\tau$$

should be computed exactly. Given that the waveform $v(t)$ was computed with finite precision and is not known in closed form, it is not in general possible to achieve this goal. However, it is possible to compute the convolution integral exactly in the special case where $v(t)$ is constant. In order to motivate why this special case is important consider the following observation:

Remark. Consider a device whose constitutive relation is described via the convolution of an impulse response $g(t) \in \mathbb{L}_1(\mathbb{R},\mathbb{R})$ with some controlling variable $v(t)$,

$$i(t) = (v \star g)(t) = \int_{-\infty}^{t} v(\tau) g(t-\tau) d\tau.$$

Let $\hat{\imath}(t_k) = \mathcal{I}\left[(-\infty, t_k], \hat{v}(t_k), g(t_k - \tau)\right]$ be the algorithm or formula used in the computation of the convolution integral of the sequence $\hat{v}_k = \hat{v}(t_k)$ with $g(t)$ at the k^{th} timepoint, t_k. The interval $(-\infty, t_k$ indicates which portion of the convolution integral is being computed, and τ is the variable taking values in that interval. Assume that this integration formula satisfies the following usual properties:

1. (Linearity)

$$\begin{aligned}\mathcal{I}\left[(-\infty, t_k], \alpha\, \hat{v}_k + \beta\, \hat{u}_k, g(t_k - \tau)\right] &= \\ &\alpha\, \mathcal{I}\left[(-\infty, t_k], \hat{v}_k, g(t_k - \tau)\right] + \beta\, \mathcal{I}\left[(-\infty, t_k], \hat{u}_k, g(t_k - \tau)\right]\end{aligned}$$

2. (Comparison)

$$\begin{aligned}&\text{If } v_k \le u_k,\ \forall\, k \in [k_1, k_2],\ \Longrightarrow \\ &\quad \Longrightarrow\ \mathcal{I}\left[[t_{k_1}, t_{k_2}], \hat{v}_k, g(t_k - \tau)\right] \le \mathcal{I}\left[[t_{k_1}, t_{k_2}], \hat{u}_k, g(t_k - \tau)\right]\end{aligned}$$

3. (Modulo)

$$\left|\mathcal{I}\left[[t_{k_1}, t_{k_2}], \hat{v}_k, g(t_k - \tau)\right]\right| \le \mathcal{I}\left[[t_{k_1}, t_{k_2}], |\hat{v}_k|, |g(t_k - \tau)|\right]$$

4. (Additivity)

$$\mathcal{I}\left[[t_{k_1}, t_{k_3}], \hat{v}_k, g(t_k - \tau)\right] = \mathcal{I}\left[[t_{k_1}, t_{k_2}], \hat{v}_k, g(t_k - \tau)\right] + \mathcal{I}\left[[t_{k_2}, t_{k_3}], \hat{v}_k, g(t_k - \tau)\right]$$

5. (Zero-Contraction) If $\forall\, \epsilon > 0, k \in [k_1, k_2], t \in [t_{k_1}, t_{k_2}]$,

$$|v_k| \le \epsilon \ \text{ or } \ |g(t_k - \tau)| \le \epsilon,\ \Longrightarrow\ \left|\mathcal{I}\left[[t_{k_1}, t_{k_2}], \hat{v}_k, g(t_k - \tau)\right]\right| \le \epsilon\, K$$

   for some K. Since ϵ can be arbitrarily small, the computed integral equals zero.

If the convolution integral can be computed exactly in the special case that the controlling variable is constant then, if the controlling variable is not constant but tends to a constant in the limit $t \to \infty$, the computed integral also approaches the exact value as $t \to \infty$. In other words

$$\text{If } \lim_{t \to \infty} v(t) = v_{ss} \quad \text{then} \quad \lim_{k \to \infty} \hat{\imath}(t_k) = \lim_{t \to \infty} i(t)$$

where $i(t)$ is the exact solution and $\hat{\imath}(t_k)$ is the sequence of computed values during the simulation run. Figure 4-20 helps visualize the situation that we are considering.

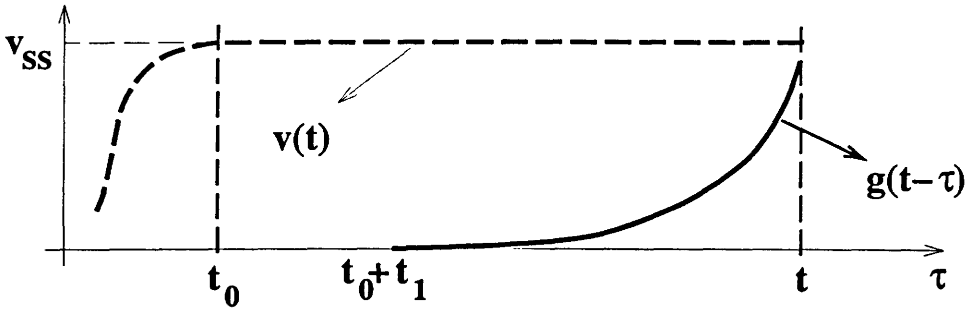

Figure 4-20: Convolution of a nearly constant waveform with a fast-decaying impulse response: convolution of $v(t)$ with $g(t)$, where $v(t)$ is approximately constant for $t > t_0$ and $g(t) \approx 0$ for $t > t_1$.

Proof. If $v(t) = v_{ss}$ is constant then, $\forall\, t$

$$i(t) = \int_{-\infty}^{t} v(t)g(t-\tau)d\tau = v_{ss}\int_{-\infty}^{t} g(t-\tau)d\tau = v_{ss}\int_{0}^{\infty} g(t)dt = v_{ss}\boldsymbol{G}(0),$$

where $\boldsymbol{G}(s)$ is the Laplace transform of $g(t)$. By hypothesis, in this case the convolution integral is computed exactly, that is

$$\hat{\imath}(t_k) = \mathcal{I}\left[(-\infty, t_k], \hat{v}(t_k), g(t_k-\tau)\right] = \mathcal{I}\left[(-\infty, t_k], v_{ss}, g(t_k-\tau)\right] = v_{ss}\boldsymbol{G}(0).$$

So the hypothesis basically translates into saying that our integration formula can compute the integral of the impulse response exactly.

Consider now the case where $v(t)$ is not constant but is such that

$$\lim_{t\to\infty} v(t) = v_{ss}, \tag{4.64}$$

where v_{ss} is a constant. Then, using the final value theorem, we know that

$$\lim_{t\to\infty} v(t) = \lim_{s\to 0} s\boldsymbol{V}(s) = v_{ss}.$$

Therefore the exact integral, as $t \to \infty$ equals

$$\lim_{t\to\infty} i(t) = \lim_{s\to 0} s\boldsymbol{I}(s) = \lim_{s\to 0} s\boldsymbol{V}(s)\boldsymbol{G}(s) = \lim_{s\to 0} s\boldsymbol{V}(s)\lim_{s\to 0}\boldsymbol{G}(s) = v_{ss}\boldsymbol{G}(0)$$

since both limits exist.

