Stability-Preserving Model Reduction for Linear and Nonlinear Systems Arising in Analog Circuit Applications

by

Bradley Neil Bond

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2010

Author .....
Department of Electrical Engineering and Computer Science
January 28, 2010

Certified by .....
Luca Daniel
Associate Professor
Thesis Supervisor

Accepted by .....
Terry P. Orlando
Chairman, Department Committee on Graduate Theses

Stability-Preserving Model Reduction for Linear and Nonlinear Systems Arising in Analog Circuit Applications

by

Bradley Neil Bond

Submitted to the Department of Electrical Engineering and Computer Science
on January 28, 2010, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Despite the increasing presence of RF and analog components in personal wireless electronics, such as mobile communication devices, the automated design and optimization of such systems is still an extremely challenging task. This is primarily due to the presence of both parasitic elements and highly nonlinear elements, which makes simulation computationally expensive and slow. The ability to generate parameterized reduced order models of analog systems could serve as a first step toward the automatic and accurate characterization of geometrically complex components and subcircuits, eventually enabling their synthesis and optimization. This thesis presents techniques for reduced order modeling of linear and nonlinear systems arising in analog applications. Emphasis is placed on developing techniques capable of preserving important system properties, such as stability, and parameter dependence in the reduced models.

The first technique is a projection-based model reduction approach for linear systems aimed at generating stable and passive models from large linear systems described by indefinite, and possibly even mildly unstable, matrices. For such systems, existing techniques are either prohibitively computationally expensive or incapable of guaranteeing stability and passivity. By forcing the reduced model to be described by definite matrices, we are able to derive a pair of stability constraints that are linear in terms of projection matrices. These constraints can be used to formulate a semidefinite optimization problem whose solution is an optimal stabilizing projection framework.

The second technique is a projection-based model reduction approach for highly nonlinear systems that is based on the trajectory piecewise linear (TPWL) method. Enforcing stability in nonlinear reduced models is an extremely difficult task that is typically ignored in most existing techniques. Our approach utilizes a new nonlinear projection in order to ensure stability in each of the local models used to describe the nonlinear reduced model. The TPWL approach is also extended to handle parameterized models, and a sensitivity-based training system is presented that allows us to efficiently select inputs and parameter values for training.

Lastly, we present a system identification approach to model reduction for both linear and nonlinear systems. This approach utilizes given time-domain data, such as input/output

samples generated from transient simulation, in order to identify a compact stable model that best fits the given data. Our procedure is based on minimization of a quantity referred to as the 'robust equation error', which, provided the model is incrementally stable, serves as up upper bound for a measure of the accuracy of the identified model termed 'linearized output error'. Minimization of this bound, subject to an incremental stability constraint, can be cast as a semidefinite optimization problem.

Thesis Supervisor: Luca Daniel
Title: Associate Professor

Acknowledgments

First and foremost I would like to thank my advisor, Professor Luca Daniel, for all of his help and support over the years. Luca was extremely patient with me early on, despite my circuit ignorance, and spent many hours helping me understand a research area that I have grown to love. Later on, when I was more independent in my research, Luca continued to provide support and advice, and has always encouraged me to work on problems that I enjoy. For all of this I am truly grateful.

I have also received valuable support from several other faculty members during my time at MIT. Over the last year I have had the pleasure of working very closely with Professor Alex Megretski, to whom I am very grateful for all of his time and patience. Professors Vladimir Stojanovic and Jacob White have provided valuable feedback on my research as members of my thesis committee, and Dr. Yehuda Avniel has provided much encouragement and advice.

Working in the Computational Prototyping Group has been a great pleasure because of all the wonderful people I have worked with. I would like to thank Kin Cheong Sou for many great conversations and many great times together, Tarek El-Moselhy for political conversations, Homer Reid for basketball conversations, Laura Proctor for never letting our office become a quiet boring place, Lei Zhang and Bo Kim for organizing many great social events, Yu-Chung Hsiao for the delicious foreign treats, Zohaib Mahmood for collaborating on recent projects, Dima Vasilyev for letting me destroy an old fence at his house, Junghoon Lee for being a great officemate, and also Omar Mysore, Tom Klemas, Carlos Coelho, Dave Willis, Jay Bardhan, Xin Hu, and Shih-Hsien Kuo.

In addition to Prof. Megretski, Prof. Stojanovic and Dr. Avniel, I have also collaborated closely with Zohaib Mahmood, Yan Li and Ranko Sredojevic on all of the work presented in Chapter 7.

I would like to thank the Center for Computational Engineering and Miltos Kambourides for awarding me the Kambourides fellowship in computational engineering, which has supported my research over this last year.

Lastly, but most importantly, I would like to thank my family for all of their help and

encouragement along the way. Without their support, none of this would have been possible.

Contents

List of Figures

List of Tables

Notation and Definitions

N, q	Order of large system and reduced system, respectively
$x \in \mathbb{R}^N$	x is a real vector of length N, typically the state vector of a large system
$A \in \mathbb{R}^{N \times N}$	A is an $N \times N$ real matrix
$\hat{x} \in \mathbb{R}^q$	State vector for a reduced system
$U, V \in \mathbb{R}^{N \times q}$	Left and right projection matrices, respectively
$L(x)$	Scalar storage function used to prove dissipation
$H(s), H(j\omega)$	Transfer function for a linear system
$h(x), H(x)$	Scalar storage function and matrix storage function (used only in Chapter 7)
$\sigma(u, x)$	Supply rate corresponding to a type of dissipation
$f(x) \succ 0$	f is a positive-definite function, i.e. $f(0) = 0$, and $f(x) > 0$ for $x \neq 0$
$f(x) \succeq 0$	f is a positive semidefinite function, i.e. $f(x) \geq 0$ for all x
$A \succ 0$	A is a positive-definite matrix, i.e. $x^T A x \succ 0$
$A \succeq 0$	A is a positive semidefinite matrix, i.e. $x^T A x \succeq 0$
SPD	Symmetric positive definite, e.g. matrix A is SPD if $A = A^T \succ 0$
PSD	Positive semi-definite e.g. matrix A is PSD if $A \succeq 0$
SPSD	Symmetric positive semi-definite e.g. matrix A is PSD if $A = A^T \succeq 0$
TPWL	Trajectory piecewise linear
EDA	Electronic design automation
MEMS	Micro-electro-mechanical system
SDP	Semidefinite program

Chapter 1

Introduction

1.1 Modeling and Simulation of Analog and RF Systems

1.1.1 Motivation

With the increasing presence of RF components in personal wireless electronics, such as cellular phones and GPS navigation hand-helds, the need for fast simulation of such systems is increasing and will be around for years to come. Additionally, communication systems are becoming smaller and more sophisticated, often implemented as System-on-Chips (SoC), requiring increasingly optimized designs for size and efficiency.

Despite their prevalence today, the automated design and optimization of RF and analog circuits, such as the RF receiver front-end shown in Figure 1-1, is still an extremely challenging task. This is primarily due to the presence of both parasitic elements, such as RF inductors, which create unwanted electromagnetic coupling within the system, and nonlinear elements, such as MOSFETs, which are described by highly nonlinear equations containing hundreds of parameters. A model that can accurately capture all of this behavior may contain several hundred thousand densely coupled nonlinear equations, making simulation computationally expensive and slow. The design of such systems is typically performed by a designer using a combination of intuition resulting from years of experience, and many hours of SPICE simulations. Although the current models are computationally inefficient in their present forms, this design task may be facilitated by the automatic extraction of

parameterized macromodels for such systems.

The ability to generate Parameterized Reduced Order Models (PROM) of nonlinear dynamical systems could serve as a first step toward the automatic and accurate characterization of geometrically complex components and subcircuits, eventually enabling their synthesis and optimization. The resulting compact models may also be used for higher level design optimization as well, enabling for instance the synthesis of an RF receiver chain, shown in Figure 1-1.

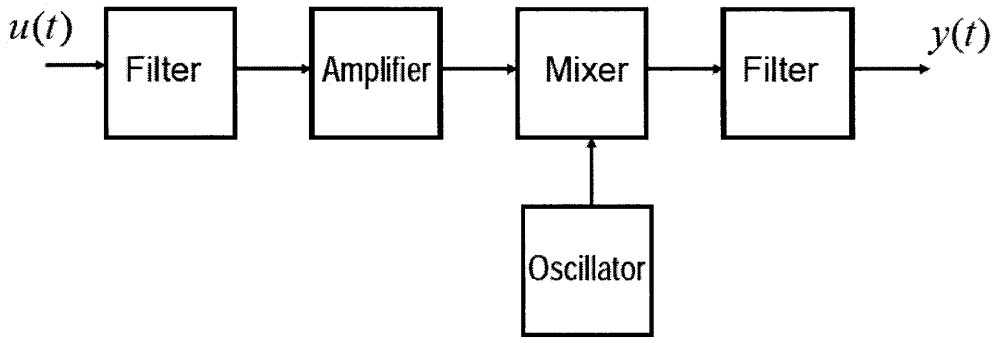


Figure 1-1: RF receiver chain

1.1.2 Challenges

Historically, the simulation of RF circuits has been an extremely difficult task. This is due to the presence of both highly nonlinear elements, such as MOSFETs, and parasitic elements, such as RF inductors. Consider the amplifier model block in Figure 1-1, which may be realized using the low noise amplifier shown in Figure 1-2, containing both nonlinear elements and parasitic elements.

Parasitic elements create unwanted electromagnetic coupling between the circuit blocks and the substrate. At high frequency, such components must be modeled with distributed elements, typically obtained by discretizing Maxwell's equations over the device geometry, resulting in possibly millions of ordinary differential equations (ODEs). Together, this makes block-level modeling insufficient to capture the non-negligible inter-block effects. It is possible to capture all coupling effects by creating a single model for the en-

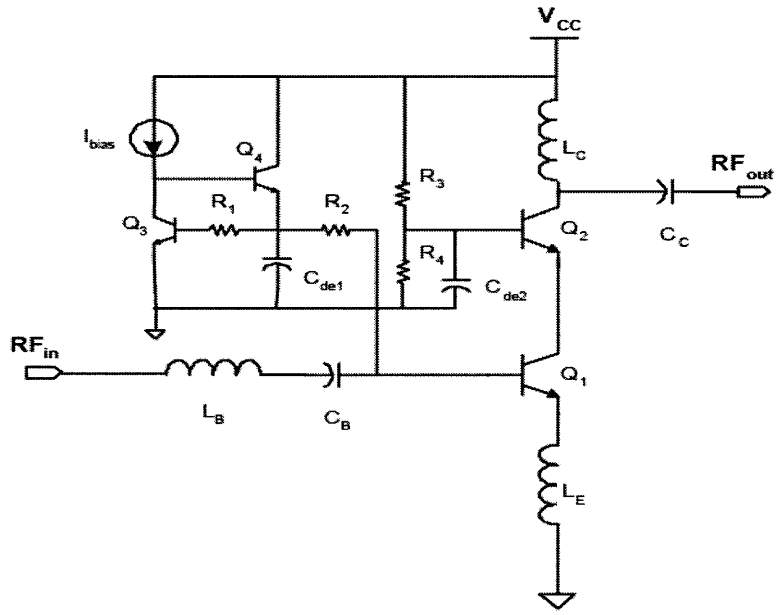


Figure 1-2: Schematic of a low-noise amplifier that may be contained in an RF receiver chain.

tire block chain, but this becomes extremely expensive computationally and results in extremely large-order systems.

Nonlinear devices are prevalent in analog circuits functioning both as linear elements and as nonlinear elements. Nonlinear blocks such as mixers are designed to behave in a strongly nonlinear manner, and thus must be represented with highly nonlinear models. Other blocks, such as amplifiers, contain nonlinear elements but are designed to behave in a linear manner. However, these blocks always exhibit nonlinear effects in some range of operation, and thus despite the desired linear performance it may be necessary to use nonlinear models to fully capture the behavior of the block. Furthermore, many of these nonlinear elements are described by extremely complicated device relations. For instance, the BSIM4 model for a MOSFET contains strong nonlinearities and hundreds of parameters, making its evaluation extremely computationally expensive relative to simple linear devices.

Consider, for example, the pair of interconnected blocks depicted in Figure 1-3, which may represent part of an RF receiver chain, and can in principle be modeled separately using the the following state-space models

$$\begin{aligned} C_f\dot{x}_f + G_f x_f &= b_f u & w &= c_f(x_f, u) \\ \dot{q}_a(x_a) + i_a(x_a) &= b_a(w) & y &= c_a(x_a, w), \end{aligned}$$

where $u(t)$ is the input to the block chain, $y(t)$ is the output, $w(t)$ is the intermediate signal, and $x_f(t)$ and $x_a(t)$ represent the internal state variables for the filter and amplifier respectively. Such state-space models can be constructed at device-level using modified nodal-analysis, similar to the manner in which SPICE would create a model for simulation.

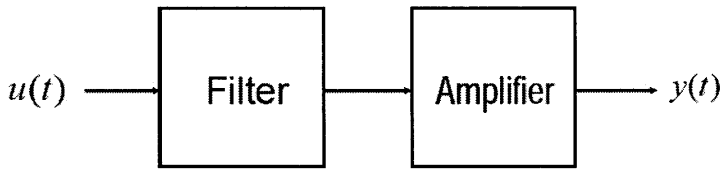


Figure 1-3: Two-block analog circuit representing part of an RF receiver chain.

In a realistic situation, however, both blocks will depend on some set of design parameters p, representing perhaps both geometric and device parameters, such as W, L, R, C. Additionally, both blocks may contain parasitic elements that couple the two blocks to one another and to the substrate. A more accurate depiction of the simple two-block analog circuit is shown in Figure 1-4, and would be modeled using the following pair of coupled and parameterized equations

$$\begin{aligned} C_f(p)\dot{x}_f + C_a(p)\dot{x}_a + G_f(p)x_f + G_a(p)x_a &= b_f u & w &= c_f(x_f, u) \\ \dot{q}_a(x_a, x_f, p) + i_a(x_a, x_f, p) &= b_a(w) & y &= c_a(x_a, w). \end{aligned} \tag{1.1}$$

System (1.1) may be extremely large, and possibly prohibitively expensive to simulate numerically for many different parameter values.

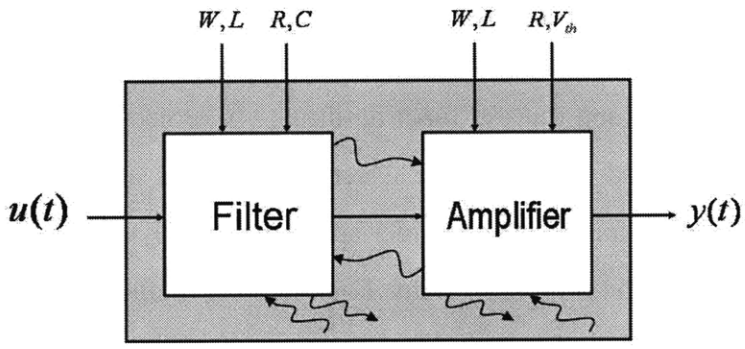


Figure 1-4: Detailed depiction of two-block analog circuit with inter-block coupling, coupling to the substrate, and parameter dependence.

1.2 Design Optimization using Reduced Models

1.2.1 Model Reduction

Due to both the enormous number of equations contained in the model and the complexity of the nonlinear equations, it is often necessary to facilitate model simulation through the use of reduced-complexity models. The goal of model reduction is to capture the "important" dynamics of a system in a compact model that has reduced complexity, i.e. is computationally cheap to simulate. Here "important" is defined on a case-by-case basis, based on the desired region of performance for the model. This includes, for example, a certain range of input frequencies. From a dynamical system point of view, model reduction aims to replace the large complex nonlinear system (1.1) with a smaller simplified system of equations that preserves both the input-output behavior and the parameter dependence of (1.1)

$$\dot{\hat{q}}(\hat{x},p) + \hat{i}(\hat{x},p) = \hat{b}u, \qquad\qquad y = \hat{c}(\hat{x},p).$$

Here we assume the number of state-variables in the reduced state vector $\hat{x}$ is significantly reduced, and the computation of the nonlinear functions $\hat{q}(\hat{x})$ and $\hat{i}(\hat{x})$ are significantly reduced. The resulting model should also capture the effects of parasitic coupling present in the original system, and should preserve important system properties of the original

system such as passivity or stability.

For linear systems, two common practices for constructing reduced-order models are state-space projection and transfer function fitting. In projection approaches, a low-order state-space is identified and the linear system is projected into that space, resulting in a lower-order linear system. The low-order space is typically chosen to preserve or match certain properties of the original system. For example, moment-matching and frequency-domain proper orthogonal decomposition (POD) preserve transfer function moments in the reduced model [32, 102, 70]. Projection methods are convenient for state-space models, which can be derived from device-level equations.

For nonlinear systems, it is possible to obtain model-complexity reduction through both function approximation and state-space projection. That is, in addition to reducing the number of dynamical system equations and state-space variables, one may obtain a complexity reduction by approximating the original nonlinear function with an alternative class of functions which are computationally cheaper to evaluate. This is particularly true for today's transistor models, such as the BSIM4 model, whose constitutive relations depend on hundreds of parameters and thus are extremely computationally expensive to evaluate. For weakly nonlinear systems, a Volterra-type expansion is common, as the resulting polynomial system can be efficiently reduced with a linear projection [67]. For highly nonlinear systems it is not possible to accurately capture all of the important nonlinear effects through a single local model. Techniques such as the Trajectory Piecewise-Linear (TPWL) technique [79, 25, 96] create Trajectory-Based Models (TBMs) which are constructed to be accurate in a much larger region of the state-space. This region of space is identified by training the system with typical inputs of interest. While there has been a large amount of interest in TBMs recently, the resulting models have yet to gain widespread acceptance due to their high cost of construction and a lack of theoretical statements concerning the accuracy of the resulting reduced models.

In all reduction techniques it is crucial that the reduced-complexity models preserve important system properties such as stability and passivity. It is common practice to sacrifice accuracy in order to guarantee stability in linear projection methods [42, 62]. While there has been significant work done in the field of stable model reduction for linear systems,

little work exists on stability-preserving techniques for nonlinear systems.

To be useful for design and optimization the resulting reduced models must be parameterized, and in addition to preserving state-space accuracy should also preserve parameter-space accuracy. That is, the reduced model should accurately predict the behavior of the system to changing parameter values. There has been a lot of work done on parameterized model reduction for linear systems: Some methods are based on statistical performance evaluation [76, 49, 41, 48], while others are based on moment matching techniques [101, 75, 22, 24, 46], Truncated Balance Realization (TBR) techniques [68], or on quasi-convex optimization techniques [87]. Very few techniques, such as [48], also apply to non-linear systems.

Finally, in addition to reducing the complexity of simulation, the reduced-complexity models must also be sufficiently cheap to construct. Linear reduction techniques such as moment-matching and POD have gained prominence over the more theoretically sound method of balanced truncation due to the extremely high computational complexity of balanced truncation. The TPWL nonlinear reduction technique is also extremely expensive as it requires many simulations of the full nonlinear system. This is generally viewed as one of its largest drawbacks.

1.2.2 Shortcomings of Existing Model Reduction Techniques

Despite the many recent innovations in both linear and nonlinear model reduction, both still have shortcomings making them impractical for modeling systems containing both nonlinear elements and parasitic elements. These shortcomings include both theoretical limitations, such as a lack of stability guarantee for the resulting models, and practical limitations, such as computationally expensive model construction.

For nonlinear model blocks, existing techniques, such as TPWL method, are not guaranteed to preserve system properties such as stability. This is particularly important because in the TPWL procedure both function approximation and projection may create unstable models. Additionally, the TPWL method must be modified to handle systems described by complicated nonlinear descriptor functions. Such systems occur when modeling MOSFETs

with current models such as the BSIM4 model. Nonlinear descriptor functions introduce questions of both stability and existence of a unique solution.

For linear model blocks, there do not currently exist computationally affordable reduction techniques for indefinite and unstable systems that are capable of enforcing stability and passivity. This is an important problem because field-solvers and parasitic extractors used in VLSI design often extract models that are stable but are described by indefinite matrices, or are numerically unstable, despite modeling stable and passive systems.

When considering frequency-dependent effects in distributed systems, it is difficult to efficiently capture coupling between blocks and coupling with the substrate. Reduction techniques for such systems also encounter problems resulting from unstable extracted linear models.

Finally, to be useful for design and optimization all reduction methods must preserve parameter dependence and must be computationally inexpensive to construct. This requires identification of the important and sensitive regions of both the state space and parameter space. Creating nonlinear reduced models using the TPWL method is particularly computationally expensive.

1.2.3 Alternative Applications for Modeling Techniques

Although this work focuses primarily on the modeling of analog systems, the contributions in this thesis could be beneficial to many electrical engineering applications outside of analog systems, such as modeling semiconductor devices, electromagnetic scattering, and control applications. Outside of electrical engineering applications, any multiphysics system, i.e. systems whose behavior is described by coupled partial differential equations (PDEs), could benefit from model reduction techniques to handle the resulting prohibitively complex systems of equations. Many applications in systems biology, such as molecular dynamics and mass action kinetics, could also benefit from new modeling and simulation techniques. Some of the techniques in this thesis have been used to model segments of the cardiovascular circulatory system that are initially described by nonlinear PDEs. These models can be used to determine blood pressure within various segments of the arterial net-

work, which may be useful, for instance, for determining drug-delivery effectiveness for medications aimed at fighting hypertension.

1.3 Overview and Contributions of this Thesis

The contributions of this thesis are in the model reduction of linear and nonlinear systems, with emphasis on preserving system properties such as stability and parameter dependence. Specifically, there are four main contributions in this thesis

- Projection approach for generating stable reduced models from indefinite and unstable linear systems;
- Projection approach for generating parameterized reduced models from highly nonlinear systems;
- Projection approach for generating stable reduced models from highly nonlinear systems;
- System identification approach for generating parameterized and stable reduced models of linear and nonlinear systems from time-domain data.

The remainder of this thesis is organized as follows: Chapter 2 presents a thorough overview of the many notions of stability in regards to dynamical systems. Many of the contributions of this thesis concern preserving stability through model reduction, so the issue of stability is covered in great detail. Chapter 3 introduces model reduction, and presents detailed descriptions of many techniques for both linear and nonlinear systems that will be useful for understanding the contributions later in this thesis. Chapter 4 presents model reduction techniques for indefinite and unstable linear systems that are aimed at enforcing stability and passivity in the reduced models. The majority of these approaches are projection-based, but a non-projection approach is also formulated. Chapter 5 presents a parameterized model reduction approach for nonlinear systems. This approach combines a parameterized moment matching approach for linear systems with the trajectory piecewise linear approach for highly nonlinear systems. Chapter 6 considers the topic of stable model

reduction for highly nonlinear systems. Results in this chapter include both theorems guaranteeing stability in reduced models, and practical algorithms for efficiently computing stable nonlinear reduced models. Chapter 7 takes an alternative approach to model reduction and presents a system identification technique for compact modeling. This technique is capable of preserving both parameter dependence and stability in the reduced models. Finally, chapter 8 concludes the thesis.

Chapter 2

Background: Dynamical Systems and Stability

This chapter introduces dynamical systems, as described by differential equations, and presents an overview of the many notions of stability of dynamical systems, as well as tools for their analysis.

2.1 State-Space models

2.1.1 Overview

Dynamical systems are useful tool for the time-domain analysis of physical systems, as they provide a relationship between input signals $u(t)$ to an output signal $y(t)$ for the system. In state-space models, the evolution of the system is described by a state vector $x(t)$, which is controlled by input $u(t)$ and from which output $y(t)$ is determined. In the most general form, a continuous time state-space model can be expressed as

$$F(\dot{x}, x, u) = 0, \qquad G(x, y) = 0. \tag{2.1}$$

Here $F : \mathbb{R}^{2N+N_u} \mapsto \mathbb{R}^n$ is a dynamical update equation, where $x \in \mathbb{R}^N$ is the state vector of internal variables and $u \in \mathbb{R}^{N_u}$ is a vector of inputs, and $G : \mathbb{R}^{N+N_y} \mapsto \mathbb{R}^{N_y}$ is a static

$\dot{x} = Ax + Bu$	Linear system
$E\dot{x} = Ax + Bu$	Linear descriptor system
$\dot{x} = f(x, u)$	Nonlinear system
$\dot{q}(x) = f(x, u)$	Nonlinear descriptor system
$F(\dot{x}, x, u) = 0$	Implicit nonlinear system

Table 2.1: Terminology for dynamical system forms commonly used in this work.

output map that maps the state $x(t)$ to an output vector $y \in \mathbb{R}^{N_y}$.

Fully implicit system such as (2.1) rarely occur in practice, so for now we will consider the simpler system

$$\dot{x} = f(x, u), \qquad y = g(x). \tag{2.2}$$

Table 2.1.1 shows several other common dynamical system structures that we will consider throughout this work. In order to guarantee that solutions (or as a stronger condition, unique solutions) exist to system (2.2), it is necessary to restrict the class of functions f. It is well known that a solution to (2.2) exists if f is continuous, and furthermore a unique solution exists if f is Lipschitz continuous. Such notions will be useful later on in this thesis, so we will define them here.

Definition 2.1.1. A function $f(x, u)$ is **locally Lipschitz continuous** at $(0, 0)$ if there exist finite positive constants k_f, r such that

$$||f(x, u) - f(z, v)|| \leq k_f \left[||x - z|| + ||u - v||\right] \tag{2.3}$$

$\forall (x, u), (z, v) \in \mathbb{B}_r$ and $\forall t \geq 0$. If $\mathbb{B}_r = \mathbb{R}^N$, then the function is **Lipschitz continuous**.

The Lipschitz continuity can be interpreted as requiring derivatives of the function to be bounded.

Observation 2.1.1. A function $f(x, u)$ is locally Lipschitz continuous with Lipschitz constant κ in the ball $\mathbb{B}_r$ if

$$\left|\frac{\partial f_i(x, u)}{\partial x_j}\right| \leq \kappa, \qquad \left|\frac{\partial f_i(x, u)}{\partial u_j}\right| \leq \kappa, \qquad \forall (x, u) \in \mathbb{B}_r$$

for all i, j.

2.1.2 Models arising in applications

Most of the example dynamical systems considered in the thesis are the result modeling physical systems using conservation laws and constitutive relations. Specifically, the majority of our circuit examples are constructed using nodal analysis. For example, consider linear system

$$E\dot{x} = Ax + Bu \qquad\qquad y = C^T x. \tag{2.4}$$

In nodal analysis, each equation in system (2.4) results from Kirchoff's current law, which states that the sum of currents entering and leaving any node must equal zero. Additionally, the E matrix contains capacitance and conductance values, the A matrix contains conductances, and the state variables x correspond to node voltages and possibly inductor currents. Although it is typically possible to transform systems from descriptor form (2.4) into standard form for easier analysis, such a transformation is extremely computationally expensive and thus we wish to develop modeling techniques that are capable of handling systems in descriptor form.

A nonlinear system modeled using MNA will produce a system of the form

$$\dot{q}(x) = f(x, u) \qquad\qquad y = g(x). \tag{2.5}$$

Here $f(x, u)$ contains branch currents through resistive devices, and $q(x)$ represents the charges accumulated in charge-storing devices, such as capacitors and transistors, such that $\dot{q}(x)$ is a current. Often times in nonlinear circuits the system inputs are applied linearly to the system, meaning the state dependence can be separated from the input dependence in the nonlinear function, i.e. $f(x, u) = f(x) + bu$.

It is possible to construct physically-based state-space models in many other areas of engineering using conservation laws and constitutive relations. In some cases, the ODEs we consider will be the result of discretizing a PDE spatially.

2.2 Dissipative Systems

In this section we introduce the notion of dissipativity, which is a generalization of the standard idea of stability. We additionally present techniques with which we can analyze the stability of dynamical systems.

2.2.1 Storage Functions and Supply Rates

In the real world, all physical systems are dissipative in some sense. Dissipativity is generalization of more common system properties such as stability and passivity, and is related to a system's ability to store and consume energy. For example, an electrical system may store energy in electromagnetic fields using inductors and capacitors, and may dissipate energy (in the form of heat) by currents flowing through resistors. Rather than defining dissipation based on the rate at which a system consumes energy, it is more convenient to instead consider the rate at which the system *stores* energy. That is, a nonlinear system

$$\dot{x} = f(x, u) \qquad\qquad y = g(x). \tag{2.6}$$

is dissipative if it stores energy at a slower rate than energy is supplied to it. The difference between the supplied energy and the stored energy is the amount of energy dissipated (i.e. consumed) by the system. As shown below, dissipation can be proven, or verified, through the use of storage functions and supply rates [103].

Definition 2.2.1. *System* (2.6) *is said to be* ***dissipative*** *with respect to supply rate σ if, for all initial times t_0 and initial conditions $x_0 = x(t_0)$, there exists a non-negative storage function $L(x) : \mathbb{R}^N \mapsto \mathbb{R}_+$ such that the dissipation constraint*

$$L(x) \leq L(x_0) + \int_{t_0}^{t} \sigma(x, u) d\tau \tag{2.7}$$

is satisfied for all $x(t), u(t)$ satisfying (2.6)

A storage function is a function whose increments are bounded (i.e. its rate of increase or decrease is bounded), and the storage function $L(x)$ can intuitively be though of as a

measure of the *internal energy* of system (2.6) when it is in state $x(t)$. If the system is initially in the state x_0, then $L(x_0)$ represents the initial energy of the system. Similarly, we can interpret $\sigma(x, u)$ as the instantaneous net power (across all input-output ports) supplied to the system at time t, and the integral of this supplied power over time represents the total net energy supplied to the system. Using these interpretations, dissipation constraint (2.7) implies that the energy in the system at time t cannot be greater than the initial energy in the system plus the energy supplied to the system.

Different notions of dissipativity correspond to different supply rates, and also possibly different constraints on the positivity of $L(x)$. The two most common notions, passivity and finite-gain stability, will be addressed in detail in section 2.2.3

Practically, it is often easier to consider the differential version of constraint (2.7)

$$\frac{\partial L(x)}{\partial t} \leq \sigma(x, u). \tag{2.8}$$

Again interpreting $L(x)$ as internal energy and σ as supplied power, we can interpret (2.8) as a constraint that the energy in the system is *decreasing* faster than it is being supplied to the system. In this notation, we use $\partial L(x)/\partial t$ (or sometimes $\dot{L}(x)$) to represent the derivative of $L(x)$ with respect to time *along solutions* $x(t)$ *to* (2.6). That is, $\dot{L}(x) = \nabla L(x)^T \dot{x}$.

In order to prove dissipation with respect to a given supply rate it is necessary to show that there exists a non-negative storage function satisfying (2.8) or (2.7). The benefit of storage functions is that they allow us to analyze the behavior of solutions to $x(t)$ to (2.6) without having to actually solve (2.6) for $x(t)$. This is possible because dissipation constraint (2.8) only requires information about $\dot{x}$, which is explicitly available from (2.6). For example, suppose we wish to show that in the absence of inputs, i.e. $u(t) = 0$, solutions always converge to zero, i.e. $||x(t)|| \to 0$. To prove this, it is sufficient to show that $\frac{d}{dt}(x^T x) < 0$ for all t, implying that $||x||$ is always decreasing. This can be achieved by considering the quadratic storage function $L(x) = x^T x$ and the supply rate $\sigma = 0$. With

these definitions, constraint (2.8) can be expressed as

$$\frac{\partial L(x)}{\partial t} = x^T \dot{x} + \dot{x}^T x = x^T f(x) + f(x)^T x \leq 0,$$

If this condition is satisfied for all possible x, then then solutions converge to zero and the system is dissipative with respect to supply rate $\sigma = 0$, and we do not need to know any specific information about the solutions to (2.6).

2.2.2 Internal Stability

Internal stability refers to the behavior of solutions to the autonomous system, i.e. the system with zero input

$$\dot{x} = f(x, 0), \tag{2.9}$$

in response to various initial conditions, and is a property of an equilibrium point of the system.

Definition 2.2.2. The state $x = x_{eq}$ is said to be an **equilibrium point** of the autonomous system (2.9) if

$$f(x_{eq}, 0) = 0.$$

If $x = x_{eq}$ is the only x such that $f(x) = 0$, then it is a **global equilibrium point**.

In many cases throughout this thesis we will assume the equilibrium point is at the origin: $x_{eq} = 0$. This can be assumed without loss of generality because the origin can always be translated to a non-zero equilibrium point using a change of coordinates. In physical systems it is common to have the origin ($x = 0$) be an equilibrium, but for many analog circuits this is not the case. This is due to constant input sources that are treated not as variable inputs $u(t)$, but as a fixed offset to the system built within the function f, resulting in a non-zero DC operating point.

As mentioned previously, internal stability concerns the behavior of the system in response to various initial conditions. One "weak" notion of stability requires that solutions

cannot stray arbitrarily far from an equilibrium point, and in such case the equilibrium point is said to be stable in the sense of Lyapunov.

Definition 2.2.3. The equilibrium x_{eq} is said to be **stable in the sense of Lyapunov** (or stable i.s.L.) if for any $\delta > 0$ there exists $\epsilon > 0$ such that $||x(t) - x_{eq}|| < \epsilon$ for all initial conditions x_0 satisfying $||x_0 - x_{eq}|| < \delta$.

In this sense, given an initial condition x_0 within a ball of radius δ around an equilibrium point, i.e. $||x_0 - x_{eq}|| < \delta$, the resulting solution $x(t)$ satisfying autonomous system (2.9) will not stray outside a ball of radius ϵ around the equilibrium point, i.e. $||x(t) - x_{eq}|| < \epsilon$, for all time. It is important to note that stability in the sense of Lyapunov does not require that solutions converge to the equilibrium point.

If solutions starting 'close' to the equilibrium always eventually converge to the equilibrium, then the equilibrium point is said to be attractive.

Definition 2.2.4. The equilibrium x_{eq} is said to be **attractive** if there exists $\delta > 0$ such that

$$\lim_{t \to \infty} ||x(t_0 + t)|| = x_{eq}, \forall t, t_0 \geq 0, \forall x_0 \in \mathbb{B}_\delta \tag{2.10}$$

Here, $\mathbb{B}_\delta$ is a ball of radius δ centered at x_{eq}. If $\mathbb{B}_\delta = \mathbb{R}^N$, then the equilibrium is said to be **globally attractive**.

It is possible for an equilibrium point to be attractive but not stable in the sense of Lyapunov. This would occur if the solution first strays arbitrarily far from the equilibrium, but then returns and converges to the equilibrium. To prevent such cases, it is convenient to also require stability in the sense of Lyapunov. If an equilibrium point is both stable in the sense of Lyapunov and attractive, then it is said to be asymptotically stable.

Definition 2.2.5. The equilibrium x_{eq} is said to be **asymptotically stable** if it is both attractive and stable in the sense of Lyapunov for all $x_0 \in \mathbb{B}_\delta$. If $\mathbb{B}_\delta = \mathbb{R}^N$, then the equilibrium is said to be **globally asymptotically stable**.

If, in addition to convergence to an equilibrium point, the solutions converge *exponentially fast*, then the equilibrium point is said to be exponentially stable

Definition 2.2.6. The equilibrium x_{eq} is said to be **exponentially stable** if there exist constants $\delta, a, b > 0$ such that

$$||x(t_0 + t) - x_{eq}|| \leq a||x_0||e^{-bt}, \forall t, t_0 \geq 0, \forall x_0 \in \mathbb{B}_\delta \tag{2.11}$$

If $\mathbb{B}_\delta = \mathbb{R}^N$, then the equilibrium is said to be **globally exponentially stable**.

For time-varying systems all of the previous definitions can be generalized to possess dependence on an initial time t_0. Stability properties that hold independent of this initial time are referred to as *uniform* stability. For details on stability of time-varying system and uniform stability, see [81].

Proving Internal Stability

The preceding notions of internal stability can be proven through the use of storage functions, as described previously in Section 2.2.1. For internal stability, a storage function is often referred to as a **Lyapunov function** or **energy function**. In such cases the supply rate should be independent of the input u, resulting in the dissipation constraint

$$\frac{dL(x)}{dt} \leq \sigma(x).$$

Different notions of stability are proven using various supply rates $\sigma(x)$ and various constraints on the positivity of $L(x)$. The following internal stability analysis via Lyapunov functions is referred to as *Lyapunov's direct method.*

Theorem 2.2.1. *Internal stability of system* (2.9)*, i.e. the stability of equilibrium point* x_{eq}*, is certified by a smooth function* $L(x)$ *satisfying the following*

$$L(x) \succ 0, \quad \frac{dL(x)}{dt} \leq 0 \quad \Rightarrow \quad \textit{stable i.s.L.} \tag{2.12}$$

$$L(x) \succ 0, \quad \frac{dL(x)}{dt} \prec 0 \quad \Rightarrow \quad \textit{asymptotically stable} \tag{2.13}$$

$$\lambda_1 x^T x \geq L(x) \geq \lambda_2 x^T x, \quad \frac{dL(x)}{dt} \leq -\lambda_3 x^T x \quad \Rightarrow \quad \textit{exponentially stable} \tag{2.14}$$

$\forall t \geq 0$, $\forall x \in \mathbb{B}_\delta$ *satisfying* (2.9), *and* $\lambda_1, \lambda_2, \lambda_3 > 0$ *If* $x_{eq} \neq 0$, *then the definiteness constraints on* L *are relative to* x_{eq}. *If* $\mathbb{B}_\delta = \mathbb{R}^N$, *then the stability is global.*

The constraints on the storage function and dissipation rate in Theorem 2.2.1 make sense based on our intuitive understanding of the storage function as a measure of the system's energy. For Lyapunov stability, solution do not necessarily converge to the equilibrium, meaning that the rate of decrease in energy can be zero, resulting in the non-strict inequality for $\dot{L}(x)$. On the other hand, asymptotic stability requires the solution always converge to the equilibrium, meaning the energy must always decrease to zero, thus requiring a strict inequality for $\dot{L}(x)$. Lastly, exponential stability requires solutions to converge exponentially fast, meaning that the energy must decrease exponentially, thus requiring a quadratic lower bound on the rate of decrease of the energy, $\dot{L}(x) \leq -\lambda_3 x^T x$. For proofs of the preceding theorem, and additional stability results, see, for example, references [100, 81].

2.2.3 External Stability

External stability is a property of the input-output system

$$\dot{x} = f(x, u), \qquad y = g(x) \tag{2.15}$$

and refers to the system's ability to amplify input signals $u(t)$ into output signals $y(t)$. In these cases we are concerned only with the behavior of the output $y(t)$, and not necessarily all of the internal states $x(t)$. In the following sections we will focus mainly on two types of external stability: finite-gain stability and passivity.

Finite-Gain Stability

Finite-gain stability, also possibly referred to as bounded-input bounded-output (BIBO) stability or input-output stability, requires that finite inputs produce finite outputs. Qualitatively, the system is said to be finite-gain stable if the system's output $y(t)$ can be bounded in some measure by a linear function of the system's input $u(t)$ in that same measure.

Definition 2.2.7 ([100]). System (2.15) is said to be **small-signal finite-gain L_p stable** if there exist constants $r_p > 0$ and $\gamma_p < \infty$ such that

$$||y||_p \leq \gamma_p ||u||_p$$

for all $t > t_0$, given initial state $x(0) = 0$ and input $u(t)$ such that $||u||_\infty < r_p$. If $r_p = \infty$, then the system is **finite-gain L_p stable**.

In terms of storage functions and supply rates, a system is finite-gain stable if it is dissipative with respect to the supply rate $\sigma = \gamma^2 u^T u - y^T y$, resulting in the dissipation constraint

$$\frac{\partial L(x)}{\partial t} \leq \gamma^2 u^T u - y^T y.$$

This constraint makes sense based on our intuitive understanding of the dissipation constrait. If the system produces unbounded outputs $y(t)$ from finite inputs $u(t)$, then for any finite γ the right side of the constraint becomes $-\infty$, and thus no storage function $L(x)$ can satisfy the constraint that the system dissipate energy infinitely fast. Here γ^2 is often referred to as the L_2 gain of a system. If $\gamma^2 \leq 1$, then the system is said to be contractive.

It is often possible to determine a system's external behavior based on its internal behavior.

Theorem 2.2.2 ([100]). *Suppose $x = 0$ is an exponentially stable equilibrium of system* (2.15). *If $f(x, u)$ is continuously differentiable and $f(x, u)$ is locally Lipschitz continuous at $(0, 0)$, then system* (2.15) *is small-signal finite-gain L_p stable. If $\mathbb{B}_r = \mathbb{R}^N$, then the system is finite-gain L_p stable.*

It is interesting to note that in general the converse is not true. That is, a system can be input-output stable without being internally stable. For example, the linear system $\dot{x} = Ax + Bu$ with order $N = 2$ described by the matrices $A = [-1, 0; 1, 0]$, $B = [1; 0]$, $C = [1; 0]$. The equilibrium $x = 0$ is clearly unstable because the matrix A has a positive eigenvalue ($\lambda = 1$), but the output $y(t)$ is bounded with finite gain because the equations are uncoupled and the output does not see the unstable mode.

Passivity

A more subtle, but equally important, property for input-output systems is passivity. A passive system is one that does not generate energy. Passivity is a particularly important system property when models are being interconnected with one another in feedback configurations for simulation. The interconnection of individual passive blocks is also guaranteed to be passive, but it is not in general true that interconnections of stable systems are always stable. Additionally, it is possible for a system to be stable but not passive, meaning outputs will be bounded, but they may be inaccurate due to artificial energy generated within the system and observable in the outputs.

There are many definition of passivity (see [104] for a detailed analysis of the various notions of passivity), but the most generic is to say that a system is **passive** if it is dissipative with respect to the supply rate $\sigma = u^T y$, resulting in the dissipation constraint

$$\frac{\partial L(x)}{\partial t} \leq u^T y$$

For a circuit modeled using nodal analysis, the input and output typically represent current and voltage, meaning that $u^T y$ has the meaning of power supplied to the system. Note that this definition of passivity requires a system to have an equal number of inputs and outputs. In some cases later in this thesis to simplify equations we will instead use the supply rate $\sigma = 2u^T y$ to verify passivity.

In the real world, all physical systems should behave in a passive manner. It is common to refer to some models of physical systems as non-passive, but typically this is only because there are inputs or sources that are treated as internal parts of the system rather than sources. For example, amplifiers are often referred to as active devices (i.e. non-passive), but this assumes that there is some power supply bias to the system that is completely ignored as an input, thus allowing the system to seemingly amplify signals and generate energy. If one accounts for all such sources as inputs to the system, then no additional energy is generated and the system is passive.

2.2.4 Incremental Stability

One stronger notion of stability that can be applied to both internal stability and external stability, is "incremental stability". Incremental stability examines the behavior of *differences between solutions* of a dynamical system. If we define $\Delta = \bar{x} - \tilde{x}$, where $\bar{x}$ and $\tilde{x}$ are solutions to $\dot{x} = f(x, u)$, incremental stability examines the behavior of the system

$$\dot{\Delta} = f(\bar{x}, u) - f(\tilde{x}, u). \tag{2.16}$$

That is, we wish to determine whether this new system is stable with respect to Δ [3].

Definition 2.2.8. *System* (7.1) *is* ***incrementally stable*** *if it is well-posed and, given any two sets of initial conditions $\bar{x}_0$ and $\hat{x}_0$, the resulting two solutions to* (2.15) *in response to the same input $u(t)$ satisfy*

$$\int_{t=t_0}^{\infty} \|\bar{x}(t) - \tilde{x}(t)\|^2 < \infty \tag{2.17}$$

for all initial conditions and inputs.

Incremental stability is a strong notion because it implies all other traditional notions of stability by simply selecting $\tilde{x} = x_{eq}$. For example, internal asymptotic stability (which requires that all solutions to the autonomous system converge to the equilibrium point) is implied by incremental stability by selecting $\tilde{x}(t < t_0) = 0$ and $u = 0$. Under these conditions, (2.46) implies that $\bar{x}(t) \rightarrow 0$ as $t \rightarrow \infty$.

One useful consequence of incremental stability is that it guarantees that perturbations to solutions decay, which is an extremely important property when considering numerical simulation of a system. For example, if $\bar{x}(t)$ and $\tilde{x}(t)$ are two solutions to system (2.15) resulting from input $u(t)$ but two different initial conditions $\bar{x}_0$ and $\tilde{x}_0$, we would like the difference between the solutions to decay $||\bar{x}(t) - \tilde{x}(t)|| \rightarrow 0$.

Like all previous notions of stability that have been introduced, incremental stability can also be proven through the use of storage functions. But now the storage functions must be functions of differences of variables, such as $L(\bar{x}, \tilde{x}) = (\bar{x} - \tilde{x})^T P(\bar{x} - \tilde{x})$. The

resulting dissipation constraint for incremental stability of system (2.16) is

$$(\bar{x} - \tilde{x})^T P(f(\bar{x}, u) - f(\tilde{x}, u)) \leq \sigma(\bar{x} - \tilde{x}).$$

Incremental stability can also be interpreted as the result of contraction behavior of the state-space. This is referred to as convergent dynamics [65] or contraction analysis [52]. Contraction analysis examines the stability of the differential system

$$\dot{\Delta} = A\Delta + K \qquad\qquad \xi = C\Delta + G \tag{2.18}$$

where

$$A = \frac{\partial f}{\partial x}\Big|_{x,u}, \qquad K = f(x, u), \qquad C = \frac{\partial g}{\partial x}\Big|_{x}, G \qquad = g(x),$$

$\Delta \in \mathbb{R}^N$ and $\xi \in \mathbb{R}^{N_y}$. The system is said to be contracting if the increments Δ and ξ converge to zero exponentially. According to Theorem 3 in [52], if System (2.18) is well-posed and stable in the differential variable Δ, i.e. Δ converges exponentially to zero, for all x, u, y satisfying $\dot{x} = f(x, u)$ and $y = g(x)$, then system (2.15) is incrementally stable. Essentially, this means that if linearizations of the nonlinear system, around all possible solutions to the nonlinear system, are stable, then the nonlinear system is stable. The converse, however, is not true in general. That is, linearizations of a stable nonlinear system can be unstable.

It is often easier to prove stability by examining the differential system (2.18) instead of the original system (2.15).

2.2.5 Construction of Storage Functions

For physically-inspired models, such as those constructed using nodal analysis, it is often possible to explicitly construct storage functions based on the physics used to construct the model. As mentioned in Section 2.2, the storage function can be interpreted as a measure of the system's internal energy. Thus, for models of physical systems, it is often possible to explicitly construct a measure of the total system energy as a quadratic function of state

variables.

For example, in an RLC circuit modeled using modified nodal analysis, the dynamic state variables are the currents through the inductors and the voltages on the capacitors, and the total energy in the system, $\mathcal{E}$, is represented by the energy stored in the inductors and the capacitors

$$\mathcal{E} = \frac{1}{2}\sum_i C_i v_i^2 + \frac{1}{2}\sum_j L i_j^2. \tag{2.19}$$

Here, C_i and v_i represent the capacitances and corresponding voltages, and L_j and i_j represent the inductances and corresponding currents.

Similarly, for a mechanical system whose state-space variables are the positions and velocities of the system nodes, the total energy in the system can be represented by a sum of the kinetic and potential energy at each node, represented as

$$\mathcal{E} = \frac{1}{2}\sum_i m_i v_i^2 + \frac{1}{2}\sum_j k_j x_j^2 \tag{2.20}$$

where m_i are the masses, v_i are particle velocities, k_j are the spring constants, and x_j are particle displacements.

It is possible in a similar manner to construct quadratic storage functions for many other types of physically-based systems [103].

2.3 Stability of Linear systems

For linear systems, stability is easy to analyze and the theory is well known. Additionally, although all previous stability analysis has been in the time, for linear systems many notions of stability can be analyzed in the frequency domain.

2.3.1 Quadratic supply rates and storage functions

When considering quadratic supply rates (which includes the cases of Lyapunov stability, passivity, and BIBO stability) it is sufficient to consider quadratic supply rates [103]. For a

linear descriptor system

$$E\dot{x} = Ax + Bu(t) \tag{2.21}$$

along with quadratic storage function $L(x) = x^T E^T P E x$ and quadratic supply rate $\sigma(u, y) = x^T Q x + x^T R u + u^T S u$, the dissipation constraint becomes

$$x^T E^T (PAx + Bu) + (u^T B^T + x^T A^T P) Ex \leq x^T Q x + x^T R u + u^T S u. \tag{2.22}$$

For a given linear model described by $\{E, A, B, C\}$, in order to prove dissipativity with respect to supply rate described by $\{Q, R, S\}$, it is necessary to solve (2.22) for a symmetric positive semidefinite (SPSD) matrix P.

2.3.2 Internal Stability and Lyapunov Matrix Equations

Internal stability can be proven using storage functions and the supply rate $\sigma = x^T Q x$, resulting in in the dissipation constraint

$$x^T (E^T P A + A^T P E) x \leq x^T Q x, \tag{2.23}$$

where $-Q$ is a symmetric positive definite (SPD) matrix. Note that we have rewritten the term $2x^T E^T P A x$ as $x^T (E^T P A + A^T P E) x$, which is allowed because this term is a scalar. Additionally, since constraint (2.23) must be satisfied for all possible x, it is sufficient to search for a SPD matrix P satisfying the following matrix equation

$$E^T P A + A^T P E - Q \preceq 0, \tag{2.24}$$

which is the well known Lyapunov matrix equation.

This condition is satisfied if the generalized eigenvalues of matrix pair (E, A) all have strictly negative real part. This is equivalent to saying $\tilde{A} = E^{-1} A$ has eigenvalues with strictly negative real part, which is the traditional stability notion for linear systems. If we allow zero eigenvalues of (E, A), this corresponds to a matrix $-Q$ that is SPSD.

Theorem 2.3.1 ([93]). *If System* (2.21) *is stable, i.e. the matrix pair* (E, A) *has all eigen-*

values with negative real part, then for any SPD matrix Q there exists a unique SPD matrix P which solves (2.24). *Conversely, if there exist SPD matrices Q, P satisfying* (2.24), *then the matrix pair (E, A) has all eigenvalues with negative real part and the system is stable. If the matrix E is singular, then there may not exist an SPD solution P for some SPD Q, and if there are solutions they may not be unique.*

As another means of understanding internal stability of linear systems, recall that the solution to an autonomous system of linear ODEs $\dot{x} = Ax$ is a matrix exponential $x(t) = \exp(At)x_0$, where the matrix exponential satisfies

$$\exp(A) = T \exp(\Lambda) T^{-1}$$

where T contains the eigenvectors of A and Λ is a diagonal matrix containing the associated eigenvalues. Thus, if the eigenvalues have negative real part, then solutions to the system decay exponentially with time. One consequence of this is that for a linear system, asymptotic stability and exponential stability are equivalent. This result is obvious from Theorem 2.2.1 because we are considering both a quadratic Lyapunov function $L(x)$ and supply rate $\sigma(x)$, and therefore the conditions for exponential stability are automatically satisfied.

