Efficient Modeling and Computation Methods for Robust AMS System Design

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Leandro Javier Gil

aus Buenos Aires

Hauptberichter:	Prof. Dr.-Ing. Martin Radetzki
Mitberichter:	Prof. Dr. Christoph Grimm
Tag der mündlichen Prüfung:	07.11.2018

Institut für Technische Informatik der Universität Stuttgart

2018

Erklärung

Hiermit versichere ich, diese Dissertation selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Stuttgart, 14.08.2018

Leandro Javier Gil

This dissertation is dedicated to my better half, big love and best friend, Susanne van Luijn, who has always been a constant source of support and encouragement during all the challenges of this work. This dissertation is also dedicated to my mother, Rosa Zaia, whose committment to success has taught me to work hard for the things that I aspire to achieve.

Acknowledgements

I would like to sincerely thank my dissertation supervisor, Prof. Dr. -Ing. Martin Radetzki, for his valuable guidance throughout this doctoral research, especially for his confidence in me and for giving me the opportunity to research in the field of embedded systems.

I also wish to thank Prof. Dr. Christoph Grimm for his valuable feedback on my publications and doctoral research as well as for evaluating my dissertation.

Finally I would like to thank all my colleagues at the University of Stuttgart for the collaboration and continuous support, in particular Sabine, Helmut, Lothar, Manuel, Jie, Gert, Marcus, Bastian, Rauf and Adan.

Contents

List of Figures

List of Tables

Acronyms

ADL	Architecture Description Language
AMS	Analog and Mixed-Signal
ASIC	Application Specific Integrated Circuit
CPS	Cyber-Physical System
DSL	Domain Specific Language
EDA	Electronic Design Automation
ESL	Electronic System Level
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
HLS	High-Level Synthesis
IMC	InTernal Model Control
ISA	Instruction Set Architecture
LMS	Linear Multi-Step
MNA	Modified Nodal Analysis
MoC	Models of computation
OEM	Original Equipment Manufacturer
OSA	Open Simulation Architecture
PBD	Platform Based Design
PWC	Piece-Wise Constant
RTL	Register Transfer Level
SiP	System in Package
SLD	System Level Design
SoC	System on Chip
TLM	Transaction Level Modeling
VHDL	Very High Speed Integrated Circuit Hardware Description Language

Abstract

This dissertation copes with the challenge regarding the development of model based design tools that better support the mixed analog and digital parts design of embedded systems. It focuses on the conception of efficient modeling and simulation methods that adequately support emerging system level design methodologies.

Starting with a deep analysis of the design activities, many weak points of today's system level design tools were captured. After considering the modeling and simulation of power electronic circuits for designing low energy embedded systems, a novel signal model that efficiently captures the dynamic behavior of analog and digital circuits is proposed and utilized for the development of computation methods that enable the fast and accurate system level simulation of AMS systems.

In order to support a stepwise system design refinement which is based on the essential system properties, behavior computation methods for linear and nonlinear analog circuits based on the novel signal model are presented and compared regarding the performance, accuracy and stability with existing numerical and analytical methods for circuit simulation.

The novel signal model in combination with the method proposed to efficiently cope with the interaction of analog and digital circuits as well as the new method for digital circuit simulation are the key contributions of this dissertation because they allow the concurrent state and event based simulation of analog and digital circuits. Using a synchronous data flow model of computation for scheduling the execution of the analog and digital model parts, very fast AMS system simulations are carried out.

As the best behavior abstraction for analog and digital circuits may be selected without the need of changing component interfaces, the implementation, validation and verification of AMS systems take advantage of the novel mixed signal representation. Changes on the modeling abstraction level do not affect the experiment setup.

The second part of this work deals with the robust design of AMS systems and its verification. After defining a mixed sensitivity based

robustness evaluation index for AMS control systems, a general robust design method leading to optimal controller tuning is presented.

To avoid over-conservative AMS system designs, the proposed robust design optimization method considers parametric uncertainty and nonlinear model characteristics. The system properties in the frequency domain needed to evaluate the system robustness during parameter optimization are obtained from the proposed signal model.

Further advantages of the presented signal model for the computation of control system performance evaluation indexes in the time domain are also investigated in combination with range arithmetic. A novel approach for capturing parameter correlations in range arithmetic based circuit behavior computation is proposed as a step towards a holistic modeling method for the robust design of AMS systems.

The several modeling and computation methods proposed to improve the support of design methodologies and tools for AMS system are validated and evaluated in the course of this dissertation considering many aspects of the modeling, simulation, design and verification of a low power embedded system implementing Adaptive Voltage and Frequency Scaling (AVFS) for energy saving.

Zusammenfassung

Diese Dissertation befasst sich mit der Herausforderung, modellbasierte Entwurfswerkzeuge zu entwickeln, die den Entwurf analoger und digitaler Komponenten eingebetteter Systeme besser unterstützen. Der Forschungsschwerpunkt der Dissertation liegt auf der Konzeption von Modellierungs- und Simulationsverfahren, die neu enstehende Entwurfsmethoden auf Systemebene ausreichend unterstützen.

Anhand einer tiefgehenden Analyse der Entwurfsaktivitäten wurden die Schwachstellen heutiger Systemebene-Entwurfswerkzeuge ermittelt.