On the other hand, computationally, as $t \to \infty$, $k \to \infty$, and $\hat{v}_k = \hat{v}(t_k)$ becomes arbitrarily close to a constant, $\hat{v}_k = v_{ss} + \epsilon_k$, where ϵ_k is arbitrarily small, that is

$$\forall\, \epsilon > 0,\ \exists\, t_0 : t_k > t_0 \ \Rightarrow\ |e_k| \le \epsilon. \tag{4.65}$$

On the other hand, since $g(t) \in \mathbb{L}_1(\mathbb{R}, \mathbb{R})$ that necessarily implies that

$$\exists\, t_1 : t > t_1 \Rightarrow |g(t)| \leq \epsilon. \tag{4.66}$$

where ϵ is the same as above. Then, for $t = t_k > t_0 + t_1$, using the additivity of the integration formula we get

$$\begin{aligned} \hat{\imath}(t_k) &= \mathcal{I}\left[(-\infty, t_k], \hat{v}_k, g(t_k - \tau)\right] = \\ &\mathcal{I}\left[(-\infty, t_0], \hat{v}_k, g(t_k - \tau)\right] + \mathcal{I}\left[[t_0, t_k], \hat{v}_k, g(t_k - \tau)\right]. \end{aligned} \tag{4.67}$$

The first of these computed values is negligible according to the zero-contraction property of the integration formula:

$$\begin{aligned} \tau \in (-\infty, t_0] \; \Rightarrow t_k - \tau > t_1 \; \Rightarrow |g(t_k - \tau)| \leq \epsilon \; \Longrightarrow \\ \Longrightarrow \; |\mathcal{I}\left[(-\infty, t_0], \hat{v}_k, g(t_k - \tau)\right]| \leq \epsilon\, K, \end{aligned} \tag{4.68}$$

which is arbitrary small. Using the linearity property, the second computed value in Eqn. (4.67) leads to

$$\mathcal{I}\left[[t_0, t_k], \hat{v}_k, g(t_k - \tau)\right] = \mathcal{I}\left[[t_0, t_k], v_{ss}, g(t_k - \tau)\right] + \mathcal{I}\left[[t_0, t_k], \epsilon_k, g(t_k - \tau)\right]. \tag{4.69}$$

Again, since for $\tau \in [t_0, t_k]$ Eqn. (4.65) shows that $|\epsilon_k| \leq \epsilon$ and therefore by the zero-contraction property this integral is negligible. Finally the first value in Eqn. (4.68) leads to, using the additivity property

$$\mathcal{I}\left[[t_0, t_k], v_{ss}, g(t_k - \tau)\right] = \mathcal{I}\left[(-\infty, t_k], v_{ss}, g(t_k - \tau)\right] - \mathcal{I}\left[(-\infty, t_0], v_{ss}, g(t_k - \tau)\right]$$

where the second value was already shown to be zero in Eqn. (4.68), so

$$\lim_{k \to \infty} \hat{\imath}(t_k) = \mathcal{I}\left[(-\infty, t_k], \hat{v}_k, g(t)\right] = \mathcal{I}\left[(-\infty, t_k], v_{ss}, g(t)\right] = v_{ss}\boldsymbol{G}(0)$$

which we know to be the exact final value for $i(t)$, thus implying that

$$\lim_{k \to \infty} \hat{\imath}(t_k) = \lim_{t \to \infty} i(t) = v_{ss}\boldsymbol{G}(0).$$

□

It should now be clear why the special case of $v(t)$ being constant is an important one. Even though in reality there is always a finite simulation interval, this interval might be long enough that most waveforms will be close to steady-state and will therefore be in a situation similar to that described in Figure 4-20. It is therefore essential that the correct steady-state be computed in this case.

Satisfying the conditions of the previous result has several implications in terms of how the convolution integral must be implemented and also in terms of how the impulse response is obtained. If the device's impulse response is given by a collection of samples obtained via an inverse transform, such as inverse Fourier transform, two conditions should be satisfied. First we must ensure that the impulse response is in $\mathbb{L}_1(\mathbb{R}^n, \mathbb{R}^n)$, that is, that its computed energy integral is finite. This translates into stating that the last sample must have a value of zero. Furthermore, in order to ensure that the convolution integral is exactly computed when the controlling variable is a constant, it is necessary that the DC frequency-value be known and matched exactly. Again recurring to our frequency-dependent admittance device, this implies that $\boldsymbol{G}(0)$ be known and that the time-domain samples be such that

$$\int_0^\infty g(t) = \boldsymbol{G}(0). \tag{4.70}$$

In order to enforce (4.70) correctly, the integration must be performed using the *same* quadrature algorithm that will be used to compute the convolution integral. Typically once the time-domain samples are obtained the quadrature algorithm is used to compute the integral. Then, in order to guarantee that Eqn. (4.70) is satisfied the time-domain samples may have to be appropriately scaled.

In the case where the frequency-dependent device is modeled with a rational function approximation and the recursive convolution algorithm is used to compute the convolution integral the conditions of the theorem can be satisfied if the following two conditions are met: first the the DC frequency-value be known and matched exactly and the current timepoint contribution to the convolution integral (step 2 of Algorithm 4.6.1) must be computed exactly if $v(t)$ is a constant. The first of these conditions was already discussed in section 4.5.3: the frequency domain rational function approximation must be exact at $s = 0$, which may require a constrained minimization to be performed to recompute the residues of the approximation. The second condition is satisfied if the integration is performed as indicated in Eqn. (4.63).

4.7 Experimental Results

In this section, results are presented for an implementation of the algorithms described applied to efficient time-domain simulation of interconnect and packaging structures. The implementation is based on a modified version of SPICE3 [41], and uses a combination of sectioning, reduced-order modeling, and fast recursive convolution. We will consider two example setups. First we model and simulate an interconnect problem where drivers and receivers are connected via transmission lines with arbitrary scattering parameter descriptions. For completeness we will also apply a more traditional FFT-based method to this problem and compare the results in terms of accuracy and computational cost. Then we will consider a packaging example where the frequency dependent data is acquired with FastHenry. The example is an investigation of crosstalk between a small set of package pins connecting on-chip drivers to off-chip receivers.

4.7.1 Transient Simulation of Circuits and Interconnect

In this section, results are presented for an example involving the time-domain simulation of transmission lines with arbitrary scattering parameter descriptions. We have already seen that the reduced-order model obtained in section 4.5.4 with truncated balanced realization has a frequency-domain accuracy comparable to the more complete sectioning based model obtained in section 4.3.2, but many fewer poles. We will now show that the reduced-order model produces nearly the same time-domain waveforms as the sectioning based model and is therefore also comparable in terms of time-domain accuracy. Then we will present an example with realistic transistor drivers and receivers, to demonstrate the ability of the method to simulate complete circuit descriptions.

In Figure 4-21 we present the time-domain results of applying a 5 volt step to a 50Ω terminated transmission line with significant skin-effect. The pulse has a $1ns$ rise time, is applied at $t = 50ns$ and the delay of the line is $250ns$. In the figure, we compare the time response of the 7-th order reduced-order model with the time response obtained using the full sectioning based approximant, which has more than twenty poles. The fact that the two responses are indistinguishable in the figure shows that an excellent match has been obtained. In the same figure we show the time response obtained using a full convolution method applied to an impulse response obtained via inverse fast Fourier transform (iFFT) on 2048 frequency data points. As can be seen from the figure, the iFFT-derived response is equally accurate as expected since a fairly large number of frequency points were used. In Table 4-1 we show the CPU times required for obtaining the three time responses shown. The total number of timesteps required for obtaining the solution in the interval shown was 1004. From the results in the table, we can see that simulation of the reduced-order model is most efficient, as expected. Since the cost of recursive convolution is proportional to the number of poles in the reduced-order model, the 7-th order model is over one and a half times more efficient than the sectioning approach. Both of these methods are over an order of magnitude faster than the full convolution method which shows that the recursive convolution procedure is extremely efficient. For a simulation on a longer interval, the difference in CPU times would tend to increase since, as we saw, the cost of a recursive convolution method is linear on the number of timesteps while the cost of a full convolution method is quadratic on the number of timesteps.