2.3.3 Passivity and the Positive-Real Lemma

One of the most important properties for linear systems arising in IC applications is passivity. Recall from Section 2.2.3 that a system is passive if it is dissipative with respect to the supply rate $\sigma = u^T y$, which simplifies dissipation constraint (2.22) to

$$x^T \left(E^T P A + A^T P E\right) x + 2x^T E^T P B u - 2u^T C^T x \leq 0. \tag{2.25}$$

The task of certifying passivity for a linear system requires finding an SPD matrix P such that inequality (2.25) is satisfied for all x, u. Since constraint (2.25) is quadratic in the

variable $[x; u]$, it can be expressed as the following quadratic form

$$\begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} E^T PA + A^T PE & E^T PB - C \\ B^T PE - C^T & 0 \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \leq 0, \tag{2.26}$$

and thus it is also sufficient to satisfy the following matrix inequality

$$\begin{bmatrix} E^T PA + A^T PE & E^T PB - C \\ B^T PE - C^T & 0 \end{bmatrix} \preceq 0 \tag{2.27}$$

Matrix inequality (2.27) can be transformed into an algebraic Riccati inequality (ARI) via Schur complement

$$E^T PA + A^T PE + (E^T PB - C)R^{-1}(B^T PE - C^T) \leq 0. \tag{2.28}$$

Alternatively, constraint (2.25) can be transformed into a pair of matrix constraints. Since (2.25) is linear in u, in order to be satisfied globally the linear terms in u must all balance one another, resulting in a pair of linear expressions

$$E^T PA + A^T PE \leq 0 \tag{2.29}$$

$$E^T PB = C. \tag{2.30}$$

Thus, the passivity constraint is simply the internal (Lyapunov) stability constraint combined with a relation between the input and output. Intuitively this makes sense because a passive system is stable with the additional constraint that feedback connections are stable, implying a relation between input vector B and output vector C.

2.3.4 Finite-Gain Stability and the Bounded-Real Lemma

Finite-gain stability implies that if a system is excited with a bounded input $u(t)$, then the output $y(t)$ will also be bounded. For a linear system with the finite-gain supply rate

$\sigma = \gamma^2 u^T u - y^t y$, the dissipation constraint simplifies to

$$x^T \left(E^T P A + A^T P E + C C^T\right) x + 2 x^T E^T P B u - \gamma^2 u^T u \leq 0, \tag{2.31}$$

which can be expressed as the quadratic form

$$\begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} E^T P A + A^T P E + C C^T & E^T P B \\ B^T P E & -\gamma^2 \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \leq 0. \tag{2.32}$$

As before, we can represent this as a matrix inequality which can then be transformed into an ARI via Schur complement

$$E^T P A + A^T P E + C C^T - \frac{1}{\gamma^2} E^T P B B^T P E \leq 0 \tag{2.33}$$

For a linear system, finite-gain stability is implied by internal stability. However this is not true in general for nonlinear systems. This result is obvious from equation (2.33), which can always be made stable by scaling P and γ, provided the term $E^T P A + A^T P E \prec 0$ (which is implied by internal stability).

2.3.5 Definite Systems

In some cases the stability analysis of a linear system can be simplified if the system matrices have some nice structure, such as sign-definiteness. We refer to a linear system as **definite** if the matrices $E = E^T \succ 0$ and $A \preceq 0$, and additionally the input and output vectors are equal, $B = C$. Such systems often arise in practice, such as when modeling RLC networks using nodal analysis. Definite systems are convenient because they are always guaranteed to be stable and passive.

Stability and passivity of a definite system can be proven with the storage function $L(x) = x^T E x$, which is a positive definite function because E is a SPD matrix. The resulting dissipation constraint $\dot{L}(x) \leq 0$ for the autonomous system (input $u = 0$) becomes

$$\dot{L}(x) = x^T E \dot{x} + \dot{x}^T E x = x^T (A + A^T) x \leq 0,$$

which is satisfied because $A \preceq 0$ by definition. Passivity is additionally enforced, as the dissipation constraint $\dot{L}(x) \leq 2u^T y$

$$\dot{L}(x) = x^T(A + A^T)x + 2x^T Bu - 2u^T C^T x \tag{2.34}$$

$$= x^T(A + A^T)x \leq 0 \tag{2.35}$$

is satisfied by the initial assumption that $B = C$. As will be shown later in section 3.4.1, preserving passivity during model reduction for definite systems is a trivial task.

2.4 Stability of Nonlinear systems

In this section we discuss stability analysis for nonlinear systems

$$\dot{x} = f(x, u), \qquad y = g(x). \tag{2.36}$$

For nonlinear systems, the issue of stability is not nearly as simple to analyze as it was for linear systems.

2.4.1 Difficulties

While it is possible to prove stability of nonlinear systems through the use of storage functions and Lyapunov functions, as was done for linear systems in Section 2.3, there are two primary difficulties. First, there is no easy "check" for whether a dissipation constraint is satisfied. For the linear case, the dissipation constraint was always transformed into a matrix inequality, for which there exist many efficient solution techniques. However, such tricks are not available for the nonlinear case. Consider, for example, system (2.36) along with the Lyapunov function $L(x) = x^T Px$, resulting in the dissipation constraint

$$x^T Pf(x, 0) + f(x, 0)^T Px \leq 0. \tag{2.37}$$

For an arbitrary nonlinear function $f(x, u)$, it is very difficult to determine whether (2.37) is satisfied globally for all x. In the linear case P could be found by solving a Lyapunov matrix equation, however no such solution exists for the nonlinear case.

The second difficulty with storage functions for nonlinear systems is that we do not know a priori the complexity of a storage function required to certify stability for a given nonlinear system. In the linear case it was sufficient to consider quadratic storage functions, but this is not true in general for nonlinear systems. Often times for physically-based systems (i.e. equations constructed from conservation laws), quadratic storage functions can be explicitly constructed based on physical laws used to construct the system equations, as discussed in Section 2.2.5, but in general storage functions for nonlinear systems will be highly nonlinear. Despite these limitations, storage functions remain one of the few tools available for analysis of nonlinear systems.

2.4.2 Local Stability Analysis

Although it is difficult to determine global stability for nonlinear systems through dissipation constraints, as described in Section 2.4.1, analyzing the local behavior of a nonlinear system turns out to be a relatively easy task. In order to determine the local behavior of a nonlinear system at an equilibrium point, it is sufficient to consider a local approximation of the nonlinear system, and analyze the stability of the approximate model. Specifically, we will consider linearizations of the nonlinear model about an equilibrium point. This is referred to as Lyapunov's indirect method.

Theorem 2.4.1 (Lyapunov's indirect method). *If the linearized system*

$$\dot{x} = Ax \qquad\qquad A = \frac{\partial f(x,u)}{\partial x}\Big|_{x=x_{eq},u=0}$$

is asymptotically stable, then x_{eq} is a locally asymptotically stable equilibrium of the system (2.36).

Thus, the equilibrium of the nonlinear system is stable if the Jacobian matrix A has eigenvalues with strictly negative real part. If any eigenvalues of A lie on the imaginary

axis, then one must consider higher order terms to determine local stability of the nonlinear system. Additionally, linearizations can be used to prove instability. If A has any eigenvalues with positive real part, then x_{eq} is an unstable equilibrium for system $\dot{x} = f(x)$. For proofs of the preceding theorem, see references [81, 58].

Lyapunov's indirect method provides a convenient approach to stability analysis of equilibrium points for the nonlinear system, but it only provides information about local behavior of the nonlinear system. Unfortunately, there is no simple way to analyze the global behavior of a nonlinear system without utilizing storage function and dissipation constraints.

2.4.3 Nonlinear Descriptor Functions

When modeling analog circuits containing nonlinear capacitances, such as those containing transistors, the resulting dynamical systems will contain a nonlinear descriptor function $q(x)$

$$\frac{d}{dt}\left[q(x)\right] = f(x,u). \tag{2.38}$$

If the function $q(x)$ is invertible, then it is possible to obtain a system of the form (2.36) through a nonlinear change of coordinates. Similarly, it is possible to rewrite the system in non-descriptor form

$$\dot{q}(x) = Q(x)\dot{x} \quad \longrightarrow \quad \dot{x} = \tilde{f}(x,u)$$

where $Q(x)$ is the Jacobian of $q(x)$ and $f(x,u) = Q(x)^{-1}f(x,u)$.

However, the functions $q(x)$ are often not invertible, making it difficult to make statements about global properties of the system. In many cases the function $q(x)$ is only valid locally due to the models used to describe the elements in the circuits being modeled. These cases will be considered in more detail in Chapter 6.

2.5 Discrete-Time Models

Up to this point we have only considered continuous time (CT) systems. However, it is often convenient to consider instead discrete-time (DT) systems. In this section we briefly present several stability results for DT systems that are analogous to the previously presented CT results.

2.5.1 Overview

A generic implicit DT state-space model has the form

$$F(x[t], x[t-1], u[t-1]), = 0 \qquad\qquad G(x[t], y[t]) = 0. \tag{2.39}$$

where $x[t]$ is the internal state $y[t] \in \mathbb{R}^{N_y}$ is the output, $u[t] \in \mathbb{R}^{N_u}$ is the input, $F \in \mathbb{R}^N$ is a dynamical relation between the internal variables and the input, and $G \in \mathbb{R}^{N_y}$ is a static relationship between the internal variables and the output. However, when we consider DT systems later in Chapter 7 we will instead consider DT systems with the slightly different more general (but not technically state-space) form

$$\begin{aligned} &F(v[t], v[t-1], \ldots, v[t-m], u[t], \ldots, u[t-k]) = 0 \\ &G(y[t], v[t]) = 0. \end{aligned} \tag{2.40}$$

Here $v[t] \in \mathbb{R}^N$ is a vector of internal variables, but to be precise we will not refer to it as the state because it does not contain all the information needed to compute the future state, as is required by state-space models. It is possible to rewrite (2.40) in the form of (2.39) by defining a state $x[t]$ containing delays of $v[t]$ and the delays of the input $u[t]$, but this will require transformations to the function F that are not desirable, and thus we shall consider only system (2.40).

Definition 2.5.1. *System* (2.40) *is* ***well-posed*** *if given any arbitrary variables* $v_1, \ldots, v_m \in \mathbb{R}^N$ *and* $u_0, \ldots, u_k \in \mathbb{R}^{N_u}$*, there exist unique solutions* $v_0 \in \mathbb{R}^N$ *and* $y \in \mathbb{R}^{N_y}$ *to* $F(v_0, v_1, \ldots, v_m, u_0, \ldots, u_k) = 0$ *and* $G(y, v_0) = 0$.

For the remainder of this thesis we shall use the following compact notation when describing DT models:

$$V = [v_0, \ldots, v_m], \qquad U = [u_0, \ldots, u_k], \tag{2.41}$$

where $v_0, \ldots, v_m$ and $u_0, \ldots, u_k$ are arbitrary variables, not necessarily inputs and outputs satisfying (2.40),

$$V_+ = [v_0, \ldots, v_{m-1}], \qquad V_- = [v_1, \ldots, v_m], \tag{2.42}$$

where V_+ contains the first m components of V and V_- contains the last m components of V,

$$V[t] = [v[t], \ldots, v[t-m]], \qquad U[t] = [u[t], \ldots, u[t-k]], \tag{2.43}$$

where $v[t]$ is the internal state of the identified model (7.1) in response to past inputs $U[t]$ and initial conditions $v[t-1], \ldots, v[t-m]$, i.e. $F(V[t], U[t]) = 0$.

As was the case for CT, in practice we will mostly consider explicit systems of the form

$$\begin{aligned} v[t] &= f(v[t-1], \ldots, v[t-m], u[t], \ldots, u[t-k]) \\ y[t] &= g(v[t]). \end{aligned} \tag{2.44}$$

In this form, DT systems are extremely convenient from a simulation point of view because solving for the new state value $v[t]$ requires only evaluating the function $f()$. For a CT model when using an implicit time-integration scheme, it is necessary to solve a nonlinear system of equations at each step, making simulation extremely computationally expensive.

Despite the computational advantage of simulating an explicit system, there is also a downside to DT systems for simulation. Their discrete nature forces a fixed time-step for integration, making simulation slow and impractical for capturing effects on a time-scales much longer than the discrete step size, which is quite common in RF systems being excited by modulated input signals.

2.5.2 Dissipation and Stability in Discrete Time

All of the previously presented notions of dissipation also apply to DT systems. As was the case for continuous time systems, dissipativity in the discrete time case can also be proven through the use of storage functions [103]. The only difference between the CT case and the DT case is that the dissipation constraint is now used to bound finite increments of the storage function, as opposed to the infinitesimal increment represented by the derivative for the CT case.

Definition 2.5.2. *System* (2.40) *is* ***dissipative*** *with respect to the supply rate* $\sigma(U, V)$ *if there exists a storage function* $L(v) \geq 0$ *such that*

$$L(V_+) \leq L(V_-) + \sigma(U, V) \tag{2.45}$$

for all V_+, V_-, U *satisfying* (2.40).

The energy interpretation of dissipation is equally valid for DT systems, as (2.45) implies that the energy at the current time step is less than the energy at the previous time step plus the energy supplied to the system at the current step. Additionally the notions of internal stability, finite-gain stability, and passivity, along with their corresponding supply rates, also apply to the DT case.

Incremental Stability

For a DT system, incremental stability is defined as follows.

Definition 2.5.3. *System* (2.40) *is* ***incrementally stable*** *if it is well-posed and, given any two sets of initial conditions* $\bar{v}[t_0 - 1], \ldots, \bar{v}[t_0 - m]$ *and* $\hat{v}[t_0 - 1], \ldots, \hat{v}[t_0 - m]$, *the resulting two solutions to* (7.1) *in response to the same input* u *satisfy*

$$\sum_{t=t_0}^{\infty} \|\bar{y}[t] - \hat{y}[t]\|^2 < \infty \tag{2.46}$$

for all initial conditions and inputs.

It has been shown in [56] that (2.40) (assuming $G = y - v_0$, i.e. an input-output system) is incrementally stable if the following dissipation constraint is satisfied

$$(v_0 - \hat{v}_0)^T \Big(F(V,U) - F(\hat{V},U)\Big) - |v_0 - \hat{v}_0|^2 + L(V_-,\hat{V}_-) - L(V_+,\hat{V}_+) \geq 0 \quad (2.47)$$

for all V, U, and all $\hat{V} = [\hat{v}_0, \ldots, \hat{v}_m]$, where L is a non-negative storage function such that $L(V_+, V_+) = 0$. Note that when V, U and $\hat{V}, U$ satisfy (2.1), dissipation constraint (2.47) simplifies to constraint (2.45) with $\sigma = -|v_0 - \hat{v}_0|$, which in turn implies (2.46).

Contraction analysis for a DT model examines the stability of the differential system [52]

$$\begin{aligned} F(V,U) + F_v(V,U)\Delta &= 0 \\ G(y,v_0) + G_v(y,v_0)\delta_0 + G_y(y,v_0)\xi &= 0, \end{aligned} \quad (2.48)$$

where

$$\Delta = \begin{bmatrix} \delta_0 \\ \vdots \\ \delta_m \end{bmatrix}, \qquad \Delta[t] = \begin{bmatrix} \delta[t] \\ \vdots \\ \delta[t-m] \end{bmatrix}$$

$$F_v = \left[\frac{\partial F}{\partial v_0}, \ldots, \frac{\partial F}{\partial v_m}\right], \qquad G_v = \frac{\partial G}{\partial v_0}, \qquad G_y = \frac{\partial G}{\partial y},$$

with $\delta_0 \in \mathbb{R}^N$ and $\xi \in \mathbb{R}^{N_y}$. The system is said to be contracting if the increments Δ and ξ converge to zero exponentially. According to Theorem 3 in [52], if System (2.48) is well-posed and stable in the differential variable Δ, i.e. Δ converges exponentially to zero, for all y, $v_0, \ldots, v_m$, and $u_0, \ldots, u_k$ satisfying $F(v_0, \ldots, v_m, u_0, \ldots, u_k) = 0$ and $G(y, v_0) = 0$, then system (2.1) is incrementally stable. It is often easier to prove stability by examining the differential system (2.48) instead of the original system (2.1).

Chapter 3

Background: Model Reduction

This chapter introduces existing model reduction techniques for linear and nonlinear systems. Special emphasis is placed on those techniques capable of preserving parameter dependence and stability in the reduced models.

3.1 Introduction

3.1.1 Motivation

When analyzing large systems, such as models of RF systems modeled using MNA as described in Section 2.1.2, numerical simulation and analysis is often prohibitively expensive. A state-space model for a single RF inductor can contain thousands of equations, and a model for an array of inductors contains hundreds of thousands of equations, making the system impossible to solve using today's technology. In many other cases the resulting state-space models are not so large as to completely prohibit simulation, but the models may need to be simulated many times while varying system inputs and possibly system parameters. What we mean by 'solve' or 'simulate' in reference to a dynamical system is, given a dynamical system of equations

$$\frac{dx}{dt} = f(x, u, p) \tag{3.1}$$

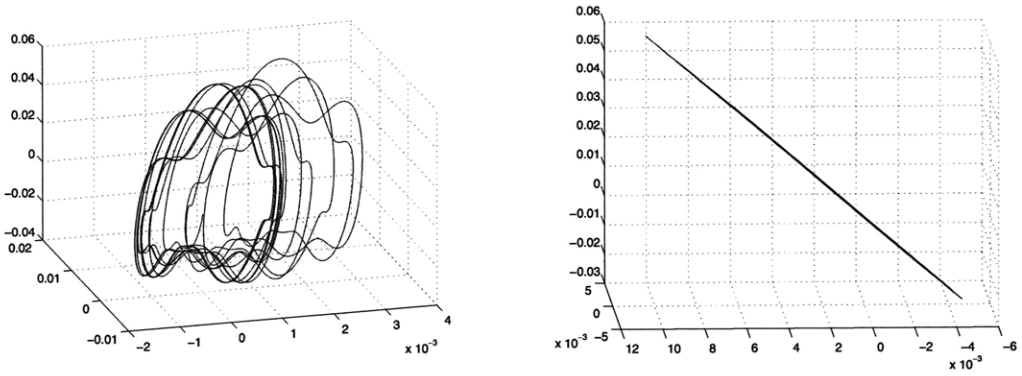
along with input waveform $u(t)$ and set of parameter values p, we wish to solve the equations numerically for the solution $x(t)$ on some time interval $t \in [0, T]$. When considering nonlinear systems, the size of the system is not the only factor determining computational complexity. A moderately sized highly nonlinear system can be extremely computationally expensive to simulate if the nonlinear function f in (3.1) is extremely expensive to evaluate.

In both such cases, the analysis of dynamical systems can benefit greatly from compact representations of the dynamical systems of interest. The task of replacing a given dynamical model with a dynamical model of reduced complexity is referred to as *model reduction.* Since the dynamical models are going to be used for numerical simulation, we generically define complexity as the computational cost associated with solving a set of equations, such as (3.1), numerically.

Although a model may be large and complex, often times the important dynamics primarily evolve in a low-dimensional space. This may be due to redundant states in the model, or 'insignificant' states, such as those that we weakly controllable and weakly observable. Model reduction aims to eliminate such redundant and insignificant dynamics. Additionally, in many cases we are only interested in the input-output behavior of the system, in which case the exact behavior of every internal state is irrelevant, provided the reduced model matches the input-output behavior of the original system.

Consider for example Figure 3-1(a), which plots a solution trajectory $x(t)$ to dynamical system of the form (3.1) for order $N = 3$. Although the trajectory traces out a curve in a three-dimensional space, the curve is actually restricted to a two-dimensional subspace. Figure 3-1(b) shows the same three-dimensional curve in a rotated coordinate system such that it is obvious that the dynamics are confined to a lower dimensional space. Therefore, this third order system of equations can be approximated extremely well with a second order system.

One important question, for which there are many possible solutions, is how to efficiently find this low-dimensional space in which the important dynamics are defined. More importantly, how does one define 'important dynamics'. These questions will be addressed in the following sections.



(a) Curve in a three-dimensional space correspondind to a solution $x(t)$ to a third order system of equations of the form (3.1).

(b) The same trajectory plotted in Figure 3-1(a), but in a rotated coordinate system to illustrate that the solution actually lies in a two-dimensional space.

Figure 3-1: Example illustrating how the solution of a dynamical system might lie in a low-dimensional subspace.

3.1.2 Model Reduction Problem Formulation

In an abstract sense, the model reduction problem can be defined as the following optimization problem: Given an original large model M, find a reduced model $\hat{M} \in \mathcal{M}$, such that the error between the reduced and the original system $||M - \hat{M}||_X$ are minimized for a predetermined complexity of reduced model $\hat{M}$

$$\begin{aligned} \min_{\hat{M}} ||M - \hat{M}||_X \quad & \text{subject to} \\ & \text{complexity}(\hat{M}) \leq q \\ & \hat{M} \in \mathcal{M} \end{aligned} \tag{3.2}$$

Here M represents the original large model being reduced, and could be, for instance, a dynamical system $\dot{x} = f(x, u)$, a collection of trasnfer function samples H_i at frequencies ω_i, or perhaps input-output data pairs $\{u(t), y(t)\}$. $\mathcal{M}$ represents the class of models considered for the reduced model, and could be, for example, the set of all linear state-space models of the form $\dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}u$, or the set of all quadratic discrete-time state-space models. The set $\mathcal{M}$ may also impose constraints such as stability or passivity on the reduced models, or possibly some structure constraint such as block-diagonal matrices for linear

systems. The measure of error, $||M - \hat{M}||_X$, quantifies the accuracy of the reduced model by comparing some aspect of the reduced model to some aspect of the original system. The choice for such a metric typically depends on what information about the large system M is available. For instance, if frequencies response samples of the large model are known, the error could be defined as the maximum frequency response error between two models, $\max_i ||H_i - \hat{H}(j\omega_i)||$ over the given samples. Similarly, if input-output pairs $\{u(t), y(t)\}$ from the large system are available, then the measure of error could be defined as the maximum output error of the reduced model in response to the given inputs, $\max_i ||y_i - \tilde{y}_i||$, where $\tilde{y}$ is the output of reduced model $\hat{M}$ in response to the given inputs $u(t)$. The complexity q of the reduced model can refer to, for instance, the order of a linear system, or the number of parameters describing a nonlinear function.

In some cases it is more desirable to define a strict accuracy requirement for the reduced model and instead minimize the complexity

$$\begin{aligned} \min_{\hat{M}} \ & \text{complexity}(\hat{M}) \quad \text{subject to} \\ & |||M - \hat{M}||_X \leq \epsilon \\ & \hat{M} \in \mathcal{M}. \end{aligned} \tag{3.3}$$

In this thesis we will specifically consider reducing dynamical systems, and usually those having the form $\dot{x} = f(x)$. In practice, when reducing such nonlinear models, it is extremely difficult to precisely define a measure of accuracy and complexity for the reduced model. And although we may not be explicitly solving an optimization problem such as (3.2), the end goal of model reduction remains the same: construct a reduced model that is accurate in some measure and is of minimal complexity.

3.1.3 Projection Approaches

The most common approach to model reduction, for both linear and nonlinear systems, is what's referred to as 'projection'. The basic idea is to approximate the solution x in

low-dimensional subspace by parameterizing it in terms of a set of basis functions, V,

$$x = V\hat{x},$$

and then to solve a reduced system of equations for the corresponding weights $\hat{x}$ for the basis functions. In general, we shall assume that $x \in \mathbb{R}^N$ and $\hat{x} \in \mathbb{R}^q$ for $q << N$, meaning that $V \in \mathbb{R}^{N \times q}$ is a 'tall and skinny' matrix. With this approximation, which can be thought of as a non-invertible change of coordinates, instead of solving for x, we shall be solving for $\hat{x}$, which represent weights for the basis vectors in V with which we are approximating the solution.

To illustrate how the columns of the matrix V define a reduced space in which the solution x is well-approximated, consider the example shown in Figure 3-1(a), where it was previously shown that three-dimensional trajectory can be well-approximated in a two-dimensional subspace. The vectors V_1, V_2 shown in Figure 3-1(b) define this reduced subspace, and can therefore be used as the columns for the projection matrix, i.e. $V = [V_1,\ V_2]$.

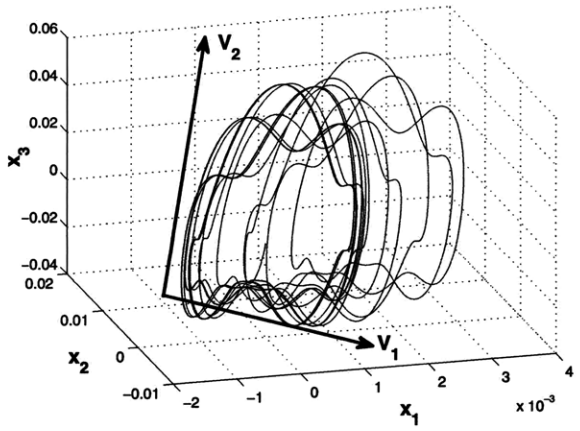


Figure 3-2: The vectors V_1 and V_2 span the two-dimension space in which the solution is well-approximated, as clear from Figure 3-1(a).

Once the projection matrix V defining the reduced space in known, it is next necessary to derive a reduced set of equations that can be solved for the reduced variables $\hat{x}$. Given a system of equations, such as $\dot{x} = f(x)$, we first apply the projection approximation

$x = V\hat{x}$, resulting in

$$V\dot{\hat{x}} \approx f(V\hat{x}).$$

This relation is now only an approximate equality because the system of equations depends on only q variables (the q coefficients for the basis function columns of V), but there are still N equations, and the under-determined system is unlikely to be solved exactly by any $\hat{x}$. Since only q equations are required to solve for q unknowns (assuming some notion of linear independence among the equations), and to make the relation exact, we reduce the number equations using a testing matrix, or left-projection matrix, $U \in \mathbb{R}^{N \times q}$ as follows

$$U^T V\dot{\hat{x}} = U^T f(V\hat{x}).$$

Such a reduction can also be interpreted as forcing the residual $r(\hat{x})$, defined as the equation error resulting from the projection approximation

$$r(\hat{x}) = V\dot{\hat{x}} - f(V\hat{x}),$$

to be orthogonal to the space spanned by the columns of U

$$U^T r(\hat{x}) = 0 \quad \longrightarrow \quad U^T V\dot{\hat{x}} = U^T f(V\hat{x}).$$

It is common practice to select U and V to be bi-orthonormal such that $U^T V = I$, but this is not required and does not affect the reduced model, as only the spans of the columns of U and V define the reduced model.

Thus, the task when using projection reduction methods is to find the optimum choices for V and U. Ideally, we would like to find these projection matrices without having to first solve the original system of equations.

While the above approach assumes a linear projection, $x = V\hat{x}$, it is also possible that the solution lives on a manifold that is not linear, resulting in a nonlinear projection $x = V(\hat{x})$. A nonlinear projection will typically allow for a "smaller" set of unknown parameters (i.e. smaller reduced order q), but at the cost of increased complexity required

to describe the nonlinear manifold and resulting nonlinear reduced model. Such approaches have been attempted in the past [34], and we even propose a nonlinear mapping for reducing the number of equations in Chapter 6 in order to preserve stability, but in general such nonlinear projections are not common.

3.2 Model Reduction for Linear Systems

In this section we introduce the projection framework for model reduction of linear systems, along with several common approaches for constructing the projection basis.

3.2.1 Projection Framework for Linear Systems

Consider a linear dynamical system described using the following state-space representation

$$E\dot{x} = Ax + Bu \qquad\qquad y = C^T x. \tag{3.4}$$

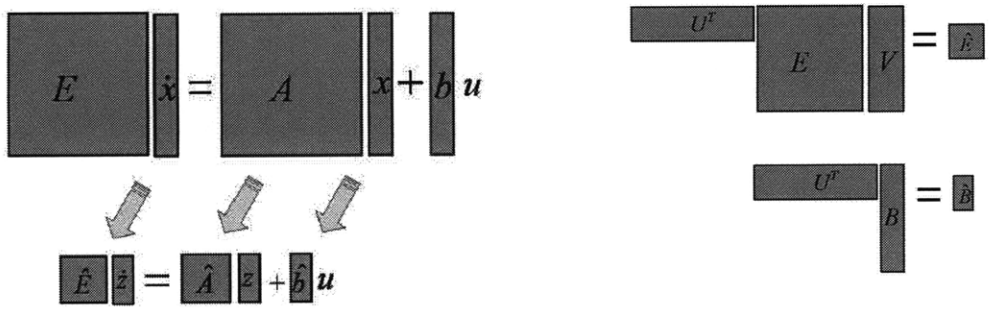
The projection framework described previously in section 3.1.3 calls for approximating the solution in a low-dimensional space, $x = V\hat{x}$, and reducing the number of equations with a left-projection matrix U, resulting in the reduced order system

$$U^T EV\dot{\hat{x}} = U^T AV\hat{x} + U^T Bu \qquad\qquad y = C^T V\hat{x}. \tag{3.5}$$

If we define the matrices $\hat{E} = U^T EV \in \mathbb{R}^{q\times q}$, $\hat{A} = U^T AV \in \mathbb{R}^{q\times q}$, $\hat{B} = U^T B \in \mathbb{R}^{q\times N_u}$, and $\hat{C} = V^T C \in \mathbb{R}^{q\times N_y}$, then the resulting reduced order model consists of q equations and q unknowns

$$\hat{E}\dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}u, \qquad\qquad y = \hat{C}^T z. \tag{3.6}$$

A graphical representation of this reduction of order through projection is shown in Figure 3.2.1.



(a) Visual representation of model order reduction applied to a linear state-space model.

(b) Projection framework reduces the size of individual matrices in the state-space model.

Figure 3-3: Graphical representation of projection framework applied to a linear system.

The complexity of the reduced linear system (3.6) is substantially lower than the complexity of the original large system (3.4) when $q << N$ because the complexity of a linear system scales with the order of the system. Typically the reduced system will be dense even if the original large system is sparse, but even in the worst case simulation cost of the reduced system will be substantially cheaper provided $q << N$.

In the following subsections we will present several common approaches for selecting the projection vectors in the matrices V and U.

3.2.2 Frequency Response of Linear Systems

In many cases it is more convenient to describe linear systems in the frequency, or Laplace, domain. In the Laplace domain linear system (3.4) is expressed as

$$sEx = Ax + Bu \qquad\qquad y = C^T x, \tag{3.7}$$

where s is the Laplace variable. The transfer function of a linear system is defined as the ratio of output to input at a particular frequency s

$$H(s) = \frac{y(s)}{u(s)} = C^T \left(sE - A\right)^{-1} B. \tag{3.8}$$

The frequency response is an extremely useful tool for linear system analysis. When excited with a sinusoidal input, a linear system produces a sinusoidal output of the same

frequency, but with amplitude and phase shifted, as defined by the magnitude and angle respectively of the transfer function at that frequency. For more information, see references [58, 63].

In many model reduction approaches for linear systems, the goal is to construct the reduced model such that it matches the frequency response of the large system over some range of frequencies. Some projection approaches attempt to implicitly match such information in the time-domain, while other transfer function fitting approaches explicitly attempt to fit a transfer function corresponding to a reduced system, as opposed to constructing a reduced state-space model [36, 33, 86].

3.2.3 Moment Matching Approaches

Moment matching can refer to any number of projection techniques wherein the projection vectors are selected to ensure the transfer function of the reduced model matches some number of values and derivatives of the transfer function of the large model at a prescribed set of frequency values.

Initially, such techniques were developed to facilitate the computation of propagation delays in digital circuits for timing analysis. The asymptotic waveform evaluation (AWE) approach created a reduced model by explicitly match moments about DC frequency [72, 71, 73]. Since explicit computation of moments is an ill-conditioned problem, implicit projection techniques using the Arnoldi process (an orthonormalization procedure) were then developed [84, 83] to make moment matching numerically robust. Later on the projection approach was extended to enable matching about non-zero frequencies [31, 32], and also generalized so that input and output moments can be implicitly matched separately using the left and right projection matrices, using the Lanczos procedure (a method for bi-orthonormalization) referred to as Pade via Lanczos (PVL) [28, 30, 4]. Recently, there has been interest in preserving the structure of the original system matrices in the reduced model through projection [29, 105].

Here we present the basic idea behind matching moments at various frequencies using projection vectors. Consider a linear system in the frequency domain, as shown in (3.7).

Rearranging to solve for x and then expanding the result in a Taylor series yields the following

$$x = [I - sM]^{-1}\tilde{B}u = \sum_{n=0}^{\infty} s^n M^n \tilde{B}u, \tag{3.9}$$

$M = (s_a E - A)^{-1}E$ and $\tilde{B} = (s_a E - A)^{-1}B$. The terms $M^n\tilde{B}$ are referred to as the *moments* of the expansion. This sequence of vectors is also referred to as the *Krylov vectors* of $M, \tilde{B}$ (or possibly *shifted Krylov vectors* because the expansion is being performed about frequency s_a).

If the projection matrix is chosen such that

$$\{\tilde{B}, M\tilde{B}, M^2\tilde{B}, ..., M^{q-1}\tilde{B}\} \subseteq \text{range}(V), \tag{3.10}$$

where $\text{range}(V) = \{x \in \mathbb{R}^N | x = V\hat{x}\ ,\ \hat{x} \in \mathbb{R}^q\ ,\ V \in \mathbb{R}^{N\times q}\}$, then the resulting reduced system transfer function

$$H(s) = \hat{C}^T\left(s\hat{E} - \hat{A}\right)^{-1}\hat{B},$$

where $\hat{B} = U^T B$, $\hat{E} = U^T E V$, $\hat{A} = U^T A V$, and $\hat{C} = V^T C$, will match the first q moments of the Taylor series expansion in the Laplace variable s of the large system transfer function about $s = s_a$. It is possible to combine vectors obtained from different frequency expansion points to obtain accurate matching at different frequencies.

If, in addition, the columns of U are chosen such that

$$\{C, MC, M^2C, ..., M^{q-1}C\} \subseteq \text{range}(U), \tag{3.11}$$

where $\tilde{M} = -[(s_b E - A)^{-1}E]^T$, then the resulting reduced system transfer function will also match the first q moments of the Taylor series expansion in the Laplace variable s of the large system transfer function about $s = s_b$. Together it is possible to implicitly match a total of $2q$ moments in the reduced model.

Although moment matching guarantees local accuracy of the reduced model transfer function, it makes no guarantees about the global accuracy of the reduced model. That is, there are no constraints on the behavior of the reduced model far away from the expansion

points about which moments are matched.

3.2.4 SVD Approaches

An alternative class of projection based methods are those that construct the important subspaces V and U using information from time-domain simulation data. That is, rather than attempting to match the frequency response of the original system, given a collection of state data X resulting from simulation of the original system, one can find the best q basis vectors for approximating the data X, and use those basis vectors to define the low order space, i.e. select them as the columns of V. These "snapshot" methods, such as Proper Orthogonal Decomposition [7] (also referred to as Principal Component Analysis (PCA) and Karhumen-Love (KVL) Expansion), provide the optimal q^{th} order basis for reconstructing the simulated data. The hope is that this will also provide a good set of basis vectors for constructing other solutions to the system. Specifically, given a set of data X, where $x_i \in X$ is a solution $x(t, u)$ at some time point t corresponding to some input u or a steady state solution for some sinusoidal input at frequency f, the subspace is constructed from the q largest singular vectors of X. In these cases the left-projection matrix U is chosen to equal the right-projection matrix V such that the error is orthogonal to the optimal basis V.

Such techniques have also been applied to frequency domain data. Instead of using state data in the time-domain, one can solve for vectors in the frequency domain and use the SVD to find the best low-order approximation to that data [102]. In the EDA community this is often referred to as Poor Man's TBR [70].

3.2.5 Balanced-Truncation

Balanced truncation, originally developed by the control community [59, 99], is one of the few projection approaches that possesses global error bounds on the accuracy of the resulting reduced model. The basic idea behind balanced truncation, also referred to as truncated balanced realizations (TBR), is to perform a change of coordinates $\tilde{x} = Tx$ to a coordinate system where the states $\tilde{x}$ are ordered from most important to least important, and then

truncate the least important states. A coordinate transform combined with a truncation can be viewed as a projection.

In TBR, the "important" states are defined as those that are either very controllable or very observable. Conversely, the states to be truncated are those that are neither controllable nor observable. The controllability and observability of states can be quantified via the controllability and observability Grammians, respectively. We will not delve into the specific algorithmic details of TBR here because they are not necessary for this work. However, it is an important approach that needs to be mentioned because of its global error bound guarantee and preservation of stability. For details on the TBR algorithm, see references [21, 97].

Although TBR provides a global error bound (and a guarantee of stability, which will be discussed more in section 3.4), it is rarely used in EDA for very large scale problems because the computational complexity is $O(N^3)$. Recently, much effort has been focused on solving efficiently matrix equations using iterative methods to help extend TBR to truly large sized problems [5, 6].

3.3 Parameterized Model Reduction for Linear Systems

Many times the systems we are trying to model possess dependence on a set of design parameters p

$$\dot{x} = f(x, u, p). \tag{3.12}$$

Here p could represent for instance geometrical parameters, such as wire width or transistor sizing, or material properties, such as wire conductivity. In these cases it is desirable to preserve this parameter dependence in the reduced model

$$\dot{\hat{x}} = \hat{f}(\hat{x}, u, p) \tag{3.13}$$

such that the reduced model is able to accurately predict the system outputs in response to both changes in the input u and parameters p.

For projection methods, the projection matrix V should be able to capture the system behavior in response to changes in parameter values as well as changes to inputs. In some sense, all of the previous reduction techniques are parameterized because they preserve the dependence of the system in response changes in the input, which can be thought of as a time-varying parameter. However, we would like to generalize to techniques that are capable of handling large numbers of parameters and give flexibility to preserve system behavior in response to parameter values in desired regions of interest.

3.3.1 Parameterized Moment Matching

The moment matching approach from section 3.2.3 was first extended to the case of a single parameter in [101], and then extended to an arbitrary number of parameters in [22]. Consider a linear system whose dynamical descriptor matrices in the Laplace domain are functions of the Laplace frequency variable s, and of some other geometrical parameters, $s_1, ..., s_\mu$,

$$E(s, s_1, ..., s_\mu)x = Bu \tag{3.14}$$

where $E \in \mathbb{R}^{N\times N}$. Using a polynomial fitting technique and introducing additional parameters $\tilde{s}$, as shown in [22, 24], one can approximate the parameterized system as

$$[\tilde{E}_0 + \tilde{s}_1\tilde{E}_1 + ... + \tilde{s}_P\tilde{E}_P]x = Bu \tag{3.15}$$

where $\tilde{E}_i \in \mathbb{R}^{N\times N}$.

System (3.15) can be rearranged and expanded in a Taylor Series to obtain

$$x = [I - \tilde{s}_1M_1 - ... - \tilde{s}_PM_P]^{-1}\tilde{B}u \tag{3.16}$$

$$= \sum_n (\tilde{s}_1M_1 + ... + \tilde{s}_PM_P)^n\tilde{B}u \tag{3.17}$$

$$= \sum_n \sum_{k_1} \cdots \sum_{k_P} \left[F^n_{k_1,\ldots k_P}(M_1, \ldots, M_P)\tilde{B}u\right] s_1^{k_1} \ldots s_P^{k_P}, \tag{3.18}$$

where $\tilde{B} = \tilde{E}_0^{-1}B$, $M_i = -\tilde{E}_0^{-1}\tilde{E}_i$, and $F_i \in \mathbb{R}^{N\times N}$. The formulae for F_i are long and convoluted. Here we present only the formula for the simplified case $P = 2$ where the pattern of the vectors F_i is more perceptible. Detailed recursive formulae for the calculation

of F_i with arbitrary P can be found in [22]. If $P = 2$, system (3.15) becomes

$$[\tilde{E}_0 + \tilde{s}_1\tilde{E}_1 + \tilde{s}_2\tilde{E}_2]x = Bu, \tag{3.19}$$

and (3.18) becomes

$$\sum_n \sum_k \left[F_k^n(M_1, M_2)\tilde{B}u\right] s_1^{n-k} s_2^k,$$

where $\tilde{B} = \tilde{E}_0^{-1}B$, $M_i = -\tilde{E}_0^{-1}\tilde{E}_i$, and

$$F_k^n(M_1, M_2) = \left\{ \begin{array}{cc} 0 & \text{if } k \notin 0, 1, ..., n \\ I & \text{if } m = 0 \\ M_1 F_k^{n-1}(M_1, M_2) + M_2 F_{k-1}^{n-1}(M_1, M_2) & \text{otherwise} \end{array} \right\}.$$

This recursive formula generates vectors of the form

$$\tilde{B}, M_1\tilde{B}, M_2\tilde{B}, M_1^2\tilde{B}, (M_1M_2 + M_2M_1)\tilde{B}, M_2^2\tilde{B}, \ldots$$

If we now construct the projection matrix V such that

$$\left[\tilde{B}, M_1\tilde{B}, M_2\tilde{B}, M_1^2\tilde{B}, (M_1M_2 + M_2M_1)\tilde{B}, \ldots\right] \subseteq \text{range}(V)$$

for the $P = 2$ case, and

$$\{F_0\tilde{B},\ F_1\tilde{B}, \ldots ...\} \subseteq \text{range}(V) \tag{3.20}$$

for arbitrary P, then the P-variable Taylor series expansion of the transfer function of reduced order system

$$[\hat{E}_0 + \tilde{s}_1\hat{E}_1 + \ldots \tilde{s}_P\hat{E}_P]\hat{x} = \hat{B}u$$

will match exactly the p-variable Taylor series transfer function moments of original system (3.15), where $\hat{E}_i \in \mathbb{R}^{q\times q} = V^T\tilde{E}_iV$ and $\hat{B} \in \mathbb{R}^{q\times r_i} = V^T\tilde{B}$.

It is noted in [22] that for a large number of parameters P and a modest number of moments m matched for each parameter, this method may generate systems of substantial order $O(P^m)$.

3.4 Stable model Reduction for Linear Systems

A difficult, yet crucial, aspect of model reduction is the preservation of stability in the reduced model. Most real physical systems behave in a stable manner, and it is therefore extremely important to preserve such behavior in reduced models. Unfortunately, preserving stability does not come free, and it is typically obtained either at the cost of accuracy, or by spending additional computational effort. Preserving stability and passivity is particularly important for linear systems because it is possible to construct a reduced model that matches the original model transfer function very accurately in the frequency-domain, even though the model is in fact unstable or non-passive. Thus, the model will appear to be accurate, but once it is used for time-domain simulations it would produce unbounded or unphysical results.

3.4.1 Galerkin Projection for Definite Systems

One benefit of dealing with definite systems, which are systems where $E = E^T \succ 0$, $A \preceq 0$, and $B = C$, as defined in section 2.3.5, is that definiteness is preserved under a congruence transform (also known as a Galerkin projection). Recall from Section 2.3.5 that a definite system is both stable and passive.

A Galerkin projection corresponds to selecting the left-projection matrix U such that $U = V$, where V is the right-projection matrix.

Theorem 3.4.1. *If linear system*

$$E\dot{x} = Ax + Bu \qquad\qquad y = C^T x$$

is definite, i.e. $E = E^T \succ 0$, $A \preceq 0$, *and* $B = C$, *then the reduced model*

$$(V^T EV)\dot{\hat{x}} = (V^T AV)\hat{x} + V^T Bu \qquad\qquad y = C^T V x$$

where V *has full column rank, is also a definite system, and therefore also stable and passive.*

This result is obvious if we recall the definition of a definite matrix. For example, matrix A is negative semidefinite if $x^T Ax \leq 0$ for all x. Therefore, the reduced matrix $\hat{A}$ satisfies

$$\hat{x}^T \hat{A} \hat{x} = \hat{x}^T (V^T AV) \hat{x} = (\hat{x}^T V^T) A (V\hat{x}) = y^T Ay \leq 0$$

and is thus also negative semidefinite.

One important observation is that Theorem 3.4.1 places no constraints on how the right-projection matrix V is constructed. The columns can be selected as Krylov vectors, using POD vectors, or any other technique. Additionally, note that when using a Galerkin projection one is sacrificing accuracy in order to obtain stability, because the left-projection matrix U could alternatively be chosen to match additional moments of the large system's transfer function. In the event that the original system is stable but not definite, then a Galerkin projection is not guaranteed to preserve stability. The resulting reduced model *might* be stable, but it is not guaranteed.

The Galerkin projection has been popular for many years in the EDA community because most very large linear systems are definite systems. This is a result of using RLC networks to model parasitic structures.

3.4.2 Solving Matrix Equations

In the more general case where the original large order system is not definite, there are alternative approaches for guaranteeing stability and passivity in the reduced model. Balanced truncation truncation always preserves stability and there are variations that are capable of preserving also passivity [69]. One less common stability-preserving projection is as follows.

Theorem 3.4.2 ([55]). *Consider a linear descriptor system*

$$E\dot{x} = Ax + bu \tag{3.21}$$

where (E, A) is a Hurwitz pair. Let P and Q be SPD matrices satisfying

$$E^T PA + A^T PE = -Q \tag{3.22}$$

let V be an orthonormal projection matrix such that $x = Vz$ and $z \in \mathbb{R}^q$ where $q << N$. If U is defined by

$$U^T = (V^T E^T PEV)^{-1} V^T E^T P,$$

and $\hat{A} = U^T AV$, then $\hat{E} = U^T EV = I$ and the reduced order system

$$\dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}u \tag{3.23}$$

is stable.

This approach is appealing becomes it gives us freedom in selecting the right-projection matrix V. That is, given any right-projection matrix V, there exists a left-projection matrix U such that the reduced model is stable. However, as with TBR, the approach in Theorem (3.4.2) requires $O(N^3)$ complexity due to solving matrix equation (3.22).

3.5 Model Reduction for Nonlinear Systems

In this section we discuss projection approaches for nonlinear systems. As will be shown, in the nonlinear case there are additional difficulties that were not present when reducing linear systems.

3.5.1 Difficulty with Projection

Consider a nonlinear system $\dot{x} = f(x)$, and apply orthonormal left and right projection matrices U and V respectively, resulting in the reduced system

$$\dot{\hat{x}} = U^T f(V\hat{x}).$$

In general, the complexity of this "reduced" model still depends on N, the order of the large system. Evaluating the reduced function $U^T f(V\hat{x})$ requires projecting the reduced state $\hat{x}$ back to the full space via $V\hat{x}$, evaluating the original nonlinear functions on the large vectors $f(V\hat{x})$, and projecting the result back into the reduced space $U^T f$.

This was not a problem for linear systems because when $f(x)$ is a linear function, the projected terms can be multiplied out ahead of time

$$U^T f(V\hat{x}) = U^T AV\hat{x} = \hat{A}\hat{x},$$

resulting in a term, $\hat{A}\hat{x}$, that depends only on the reduced order q.

As a result, model reduction of nonlinear systems requires, in addition to a projection $x = V\hat{x}$, a low-complexity approximation of the projected nonlinear function $U^T f(V\hat{x})$.

3.5.2 TPWL for Highly Nonlinear Systems

For highly nonlinear systems, local methods such as polynomial expansion are not sufficient to capture all important nonlinear effects. Instead, we might consider using a series of local low-complexity approximations, and interpolating between these simple local models. Consider a nonlinear descriptor system of order N

$$\frac{\partial}{\partial t}\left[q(x)\right] = f(x) + bu, \qquad y = c^T x. \tag{3.24}$$

We shall approximate the nonlinear functions $f(x), q(x)$ over important regions of the state-space using a convex combination of locally accurate affine functions

$$f(x) \approx \sum_i w_i(x)\left[A_i x + k_i\right],$$
$$q(x) \approx \sum_i w_i(x)\left[E_i x + h_i\right],$$

where

$$A_i = \frac{\partial f(x)}{\partial x}\Big|_{x_i} \qquad k_i = f(x_i) - A_i x_i$$
$$E_i = \frac{\partial q(x)}{\partial x}\Big|_{x_i} \qquad h_i = q(x_i) - E_i x_i$$

are linearizations of $f(x)$, and $w_i(x)$ are weighting functions such that $w_i \in [0,1]$ and $\sum_i w_i = 1$. Two examples of possible weighting functions are ([78, 96])

$$w_i(x) = \frac{\exp\left(\frac{-\beta||x-x_i||_2^2}{\min_k ||x-x_k||_2^2}\right)}{\sum_j \exp\left(-\frac{\beta||x-x_j||_2^2}{\min_k ||x-x_k||_2^2}\right)}$$

or

$$w_i(x) = \frac{(e^{-\beta||x-x_i||_2^2})^{-k}}{\sum_j (e^{-\beta||x-x_j||_2^2})^{-k}}$$

A PWL approximation to system (3.24) takes the form [79]

$$\frac{d}{dt}\left[\sum_i w_i(x)(E_i x + h_i)\right] = \sum_i w_i(x)\left(A_i x + k_i\right) + Bu. \tag{3.25}$$

The PWL approximation (3.25) is still a large model (order N), so we now introduce a linear projection $x = V\hat{x}$, where $\hat{x} \in \mathbb{R}^q$ and $q << N$, such that each linear system is projected into the subspace spanned by the columns of V, creating the piecewise-linear reduced order model [79]

$$\frac{d}{dt}\left[\sum_i w_i(\hat{x})(\hat{E}_i\hat{x} + \hat{h}_i)\right] = \sum_i w_i(\hat{x})\left[\hat{A}_i\hat{x} + \hat{K}_i\right] + \hat{B}u$$

in which we have defined $\hat{E}_i = V^T E_i V$, $\hat{A}_i = V^T A_i V$, $\hat{k}_i = V^T k_i$, $\hat{h}_i = V^T h_i$, $\hat{B} = V^T B$, and $\hat{c} = V^T c$. Here we have reduced the model complexity by both approximating the nonlinear functions $f(x), q(x)$, and by reducing the number of state variables.

We now have a procedure for generating PWL reduced order models for highly nonlinear systems, but in order to obtain an accurate model, the local linearizations must to

selected in an intelligent manner. We cannot afford to sample uniformly the entire state-space, so we instead only create local models at "important" regions of the space. Since we are interested in capturing the input-output behavior of the original nonlinear model, we define important regions as those that are visited by the state $x(t)$ of system (3.24) in response to typical inputs of interest $u(t)$. Thus, given a set of "training inputs" $u(t)$, system (3.24) is simulated to obtain a set of "training trajectories" $x(t)$, and then the local models are created by linearizing the nonlinear system at points along the training trajectories. See reference [77] for more details on TPWL models.

3.5.3 Projection vectors for Nonlinear Systems

As was the case for linear projection methods, there are many possible methods for constructing the projection matrices U and V when reducing nonlinear systems. However, unlike in the linear case, there are few accuracy guarantees imposed on the reduced model by these choices. For instance, there is no simple notion of a transfer function for a nonlinear system, so we cannot use moment matching approaches to generate vectors.

The most common approach is to use the SVD techniques based off of time-domain data. Some works have also used linear techniques to generate vectors from linearizations of the nonlinear system. For instance, Krylov vectors from the individual linearized systems [79, 78] have been used for TPWL model reduction, as have TBR vectors from linearized models [98].

Some nonlinear balancing techniques have been proposed in [82], but at the moment there exists no efficient methods for computing the necessary transformations.

3.6 System Identification

In this section we introduce system identification techniques as an alternative approach to model reduction.

3.6.1 An Alternative Approach to Model Reduction

The previously described model reduction techniques all involve "reducing" the large nonlinear systems, produced by circuit schematics or parasitic extractors, by explicitly projecting the original large system of equations. However, one shortcoming of such techniques is the extreme difficulty in preserving stability in the reduced model. Additionally, such model reduction approaches are quite restrictive because they require explicit knowledge of the nonlinear functions $f(x)$ and $q(x)$ describing the original large system of equations. This requires knowing not only the schematic of the circuit, which is typically readily available, but also the exceedingly complicated transistor models. Transistor models typically consist of a collection of empirical formulas with hundreds of special cases, and hundreds of fitted parameters, which are not as easily available. Furthermore, nonlinear models produced by existing projection reduction techniques (such as piecewise linear methods) are often not easily realizable using existing commercial circuit simulators due to complicated nonlinearities in the model, and would require internal modifications that, although simple, cannot be implemented by the end user of such simulators without access to the source code.

An alternative approach to achieve the same final goal, i.e. the automatic generation of accurate compact models, without "reducing" a given large system, but rather using a system identification approach to model reduction. The term system identification (SYSID) refers to the task of finding a stable dynamical model of low complexity that delivers the best match for a collection of dynamical input-output (or input-state-output) data. In classical control applications, the data is usually available in the form of actual physical measurements, and SYSID provides adequate models for systems for which no reliable first principles equations are available, either due to parameter uncertainty or system complexity. In integrated circuit applications, SYSID is the only viable option in generating compact models of circuits blocks when only input-output physical measurements are available. In addition, although it is true that in many cases circuit schematics of the original system are actually available, SYSID often still remains the most practical solution to compact modeling.

Within the control community, SYSID for linear-time-invariant (LTI) system is well understood and mature [50]. One can argue that also some of the approaches developed by the Electronic Design Automation community for LTI model order reduction could be interpreted as SYSID approaches, such as those based on transfer function fitting via least squares or optimization techniques [8, 37, 20, 88].

Conversely, SYSID for nonlinear systems is still a problem that needs to be addressed on a case by case basis [38, 51]. Among the most general and used approaches in behavioral modeling one finds the Volterra series method [18, 54, 90]. In some more specific approaches one assumes an internal structure (e.g. a Wiener [9, 39], or Wiener-Hammerstein [50, 38, 94, 43] or Wiener-Hammerstein with feedback structure [89]), and proceeds in identifying the coefficients for such structures [85, 74]. This forms the basis of the block diagram oriented system identification schemes such as [106]. As a general observation, a significant difficulty in implementing any kind of SYSID based approach is caused by lack of efficient SYSID tools for generic nonlinear systems. Typically one is required to solve, for instance, very expensive non-convex quadratic programming optimization problems.

3.6.2 Robust Nonlinear Identification

In standard system identification (SYSID) techniques for both discrete and continuous time systems, data is exclusively available in the form of a finite length vector of input-state-output $(\tilde{u}[t], \tilde{v}[t], \tilde{y}[t])$, or just input-output, sampled pairs. Such data can be generated either by physical measurements of a fabricated integrated circuit, or by simulation of an available circuit schematic. The objective of a SYSID algorithm is to generate automatically from training data, a dynamical system description, such as the following implicit DT model

$$F(v[t], \ldots, v[t-m], v[t-1], \ldots, u[t-k]) = 0, \qquad G(y[t], v[t]) = 0, \tag{3.26}$$

such that the predicted output of the identified model minimizes the "output error", and it is "easy" to compute each new output sample when given previously computed past values

of the input and output samples.

Definition 3.6.1. *Given a sequence of inputs $\tilde{u}[0], \ldots, \tilde{u}[T]$, the corresponding states $\tilde{v}[0], \ldots, \tilde{v}[T]$, and outputs $\tilde{y}[0], \ldots, \tilde{y}[T]$, the **output error** of an identified model is defined as*

$$E(F, G, \mathcal{X}) = \sum_t |y[t] - \tilde{y}[t]|^2, \tag{3.27}$$

where $y[t]$ are solutions to the identified model in response to training data inputs and initial conditions $\tilde{v}[t-1], \ldots, \tilde{v}[t-m]$, and $\mathcal{X}$ represents the training data set containing all given $\tilde{u}[t], \tilde{v}[t], \tilde{y}[t]$ pairs.

In general, minimization of the true output error is computationally extremely difficult as it is a highly non-convex problem. Most approaches suggested by the classical literature in system identification [50] instead attempt to minimize the overall "equation error".

Definition 3.6.2. *The **equation error** is defined as the sum of squared mismatches obtained from evaluating the identified model* (3.26) *on all of the training data samples $(\tilde{u}[t], \tilde{v}[t], \tilde{y}[t]) \in \mathcal{X}$*

$$\tilde{E}(F, G, \mathcal{X}) = \sum_t |F(\tilde{V}[t], \tilde{U}[t])|^2 + |G(\tilde{y}[t], \tilde{v}[t])|^2 \tag{3.28}$$

It is, however, misleading to assume that a small equation error always leads to a small output error. It is possible to identify unstable models whose system equations are satisfied accurately by the given data, resulting in small equation error, but produce unstable outputs during simulation, resulting in large output error.

It has been shown in [56] that if system (3.26) is incrementally stable (i.e. satisfies (2.47)), then the equation error for the resulting system provides an upper bound for the model output error over the training data set. Minimization of this upper bound subject to incremental stability constraint can be cast as a semidefinite program, however, this approach typically produces overly conservative upper bounds for the output error due to the strong constraints imposed by (2.47).

Chapter 4

Stable Model Reduction for Indefinite Linear Systems

4.1 Introduction

In Section 3.4, several stability-preserving model reduction approaches for linear systems were discussed. However, all of these methods are either restrictive, by imposing strong constraints on the matrices of the large system being reduced, or are too computationally expensive to handle very large systems. Many stable systems are not described by semi-definite matrices. Such matrices frequently arise when modeling nonlinear systems or when considering linearizations of nonlinear systems, such as small-signal analog circuit models created by linearizing around the DC operating point. There exist a small number of methods, such as balanced truncation, that preserve stability [60] and passivity [66] for asymmetric and indefinite systems, but these techniques are computationally extremely expensive.

Additionally, models of stable passive systems extracted by field solvers may often be unstable due to error arising from discretization and from ignoring higher-order physical effects. These errors become a serious problem when attempting to connect unstable system blocks into a larger stable system, or when modeling distributed systems with frequency-dependent matrices, which requires interpolation between matrices [23]. To our knowl-

[0]Some of the material in this chapter has been previously published in [16].

edge, no projection method is able to reliably create accurate stable reduced models from originally unstable systems.

In this chapter we address the issue of efficient stable reduction of indefinite stable systems and mildly unstable systems. Specifically, we present a new approach for computing projection matrices that are guaranteed to preserve stability (and passivity). The projection matrices are typically chosen such that the right projection matrix guarantees accuracy (e.g. matching moments), and the left-projection matrix guarantees stability. This is achieved by deriving a set of linear constraints based on the idea of Lyapunov functions, resulting in a linear matrix inequality (LMI) whose solution is a stability-preserving projection matrix. Particular attention was devoted to formulating the LMI problem such that it is independent of the size of the original system, and only depends on the order of the reduced model.

This chapter is organized as follows: In section 4.2, we explain how existing stability-preserving projection techniques can be interpreted as a preservation of dissipation. In Section 4.3 we derive a set of linear stability constraints for the reduced model in terms of projection matrices, allowing for stability to be enforced, rather than preserved, during projection. In Section 4.4 we formulate optimization problems aimed at finding the optimal set of projection matrices that enforce both accuracy and stability. Section 4.5 presents several non-projection approaches that can enforce stability using using the previously derived stability constraints. Finally, Section 4.6 tests the proposed stabilizing methods on several examples and compares their speed and accuracy to traditional methods. Background information on stability-preserving model reduction can be found in Section 3.4.

4.2 Preserving Dissipation Through Projection

This section explains how stability-preserving reduction techniques can be viewed as a preservation of storage functions and dissipation constraints. This allows us to interpret all stability-preserving techniques as a generalization of the Galerkin projection in a different coordinate system. These ideas are based on a generalization of results presented in section 3.4.2 and [55].

4.2.1 Restricting Storage Functions to a Subspace

Consider a linear system

$$E\dot{x} = Ax + Bu, \qquad y = C^T x \tag{4.1}$$

that is known to be dissipative with respect to a quadratic supply rate $\sigma(x, u)$ with quadratic storage function $L(x)$

$$L(x) = x^T E^T P E x \tag{4.2}$$

$$\sigma(x, u) = x^T Q x + x^T R u + u^T S u, \tag{4.3}$$

meaning the following dissipation constraint is satisfied

$$2x^T E^T P \left(Ax + Bu\right) \leq \sigma(x, u). \tag{4.4}$$

Recall that (4.4) is obtained by forcing $\dot{L}(x) \leq \sigma(x, u)$ for all x, u. Since dissipation constraint (4.4) must hold for all solutions x to system (4.1), then it must hold for all solutions restricted to the subspace spanned by a matrix $V \in \mathbb{R}^{N \times q}$, i.e. $\{x | x = V\hat{x}\}$,

$$\dot{L}(V\hat{x}) = 2\hat{x}^T V^T E^T P \left(AV\hat{x} + Bu\right) \leq \sigma(V\hat{x}, u). \tag{4.5}$$

If we define the following reduced order matrices

$$\hat{A} = V^T E^T P A V, \qquad \hat{B} = V^T E^T P B, \tag{4.6}$$

then constraint (4.5) can be rewritten as

$$\dot{L}(V\hat{x}) = 2\hat{x}^T (\hat{A}\hat{x} + \hat{B}u) \leq \sigma(V\hat{x}, u). \tag{4.7}$$

Additionally, we may define the matrices $\hat{Q}$ and $\hat{R}$ such that

$$\begin{aligned}\sigma(V\hat{x}, u) &= \hat{x}^T V^T Q V \hat{x} + u^T R u + \hat{x}^T V^T S u \\ &= \hat{x}^T \hat{Q} \hat{x} + \hat{x}^T \hat{R} u + u^T S u.\end{aligned} \tag{4.8}$$

With the above defined matrices it is possible to construct a reduced model that is dissipative with respect to supply rate $\sigma(V\hat{x}, u)$ with storage function $L(V\hat{x})$.

Theorem 4.2.1. *If System* (4.1) *is dissipative with respect to quadratic supply rate* (4.3) *with quadratic storage function* (4.2)*, then the reduced model*

$$\hat{E}\dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}u, \qquad y = \hat{C}^T\hat{x} \tag{4.9}$$

with the defined matrices

$$\hat{A} = V^T E^T P A V \qquad \hat{B} = V^T E^T P B$$
$$\hat{E} = \hat{E}^T \succ 0 \qquad \hat{C} = V^T C$$

where $\hat{E}$ is any SPD matrix, is dissipative with respect to supply rate

$$\hat{\sigma}(\hat{x}, u) = \hat{x}^T \hat{Q} \hat{x} + \hat{x}^T \hat{R} u + u^T S u,$$

with the defined matrices

$$\hat{Q} = V^T Q V \qquad \hat{R} = V^T S$$

with storage function $\hat{L}(\hat{x}) = \hat{x}^T \hat{E} \hat{x}$.

Proof. We wish to show that $\hat{L}(\hat{x})$ is a storage function and that the dissipation constraint holds with supply rate $\hat{\sigma}(\hat{x}, u)$. First, by definition $\hat{E}$ is SPD, and therefore $\hat{L}(\hat{x}) \succ 0$.

Second, by (4.7) and (4.8) we find that

$$\dot{\hat{L}}(\hat{x}) = 2\hat{x}^T(\hat{A}\hat{x} + \hat{B}u) = \dot{L}(V\hat{x})$$
$$\hat{\sigma}(\hat{x}, u) = \hat{x}^T\hat{Q}\hat{x} + \hat{x}^T R u + u^T\hat{S}u = \sigma(V\hat{x}, u).$$

Thus, the dissipation constraint holds

$$\dot{\hat{L}}(\hat{x}) = \dot{L}(Vz) \leq \sigma(V\hat{x}, u) = \hat{\sigma}(\hat{x}, u)$$

and the system is dissipative. □

Here we have used the known storage function and dissipation of the original system to define the reduced matrices such that the reduced model dissipation is that of the original system restricted to the subspace spanned by V. Based on the supply rate chosen, it it may be possible to reduce the constraints on some of the system matrices. For example, when considering internal stability ($\sigma(x, u) = x^TQx$ and $u = 0$), there are no constraints on the reduced matrices $\hat{B}$ and $\hat{C}$. Figure 4-1 illustrates this idea of restricting a storage function to a subspace. In this example $L(x) \succ 0$ is a quadratic function, and it is obvious that $L(V\hat{x})$ is also a PSD quadratic function.

Although Theorem 4.2.1 guarantees a dissipative model, it provides no explicit guarantees that reduced model (4.9) will be an accurate model of original system (4.1). Since the model is dissipative for any SPD matrix $\hat{E}$, we may use this degree of freedom to enforce additional accuracy in the model. In the following section we present one possible approach for selecting $\hat{E}$.

4.2.2 Projection Interpretation

In the previous section it was shown that the reduced system (4.9) is stable for any SPD matrix $\hat{E}$ when $\hat{A} = V^TE^TPAV$ and $\hat{B} = V^TE^TB$. Here we consider one possible such choice: $\hat{E} = V^TE^TPEV$. With this choice, the reduced system matrices become

$$\hat{E} = V^TE^TPEV, \quad \hat{A} = V^TE^TPAV, \quad \hat{B} = V^TE^TB, \quad \hat{C} = V^TC, \tag{4.10}$$

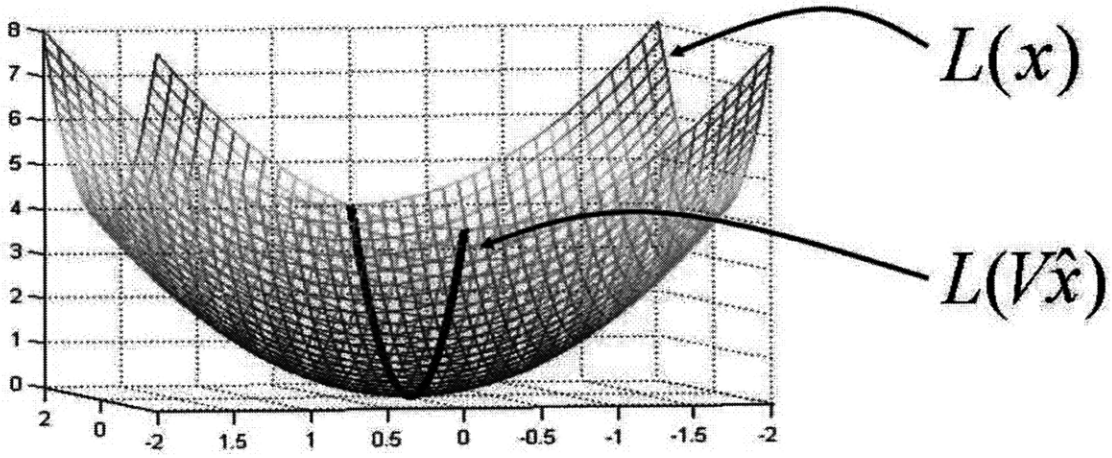


Figure 4-1: Example of restricting a storage function to a subspace. From this picture it is obvious that $L(v\hat{x}) \succ 0$ when $L(x) \succ 0$.

allowing the reduced system to be written as

$$(V^T E^T P) E V \dot{\hat{x}} = (V^T E^T P) A V \hat{x} + (V^T E^T P) B u, \tag{4.11}$$

which can also be obtained by projection of the original system (4.1) with projection matrices U and V, where $U = PEV$.

Theorem 4.2.2. *If System* (4.1) *is dissipative with respect to quadratic supply rate* (4.3) *with quadratic storage function* (4.2)*, then the reduced model*

$$\hat{E}\dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}u, \qquad y = \hat{C}^T\hat{x}, \tag{4.12}$$

with the defined matrices

$$\hat{E} = U^T E V, \qquad \hat{A} = U^T A V, \qquad \hat{B} = U^T B, \qquad \hat{C} = V^T C, \tag{4.13}$$

where $U = PEV$, is dissipative with respect to supply rate

$$\hat{\sigma}(\hat{x}, u) = \hat{x}^T \hat{Q} \hat{x} + u^T R u + \hat{x}^T \hat{S} u, \tag{4.14}$$

with the defined matrices

$$\hat{Q} = V^T Q V \qquad\qquad \hat{S} = V^T S, \tag{4.15}$$

with storage function $\hat{L}(\hat{x}) = \hat{x}^T \hat{E} \hat{x}$.

Proof. By the choice of U, note that $\hat{E} = V^T E^T P E V$ is a SPD matrix. Therefore, we may apply Theorem 4.2.1, which proves dissipativity of the reduced model. □

We wish to point out here that although Theorem 4.2.2 is simply a reproduction of Theorem 3.4.2 generalized to arbitrary quadratic supply rates, the important contribution of this section is the interpretation of the proposed projection using preservation of dissipation. This interpretation is key to results presented in the following sections, and also several later results in Chapter 6.

When selecting $\hat{E} = V^T E^T P E V$, we are enforcing accuracy in the reduced model through the projection framework. The projection guarantees that the residual of solutions approximated in the space spanned by V are orthogonal to the space spanned by U, i.e.

$$U^T \left(EV\dot{\hat{x}} - AV\hat{x} - Bu \right) = 0. \tag{4.16}$$

Additionally, accuracy may also be enforced by the choice of V, through, for instance, moment matching in the frequency domain, or any other linear projection technique described in Section 3.2.

Generalization of Galerkin Projection

To obtain insight into why the aforementioned projection framework preserves stability, consider the linear system

$$\tilde{E}\dot{x} = \tilde{A}x + \tilde{B}u, \qquad\qquad y = C^T x,$$

$$\tilde{E} = E^T P E, \qquad \tilde{A} = E^T P A, \qquad \tilde{B} = E^T P B, \qquad y = C^T x$$

which is obtained by premultiplying System (4.1) by the matrix $E^T P$. This new linear system is a definite system because $\tilde{E} = \tilde{E}^T \succ 0$ and $\tilde{A} + \tilde{A}^T \prec 0$, thus allowing a congruence transform to preserve stability

$$V^T(E^T P E)V\dot{\hat{x}} = V^T(E^T P A)V\hat{x} + V^T(E^T P B)u. \tag{4.17}$$

The resulting reduced model is the same model obtained from the proposed projection applied to the original system. From this point of view, the projection $U = PEV$ can be interpreted as first transforming the linear system to a coordinate system where it is described by definite system matrices, and then applying a congruence transform.

4.2.3 Extension to Nonlinear Systems

Preservation of dissipation through projection can also be applied to nonlinear systems. For example, if the nonlinear system $\dot{x} = f(x)$ is dissipative with respect to supply rate $\sigma = 0$ for storage function $L(x) = x^T x$, i.e. dissipation constraint

$$x^T f(x) + f(x)^T x \leq 0$$

holds for all solutions x, then the restriction of this dissipation constraint to $x = V\hat{x}$,

$$\hat{x}^T V^T f(V\hat{x}) + f(V\hat{x})^T V\hat{x} \leq 0,$$

certifies dissipation of the reduced system $\dot{\hat{x}} = V^T f(V\hat{x})$ with storage function $\hat{L}(\hat{x}) = \hat{x}^T\hat{x}$.

In general, it is more difficult to find storage functions for nonlinear systems, and difficult to "check" whether dissipation constraints are satisfied globally. However in the cases where a storage function is known, the previously described approach can be used to guarantee stability of a nonlinear reduced model.

4.3 Introducing Dissipation Through Projection

4.3.1 Difficulties with Dissipation Preservation

The previous section presented a projection approach for creating stable reduced models from stable large models that is independent of the structure of the large matrices, i.e. does not require a definite original large system. However, there is one main problem with such dissipation-preservation approach.

The projection approach described in Section 4.2 assumes explicit knowledge of a storage function matrix P for the large system. In some cases this matrix is known or can be constructed, as described in Section 2.2.5, but in general this is not the case. Often times a given linear system results from an unknown formulation, making it impossible to construct P based on physical intuition from the problem. Even if the formulation is known and P can be explicitly constructed, it is possible that numerical errors in the system description prevent the assumed P from actually being a storage function matrix for that particular system. In these cases it is possible to solve a Lyapunov matrix equation to find P, however this requires N^3 complexity, and if one is going to spend that much computation, then it might be better to use for instance balanced truncation instead.

The biggest problem, however, is that it is even possible that the given linear model of a stable passive system is in fact not stable or passive. This could be a result of numerical errors during the formulation of the large system resulting from discretization error in the field solver, when using for instance boundary element methods or finite element method formulations. In most cases it is not even possible to know whether or not the large system is stable, since checking stability of a linear system is typically N^3 complexity. If the large

linear system is ill-conditioned it is possible some eigenvalues have very small positive real parts. Even though effects resulting from such eigenvalues should be negligible, when simulated in time-domain long enough the results will become unstable.

As an alternative approach, it may be beneficial, or even necessary, to *enforce* stability or passivity in the reduced model through projection, rather than attempting to *preserve* it.

4.3.2 Stability and Passivity Constraints

Although the following approach can be extended to arbitrary notions of dissipation resulting from quadratic supply rates, for the remainder of this chapter we will consider only asymptotic stability and passivity, as these are the most important properties from a time-domain modeling point of view.

Given a linear descriptor System (4.1) and a pair of projection matrices U and V, we wish to find conditions under which the reduced order model

$$U^T E V \dot{\hat{x}} = U^T A V \hat{x} + U^T B u, \qquad y = C^T V \hat{x} \tag{4.18}$$

is stable. In this work we are primarily interested in the selection of the left-projection matrix U. That is, given a right-projection matrix V (generated by any well known technique in VLSI model order reduction such as moment matching, POD or Poor-Man's TBR), we will propose a routine for automatically generating left-projection matrices U which preserve stability.

From Section 2.3, we know the reduced system will be stable if there exists an SPD matrix $\hat{P}$ such that

$$\hat{E}^T \hat{P} \hat{A} + \hat{A}^T \hat{P} \hat{E} = -Q_2 \prec 0, \tag{4.19}$$

for some SPD matrix Q_2. Recall that this equation is the result of the condition $\dot{\hat{L}}(\hat{x}) \prec 0$, corresponding to the Lyapunov function $\hat{L}(\hat{x}) = \hat{x}^T \hat{E}^T \hat{P} \hat{E} \hat{x}$. That is, given reduced model matrices $\hat{E}, \hat{A}, \hat{B}, \hat{C}$, if there exists an SPD solution $\hat{P}$ to condition (4.19), then $\hat{L}(\hat{x})$ is a Lyapunov function for System (4.18), and thus System (4.18) is asymptotically stable.

Condition (4.19), however, is not desirable from a computational point of view, because it is quadratic in U

$$V^T E^T U \hat{P} U^T A V + V^T A^T U \hat{P} U^T E V = -Q_2 \prec 0, \tag{4.20}$$

making it an extremely difficult system of equations to solve.

As was discussed in Section 4.2.2, it is always possible to transform a stable linear system into a definite system. *Considering directly only definite reduced models*, the stability conditions for system 4.18 simplify to

$$\hat{E} \succ 0, \qquad \hat{A} + \hat{A}^T \prec 0. \tag{4.21}$$

In terms of projection matrices, these new constraints are linear in both U and V

$$\begin{aligned} U^T E V &= Q_1 \succ 0 \\ U^T A V + V^T A^T U &= -Q_2 \prec 0. \end{aligned} \tag{4.22}$$

Prop. 4.3.1 (Equality of Stability Constraints)**.** *The stability conditions* (4.20) *and* (4.22) *are equivalent, provided the reduced matrix $U^T E V$ has full rank.*

Proof. Assume $U, \hat{P}, Q_2$ solve (4.20). We may then choose $\tilde{U} = U \hat{P} U^T E V$ as a solution to (4.22) for Q_2, and we find that $Q_1 = V^T E^T U \hat{P} U^T E V$, which is SPD because it is a congruence transform of an SPD matrix $\hat{P}$ with a full rank matrix operator $U^T E V$. On the other hand, assume $\tilde{U}, Q_1, Q_2$ solves (4.22). We may then choose $U = \tilde{U}$ as a solution to (4.20) for Q_2 and for $\hat{P} = (\tilde{U}^T E V)^{-1} = Q_1^{-1}$, which is an SPD matrix because Q_1 is an SPD matrix. Thus, the two sets of constraints are equivalent. □

As a result of this equivalence between stability conditions from here on we shall consider only the set of linear constraints

$$U^T E V \succ 0 \tag{4.23}$$

$$U^T A V + V^T A^T U \prec 0. \tag{4.24}$$

Any U satisfying these constraints will create a stable reduced model. In addition, note that there is no requirement that E be invertible, so we may handle the case of singular E matrix, provided $U^T EV$ is non-singular.

It is important to note here that if for numerical reasons we need to further enforce orthogonality between U and EV, then given a U which solves (4.23) and (4.24), we may always redefine $U = U(V^T E^T U)^{-1}$ to ensure that $U^T EV = I$, where I is an identity matrix. This redefinition does not affect the stability or accuracy of the reduced model because it does not change the column span of U. Such a redefinition can be thought of as multiplying both sides of the reduced model by $\hat{E}^{-1}$, which we know is invertible because we have forced it to be an SPD matrix (Q_1). Additionally, we point out that instead of fixing the right-projection matrix U and finding a stabilizing left-projection V, it is also possible to do the opposite and fix the left-projection U while finding a stabilizing right-projection V.

In this section and for the remainder of the paper we consider the asymptotic stability constraint that Q_2 is an SPD matrix. It is also of course possible to consider the looser constraint that Q_2 is an SPSD matrix without changing any of the results presented here.

Passivity Enforcement

If, in addition to stability, we wish to enforce passivity, we add the third constraint that $\hat{B} = \hat{C}$, since we are forcing the reduced system to be definite. Since $\hat{B} \neq \hat{C}$ in general even if $B = C$, because $\hat{B} = U^T B$ while $\hat{C} = V^T C$, we must enforce this as an additional constraint

$$U^T EV \succ 0 \tag{4.25}$$

$$U^T AV + V^T A^T U \prec 0 \tag{4.26}$$

$$U^T B = V^T C. \tag{4.27}$$

Thus, passivity is preserved by adding one additional linear constraint. In the remainder of this chapter, in order to keep the explanation simple, we will refer only to stability and consider only the first two linear constraints; however, ***it is always possible and a trivial***

extension to add the third linear constraint to ensure passivity when needed.

4.3.3 Existence of Stabilizing Projections

In the previous section we derived a set of linear constraints for a stabilizing projection matrix U. In this section we examine the conditions guaranteeing that solutions to the system of constraints exist.

First, note that it is possible to transform the matrix equations into a single linear system. We may rewrite equation (4.24) as a single matrix linear in U and concatenate the result with equation (4.23) to obtain

$$M^T U = B^T, \qquad M = [V_e,\ V_a], \qquad B = [Q_1,\ -Q_2]. \tag{4.28}$$

where $V_e = EV$ and V_a is a rearrangement (4.24) such that $U^T V_a = U^T A V + V^T A^T U$.

Since $U \in \mathbb{R}^{N\times q}$ and $M \in \mathbb{R}^{N\times 2q}$, this is an underdetermined system of equations with q^2 equations and Nq unknowns. If the columns of the constraint matrix M are linearly independent, then there exist an infinite number of stabilizing solutions for any SPD matrices Q_1, Q_2. Note that there is no dependence on the eigenvalues of (E, A). In fact, there exist stabilizing solutions ***even if the original large-order system is unstable***. Thus, not only can we preserve stability, but we also have the option of enforcing stability when needed.

If the constraint matrix M in (4.28) has linearly dependent columns and the original system is stable, then there exist stabilizing solutions only for certain SPD matrices Q_1, Q_2.

Prop. 4.3.2 (Existence of Stabilizing Projection). *If (E, A) is an asymptotically stable pair, then for any orthogonal right-projection matrix V and any SPD matrix Q_2, there exists at least one SPD matrix Q_1 such that the linear constraints* (4.23) *and* (4.24) *are consistent (i.e. a solution exists).*

Proof. By assumption, (E, A) is an asymptotically stable pair, and therefore given any SPD matrix $\tilde{Q}$ there exists an SPD matrix P such that

$$E^T P A + A^T P E = -\tilde{Q}. \tag{4.29}$$

Furthermore, given any right-projection matrix V, it is also true that

$$V^T E^T P A V + V^T A^T P E V = -V^T \tilde{Q} V, \tag{4.30}$$

and that $V^T \tilde{Q} V$ is also SPD because congruence transforms preserve definiteness. Thus, we select $U = PEV$ and $Q_2 = V^T \tilde{Q} V$ to satisfy constraint (4.24), and find that $U^T E V = V^T E^T P E V \succ 0$ satisfies constraint (4.23). Lastly, we need to show that there exists such a $\tilde{Q}$ for any Q_2. Since Q_2 is SPD, we may factor it such that $q^T q = Q_2$, where $q \in \mathbb{R}^{q \times q}$. Now define $\tilde{q}^T = [X,\ \mathcal{N}(V)]$ where $\mathcal{N}(V)$ is the null space of V^T, and $X \in \mathbb{R}^{N \times q}$ is the solution to $X^T V = q$ (which exists because it is an underdetermined system of equations and V has linearly independent columns by assumption). Finally, define $\tilde{Q} = \tilde{q}^T \tilde{q}$, resulting in $Q_2 = V^T \tilde{Q} V$. □

A second important observation is that not only are there an infinite number of stabilizing solutions U to the underdetermined System (4.28), but there are also an infinite number of subspaces spanned by the set of stabilizing solutions U. That is, if we define $T = \text{range}(U)$ to be the space spanned by the columns of U, then there exist an infinite number of stabilizing spaces T. This is an important distinction to make, as there exist many different representations for a single subspace which will produce reduced models all identical up to an invertible change of coordinates. Furthermore, the feasible set is a convex cone. Let $U_a, U_b \in \mathcal{U}$, i.e.

$$\begin{aligned} U_a^T E V &= Q_{1a} \succ 0, & U_a^T A V + V^T A^T U_a &= Q_{2a} \prec 0 \\ U_b^T E V &= Q_{1b} \succ 0, & U_b^T A V + V^T A^T U_b &= Q_{2b} \prec 0, \end{aligned}$$

then $U = \alpha U_a + \beta U_b \in \mathcal{U}$ for all $\alpha, \beta > 0$

$$U^T E V = (\alpha U_a + \beta U_b)^T E V = \alpha U_a^T E V + \beta U_b^T E V = \alpha Q_{1a} + \beta Q_{1b} \succ 0,$$

and

$$\begin{aligned} U^T AV + V^T AU &= (\alpha U_a + \beta U_b)^T AV + V^T A^T (\alpha U_a + \beta U_b) \\ &= \alpha (U_a^T AV + V^T A^T U_a) + \beta (U_b^T AV + V^T A^T U_b) \\ &= \alpha Q_{2a} + \beta Q_{2b} \prec 0. \end{aligned}$$

If the constraint matrix M has linearly dependent columns and the original system (E, A) is **not** stable, then there may or may not exist a stabilizing projection. It is possible to construct systems where there does not exist a stabilizing projection, e.g. select $E = A = I$ as identity matrices. However, if the number of unstable modes is smaller than the order of the reduced model, then there always exists a stabilizing projection pair U, V that is equivalent to truncating the unstable modes.

We would like to point out here that creating stable reduced models of unstable original systems may in general result in inaccurate reduced systems. Ideally the large-order extracted systems would be stable and the projection framework would simply preserve stability, but unfortunately this is not the case in the vast majority (if not complete totality) of field solvers and parasitic extractors used in current VLSI design flows to extract the original large order systems. We have found that in most cases the numerically unstable modes in such extracted linear systems correspond to very small eigenvalues and are typically not excited by inputs of interest. Explicitly truncating, or stabilizing, these unstable modes in the large-order system would therefore likely yield accurate stable models. However, this operation is computationally infeasible since it would require the preliminary eigendecomposition of the original large order system. Nevertheless, unstable modes must be eliminated, because a poorly-chosen projection could amplify the unstable modes and cause major instabilities in the reduced model. The approach presented in this paper may be used to eliminate such parasitic-extraction-generated artificial numerical instabilities, avoiding the expensive eigendecomposition of the original large-order system.

The unstable systems we are trying to address here are the ones generated by parasitic extractors discretizing PDEs, hence they typically contain a very small number of numerically generated positive eigenvalues. For those systems we are often able to find stabilizing

projections for unstable models.

4.3.4 Analytic Solutions for Independent Constraints

In the case of linearly indepedent constraints, as described in section 4.3.3, it is possible to explicitly solve the stability constraints

$$U^T E V = Q_1 \succ 0 \qquad\qquad U^T A V + V^T A^T U = -Q_2 \prec 0$$

for a stabilizing left-projection matrix U. It is possible to solve this system directly using Kronecker identities to transform the matrix equation into one large linear system, but this would be very expensive with a computational cost on the order of $(Nq)^3$, where N and q are the sizes of the large-order and reduced-order systems respectively. Our goal is to eliminate dependence of the complexity on N.

As stated previously, we can transform the matrix equation into a linear system. Consider factoring the matrix V_a into two matrices Q_a and R_a such that $V_a = Q_a R_a$, where $Q_a \in \mathbb{R}^{N \times q}$ and $R_a \in \mathbb{R}^{q \times q}$. One approach is to use a modified Graham Schmidt orthogonalization for Q_a. This is cheap computationally because only the first q orthogonal columns are needed. For the sake of consistency we may do the same for the other constraint, factoring V_e into Q_e and R_e. We now define a new pair of variables $Y_a = U^T Q_a$ and $Y_e = U^T Q_e$. The result is the linear system

$$[Q_e \; Q_a]^T U = [Y_e \; Y_a]^T \tag{4.31}$$

where $Y_e, Y_a \in \mathbb{R}^{q \times q}$ are the solutions to the order q matrix equations

$$R_e^T Y_e = Q_1, \qquad\qquad R_a^T Y_a + Y_a^T R_a = -Q_2,$$

which can be solved in at worst $O(q^3)$ operations. Note that since V_e and V_a are linearly independent, then Q_e and Q_a are linearly independent, $[Q_e \; Q_a]^T$ has $2q$ linearly independent columns, and thus there always exists a solution to linear system (4.31). This procedure is summarized in Algorithm 1, and the resulting underdetermined system can be solved

Algorithm 1 Linear Constraint Transformation

1: Given matrices $Q_1, Q_2 \succ 0$, and E, A, V

2: Define $V_e = EV$, $V_a = AV$, and then factor into matrices $Q_e, Q_a \in \mathbb{R}^{N\times q}$, $R_e, R_a \in \mathbb{R}^{q\times q}$ such that

$$V_e = Q_e R_e \qquad\qquad V_a = Q_a R_a$$

3: Solve order q matrix equations separately for Y_e and Y_a

$$Y_e^T R_e = Q_1 \qquad\qquad Y_a R_a + R_a^T Y_a = -Q_2$$

4: Solve underdetermined linear system for U

$$[Q_e\ Q_a]^T U = [Y_e\ Y_a]^T$$

efficiently using any standard technique.

4.4 Optimal Projection

In the previous section we have shown that given any right projection matrix V, there exist left-projection matrices U that enforce stability and passivity in the reduced model. However, we have not yet addressed the issue of additionally enforcing accuracy through the choice of U. For the dissipation-preserving projection $U = PEV$ from section 4.2.1, there exist many possible choices for P resulting in stable models, and it is difficult to know a priori which will yield the most accurate reduced model over the range of interest. When enforcing dissipation through projection as discussed in section 4.2.2, it was shown in section 4.3.3 that there exist many stabilizing U matrices. For example, Figure 4-2 plots the quality factor of an RF inductor model (solid line) along with several sixth-order stable reduced models (dashed lines) all constructed with the same right-projection matrix V and different left-projection matrices U. All models match the transfer function at the places where V enforces moments matching, at 1GHz and 10GHz, but some models are more accurate than others at frequency points in between. In this section we present several methods for selecting U such that both stability and accuracy are enforced in the reduced model.

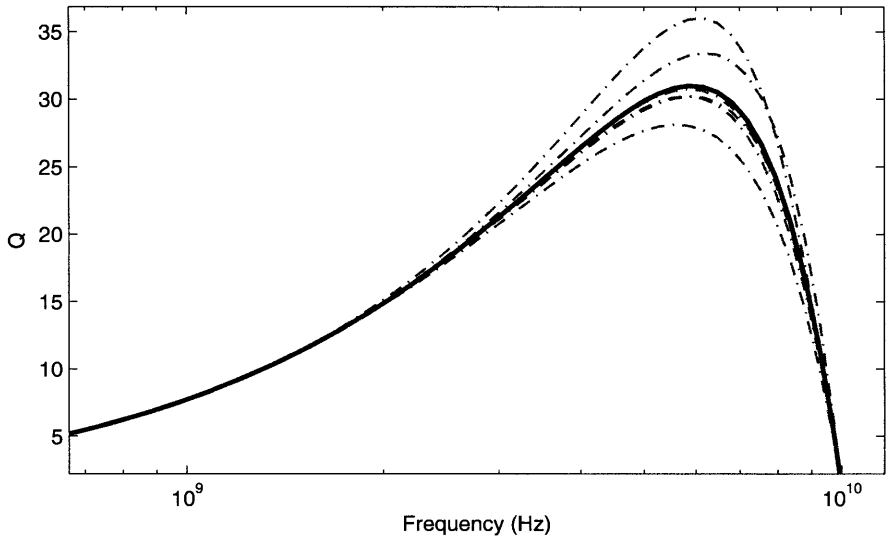


Figure 4-2: Six different sixth-order reduced models (dashed lines) created with the same right-projection matrix V matching moments at $0.1, 1, 10$ GHz and four different left-projection matrices U. The solid line indicates the response of the original transmission line model. The curves represent quality factors of an RF inductor model.

4.4.1 Optimal Stabilizing Projection

Given the large number of stabilizing left projection matrices U, as shown for example in Figure 4-2, it might be beneficial to spend more computation in an attempt to increase the reduced model accuracy by searching for a stable *and* accurate projection. Utilizing the linear stability constraints derived in Section 4.3.2, we can formulate the following optimization problem

$$\begin{aligned} &\min_U f(U) \quad \text{subject to} \\ &\quad U^T E V \succ 0 \\ &\quad U^T A V + V^T A^T U \prec 0. \end{aligned}$$

Here $f(U)$ is some objective function chosen to enforce accuracy through the left-projection matrix U.

Let $\mathcal{U}$ represent the space of all stabilizing projection matrices U, and assume we are given a normalized left-projection matrix U_0 that does not preserve stability. One choice

for objective function $f(U)$ is to try to minimize the "distance" between the given left-projection matrix U_0, constructed to enforce accuracy, and some stabilizing solution $U \in \mathcal{U}$. To this end, we will define $U = U_0 + U_\delta$ as a stabilizing solution, and shall solve only for the perturbation term U_δ

$$\begin{aligned} \min_U \ & ||U_\delta||_S \quad \text{subject to} \\ & U_\delta^T EV + \Delta Q_1 \succ 0 \\ & U_\delta^T AV + V^T A^T U_\delta + \Delta Q_2 \prec 0, \end{aligned} \tag{4.32}$$

where $\Delta Q_1 = U_0^T EV$ and $\Delta Q_2 = U_0^T AV + V^T A^T U_0$. What we are essentially doing here is searching for a minimal perturbation to the matrix U_0, which was constructed for accuracy, such that the resulting matrix $U = U_0 + U_\delta$ is accurate and guarantees stability. In this setup our measure of optimality $||\cdot||_S$ should quantify the distance between the subspaces spanned by U and U_0. There are many possible norms one can use here to measure this distance, and we present two such options in the following sections.

4.4.2 Minimal Perturbation for Stabilizing Projection

Given a matrix U_0 that was constructed to ensure accuracy, we wish to find a stabilizing projection U that is "close" to U_0 in some sense. One possible choice for measuring this distance between projection matrices is to use the familiar and computationally affordable L_2 norm

$$\begin{aligned} \min_U & ||U - U_0||_2^2 \quad \text{subject to} \\ & U_\delta^T EV + \Delta Q_1 \succ 0 \\ & U_\delta^T AV + V^T A^T U_\delta + \Delta Q_2 \prec 0. \end{aligned} \tag{4.33}$$

This measure is not necessarily ideal because while it measures the distance between the matrices U and U_0, it does not exactly measure the distance between the subspaces spanned by these matrices. It is possible that $||U - U_0||$ could be large while $||U - \tilde{U}_0||$ could be small

when U and $\tilde{U}$ have the same column span. However, one advantage of minimizing the L_2 norm is that optimization problem (4.33) is convex, and furthermore can be formulated as a semidefinite program allowing it to be solved efficiently using known techniques.

As stated, optimization problem (4.33) is still computationally N-dependent because U_δ has size $N \times q$. Since the system is underdetermined, we may fix $N-2q$ variables and solve instead a size $2q$ square system. Define $V_e = EV$ and $V_a = AV$ and consider partitioning U_δ, V_a, and V_e such that U_q, V_{aq}, and V_{eq} are the first $2q$ rows of each respectively. Now the size $N \times q$ system

$$\begin{bmatrix} V_{eq}^T & V_{e2}^T \end{bmatrix} \begin{bmatrix} U_q \\ U_2 \end{bmatrix} = Q_1$$

is equivalent to the size $2q \times 2q$ system

$$V_{eq}^T U_q + (V_{e2}^T U_2) \succ 0,$$

where U_2 is a predetermined size $(N - 2q) \times q$ matrix. We may partition the remaining constraint in a similar manner, resulting in a pair of square size $2q$ constraints and a new optimization problem where we have defined $\Delta Q_1 = V_e^T U_0 + V_{e2}^T U_2$ and $\Delta Q_2 = V_a^T U_0 + U_0^T V_a + (V_{a2}^T U_2 + U_2^T V_{a2})$. In most cases, since we want U_δ to be small, we will select $U_2 = 0$. This new problem has $O(q^2)$ constraints and unknowns, resulting in a cost of $O(q^4)$. A feasible solution to the original problem can then be chosen as $U = [U_q^T \; U_2^T]^T$. Thus we may now compute the stabilizing solution by solving an order-q LMI that has ***no dependence on N***, the order of the original system.

There remains one important detail not to be overlooked. We are assuming that the new order $2q$ system has a solution, but this is not necessarily true for all V_{aq}, V_{eq}. For example, it is possible to choose $2q$ rows from V_a and V_e such that V_{aq} and V_{eq} are all zeroes, resulting in no solution. However, by simply selecting the $2q$ rows such that there are no zero rows in either truncated matrix, we have not found this to be a problem in practice. It is also possible to increase the number of rows in U_q to some number p such that $2q < p < N$ if a stabilizing solution is not found when considering only $2q$ rows. A sample procedure is

Algorithm 2 Stabilizing Projection via Row Perturbation

1: Given E, A, V, U_0, define

$$V_e = EV, \qquad V_a = AV,$$
$$\Delta Q_1 = V_e^T U_0, \qquad \Delta Q_2 = V_a^T U_0 + U_0^T V_a$$

2: Select p nonzero rows from V_e and V_a and denote them V_{ep} and V_{ap} respectively

3: Solve the optimization problem for U_δ

$$\min_{U_\delta} \ ||U_\delta|| \quad \text{subject to}$$
$$U_\delta^T V_{ep} + \Delta Q_1 \succ 0$$
$$U_\delta^T V_{ap} + V_{ap}^T U_\delta + \Delta Q_2 \prec 0.$$

4: Define $U = U_0 + \Delta U$, where ΔU is zeroes except for the p rows corresponding to the rows of U_δ

presented in Algorithm 2.

4.4.3 Moment-Preserving Perturbation for Stabilizing Projection

One problem with the previous approach, i.e. minimizing the L_2 norm $||U_\delta||$, is that if the columns of U were chosen to match moments, then perturbing each column a small amount will ruin all moment matching ability of U despite only making small perturbations. On the other hand, making one large perturbation to a single column of U will only affect the matching of one moment, even though the total perturbation to the matrix is large in the L_2 sense. Therefore, it may be beneficial instead to consider perturbing columns sequentially in order to find a stabilizing U while preserving as much moment-matching as possible.

Let U be a given nominal projection matrix that does not enforce stability, meaning that the stability constraints

$$U^T E V \succ 0, \qquad U^T A V + V^T A^T U \prec 0$$

are **not** satisfied. Now consider $U_k \in \mathbb{R}^{N \times q}$ as a matrix with all zeros except the k_{th} column, such that $U + U_k$ perturbs the k_{th} column of U. We wish to find the minimal perturbation U_k such that the stability constraints are satisfied. Since it is possible that no perturbation

U_k will stabilize the reduced model (since we are only perturbing one column), to make the optimization problem feasible we instead attempt to satisfy the constraints

$$(U+U_k)^T EV + \gamma_1 \succ 0, \qquad (U+U_k)^T AV + V^T A^T (U+U_k) - \gamma_2 \prec 0$$

while minimizing both the norm of U_k and the quantities γ_1, γ_2

$$\begin{aligned} \min_{U_k,\gamma_1,\gamma_2} \; & ||U_k||_2^2 + \gamma_1 + \gamma_2 \quad \text{subject to} \\ & (U+U_k)^T V_e + Q_1 + \gamma_1 \succ 0 \\ & (U+U_k)^T V_a + V_a^T (U+U_k) + Q_2 - \gamma_2 \prec 0. \end{aligned}$$

If there exists a U_k stabilizing the system, then the constraints will be satisfied with γ_1, γ_2 negative. If there exists no such U_k, then γ_1, γ_2 will have to be positive in order for the constraints to be satisfied. In this case, we then attempt to perturb another column (change the value of k), and solve again. This procedure can be iterated over the various columns of U, with the matrix U being updated by the perturbations U_k at each step, until a U is found that stabilizing the reduced model, meaning that γ_1, γ_2 will be negative. This procedure is described in Algorithm 3.

If this procedure terminates after k steps, then the remaining $q-k$ columns of U will match exactly $q-k$ moments as they were originally constructed to do. As in section 4.4.2, it is possible to perturb only some of the entries in the column U_k in order to make the problem computationally cheaper.

4.4.4 Stabilizing Second Stage Reduction

When handing extremely large systems of equations, it is common practice to use Krylov methods as a first stage of reduction, and then use balanced truncation as a second stage. However, if the first stage reduction procedure produces an unstable model, then balanced truncation is not applicable. In this case the stabilizing techniques proposed in the previous sections may be the only possible method for obtaining accurate and stable low-order models.

Algorithm 3 Stabilizing Projection via Column Perturbation

1: Given E, A, V, U_0, define $V_e = EV,\ V_a = AV$

$$V_e = EV, \qquad V_a = AV,$$

2: Set $U = U_0,\ k = 1$
3: **while** $\gamma_1, \gamma_2 > 0$ **do**
4: Update Q_1, Q_2

$$Q_1 \leftarrow V_e^T U \qquad Q_2 \leftarrow V_a^T U + U^T V_a$$

5: Solve for U_k, an $N \times q$ matrix with all zeroes except for the k_{th} column

$$\begin{aligned} \min_{U_k, \gamma_1, \gamma_2} \ & ||U_k||_2^2 + \gamma_1 + \gamma_2 \quad \text{subject to} \\ & U_k^T V_e + Q_1 + \gamma_1 \succ 0 \\ & U_k^T V_a + V_a^T U_k + Q_2 - \gamma_2 \prec 0. \end{aligned}$$

6: Add U_k to the kth column of U
7: move to next column: $k = k + 1$
8: **end while**
9: U now satisfies stability constraints and creates a stable reduced model

4.5 Optimal Non-Projection Approaches

As mentioned in Section 4.2.2, if we define $\hat{A}$ and $\hat{B}$ as in (4.6), then the reduced model (4.9) is stable for any SPD $\hat{E}$. One possible alternative to using the projection approach described in Section 4.3 to define $\hat{E}$ is to identify it through solving optimization problem.

In an optimization formulation, rather than forcing orthogonality of the residual with respect to the space spanned by U, as a projection does, we can instead select $\hat{E}$ to minimize the 'error' of the reduced model on a given set of data subject to the constraint that $\hat{E} = \hat{E}^T$

is SPD. For example, we may define $\hat{E}$ as the solution to the problem

$$\begin{aligned} \min_{\hat{E}} \; & f(\hat{E}, \hat{A}, \hat{B}, \hat{C}) \quad \text{subject to,} \\ & \hat{E} \succ 0 \\ & \hat{A} = V^T E^T P A V \\ & \hat{B} = V^T E^T P B \\ & \hat{C} = V^T C \end{aligned} \tag{4.34}$$

where f is an objective function that measures error of the reduced model. One possible choice for f is the following

$$f(\hat{E}) = \sum_i |\hat{E}\dot{\hat{x}}_i - \hat{A}\hat{x}_i - \hat{B}u_i|^2, \tag{4.35}$$

where $\hat{x}_i = V^T x_i$ are a projection of solutions x_i to the original system (4.1) in response to input u_i. For this particular choice of f, the goal of the optimization procedure is to minimize the equation mismatch over the given training data samples $(\hat{x}_i, u_i)$. With proper selection of f, such as in (4.35), optimization problem (4.36) is convex and can be efficiently solved using existing techniques. Additionally, given that we know one feasible solution from Section 4.2.2 (i.e $\hat{E} = V^T E^T P E V$), the procedure is guaranteed to identify a model no worse than the model obtained via the projection approach in terms of the error metric defined by f.

Furthermore, if we are going to optimize projection matrices, and to avoid the previously described issues associated with requiring explicit knowledge of the storage function matrix P, we can completely ignore projection and optimize directly all projection matrices.

$$\begin{aligned} \min_{\hat{E},\hat{A},\hat{B},\hat{C}} \; & f(\hat{E}, \hat{A}, \hat{B}, \hat{C}), \quad \text{subject to} \\ & \hat{E} \succ 0 \\ & \hat{A} + \hat{A}^T \prec 0 \end{aligned} \tag{4.36}$$

It is important to note that since there is no projection, even if the matrix V was constructed to match moments, it is not guaranteed that the transfer function of the reduced system will match these moments. However, the optimization procedure provides a different measure of accuracy for the reduced model, and this can be sufficient to obtain high accuracy in transfer function matching as well. An alternative approach for generating stable reduced models by minimizing equation error is presented in Chapter 7.

4.6 Results

4.6.1 Implementation

The proposed methods were tested on several linear system examples. All of the following results were generated using Algorithm 2. The optimization problems were solved using the open source solver SeDuMi [91], and were formulated using either YALMIP [45] or POT [57], all of which are freely available. In all of these examples we have chosen $U_0 = V$, where V was constructed to match input moments.

As a last note, we have found that due to the extreme ill-conditioning of the constraint system, one must normalize the constraints such that V_e and V_a are of the same order. This is equivalent to simply scaling the right hand side matrices Q_1, Q_2. This operation is allowed because the matrices would still be SPD.

4.6.2 Inductor Model

The first examples considered are two-turn RF inductor models. The inductor models were created using a public domain electro-magneto-quasi-static (EMQS) mixed-potential integral equation solver [61, 1]. All of the selected extracted large order models are numerically *unstable*, despite modeling passive physical systems.

As a result of instability, the traditional Galerkin projection $U = V$ is not guaranteed to create a stable model. Results from several different inductor models and stable reduced models generated using Algorithm 2 are plotted in Figure 4-3. All reduced models have order $q = 8$, and were each constructed in under 2 seconds. The original large order

systems all have order $N = 647$ and are all unstable.