Algorithm	CPU time (s)
Full convolution	133
Section-by-section	13
Reduced-order model	8

Table 4-1: CPU time comparisons for full convolution versus recursive convolution methods. Times are in seconds on a SUN IPX.

In Figure 4-23 we present the time-domain results obtained from the circuit in Figure 4-22, where the transmission line frequency response was shown previously. The driver and the load are both CMOS inverters, where the transistors are described using SPICE3's default level 2 model with $W/L = 750$ for the p-type pull-up devices and $W/L = 400$ for the n-type pull-down devices. The simulation results clearly show that the improper line termination causes reflections to transmit back and forth on the line and falsely trigger the load inverter.

4.7.2 Transient Simulation of Circuits and Coupling 3-D Packaging

In this section we describe an example that demonstrates the value of using the reduced-order models with the frequency dependent data obtained from a packaging problem. The frequency dependent resistance and inductance matrices describing the terminal behavior of a set of conductors can be rapidly computed with the multipole-accelerated mesh-formulation approach as implemented in FASTHENRY [42, 43]. FASTHENRY is a program that allows the efficient computation of frequency dependent inductances and resistances for complex three-dimensional geometries of conductors. It is based on a mesh analysis equation formulation technique which combined with a multipole-accelerated Generalized Minimal Residual (GMRES) matrix solution algorithm is used to compute the inductance and resistance matrices in nearly order n time and memory where n is the number of volume-filaments. Previous approaches have required order n^3 time and order n^2 memory [44]. The example that follows is an investigation of crosstalk between

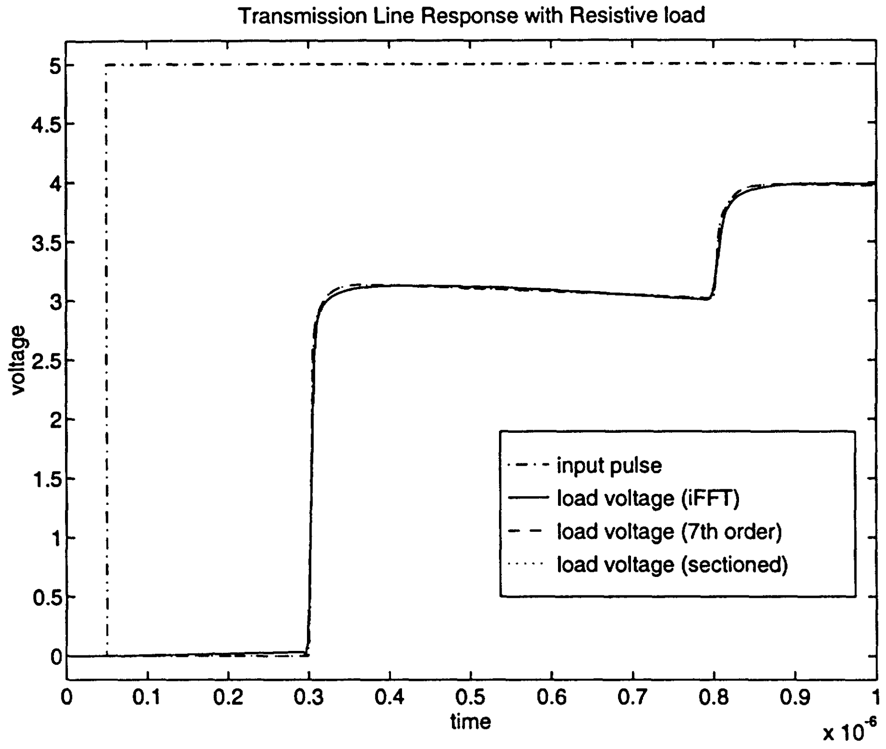

Figure 4-21: Time response obtained from applying a $5V$ pulse with a $1ns$ rise time at $t = 50ns$ to a resistively terminated transmission line. The figure shows the response of a line modeled with a 7 pole reduced-order model and that of a line modeled with the approximation resulting from our sectioning algorithm, which has more than 20 poles. The figure also shows the response of the line computed using full convolution with an impulse response obtained via inverse fast Fourier transform. For this examples 2048 frequency points were used for the iFFT algorithm. The three waveforms are indistinguishable. The delay of the transmission line is $250ns$.

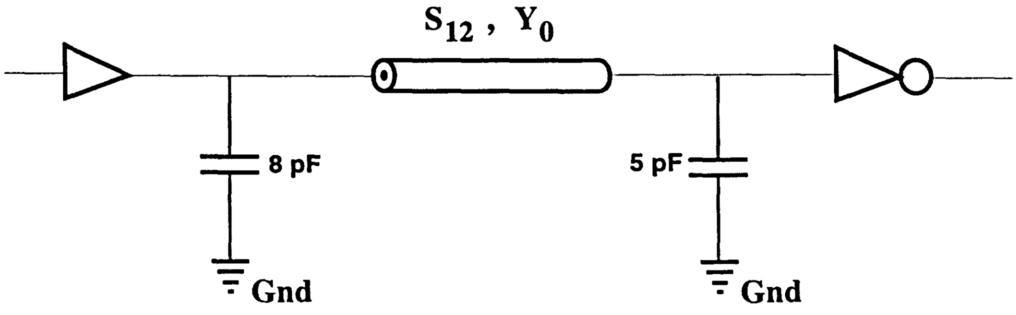

Figure 4-22: CMOS driver and load connected by a transmission line with skin-effect.