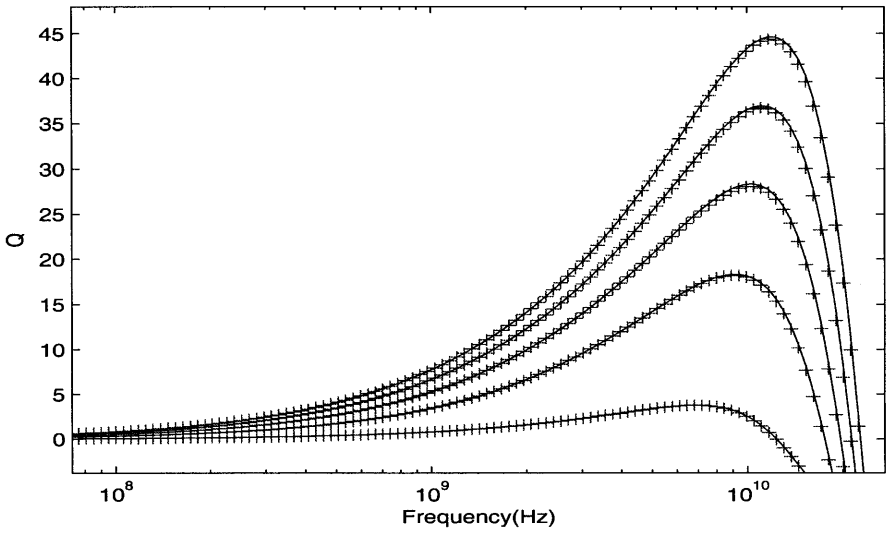


Figure 4-3: Quality factor of two-turn inductor models corresponding to various wire width and wire separation parameter values. The original models (blue solid lines) are unstable and all have order $N = 647$, while the reduced models (red plusses) are stable and all have order $q = 8$.

For these inductor models we cannot compare the computational time required to compute the stabilizing projection to other similar methods, because to our knowledge no other available method is guaranteed to create a stabilizing left-projection matrix U for unstable systems when given a right-projection matrix V. However, it is possible to first stabilize the large-order model, e.g. truncate the unstable modes in the large model via eigendecomposition, and then compute a stabilizing left-projection matrix corresponding to the given right-projection matrix V. Doing so by solving a Lyapunov equation for the large model was 10 times slower than our proposed method for this example, and this does not include the time required to stabilize the large-order model via eigendecomposition. When factoring in the cost of the eigendecomposition, our proposed method is 15 times faster.

4.6.3 Power Grid

The second example is a 3×3 power grid in free space. The grid consists of copper wires with width $2\mu m$ and thickness $2\mu m$, and resulted in an order $N = 1566$ unstable model,

created using the same EMQS solver . Using Algorithm 2 we were able to create an order $q = 10$ stable reduced model by constructing V to match moments and selecting $U_0 = V$. For this example the Galerkin projection $U = V$ resulted in an unstable model. Figure 4-4 plots the real part of the impedance of the original system, the stabilized reduced model, and an unstable reduced model created using a Galerkin projection.

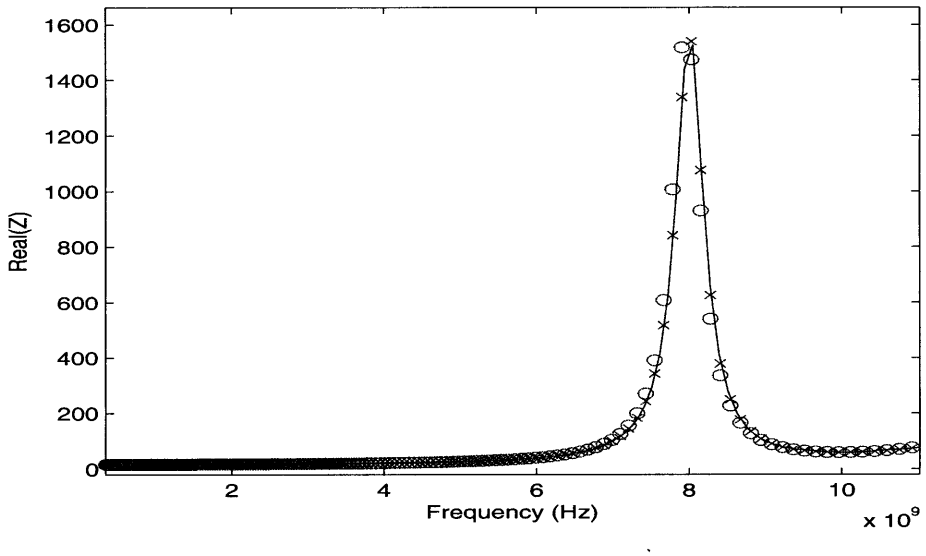


Figure 4-4: Real part of the impedance for a 3×3 power grid. The original model (solid blue line) is unstable and has order $N = 1566$. An unstable reduced model created through Galerkin projection (green crosses) of order $q = 10$ is comparable in accuracy to a stable reduced model (red circles) of order $q = 10$ created by our proposed method.

As a reminder we would like to point out that although the frequency-domain results of the unstable reduced models in Figure 4-4 look nice, it is possible that certain inputs, or simply numerical noise in the simulator, could excite the unstable modes in the model resulting in unphysical behavior. For instance a non-zero initial condition could cause the solution to explode, which cannot happen for a passive system such as a power grid.

4.6.4 Linearized Systems

The final example considered is a nonlinear Microelectromechanical System (MEMS) switch [27]. The need to efficiently construct stabilizing projections for indefinite linearized models arises in small-signal analysis of analog circuits, and when using nonlinear

projection functions to create stable nonlinear reduced models of nonlinear systems [15]. The nonlinear model is obtained from the discretization of a pair of nonlinear PDEs, and the linear model is obtained by linearizing at the equilibrium state. For this system the matrices are neither symmetric nor definite. The original model has order $N = 1680$, and we were able to create an accurate stable reduced model with order $q = 12$.

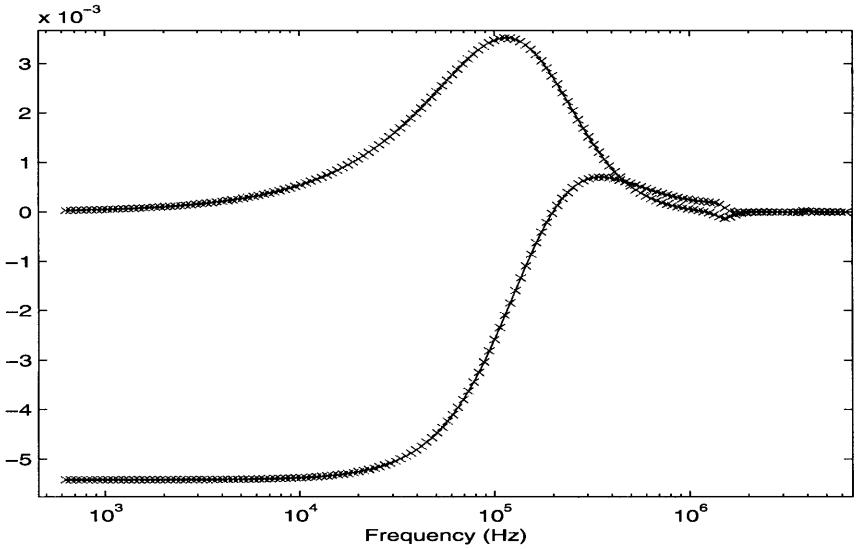


Figure 4-5: Real and imaginary transfer function parts of a linearized model of a nonlinear MEMS device. The original stable model (solid blue line) has order $N = 1680$, while the reduced stable model (red crosses) has order $q = 12$. A Galerkin projection on this indefinite system does not preserve stability.

For this example a Galerkin projection does not preserve stability, and constructing a stability-preserving projection by solving a Lyapunov equation for the full system was 10 times slower than our proposed algorithm.

Chapter 5

Parameterizing Trajectory Piecewise Linear Models

5.1 Introduction

In order to be useful for design optimization, a reduced model must be parameterized in some set of design parameters to allow for design changes. Given a large system depending on some parameters p, we wish to create a reduced model that *preserves* the dependence on the set of parameters

$$\dot{x} = f(x, u, p) \quad \longrightarrow \quad \dot{\hat{x}} = \hat{f}(\hat{x}, u, p).$$

While there exist several parameterized model reduction approaches for linear systems, none are capable of handling nonlinear systems.

In this chapter we present a parameterized model reduction approach for highly nonlinear systems. This approach is based on the trajectory piecewise linear (TPWL) model reduction method combined with a parameterized moment matching approach. To recap briefly, the TPWL approach, described in detail in Section 3.5.2, approximates a nonlinear

[0]The material in this chapter has been previously published in [13, 14].

system with a collection of local linear models that are used for interpolation

$$f(x) \longrightarrow \sum_i w_i(x) A_i x + k_i$$

In addition, we propose an adaptive training scheme which selects parameter-space points for training by using available trajectory information to approximate the system sensitivity to the parameters. Requiring fewer training trajectories reduces the model generation cost and potentially eliminates redundant linear models.

This chapter is organized as follows: Section 5.2 presents the parameterized TPWL approach along with an algorithm for its implementation. Techniques are presented for selecting linearization points for local models and vectors for the projection matrix. In Section 5.3, three parameterized nonlinear system examples chosen to test the proposed method are described in detail. Results from these examples along with algorithm analysis and parameter-space accuracy analysis are presented in Section 5.4.

5.2 Parameterized Model Order Reduction for Nonlinear Systems

5.2.1 PROM Description Derivation

Given a system possessing nonlinear dependence on both the state $x(t)$ and some set of parameters p_i

$$\dot{x} = f\left(x(t), p_1, p_2, \ldots, p_\mu\right) + B(p_1, \ldots, p_\mu)u(t), \qquad y = C^T x \tag{5.1}$$

where $x \in \mathbb{R}^N$, $f \in \mathbb{R}^N$, $B \in \mathbb{R}^{N \times N_u}$, and $C \in \mathbb{R}^{N \times N_y}$, our goal is to obtain a nonlinear parameterized reduced model

$$\dot{\hat{x}} = \hat{f}\left(\hat{x}(t), p_1, p_2, \ldots, p_\mu\right) + \hat{B}(p_1, \ldots, p_\mu)u(t),$$

where $\hat{f}$ is of low complexity, and the reduced model accurately predicts the behavior of (5.1) in response to changes in inputs and parameters.

The first step in obtaining such a reduced model is to separate the parameter dependence from the state dependence in the nonlinear function. Using, for instance, a polynomial fitting scheme or a Taylor series approximation in the parameters, we can approximate the nonlinear functions as follows

$$\begin{aligned} f(x, p_1, \ldots, p_\mu) &\approx \sum_{j=0}^{P-1} g_j(p_1, \ldots, p_\mu) f_j(x) \\ B(p_1, \ldots, p_\mu)u(t) &\approx \sum_{j=0}^{P-1} g_j(p_1, \ldots, p_\mu) B_j u(t), \end{aligned} \tag{5.2}$$

where $g_j(p_1, \ldots, p_\mu)$ are scalar functions of the parameters, $f_j(x)$ are vector-valued functions of the state, and B_j are constant input matrices. For example, a first order Taylor series expansion on $f(x, p)$ would yield

$$\begin{aligned} f_j(x) &= \frac{\partial f(x,p)}{\partial p_j}\Big|_{p_0} \\ f_0(x) &= f(x, p_0) - \sum_j p_{j0} \frac{\partial f(x,p)}{\partial p_j}\Big|_{p_0}. \end{aligned}$$

By introducing a new set of parameters $\tilde{p}_j = g_j(p_1, \ldots, p_\mu)$ it is finally possible to make the system affine in the new parameters

$$\frac{dx}{dt} = \sum_{j=0}^{P-1} \tilde{p}_j \left[f_j(x) + B_j u(t) \right] \tag{5.3}$$

while retaining the nonlinear dependence on the original parameters. In order to keep the equations concise we choose $f_0(x)$ and B_0 to be the terms with no parameter dependence and thus define $\tilde{p}_0 = 1$.

It is desirable to write the system as (5.3) because such form permits approximating each nonlinear function $f_j(x)$ as an affine function of the state without affecting the param-

eter dependence of the system:

$$\tilde{p}_j f_j(x) \approx \tilde{p}_j \left[f_j(x_i) + \frac{\partial f_j}{\partial x}(x - x_i) \right] = \tilde{p}_j \left[A_{ij} x + k_{ij} \right]$$

$$A_{ij} = \left. \frac{\partial f_j(x)}{\partial x} \right|_{x_i}, \qquad k_{ij} = f_j(x_i) - \left. \frac{\partial f_j(x)}{\partial x} \right|_{x_i} x_i,$$

where $A_{ij} \in \mathbb{R}^{N \times N}$ and $k_{ij} \in \mathbb{R}^N$.

This allows, as in the standard TPWL approach [78], approximation of the nonlinear functions $f_j(x)$ as a collection of local linearizations around different points x_i in the state space:

$$\frac{dx}{dt} = \sum_{i=0}^{\kappa-1} \sum_{j=0}^{P-1} w_i(x, X) \tilde{p}_j \left[A_{ij} x + k_{ij} + B_j u \right] \tag{5.4}$$

where $w_i(x, X)$ are weighting functions which vary dynamically with the state.

Now that the system matrices A_{ij} have no implicit parameter dependence, standard projection techniques can be applied to each of the linear systems in (5.4). For example, using a projection matrix $V \in \mathbb{R}^{N \times q}$, system (5.4) is reduced to

$$\frac{d\hat{x}}{dt} = \sum_{i=0}^{\kappa-1} \sum_{j=0}^{P-1} w_i(\hat{x}, \hat{X}) \tilde{p}_j \left[\hat{A}_{ij} \hat{x} + \hat{k}_{ij} + \hat{B}_j u \right] \tag{5.5}$$

$$y = \hat{C}^T \hat{x}$$

where $\hat{x} \in \mathbb{R}^q$, $\hat{A}_{ij} \in \mathbb{R}^{q \times q} = V^T A_{ij} V$, $\hat{k}_{ij} \in \mathbb{R}^q = V^T k_{ij}$, $\hat{B}_j \in \mathbb{R}^{q \times N_u} = V^T B_j$, $\hat{C} \in \mathbb{R}^{q \times N_y} = V^T C$, $x = V\hat{x}$, and $\hat{X} \in \mathbb{R}^{q \times \kappa} = [V^T x_0, ..., V^T x_{\kappa-1}]$, resulting in a reduced order model possessing a nonlinear parameter dependence similar to that of the original model.

In order to complete the procedure, two algorithms remain to be specified: how to choose linearization points x_i, and how to construct projection matrix V. These two methods will be discussed in detail in the following sections, and then combined to create the proposed Nonlinear Parameterized Model Order Reduction (NLPMOR) algorithm.

5.2.2 Selecting Linearization Points

In standard TPWL [19, 79, 78, 25, 26] linearization points are chosen along state trajectories generated by typical training inputs. Using a similar idea, additional trajectories can be created by training with system (5.3) at a set of points in the parameter space $\{\tilde{p}_j\}$. This additional training produces linear models in new state-space regions where variations in the parameter are likely to drive the state. As with training inputs, if we know a range of practical parameter values over which the system will be evaluated, we can restrict parameter training points to that set. Additionally, if we have information about the sensitivity of the system to each parameter, training should be performed in regions where the system is most sensitive to the parameter. Section 5.2.4 presents a method for approximating these sensitivities and using this information to select training points.

Computing the exact training trajectories requires simulation of the full nonlinear system, which may be prohibitively expensive. Alternatively, one could use 'approximate training trajectories'. In this case, rather than simulating the full nonlinear system, we simulate a linearzied model. It is assumed that this linearized model is accurate as long as the current simulated state stays in some neighborhood of the linearization state. Once the current simulated state leaves such neighborhood, a new linearized model is created at the current state, and the procedure continues on in this manner.

The additional trajectories created by parameter-space training increase the cost of constructing the model, but do not significantly affect the cost of simulating the ROM. Since the weighting functions in (5.5) are typically nonzero for just a few models at any particular time, only the closest models are considered for weighting, and a larger set of models does not significantly affect simulation time [95]. Thus, by holding the order of the reduced system fixed and adding additional models from new trajectories, the interpolation of the nonlinearity $f_j(x)$ in (5.3) can be improved without significantly increasing the simulation time.

5.2.3 Constructing the Projection Matrix

As in PMOR for linear systems [101, 22, 24], the columns of the projection matrix V can be chosen to span the subspace generated by the vectors from a multivariable Taylor series expansion about each parameter $\tilde{p}_j$ in (5.4). This is similar to the scheme used in section 3.3.1, except in this case the model is nonlinear. Therefore, the projection vectors are constructed using the vectors produced by a multivariable Taylor series expansion (with respect to the frequency and all of the parameters) of the transfer functions of each of the κ linearized models created during training. Constructing V in this manner ensures that the PROM will match moments of the transfer functions of each of the linearized systems with respect to both frequency and parameter values.

To briefly recap moment-matching, the input moments of the linear system $E\dot{x} = Ax + Bu$ are defined as the vectors

$$\tilde{B}, M\tilde{B}, M^2\tilde{B}, M^3\tilde{B}, \ldots \tag{5.6}$$

where $M = A^{-1}E$ and $\tilde{B} = A^{-1}B$. Similarly, the parameterized moment-matching input vectors for the parameterized linear system $E\dot{x} = A_0x + pA_1x + Bu$ are defined as

$$\tilde{B}, M_1\tilde{B}, M_2\tilde{B}, (M_1M_2 + M_2M_1)\tilde{B}, \ldots \tag{5.7}$$

where $M_1 = A_1^{-1}E$, $M_2 = A_1^{-1}A_2$, and $\tilde{B} = A_1^{-1}B$. See section 3.2.3 and section 3.3.1 respectively for more details.

It is important to note here that it would be possible to generate parameterized projection vectors using other projection-based PMOR methods (for example [35, 75, 46, 47, 68]). However, moment matching is suitable for this method because it is relatively cheap to compute a few moments from each linearization while it is being generated, and the parameterized moments allow us to fit the transfer functions more carefully around the frequencies and parameter values at which the training trajectories were created.

The training procedure produces κ linear models which capture the nonlinear effects of the original system. In addition to creating Krylov vectors from these κ models, it may

be beneficial to also create Krylov vectors at additional points along the training trajectories. This does not significantly increase the computational cost because when solving the nonlinear system with an implicit time integration scheme (e.g. Newton's method with Backward Euler) we produce anyway linearizations at every time step, hence the additional cost is merely a few system solves per additional Krylov vector.

One additional difference between the linear case in 5.6 and the nonlinear case is the constant vector k in (5.4) – an artifact of the state linearizations. This term can be treated as a second input vector b_2 with constant input $u_2(t) = 1$. Thus (5.4) becomes

$$\frac{dx}{dt} = \sum_{i=0}^{\kappa-1} \sum_{j=0}^{P-1} w_i(x, X)\tilde{p}_j \left[A_{ij}x(t) + b_{2ij}u_2(t) + B_j u(t)\right].$$

To account for this term, several Krylov vectors should also be generated as in section 3.3.1 for each linear model with k in place of B.

Matching moments about multiple expansion points for every linear model may quickly increase the number of columns in the projection matrix. As V becomes large, simulation of the reduced order model will become costly. One way to keep the size of the reduced system small is to perform a singular value decomposition (SVD) on the projection matrix [59, 102, 68]. The SVD is relatively inexpensive because the projection matrix is very tall, but also relatively "skinny". After SVD, only vectors corresponding to the largest q singular values are selected as columns for the new projection matrix V, resulting in a reduced system of small order q.

5.2.4 Selecting Parameter-Space Training Points

One possible method for selecting parameter values for training in the parameter-space is to predict whether or not a change in parameter value will cause the state to visit regions of the state-space that are not supported by the current projection operation subspace. Let us define $x(t, \tilde{p}_a, \omega_a) \in \mathbb{R}^N$ for $t \in [0, T]$ as a trajectory which solves (5.3) at $\tilde{p}_a = [\tilde{p}_{0a}, \tilde{p}_{1a}, \ldots, \tilde{p}_{P-1a}]^T$ driven by a sinusoidal input at frequency ω_a, and $\mathbb{V}$ as the subspace spanned by the columns of projection matrix V, which was constructed such that

$x(t, \tilde{p}_a, \omega_a) \in \mathbb{V}$. If it can be shown that $x(t, \tilde{p}_b, \omega_a) \in \mathbb{V}$ for some $\tilde{p}_b = \tilde{p}_a + \Delta\tilde{p}$ without computing $x(t, \tilde{p}_b, \omega_a)$, then there is no need to train at $\tilde{p}_b$ to generate more projection vectors.

The solution $x(t, \tilde{p}_b, \omega_a)$ can be approximated with a first order Taylor series expansion in $\tilde{p}$ as

$$x(t, \tilde{p}_b, \omega_a) \approx x(t, \tilde{p}_a, \omega_a) + \frac{\partial x}{\partial \tilde{p}}\Big|_{\tilde{p}_a} \Delta\tilde{p}. \tag{5.8}$$

where $\Delta\tilde{p} = [\Delta\tilde{p}_0, \ldots, \Delta\tilde{p}_{P-1}]^T \in \mathbb{R}^P$, and $\frac{\partial x}{\partial \tilde{p}} \in \mathbb{R}^{N \times P}$. If $\frac{\partial x}{\partial \tilde{p}} \in \mathbb{V}$, then $x(t, \tilde{p}_b, \omega_a) \in \mathbb{V}$ because $\mathbb{V}$ is a linear subspace, so linear combinations of elements in $\mathbb{V}$ are also in $\mathbb{V}$.

To compute $\frac{\partial x}{\partial \tilde{p}}$, let us first denote $x_k = x(t_k, \tilde{p}_a, \omega_a)$ for $1 \leq k \leq \tau$ as a sample of trajectory $x(t, \tilde{p}_a, \omega_a)$ at t_k, and then define $\bar{x} \in \mathbb{R}^{N\tau} = [x_1^T, \ldots, x_\tau^T]^T$ as a stack of the τ trajectory samples into one long vector. Since x solves (5.3), this new variable $\bar{x}$ approximately solves the system

$$\bar{D}\bar{x} = \sum_{j=0}^{P-1} \tilde{p}_j \left[\bar{f}_j(\bar{x}) + \bar{B}_j \right]$$

where $\bar{D} \in \mathbb{R}^{N\tau \times N\tau}$ is a finite difference time-derivative operator, $\bar{f}_j : \mathbb{R}^{N\tau} \mapsto \mathbb{R}^{N\tau} = [f_j^T(x_1), \ldots, f_j^T(x_\tau)]^T$, and $\bar{B}_j \in \mathbb{R}^{N\tau} = [(Bu(t_1))^T, \ldots, (Bu(t_\tau))^T]^T$. Differentiating this system with respect to each of the parameters yields

$$\bar{D}\frac{\partial \bar{x}}{\partial \tilde{p}} = \tilde{\bar{f}}(\bar{x}) + \sum_{j=0}^{P-1} \tilde{p}_j \frac{\partial \bar{f}_j}{\partial x}\frac{\partial \bar{x}}{\partial \tilde{p}}$$

where $\frac{\partial \bar{f}}{\partial \bar{x}} \in \mathbb{R}^{N\tau \times N\tau}$, $\frac{\partial \bar{x}}{\partial \tilde{p}} \in \mathbb{R}^{N\tau \times P}$ and $\tilde{\bar{f}} \in \mathbb{R}^{N\tau \times P}$ is

$$\tilde{\bar{f}}(\bar{x}) = \left[\left(\bar{f}_0(\bar{x}) + \bar{B}_0 \right), \ldots, \left(\bar{f}_{P-1}(\bar{x}) + \bar{B}_{P-1} \right) \right].$$

This can be rearranged into the linear system

$$\left[\bar{D} - \sum_{j=0}^{P-1} \tilde{p}_j \frac{\partial \bar{f}_j}{\partial \bar{x}} \right] \frac{\partial \bar{x}}{\partial \tilde{p}} = \tilde{\bar{f}}(\bar{x}) \tag{5.9}$$

whose solution is a narrow matrix $\frac{\partial \bar{x}}{\partial \tilde{p}}$ such that

$$\frac{\partial \bar{x}}{\partial \tilde{p}} = \begin{bmatrix} \frac{\partial x}{\partial \tilde{p}_0}\Big|_{t=t_1} & \cdots & \frac{\partial x}{\partial \tilde{p}_{P-1}}\Big|_{t=t_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial x}{\partial \tilde{p}_0}\Big|_{t=t_T} & \cdots & \frac{\partial x}{\partial \tilde{p}_{P-1}}\Big|_{t=t_T} \end{bmatrix}$$

System (5.9) is large, but $\frac{\partial \bar{f}}{\partial \bar{x}}$ is very sparse and $\bar{\tilde{f}}$ is narrow and sparse. Assembling the system requires no extra work because both the Jacobians and the function evaluations in (5.9) were already computed at every time step during the training process.

If $\frac{\partial x}{\partial \tilde{p}}$ is well approximated by vectors in $\mathbb{V}$, then its largest singular vectors, defined to be $\frac{\tilde{\partial x}}{\partial p}$, are orthogonal to the nullspace of V^T, defined as $\mathcal{N}(V^T)$. That is,

$$\left|\left| \left(\frac{\tilde{\partial x}}{\partial p} \right)^T \mathcal{N}(V^T) \right|\right| \leq \epsilon$$

where ϵ is a small tolerance.

Note that even if solution $\frac{\tilde{\partial x}}{\partial p}$ is in $\mathbb{V}$, it may still be beneficial to add new linearization points from the trajectory $x(t, \tilde{p}_b, \omega_a)$. If each linearized model is assumed to be accurate in some $\delta - ball$ around its linearization point, then no models are needed if

$$||x(t, \tilde{p}_a, \omega_a) - x(t, \tilde{p}_b, \omega_a)|| < \delta$$

for all t. From (5.8), this is equivalent to

$$\left|\left| \frac{\partial x(t)}{\partial p} \right|\right| \leq \frac{\delta}{||\Delta p||}$$

for all t.

Thus one could perform the above checks while the trajectory at $\tilde{p}_a$ is being created and would know by the end of the trajectory whether or not it is necessary to train at a nearby parameter-space point $\tilde{p}_b$.

	Training at Single Point in Parameter Space	Training at Multiple Points in Parameter Space
Exact Trajectories	T_{es}	T_{em}
Approximate Trajectories	T_{as}	T_{am}

Table 5.1: The 4 options for selecting linearization points from training trajectories

5.2.5 Proposed NLPMOR Algorithm

An algorithm for NLPMOR is constructed by defining both a linearization scheme (i.e. a method to choose linearization points for linear models) and a projection scheme (i.e. a method to construct the projection matrix V). By combining the parameterization options in sections 5.2.2 and 5.2.3, we obtain four different schemes for training, presented in Table 5.1, and four different schemes for constructing V, presented in Table 5.2. A generic NLPMOR algorithm which incorporates each of these options is presented in Algorithm 4.

In order to select a linearization scheme, two decisions need to be made: whether to exactly compute the trajectories or to merely approximate the trajectories, and whether or not to train in the parameter space. If exact trajectories are used, the nonlinear system and the current linear model are solved to obtain x_t and x_i, the states of the nonlinear system and linearized model respectively at time t. Letting $\Delta = ||x_t - x_i||$ be the distance between the solution of the nonlinear system and the solution of the linearized system, a new model is created whenever $\Delta > \delta$, where δ is some preset tolerance. A linearization of the original nonlinear system consists of a pair $\{A_{tj}, k_{tj}\}$ as in (5.4). If approximate training trajectories are used, rather than comparing the linear system solution x_i to the nonlinear system solution, we compare x_i to the previous linearization points x_{L_j}. By setting $\Delta = \min_j ||x_i - x_{L_j}||$, linearizations are created when the state x_i strays too far from the closest of all the precalculated linearization points x_{L_j}, i.e. $\Delta > \delta$.

In order to select a projection scheme we need to decide which linear models the Krylov vectors are generated from, and which Taylor expansion is used to compute the vectors. When the system is trained with exact trajectories, linear models are available at every time step and it is cheap to create several Krylov vectors at each step (achieved in Algorithm 4

	Krylov Vectors From All States	Krylov Vectors Only From Linearized Systems
MOR V Moment Matching in V	V_{ma}	V_{ml}
PMOR V Moment Matching in V	V_{pa}	V_{pl}

Table 5.2: The 4 options for constructing the projection matrix

by setting $KrAll = 1$). If approximate trajectories are used for training, or if the cost of creating Krylov vectors at every time step is prohibitive, then Krylov vectors are computed only from the linear models (obtained in Algorithm 4 by setting $KrAll = 0$).

Finally, Krylov vectors could be created either with a single variable Taylor series expansion about the Laplace variable s, referred to in Algorithm 4 as $MORV$, or with a multivariable Taylor series expansion about s and all of the parameters, referred to in Algorithm 4 as $PMORV$.

The notation in Tables 5.1 and 5.2 will be used in this paper to identify different kinds of model reduction algorithms. For instance, when we write a $T_{em}V_{pl}$ PROM we mean that the reduced model is created by training with **e**xact trajectories at **m**ultiple points in the parameter space, and is reduced with a **P**MOR projection matrix with vectors taken only from the **l**inear models created. As another example of our notation, when in Section 5.4 we compare $T_{xx}V_{mx}$ and $T_{xx}V_{px}$ models we mean that we intend to examine only the effects of MOR moment matching versus PMOR moment matching.

5.2.6 Algorithm Costs

Since computing each Krylov vector requires one system solve, the cost of constructing the projection matrix can be measured in number of system solves. Such costs are summarized in Table 5.3. For a projection matrix created by generating m MORV Krylov vectors from the κ linear models, the cost of constructing the projection matrix is $O(\kappa m)$. If instead, PMORV vectors are chosen and the system has P parameters, then the cost of constructing V becomes $O(\kappa P^m)$. When Krylov vectors are generated from every trajectory step, the costs becomes $O(Tm)$ and $O(TP^m)$ respectively, where T is the total number of time steps

	Krylov Vectors From All States	Krylov Vectors Only From Linearized Systems
MOR V Moment Matching in V	$O(Tm)$	$O(\kappa m)$
PMOR V Moment Matching in V	$O(TP^m)$	$O(\kappa P^m)$

Table 5.3: Costs of constructing the projection matrix using the 4 available options, measured in system solves per trajectory

	Training at Single Point in Parameter Space	Training at Multiple Points in Parameter Space
Exact Trajectories	$O(\gamma T)$	$O(\gamma T r^P)$
Approximate Trajectories	$O(T)$	$O(T r^P)$

Table 5.4: Costs of training the system using the 4 available options, measured in system solves per input

in a trajectory.

The exact training trajectories are created by solving the large nonlinear system at each time step. If each trajectory contains T points and each nonlinear solve requires γ Newton iterations, a single trajectory will cost $O(\gamma T)$ system solves. For the approximate trajectory algorithms, the cost of a single trajectory is reduced to $O(T)$ solves, as shown in Table 5.4. Finally, for a system with P parameters and r training values for each parameter, a single input will generate r^P different training trajectories.

5.3 Examples

Three example systems were chosen to help illustrate the advantages of NLPMOR. All three examples are physical systems which contain strong nonlinearities that are distributed throughout the devices, and possess dependence on several geometrical parameters. For each example a derivation of the original system model is presented, followed by results from our different algorithms.

5.3.1 Diode Transmission Line

The first example considered is a diode transmission line, which was used in the original TPWL papers [78, 98]. This allows for some relative accuracy comparisons between our new method and a well established result in literature. The transmission line, shown in Figure 5-1, is a nonlinear analog circuit containing a chain of strongly nonlinear diodes, resistors and capacitors.

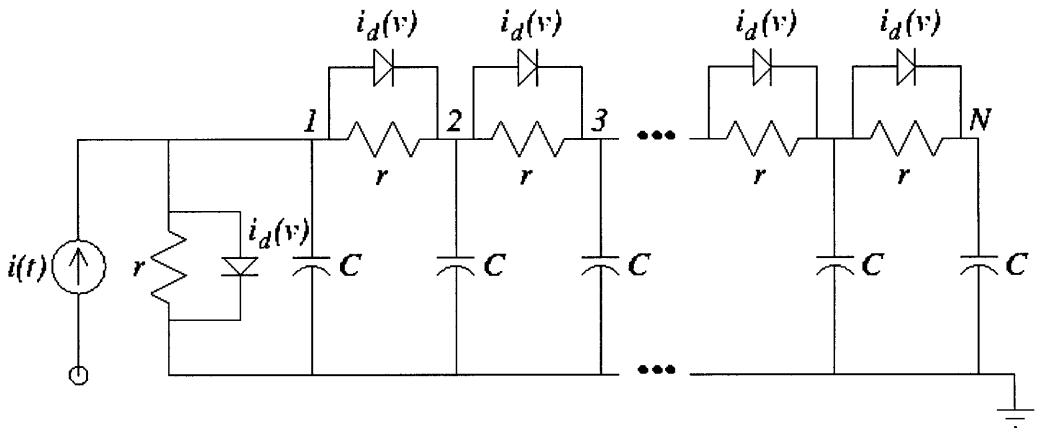


Figure 5-1: A nonlinear transmission line circuit containing diodes [78, 98].

We choose the nodal voltages as the system state and derived the system equations using Kirchoff's current law and nodal analysis. An equation for interior node j has the form

$$C\frac{dx_j}{dt} = \frac{x_{j-1} - 2x_j + x_{j+1}}{r} + I_d[e^{\frac{1}{v_T}(x_{j-1}-x_j)} - e^{\frac{1}{v_T}(x_j - x_{j+1})}] \tag{5.10}$$

leading to a state space system of the form

$$E\frac{dx}{dt} = -\frac{1}{r}Q^TQx - I_dQ^Td(x, v_T) + bu(t). \tag{5.11}$$

Here $Q \in \mathbb{R}^{N\times N}$ is the adjacency matrix for the resistor and diode network. Matrix $E \in \mathbb{R}^{N\times N}$ is the capacitance matrix. Vector $d(x, v_T) = -Q^T\phi(x, v_T)$ where $\phi(x, v_T) : \mathbb{R} \times \mathbb{R}^N \mapsto \mathbb{R}^N$. Its j^{th} row is $\phi_j(x, v_T) = e^{\frac{1}{v_T}q_j^Tx} - 1$ where q_j is the j^{th} column of Q. Vector $b \in \mathbb{R}^N$ relates the state equations to the input which is an ideal current source $u(t) = i(t)$. All resistors have value 1Ω and all capacitors are $10pF$. The constitutive relation for the

diodes is $\phi(v) = I_d(e^{\frac{1}{v_T}v} - 1)$, where v_T is the threshold voltage, and v is the voltage across the device. Values of $I_d = 0.1nA$ and $v_t = 25mV$ were used as nominal values. Three parameters were considered for the diode transmission line: the resistor values r, the diode threshold voltage v_T, and the diode saturation current I_d. The situation is simplified if the parameters are defined as $p_G = \frac{1}{r}$, $p_V = \frac{1}{v_T}$, and $p_I = I_d$. Since (5.11) possesses nonlinear dependence on p_V, the system must first be expanded in powers of p_V. We chose to use a second order expansion about the nominal value $p_V = \frac{1}{25\text{mV}} = 40\text{V}^{-1}$

$$E\frac{dx}{dt} = p_G G x + p_I d_0(x) + p_I p_V d_1(x) + p_I p_V^2 d_2(x) + bu(t), \tag{5.12}$$

where

$$\begin{aligned}
G &= -Q^T Q \\
d_0(x) &= -Q^T \left[d(x, v_{T_0}) - \frac{1}{v_{T_0}} \frac{\partial d(x, v_{T_0})}{\partial(\frac{1}{v_{T_0}})} + \frac{1}{2v_{T_0}^2} \frac{\partial^2 d(x, v_{T_0})}{\partial(\frac{1}{v_{T_0}})^2} \right] \\
d_1(x) &= -Q^T \left[\frac{\partial d(x, v_{T_0})}{\partial(\frac{1}{v_{T_0}})} - \frac{1}{v_{T_0}} \frac{\partial^2 d(x, v_{T_0})}{\partial(\frac{1}{v_{T_0}})^2} \right] \\
d_2(x) &= -Q^T \frac{1}{2} \frac{\partial^2 d(x, v_{T0})}{\partial(\frac{1}{v_{T_0}})^2}.
\end{aligned}$$

Note that the system is still nonlinear in the state. To test our reduction algorithms, we created a reduced model parameterized in p_G. Figure 5-2 compares the simulation output of the full nonlinear system with that of the PROM over a large range of parameter values which vary from the nominal value by as much as -30% and $+150\%$.

5.3.2 Micromachined Switch

The second example is a micromachined switch [78, 98]. The switch consists of a polysilicon fixed-fixed beam suspended over a polysilicon pad on a silicon substrate as shown in Fig. 5-3. When a voltage is applied between the beam and the substrate, the electrostatic force generated pulls the beam down towards the pad. If the force is large enough, the beam will come into contact with the pad closing the circuit. In addition to being used as a switch,

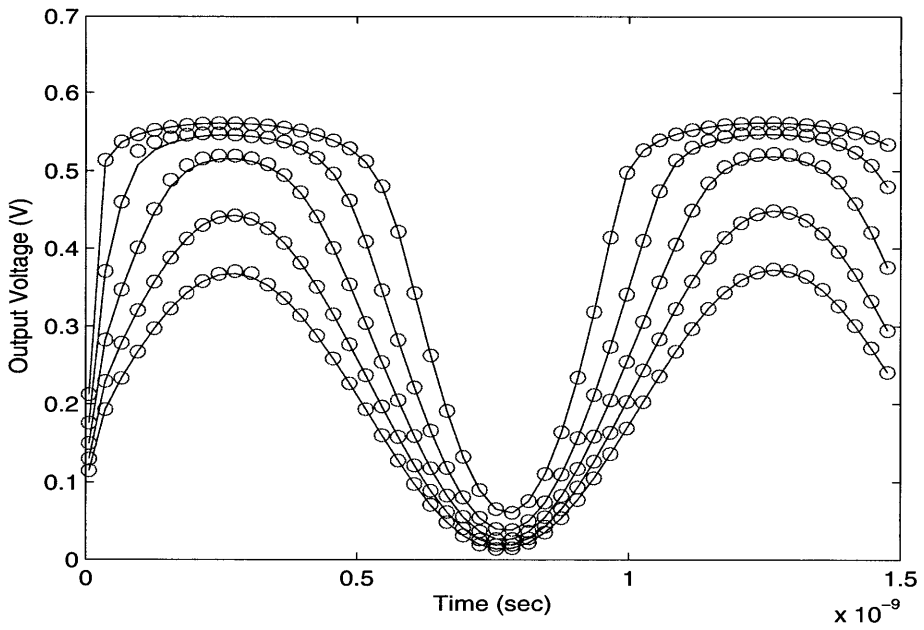


Figure 5-2: A model created for the diode transmission line with original size $N = 200$, parameterized in $p_G = \frac{1}{r}$ by training at $p_{G0} = \{0.75p_{G0}, p_{G0}, 2p_{G0}\}$ with $p_{G0} = 1$. The PROM has size $q = 10$ and was simulated at a range of p_G values in the interval $[0.7p_{G0}, 2.5p_{G0}]$, resulting in a speedup of about $10\times$.

this device can be used as a pressure sensor due to its extreme sensitivity to surrounding atmospheric conditions. The unknowns of interest in this system are the deflection of the beam, $z(x,t)$, and the air pressure between the beam and substrate, $P(x,y,t)$. The system of equations is assembled by discretizing the coupled 1D Euler's Beam Equation (5.13) and the 2D Reynold's squeeze film damping equation (5.14), taken from [78]. A finite difference scheme was used for the discretization, using m points for the length and n points for the width, and since the length of the beam is much greater than the width, the vertical deflection is assumed to be uniform across the width and only pressure was discretized in the width

$$\hat{E}I_0 h^3 w \frac{\partial^4 z}{\partial x^4} - S_0 hw \frac{\partial^2 z}{\partial x^2} = F_{elec} + \int_0^w (P - P_a)dy - \rho_0 hw \frac{\partial^2 z}{\partial t^2} \tag{5.13}$$

$$\nabla \cdot \left((1+6K)\, z^3 P \nabla P\right) = 12\mu \frac{\partial (Pz)}{\partial t}. \tag{5.14}$$

Here, $F_{elec} = -\frac{\epsilon_0 w v^2}{wu^2}$ is the electrostatic force across the plates resulting from the applied voltage v, while v^2 is the input to the system. The beam is $2.2\mu m$ above the substrate ($z_0 =$

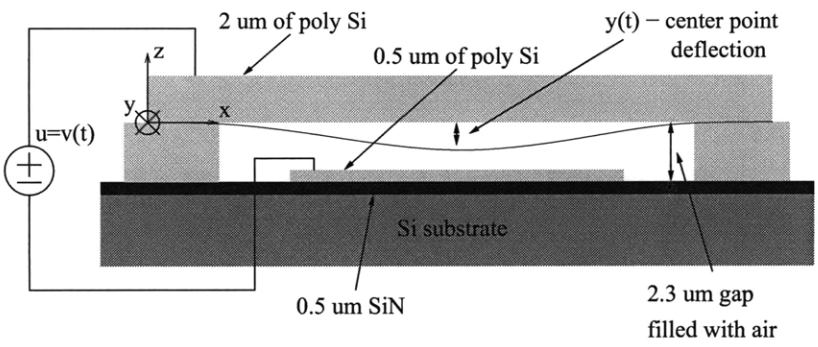


Figure 5-3: The MEM switch is a polysilicon beam fixed at both ends and suspended over a semiconducting pad and substrate [78, 98].

$2.2\mu m$), $610\mu m$ in length, and has a width of $40\mu m$. The other constants are permittivity of free space $\epsilon_0 = 8.854 \times 10^{-6}\frac{F}{m}$, permeability $\mu = 1.82 \times 10^{-5}\frac{kg}{m\dot{s}}$, moment of inertia $I_0 = 1/12$, Young's modulus $\hat{E} = 149GPa$, Knudsen number $K = \frac{\lambda}{z_0}$, $\lambda = 0.064$, stress coefficient $S_0 = -3.7$, and density $\rho_0 = 2300\frac{kg}{m^3}$. The above equations can be separated into three partial differential equations

$$\begin{aligned}
\frac{\partial z}{\partial t} &= \frac{\partial^3 z}{\partial t^3}\frac{1}{3z^2} \\
\frac{\partial^4 z}{\partial t^4} &= \left(\frac{\partial^3 z}{\partial t^3}\right)^2 \frac{2}{3z^3} - \frac{3\epsilon_0}{2\rho_0 h}v^2 + \frac{3z^2}{\rho_0 hw}S_0 hw\frac{\partial^2 z}{\partial x^2} \\
&\quad + \frac{3z^2}{\rho_0 hw}\left[\int_0^w (P - P_a)dy - EIh^3 w\frac{\partial^4 z}{\partial x^4}\right] \\
\frac{\partial P}{\partial t} &= -\frac{\partial^3 z}{\partial t^3}\frac{P}{3z^3} + \frac{1}{12\mu z}\nabla\left(\left(1 + 6\frac{\lambda}{z}\right)z^3 P\nabla P\right).
\end{aligned}$$

We choose the state-space variables to be $x_1 \in \mathbb{R}^m = z$, $x_2 \in \mathbb{R}^m = \frac{\partial u^3}{\partial t}$, and $x_3 \in \mathbb{R}^{mn} = P$, and the parameters to be Young's modulus $p_E = E$ and stress coefficient $p_S = S$. Rearranging the discretized system equations to obtain linearity in each parameter results in the system

$$\begin{aligned}
\frac{\partial x_1}{\partial t} &= f_{1,0}(x_1, x_2) \\
\frac{\partial x_2}{\partial t} &= f_{2,0}(x_1, x_2, x_3) + p_S f_{2,1}(x_1, x_2) + p_E f_{2,2}(x_1) + bu(t) \\
\frac{\partial x_3}{\partial t} &= f_{3,0}(x_1, x_2, x_3),
\end{aligned}$$

where the total system has order $N = m(n+2)$, $f_{1,0} \in \mathbb{R}^m$, $f_{2,0}, f_{2,1}, f_{2,2} \in \mathbb{R}^m$, and $f_{3,0} \in \mathbb{R}^{mn}$. A detailed description of these functions can be found in [77]. The beam is fixed at both ends and initially in equilibrium, so the applied boundary conditions are

$$z(x,0) = z_0, \qquad P(x,y,0) = P_a, \qquad z(0,t) = z(l,t) = z_0.$$

Other constraints enforced are

$$\frac{\partial P(0,y,t)}{\partial x} = \frac{\partial P(l,y,t)}{\partial x} = 0, \qquad P(x,0,t) = P(x,w,t) = P_a$$

where the initial height and pressure are $z_0 = 2.3\mu m$ and $P_a = 1.103 \times 10^5$Pa. Typical inputs for this system are sinusoids, $u(t) = (v\cos(\omega t))^2$, with $\omega = \frac{10\pi}{30} MHz$ and $v = 7$, or a step input, $u(t) = v^2$ for $t > 0$ with $v = 7$. The system output is the deflection of the beam center point.

For this example a reduced model parameterized in p_E and p_S was created by training with sinusoidal inputs. The model was then simulated at nine different sets of parameter values on an evenly spaced grid with each parameter varying up to $\pm 40\%$ from the nominal values. The outputs from the simulation along with the output of the full nonlinear system are shown in Figure 5-4.

5.3.3 Pulse-Narrowing Transmission Line

The final example considered is a nonlinear transmission line used for signal shaping. One example of such a line, shown in Figure 5-5, contains distributed nonlinear capacitors. The resulting wave equation for this transmission line contains a nonlinear term which sharpens the peaks in a wave travelling down the line. Hence these devices may be useful in pulse narrowing applications. A thorough analysis of this line can be found in [2]. The nonlinearity arises from the voltage dependence of the capacitors, $C_n = C(V_n) \approx C_0(1 - b_c V_n)$. Setting the system state to the node voltages and branch currents, system equations can be derived using Kirchoff's current law and nodal analysis. The input is an ideal voltage source $u(t) = V_s(t)$, and the output is the voltage at some node m along the

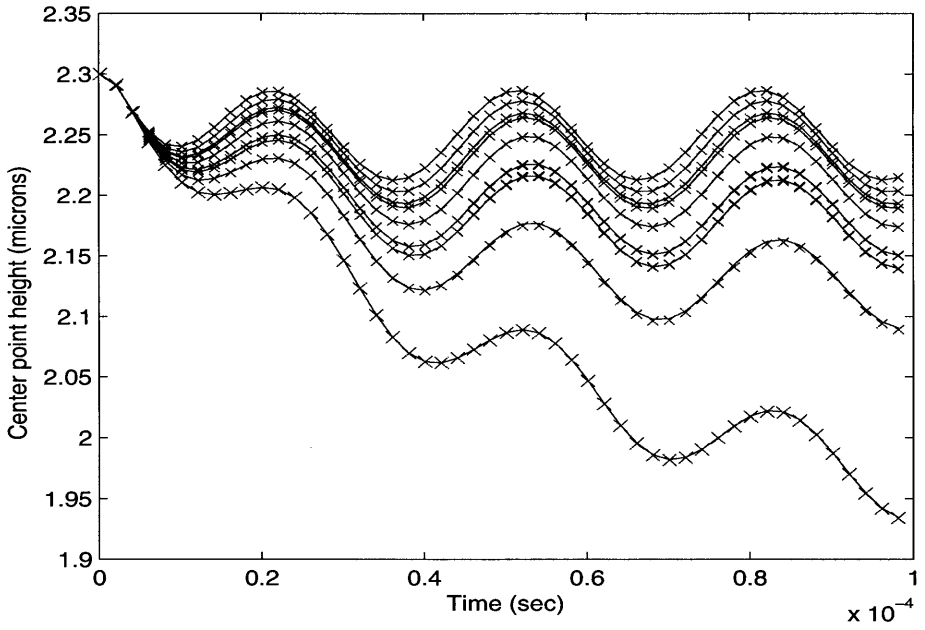


Figure 5-4: Output of a micromachined switch model, parameterized in p_E and p_S, simulated at nine different sets of parameter values on an evenly spaced grid where $p_E \in [0.6p_{E0}, 1.4p_{E0}]$ and $p_S \in [0.6p_{S0}, 1.4p_{S0}]$. The solid lines represent the original model with order $N = 144$, the crosses represent the reduced model of order $q = 20$, the resulting speedup in simulation was about $15\times$, and the nominal parameter values are $[p_{E0}, p_{S0}] = [1.49 \times 10^5, -3.7]$.

line, $y(t) = V_m(t)$. Using this formulation, the system equations for an interior node n would be of the form

$$C_n(V_n)\frac{dV_n}{dt} = I_{n-1} - I_n$$
$$L_n\frac{dI_n}{dt} = V_n - V_{n+1}$$

leading to the state space model

$$\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \frac{1}{C_0} f_V(x,z) \\ \frac{1}{L} f_I(x,z) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} b \end{bmatrix} u(t)$$

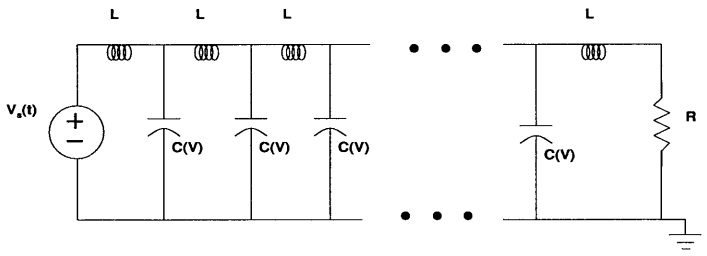


Figure 5-5: A pulse narrowing transmission line circuit containing nonlinear capacitors [2].

where the n^{th} equations of f_V and f_I are

$$f_{Vn}(x,z) = \frac{z_{n-1} - z_n}{1 - b_c x_n} \tag{5.15}$$

$$f_{In}(x,z) = x_n - x_{n+1}. \tag{5.16}$$

Here b is the vector of voltage source inputs. Typical capacitor and inductor values are 100 picoFarads and 100 picoHenries respectively. Parameters of interest for the pulse narrowing transmission line are the inductor values, the capacitor values, and b_c, a parameter which adjusts the nonlinearity of the line. These three parameters all affect the shaping of the wave as it travels down the line. For this example, PROMs were created by training with a sinusoidal input with $u(t) = v\sin(\omega t)$ at frequency $5GHz$. PMOR moment matching generated moments about parameter expansion points equal to the parameter values used in training. To test this example, we parameterized the system in $p_C = \frac{1}{C}$ and $p_L = \frac{1}{L}$, resulting in a system of the form

$$\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = p_L \left(\begin{bmatrix} 0 \\ f_I(x,z) \end{bmatrix} + \begin{bmatrix} 0 \\ b_1 \end{bmatrix} u(t) \right) + p_C \begin{bmatrix} f_V(x,z) \\ 0 \end{bmatrix}.$$

Figure 5-6 compares the output of the full system and a PROM for the pulse-narrowing transmission line simulated at five different parameter values varying as much as -90%

and $+100\%$.

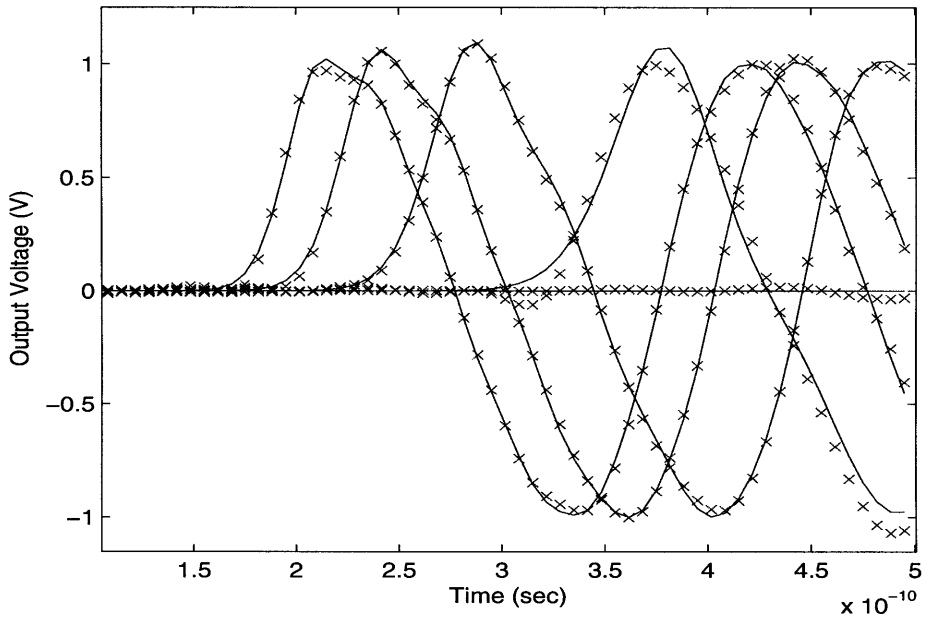


Figure 5-6: Output from a model of the pulse narrowing transmission line simulated at five different values of $p_L = \frac{1}{L}$ on the interval $[0.1p_{L0}, 2p_{L0}]$, where $p_{L0} = 10^{11}$. The model was reduced from large order $N = 200$ to reduced order $q = 50$ which resulted in a speedup of $\sim 5\times$.

5.4 Comparison of Algorithms

In this section we examine the accuracy of models created with the different linearization and projection schemes from Tables 5.1 and 5.2. Specifically considered is how the different linearization and projection options affect the accuracy of the PROM in the parameter space. We also wish to determine whether or not the parameter-space accuracy of the PROM is limited by the original linearization of the nonlinear system with respect to the parameters.

5.4.1 Training in the Parameter Space

The effects of training at different points in the parameter space (described in section 5.2.2) can be seen by comparing $T_{xm}V_{xx}$ models with $T_{xs}V_{xx}$ models. As explained at the end

of Section 5.2.5 we use the notations from Tables 5.1 and 5.2 to identify different kinds of models. Trajectories created with different parameter values will likely evolve in different regions of the state space, thus resulting in different collections of linear models. The first

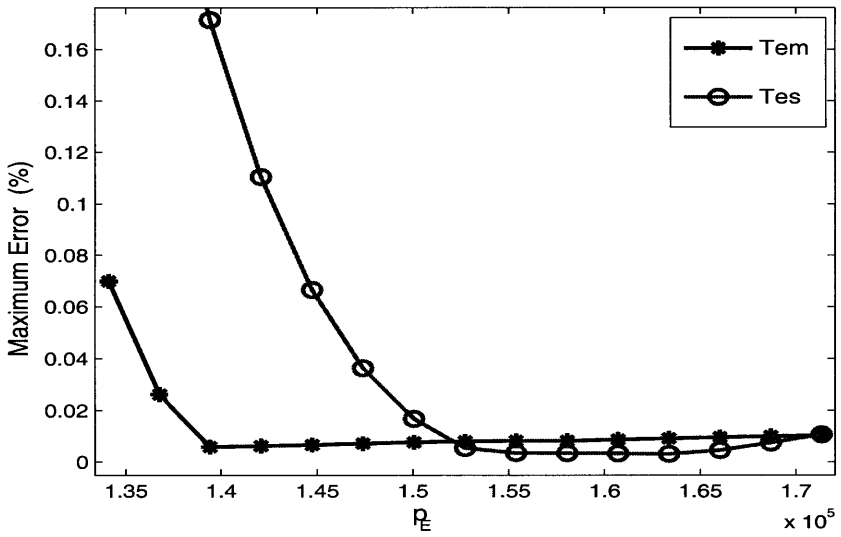


Figure 5-7: $T_{em}V_{pl}$ and $T_{es}V_{pl}$ models of the micromachined switch example parameterized in p_E and simulated over a range of parameter values. Each model was reduced from large order $N = 150$ to reduced order $q = 30$.

test compares $T_{em}V_{pl}$ and $T_{es}V_{pl}$ models of the micromachined switch. Considering p_E as the parameter, one model was created by training at $p_E = p_{E0} = 149GPa$, and the other by training at $p_E = [0.95p_{E0}, 1.05p_{E0}]$. The projection matrices for both models were created by matching parameter moments at E_0 and frequency moments at the input frequency $f = 1GHz$. The models were simulated at a set of parameter values in the range $[0.9p_{E0}, 1.1p_{E0}]$. Figure 5-7 compares the maximum percent error for each model, defined as

$$\max_t \left(\frac{|y(t) - \hat{y}(t)|}{|y(t)|} \right) \times 100 \tag{5.17}$$

A similar comparison is made in Figure 5-8 with $T_{es}V_{ml}$ and $T_{em}V_{ml}$ models of the diode transmission line parameterized in p_I. In this figure, the error plotted is the norm of $e(t)$, where

$$e(t) = |y(t) - \hat{y}(t)|. \tag{5.18}$$

These models were constructed by training at $p_I = p_{I_0} = 0.1nA$ for T_{es} and $p_I =$

$[0.5p_{I_0}, 1.3p_{I_0}]$ for T_{em}.

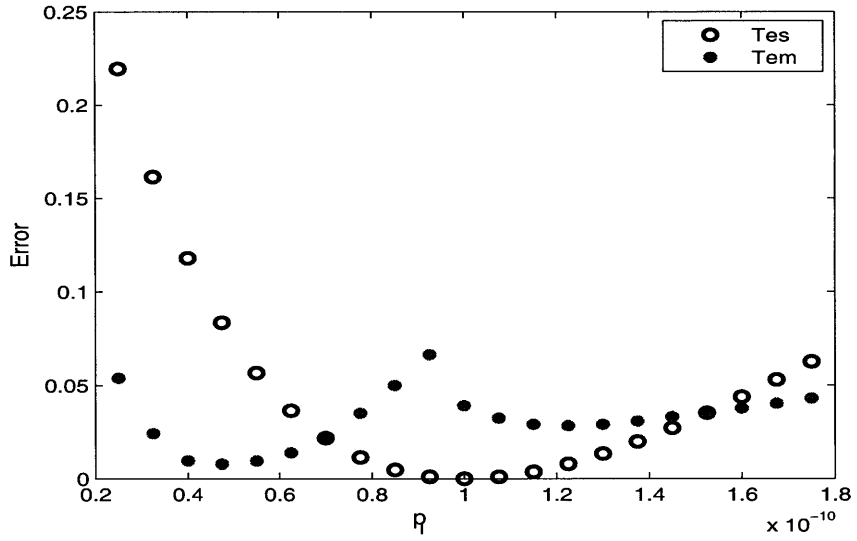


Figure 5-8: Norm of the error, as defined in (5.18), over time for a $T_{es}V_{ml}$ model trained at $p_I = 10^{-10}A$, and a $T_{em}V_{ml}$ model trained at $[0.5p_{I_0}, 1.3p_{I_0}]$. The system was reduced from original order $N = 100$ to reduced order $q = 50$.

Both Figures 5-7 and 5-8 show that the greatest accuracy occurs close to the training parameter value for the model created by training at a single point. However both figures also show that the model created by training at multiple parameter-space points is more accurate in a larger region around the training values.

5.4.2 Parameterizing the Projection Matrix

The benefits of parameterizing the projection matrix via PMOR moment matching, as in section 5.2.3, can be examined by comparing $T_{xx}V_{mx}$ models with $T_{xx}V_{px}$ models.

Figure 5-9 compares the total simulation error at different parameter values for $T_{es}V_{ml}$ and $T_{es}V_{pl}$ models of the diode transmission line parameterized in p_I. As with parameter-space training, this figure suggests that a V_{ml} model is more accurate close to the nominal parameter value, and a V_{pl} model is less accurate at the nominal value, but more accurate over a larger range of parameter values.

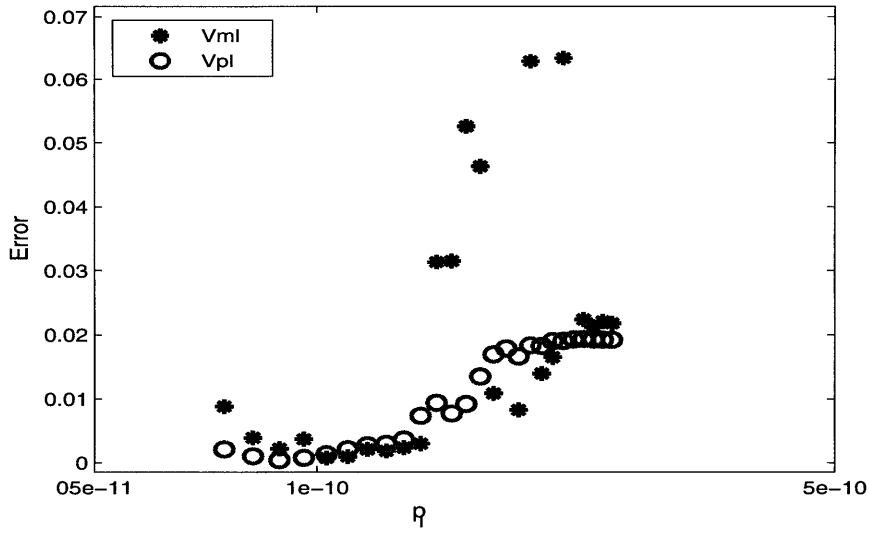


Figure 5-9: Norm of the error, as defined in (5.18), for two reduced order models of the diode transmission line parameterized in p_I and simulated at a range of parameter values using sinusoidal inputs. The models are $T_{es}V_{ml}$ and $T_{es}V_{pl}$ models and were reduced from large order $N = 100$ to reduced order $q = 40$.

5.4.3 Krylov Vectors from Extra Models

To determine whether or not the linear models created during training produce Krylov vectors which span a near-optimal reduced space, we compare $T_{xx}V_{xl}$ models with $T_{xx}V_{xp}$ models. Both PROMs contain the same number of linear models, κ, and have the same reduced order, q. Figure 5-10 compares the output from these two models. The results, however, are system-dependent.

We also considered the diode transmission line parameterized in p_R, and in this case there is no discernable advantage to a V_{xa} model. In general we suspect that a V_{xa} model will not be less accurate than a V_{xl} model. Before the SVD in step 32 of Algorithm 1, the V_{xa} projection matrix contains all of the columns in the V_{xl} projection matrix. Therefore, from a practical point of view, after SVD the V_{xa} projection matrix will correspond to a subspace at least approximately as good as the projection matrix from the V_{xl} model. However, theoretically it is important to note here that using a projection matrix constructed using an SVD in this manner can no longer guarantee an absolutely exact match of transfer function moments between the original linearized models and the reduced models.

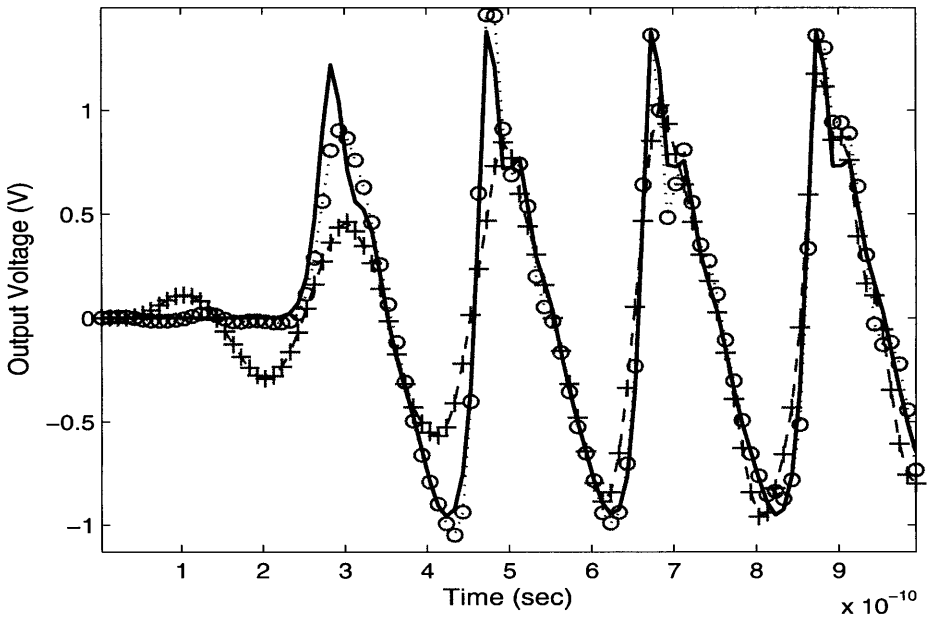


Figure 5-10: Two models of the pulse narrowing transmission line parameterized in p_L. The circles use vectors from every point of the trajectories while the crosses use vectors only from the $k = 181$ linear models. In both cases an SVD was used on V and both models were projected from large order N=200 down to the same reduced order $q = 50$.

5.4.4 Approximate Training Trajectories

Generating exact training trajectories can be often very expensive. Alternatively, one could instead use approximate training trajectories. In this section we compare the two approaches examining $T_{ax}V_{xx}$ models and $T_{ex}V_{xx}$ models.

Using the micromachined switch example parameterized in p_E, a $T_{es}V_{ml}$ model and a $T_{as}V_{ml}$ model were created. The two models were then simulated at three different parameter values close to the training values. Figure 5-11 compares the percent error of the PROM output for the two models.

Although the model created with exact trajectories is more accurate, both models still produce outputs with a maximum error smaller than 0.5%.

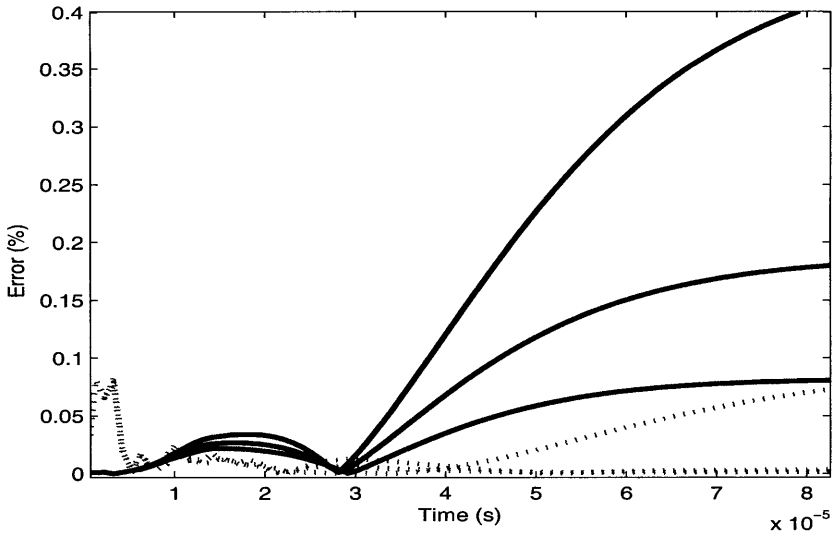


Figure 5-11: Percent error in the output of reduced models of micromachined switch. The solid curves correspond to models constructed with approximate training trajectories (T_{ax} models) and the dashed curves correspond to models constructed with exact training trajectories (T_{ex} models). Both models were then simulated at three different parameter values.

5.4.5 Effects of Linearizing in Parameters

Lastly, we consider the effects of linearizing the original nonlinear system (5.1) with respect to the parameters (5.3). An important question to ask is whether the dominant factor in determining the accuracy of the PROM is a result of projecting the system into a low-order subspace, or a result of this linearization in the parameters.

To investigate this we considered the diode transmission line with parameter p_V. Since the original system was nonlinear in p_V, (5.11) was expanded to second order about some nominal value p_{V_0} to obtain system (5.12) which is linear in powers of p_V but still nonlinear in the state x. A model was created using sinusoids as training inputs and a nominal parameter value $p_{V0} = 40$ for expansion and training. Figure 5-12 compares the output error at different parameter values between the original system (5.1), the model expanded in powers of the parameters (5.3), and the PROM (5.5). In this case we define the error $e_m(p)$ as

$$e_m(p) = \left(\frac{\max_t |y(t) - y_0(t)|}{\max_t |y_0(t)|} \right) \times 100 \tag{5.19}$$

where $y(t)$ is the output of one system at parameter value p and $y_0(t)$ is the output of the

other system at parameter value p.

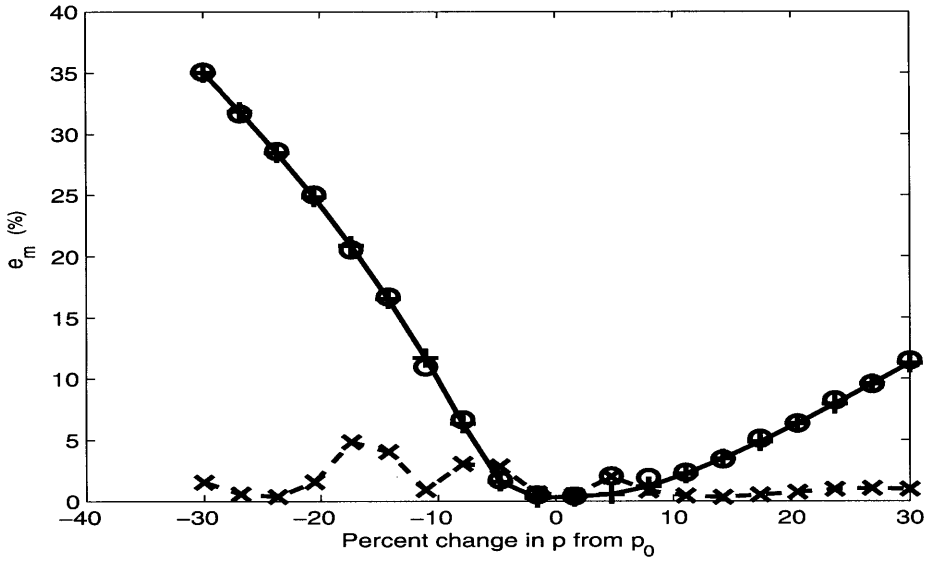


Figure 5-12: Output error, as defined in (5.19), between three different diode line models parameterized in p_V and simulated over a range of parameter values. The pluses correspond to error between system (5.5) and system (5.1), the circles correspond to error between system (5.1) and system (5.3), and the crosses correspond to error between system (5.5) and system (5.3).

It can be seen that for this particular case the PROM error is not significantly worse than the error from the large nonlinear system which was expanded in powers of the parameters (5.3). This indicates that both aspects of the reduction process, i.e. finding a good subspace and selecting linearization points, worked well for this example. However, the accuracy of both models compared to original system (5.1) declines rapidly as the parameter value moves beyond $\pm 10\%$. For this example we can conclude that if an accuracy over a larger range of parameter values is needed, a higher order expansion in the parameter would be required.

5.4.6 Sensitivity to Parameters

To determine how accurately the parameter dependence of the original system is captured in the PROM, we can compare output sensitivity to changes in the parameters for both the original system and the PROMs. Figure 5-13 compares these sensitivities for several

parameters for each of the three example systems. We define the sensitivity $\delta y(p)$ as

$$\delta y(p) = \left(\frac{\max_t |y(t) - y_0(t)|}{\max_t |y_0(t)|} \right) \times 100 \tag{5.20}$$

where $y(t)$ is the system output at parameter value p and $y_0(t)$ is the system output at nominal parameter value p_0.

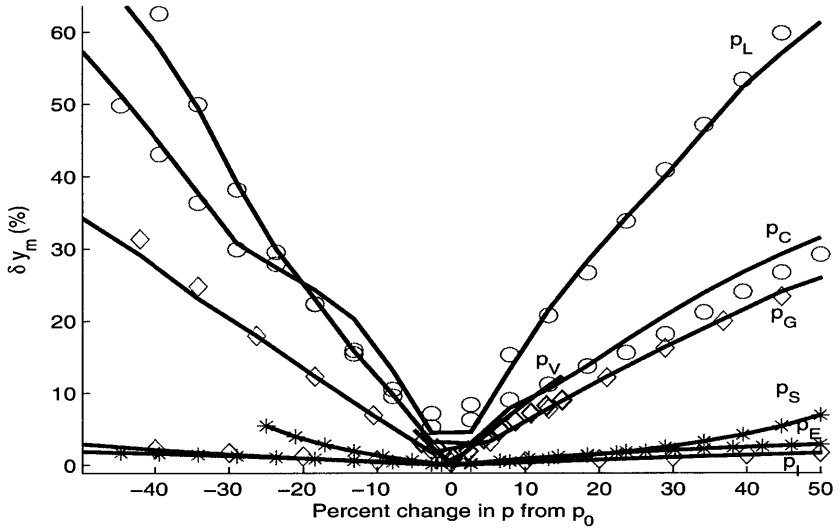


Figure 5-13: Output sensitivity of models to parameter changes, as defined in (5.19). The solid lines represent the original systems and the symbols represent the associated PROMs. Several parameters were considered for each example system, with the circles corresponding to the pulse-narrowing transmission line, the stars corresponding to the MEMs switch, and the diamonds corresponding to the diode line.

The figure shows that the PROMs do in fact capture the parameter dependence of the original system over a significant range of parameter values. The exact range of values depends on the system and parameter considered, as the system sensitivity is different for each parameter.

To validate our parameter-selecting training scheme in Section 5.2.4 we approximate the gradient of the state with respect to the parameters, $\frac{\partial x}{\partial p}$, and examine whether or not it lies in the subspace spanned by the columns of the projection matrix V. By the fundamental theorem of linear algebra, this can be determined by checking if $\frac{\partial x}{\partial p}$ is orthogonal to the left

	$0.5p_0$	p_0	$1.5p_0$
p_G	0.55	0.58	0.56
p_V	0.31	0.06	0.46
p_I	0.031	0.027	0.025
p_E	0.17	0.18	0.070
p_S	0.17	0.23	0.18
p_L	0.81	0.86	0.90
p_C	0.66	0.52	0.47

Table 5.5: Equation (5.21) computed on three different trajectories, corresponding to $0.5p_0, p_0, 1.5p_0$, for each parameter in our three examples

nullspace of V, which we define as

$$e_p(p) = \left\| \left(\frac{\tilde{\partial x}}{\partial p} \right)^T \mathcal{N}(V^T) \right\|. \tag{5.21}$$

Here $\frac{\tilde{\partial x}}{\partial p}$ is the largest three singular vectors of $\frac{\partial x}{\partial p}$ as computed in (5.9). Both the singular vectors and $\mathcal{N}(V^T)$ are normalized, hence $e_p(p) \in [0, 1]$, with $e_p(p) = 0$ meaning $\frac{\tilde{\partial x}}{\partial p}$ is exactly in the subspace spanned by the columns of V, and $e_p(p) = 1$ meaning $\frac{\tilde{\partial x}}{\partial p}$ is exactly orthogonal to the subspace spanned by the columns of V. We have computed this quantity at three different parameter values for each parameter in each system, and compare the results in Table 5.5.

The results of this test indicate that in some cases, such as for parameters p_E and p_S in the MEMs switch, we do not need to train at additional parameter values to capture the parameter dependence of the original system. It also shows that in other cases, such as for parameters p_L and p_C in the pulse-narrowing transmission line, increasing the range of the parameter values will take the trajectory to a significantly different subspace, and the reduced model would need to be updated by training that system at the additional parameter values. In general, these results match what we experienced in the training process, as we found the pulse-narrowing transmission line to be the most difficult system to model, and the MEMs switch to be the easiest. As a matter of fact, we can observe a correlation between the results in Table 5.5 and the sensitivities shown in Figure 5-13. Both tests indicate the pulse-narrowing transmission line is the most sensitive to changes in the parameters,

and that the MEMs switch is least sensitive to parameter changes.

Algorithm 4 NLPMOR

for each Training Input Signal **do**
  for each Training Point in the Parameter-Space **do**
    Linearize nonlinear system at initial state x_{L_0}
    while $t < t_{final}$ **do**
      Simulate linearized model to compute its next state x_i
      Set $KrLin = 0$
      if Exact Training Trajectories **then**
        Simulate nonlinear system to compute its next state x_t
        Compute $\Delta = ||x_t - x_i||$
        Set $x_n = x_t$
      else if Approximate Training Trajectories **then**
        Compute $\Delta = \min_j ||x_{L_j} - x_i||$
        Set $x_n = x_i$
      end if
      if $\Delta > \delta$ **then**
        Linearize nonlinear system at current state x_n
        $j \leftarrow j + 1$
        $x_{L_j} = x_n$
        $KrLin = 1$
      end if
      if $(KrAll \mid\mid KrLin)$ **then**
        if MORV **then**
          Use equation (5.6) to compute V_{new}
        else if PMORV **then**
          Use equation (5.7) to compute V_{new}
        end if
        $\tilde{V} = [\tilde{V}\ V_{new}]$
      end if
    end while
  end for
end for
Construct a new projection matrix V using only the dominant singular vectors of $\tilde{V}$
Project systems using V
Select weighting functions $w(x, X)$

Chapter 6

Stabilizing Trajectory Piecewise Linear Models

6.1 Introduction

One major reason nonlinear reduced models, such as piecewise linear (PWL) models, have failed to gain widespread acceptance and use is due to the lack of rigorous statements concerning their properties. One such property is stability. Without a guarantee of stability for PWL models, the models cannot be fully trusted for time-domain simulation, as the simulation results may grow unbounded. While there are many results for stable model reduction of linear systems, as described in detail in section 3.4, there exist few results for nonlinear systems.

To briefly recap, the trajectory piecewise linear (TPWL) method approximates nonlinear descriptor systems using a series of local linear approximations

$$\dot{q}(x) = f(x) \quad \longrightarrow \quad \frac{\partial}{\partial t}\left[\sum_i w_i(x)(E_i x + h_i)\right] = \sum_i w_i(x)[A_i x + k_i].$$

The resulting local linear models are then projected into a carefully chosen subspace, re-

[0]The material in this chapter has been previously published in [15, 17].

sulting in the nonlinear reduced model

$$\frac{d}{dt}\left[\sum_i w_i(z)(\hat{E}_i z + \hat{h}_i)\right] = \sum_i w_i(\hat{x})\left[\hat{A}_i\hat{x} + \hat{K}_i\right] + \hat{b}u.$$

It is important to notice here that the TPWL procedure may potentially produce unstable reduced models from originally stable systems. Such instabilities may arise in three places:

- Jacobian matrix pairs (E_i, A_i) of stable nonlinear systems are not guaranteed to be Hurwitz;
- the pair $(V^T E_i V, V^T A_i V)$ is not guaranteed to be Hurwitz even if (E_i, A_i) is Hurwitz;
- convex combinations of Hurwitz matrix pairs $(\sum_i w_i E_i, \sum_i w_i A_i)$ are not guaranteed to be Hurwitz.

In this chapter we present several approaches for efficiently creating stable reduced order PWL models from various classes of nonlinear systems. Stabilization of such reduced models is obtained both through reformulation of the system equations, and through the use of a nonlinear projection function. Additionally, several methods are presented for efficiently computing the nonlinear projection functions.

This chapter is organized as follows: In Section 6.2 we briefly review several important results concerning the stability of dynamical systems that will be used in this chapter. In Section 6.3 we consider first the case of stabilizing reduced models created from systems with linear descriptor functions, $q(x) = Ex$. In section 6.4 we propose a new model formulation allowing us to extend our stability results to some nonlinear descriptor functions $q(x)$. In Section 6.5 we consider all other systems for which we cannot guarantee global stability or which are originally unstable. In these cases we propose a nonlinear projection that is guaranteed to preserve stability for every stable local linear model resulting in a guarantee of local stability. Section 6.6 presents algorithms to compute efficiently the resulting nonlinear reduced models, with an emphasis on constructing the nonlinear left-projection functions. Finally, Section 6.7 presents results from the proposed algorithms applied to several examples of nonlinear systems including analog circuits and a MEMS device.

6.2 Review of Stability Theory

In this section we briefly review several definitions and results concerning the stability of dynamical systems that will be used throughout this chapter. For a more detailed analysis of stability, see Chapter 2.

The equilibrium point of a nonlinear dynamical system

$$E\dot{x} = f(x,u), \qquad y = c^T x, \tag{6.1}$$

is said to be **exponentially stable** if all solutions to the autonomous system (i.e. input $u = 0$) converge to the equilibrium state x_{eq} exponentially fast, regardless of initial condition. Without a loss of generality we may transform the coordinate system such that $x_{eq} = 0$. Internal stability can be proven through Lyapunov functions.

Theorem 6.2.1 ([100]). *The equilibrium point $x_{eq} = 0$ of system (6.1) is exponentially stable if there exist constants $\lambda_1, \lambda_2, \lambda_3 > 0$ and a continuously differentiable Lyapunov function $L(x)$ such that*

$$\lambda_1 x^T x \leq L(x) \leq \lambda_2 x^T x \tag{6.2}$$

$$\frac{\partial}{\partial t} L(x) \leq -\lambda_3 x^T x \tag{6.3}$$

$\forall t \geq 0$, $\forall x \in \mathbb{B}_r$. If $\mathbb{B}_r = \mathbb{R}^N$, then the equilibrium point is globally exponentially stable.

External stability concerns the system's ability to amplify signals from input u to output y. System (6.1) is said to be **small-signal finite-gain L_p stable** if there exist constants $r_p > 0$ and $\gamma_p < \infty$ such that

$$||y||_p \leq \gamma_p ||u||_p$$

for all $t > t_0$, given initial state $x(0) = 0$ and input $u(t)$ such that $||u||_\infty < r_p$. If $r_p = \infty$, then the system is simply said to be finite-gain L_p stable. External stability follows from internal stability provided the nonlinear functions describing the system are sufficiently well behaved.

Theorem 6.2.2 ([100]). *Suppose $x = 0$ is an exponentially stable equilibrium of system* (6.1). *If $f(x, u)$ is continuously differentiable and $f(x, u)$ is locally Lipschitz continuous at $(0, 0)$, then system* (6.1) *is small-signal finite-gain L_p stable. If $\mathbb{B}_r = \mathbb{R}^N$, then the system is finite-gain L_p stable.*

For the remainder of this chapter we will consider small-signal finite-gain L_2 stability and refer to it simply as input-output stability.

A linear system

$$E\dot{x} = Ax + bu(t)$$

the system is said to be stable if the generalized eigenvalues of the pair (E, A) have negative real part. Equivalently, we say the pair (E, A) is Hurwitz, or stable.

6.3 Stability of Piecewise-Linear Systems with Constant Descriptor Matrix

In general, PWL models created from stable nonlinear systems are not stable. This is because linearizations of an arbitrary stable nonlinear system are not necessarily stable, and interpolating between arbitrary stable linear models will not necessarily produce a stable model. However, there exist many nonlinear systems for which we can guarantee both that linearizations will always be stable, and that convex combinations of the resulting stable linear systems will also be stable. In this section we examine nonlinear systems that provably generate structured and stable linearizations, and can be formulated, either directly or through a change of coordinates as described in Subsection 2.4.3, to possess a constant descriptor matrix. In these cases we obtain finite-gain stability guarantees for the large-order PWL models as well as a stability-preserving linear projection framework.

6.3.1 Relaxing the Model

To begin, we introduce a new notation to concisely represent the PWL model. Define the matrix-valued functions

$$A_p(x) = \sum_i w_i(x) A_i \qquad\qquad E_p(x) = \sum_i w_i(x) E_i$$

and the vector-valued functions

$$k_p(x) = \sum_i w_i(x) k_i \qquad\qquad h_p(x) = \sum_i w_i(x) h_i$$

such that the large-order piecewise-linear approximation becomes

$$\begin{cases} \frac{d}{dt}\left[E_p(x)x + h_p(x)\right] = A_p(x)x + k_p(x) + bu \\ y = c^T x \end{cases} . \tag{6.4}$$

Here $E_p(x)$ and $A_p(x)$ are nonlinear matrix-valued functions that interpolate between the local matrices.

In addition, note that the PWL model (6.4) can be rewritten in a more general form as

$$\frac{d}{dt}\left[E_p(x)x + h_p(x)\right] = A_p(x)x + B_p(x,u) \qquad\qquad y = c^T x, \tag{6.5}$$

where

$$B_p(x,u) = bu_1 + k_p(x)u_2 \tag{6.6}$$

is a state-dependent input matrix. In this formulation we are treating the constant offset vectors resulting from the linearizations as additional input vectors with the new input $u_2(t)$. System (6.4) is obtained by selecting $u_2 = 1$ for all $t > 0$. Thus, systems of the form (6.4) are a subset of systems of the form (6.5), and any stability results that apply to the latter will also apply to the former.

6.3.2 Stability from Structured Matrices

We first consider systems described by models containing a constant descriptor matrix

$$E\dot{x} = A_p(x)x + B_p(x,u) \qquad\qquad y = c^T x. \tag{6.7}$$

Internal stability can be proven through the existence of Lyapunov functions satisfying constraints (6.2) and (6.3), as described in section 6.2. Finding Lyapunov functions for arbitrary nonlinear systems is difficult. However, often a PWL system's Jacobian matrices A_i will all share some nice structure because they are all linearizations of the same nonlinear function, and in those cases it may be possible to find a Lyapunov function that proves internal stability of the autonomous PWL system

$$\begin{cases} E\dot{x} = A_p(x)x \\ A_p(x) = \sum_i w_i(x)A_i. \end{cases} \tag{6.8}$$

For example, a Lyapunov function that proves stability for each individual linear system, and thus also for an interpolation of the systems, would suffice, and is specified by the following proposition.

Prop. 6.3.1 (Exponential Stability). *If $w_i(x) : \mathbb{R}^N \mapsto [0,1]$ are continuously differentiable functions such that $\sum_i w_i = 1$, and there exists an SPD matrix $P \succ 0$ such that the matrices*

$$Q_i = -\left(E^T P A_i + A_i^T P E\right) \tag{6.9}$$

are SPD for all i, then $L(x) = x^T E^T P E x$ is a Lyapunov function for System (6.8)*, and System* (6.8) *has a globally exponentially stable equilibrium at the origin.*

Proof. Consider the candidate Lyapunov function $L = x^T E^T P E x$. Since E is nonsingular and P is SPD, then $E^T P E$ is also SPD because it is a congruence transform of an SPD matrix, and condition (6.2) is satisfied

$$x^T x \left(\sigma_{min}\left(E^T P E\right)\right) \leq L(x) \leq x^T x \left(\sigma_{max}\left(E^T P E\right)\right).$$

Similarly, condition (6.3) is satisfied

$$\begin{aligned}\dot{L}(x) &= 2x^T E^T P E\dot{x} = x^T \left(E^T P A_p(x) + A_p(x)^T P E\right) x \\ &= \sum_i w_i(x) x^T \left(E^T P A_i + A_i^T P E\right) x \\ &= -\sum_i w_i(x) x^T Q_i x \leq -x^T x \min_i \{\sigma_{min}\left(Q_i\right)\}.\end{aligned}$$

Thus $L(x)$ is a Lyapunov function, and by Theorem 6.2.1 the system is globally exponentially stable. □

It is now possible, using the results from Proposition 6.3.1, to prove input-output stability for System (6.7).

Prop. 6.3.2 (I/O Stability). *If System* (6.8) *is globally exponentially stable with Lyapunov function* $L(x) = x^T E^T P E x$ *for some SPD matrix* P *(e.g. the assumptions of Proposition 6.3.1 hold),* $w_i(x) : \mathbb{R}^N \mapsto [0, 1]$ *are continuously differentiable functions such that* $\sum_i w_i = 1$*, then System* (6.7) *is input-output stable, and therefore the PWL system*

$$E\dot{x} = A_p(x)x + k_p(x) + bu, \qquad y = c^T x \tag{6.10}$$

is input-output stable.

Proof. This can be proven using Theorem 6.2.2. By assumption, the autonomous system is globally exponentially stable. Also by assumption, $w_i(x)$ are all continuously differentiable, and therefore $f(x, u)$ is also continuously differentiable. To prove Lipschitz continuity we examine the partial derivatives of $f(x, u) = A_p(x)x + B_p(x, u)$

$$\begin{aligned}\frac{\partial f_j}{\partial x_k} = &\sum_i \left[w_i(x) a^i_{jk} + \frac{\partial w_i(x)}{\partial x_k} \sum_m \left(a^i_{jm} x_m\right)\right] \\ &+ \sum_i \left[\frac{\partial w_i(x)}{\partial x_k} k_i u_2\right]\end{aligned}$$

$$\frac{\partial f_j}{\partial u_1} = b \qquad\qquad \frac{\partial f_j}{\partial u_2} = \sum_i w_i(x) k_i.$$

Here a^i_{jk} is the element of A_i in the j^{th} row and k^{th} column. By assumption, $\partial f_j / \partial u_1$ is bounded because it is constant, and $\partial f_j / \partial u_2$ is bounded for all x because w_i is bounded. Similarly, since $w_i(x)$ are Lipschitz, the derivatives $\partial w_i / \partial x_k$ are bounded. Thus, the Jacobian is locally bounded for all x, u, the functions are Lipschitz continuous by Observation 2.1.1, and the system is input-output stable by Theorem 6.2.2. □

In order to obtain global finite-gain stability we must add the additional constraint that $(\partial w_i(x)/\partial x)x$ is bounded for all x. This constraint is not restrictive, as it merely requires that the weights converge to some uniform value when the state becomes sufficiently large, rather than oscillate back and forth indefinitely. Practically, the PWL model is comprised of a finite number of linearizations that are locally accurate, so for x sufficiently far away from all local models the interpolation is no longer accurate regardless of the weighting functions, and thus the constraint will not affect the accuracy.

Examples of systems for which stability may be guaranteed through Propositions 6.3.1 and 6.3.2 are those that produce negative-definite Jacobian matrices. These include analog circuits comprised of monotonic elements such as inductors, capacitors, linear and nonlinear resistors, and diodes. One such example is presented in Section 6.7.1.

Note that the finite-gain stability results are based solely on the existence of the quadratic Lyapunov function, and do not explicitly require any special structure in the matrices A_i. Structured matrices, such as negative-definite matrices, are a sufficient condition for the existence of such a Lyapunov function, but are not a necessary condition.

6.3.3 Stability-Preserving Projection

In the previous section we presented conditions under which large-order PWL systems are both internally stable and finite-gain stable. In this section we present a projection framework that preserves these two stability properties in the reduced model.

Consider the PWL model System (6.7), and approximate the solution x in a low-

dimensional subspace as $x = V\hat{x}$, such that

$$EV\dot{\hat{x}} = A_p(V\hat{x})V\hat{x} + B_p(V\hat{x}, u). \tag{6.11}$$

A left-projection matrix U is next chosen to reduce the number of equations, resulting in the reduced-order model

$$U^T EV\dot{\hat{x}} = U^T A_p(V\hat{x})V\hat{x} + U^T B_p(V\hat{x}, u), \qquad y = c^T V\hat{x}. \tag{6.12}$$

By proper selection of the matrix U, it is possible to preserve internal stability in the reduced-order system

$$\begin{cases} \hat{E}\dot{\hat{x}} = \hat{A}_p(\hat{x})\hat{x} \\ \hat{A}_p(\hat{x}) = \sum_i w_i(\hat{x})U^T A_i V \\ \hat{E} = U^T EV. \end{cases} \tag{6.13}$$

Prop. 6.3.3 (Preservation of Lyapunov Functions). *If $L(x) = x^T E^T PEx$ is a Lyapunov function for System* (6.8) *(e.g. the assumptions of Proposition 6.3.1 hold), then given any right-projection matrix V, if we define a left-projection matrix $U^T = V^T E^T P$, then $\hat{L}(\hat{x}) = \hat{x}^T \hat{E}\hat{x}$ is a Lyapunov function for System* (6.13)*, where $\hat{E} = U^T EV$.*

Proof. To begin, note that the proposed Lyapunov function $\hat{L}(\hat{x})$ satisfies

$$\begin{aligned} \hat{L}(\hat{x}) &= \hat{x}^T V^T E^T PEV\hat{x} = L(V\hat{x}) \\ \dot{\hat{L}}(\hat{x}) &= 2\hat{x}^T \hat{E}\dot{\hat{x}} = 2\hat{x}^T \hat{A}(\hat{x})\hat{x} \\ &= 2\hat{x}^T V^T E^T PA(V\hat{x})\hat{x} = \dot{L}(V\hat{x}). \end{aligned}$$

By assumption, $L(x)$, and therefore $L(V\hat{x})$, satisfies (6.2) and (6.3). Thus $\hat{L}(\hat{x})$ satisfies conditions (6.2) and (6.3) and is a Lyapunov function for System (6.13). □

This result can also be seen as a direct application of Theorem 4.2.1 (which proves stability preservation through preservation of storage functions) to a nonlinear system, as

proposed in Section 4.2.3.

Given the existence of a quadratic Lyapunov function for the reduced model, it is now possible to apply the results of Section 6.3.2 to obtain guarantees for the various notions of stability for the reduced model.

Corollary 6.3.1. *If $L = x^T E^T P E x$ is a Lyapunov function for system* (6.8)*, V is a right-projection matrix, and $w_i(x) : \mathbb{R}^N \mapsto [0, 1]$ are continuously differentiable functions such that $\sum_i w_i = 1$, then if we define the left-projection matrix $U = V^T E^T P$, the reduced-order PWL model* (6.13) *is globally exponentially stable and system* (6.12) *is input-output stable.*

Proof. By Proposition 6.3.3, $\hat{L}(\hat{x}) = \hat{x}^T V^T E P E V \hat{x}$ is a Lyapunov function for System (6.13), and therefore the reduced model is globally exponentially stable. Exponential stability combined with Proposition 6.3.2 yields finite-gain stability for the reduced model. □

6.4 Stability of Piecewise-Linear Systems with Nonlinear Descriptor Functions

6.4.1 Difficulties with Nonlinear Descriptor Functions

In the previous section we considered only systems described by models possessing a constant descriptor matrix E. In this section we extend the results to the case where the descriptor function $q(x)$ is nonlinear.

It is more difficult to prove stability for systems with nonlinear descriptor functions because the quadratic Lyapunov function approach from Section 6.3.2 does not directly apply. Additionally, even if the large-order PWL system is stable, we cannot directly apply the approach of Section 6.3.3 to preserve stability in the reduced model.

For example, consider the PWL model with nonlinear descriptor function whose state is approximated in the reduced space as $x = V\hat{x}$

$$\frac{d}{dt}\left[E_p(V\hat{x})V\hat{x} + h_p(V\hat{x})\right] = A_p(V\hat{x})V\hat{x} + B_p(V\hat{x}, u). \tag{6.14}$$

Attempting to preserve stability by preserving Lyapunov functions as done in Section 6.3.3, i.e. to ensure that $\hat{L}(\hat{x}) = L(V\hat{x})$, requires the selection of a nonlinear left-projection matrix

$$U(\hat{x})^T = V^T E_p(V\hat{x})P. \tag{6.15}$$

Applying a nonlinear left-projection $U(\hat{x})$ to (6.14) results in

$$U(\hat{x})^T \frac{d}{dt}\left[E_p(V\hat{x})V\hat{x} + h(V\hat{x})\right] = U(\hat{x})^T A_p(V\hat{x})V\hat{x} + U(\hat{x})^T B_p(V\hat{x}, u), \tag{6.16}$$

which is not a reduced-order system in the typical sense. The expression on the left cannot be explicitly multiplied out because the time-dependence in $U(\hat{x})$ prevents it from passing directly through the time-derivative operators. As a result, systems of the form (6.16) require $O(N)$ computations to evaluate and are not desirable for the purpose of simulation.

6.4.2 Alternative Formulations

To avoid the projection problems resulting from nonlinear descriptor functions, we will rewrite the system in a manner that separates nonlinearities from the time-derivative operator. First assume there is no explicit time-dependence in $q(x)$. This allows for the nonlinear descriptor system

$$\frac{d}{dt}\left[q(x)\right] = f(x)$$

to be rewritten as

$$Q(x)\dot{x} = f(x)$$

where $Q(x) = \frac{\partial q(x)}{\partial x}$ is a nonlinear matrix-valued function. Additionally, the system can be rewritten as

$$\dot{x} = Q(x)^{-1} f(x) = g(x). \tag{6.17}$$

Note that no approximations have been made so far.

A linearization of System (6.17) at state x_i yields the local linear model

$$\dot{x} = A_i x + k_i \tag{6.18}$$

with system matrices

$$\begin{aligned} A_i &= Q(x_i)^{-1} J(x_i) - Q(x)^{-1} \frac{\partial Q(x)}{\partial x} Q(x)^{-1} f(x_i) \\ k_i &= Q(x_i)^{-1} f(x_i) - A_i x_i \end{aligned} \tag{6.19}$$

where $J(x) = \partial f / \partial x$. If the function $q(x)$ is known explicitly, then $Q(x)$ and $\partial Q / \partial x$ can also be computed, resulting in an accurate constant-descriptor PWL model

$$\begin{cases} \dot{x} = A_p(x) x \\ A_p(x) = \sum_i w_i(x) A_i \end{cases} \tag{6.20}$$

where A_i are defined in (6.19), and each linear model is accurate to first-order in $f(x)$ and to first-order in $Q(x)$.

However, the function $q(x)$ is not always available analytically. Often, only samples of $q(x)$ and $Q(x)$ are available. In this case it is possible to ignore the derivative of $Q(x)$, simplifying the linearizations to

$$A_i = Q(x_i)^{-1} J(x_i), \qquad k_i = Q(x_i)^{-1} f(x_i) - A_i x_i. \tag{6.21}$$

The resulting PWL system has the form of System (6.20), where the system matrices A_i are defined in (6.21), and each linear model is accurate to first-order in $f(x)$ and zeroth-

order in $Q(x)$. A piecewise-constant approximation of $Q(x)$ is not a poor approximation, because in general the function $q(x)$ must be well-behaved simply to ensure that a unique solution to the nonlinear system exists. In addition, if $Q(x)$ changes sharply, the accuracy of the approximation can always be increased by increasing the number of linearization points.

6.4.3 Stable PWL Systems from Nonlinear Descriptor Functions

For System (6.20), regardless of whether using system matrices (6.19) or (6.21), we can directly apply Proposition 6.3.1 to obtain an exponential stability guarantee because of the constant descriptor matrix $E = I$. However, we must consider one additional factor before applying Proposition 6.3.2 to obtain a finite-gain stability guarantee. As a result of the reformulation of the equations, the system now possesses additional state-dependence in the input function $B_p(x, u)$,

$$\begin{cases} \dot{x} = A_p(x)x + B_p(x,u) \\ A_p(x) = \sum_i w_i(x) A_i \\ B_p(x,u) = \sum_i w_i(x) \left[b_i u_1 + k_i u_2 \right], \end{cases} \tag{6.22}$$

where, for example, $b_i = Q_i^{-1} b$. All of the other assumptions of Proposition 6.3.2 hold, so it merely needs to be shown that the input function $B_p(x, u)$ is still Lipschitz continuous.

Prop. 6.4.1 (I/O Stability). *If System* (6.20) *is globally exponentially stable with Lyapunov function $L(x) = x^T P x$ for some SPD matrix P (e.g. the assumptions of Proposition 6.3.1 hold), and $w_i(x) : \mathbb{R}^N \mapsto [0, 1]$ are continuously differentiable functions such that $\sum_i w_i = 1$, then System* (6.22) *is input-output stable.*

Proof. Following the proof of Proposition 6.3.2, we simply need to show that $B_p(x, u)$ is Lipschitz continuous. The partial derivatives are

$$\frac{\partial B_p}{\partial x_k} = \sum_i \left[\frac{\partial w_i(x)}{\partial x_k} \left(b_i u_1 + k_i u_2 \right) \right]$$

$$\frac{\partial B_p}{\partial u_1} = \sum_i w_i(x) b_i, \qquad \frac{\partial B_p}{\partial u_2} = \sum_i w_i(x) k_i,$$

which are all locally bounded by assumption. Thus, by Theorem 6.2.2, System (6.22) is input-output stable. □

Additionally, given a right-projection matrix V, the left-projection matrix U can be chosen such that the reduced model resulting from application of U and V to System (6.22) is input-output stable.

Corollary 6.4.1. *If $L = x^T P x$ is a Lyapunov function for system* (6.20)*, V is a right-projection matrix, and $w_i(x) : \mathbb{R}^N \mapsto [0, 1]$ are continuously differentiable functions such that $\sum_i w_i = 1$, then if we define the left-projection matrix $U = V^T P$, the reduced-order PWL model*

$$\begin{cases} \hat{E}\dot{\hat{x}} = \hat{A}_p(\hat{x})\hat{x} + \hat{B}_p(\hat{x}, u) \\ \hat{A}_p(\hat{x}) = \sum_i w_i(\hat{x}) U^T A_i V \\ \hat{B}_p(\hat{x}) = \sum_i w_i(\hat{x}) U^T \left[b_i u_1 + k_i u_2 \right] \\ \hat{E} = U^T V. \end{cases} \tag{6.23}$$

is input-output stable.

Proof. Proposition 6.3.3 guarantees that $\hat{L}(\hat{x}) = \hat{x}^T V^T P V \hat{x}$ is a Lyapunov function for the reduced model, and Proposition 6.4.1 applied to System (6.23) guarantees input-output stability. □

Note that the reduced model terms such as $U^T Q_i^{-1} J_i V$ can be efficiently computed by first solving the linear system $Q_i^T M = U$ for the matrix $M \in \mathbb{R}^{N \times q}$, and subsequently evaluating $(U^T Q_i^{-1}) J_i V = M^T J_i V$.

For systems with complicated and unstructured descriptor functions, it becomes difficult to prove stability with quadratic Lyapunov functions. These issues will be addressed in the following section.

6.5 Unstructured and Unstable PWL Systems

All of the results presented up to this point have relied on the assumption that the large-order PWL system is stable and that there exists a quadratic Lyapunov function. However, in general it may not be easy, or even possible, to find a quadratic Lyapunov function for a stable PWL system. Additionally, a stable nonlinear system may produce an unstable PWL model. In these cases we will try both to "eliminate" as much instability as possible from the large-order PWL system through equation reformulation, and to utilize a projection that preserves or regenerates stability in as many of the linear models as possible.

6.5.1 Stability Through Reformulation

Although the reformulation in the previous section permits the application of the results from Section 6.3, it is possible that an alternative reformulation may be more useful in some situations. Consider the case where $Q(x)$ is approximated by a zeroth-order expansion, and interpolate the descriptor matrices on the left side of the equation directly. The resulting system

$$\begin{cases} E_p(x)\dot{x} = A_p(x)x \\ A_p(x) = \sum_i w_i(x)A_i \qquad A_i = J(x_i) \\ E_p(x) = \sum_i w_i(x)E_i \qquad E_i = Q(x_i) \end{cases} \tag{6.24}$$

has a nonlinear descriptor matrix, is comprised of local linear models that are accurate to zeroth-order in $Q(x)$ and first-order in $f(x)$, and can be efficiently reduced with a nonlinear projection operator.

One possible benefit of this formulation is that the system matrices E_i and A_i in System (6.24) are much more likely to have a nice structure, such as symmetry or definiteness, than the system matrices (6.19) and (6.21). In general, structured system matrices make it easier to find Lyapunov functions as described in the previous section. One circuit example for which reformulation of the equations improves stability of PWL approximations is presented in Section 6.7.2.