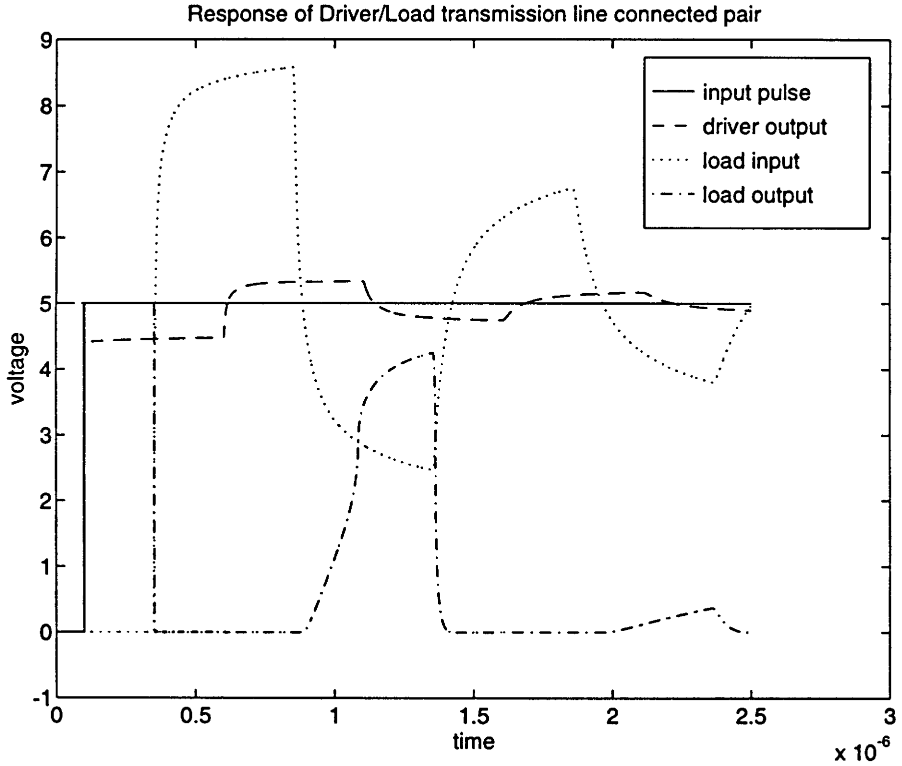

Figure 4-23: Time response of a nonlinear circuit with a transmission line connecting driver and load. The transmission line is modeled with a 7 pole reduced-order model.

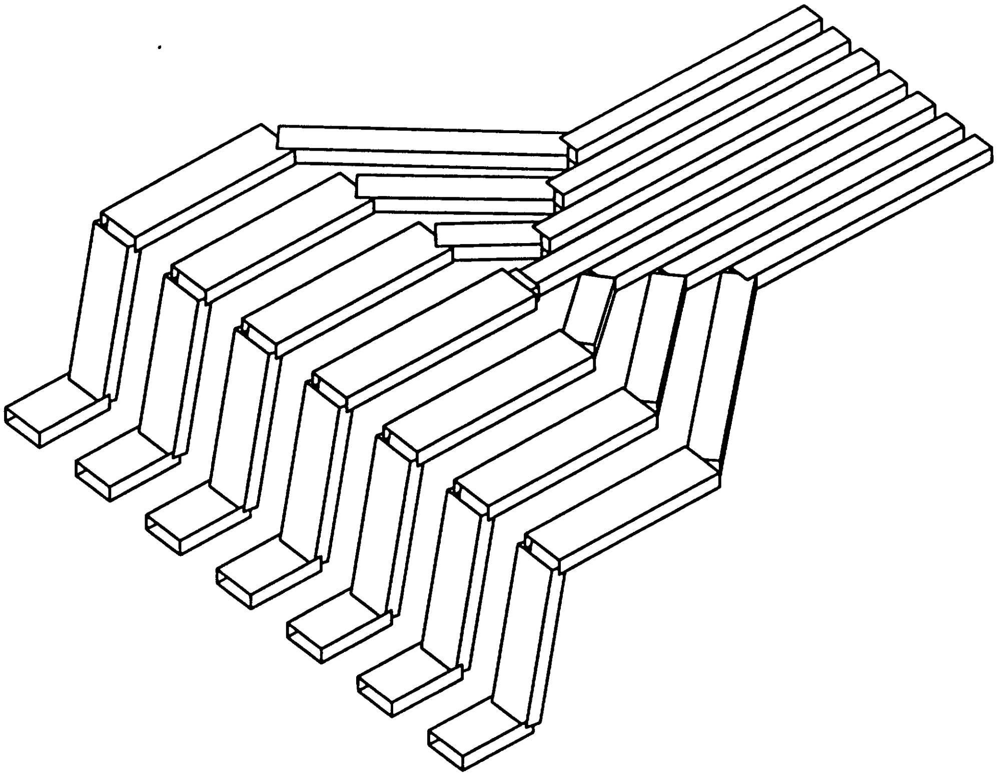

Figure 4-24: Seven pins of a cerquad pin package.

a small set of package pins connecting on-chip drivers to off-chip receivers.

Consider the crosstalk between seven adjacent pins of a 68-pin cerquad package as shown in Fig. 4-24. Assume the five middle lines carry output signals from the chip and the two outer pins carry power and ground. The signals are driven and received with CMOS inverters. The drivers are capable of driving a large current to compensate for the impedance of the package pins. The inductance of the pins is computed with FastHenry and the capacitance is assumed to be 8pF. The interconnect from the end of pin to the receiver is modelled with a capacitance of 5pF. The overall configuration is illustrated in Fig. 4-25 and a more detailed view for a single pin is given in Fig. 4-26. A $0.1\mu F$ decoupling capacitor is connected between the driver's power and ground to minimize supply fluctuations.

To compute the resistance and inductance matrix at each frequency with FastHenry, the pins were discretized into five filaments along their height and nine along their length. This allows accurate modeling of changes in resistance and inductance due to skin and proximity effects. Using FastHenry matrices were generated for the frequency range 1MHz to 10MHz, with three points per decade. For more details see [45].

The frequency dependence of each element in the admittance matrix is fit with a

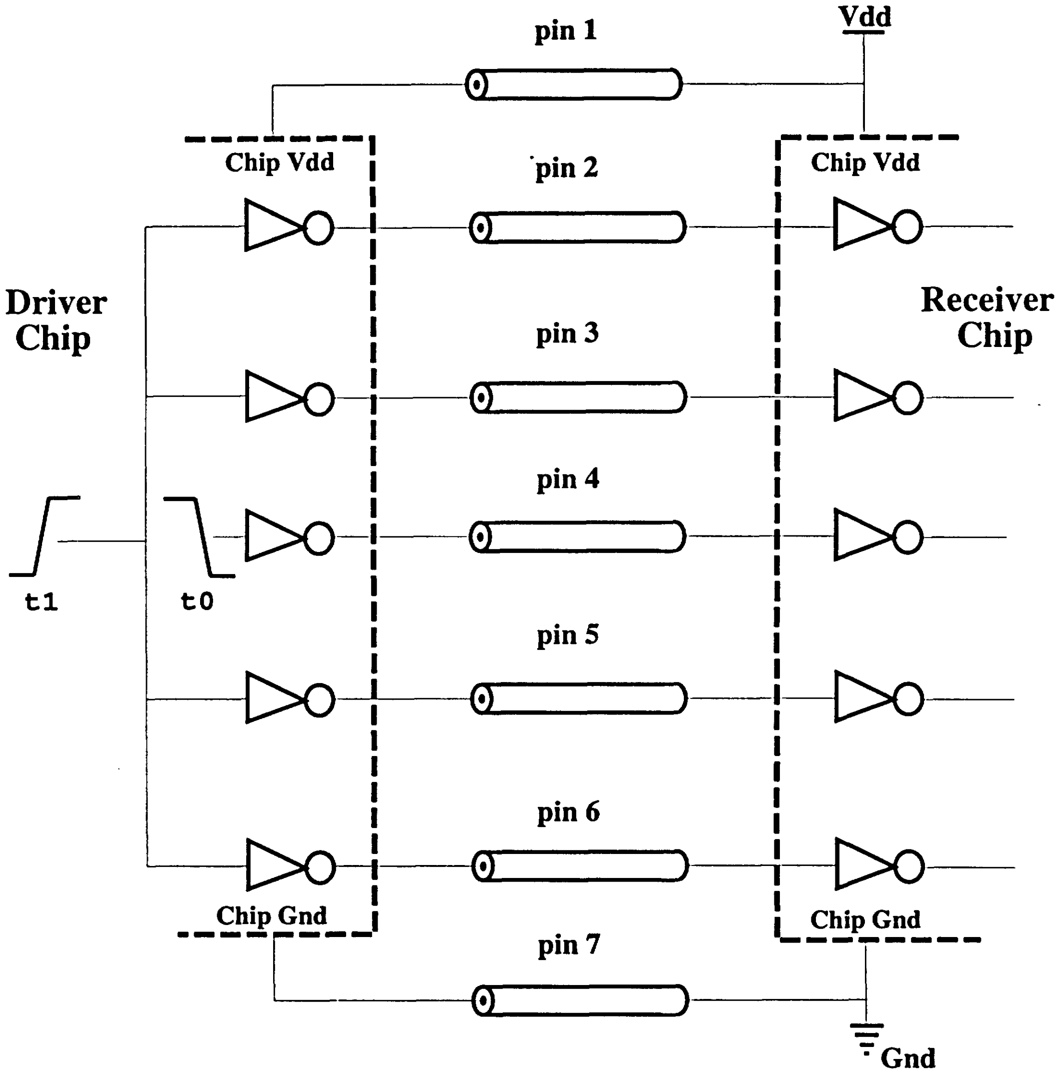

Figure 4-25: General configuration for the connection between received and driver chips. All the circuit elements inside the same chip share that chip's power and ground.

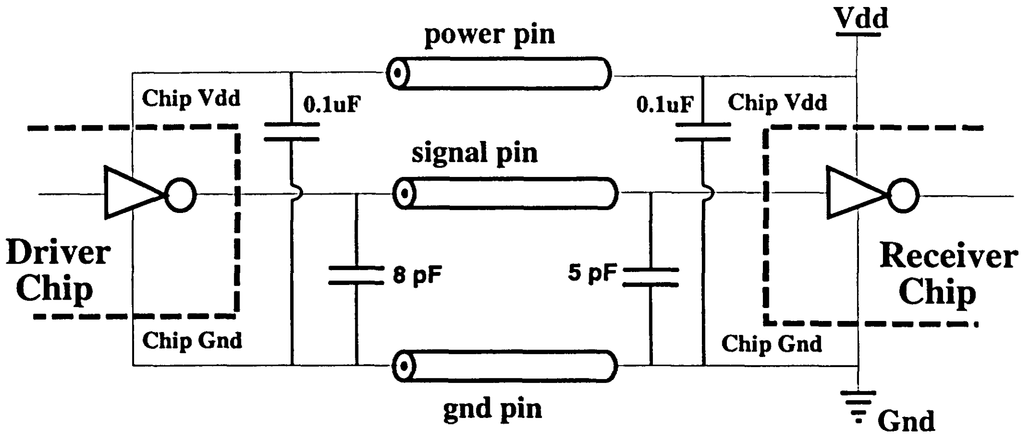

Figure 4-26: Detailed view of the connection between driving and receiving chips, showing the power and ground connections. Decoupling capacitance between the power and ground lines are also shown. Pin capacitance and receiver interconnect capacitance are also modeled as small capacitive loads.

rational approximation using the algorithms described in Section 4.3. First, the section-by-section approach is used to obtain approximations which have orders in the range of 12 to 24. Following the section-by-section algorithm a realization is determined and balanced. We have found that truncated models of 3^{rd} order are sufficiently accurate to provide approximation with less than 5% error. The following two figures demonstrate this fact. Figure 4-27 shows the magnitude of the self-admittance term at pin 4. Shown in the plot are data points computed with FastHenry, the 12^{th} order section-by-section approximant and the 3^{rd} order reduced model computed by truncating the balanced realization. As can be seen on the plot, the three curves match each other almost perfectly. Figure 4-28 shows the magnitude of the mutual admittance term between pins 3 and 4. Again, shown in the plot are data points computed with FastHenry, a 20^{th} order section-by-section approximant and the 3^{rd} order reduced model. As on the previous plot, the three curves match each other almost perfectly. Similar accuracy is also evident in all of the remaining matrix entries.

The reduced-order model for each entry in the admittance matrix is incorporated into SPICE3 as a frequency-dependent voltage-controlled current source VCCS. As a sample time domain simulation, imagine that at time $t_0 = 4$ns the signal on pin 4 of Fig.4-25 is to switch from high to low and pins $2, 3, 5$, and 6 are to switch from low to high but that due to delay on chip, pins $2, 3, 5$, and 6 switch at $t_1 = 5$ns. In this case, significant current will suddenly pass through the late pins while pin 4 is in transition. Due to crosstalk, this large transient of current has significant effects on the input of the receiver on pin 4, as shown in Fig. 4-29. Note that the input does not rise monotonically. Fig. 4-29 also

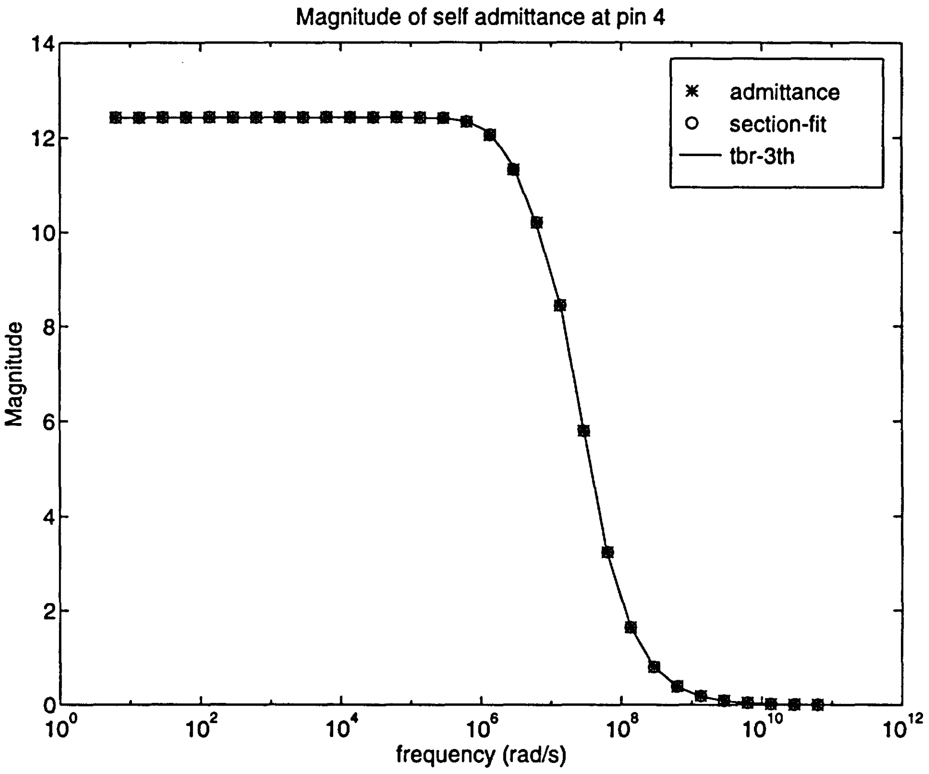

Figure 4-27: Magnitude of the self-admittance term at pin 4. Shown in the plot are data points computed with FastHenry, the 12^{th} order section-by-section approximant and the 3^{rd} order reduced model computed by truncating the balanced realization. The error in both approximations is less than 0.5%.