6.5.2 Stability From Projection

Finally, we consider the case where the PWL system is not stable, and some of the linear models (E_i, A_i) are unstable. Given the large-order PWL system and reduced-state approximation $x = V\hat{x}$, we may reduce the number of equations with a weighted piecewise-constant left-projection function

$$U(\hat{x}) = \sum_{k=1}^{\kappa} \mu_k(\hat{x}) U_k \tag{6.25}$$

where $\mu_k(\hat{x}) \in [0, 1]$, $\sum_k \mu_k(\hat{x}) = 1$, and $U_k \in \mathbb{R}^{N \times q}$. Consider System (6.24) evaluated at $x = V\hat{x}$ and left-multiplied by $U(\hat{x})^T$, leading to

$$\sum_k \mu_k(\hat{x}) U_k^T \sum_i w_i(\hat{x}) E_i V \dot{\hat{x}} = \sum_k \mu_k(\hat{x}) U_k^T \left(\sum_i w_i(\hat{x}) \left[A_i V \hat{x} \right] + B(\hat{x}) u \right),$$

which can be rearranged as

$$\sum_k \sum_i \mu_k(\hat{x}) w_i(\hat{x}) U_k^T E_i V \dot{\hat{x}} = \sum_k \sum_i \mu_k(\hat{x}) w_i(\hat{x}) U_k^T A_i V \hat{x} + \sum_k \mu_k(\hat{x}) U_k^T B_k(\hat{x}) u.$$

To simplify the notation, we define

$$\hat{B}_{ki} = U_k^T [b_i,\ k_i], \qquad \hat{B}_p(\hat{x}, u) = \sum_i \sum_k \mu_k(\hat{x}) w_i(\hat{x}) \hat{B}_{ki} u$$

$$\hat{E}_{ki} = U_k^T E_i V, \qquad \hat{E}_p(\hat{x}) = \sum_i \sum_k \mu_k(\hat{x}) w_i(\hat{x}) \hat{E}_{ki}$$

$$\hat{A}_{ki} = U_k^T A_i V, \qquad \hat{A}_p(\hat{x}) = \sum_i \sum_k \mu_k(\hat{x}) w_i(\hat{x}) \hat{A}_{ki}$$

resulting in the final reduced nonlinear descriptor system

$$\hat{E}_p(\hat{x}) \dot{\hat{x}} = \hat{A}_p(\hat{x}) \hat{x} + \hat{B}_p(\hat{x}) u. \tag{6.26}$$

We wish to select the matrices U_k such that the reduced models $(\hat{E}_{kk}, \hat{A}_{kk})$ are stable for all k. Two possible available methods for computing such U_k are to select $U_k = PE_kV$,

where P solves

$$E^T P A + A^T P E \preceq 0$$

(see Section 3.4 for details), or as the solution to an optimization problem

$$\begin{aligned} &\min_{U_k} ||U_k - U_0||_S \\ &U_k^T E_k V \succ 0 \\ &U_k^T A_k V + V^T A_k^T U_k \preceq 0 \end{aligned} \tag{6.27}$$

(see Section 4.4 for details). These two approaches were first presented in [55] and [16] respectively. For example, if we define the matrices

$$U_k^T = \begin{cases} \left(V^T E_k^T P_k E_k V\right)^{-1} V^T E_k^T P_k & \text{if } k \in \mathbb{I} \\ V^T \left(V^T E_k V\right)^{-1} & \text{if } k \notin \mathbb{I} \end{cases} \tag{6.28}$$

where

$$\mathbb{I} = \left\{ k \in \{1, \ldots, \kappa\} \;\middle|\; (E_k, A_k) \text{ is Hurwitz} \right\}, \tag{6.29}$$

and P_k solves

$$E_k^T P_k A_k + A_k^T P_k E_k = -Q_k \prec 0 \tag{6.30}$$

for (E_k, A_k), then $\hat{E}_{kk} = U_k^T E_k V = I$ and $\hat{A}_{kk}$ is Hurwitz for all k.

To completely specify the reduced order system (6.26), we must specify a set of left-projection weights $\mu_k(\hat{x})$. One possible choice of $\mu_k(\hat{x})$ that simplifies the model is

$$\mu_k(\hat{x}) = \begin{cases} 1 & \text{if } \hat{x}_k = \text{argmin}_i \, ||\hat{x} - \hat{x}_i|| \\ 0 & \text{else} , \end{cases}$$

resulting in reduced model terms

$$\hat{E}_p(\hat{x}) = \sum_i w_i(\hat{x})\hat{E}_{ii}, \qquad \hat{A}_p(\hat{x}) = \sum_i w_i(\hat{x})\hat{A}_{ii}$$
$$\hat{B}_p(\hat{x}, u) = \sum_i w_i(\hat{x})\hat{B}_i u.$$

Note that by our choice of U_k we also obtain $\hat{E}_p(\hat{x}) = I$.

While we cannot guarantee global stability of the resulting reduced model through the existence of a Lyapunov function, our projection guarantees that stability will be preserved for all stable local linear models, and additionally that the equilibrium point of the reduced model will be at least locally stable because there will always be a stable local model at the equilibrium point. In our experience, reduced models created with the proposed stabilizing projection have always produced stable outputs in response to typical inputs of interest, even though the models are not provably globally stable. Several examples using this stabilizing projection scheme will be presented in Sections 6.7.3 and 6.7.4.

6.6 Implementation

For nonlinear systems producing unstructured and unstable Jacobian matrices, the stabilizing nonlinear left-projection technique presented in Section 6.5.2 must be used to create stable reduced models. Constructing the nonlinear projection can be extremely computationally expensive, as it requires computing a stabilizing left-projection matrix for every local linear model. The left projection matrices defined in (6.28) are particularly expensive as they require solving Lyapunov matrix equations for each linear system. Although there exist methods (such as [5, 6]) for solving Lyapunov equations that perform better than $O(N^3)$, this matrix equation solution is the dominant computational cost in creating the reduced models using (6.28). In this section we present one approach to reduce the computational costs of solving Lyapunov equations for the linearized systems, as well as present our full model reduction algorithm.

6.6.1 Reusability of Lyapunov Equation Solutions

We first consider the nonlinear left-projection function $U(\hat{x})$, as defined in (6.25), with U_k as defined in (6.28). Constructing $U(\hat{x})$ is expensive because we assume that there does not exist one matrix $P \succ 0$ that satisfies (6.30) for *all* k, and thus we have to solve Lyapunov matrix equations for every local linear model. However, there may be a matrix $P \succ 0$ that satisfies (6.30) for *some* set of k. That is, given a solution P_k to (6.30) for a single k, there may exist $j \neq k$ such that

$$E_j^T P_k A_j + A_j^T P_k E_j \prec 0.$$

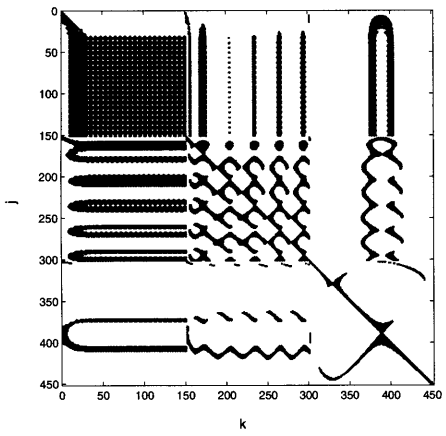
In this case we may "reuse" the Lyapunov matrix equation solution to also prove stability for linear model j. Additionally, we may also reuse the matrix P_k for constructing the stability-preserving local left-projection matrix U_j for linear model j, such that $U_j = P_k E_j V (V^T E_j^T P_k E_j V)^{-1}$ preserves stability for linear system (E_j, A_j). Since all local models are linearizations of the same physical system, it is likely that some matrix equation solutions P_k may be reused to satisfy other matrix equations.

Figure 6-1(a) plots the reusability of Lyapunov equation solutions for linearizations of a system, discussed in detail in Section 6.7.4, described by a model of the form

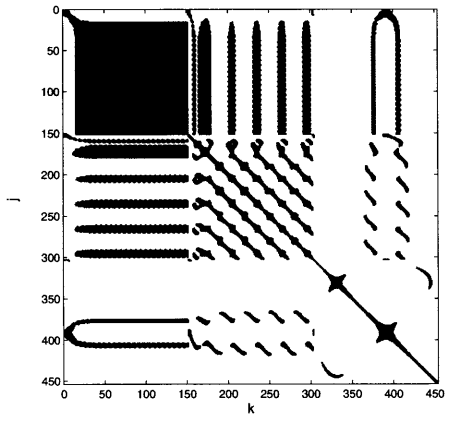
$$Q(x)\dot{x} = f(x) + b(x)u. \tag{6.31}$$

A dark dot in location (j, k) on Figure 6-1(a) signifies that the matrix P_k (which was constructed to satisfy $E_k^T P_k A_k + A_k^T P_k E_k \prec 0$) satisfies $E_j^T P_k A_j + A_j^T P_k E_j \prec 0$. Note that the plot is not symmetric because $A_i \neq A_i^T$ in this example.

From the near-periodic structure of this plot, it appears there is a correlation between reusability of Lyapunov equation solutions and state-space location of the linearization points from which the linear models are created. Figure 6-1(b) displays the proximity of linearization points in the state space. In this plot, a dark dot in location (j, k) indicates that the distance between linearization points x_j and x_k is less than some small tolerance: $||x_j - x_k|| < \epsilon$.



(a) Reusability of Lyapunov solutions – All pairs $\{P_k, (E_j, A_j)\}$ that satisfy $E_j^T P_k A_j + A_j^T P_k E_j \prec 0$. The system matrices (E_i, A_i) were obtained by training system (6.38) with three different sinusoidal inputs.



(b) All linearization point pairs (x_j, x_k) that satisfy $||x_j - x_k|| < \epsilon$. The linearization points were obtained by training system (6.38) with three different sinusoidal inputs.

Figure 6-1: Comparison of reusability of Lyapunov matrix equation solutions with proximity of linearization points.

The correlation between the patterns in Figures 6-1(a) and 6-1(b) can be explained by considering the perturbed linear system

$$(E_k + \Delta E)\,\dot{x} = (A_k + \Delta A)\,x.$$

This system is stable if there exist SPD matrices P_k, Q_k such that

$$E_k^T P_k A_k + A_k^T P_k E_k = -Q_k - \tilde{Q} \tag{6.32}$$

where

$$\begin{aligned}\tilde{Q} =& (\Delta E^T P \Delta A + \Delta A^T P \Delta E) + (\Delta E^T P A + A^T P \Delta E) \\ &+ (E^T P \Delta A + \Delta A^T P E).\end{aligned}$$

Assume there exist SPD matrices P_k, Q_k that satisfy (6.32) for $\Delta E = 0$ and $\Delta A = 0$. Next, define $\Delta E = E_j - E_k$ and $\Delta A = A_j - A_k$ such that the perturbed system is actually linear system j. If $\sigma_{\max}(\tilde{Q}) < \sigma_{\min}(Q)$, then we may take $P_j = P_k$ and $Q_j = Q_k - \tilde{Q}$

to satisfy the Lyapunov equation for linear system (E_j, A_j). Thus for ΔE and ΔA small enough, the Lyapunov solutions are reusable. Note that $\tilde{Q} \to 0$ as $\Delta E \to 0$ and $\Delta A \to 0$. From their definitions,

$$\Delta A = A_j - A_k = \frac{\partial f(x_j)}{\partial x} - \frac{\partial f(x_k)}{\partial x}$$
$$\Delta E = E_j - E_k = \frac{\partial q(x_j)}{\partial x} - \frac{\partial q(x_k)}{\partial x},$$

and by the smoothness of $f(x)$ and $q(x)$, we find that the perturbations go to zero as $x_j \to x_k$, and thus the Lyapunov equation solutions will be reusable for models arising from sufficiently close linearization points. If linearization points are too close and the models are too similar, then the model is redundant and not needed in the PWL approximation. However, in our experience, even after removing all redundant models we have found that it is often still possible to reduce the number of required Lyapunov equation solves by at least 50% by reusing Lyapunov matrix equation solutions.

An algorithm to exploit this fact might first search through existing solutions to Lyapunov equations corresponding to nearby linearized models, and then test those existing solutions on the given model before solving a new Lyapunov equation for the given model. Although this procedure will require fewer Lyapunov equations solutions, it is still expensive because it requires matrix-matrix products and eigen-decompositions for matrices in $\mathbb{R}^{N\times N}$. However, since it is only required that $\hat{Q}_{jk} = V^T E_j^T P_k A_j V + V^T A_j^T P_k E_j V \in \mathbb{R}^{q\times q}$ be a symmetric negative-definite matrix, it is possible instead to check if this smaller term is negative-definite. The eigen-decomposition is now performed on a size $q \times q$ matrix instead of a size $N \times N$ matrix, and the cost of matrix-matrix products is reduced from $O(N^3)$ to $O(N^2 q)$. An example of this procedure is presented in Algorithm 5, where the parameter τ defines the maximum number of existing solutions P_k that will be tested (in our experiments $\tau = 25$ has produced good results).

Algorithm 5 Reusability of Matrix Equation Solutions

1: Given a linear model pair (E_j, A_j) and linearization point x_j, a set $\mathcal{P} = \{P_k\}$ of SPD matrices P_k, a set of linearization points $X = \{x_k\}$, and an orthonormal right-projection matrix V
2: Compute $V_{aj} \in \mathbb{R}^{N\times q}$, $V_{ej} \in \mathbb{R}^{N\times q}$

$$V_{aj} = A_j V, \qquad V_{ej} = E_j V \tag{6.33}$$

3: **for** $m = 1 : \tau$ **do**
4: Find $x_k = \text{argmin}_i ||x_j - x_i||$ where $x_i \in X$
5: Compute $\hat{Q}_{jk} = V_{ej}^T P_k V_{aj} + V_{aj}^T P_k V_{ej}$
6: **if** $\lambda_{max}(\hat{Q}_{jk}) \prec 0$ **then**
7: Define $U_j = P_k V_{ej}(V_{ej}^T P_k V_{ej})^{-1}$
8: **break for**
9: **else**
10: Remove x_k from X and try again
11: **end if**
12: **end for**
13: **if** no reusable solution found **then**
14: Solve for P_j: $E_j^T P_j A_j + A_j^T P_j E_j = -I$
15: Define $U_j = P_j V_{ej}(V_{ej}^T P_j V_{ej})^{-1}$
16: Add P_j to $\mathcal{P}$: $\mathcal{P} = \{\mathcal{P}, P_j\}$
17: **end if**

6.6.2 Algorithm

In this section we present a routine to create stable PWL reduced models from originally stable nonlinear systems by using the nonlinear projection methodology described in Section 6.5.2. Our procedure, summarized in Algorithm 6, is described as follows.

Given a stable nonlinear descriptor system

$$\frac{d}{dt}[q(x)] = f(x) + bu, \tag{6.34}$$

a training procedure is used to obtain κ linear models. Information from the trajectories and linear models is then used to construct an orthonormal projection matrix V, using, for example, Krylov vectors. Details on methods for training and constructing V can be found in [78, 98, 95, 14]. At this point, stabilizing left-projection matrices U_m are computed for each Hurwitz matrix pair (E_m, A_m). Such U_m can be computed using the technique

Algorithm 6 STPWL: Stabilizing Trajectory Piecewise Linear

1: Train System (6.34) to obtain κ linear model pairs (E_i, A_i) with corresponding linearization points x_i
2: Construct orthonormal right-projection matrix V
3: Set $\mathcal{P} = \{\}$
4: **for** $j = 1 : \kappa$ **do**
5: Compute stabilizing U_j for model (E_j, A_j) using, for instance, Algorithm 5 or optimization problem (6.27)
6: **if** no solution found **then**
7: Unstable model: Define $U_j = V(V^T E_j^T V)^{-1}$
8: **end if**
9: Project system j with U_j

$$\hat{A}_j = U_j^T A_j V, \qquad \hat{k}_j = U_j^T k_j, \qquad \hat{b}_j = U_j^T b$$

10: Project linearization points $\hat{x}_j = V^T x_j$
11: **end for**
12: Obtain ROM of the form

$$\dot{\hat{x}} = \sum_{j=1}^{\kappa} w_j(\hat{x}) \hat{A}_j \hat{x} + \hat{B}(\hat{x}) u$$

described in Algorithm 5, or by solving (6.27). In the case of unstable linear models we cannot guarantee stability in the corresponding local reduced model through projection, so we define the projection matrix as $U_m = V(V^T E_m^T V)^{-1}$ to ensure that $U_m^T E_m V = I$. The resulting local left-projection matrices U_m are used to project the local models (E_m, A_m). The final result is a collection of stable linear models that are the basis of a reduced-order PWL model.

The storage cost and simulation cost of the final reduced model is the same as that of a model created with the traditional projection approach, i.e. $U = V$. All of the additional computational costs occur offline as part of model generation. Additionally, using stabilizing left-projection matrices obtained from optimization problem (6.27) can be much cheaper computationally than the alternative proposed approach. For example, when reducing a linear system of size $N = 1500$ to reduced order $q = 15$, we have found that solving (6.27) is more than 10 times faster than solving the corresponding Lyapunov matrix equation.

6.7 Examples

In this section we will examine several nonlinear systems whose PWL approximations exhibit the properties considered in Sections 6.3, 6.4, and 6.5, and present results from the proposed reduction algorithms applied to such systems. All of the model generation and simulation results for both the large and reduced systems were performed in Matlab on a desktop computer with a 3.33GHz Intel dual core processor and 4GB of RAM. Additional simulation speedups of the PWL reduced models could be obtained by using previously reported techniques in [95], such as fast nearest-neighbor searches, without altering any of the stability results obtained from projection. Additionally, solving Lyapunov equations for constructing the left-projection matrices can be performed much faster using recent algorithms for solving matrix equations [5, 6] instead of Matlab solvers.

6.7.1 Example of Systems with Constant Descriptor Matrix

We first consider nonlinear systems described by models with constant descriptor matrices and structured Jacobian matrices such that the system satisfies the assumptions of Proposition 6.3.1 in Section 6.3.2. One such example is a system whose Jacobian matrices A_i are negative-definite, and whose descriptor matrix E is an SPD matrix. In this case select $P = E^{-1}$, which is also SPD, leading to the Lyapunov function $L(x) = x^T Ex$ and left-projection matrix $U = V$. Such systems with structured Jacobian matrices are encountered, for instance, when using PWL approximations on nonlinear circuits comprised of monotonic nonlinear elements such as nonlinear resistors and diodes.

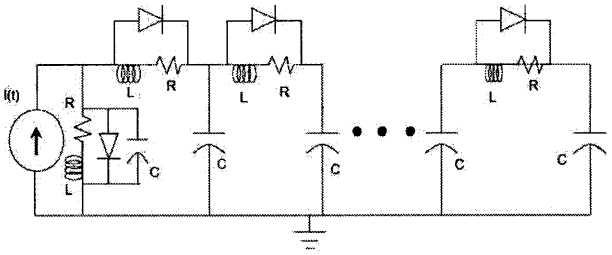


Figure 6-2: Nonlinear analog circuit containing monotonic devices.

For example, consider the following nonlinear analog circuit, shown in Figure 6-2 and first considered in [98], satisfying the previous criteria (note that this example is slightly different than the diode line example considered in section 5.3.1 because it contains inductors). The circuit contains the monotonic elements resistors, capacitors, inductors, and diodes. The current conservation law applied to this circuit produces the conservation equations

$$\begin{aligned} C_n \frac{dV_n}{dt} &= I_n - I_{n+1} + I_0 \left(e^{\alpha(V_{n-1}-V_n)} - 1\right) \\ &\quad - I_0 \left(e^{\alpha(V_n - V_{n+1})} - 1\right) \\ L_n \frac{dI_n}{dt} &= V_{n-1} - V_n - I_n R_n \end{aligned}$$

at each node, leading to the state space model

$$E\dot{x} = Gx + f(x) + bu \tag{6.35}$$

where E is a constant SPD matrix, G is a constant stable (but not symmetric) matrix, and f is a nonlinearity whose Jacobians are always negative-definite. For this system, we find that $A_i + A_i^T$ is always negative-definite, and thus $L(x) = x^T E x$ is a Lyapunov function for the PWL system. Thus, Proposition 6.3.1 guarantees internal stability.

Additionally, Proposition 6.3.2 in Section 6.3.2 guarantees input-output stability for the large-order PWL model, and Proposition 6.3.3 together with Corollary 6.3.1 from Section 6.3.3 guarantee that the reduced model created with the left-projection matrix $U = V$ will also be input-output stable for any right-projection matrix V. Figure 6-3 plots several outputs of a stable reduced model of the nonlinear transmission line created with the projection $U = V$ where V was constructed to match moments. The original system has order $N = 500$ and was trained with sinusoidal inputs of varying amplitude and frequency around 1GHz. The resulting reduced model has order $q = 15$ and consists of approximately $2,000$ local linear models, resulting in a simulation speedup factor of about 15.

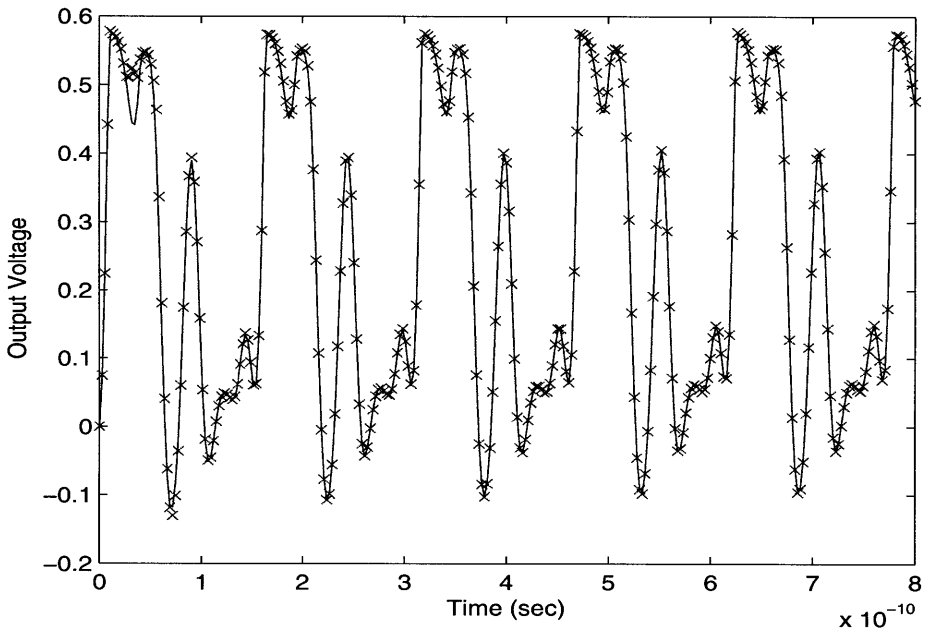


Figure 6-3: Comparison of outputs from a 15th order stable reduced model (crosses) and a 500th order stable full model (solid line) resulting from a multi-tone input to the RLCD line shown in Figure 6-2.

6.7.2 Reformulation for Systems with Nonlinear Descriptor Functions

In section 6.5.1 we considered reformulating nonlinear descriptor systems such that the resulting Jacobian matrices, while less accurate, were more likely to be structured and stable. To illustrate this point, we consider the nonlinear transmission line used for signal shaping described in Section 5.3.3.

The nonlinearity arises from the voltage dependence of the capacitors, which is approximated as $C_n = C(V_n) \approx C_0(1 - b_c V_n)$. Setting the system state to the node voltages and inductor currents, system equations can be derived using Kirchoff's current law and nodal analysis. The input is an ideal voltage source $u(t) = V_s(t)$, and the output is the voltage at some node m along the line, $y(t) = V_m(t)$. Using this formulation, the system equations for an interior node n using the traditional constant-descriptor formulation would be of the

form

$$\frac{dV_n}{dt} = \frac{I_n - I_{n+1}}{C_n(V_n)}, \qquad \frac{dI_n}{dt} = \frac{V_{n-1} - V_n}{L_n},$$

resulting in a model of the form

$$\dot{x} = f(x) + b(x)u. \tag{6.36}$$

If, on the other hand, one were to allow the system to possess a nonlinear descriptor matrix

$$C_n(V_n)\frac{dV_n}{dt} = I_n - I_{n+1}, \qquad L_n\frac{dI_n}{dt} = V_{n-1} - V_n,$$

the state space model becomes

$$Q(x)\dot{x} = Ax + bu \qquad y = c^T x. \tag{6.37}$$

Although the nonlinear descriptor formulation (6.37) will produce less accurate local models, the PWL interpolation of the collection of models is still sufficiently accurate. Figure 6-4 compares the highly nonlinear outputs of a large-order PWL model (with order $N = 200$) created from system (6.37) and the original nonlinear system in response to a sinusoidal input. In this example the PWL system was created by training with sinusoidal inputs and consists of approximately $3,000$ local models.

Now consider PWL approximations to these two nonlinear systems, each of which is comprised of linear models created at the same set of linearization points. These two sets of linear models will be different. Table 6.1 compares the number of unstable linear models generated by linearizations of the two nonlinear systems, as well as the number of unstable reduced-order linear models created by a Galerkin projection framework ($U = V$) when projected down to reduced order $q = 40$.

For this example the piecewise-constant descriptor formulation (6.37) produces fewer unstable large-order linear models and zero unstable reduced-order linear models, while the

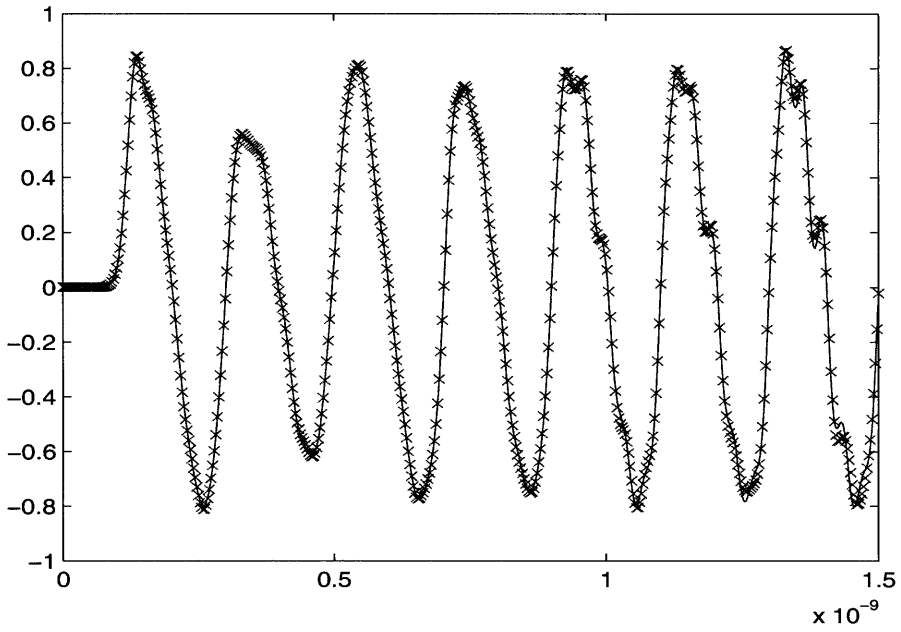


Figure 6-4: Comparison of full nonlinear system output (solid line) with output of reformulated nonlinear descriptor-form large-order PWL model (crosses) for System (6.37).

Table 6.1: Number of unstable linear models generated from 3001 unique linearization points

	Non-descriptor form Equation (6.36)	Descriptor form Equation (6.37)
Large PWL	2927	1391
Reduced PWL	1882	0

more accurate constant-descriptor formulation (6.36) produces many unstable linear models for both large order and reduced order. This result is due to the improved structure of the Jacobian matrices resulting from a reformulation of the system equations into nonlinear descriptor form. Thus, the reformulation allows us to create stable reduced local models without employing the more expensive nonlinear stabilizing projection. In this case the stable reduced-model provided a speedup of about 4 times over the original large system.

6.7.3 Unstructured Analog Circuit

To illustrate the nonlinear left-projection technique proposed in Section 6.5.2 for unstable and unstructured large-order PWL models, and Algorithm 5 for reusing matrix equation solutions, we consider a distributed amplifier circuit shown in Figure 6-5. It is not uncommon for analog circuits, such as this one, to produce PWL models that do not contain symmetric or sign-definite system matrices, making it difficult to guarantee stability for the PWL system through the use of a quadratic Lyapunov function. For this example the transconductances in the small-signal transistor models do not appear symmetrically in the linearized system matrices A_i, making it necessary to utilize the nonlinear left-projection function $U(\hat{x})$, defined in (6.25), to preserve stability.

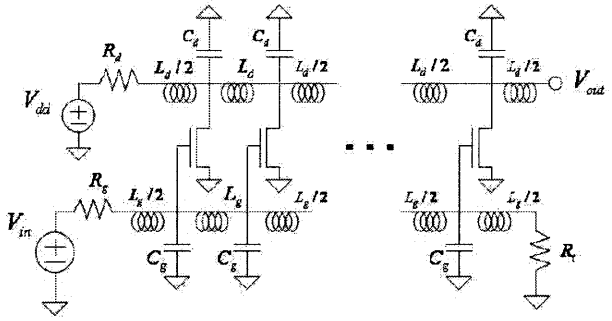


Figure 6-5: Distributed amplifier circuit

A collection of about $10,000$ linear models of original order $N = 106$ were created by training the system with multi-tone inputs of the form

$$u(t) = \alpha_1 \sin(2\pi f_1 t) + \alpha_2 \sin(2\pi f_2 t)$$

while varying the amplitudes α_1, α_2 and the frequencies f_1, f_2, which were near 1GHz. To examine the stability of the local linear models, the maximum real part of the eigenvalues for each large-order linear model are plotted in Figure 6-6 as the solid line. In this figure, a point with a positive value corresponds to an unstable linear model. Out of a total of almost $10,000$ linear models, only 368 are unstable.

From these large linear systems, two different sets of reduced models (with reduced

order $q = 10$) were created: the first using a constant left-projection matrix $U = V$, and the second using the nonlinear left-projection function described in (6.28). Algorithm 5 was used to reduce the number of matrix equation solves required to construct the stabilizing projection matrix. The maximum real part of the resulting local reduced models are also plotted in Figure 6-6 compares the stability of the two resulting sets of reduced-order linear model pairs by plotting the sorted maximum real part of the eigenvalues for each linear model. The dashed line corresponds to the reduced models created with the constant left-projection matrix ($U = V$). Note that for this model about two-thirds of the $10,000$ reduced linear models are unstable. The reduced models created with the proposed nonlinear left-projection are represented by the dotted line. Although it is not easy to see in the figure, for this reduced model there are precisely 368 unstable models, which correspond exactly to the original 368 unstable large-order linear models.

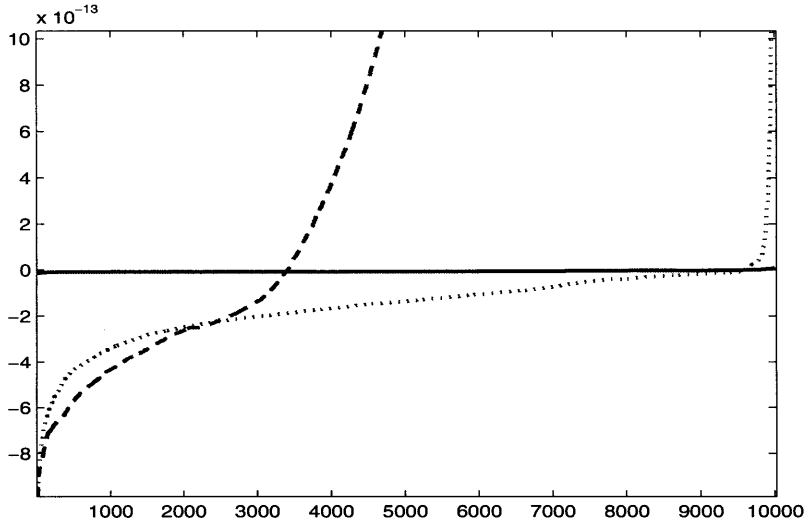


Figure 6-6: Maximum real part of the eigenvalues of the linear models for the large system (solid line), containing only 368 unstable linear models, the reduced models created with the proposed stabilizing nonlinear left-projection function (dotted line), also resulting in 368 unstable models, and the reduced system created with the traditional constant left-projection matrix (dashed line), resulting in over $6,000$ unstable local models.

The stabilizing projection matrices were computed using the procedure described in Algorithm 5, which solved approximately $4,000$ matrix equations to generate the approximately $10,000$ local projection matrices, all of which took less than 5 minutes.

6.7.4 Unstructured Microelectromechanical System

To illustrate the full stabilizing nonlinear model reduction algorithm from Section 6.6.2, we consider a nonlinear descriptor system that produces unstructured Jacobian pairs (E_i, A_i) that are not all stabilizable by a constant left-projection. This example is a micromachined switch MEMS device, first described in Section 5.3.2.

After discretizing the device into m sections lengthwise and n sections widthwise, this model can be written in the form

$$Q(x)\dot{x} = f(x) + b(x)u, \tag{6.38}$$

where the state variable $x \in \mathbb{R}^{mn+2m}$ is chosen to contain the vertical positions of the beam, the pressure beneath the beam, and a quantity related to the rate of change in pressure, all on the discretized grid. A detailed description of these functions can be found in [77].

To test Algorithm 6 from Section 6.6.2, the nonlinear descriptor system (6.38) was trained with a series inputs of the form

$$u(t) = \left(\alpha_1 \sin\left(2\pi f_1 t\right) + \alpha_2 \sin\left(2\pi f_2 t\right)\right)^2 \tag{6.39}$$

with frequencies near 30MHz to obtain a set of κ linear models, and then a right projection matrix V was constructed with a moment matching approach. From this point, two separate reduced models were created – one using the traditional TPWL projection technique with a constant left-projection matrix ($U = V$), referred to as the TPWL-ROM, and one generated using Algorithm 6. The original large-order system has order $N = 360$, while both reduced models have order $q = 20$ and are comprised of approximately 1100 local models created from the same set of linearization points. For this example the stabilizing projection matrices were created using Algorithm 5, which solved 375 matrix equations for the approximately 1100 local models. The entire reduction process was completed in under 15 minutes.

Figure 6-7 plots the maximum real part of the eigenvalues of the linear models for each large-order linearized model and the two reduced models. Despite most of the large

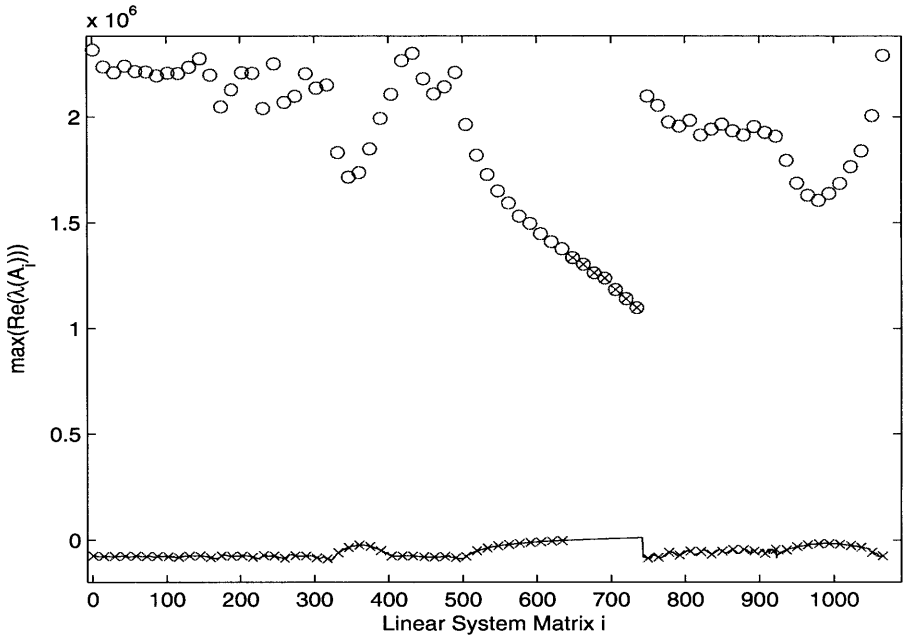


Figure 6-7: Maximum real part of the eigenvalues for individual linear models comprising the large-order PWL model (solid line) and two reduced-order PWL models. The crosses correspond to the reduced models created with the proposed projection method in (6.28), while the circles correspond to reduced models created with the traditional constant left-projection matrix.

matrix pairs (E_m, A_m) being stable (represented by the solid line), Figure 6-7 shows that in every case the reduced models created with the constant left-projection matrix (circles) are unstable. The models created from the nonlinear left-projection (crosses), however, preserve stability in the local models in all cases where the original models were stable.

The two reduced order models were then simulated with a set of inputs of the form (6.39) with different frequency and amplitude from the training inputs from which the linearized models were created. Figure 6-8 plots the output of the full nonlinear system and the two reduced models for several different inputs. The output of the PWL-ROM created with the traditional constant left-projection matrix (dotted line with circles) grows unboundedly because the reduced model is unstable, while the PWL-ROM created with Algorithm 6 (crosses) is both accurate and stable. Additionally, the stable ROM simulated approximately 25 times faster than the full nonlinear model.

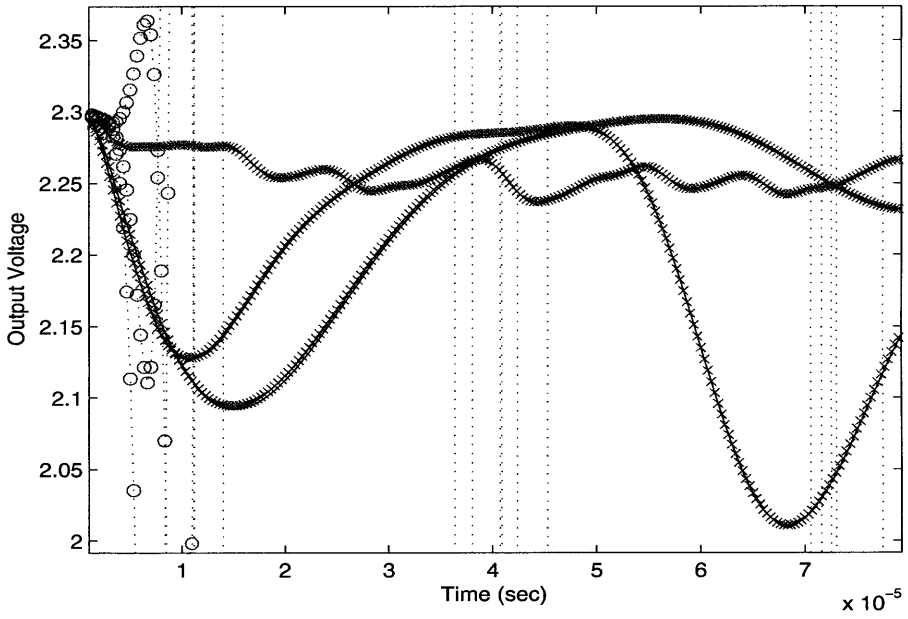


Figure 6-8: Output responses of the full nonlinear system and two reduced models to three different inputs. The full nonlinear system (solid line) and the PWL reduced model created by Algorithm 6 (crosses matching the solid line underneath) are both stable, while the PWL model created using the traditional constant left-projection matrix (dotted line with circles) is unstable.

Chapter 7

A System Identification Approach to Stable Model Reduction

7.1 Introduction

One problem with the previously presented projection approaches in Chapters 4, 5, and 6 is that they rely on reducing a large system of equations, and therefore require precise knowledge about the system being modeled. This requires knowing not only the schematic of the circuit, which is typically readily available, but also the exceedingly complicated transistor models. Transistor models typically consist of a collection of empirical formulas with hundreds of special cases, and hundreds of fitted parameters, which are not as easily available.

System identification (SYSID) provides an alternative approach to model reduction. Such techniques are capable of generating compact models using only input-output (or possibly input-output-state) data, thus eliminating the need for a large dynamical system of equations to be reduced. Data samples for identification can be generated for instance from simulation of a circuit schematic using a commercial simulator, or from measurements performed on a fabricated device.

In this chapter we propose a new alternative SYSID method for compact modeling of nonlinear circuit blocks and MEM components. Our approach is based on optimizing system coefficients to match given data while simultaneously enforcing stability. What

distinguishes our SYSID method from existing "reduction" approaches is the ability to explicitly preserve the properties of nonlinear systems, such as stability, while controlling model accuracy. The efficiency issues encountered by past SYSID techniques are addressed in our approach by adopting recently developed semidefinite programming techniques for nonlinear system analysis [56]. Additionally, we have provided Matlab code implementing our approach [12] to aid the reader in implementing our technique.

The remained of this chapter is organized as follows. In Section 7.2 we review robust nonlinear identification using SYSID. In Section 7.3 we develop the theoretical framework for our proposed SYSID approach to compact modeling of nonlinear systems and formulate the identification problem as a semidefinite program. In Section 7.4 we present one approach to solve efficiently the previously derived optimization problem by selecting a polynomial basis and rational model formulation, resulting in a Sum of Squares (SOS) problem. In Section 7.6 we present two approaches for reducing the complexity of the optimization problem and the complexity of the resulting model through the use of state projection and reduced basis identification. Finally, in Section 7.7 we show the effectiveness of the proposed approach in modeling practical circuit blocks, such as low noise amplifiers, power amplifiers, and MEM devices. The proposed approach is also compared to several existing SYSID techniques.

7.2 Review of Robust Nonlinear Identification

In order to capture both the nonlinear effects and memory effects in our models, we consider implicit sampled dynamical systems of the form

$$\begin{aligned} F(v[t], \ldots, v[t-m], u[t], \ldots, u[t-k]) &= 0 \\ G(y[t], v[t]) &= 0 \end{aligned} \tag{7.1}$$

where $v[t] \in \mathbb{R}^N$ is a vector of internal variables at time t, $u[t] \in \mathbb{R}^{N_u}$ is a vector of inputs, $y[t] \in \mathbb{R}^{N_y}$ is a vector of outputs, $F \in \mathbb{R}^N$ is a dynamical relation between the internal state and the input, and $G \in \mathbb{R}^{N_y}$ is a static relationship between the internal state and the

output. We additionally require that the functions F and G are such that the corresponding System (7.1) is well-posed. In the event that state data $v[t]$ is not available, we may identify input-output models by selecting $v[t] = y[t]$ and defining $G(y, v) = y[t] - v[t]$. For example, the explicit input-output system

$$y[t] = \left(\alpha y[t-1] + \beta y[t-1]^2 + u[t-1]\right)^3$$

can be rewritten as an implicit model with internal states in the form of (7.1) as

$$F = v[t] - \alpha v[t-1] - \beta v[t-1]^2 - u[t-1] = 0$$
$$G = y[t] - v[t]^3 = 0.$$

The following development is focused primarily on the generation of compact sampled discrete time (DT) models, and is then extended to the continuous time (CT) case in Section 7.3.3. For the remainder of this chapter we shall use the following compact notation:

$$V = [v_0, \ldots, v_m], \qquad U = [u_0, \ldots, u_k], \tag{7.2}$$

where $v_0, \ldots, v_m$ and $u_0, \ldots, u_k$ are arbitrary variables (not necessarily inputs and outputs satisfying (7.1)),

$$V_+ = [v_0, \ldots, v_{m-1}], \qquad V_- = [v_1, \ldots, v_m], \tag{7.3}$$

where V_+ contains the first m components of V and V_- contains the last m components of V,

$$V[t] = [v[t], \ldots, v[t-m]], \qquad U[t] = [u[t], \ldots, u[t-k]], \tag{7.4}$$

where $v[t]$ is the internal state of the identified model (7.1) in response to past inputs $U[t]$ and initial conditions $v[t-1], \ldots, v[t-m]$, i.e. $F(V[t], U[t]) = 0$, and

$$\tilde{V}[t] = [\tilde{v}[t], \ldots, \tilde{v}[t-m]], \qquad \tilde{U}[t] = [\tilde{u}[t], \ldots, \tilde{u}[t-k]], \tag{7.5}$$

where $\tilde{v}[t]$ are training data state samples in response to training inputs $\tilde{u}[t]$. Similarly, y shall represent an arbitrary variable, $y[t]$ is the solution to $G(y[t], v[t]) = 0$, and $\tilde{y}[t]$ is a given training data output sample.

Ideally, SYSID aims to minimize the output error between a given set of training data $\chi = \{\tilde{U}, \tilde{V}, \tilde{y}\}$ and the identified model.

Definition 7.2.1. *Given a sequence of inputs $\tilde{u}[0], \ldots, \tilde{u}[T]$, the corresponding states $\tilde{v}[0], \ldots, \tilde{v}[T]$, and outputs $\tilde{y}[0], \ldots, \tilde{y}[T]$, the **output error** of an identified model is defined as*

$$E(F, G, \mathcal{X}) = \sum_t |y[t] - \tilde{y}[t]|^2, \tag{7.6}$$

where $y[t]$ are solutions to the identified model in response to training data inputs and initial conditions $\tilde{v}[t-1], \ldots, \tilde{v}[t-m]$, and $\mathcal{X}$ represents the training data set containing all given $\tilde{u}[t], \tilde{v}[t], \tilde{y}[t]$ pairs.

In general, minimization of the true output error is computationally extremely difficult as it is a highly non-convex problem. Most approaches suggested by the classical literature in system identification [50] instead attempt to minimize the overall "equation error".

Definition 7.2.2. *The **equation error** is defined as the sum of squared mismatches obtained from evaluating the identified model* (7.1) *on all of the training data samples $(\tilde{u}[t], \tilde{v}[t], \tilde{y}[t]) \in \mathcal{X}$*

$$\tilde{E}(F, G, \mathcal{X}) = \sum_t |F(\tilde{V}[t], \tilde{U}[t])|^2 + |G(\tilde{y}[t], \tilde{v}[t])|^2 \tag{7.7}$$

It is, however, misleading to assume that a small equation error always leads to a small output error. It is possible to identify unstable models whose system equations are satisfied

accurately by the given data, resulting in small equation error, but produce unstable outputs during simulation, resulting in large output error.

Definition 7.2.3. *System* (7.1) *is* ***incrementally stable*** *if it is well-posed and, given any two sets of initial conditions* $\bar{v}[t_0-1],\ldots,\bar{v}[t_0-m]$ *and* $\hat{v}[t_0-1],\ldots,\hat{v}[t_0-m]$*, the resulting two solutions to* (7.1) *in response to the same input* u *satisfy*

$$\sum_{t=t_0}^{\infty} \|\bar{y}[t]-\hat{y}[t]\|^2 < \infty \tag{7.8}$$

for all initial conditions and inputs.

As was the case for continuous time systems, dissipativity in the discrete time case can also be proven through the use of storage functions [103]. System (7.1) is said to be dissipative with respect to the supply rate $\sigma(u,v,y)$ if there exists a storage function $L(v) \geq 0$ such that

$$L(V_+) \leq L(V_-) + \sigma(U,V,y) \tag{7.9}$$

for all V, U, y satisfying (7.1). It has been shown in [56] that (7.1) (assuming $G = y - v_0$, i.e. an input-output system) is incrementally stable if the following dissipation constraint is satisfied

$$(v_0-\hat{v}_0)^T\Big(F(V,U)-F(\hat{V},U)\Big) - |v_0-\hat{v}_0|^2 + L(V_-,\hat{V}_-) - L(V_+,\hat{V}_+) \geq 0 \tag{7.10}$$

for all V, U, and all $\hat{V} = [\hat{v}_0,\ldots,\hat{v}_m]$, where L is a non-negative storage function such that $L(V_+,V_+) = 0$. Note that when V, U and $\hat{V}, U$ satisfy (7.1), dissipation constraint (7.10) simplifies to constraint (7.9) with $\sigma = -|v_0-\hat{v}_0|$, which in turn implies (7.8).

7.3 SYSID Formulation

In this section we present the theoretical development for our modeling framework.

7.3.1 Incremental Stability and Robustness

Minimization of the exact output error by enforcing dissipation constraint (7.10), as described in section 7.2, is a computationally difficult problem and typically yields overly conservative upper bounds. Therefore we we consider instead a reasonable alternative measure of output error, referred to as the "linearized output error". First, we define linearizations of (7.1) around y, V, U as

$$\begin{aligned} \bar{F}(V, U, \Delta) &= F(V, U) + F_v(V, U)\Delta = 0 \\ \bar{G}(y, v_0, \delta_0, \xi) &= G(y, v_0) + G_v(y, v_0)\delta_0 + G_y(y, v_0)\xi = 0, \end{aligned} \tag{7.11}$$

where

$$\Delta = \begin{bmatrix} \delta_0 \\ \vdots \\ \delta_m \end{bmatrix}, \qquad \Delta[t] = \begin{bmatrix} \delta[t] \\ \vdots \\ \delta[t-m] \end{bmatrix}$$

$$F_v = \left[\frac{\partial F}{\partial v_0}, \ldots, \frac{\partial F}{\partial v_m}\right], \qquad G_v = \frac{\partial G}{\partial v_0}, \qquad G_y = \frac{\partial G}{\partial y},$$

with $\delta_0 \in \mathbb{R}^N$ and $\xi \in \mathbb{R}^{N_y}$.

Definition 7.3.1. *The **linearized output error** of identified model* (7.1) *is defined as*

$$S(F, G, \mathcal{X}) = \sum_t |\xi[t]|^2,$$

where $\xi[t]$ are solutions to

$$\bar{F}(\tilde{V}[t], \tilde{U}[t], \Delta[t]) = 0, \qquad \bar{G}(\tilde{y}[t], \tilde{v}[t], \delta[t], \xi[t]) = 0, \tag{7.12}$$

in response to the zero initial condition $\delta[t-1], \ldots, \delta[t-m] = 0$ when evaluated on the training data.

In the case of linear systems, the true output error is exactly equal to the linearized

output error.

In order to prove incremental stability of system (7.1), it is sufficient to show that linearizations of (7.1), as defined in (7.11), around all possible U, V, y satisfying (7.1) are stable, as was proposed in [52]. This can be proven with the following dissipation inequality

$$h(V_+, \Delta_+) \leq h(V_-, \Delta_-) - |\xi|^2 - \epsilon|\Delta|^2 + 2\delta_0^T F_v(V, U)\Delta \\ + 2\xi^T(G_v(y, v_0)\delta_0 + G_y(y, v_0)\xi), \quad \forall\, y, V, U, \Delta, \xi, \quad (7.13)$$

where h is a storage function, defined as

$$h(V_+, \Delta_+) = \Delta_+^T H(V_+)\Delta_+ \quad (7.14)$$
$$h(V_-, \Delta_-) = \Delta_-^T H(V_-)\Delta_-,$$

and $\epsilon > 0$. Since (7.11) is linear in Δ, it is sufficient to consider storage functions that are quadratic in Δ [52]. Note that for U, V, y, Δ, ξ satisfying (7.1) and (7.11), constraint (7.13) simplifies to (7.9) with supply rate $\sigma = -|\xi|^2 - \epsilon|\Delta|^2$. Inequality (7.13) can be thought of as a linearized version of inequality (7.10), and is less restrictive because although a stable system satisfying (7.10) also satisfies (7.13), there are many stable systems satisfying (7.13) that do not satisfy (7.10).