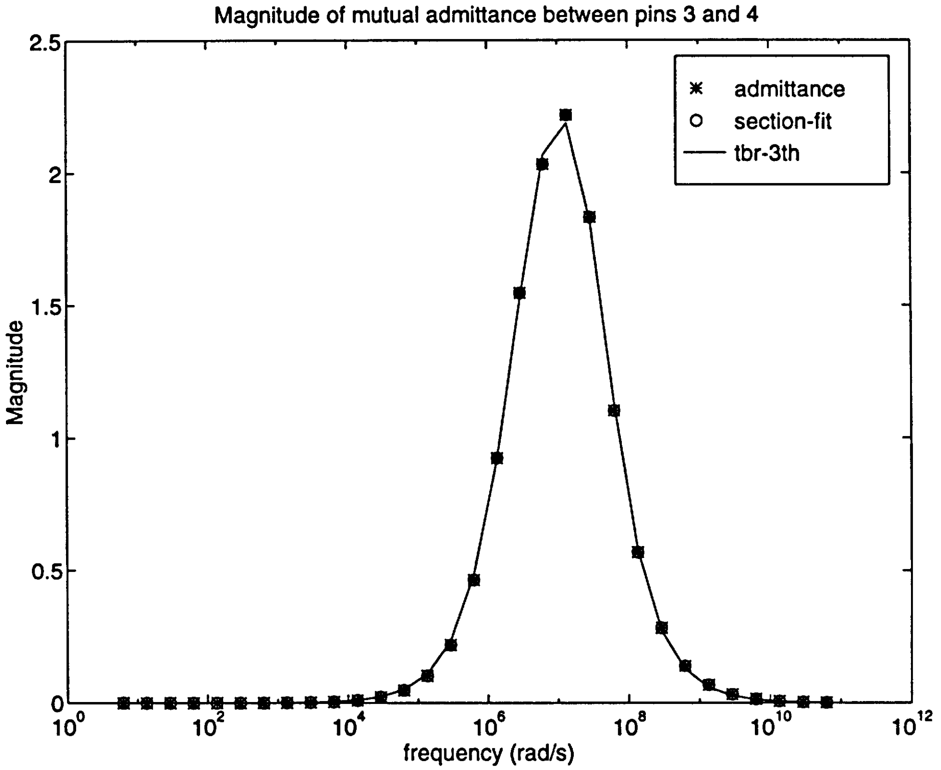

Figure 4-28: Magnitude of the mutual admittance term between pins 3 and 4. Shown in the plot are data points computed with FastHenry, a 20^{th} order section-by-section approximant and the 3^{rd} order reduced model. The error in both approximations is less than 1%.

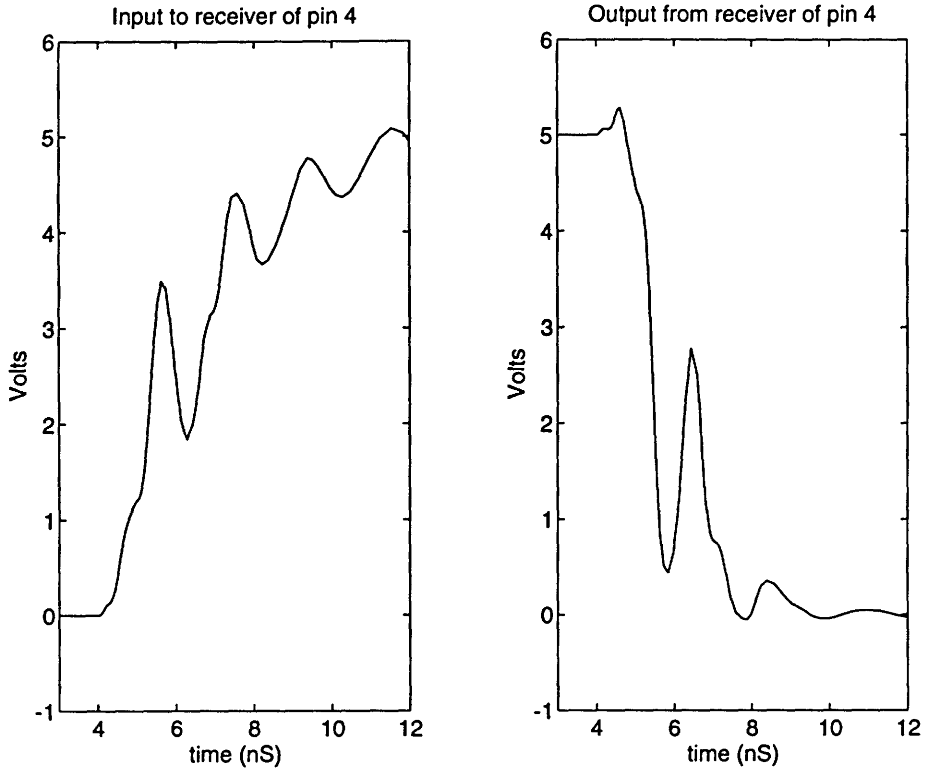

Figure 4-29: Results of the timing simulation of a receiver pin in the presence of changes on other adjacent pins. Pin 4's receiver when four adjacent pins switch 1ns after pin 4.

shows that the bump in the waveform is carried through to the output of receiver, as a large glitch.

Now consider changing the design by swapping the ground pin, pin 7, with signal pin 5. Now the ground pin sits between signal lines and adds greater separation between pin 4 and the signals which are now on lines 6 and 7. As might be expected, the crosstalk is significantly reduced and the voltage bump does not exceed $1.5V$ as shown in Fig.4-30.

4.8 Conclusions

In this chapter, we have proposed a robust algorithm for conjuring up stable, low-order, accurate, frequency-domain models for transmission lines based on realistic scattering data.

The algorithm described is based on a two-step procedure. In the first step of our algorithm, a stable, high-order transfer function is fitted to the scattering data using a section-by-section algorithm. For that purpose The frequency range is sectioned, and a section-by-section constrained ℓ_2, forced stable rational function approximation is fitted

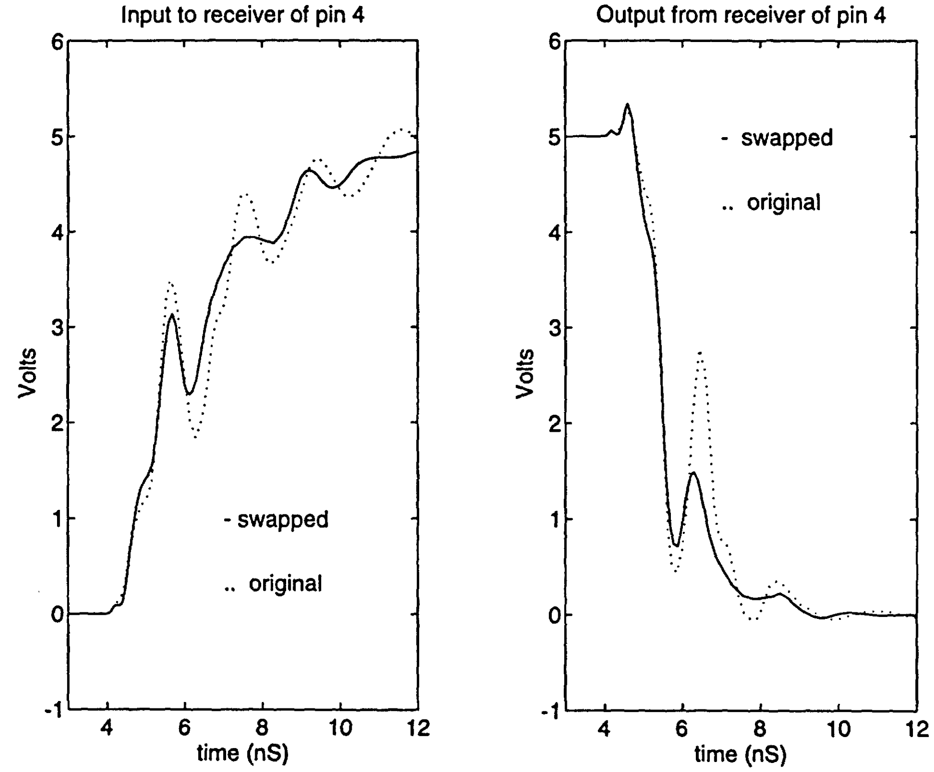

Figure 4-30: Results of the timing simulation of a receiver pin in the presence of changes on other adjacent pins. Pin 4's receiver with ground pin 7 and signal line 5 swapped.

to the data in each frequency section. Then the section transfer functions are combined using a global ℓ_2 criterion to obtain a stable, accurate, high order model valid for the whole frequency range. In the second step of the algorithm, a guaranteed stable, low-order model is obtained from the high-order model using the method of truncated balanced realizations. Finally, the DC gain of the low-order model and the correct steady-state are matched to those of the full model using a constrained global ℓ_2 minimization scheme.

We have shown that our section-by-section approximation is very accurate and that the final stable low-order approximation derived using the truncated balanced realization has excellent match with the frequency response of the full model. The resulting rational transfer function was incorporated into a circuit simulator, and the numerical experiments that we have conducted on several interconnect and packaging example problems have shown that it produces time-domain responses that match those obtained using the more computationally expensive convolution procedures currently in use for transmission line simulations.

Several aspects of the algorithm could be improved and deserve further research. Currently, in the section-by-section algorithm, a forced stable rational function is obtained from the local minimization approximation. While this has shown to produce accurate results in most cases, it would be desirable to devise a more robust sectioning technique that need not sacrifice accuracy for stability, since the order of the approximation at this stage is not important. Other points that deserve further attention are related to ensuring that the correct steady-state is maintained after the truncation of the balanced realization. Currently this is done with an a-posteriori global minimization that recomputes the residues of the approximation. Also, in certain situations, it is necessary to make sure that the state initial conditions are satisfied, a problem for which a more satisfactory solution than the one used in this thesis should be sought. Another point deserving further study is to develop an automatic way to parameterize the balanced realization in order to obtain time-domain error bounds on the approximation. Finally, the study of methods that could improve the computational cost of obtaining a truncated balanced realization should be pursued, as this would substantially increase the applicability of the model order reduction techniques described. Some research has already been conducted in this area but no definite algorithm exists at this point.

References

[1] J. E. Schutt-Aine and R. Mittra, "Scattering Parameter Transient Analysis of Transmissions Lines loaded with Nonlinear Terminations," *IEEE Transactions on Microwave Theory and Techniques*, vol. MTT-36, pp. 529–536, 1988.

[2] S. Lin and E. S. Kuh, "Transient Simulation of Lossy Interconnects Based on the Recursive Convolution Formulation," *IEEE Trans. Circuits Syst.*, vol. 39, pp. 879–892, November 1992.

[3] B. Moore, "Principal Component Analysis in Linear Systems: Controllability, Observability, and Model Reduction," *IEEE Transactions on Automatic Control*, vol. AC-26, pp. 17–32, February 1981.

[4] K. Glover, "All optimal Hankel-norm approximations of linear multivariable systems and their l^∞-error bounds," *International Journal on Control*, vol. 39, pp. 1115–1193, June 1984.

[5] F.-Y. Chang, "Waveform Relaxation Analysis of RLGC Transmission Lines," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1394–1415, November 1990.

[6] J. S. Roychowdhury and D. O. Pederson, "Efficient Transient Simulation of Lossy Interconnect," in 28th *ACM/IEEE Design Automation Conference*, pp. 740–745, June 1991.

[7] J. R. Griffith and M. S. Nakhla, "Time-Domain Analysis of Lossy Coupled Transmission Lines," *IEEE Transactions on Microwave Theory and Techniques*, vol. 38, pp. 1480–1487, October 1990.

[8] A. R. Djordjevic, T. K. Sarkar, and R. F. Harrington, "Analysis of Lossy Transmission Lines with Arbitrary Nonlinear Terminal Networks," *IEEE Transactions on Microwave Theory and Techniques*, vol. MTT-34, pp. 660–666, June 1986.

[9] J. S. Roychowdhury, A. R. Newton, and D. O. Pederson, "An Impulse-Response based Linear Time-Complexity Algorithm for Lossy Interconnect Simulation," in *International Conference on Computer Aided-Design*, pp. 62–65, November 1991.

[10] R. Wang and O. WIng, "Analysis of VLSI Multiconductor Systems by By-Level Waveform Relaxation," in *International Conference on Computer Aided-Design*, pp. 166–169, November 1991.

[11] S. P. McCormick, *Modeling and Simulation of VLSI Interconnections with Moments.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1989.

[12] H. M. Paynter, "ON an Analogy between Stochastic Processes and Monotone Dynamic Systems," in *Regelungstechnik. Moderne Theorien und ihre Verwendbarkeit* (G. Müller, ed.), pp. 243–250, R. Oldenbourg, 1956.

[13] L. G. Gbilaro and F. P. Lees, "The Reduction of Complex Transfer Function Models to Simple Models using the Method of Moments," *Chemical Engineering Science*, vol. 24, pp. 85–93, 1969.

[14] F. P. Lees, "The determination of the Moments of the Impulse Response of Chemical Processes from the basic transformed equations," *Chemical Engineering Science*, vol. 24, pp. 1607–1613, 1969.

[15] F. P. Lees, *Unsteady State Models of Gas Absorption Columns.* PhD thesis, Loughborough University of Technology, U.K., 1969.

[16] M. J. Bosley and F. P. Lees, "A Survey of Simple Transfer-Function Derivations from High-Order State-Variable Models," *Automatica*, vol. 8, pp. 765–775, 1972.

[17] L. T. Pillage, X. Huang, and R. A. Rohrer, "AWEsim: Asymptotic Waveform Evaluation for Timing Analysis," in 26th *ACM/IEEE Design Automation Conference*, (Las Vegas, Nevada), pp. 634–637, June 1989.

[18] G. A. Baker Jr., "The Numerical Calculation of Padé Approximants," in *Proceeding of the Antwerpen Conference* (L. Wuytack, ed.), pp. 231–245, Springer-Verlag, 1979.

[19] G. A. Baker Jr., *Essentials of Padé Approximants.* New York, NY: Academic Press, First ed., 1975.