Definition 7.3.2. *The **robust equation error**, $\hat{r}$, of System* (7.1) *over training data set $\mathcal{X}$ is defined as*

$$\hat{r}(F, G, H, \mathcal{X}) = \sum_t r(\tilde{y}[t], \tilde{V}[t], \tilde{U}[t])$$

where

$$r(y, V, U) = \\ \max_{\Delta, \xi} \{h(V_+, \Delta_+) - h(V_-, \Delta_-) - 2\delta_0^T \bar{F}(V, U, \Delta) - 2\xi^T \bar{G}(y, v_0, \delta_0, \xi) + |\xi|^2\}. \quad (7.15)$$

The robust equation error serves as an upper bound for the linearized output error.

Theorem 7.3.1. *If there exists a positive semidefinite function* $H : \mathbf{R}^m \mapsto \mathbf{R}^{m\times m}$*, positive scalars* $\epsilon, \epsilon_1, \epsilon_2 > 0$ *such that* $\epsilon_1 I < H < \epsilon_2 I$ *and* (7.13) *is satisfied for all* $\tilde{y}, \tilde{V}, \tilde{U} \in \mathcal{X}$*, and for all possible* Δ, ξ*, then System* (7.1) *is locally incrementally stable and the linearized output error on the training set is bounded from above by the robust equation error*

$$S(F, G, \mathcal{X}) \leq \hat{r}(F, G, H, \mathcal{X}).$$

If, in addition, H *is continuously differentiable and* (7.13) *is satisfied for all* y, V, U*, then System* (7.1) *is also globally incrementally stable.*

Proof. Incremental stability is implied by (7.13) using a standard proof following the principles of [52]. It follows from (7.15) that

$$|\xi|^2 \leq r(y, V, U) + 2\delta_0^T \bar{F}(V, U, \Delta) + 2\xi^T \bar{G}(y, v_0, \delta_0, \xi) + h(V_-, \Delta_-) - h(V_+, \Delta_+) \quad (7.16)$$

is satisfied for all y, V, U, Δ, ξ. To obtain the linearized output error, we sum (7.16) over all training data samples $\tilde{y}[t], \tilde{V}[t], \tilde{U}[t]$ and incremental variables $\Delta[t], \xi[t]$ satisfying (7.12), resulting in

$$\begin{aligned} S(\mathcal{X}, F, G) &= \sum_t |\xi[t]|^2 \qquad (7.17)\\ &\leq \sum_t [r(\tilde{y}[t], \tilde{V}[t], \tilde{U}[t]) + h(\tilde{V}_-[t], \Delta_-[t]) - h(\tilde{V}_+[t], \Delta_+[t])] \\ &\leq \sum_t r(\tilde{y}[t], \tilde{V}[t], \tilde{U}[t]) = \hat{r}(F, G, H, \mathcal{X}). \end{aligned}$$

Here we have also used the fact that

$$\sum_{t=0}^{T} \Big[h(\tilde{V}_-[t]) - h(\tilde{V}_+[t])\Big] = -h(\tilde{V}_+[T]) \leq 0$$

by definition of h and by the zero initial condition of Δ. Note that finiteness of $\hat{r}$ is guar-

anteed by (7.13). □

In summary, for a given model F, G, if there exists a storage function h as defined in (7.14) that satisfies (7.13), then system (7.1) is incrementally stable. Furthermore, for such F, G, h, the robust equation error serves as an upper bound for the linearized output error over the training data set, as shown in (7.17).

7.3.2 Identification Procedure

The proposed system identification algorithm is based on minimization (with respect to F, G, H, and r) of the *linearized output error* upper bound, r, over the training data set $\mathcal{X}$ subject to a dissipation constraint

$$\begin{aligned} &\min_{r,F,G,H} \sum_t r_t \qquad \text{subject to} \\ &r_t + 2\delta_0^T \bar{F}_t(\Delta) + 2\xi^T \bar{G}_t(\delta_0, \xi) - |\xi|^2 \\ &+ h_{t-1}(\Delta_-) - h_t(\Delta_+) \geq 0, \quad \forall\, t, \Delta, \xi. \end{aligned} \tag{7.18}$$

where we define $r_t = r(\tilde{y}[t], \tilde{V}[t], \tilde{U}[t])$, $\bar{F}_t(\Delta) = \bar{F}(\tilde{V}[t], \tilde{U}[t], \Delta)$, $\bar{G}_t = \bar{G}(\tilde{y}[t], \tilde{v}[t], \delta_0, \xi)$, $h_{t-1}(\Delta_-) = h(\tilde{V}_-, \Delta_-)$, and $h_t(\Delta_+) = h(\tilde{V}_-, \Delta_-)$. In this formulation we simultaneously enforce accuracy by minimizing the linearized output error upper bound at the training data samples, and also enforce local incremental stability at each training sample through the constraint.

By construction, the robustness constraint is jointly *convex* with respect to the unknown functions F, G, H, r, and is a quadratic form in the incremental variables Δ, ξ. If the unknown functions are chosen among linear combinations of a finite set of basis functions Φ

$$\begin{aligned} F &= \sum_{j \in \mathbb{N}_f} \alpha_j^F \phi_j^F(V, U), & G &= \sum_{j \in \mathbb{N}_g} \alpha_j^G \phi_j^G(y, v_0) \\ H &= \sum_{j \in \mathbb{N}_h} \alpha_j^H \phi_j^H(V), & r &= \sum_{j \in \mathbb{N}_r} \alpha_j^r \phi_j^r(y, V, U), \end{aligned} \tag{7.19}$$

where $\phi^F, \phi^G, \phi^H, \phi^r \in \Phi$, then $\alpha^F, \alpha^G, \alpha^H, \alpha^r$ become the free variables and the optimization problem becomes a semidefinite program (SDP). Additional details on semidefinite programming are given in Section 7.6.

In order to obtain global incremental stability, it is necessary to additionally enforce constraint (7.13) globally for all y, V, U and to ensure that the storage function H is smooth with respect to all arguments. In this case, the complexity of the optimization problem depends heavily on the choice of basis functions ϕ for the unknown functions F, G, H, r. The basis must be chosen carefully to ensure that the inequalities in problem (7.18) can be easily verified numerically, and that feasible solutions exist. In Section 7.4 we describe one possible choice for the basis functions Φ that results in an optimization problem that can be efficiently solved.

7.3.3 Extension to Continuous Time Models

In this paper we focus mainly on generating DT models for many typical circuit blocks in the signaling path that are also usable in high level system simulation and design, using for instance Cadence AMS or Verilog A. In addition, it is possible to extend the previously developed dissipation-based identification approach to generate CT systems for greater compatibility with lower level circuit simulators. In this case there are however additional constraints on the choice of F to ensure that the system is uniquely solvable. For instance, F should not possess nonlinear dependence on derivatives of the input, otherwise the system may not be well-posed. Additionally, there are strong constraints on the relationship between the function F and the storage function H in order to guarantee existence of solutions to the optimization problem. To avoid excessive technicalities, we consider here only CT systems described in state-space form

$$\begin{aligned} F\left(\dot{v}(t), v(t), u(t)\right) &= 0 \\ G\left(y(t), v(t)\right) &= 0. \end{aligned} \tag{7.20}$$

along with constant PSD storage function matrices $H(v) = H$. As in the DT case, we define a robust dissipation inequality

$$\frac{\partial h(\Delta)}{\partial t} \leq 2\delta^T F_v(v,u)\Delta + 2\xi^T G_y(y,v)\xi + 2\xi^T G_v(y,v)\delta - |\xi|^2 - |\delta|^2, \tag{7.21}$$

where $h(\Delta) = \Delta^T H \Delta$, such that System (7.20) is incrementally stable and the linearized output error is bounded from above by the robust equation error if there exists a storage function matrix H such that (7.21) holds for all y, v, u, Δ, ξ. Constraint (7.21) can then be used to formulate an optimization problem similar to (7.18). Results for CT modeling using this approach are presented in Section 7.7.2.

7.3.4 Identification of MIMO Models

The previously derived identification procedure is capable of identifying models with multiple inputs, multiple states, and multiple outputs. Multi-port models can also be used to capture loading effects. If one of the ports is connected to a load, then varying the load will produce different input-output data for that port, which can then be used for training the model in order to capture loading effects. Our resulting multi-port model can then be described for instance in Verilog-A and connected with other circuit blocks inside a commercial simulator. Increasing the number of inputs and outputs increases the complexity of the optimization problem by adding more unknown parameters, and when increasing the number of states in the model (or the number of outputs in the case of an input-output model), it may also be necessary to increase the complexity of the storage function to ensure that the system is dissipative with respect to all states. In Section 7.7 we present results for systems with multiple inputs (section 7.7.5), multiple states (section 7.7.2), and multiple outputs (section 7.7.4).

7.3.5 Extension to Parameterized Models

Our approach can easily be extended to identify models parameterized by, for instance, device parameters or geometrical parameters. This is achieved by selecting the basis func-

tions for F, G, H, r to possess dependence on design parameters Z, e.g. $\phi^F = \phi^F(V, U, Z)$, where $Z = [z_1, \ldots, z_p]$ is a vector of parameters. Conceptually this is equivalent to treating the parameters as constant inputs with no memory. Results using this parametrization approach are presented in Section 7.7.3.

7.4 Identification of Rational Models in a Polynomial Basis

In this section we present one possible choice of basis functions for representing the nonlinear functions in optimization problem (7.18), and we also discuss the existence of solutions to the optimization problem.

7.4.1 Polynomial Basis

The complexity of optimization problem (7.18) with global stability constraint (7.13) depends on the choice of basis functions for the nonlinear function, robustness measure, and storage function. One possible choice resulting in a convenient formulation is a polynomial basis.

If we constrain F, G, H, r to be polynomial functions of the internal variables and inputs, i.e. define ϕ from Section 7.3.2 as

$$\phi(y[t], V[t], U[t]) = \prod_{i,j,k} v[t-\tau_i]^{p_i} u[t-\tau_j]^{p_j} y[t]^{p_k}, \tag{7.22}$$

then we can formulate optimization problem (7.18) as a sum-of-squares (SOS) problem. Proving global positivity of a multivariate polynomial is in general a hard problem (i.e. computationally challenging), however SOS provides an efficient convex relaxation for such problem. Guaranteeing global positivity in the stability constraints is transformed to the task of solving for a positive semi-definite (PSD) symmetric matrix $S = S^T$ such that

optimization problem (7.18) along with constraint (7.13) is expressed as

$$\min_{r,F,G,H} \sum_t r_t \qquad \text{subject to} \tag{7.23}$$

$$r_t + 2\delta_0^T \bar{F}_t(\Delta) + 2\xi^T \bar{G}_t(\delta_0, \xi) - |\xi|^2 + h_{t-1}(\Delta_-) - h_t(\Delta_+) \geq 0, \qquad \forall\, t, \Delta, \xi$$

$$h(V_-, \Delta_-) - h(V_+, \Delta_+) + 2\delta_0^T F_v(V, U)\Delta + 2\xi^T G_v(y, v_0)\delta_0$$

$$+ 2\xi^T G_y(y, v_0)\xi - |\xi|^2 = \Psi^T S \Psi, \qquad \forall\, y, V, U, \Delta, \xi,$$

$$S = S^T \geq 0.$$

Note that the second constraint must be satisfied for all possible y, V, U, while the first constraint only must be satisfied for the training samples $\tilde{y}[t], \tilde{V}[t], \tilde{U}[t]$. Here Ψ is a vector of basis functions ψ such that all basis functions ϕ can be represented by the product $\Psi^T S \Psi$. That is, for every ϕ_i there exist ψ_j and ψ_k such that $\phi_i \propto \psi_j \psi_k$. Conceptually, the vector Ψ must contain the monomial terms present in the 'square root' of the dissipation constraint, and for nonlinear systems these entries can be automatically selected from the Newton Polytope of the robustness constraint. See [64, 92, 80] for details on SOS programming and the Newton Polytope, and see [57] or [12] for our software implementation.

It is important to note that although we are using a polynomial basis, we are not identifying polynomial models. Specifically, the implicit representation of the nonlinear system (7.1) allows us to identify, for instance, rational models as described in the following section. In this way we can represent highly nonlinear models in a much more compact form than is possible using traditional polynomial models such as Volterra expansions.

7.4.2 Rational Model Description

In general, the identified implicit nonlinear model (7.1) can be extremely expensive to simulate. To ensure that the resulting DT model can be simulated in an efficient manner, we consider only models that are linear in the unknowns $v[t]$. For example, consider the

model

$$F(V[t], U[t]) = Q(V_-[t], U[t])v[t] - p(V_-[t], U[t]) = 0$$
$$G(y[t], v[t]) = g_q(v[t])y[t] - g_p(v[t]) = 0, \tag{7.24}$$

where $Q \in \mathbb{R}^{N \times N}$ is a matrix of nonlinear functions, $p \in \mathbb{R}^N$ is a vector of nonlinear functions, and $V_-[t] = [v[t-1], \ldots, v[t-m]]$. Although F is defined implicitly, the system is linear in the unknowns, making the simulation of this discrete time system equivalent to linear system solves when all previous values of the state, $v[t-1], \ldots, v[t-m]$, and input, $u[t], \ldots, u[t-k]$, are known

$$v[t] = Q(V_-[t], U[t])^{-1} p(V_-[t], U[t]). \tag{7.25}$$

The presence of the nonlinear matrix function $Q(V_-, U)$ is extremely important, as it allows the model to capture nonlinear effects that are significantly stronger than those that would be captured by considering the case where $Q = I$, without significantly increasing the complexity of the optimization problem and of simulation.

7.4.3 Existence of Solutions

Given a nonlinear function F, the existence of solutions to (7.28) depends on the ability of the storage function h to certify stability for that particular nonlinear function. As a result, the basis functions describing H are dependent upon the basis functions describing F.

For models without feedback, such as the Volterra model

$$y_t = p(u_t, u_{t-1}, \ldots, u_{t-k}), \tag{7.26}$$

a storage function is not required to prove stability, and solutions always exist. One implication of this is that Volterra models are a strict subset of the stable models identifiable by our approach.

When feedback is present in the model, for certain functions F, storage functions are

available to prove stability. For example, for a linear system, it is always possible to certify stability with a constant matrix H. As a result, if the polynomial basis contains linear terms, then there always exists a solution to (7.28) described by a linear function F and constant matrix H. Additionally, since the storage function and stability of the resulting model do not depend strictly on the inputs u to the system, a constant matrix H can certify stability for a system that is linear in v and highly nonlinear in u.

It is also possible to guarantee that solutions of (7.28) that are nonlinear in v do exist. Let us define $f = f(V,U)$ to be an unknown nonlinear function, and $d = d(V,U) > 0$ to be a strictly positive scalar polynomial function such that $F = \frac{f}{d}$. Note that any solution of $F(V,U) = 0$ also satisfies $f(V,U) = 0$. If we then set $f = d(V,U)f_L(V,U)$ where $f_L(V,U)$ is a function linear in v, then $F = f_L$ is linear and thus can be certified stable by a quadratic storage function, and therefore $f(V,U) = d(V,U)f_L(V,U)$ defines a stable nonlinear model. Further, because the set of multivariate polynomials that are sums-of-squares in a strict sense is open, any nonlinear function $f(V,U)$ containing polynomial terms of order less than or equal to those in the product $d(V,U)f_L(V,U)$ can be certified stable by a constant storage function matrix H. Note that with this definition of F, $\bar{F}$ in (7.28) becomes

$$\bar{F} = \frac{f}{d} + \frac{\frac{\partial f}{\partial v}d - f\frac{\partial d}{\partial v}}{d^2}\Delta,$$

and if the constraints in (7.28) is multiplied through by d^2, the resulting problem is still an SOS problem.

7.5 Efficient Identification through Reduction of States and Basis

In this section we present several techniques for reducing the complexity of the optimization problem by fitting to a reduced set of states and identifying with a reduced set of basis vectors.

7.5.1 Complexity Analysis

The computational complexity of solving SOS problem (7.28) depends on the number of unknown decision parameters describing the nonlinear functions F, G, H, r. The number of decision parameters in F is roughly $N\binom{N_u k + Nm + \rho - 1}{\rho}$, where N is the size of the vector of unknowns $v[t]$, m is the number of delays for each element of $v[t]$, N_u is the number of inputs, k is the number of delays for each input, and ρ is the maximum polynomial order in F. For models with a high polynomial degree ρ, large number of states N, or large number of delays, this problem quickly becomes intractable.

Once the model has been identified, the cost of simulation becomes the cost of simulating an N^{th} order polynomial dynamical system. For a DT model or a CT model being solved with an explicit integration scheme, simulation requires evaluating the ρ^{th} order polynomial functions p and Q and then solving a N^{th} order linear system for every time step. When integrating a CT system with an implicit scheme, using for instance Newton's method, simulation will require several function evaluations and system solves at each step.

For both the identification and simulation of the model, the computational cost depends heavily on the number of states in the model and the degree of the polynomials. However, the above complexity analysis assumes that all states are equally important for identification and that all possible combinations of polynomial products of basis vectors are present in the model, and this is not true in general. Thus, two possible methods for reducing the complexity of solving the optimization problem and simulating the resulting model, while still allowing us to fit highly nonlinear functions, are to fit to a reduced set of states, and to fit using a reduced set of basis vectors.

7.5.2 Identification of States through Projection

In the event where data is available for a large number of internal states (i.e. N is large), it is possible to identify a low-order space in which the system states are well-approximated, and fit by projection to a set of reduced vectors, $\hat{v}[t] \in \mathbb{R}^{\hat{N}}$, where $\hat{N} < N$.

For example, given a collection of training samples, $X = [\tilde{v}[t_1], \tilde{v}[t_2], \ldots, \tilde{v}[t_T]] \in \mathbb{R}^{N \times T}$, it is possible to identify a low-order basis $\Theta \in \mathbb{R}^{N \times \hat{N}}$ such that $X \approx \Theta \hat{X}$, where

$\hat{X} = \Theta^T X$ is a projection of the training data onto the reduced space. The projection matrix Θ can be computed using any standard projection technique, such as POD [53, 102] using training data X. The system identification is then performed using the reduced-order training data set $\hat{X}$, resulting in a model with $\hat{N}$ states.

This approach is similar to traditional model reduction techniques utilizing projection in the sense that we approximate the solution in a low-dimensional space spanned by Θ. However, the key difference of our approach is that instead of constructing the reduced model by projecting the system equations explicitly, we instead identify the reduced equations through an optimization procedure in order to minimize the reduced model output error over a given set of training data. Numerical results obtained from this projection approach are presented in Section 7.7.2.

7.5.3 Identification of Polynomial Basis through Fitting

In addition to the number of state variables, the cost of identifying and simulating the models also depends on the number of delays (memory) of the system. To decrease this cost without reducing the polynomial order of the desired function, it is useful to consider only important polynomial basis terms for identification. Let $\tilde{\Phi}_{[0,n]}$ denote a nominal set of basis functions comprised of variables u, v with up to n delays. Important basis terms $\hat{\phi}$ may be selected as linear combinations of the nominal basis components:

$$\hat{\phi}_i = \sum_j \beta_{j,i} \tilde{\phi}_j, \qquad \tilde{\phi}_j \in \tilde{\Phi}_{[0,n]}. \tag{7.27}$$

The coefficients $\beta_{j,i}$ can be identified by fitting a linear model with memory n to the training data with basis $\tilde{\Phi}$.

For example, suppose the nominal basis functions are selected to be input samples, i.e. $\tilde{\Phi}_{[0,k]} = [u[t], \ldots, u[t-k]]$. The training data $(\tilde{y}[t], \tilde{u}[t])$ can be used to identify a linear model with memory $\hat{k}$ and output $w[t] \approx y[t]$

$$w[t] = \sum_{j=0}^{\hat{k}} b_j u[t-j].$$

This identified linear model now defines a linear transformation of the nominal basis vectors to the reduced basis vector if we define $\beta_{j,1} = b_j$ for the new basis vector set $\hat{\Phi}_{[0,\hat{k}]} = [w[t], \ldots, w[t-\hat{k}]]$ for some $\hat{k} < k$. This new basis vector can then be used for identification of a nonlinear model with low memory. Conceptually, this is equivalent to treating $w[t]$ as an additional input to a new nonlinear model

$$Q(V\_[t], W[t], U[t])v[t] = p(V\_[t], W[t], U[t])$$

as depicted in Figure 7-1.

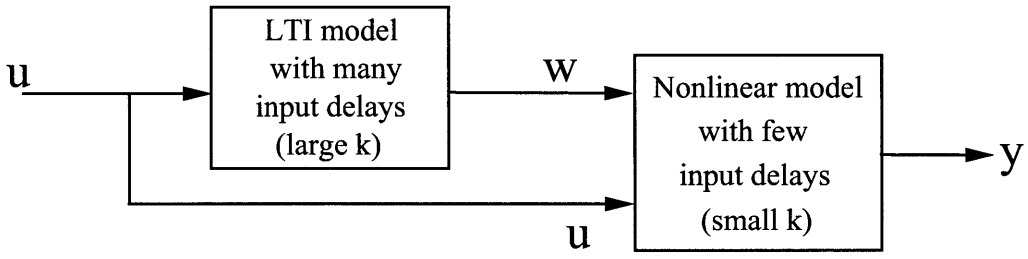


Figure 7-1: Block diagram illustrating one approach to implementing the reduced basis selection technique.

Since the identification of linear systems is cheap, even when m and k are large, this approach can be very useful for reducing the complexity of the final nonlinear model by automatically selecting important combinations of basis vectors. Numerical examples using this reduced basis identification approach are presented in Section 7.7.5.

7.6 Implementation

The optimization problem (7.18) derived in Section 7.3.2, along with global stability constraint (7.13), can be expressed generically as the following

$$\min_{r,F,G,H} \sum_t r_t \qquad \text{subject to} \tag{7.28}$$

$$r_t + 2\delta_0^T \bar{F}_t(\Delta) + 2\xi^T \bar{G}_t(\delta_0, \xi) - |\xi|^2 + h_{t-1}(\Delta_-) - h_t(\Delta_+) \geq 0, \qquad \forall\, t, \Delta, \xi \tag{7.28a}$$

$$h(V_-, \Delta_-) - h(V_+, \Delta_+) + 2\delta_0^T F_v(V, U)\Delta - |\xi|^2 + 2\xi^T G_v(y, v_0)\delta_0 + 2\xi^T G_y(y, v_0)\xi \geq 0, \qquad \forall\, y, V, U, \Delta, \xi. \tag{7.28b}$$

In this section we describe how to formulate (7.28) as a semidefinite program (SDP) when using a polynomial basis, and how to solve the resulting SDP using freely available software.

7.6.1 Implementation as a Semidefinite Program

The benefit of formulating (7.28) as an SDP is that it can be solved efficiently using readily available software routines. Roughly speaking, a semidefinite program is one whose objective function is linear, and whose constraints can be expressed as requiring matrices to be positive semidefinite (PSD).

By construction, constraint (7.28a) is a quadratic form in the variable $\zeta = [1,\ \Delta^T, \xi^T]^T$, and can therefore be expressed as

$$r(y, V, U) + 2\delta_0^T \bar{F}(V, U, \Delta) + 2\xi^T \bar{G}(y, v_0, \delta_0, \xi) - |\xi|^2 + h(V_-, \Delta_-) - h(V_+, \Delta_+) = \zeta^T M(U, V, y)\zeta \tag{7.29}$$

for some symmetric matrix M. Thus, global positivity of (7.29) is satisfied if the matrix M is PSD. In (7.28) we are not requiring $M(U, V, y)$ to be PSD for all U, V, y, but rather only when evaluated at the given training data samples. That is, $M_t = M(\tilde{U}[t], \tilde{V}[t], \tilde{y}[t])$ is PSD for all t.

Algorithm 7 Implementation as SDP using POT

1: Given symbolic functions F, G, H, r defined as in (7.19) and training data set χ
2: Initialize optimization problem `pr`
   `pr=mssprog`
3: Assign free variables
   `pr.free=`$\{\alpha^F, \alpha^G, \alpha^H, \alpha^R\}$
4: **for** t=1:T **do**
5: Compute $M_t = M(\tilde{U}[t], \tilde{V}[t], \tilde{y}[t])$ as defined in (7.29)
6: Assign local robustness constraint (7.28a)
   `pr.PSD=`M_t
7: **end for**
8: Assign global stability constraint (7.28b)
   `pr.SOS=` (7.28b)
9: Call solver to minimize $\sum_t r_t$ subject to given constraints
   `pr.min=`$\sum_t r_t$
10: Output is coefficients $\{\alpha^F, \alpha^G, \alpha^H, \alpha^R\}$

On the other hand, the global stability constraint (7.28b) must be satisfied globally for all U, V, y. This can be achieved by the SOS relaxation described in Section 7.4.1, which transforms constraint (7.28b) into a single semidefinite matrix constraint. While it is easy to construct the $M(U, V, y)$ matrix explicitly, and possible to construct S from Section 7.4.1 by hand, these tasks can be performed automatically by the freely available software POT [57], when given symbolic constraints in the form of (7.28a) and (7.28b).

In Algorithm 7 we outline how optimization problem (7.28) can be defined and solved in Matlab using the freely available software POT [57] and SeDuMi [91]. POT is a 'parser', which takes as input a high-level symbolic description of the optimization problem and reformulates it in such a manner that it can be solved by an optimization 'solver' (in this case, SeDuMi). For additional details and a sample implementation of this approach, see [57, 12].

7.6.2 The Complete Algorithm

Our entire identification process is summarized in Algorithm 8. The first step in identifying a model in the form of (7.24) is to select the number of states (N), the number of state delays (m), the number of input delays (k), the maximum polynomial degree for Q (ρ_Q), the maximum polynomial degree for p (ρ_p), and the maximum polynomial degree for

storage function matrix H (ρ_H). These parameters generally depend on the behavior of the system being modeled, and can be selected either by intuition (based on the system's expected behavior) or through experiment. One approach that we have found to be effective is described below in Section 7.3.5. For CT models, it is often possible to obtain derivatives of states and outputs for step 7 directly from the simulator, as they are typically required internally for simulation. For systems with a large delay between input and output, the reduced basis technique described in Section 7.5.3 should be used at step 10 to reduce the required number of basis functions. Typically we have found that selecting $\tilde{\Phi}$ as containing the past $15 - 20$ input samples can produce good results for such systems. The basis set Φ for the final model can then be selected at step 11 as a small number of delayed samples of the true input u and the delayed input w, as well as delays of the state and output. Finally, when considering only local stability for the identified model, it is only necessary to enforce first (7.28a) constraint in optimization problem (7.28).

7.6.3 Selecting the Model Parameters

For a given set of parameters $N, m, k, \rho_Q, \rho_p, \rho_H$, as defined in 7.6.2, let $\Upsilon = \{N, m, k, \rho_Q, \rho_p\}$ denote the set of all possible models with these parameters, and let $\mathcal{H}$ denote the set of all storage functions of maximum polynomial degree ρ_H. The goal of the identification procedure is to find a stable model in Υ that accurately fits the training data χ and is certified stable by a storage function in $\mathcal{H}$.

It is difficult to accurately determine Υ and $\mathcal{H}$ a priori, but we have found the following procedure to be quite effective. First, we select a set Υ and attempt to fit a model *with no stability constraints*. This can be achieved, for instance, by using a least-squares solve to minimize equation error (which is computationally cheap). Varying Υ through experiment, it is possible to identify a model that accurately fits χ.

Next, it is necessary to determine whether there exists a *stable* model in Υ that is certifiable by $\mathcal{H}$. To determine this, we select ρ_H and solve (7.28) using Algorithm 8 while first enforcing *only local stability* constraint (7.28a) in Algorithm 7. If no accurate locally stable model is found, then ρ_H should be increased. If, for large ρ_H, no accurate stable

Algorithm 8 SOS Identification of Robust Compact Models

1: Generate training data sample set $\mathcal{X} = \{\tilde{U}[t], \tilde{V}[t], \tilde{y}[t]\}$ from simulation or measurement of the original system
2: Select the model parameters m, k, N, ρ corresponding to output order, input order, number of states, and maximum polynomial degree respectively
3: **if** State data is available and N is large **then**
4: Use SVD to identify low-order basis for states, $\Theta \in \mathbb{R}^{N \times \hat{N}}$, $\hat{N} < N$, as described in Section 7.5.2
5: Project data samples: $\tilde{v}[t] \leftarrow \Theta^T \tilde{v}[t]$
6: **end if**
7: Compute delays (for DT) or derivatives (for CT) from data $\mathcal{X}$ based on m, k
8: Select nominal set of basis functions $\tilde{\Phi}_{[0,k]}$
9: **if** Large delay between input and output **then**
10: Identify linear model defining coefficients $\beta_{j,i}$ of new basis functions $\hat{\phi}$ defined in (7.27), as described in Section 7.5.3
11: $\Phi \leftarrow [\tilde{\Phi}_{[0,\hat{k}]}, \hat{\Phi}_{[0,\hat{k}]}]$ for $\hat{k} < k$.
12: **else**
13: $\Phi \leftarrow \tilde{\Phi}_{[0,k]}$
14: **end if**
15: Use Algorithm 7 to solve SOS problem (7.28) for coefficients α_i
16: Define F, G, H, r, as in (7.19), resulting in the model

$$Q(V_-[t], U[t])v[t] = p(V_-[t], U[t]), \qquad g_q(v[t])y[t] = g_p(v[t])$$

certified stable by Theorem 7.3.1 for matrix function H, and with $\sum_t r_t$ serving as a measure of the model's accuracy on the training data.

model is found, then Υ should be increased (i.e. increase any of N, m, k, ρ).

Once an accurate locally stable model is found, then (7.28) should be solved using Algorithm 8, this time also enforcing global stability constraint (7.28b). If no accurate globally stable model is found, then ρ_H and Υ should be increased, as described above. If stability constraint (7.28b) is not enforced, then the robust equation error is not guaranteed to be an upper bound for the linearized output error, meaning that simulation of the resulting model, even over the training data set, could lead to inaccurate results.

7.6.4 Constructing Basis Functions

For a given set of parameters $N, m, k, \rho_Q, \rho_p, \rho_H$, the basis functions for F, G and H can be constructed as defined in (7.22), where $p_i + p_j + p_k \leq \rho$, $\tau_i < m$, and $\tau_j < k$. In

general, ρ_Q should be an even integer to ensure that matrix function Q is always invertible. For the robustness measure r, we typically use a piecewise-constant function, resulting in one unknown parameter for each training data point.

7.7 Examples

7.7.1 Testing Procedure

In this section we present numerical results for several examples of nonlinear systems modeled using our proposed approach. The identified models include a CT model, a DT parameterized model, and a DT SIMO model. For each example, training data was generated from simulations of the full system in response to a series of periodic inputs, using Spectre circuit simulator for the circuit examples and a MATLAB simulator for the MEMS example. The training inputs must be carefully chosen in order to excite all possible behavior of interest in the system, while avoiding driving the system to regions of the space that will not be excited by typical inputs of interest. Attempting to model dynamics not encountered by testing inputs could greatly increase the complexity of the identified model. In order to maximize robustness while minimizing complexity, in our experience, the best approach is to train with inputs having the same form (e.g. sum of sinusoids) as the inputs to be used for testing. In this case the amplitudes, frequencies, and phases of the training inputs may be varied over a wide range containing all possible values to be used for testing. For all examples the models are identified in a polynomial basis with a rational description, as described in Section 7.4.

All of the model generation and simulation times reported were obtained using a desktop PC with a dual core 3.33GHz processor and 4GB of RAM. The SOS problem (7.28) was solved using the Polynomial Optimization Toolbox [57], which uses SeDuMi [91].

7.7.2 MEM Device

In our first example we identify a CT model of a MEMS device (described in Section 5.3.2 to show that our preliminary CT approach from Section 7.3.3 and projection approach from

Section 7.5.2 are feasible.

For this example the training data was generated from inputs of the form

$$u(t) = \left[A_1 \sin(\omega_1 t) + A_2 \sin(\omega_2 t) + A_3 \sin(\omega_3 t)\right]^2, \tag{7.30}$$

where A_i vary between 4 Volts and 7 Volts, and $f_i = \frac{2\pi}{\omega_i}$ vary between 1.5kHz and 240kHz. From this data, we identified a 4th order nonlinear CT model suitable for usage in any ODE integrator and in particular a low level circuit simulator

$$Q_2(v,u)\dot{v} = p_7(v,u), \qquad\qquad y = C^T v$$

where $v \in \mathbb{R}^4$, $Q_2 \in \mathbb{R}^{4\times 4}$ is a matrix of second order polynomials, $p_7 \in \mathbb{R}^4$ is a vector of seventh order polynomials, and $C \in \mathbb{R}^4$ is a constant vector, all identified using the projection technique described in Section 7.5.2 and the reduced basis technique from section 7.5.3, resulting in only 52 parameters in the reduced model. For this model the identification procedure took less than two minutes.

The identified model was tested on an input of the form (7.30) with A_i and f_i different from the training set, and the resulting output is compared to the output of the full nonlinear system in Figure 7-2.

To make the comparison fair, both full and reduced models were simulated using the same MATLAB built in ODE solver. Simulation of the full 400th order nonlinear system for this example required approximately 400 seconds to integrate for $5,000$ time points, while the reduced model was simulated in response to the same input for the same number of time steps in just 10 seconds, resulting in a speedup of about 40 times.

7.7.3 Operational Amplifier

In our second example we identify a parameterized model, using the approach described in Section 7.3.5, for a two-stage operational amplifier. The opamp, shown in Fig. 7-3, is designed with a 90nm predictive model and nominal reference current as 10μA. It has a open-loop DC gain of 260 and unity-gain bandwidth 125MHz. For the parameterized

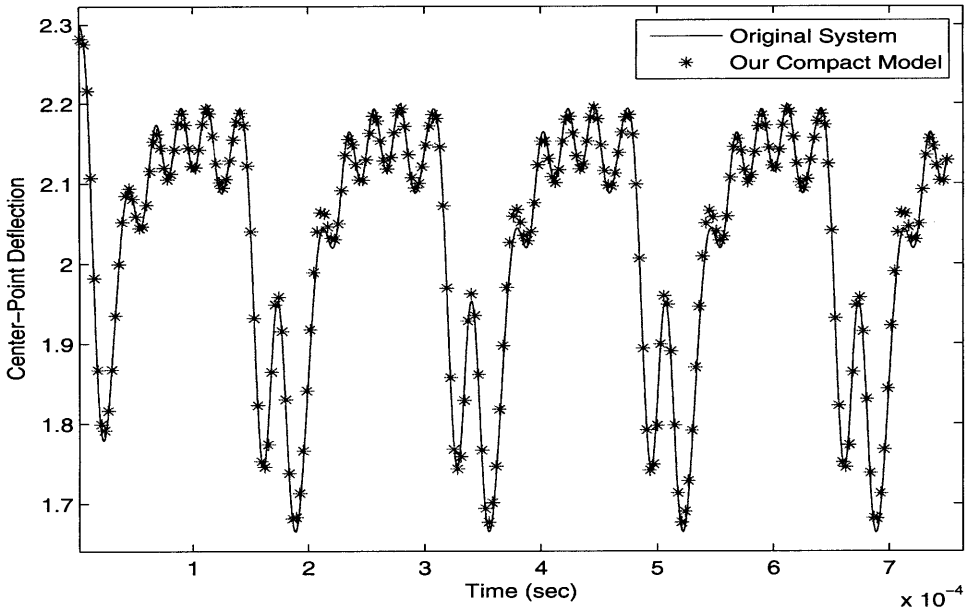


Figure 7-2: Output of order 400 original system (solid line) and our order 4 model (stars) tested on a periodic input of amplitude and frequency different from the training inputs.

model, the reference current is considered as a circuit parameter and varies from 7μA to 19μA.

Training data was generated using inputs of the form

$$u(t) = \text{inp-inn} = \sum_{i=1}^{5} A_i \sin(2\pi f_i t + \phi_i), \tag{7.31}$$

where A_i are chosen randomly, but large enough to saturate the opamp, f_i are randomly sampled between DC to unity-gain frequency, and ϕ_i are randomly sampled in $[0^o, 360^o]$. The resulting model was a parameterized input-output model of the form

$$y[t] = \frac{p(y[t-1], u[t], u[t-1], z)}{q(y[t-1], u[t], u[t-1])}, \tag{7.32}$$

where p is cubic in u, y and quadratic in z, q is a fourth order polynomial of u, y, and the model contains 97 terms.

The identified model was tested on 140 randomly generated inputs of the form (7.31)

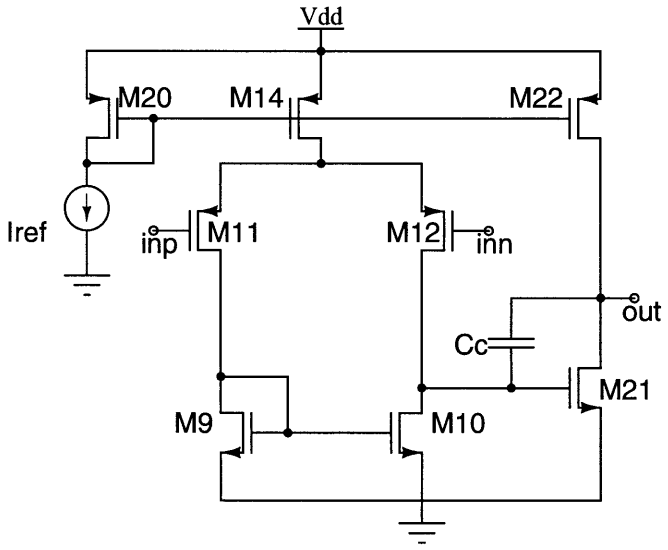


Figure 7-3: Schematic of operational amplifier.

with parameter values randomly selected between 7μA and 19μA. Figure 7-4 plots the model output and output error, defined as

$$e[t] = \frac{|y[t] - \tilde{y}[t]|}{\max_t |\tilde{Y}|} \times 100 \tag{7.33}$$

for one of these testing signals, while Figure 7-5 plots the maximum error over the entire signal for each testing set, defined as

$$e_m = \max_t e[t] \tag{7.34}$$

where $y[t]$ is the output of our model at time t, $\tilde{y}[t]$ is the output of Spectre at time t, and $\tilde{Y}$ is the full waveform of Spectre outputs over one period.

7.7.4 Low Noise Amplifier

In our third example we identify a SIMO model of a single ended Low Noise Amplifier (LNA) designed in 0.5μm CMOS technology [44], shown in Figure 7-6. The designed LNA has a gain of approximately 13dB centered around 1.5GHz.

For this example we wish to capture the nonlinear behavior of both the amplifier output

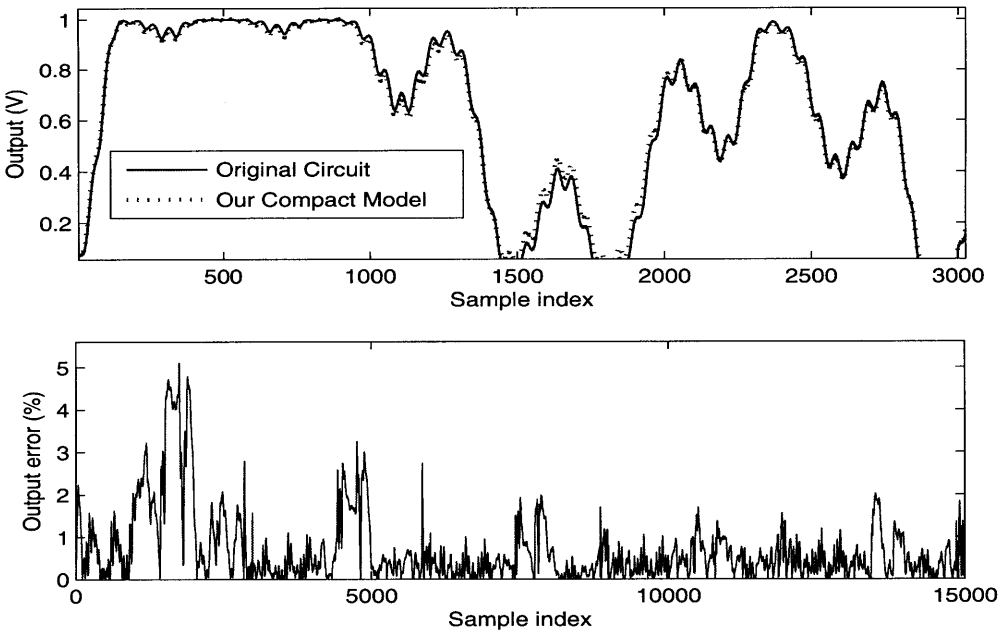


Figure 7-4: Time-domain output and error, as defined in (7.33), for our identified model in response to a random input of the form (7.31) and random parameter value between 7μA and 19μA.

V_{out} and the supply current in response to a modulated input signal with an added jamming signal. The overall input to the system is

$$\mathrm{V_{IN}} = A_j \cos(2\pi f_j t) + \sum_{n=0,1,3,5} A\cos(2\pi n f_0 t)\cos(2\pi f_c t), \tag{7.35}$$

with carrier frequency $f_c = 1.5$GHz, sideband frequency $f_0 = 5$MHz, and jamming frequency $f_j = 1$GHz. The system was trained by varying the amplitude A between 15mV and 85mV, and the jamming amplitude between 0mV and 250mV.

The identified model in this example is a DT multiple-input multiple-output model, usable for instance by a Verilog-A or higher level simulator, described by the rational model

$$Q_2(y[t-1], U[t])y[t] = p_3(y[t-1], U[t]) \tag{7.36}$$

where $U[t] = [u[t], u[t-1], u[t-2]]$, $Q_2 \in \mathbb{R}^{2\times 2}$ is a matrix of second order polynomials, $p_3 \in \mathbb{R}^{2\times 1}$ is a vector of third order polynomials, $y \in \mathbb{R}^{2\times 1}$, and $u \in \mathbb{R}^{2\times 1}$. The rational

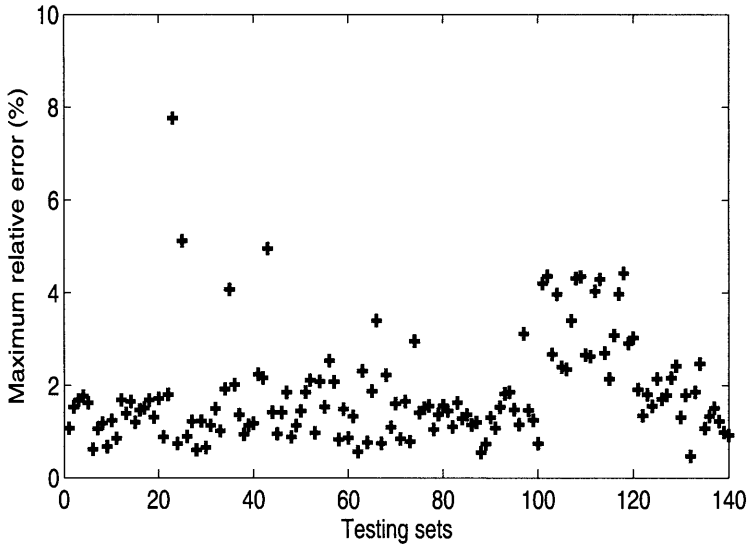


Figure 7-5: Maximum output error, as defined in (7.34), of our model tested on 140 random input signals (7.31) and parameter values ranging from 7μA to 19μA.

nonlinearity is sufficient to capture the highly nonlinear behavior resulting from the large jamming signal, and the total number of parameters describing the identified model is 102. The entire identification procedure took less than two minutes, and the resulting model can be simulated in Matlab for $15,000$ time steps in under three seconds.

To test the model, it was simulated over a full period with six pairs of amplitudes, A and A_j, differing from the training data amplitudes, producing outputs with approximately 4% maximum error from the outputs of the original circuit. Figure 7-7 compares the two time domain outputs, over a small portion of the period, of the model identified by our procedure (dots) with the outputs of the full original circuit (solid lines) over a small time interval in response to an input with $A = 50$mV and $A_j = 150$mV, while Figure 7-8 compares the output percent error over the full period, as defined in (7.33)

7.7.5 Distributed Power Amplifier

The final example considered is a distributed power amplifier designed in 90nm CMOS technology, with a distributed architecture and transformer based power combiner as proposed in [40]. The amplifier delivers 24dBm power with a gain of 6dB at 5.8GHz. A

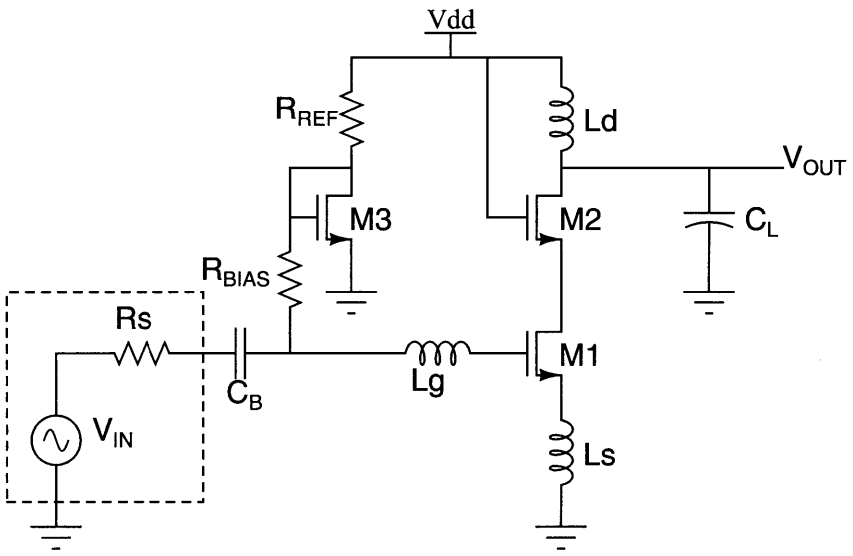


Figure 7-6: Schematic of Low Noise Amplifier (LNA) [44].

simplified schematic of the amplifier is shown in Figure 7-9. Transistors are biased to operate in differential Class-B configuration to improve efficiency, however at the cost of linearity. Nonlinearities also arise because of parasitic inductance introduced by supply and ground bond wires.

Power combining is achieved by using $1:1$ transformers, as shown in Figure 7-9. Losses in primary and secondary inductors are modeled by using quality factor of 12.5 and coupling coefficient of 0.7, based on which optimum values of inductances were selected as Lp=Ls= 157nH [40]. Similarly, the following parameters were selected based on optimized performance of the amplifier at 5.8GHz [40]: Vdd= 1.0V, W/L of transistors = 1.2mm/90nm, $\mathrm{C_{IN}}$ = 2.6pF, $\mathrm{C_{OUT}}$ = 610fF, $\mathrm{R_L} = 50\Omega$, $\mathrm{L_{Vdd}} = L_{GND}$ = 1nH, $\mathrm{C_B}$ = 20pF, Rg = 18Ω.

For this example, training data samples were generated in response to periodic inputs of the form

$$\mathrm{V_{IN}} = V_{DC} + \sum_{n=0,1,3,5} A\cos(2\pi n f_0 t)\cos(2\pi f_c t), \tag{7.37}$$

with carrier frequency $f_c = 5.8$GHz, $f_0 \in \{25, 50\}$MHz, and amplitude $A \in \{30, 90\}$mV. The simulation was performed with Spectre, whose model for the power amplifier con-

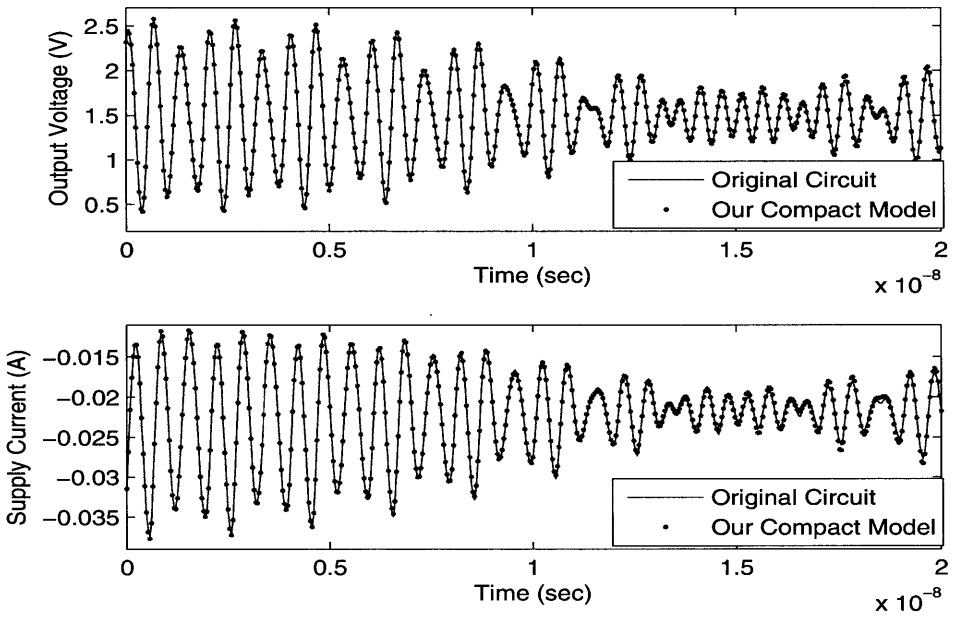


Figure 7-7: Time domain outputs, over a small portion of the period, of the original LNA circuit (solid line) and the compact model identified by our procedure (dots) in response to an input signal different from the training signals.

tained 284 equations, 95 internal variables, and 8 transistors modeled with 90nm predictive technology models.

The specific parameter pairs (A, f_0) used for training and testing are shown in Figure 7-11, where a plus indicates a training input and a dot indicates a testing input not used for training.

The identification procedure, using the reduced basis technique from section 7.5.3, identified a DT input-output model

$$y[t] = \frac{p_3\left(y[t-1], u[t], u[t-1], w[t], w[t-1]\right)}{q_4\left(u[t], u[t-1], w[t], w[t-1]\right)}, \tag{7.38}$$

where

$$w[t] = \sum_{j=0}^{19} b_j u[t-j]$$

is a linear transformation of the input $u[t]$ with coefficients b_j determined by first fitting a

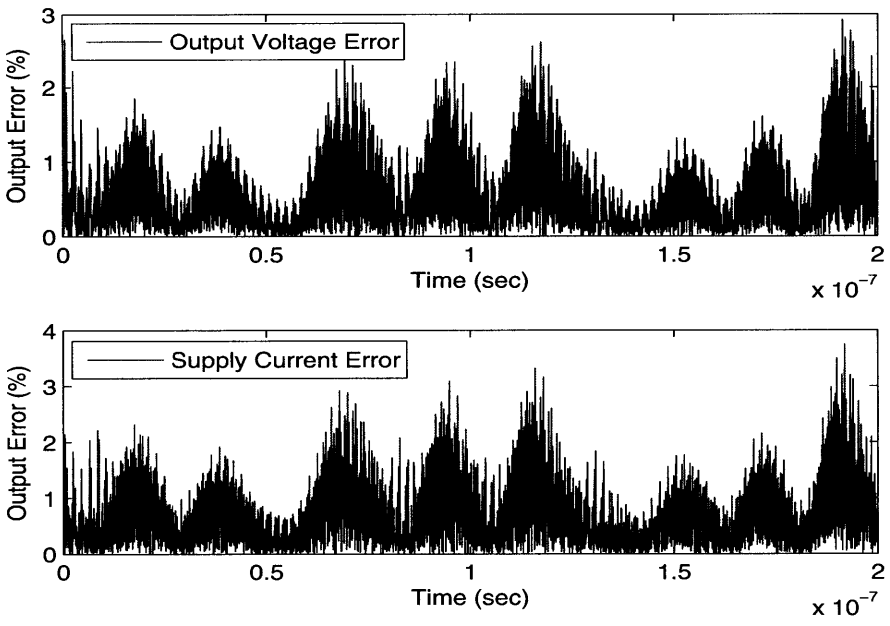


Figure 7-8: Output errors, as defined in (7.33), of our compact model over the full period of the signal from Figure 7-7.

linear system as described in Section 7.5.3. Here p_3 indicates a third order polynomial, q_4 represents a fourth order polynomial, and the total number of parameters in the model is 106. The entire identification procedure took approximately 12 minutes.