[20] G. A. Baker Jr. and P. Graves-Morris, *Padé Approximants Part I: Basic Theory.* Encyclopedia of Mathematics and its Applications, Reading, MA: Addison-Wesley Publishing Company, First ed., 1981.

[21] J. Karlssson, "Rational Interpolation and Best Rational Approximation," *SIAM J. Matrix Anal. Appl.*, vol. 53, pp. 38–52, 1976.

[22] D. S. Lubinsky, "Divergence of Complex Rational Approximations," *Pacific Journal of Mathematics*, vol. 108, no. 1, pp. 141–153, 1983.

[23] H. Wallin, "The Convergence of padÉ Approximants and the Size of the Power Series Coefficients," *Applicable Analysis*, vol. 4, pp. 235–251, 1975.

[24] H. Stahl, "Orthogonal Polynomials with Complex-Valued Weight Functions, I," *Constructive Approximation*, vol. 2, pp. 225–240, March 1986.

[25] H. Stahl, "Orthogonal Polynomials with Complex-Valued Weight Functions, II," *Constructive Approximation*, vol. 2, pp. 241–251, March 1986.

[26] D. S. Lubinsky, "Distribution of Poles of Diagonal Rational Approximants to Functions of Fast Rational Approximability," *Constructive Approximation*, vol. 7, pp. 501–519, April 1991.

[27] X. Huang, *Padé Approximation of Linear(ized) Circuit Responses.* PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellong University, Pittsburgh, PA, November 1990.

[28] D. F. Anastasakis, N. Gopal, S. Y. Kim, and L. T. Pillage, "On the Stability of Moment-Matching Approximations in Asymptotic Waveform Evaluation," in 29th *ACM/IEEE Design Automation Conference*, (Anaheim, California), pp. 207–212, June 1992.

[29] L. N. Trefethen and M. H. Gutknecht, "On Convergence and Degeneracy in Rational Padé and Chebyshev Approximations," *SIAM Journal on Numerical Analysis*, vol. 16, pp. 198–210, January 1985.

[30] L. N. Trefethen and M. H. Gutknecht, "The Carathéodory-Fejér Method for Real Rational Approximation," *SIAM Journal on Numerical Analysis*, vol. 20, pp. 420–435, April 1985.

[31] E. Chiprout and M. Nakhla, "Generalized Moment-Matching Methods for Transient Analysis of Interconnect Networks," in 29^{th} *ACM/IEEE Design Automation Conference*, (Anaheim, California), pp. 201–206, June 1992.

[32] T. Kailath, *Linear Systems.* Information and System Science Series, Englewood Cliffs, New Jersey: Prentice-Hall, First ed., 1980.

[33] L. Pernebo and L. M. Silverman, "Model Reduction via Balanced State Space Representations," *IEEE Transactions on Automatic Control*, vol. AC-27, April 1982.

[34] A. J. Laub, "Computation of "Balancing" Transformations," in *Proceedings of the Joint Automatic Control Conference*, (San Francisco, CA), August 1980.

[35] R. H. Bartels and G. H. Stewart, "Solution of the matrix equation $ax + xb = c$," *Communications of the ACM*, vol. 15, pp. 820–826, 1972.

[36] G. H. Golub and C. F. Van Loan, *Matrix Computations.* Baltimore, Maryland: The John Hopkins University Press, 1983.

[37] P. T. Kambala, "Balanced Gains and Their Significance for L^2 Model Reduction," *IEEE Transactions on Automatic Control*, vol. AC-30, pp. 690–693, July 1985.

[38] P. T. Kambala, "Balanced Forms: Canonicity and Parametrization," *IEEE Transactions on Automatic Control*, vol. AC-30, pp. 1106–1109, November 1985.

[39] L. M. Silveira, I. M. Elfadel, and J. K. White, "A Guaranteed Stable Order Reduction Algorithm for Packaging and Interconnect Simulation," in *Proceedings of the* 2^{nd} *Topical Meeting on Electrical Performance of Electronic Packaging*, (Monterey, California), pp. 165–168, October 1993.

[40] L. M. Silveira, I. M. Elfadel, and J. K. White, "Efficient Frequency-Domain Modeling and Circuit Simulation of Transmission Lines," *Submitted to the special issue of the IEEE Transactions on Components, Hybrids, and Manufacturing Technology on the Electrical Performance of Electronic Packaging*, 1994.

[41] T. L. Quarles, "The SPICE3 Implementation Guide," Tech. Rep. ERL M89/44, Electronics Research Laboratory Report, University of California at Berkeley, Berkeley, California, April 1989.

[42] M. Kamon, M. Tsuk, C. Smithhisler, and J. White, "Efficient techniques for inductance extraction of complex 3-d geometries," in *Proceedings of the Int. Conf. on Comp. Aided Design*, November 1992.

[43] M. Kamon, M. J. Tsuk, and J. White, "Fasthenry, a multipole-accelerated 3-d inductance extraction program," in *Proceedings of the ACM/IEEE Design Automation Conference*, (Dallas), June 1993.

[44] M. Kamon, "Efficient techniques for inductance extraction of complex 3-d geometries," Master's thesis, Massachusetts Institute of Technology, February 1994.

[45] L. M. Silveira, M. Kamon, and J. K. White, "Algorithms for Coupled Transient Simulation of Circuits and Complicated 3-D Packaging," in *Proceedings of the* 44^{st} *Electronics Components and Technology Conference*, (Washington, DC), pp. 962–970, May 1994.

5

Conclusions

This thesis presents theoretical and practical aspects of model order reduction techniques for use in the context of circuit simulation. The observation was made in the introductory chapter, that improving the efficiency of the simulation process can only be accomplished by devising specific algorithms that directly exploit the characteristic features of each particular problem. To this end two different types of approach were considered. First, model order reduction techniques were applied to the simulation of clocked analog circuits. Second, model order reduction was used as a modeling technique for frequency-dependent interconnect and packaging structures.

Simulating the transient behavior of clocked analog circuits is computationally expensive because these circuits are clocked at a frequency whose period is orders of magnitude smaller than the time interval of interest to the designer. It is possible to improve simulation efficiency by exploiting the property that the behavior of the circuit state variables in a given high-frequency clock cycle is similar to the behavior in preceding and following cycles. The Envelope-Following technique studied in this thesis reduces the simulation time without compromising accuracy by accurately computing the envelope of the state variables while simulating the circuit behavior over occasional cycles. Efficiency can be further improved by close examination of the sensitivity information obtained during the simulation of one cycle. Based on this data a partitioning of the circuit variables is possible, such that only those variables that contain state information are envelope-followed. Speedups of over an order of magnitude over standard circuit simulation are reported for several classes of clocked analog circuits.

Also, a robust algorithm for conjuring up stable, low-order, accurate, frequency-domain models for interconnect and packaging problems was presented. This algorithm is based on a section-by-section fitting of measured or tabulated frequency data, followed by a robust guaranteed stable truncation procedure based on the method of balanced realizations. It was shown that both the original approximant and its reduced-order model have excellent match with the frequency response of the full models. The resulting rational transfer function is incorporated into a circuit simulator, and the numerical experiments conducted on several interconnect and packaging example problems have shown that it produces accurate time-domain responses. Furthermore, the use of recursive convolution techniques provides a computational advantage over the more computationally expensive full convolution procedures currently in use for interconnect simulation.