The identified model was able to reproduce the training data with less than 4% maximum error in the time-domain, and was able to be simulated for $10,000$ time steps in under 2 seconds. When tested with non-training inputs of the form (7.37) with parameters $A \in \{10, 30, 60, 90\}$mV and $f_0 \in \{10, 25, 40, 50\}$MHz, our model reproduces the outputs of the original circuit with an average error of less than 1% for each testing input. Figure 7-12 compares the output of the identified model with the output of the original power amplifier circuit in response to a testing input with $A = 60$mV and $f_0 = 10$MHz, both differing from the training data set. For clarity, the top plot in Figure 7-12 shows a small portion of the output signals, while the bottom plot shows the model output error, as defined in (7.33), over a full period of the signal. To verify that the model is capturing nonlinear effects, Figure 7-13 compares the magnitude of the Fourier coefficients of an FFT of the output signal in Figure 7-12.

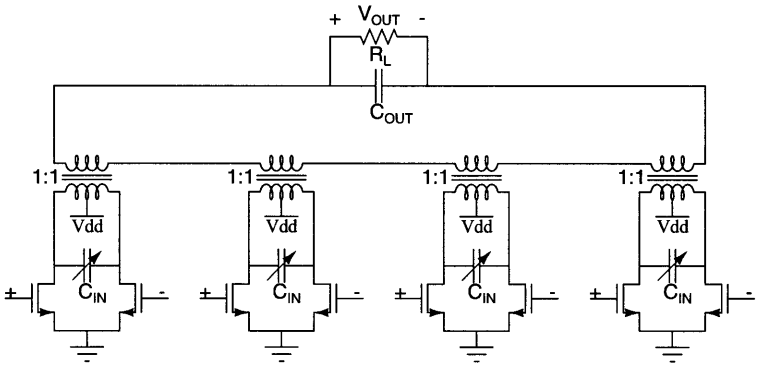


Figure 7-9: Transformer-based power amplifier with distributed architecture [40].

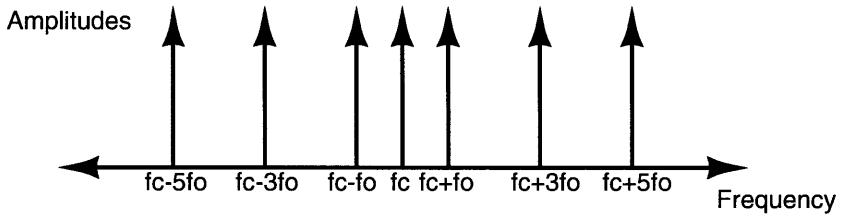


Figure 7-10: Input spectrum for distributed amplifier.

To show that our model, trained only with amplitude-modulated sinusoids, is capable of capturing the circuit behavior in response to also ***different classes of inputs***, Figure 7-14 plots the constellation diagram for the output from our model in response to a 16-QAM input signal, which is a non-smooth input.

A Volterra model with approximately the same number of parameters identified with our procedure for this example produced over three times the average error on the training data set compared to model (7.38). With our current testing setup, it was not possible to obtain a pure Volterra model that is as accurate as model (7.36) due to memory constraints in our computer (4GB). This is a result of the large number of parameters that would be required in the Volterra model of high order and with many delays.

In addition to matching input-output behavior, it is important that our identified models can also accurately predict the performance curves of the circuits being modeled. The top plot in Figure 7-15 plots output power versus input power (compression curve) at 5.8GHz for the original circuit (solid line) and our identified model (circles), while the bottom plot show the drain efficiency (defined as the ratio of output RF power to input DC power)

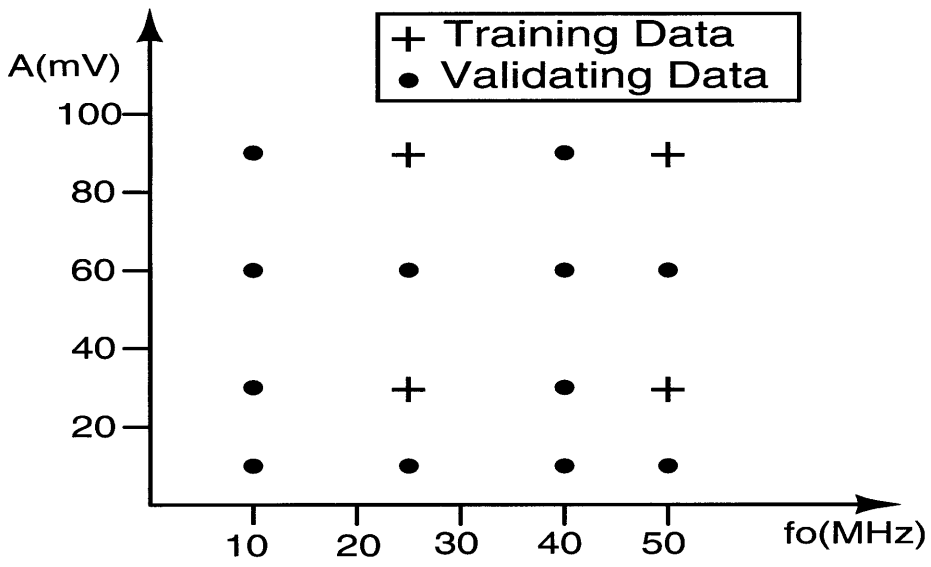


Figure 7-11: Grid of amplitude, A, and modulating frequency, f_0, values defining inputs (7.37) used for training (pluses) and validating (dots) the power amplifier compact model.

versus output power for the original circuit (solid line) and our identified model (circles), also at 5.8GHz. This model was identified by training with sinusoids at 5.8GHz with amplitudes $A \in [100, 400, 800, 1200]$mV, and was tested at 12 amplitudes evenly spaced between 100mV and 1200mV. We want to emphasize that these performance curves were obtained from simulation of our identified ***dynamical models***, and not by simply fitting performance curves.

7.7.6 Comparison to Existing SYSID Techniques

Finally, we compare our proposed approach to several existing SYSID techniques from literature. Traditional identification techniques suffer from several shortcomings. Some techniques, such as the Hammerstein-Wiener (H-W) model [50] (a cascade connection of an LTI system between two memoryless nonlinearities), forces a specific block-structure on the model which restricts the types of systems that can be accurately modeled. Volterra models, such as 7.26, do not force a specific block structure, but require many parameters to represent complex systems due to a lack of feedback and polynomial nonlinearities. More general nonlinear models, such as nonlinear autoregressive model with exogenous inputs

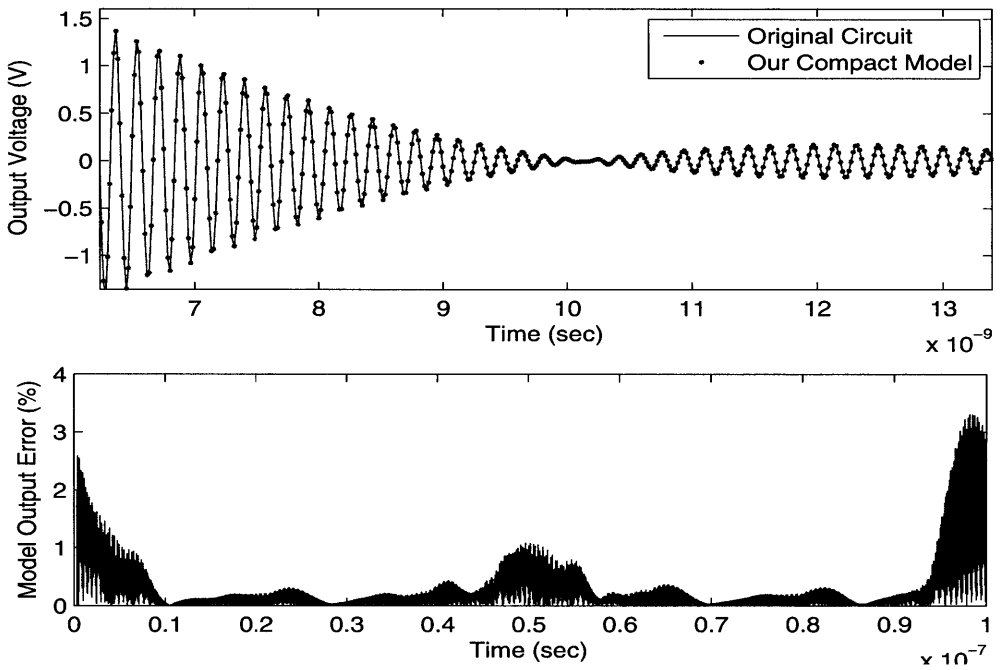


Figure 7-12: (a): Time domain output of the original circuit (solid line) and the compact model identified by our procedure (dots) in response to a testing input with amplitude and frequency different from the training inputs. (b): Output error e_t, as defined in (7.33), of our model over the full signal from (a)

(NLARX) [50], have the more general structure

$$y[t] = f(y[t-1], \ldots, y[t-m], u[t], \ldots, u[t-k]),$$

which is similar to the DT models identified by our proposed procedure, and do incorporate feedback, but they typically do not explicitly enforce stability during identification. For both the H-W and NLARX models, the nonlinearities are typically identified as a linear combination of nonlinear basis functions.

The same training data sets from section 7.7.5 were used to identify models of the distributed power amplifier in the form of a H-W model and a NLARX model. These models were generated using the MATLAB system identification toolbox, which uses techniques described in [50]. In general, both types of models were found to be less accurate than our proposed approach, with the H-W models producing average errors between 5% and 10%, and the NLARX models producing average errors between 3% and 5%, compared to aver-

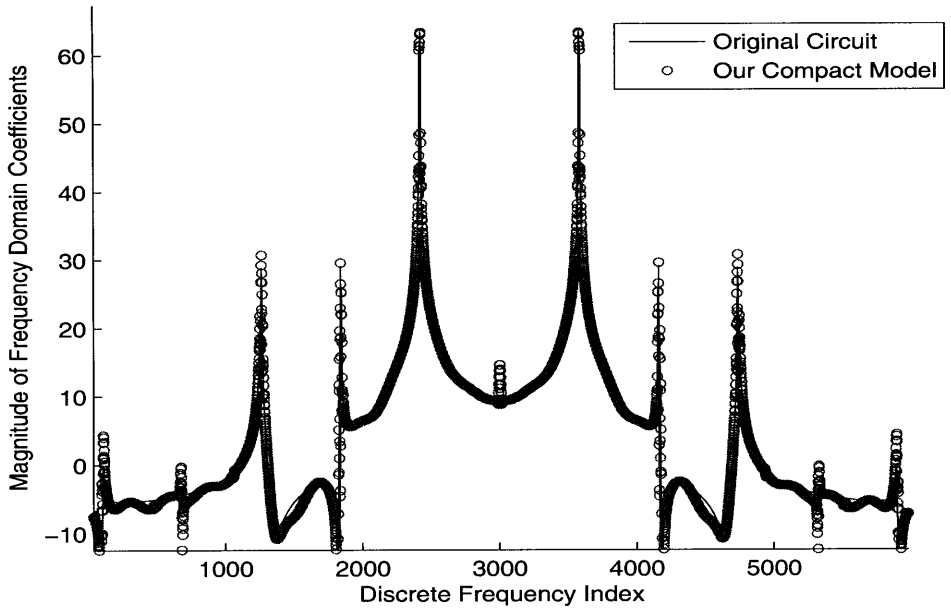


Figure 7-13: Magnitude of the frequency domain coefficients of an FFT of the time domain output of the original circuit (solid line) and our compact model (circles) in response to a testing input with frequency and amplitude different from the training inputs.

age errors of 1% from model (7.38) identified by our technique. Additionally, ***the NLARX models were often unstable***, and as a result, the testing inputs often produced unbounded outputs. Figure 7-16 plots the output response of our compact model (circles), a H-W model (pluses), and a NLARX model (stars), all generated from the four training inputs from Section 7.7.5, compared to the output of the original circuit (solid line) in response to one testing input with frequency and amplitude different from the training data. For this example all three identified models contain approximately the same number of parameters, and the nonlinearities in both the HW and NLARX models were described by sigmoidnet functions.

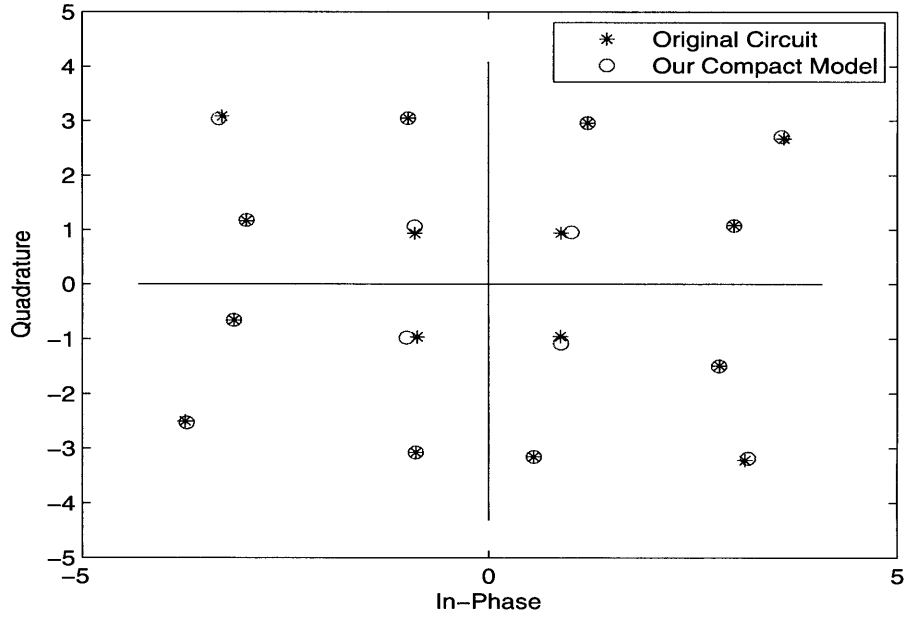


Figure 7-14: Constellation diagram for output of the power amplifier model (trained using only amplitude-modulated sinusoids) in response to a 16-QAM input signal.

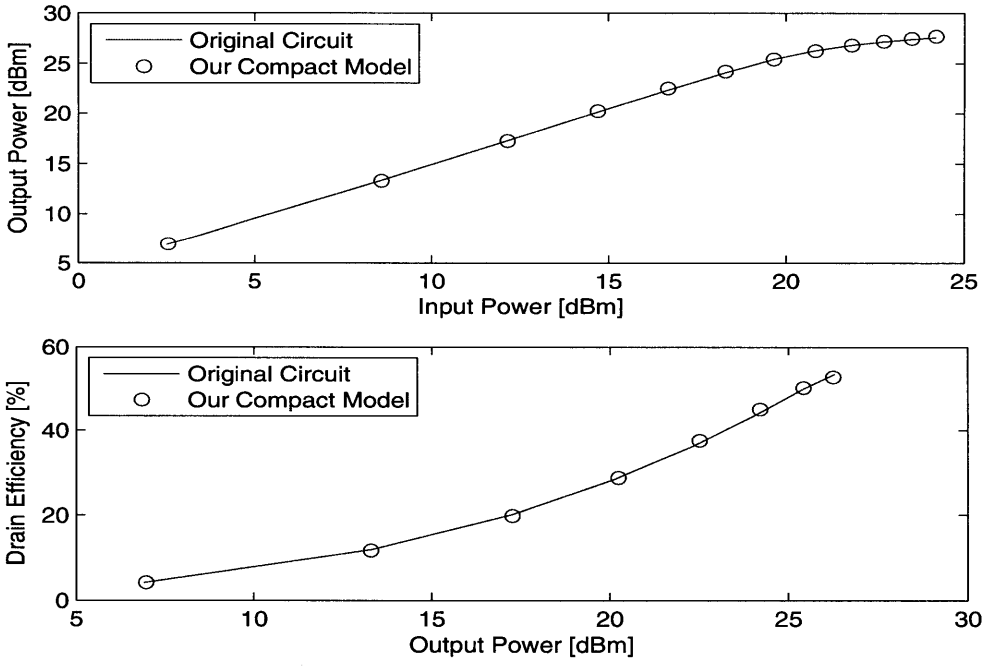


Figure 7-15: (top): Compression curve, plotting input power versus output power, at 5.8GHz for the original circuit (solid line) and our compact model (circles). (bottom): Drain efficiency versus output power at 5.8GHz for the original circuit (solid line) and our compact model (circles).

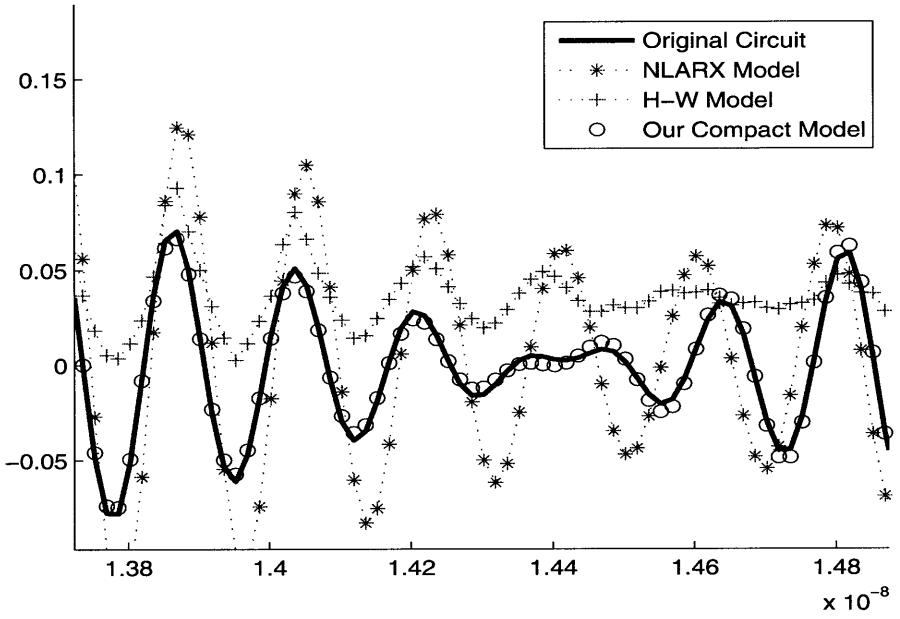


Figure 7-16: Outputs of our compact model (circles), a Hammerstein-Wiener (H-W) model (pluses), and a NLARX model (stars) all generated from the four training inputs used in Section 7.7.5, compared to the output of the original circuit (solid line) in response to a training input with $f_0 = 10\text{MHz}$ and $A = 60\text{mV}$.

Chapter 8

Conclusion

The ability to automatically generate compact models of analog system blocks could greatly facilitate the design and optimization of such systems. Due to the presence of nonlinear and parasitic components the numerical analysis of complex analog and RF circuits is often prohibitively expensive. In this thesis we have presented several new model reduction techniques for linear and nonlinear systems. Our approaches have specifically focused on preserving stability and passivity in the reduced models. Stability and passivity are crucial features of a model if it is to be used for time-domain simulation, otherwise the resulting simulations could produce unrealistic results. Additionally, we have focused on preserving parameter dependence of the original system so that the reduced models can be used for design optimization.

For linear systems, we have presented projection techniques for creating stable and passive reduced models from originally indefinite and unstable large models, which is not possible using existing techniques. For highly nonlinear systems we have presented projection approaches capable of preserving parameter dependence in the reduced model and also techniques for efficiently stabilizing the nonlinear reduced models. Instability in reduced models is an unavoidable problem for most other nonlinear reduction techniques. Lastly, since traditional projection reduction techniques are in some cases prohibitive due to their requirement of precise knowledge of the large dynamical system equations, we presented a system identification approach, for both linear and nonlinear systems, capable of generating parameterized and stable compact models based only on time-domain data from the

original system.

As a next step towards automatic design optimization, the model reduction techniques developed in this thesis can be used to generate parameterized models of entire systems, such as an RF receiver chain. The resulting parameterized compact model can be efficiently used in an optimization routine in order to obtain optimized system-level designs.

Appendix A

Modeling and Simulation Tools

This appendix outlines briefly some of the modeling and simulation tools developed during the course of this thesis work. These tools include a basic circuit simulator used for generating dynamical system equations, an implementation of our SYSID approach from Chapter 7, and Matlab code for modeling the MEMS device nonlinear system example.

A.1 SMORES: A Matlab tool for Simulation and Model Order Reduction of Electrical Systems

SMORES (Simulation and Model Order Reduction for Electrical Systems) is a Matlab-based circuit simulator integrated with model reduction, which is freely available online at [11]. By giving users access to the dynamical system equations describing the circuits being simulated, we hope this software will also facilitate the development of new model reduction techniques and provide some standard benchmark examples to allow easier comparison between methods. We wish to emphasize that there are no novel contributions to numerical simulation in this tool and it is not intended to be a commercial caliber simulator, but rather an open source prototyping tool to enable the testing and development of new modeling and simulation techniques.

SMORES takes as input a spice-like netlist describing the circuit, such as the example shown in Table A.1, which specifies transient simulation of a simple inverter circuit.

SMORES can perform several types of analysis (e.g. transient or steady-state) as well as

```
*opamp.ckt
.t 0 4e-5 5e-8
.n 20 1e-8 1e-8 0
Vs1 2 9 2.5e5 0 0 5e-3
Vs2 9 10 3.75e5 0 0 5e-3
Vs3 10 1 8.75e5 0 0 5e-3
Vdc 1 0 250e-3
Vdd 3 0 1
C0 5 4 100e-15
I0 6 0 10e-6
P12 5 2 7 3 1 712e-9 128e-9 0 0 1e-4
P11 8 1 7 3 1 712e-9 128e-9 0 0 1e-4
P22 4 6 3 3 1 460e-9 180e-9 0 0 1e-4
P14 7 6 3 3 1 460e-9 180e-9 0 0 1e-4
P20 6 6 3 3 1 460e-9 180e-9 0 0 1e-4
N21 4 5 0 0 1 3.495e-6 100e-9 0 0 1e-4
N10 5 8 0 0 1 781e-9 125e-9 0 0 1e-4
N9  8 8 0 0 1 781e-9 125e-9 0 0 1e-4
```

Table A.1: Sample netlist for an opamp.

several types of model order reduction. For example, invoking SMORES on the netlist in Table A.1, which specifies transient analysis for an opamp, will solve for the node voltages in the circuit, shown in Figure A-1. Additionally, SMORES allows the user to access the dynamical system equations describing the circuit. In this simulator, equation formulation is based on nodal analysis and uses a stamping procedure to translate device-level netlist descriptions of a circuit into a dynamical system of the form

$$\frac{\partial q(x)}{\partial t} + \mathcal{C}\frac{\partial x}{\partial t} + f(x) + \mathcal{G}x = B_i u_i + B_v u_v. \tag{A.1}$$

Here x contains the node voltages and inductor currents, q contains charges from nonlinear elements, $\mathcal{C}$ contains linear capacitors and inductors, f contains currents from nonlinear elements, $\mathcal{G}$ contains linear resistors and adjacency matrix for inductors, B_i is the input matrix for current sources, B_v is the input matrix for voltage sources, u_i is a vector of current inputs, and u_v is a vector of voltage inputs. We have specifically separated out the linear elements in the matrices $\mathcal{C}$ and $\mathcal{G}$ to make numerical simulation more efficient. A

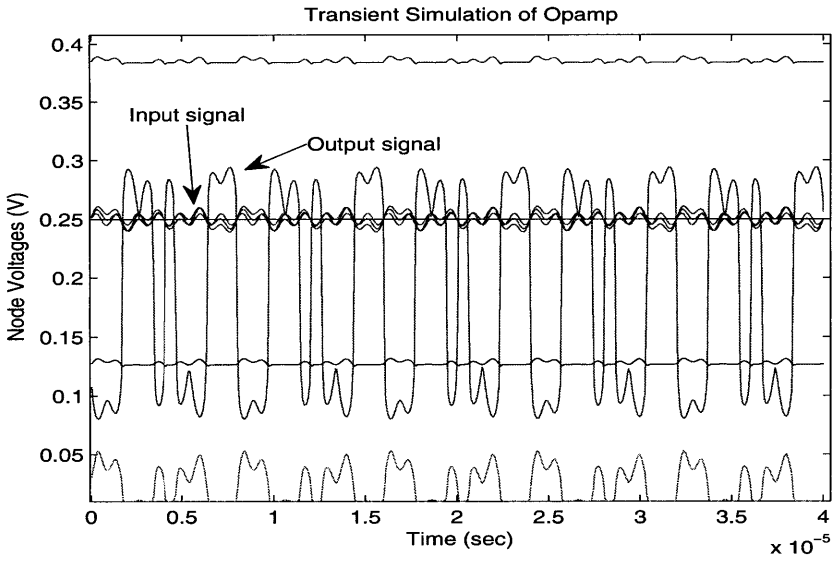


Figure A-1: Transient simulation response of opamp circuit described by netlist in Table A.1.

detailed list of available device models, analysis types, and netlist syntax can be found in the software documentation [11]. It is also possible for users to define new device models to be used by the simulator.

To facilitate testing new model reduction techniques, our simulator is capable of returning to the user linearizations of the nonlinear system at points along the solution trajectory, and Krylov vectors generated at specified frequency expansion points for each linear model. A linearization of nonlinear system (A.1) at state x_k has the form

$$E_k\dot{x} = A_k x + K_k + B_i u_i + B_v u_v,$$

where the Jacobian matrices E_k and A_k are defined as

$$E_k = \frac{\partial q}{\partial x}\Big|_{x_k} + \mathcal{C}, \qquad A_k = -\frac{\partial f}{\partial x}\Big|_{x_k} - \mathcal{G}, \qquad K_k = \mathcal{G}x_k - f(x_k).$$

With these capabilities our software may serve as a useful prototyping tool for testing nonlinear model reduction techniques, and also provides easy access to nonlinear system examples that may be used as benchmarks for comparisons between model reduction methods.

A.2 STINS: A Matlab tool for Stable Identification of Nonlinear systems via Semidefinite programming

STINS (Stable Identification of Nonlinear systems via Semidefinite programming) is a Matlab implementation of our SYSID approach from chapter 7 for identifying stable discrete-time input-output models, and is freely available online at [12]. In STINS, a user provides as input to the software training data input-output signal pairs and a set of parameters describing a rational model to be identified, and the software returns a set of coefficients defining a stable model described by the previously defined parameters.

For example, consider the two pairs of input-output signals shown in Figure A-2. Using

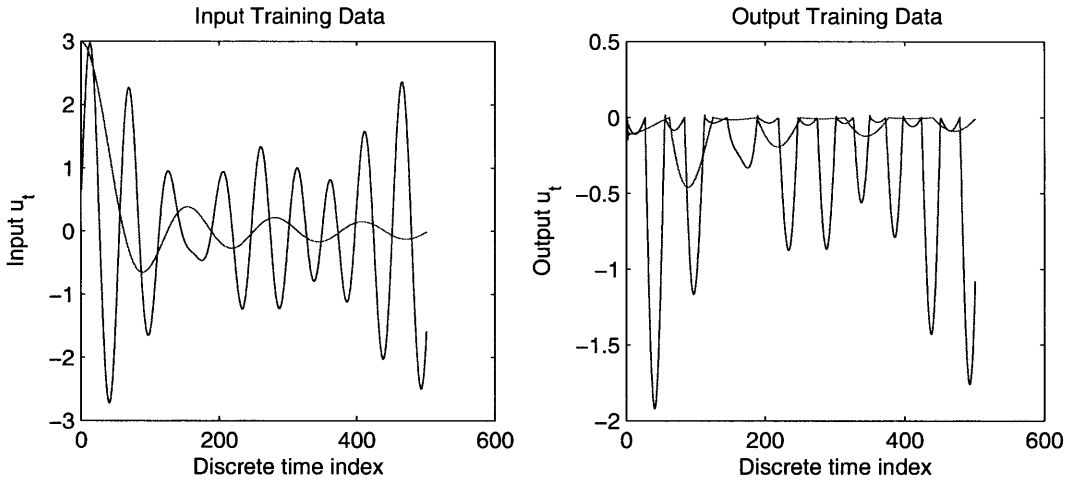


Figure A-2: Sample input and output pairs used as training data for model identification.

these signals as training data and selecting model parameters describing a model of the form

$$y_t = \frac{p_3(y_{t-1}, u_{t-1})}{q_2(y_{t-1}, u_{t-1})}, \tag{A.2}$$

where $p_3(\cdot)$ indicates a third order polynomial and $q_2(\cdot)$ indicates a second order polynomial, STINS solves for the coefficients describing the functions $p_3(\cdot)$ and $q_2(\cdot)$ such that system (A.2) is stable and optimally fits the given training data. STINS also allows for efficient simulation of the identified model (A.2) in response to testing input signals differing from the training data. One such result is shown in Figure A-3.

A detailed explanation and derivation of the algorithm in STINS is found in Chapter 7,

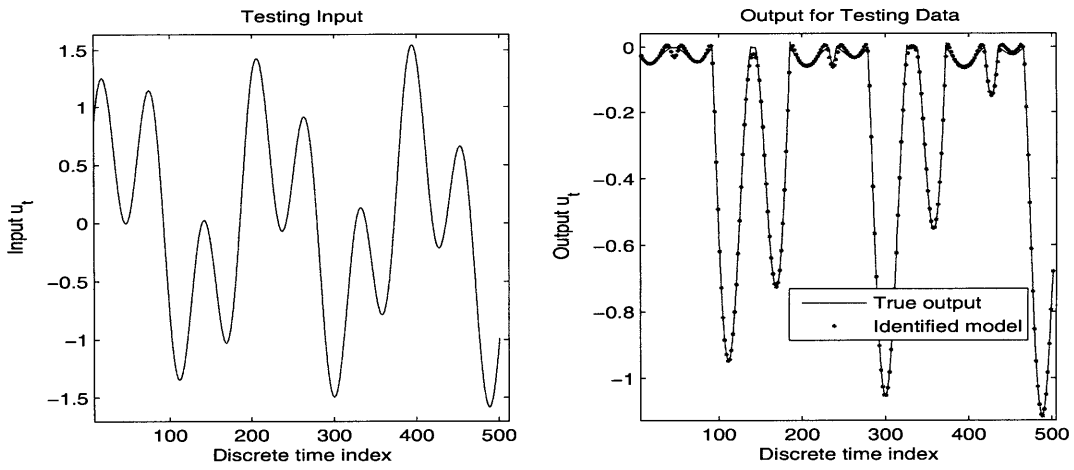


Figure A-3: Output of model identified using training data in Figure A-2 in response to an input differing from the training data.

and detailed documentation for using the software can be found online at [12].

A.3 Matlab Code for Micromachined Switch MEMS Example

Lastly, we have made freely available the Matlab functions necessary for modeling the nonlinear MEMS device example considered throughout this thesis (see e.g. section 5.3.2). The dynamical system describing the MEMS device's behavior has the form

$$\dot{x} = f(x) + Bu,$$

where a detailed description of how the function $f(x)$ is constructed was previously presented in [77]. In order to simulate this dynamical system numerically it is necessary to be able to evaluate the vector field function $f(x)$ and the Jacobian matrix $J(x) = \partial f / \partial x$ at any given state vector x. To this end, we have provided a function `BeamFunc.m` that evaluates the function $f(x)$, a function `MemsJac.m` that evaluates the Jacobian $J(x)$, a function `MemsConstsAndOperators.m` that defines a set of parameters describing the geometry and physical properties of the MEMS device, and also a short script `MemsTest.m` that illustrates how to invoke the previous three functions [10]. With these Matlab functions

the nonlinear dynamical system describing the MEMS device can be simulated using any standard ODE solver.

Bibliography

[1] Fastmaxwell. *http://www.rle.mit.edu/cpg/fastmaxwell.*

[2] E. Afshari and A. Hajimiri. A non-linear transmission line for pulse shaping in silicon. *IEEE Journal Solid-State Circuits*, 40(3):744–52, 2005.

[3] D. Angeli. A lyapunov approach to incremental stability properties. *Automatic Control, IEEE Transactions on*, 47(3):410–421, Mar 2002.

[4] Z. Bai, P. Feldmann, and R.W. Freund. Stable and passive reduced order models based on partial pade approximation via the lanczos process. Technical Report Numerical Analysis Manuscript No.97-3-10, Bell Laboratories, Lucent Technologies, Murray Hill, New Jersey, October 1997.

[5] P. Benner, J. Li, and T. Penzl. Numerical solution of large lyapunov equations, riccati equations, and linear-quadratic control problems. *Numerical Linear Algebra with Applications*, 15(9):755–777, 2008.

[6] P. Benner, H. Mena, and J. Saak. Numerical solution of large lyapunov equations, riccati equations, and linear-quadratic control problems. *Electronic Transactions on Numerical Analysis*, 29:136–149, 2008.

[7] G Berkooz, P Holmes, and J L Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25(1):539–575, 1993.

[8] W. Beyene and Schutt-Aine. Efficient Transient Simulation of High-Speed Interconnects Characterized by Sampled Data. *IEEE Trans on Components, Packaging, and Manufacturing Technology-Part B*, 21(1):105–114, February 1998.

[9] S. Billings and S. Fakhouri. Identification of a class of nonlinear systems using correlation analysis. *Proceedings of Institution of Electrical Engineers*, 125:691–697, July 1978.

[10] B. Bond. Matlab functions for simulation of a micromachined switch mems device, 2010. http://www.bnbond.com/software/mems.

[11] B. Bond. Smores: A matlab tool for simulation and model order reduction of electrical systems, 2010. http://www.bnbond.com/software/smores.

[12] B. Bond. Stins: A matlab tool for stable identification of nonlinear models via semidefinite programming, 2010. http://www.bnbond.com/software/stins.

[13] B. Bond and L. Daniel. Parameterized model order reduction for nonlinear dynamical systems. In *Proc. of the IEEE/ACM International Conference on Computer-Aided Design*, pages 487–494, San Jose, CA, 2005.

[14] B. Bond and L. Daniel. A piecewise-linear moment-matching approach to parameterized model-order reduction for highly nonlinear systems. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 26(12):2116–2129, Dec 2007.

[15] B. Bond and L. Daniel. Stabilizing schemes for piecewise-linear reduced order models via projection and weighting functions. In *Proc. of the IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, 2007.

[16] B. Bond and L. Daniel. Guaranteed stable projection-based model reduction for indefinite and unstable linear systems. In *Proc. of the IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, 2008.

[17] B. Bond and L. Daniel. Stable reduced models of nonlinear descriptor systems through piecewise linear approximation and projection. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 28(10):1467–1480, Oct 2009.

[18] S. Boyd. Volterra Series: Engineering Fundamentals. PhD thesis, University of California, Berkeley, 1985.

[19] Y. Chen. Model order reduction for nonlinear systems. M.S. Thesis, Massachusetts Institute of Technology, September 1999.

[20] C. Coelho, J. Phillips, and L. Silveira. Optimization Based Passive Constrained Fitting. In *the 2002 International Conference on Computer Aided Design*, pages 10–14, San Jose, California, November 2002.

[21] L. Daniel. Simulation and modeling techniques for signal integrity and electromagnetic interference on high frequency electronic systems. P.hD. Thesis, University of California at Berkeley, June 2003.

[22] L. Daniel, C. S. Ong, S. C. Low, K. H. Lee, and J. K. White. A multiparameter moment matching model reduction approach for generating geometrically parameterized interconnect performance models. *IEEE Trans. Computer-Aided Design*, 23(5):678–93, May 2004.

[23] L. Daniel and J. R. Phillips. Model order reduction for strictly passive and causal distributed systems. In *Proc. of the ACM/IEEE Design Automation Conference*, New Orleans, LA, June 2002.

[24] L. Daniel and J. White. Automatic generation of geometrically parameterized reduced order models for integrated spiral rf-inductors. In *IEEE Intern. Workshop on Behavioral Modeling and Simulation*, San Jose, CA, September 2003.

[25] N. Dong and J. Roychowdhury. Piecewise polynomial nonlinear model reduction. In *Proc. of the ACM/IEEE Design Automation Conference*, June 2003.

[26] N. Dong and J. Roychowdhury. Automated extraction of broadly applicable nonlinear analog macromodels from spice-level descriptions. In *Proc. of the IEEE Custom Integrated Circuits Conference*, October 2004.

[27] S.D. Senturia E. Hung, Y.-J. Yahg. Low-order models for fast dynamical simulation of mems microstructures. In *Proceedings of IEEE International Conference on Solid State Sensors and Actuators*, volume 2, pages 1101–1104, 1997.

[28] Peter Feldmann and Roland W. Freund. Efficient linear circuit analysis by Padé approximation via the Lanczos process. In *EURO-DAC'94 with EURO-VHDL'94*, September 1994.

[29] R. W. Freund. Sprim: structure-preserving reduced-order interconnect macromodeling. In *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 80–87, Washington, DC, USA, 2004. IEEE Computer Society.

[30] Roland W. Freund and Peter Feldmann. Efficient Small-Signal Circuit Analysis and Sensitivity Computations with the PVL Algorithm. In *Proc. of IEEE/ACM International Conference on Computer Aided-Design*, San Jose, California, November 1994.

[31] K. Gallivan, E. Grimme, and P. Van Dooren. Multi-point padé approximants of large-scale systems via a two-sided rational krylov algorithm. In 33^{rd} *IEEE Conference on Decision and Control*, Lake Buena Vista, FL, December 1994.

[32] Eric Grimme. *Krylov Projection Methods for Model Reduction.* PhD thesis, Coordinated-Science Laboratory, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL, 1997.

[33] S. Grivet-Talocia and A. Ubolli. A comparative study of passivity enforcement schemes for linear lumped macromodels. *IEEE Trans. on Advanced Packaging*, 31(4), Nov. 2008.

[34] C. Gu and J. Roychowdhury. Model reduction via projection onto nonlinear manifolds, with applications to analog circuits and biochemical systems. In *ICCAD '08: Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 85–92, Piscataway, NJ, USA, 2008. IEEE Press.

[35] P. Gunupudi and M. Nakhla. Multi-dimensional model reduction of vlsi interconnects. In *Proc. of the Custom Integrated Circuits Conference*, pages 499–502, Orlando, FL, 2000.

[36] B. Gustavsen and A. Semlyen. Rational approximation of frequency domain responses by vector fitting. *IEEE Trans. on Power Delivery*, 14(3), Jul 1999.

[37] B. Gustavsen and A. Semlyen. Rational Approximation of Frequency Domain Responses by Vector Fitting. *IEEE Trans on Power Delivery*, 14(3):1052–1061, July 1999.

[38] R. Haber and L. Keviczky. *Nonlinear System Identification - Input-Output Modeling Approach: Volume 1: Nonlinear System Parameter Identification*. Springer, 1999.

[39] Anna Hagenblad, Lennart Ljung, and Adrian Wills. Maximum Likelihood Identification of Wiener Models. Submitted to Automatica, November 2007.

[40] P. Haldi, D. Chowdhury, P. Reynaert, Gang Liu, and A.M. Niknejad. A 5.8 ghz 1 v linear power amplifier using a novel on-chip transformer power combiner in standard 90 nm cmos. *Solid-State Circuits, IEEE Journal of*, 43(5):1054–1063, May 2008.

[41] P. Heydari and M. Pedram. Model reduction of variable-geometry interconnects using variational spectrally-weighted balanced truncation. In *Proc. of IEEE/ACM International Conference on Computer Aided-Design*, San Jose, CA, November 2001.

[42] K. J. Kerns, I. L. Wemple, and A. T. Yang. Stable and efficient reduction of substrate model networks using congruence transforms. In *Proc. of IEEE/ACM International Conference on Computer Aided-Design*, pages 207 – 214, San Jose, CA, November 1995.

[43] M. Kozek and C. Hametner. Block-oriented identification of Hammerstein/Wiener-models using the RLS-algorithm. *International Journal of Applied Electromagnetics and Mechanics*, 25:529–535, 2007.

[44] Thomas H. Lee. *The Design of CMOS Radio-Frequency Integrated Circuits*. Cambridge University Press, 1998.

[45] J. Lfberg. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[46] P. Li, F. Liu, S. Nassif, and L. Pileggi. Modeling interconnect variability using efficient parametric model order reduction. In *Design, Automation and Test Conference in Europe*, March 2005.

[47] X. Li, P. Li, and L. T. Pileggi. Parameterized interconnect order reduction with explicit-and-implicit multi-parameter moment matching for inter/intra-dit variations. In *Proc. of IEEE/ACM International Conference on Computer Aided-Design*, pages 806–812, San Jose, CA, 2005.

[48] H. Liu, A. Singhee, R. A. Rutenbar, and L. R. Carley. Remembrance of circuits past: macromodeling by data mining in large analog design spaces. In *Proc. of the ACM/IEEE Design Automation Conference*, pages 437–42, June 2002.

[49] Y Liu, Lawrence T. Pileggi, and Andrzej J. Strojwas. Model order-reduction of RCL interconnect including variational analysis. In *Proc. of the ACM/IEEE Design Automation Conference*, pages 201–206, New Orleans, Louisiana, June 1999.

[50] L. Ljung. *System Identification, Theory for the User*. Prentice Hall, 2nd edition, 1999.

[51] Lennart Ljung. Identification of nonlinear systems. Technical Report LiTH-ISY-R-2784, Department of Electrical Engineering, Linkping University, SE-581 83 Linkping, Sweden, June 2007.

[52] W. Lohmiller and J. Slotline. On contraction analysis for non-linear systems. *Automatica*, 34(6):683–696, 1998.

[53] J. Lumley. The structures of inhomogeneousturbulent flow. *Atmospheric Turbulence and Radio Wave Propagation*.

[54] S. Maas. How to model intermodulation distortion. In *Microwave Symposium Digest, 1991., IEEE MTT-S International*, pages 149–151, Boston, MA, 1991.

[55] A. Megretski. Lecture notes on projection-based model reduction. *http://web.mit.edu/6.242/www/syll.html*, September 2004.

[56] A. Megretski. Convex optimization in robust identification of nonlinear feedback. In *IEEE Conference on Decision and Control*, December 2008.

[57] A. Megretski. Pot: Polynomial optimization toolbox, 2009. http://web.mit.edu/ameg/www/.

[58] MIT. Lecture notes for 6.241 dynamic systems and control, 2003. http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-241Fall2003/LectureNotes/index.htm.

[59] B. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, 26(1):17–31, 1981.

[60] Bruce Moore. Principal Component Analysis in Linear Systems: Controllability, Observability, and Model Reduction. *IEEE Transactions on Automatic Control*, AC-26(1):17–32, February 1981.

[61] T. Moselhy, X. Hu, and L. Daniel. pfft in fastmaxwell: a fast impedance extraction solver for 3d conductor structures over substrate. In *DATE '07: Proceedings of the conference on Design, automation and test in Europe*, pages 1194–1199, San Jose, CA, USA, 2007. EDA Consortium.

[62] A. Odabasioglu, M. Celik, and L. T. Pileggi. PRIMA: passive reduced-order interconnect macromodeling algorithm. *IEEE Trans. Computer-Aided Design*, 17(8):645–654, August 1998.

[63] A. Oppenheim, A. Willsky, and S. Nawab. *Signals & systems (2nd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[64] P. Parillo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, 2000.

[65] A. Pavlov, A. Pogromsky, N. van de Wouw, and H. Nijmeijer. Convergent dynamics, a tribute to boris pavlovich demidovich. *Systems & Control Letters*, 52(3-4):257 – 261, 2004.

[66] J. R. Phillips, L. Daniel, and M. Silveira. Guaranteed passive balancing transformations for model order reduction. *IEEE Trans. Computer-Aided Design*, 22(8):1027–41, Aug 2003.

[67] J.R. Phillips. Projection-based approaches for model reduction of weakly nonlinear, time-varying systems. *IEEE Trans. Computer-Aided Design*, 22(2):171–87, 2003.

[68] J.R. Phillips. Variational interconnect analysis via PMTBR. In *Proc. of IEEE/ACM International Conference on Computer Aided-Design*, pages 872–9, November 2004.

[69] J.R. Phillips, L. Daniel, and L.M. Silveira. Guaranteed passive balancing transformations for model order reduction. In *DAC '02: Proceedings of the 39th annual Design Automation Conference*, pages 52–57, New York, NY, USA, 2002. ACM.

[70] J.R. Phillips and L. M. Silveira. Poor man's tbr: A simple model reduction scheme. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, page 20938, Washington, DC, USA, 2004. IEEE Computer Society.

[71] L.T. Pillage, X. Huang, and R.A. Rohrer. AWEsim: Asymptotic Waveform Evaluation for Timing Analysis. In 26^{th} *ACM/IEEE Design Automation Conference*, pages 634–637, Las Vegas, Nevada, June 1989.

[72] L.T. Pillage and R.A. Rohrer. Asymptotic Waveform Evaluation for Timing Analysis. *IEEE Transactions on Computer-Aided Design*, 9(4):352–366, April 1990.

[73] L.T. Pillage and R.A. Rohrer. Asymptotic Waveform Evaluation for Timing Analysis. *IEEE Trans. Computer-Aided Design*, 9(4):352–366, April 1990.

[74] K. Poolla, P. Khargonekar, A. Tikku, J. Krause, and K. Nagpal. A Time-Domain Approach to Model Validation. *IEEE Trans. on Automatic Control*, 39(5):951–959, May 1994.

[75] C. Prud'homme, D. Rovas, K. Veroy, Y. Maday, A.T. Patera, and G. Turinici. Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bounds methods. *Journal of Fluids Engineering*, 2002.

[76] S. Pullela, N. Menezes, and L.T. Pileggi. Moment-sensitivity-based wire sizing for skew reduction in on-chip clock nets. *IEEE Trans. Computer-Aided Design*, 16(2):210–215, February 1997.

[77] M. Rewienski. A trajectory piecewise-linear approach to model order reduction of nonlinear dynamical systems. P.hD. Thesis, Massachusetts Institute of Technology, June 2003.

[78] M. Rewienski and J. White. A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. *IEEE Trans. Computer-Aided Design*, 22(2):155–70, Feb 2003.

[79] M. Rewienski and J. K. White. A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. In *Proc. of IEEE/ACM International Conference on Computer Aided-Design*, pages 252–7, San Jose, CA, USA, November 2001.

[80] B. Reznick. Extremal psd forms with few terms. *Duke Math. J.*, 45(2):363–374, 1978.

[81] S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer, New York, 1999.

[82] J. Scherpen. Balancing for nonlinear systems. *Systems & Control Letters*, 21:143–153, 1993.

[83] L. Miguel Silveira, Mattan Kamon, Ibrahim Elfadel, and Jacob K. White. A coordinate-transformed Arnoldi algorithm for generating guaranteed stable reduced-order models of RLC circuits. In *Proc. of IEEE/ACM International Conference on Computer Aided-Design*, pages 288–294, San Jose, California, November 1996.

[84] L. Miguel Silveira, Mattan Kamon, and Jacob K. White. Efficient reduced-order modeling of frequency-dependent coupling inductances associated with 3-d interconnect structures. In *proceedings of the European Design and Test Conference*, pages 534–538, Paris, France, March 1995.

[85] R. Smith and J. Doyle. Model Validation: A Connection Between Robust Control and Identification. *IEEE Trans. on Automatic Control*, 37(7):942–952, July 1992.

[86] K. Sou, A. Megretski, and L. Daniel. A quasi-convex optimization approach to parameterized model order reduction. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 27(3):456–469, 2008.

[87] K. C. Sou, A. Megretski, and L. Daniel. A quasi-convex optimization approach to parameterized model order reduction. In *Proc. of the IEEE/ACM Design Automation Conference*, CA, June 2005.

[88] K. C. Sou, A. Megretski, and L. Daniel. A quasi-convex optimization approach to parameterized model order reduction. ieee transactions on computer-aided design of integrated circuits and systems. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 27(3), Mach 2008.

[89] K.C. Sou, A. Megretski, and L. Daniel. Convex relaxation approach to the identification of the wiener-hammerstein model. In *IEEE Conference on Decision and Control*, Cancun, Mexico, December 2008.

[90] A. Soury, E. Ngoya, and J. Rousset. Behavioral modeling of RF and microwave circuit blocks for hierarchical simulation of modern transceivers. In *Microwave Symposium Digest, 2005 IEEE MTT-S International*, pages 975–978, 2005.

[91] J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999.

[92] B. Sturmfels. Polynomial equations and convex polytopes. *The American Mathematical Monthly*, 105:907–922, 1998.

[93] T. Stykel. Stability and inertia theorems for generalized lyapunov equations. *Linear Algebra and its Applications, volume=*.

[94] A.H. Tan and K. Godfrey. Identification of WienerHammerstein Models Using Linear Interpolation in the Frequency Domain (LIFRED). *IEEE Transactions on Instrumentation and Measurement*, 51(3):509–521, June 2002.

[95] S. Tiwary and R. Rutenbar. Scalable trajectory methods for on-demand analog macromodel extraction. In 42^{nd} *ACM/IEEE Design Automation Conference*, June 2005.

[96] S. Tiwary and R. Rutenbar. Faster, parametric trajectory-based macromodels via localized linear reductions. In *Proc. of the IEEE/ACM International Conference on Computer-Aided Design*, November 2006.

[97] D. Vasilyev. Theoretical and practical aspects of linear and nonlinear model order reudction techniques. P.hD. Thesis, Massachusetts Institute of Technology, February 2008.

[98] D. Vasilyev, M. Rewienski, and J. White. A tbr-based trajectory piecewise-linear algorithm for generating accurate low-order models for nonlinear analog circuits and mems. In *Proc. of the ACM/IEEE Design Automation Conference*, pages 490–5, June 2003.

[99] I.I. Verriest and T. Kailath. On generalized balanced realizations. *IEEE Transactions on Automatic Control*, 28:833–844, 1983.

[100] M. Vidyagasar. *Nonlinear Systems Analysis*. Prentice Hall, New York, 1978.

[101] D. S. Weile, E. Michielssen, Eric Grimme, and K. Gallivan. A method for generating rational interpolant reduced order models of two-parameter linear systems. *Applied Mathematics Letters*, 12:93–102, 1999.

[102] K. Willcox and J. Peraire. Balanced model reduction via the proper orthogonal decomposition. In *Proceedings of the 15th AIAA Computational Fluid Dynamics Conference*, Anaheim, CA, June 2001.

[103] J. C. Willems. Dissipative dynamical systems. *Arch. Rational Mechanics and Analysis*, 45:321–393, 1972.

[104] J. Wyatt, L. Chua, J. Gannett, I. Goknar, and D. Green. Energy concepts in the state-space theory of nonlinear n-ports: Part i-passivity. *Circuits and Systems, IEEE Transactions on*, 28(1):48–61, Jan 1981.

[105] Z. Ye, D. Vasilyev, Z. Zhu, and J.R. Phillips. Sparse implicit projection (sip) for reduction of general many-terminal networks. In *ICCAD '08: Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 736–743, Piscataway, NJ, USA, 2008. IEEE Press.

[106] T. Zhou and H. Kimura. Time domain identification for robust control. *Systems & Control Letters*, 20:167–178, 1993.