Nachdem die Modellierungs- und Simulationsverfahren leistungselektronischer Schaltungen in energiesparenden eingebetteten Systemen dargestellt sind, wird ein neuartiges Signalmodell vorgeschlagen, welches das dynamische Verhalten analoger und digitaler Schaltungen erfasst. Es ermöglicht die schnelle und genaue Simulation der sogennanten AMS-Systeme auf Systemebene. Um eine schrittweise Verfeinerung des Designentwurfes auf Basis wesentlicher Systemeigenschaften möglich zu machen, werden Berechnungsmethoden für die Simulation linearer und nichtlinearer Schaltungen, die das neue Signalmodell nutzt, abgeleitet und bezüglich der Performance, Genauigkeit und Stabilität mit bekannten numerischen und analytischen Berechnungsverfahren der Schaltungstechnik verglichen.

Neben dem neuen Signalmodell sind das vorgeschlagene Verfahren zur effizienten Handhabung der Interaktion zwischen analogen und digitalen Schaltungen wie auch das entsprechende Verfahren zur Simulation digitaler Schaltungen Schlüsselbeiträge dieser Dissertation, da sie die gleichzeitige zustands- und ereignisbasierte Simulation analoger und digitaler Schaltungen ermöglichen. Unter Einsatz eines synchronen Datenfluss-Berechnungsmodells zur Bestimmung der Ausführungsreihenfolge analoger und digitaler Modellbestandteile werden sehr schnelle Simulationen von AMS-Systemen durchgeführt.

Da die am besten passende Verhaltensabstraktion für analoge und digitale Schaltungen ausgewählt werden kann, ohne dass eine Änderung der Schnittstellen erforderlich wird, profitieren die Entwicklungsschritte wie Implementierung, Validierung und Verifikation von der

neuen gleichzeitigen Darstellung gemischter analoger und digitaler Signale. Änderungen des Gesamtmodells oder des Simulationsaufbaus sind normalerweise nicht notwendig, um detaillierte Modellkomponenten einzusetzen.

Der zweite Teil dieser Arbeit befasst sich mit dem robusten Entwurf von AMS-Systemen und seiner Verifikation. Der Definition eines Mixed-Sensitivity-Indexes zur Robustheitsbewertung von AMS-Regelungssystemen folgt die Beschreibung einer allgemeingültigen Methode für den Entwurf robuster Systeme, die zu einer optimalen Einstellung der Reglereigenschaften führt.

Um überkonservative Entwürfe von AMS-Systemen zu vermeiden, zieht die vorgeschlagene Methode zur robusten Entwurfsoptimierung sowohl parametrische Unsicherheiten als auch die nichtlinearen Modelleigenschaften in Betracht. Um die nötigen Systemmerkmale im Frequenzbereich zu bestimmen, die zur Ermittlung und Optimierung der Systemrobustheit benötigt werden, wird das vorgeschlagene Signalmodell eingesetzt.

Weitere Vorteile des neuen Signalmodells für die Berechnung von Performance-Metriken, die zur Evaluierung von Regelungssystemen in der Zeitdomäne nützlich sind, werden in Kombination mit Bereichsarithmetik erforscht. Auf dem Weg zur Definition einer ganzheitlichen Robustentwurfsmethodik für AMS-Systeme wird eine neue Methode zur Betrachtung von Parameterkorrelationen in Schaltungsberechnungsverfahren basierend auf Bereichsarithmetik eingeführt.

Die verschiedenen Modellierungs- und Berechnungsverfahren, die eine Verbesserung der Entwurfsmethoden und -werkzeuge ermöglichen, werden im Verlauf dieser Dissertation validiert und bewertet. Dabei werden die Modellierung, Simulation, Entwurf, Analyse und Verifikation energiesparender eingebetteter Systeme auf Basis von Adaptive Voltage and Frequency Scaling (AVFS) betrachtet.

Chapter 1
Introduction

System level design (SLD) languages and tools are fundamental for efficient embedded system design, analysis, validation and verification. In order to adequately support emerging embedded system design methodologies, design tools must be continuously improved. Modeling methods supporting many abstraction levels and more efficient computation methods are needed for properly coping with contemporary system design activities. This dissertation proposes a set of novel modeling and computation methods that allow to efficiently carry out the simulation, design, analysis and verification of analog and mixed signal (AMS) systems for faster achieving robust and reliable designs. This chapter provides an outline of current productivity challenges in AMS system design and presents the contributions of this dissertation to the state-of-the-art in design tools for AMS systems. It is organized as follows. Section 1.1 describes the motivation of this thesis regarding the improvement of system level design tools' efficiency and modeling support. Section 1.2 states the main research goals. Section 1.3 indicates the dissertation structure and chapter contents. Section 1.4 presents the main contributions of the dissertation. Finally, section 1.5 states the main advantages of the novel modeling and computation methods.

1.1 Motivation

The engineering techniques required to design, analyze and verify embedded systems are sophisticated because many non-functional requirements need to be considered. For example, they have to respond sufficiently fast to their environment in order to perform a correct system operation. A careful timing analysis is necessary for a proper task execution [71]. Safety-critical embedded systems must guarantee a high level of reliability and predictability and battery-driven embedded systems must be power-efficient [54]. As a consequence of the large number of design constraints, optimization techniques are often needed for designing embedded systems [71].

A contemporary challenge in the design of embedded systems is the growing functionality and heterogeneity. More and more specialized systems are being interconnected [106]. System on chips (SoCs) integrate on a single chip a lot of analog and digital hardware devices [54]. The advances in packaging technology even allow the integration of many chips in a single package (System-in-Package, SiP). Thus, analog components requiring more processing steps can be implemented in a separated chip in order to improve the manufacturing process [106]. As a very large number of features can be implemented on a chip or package, the decision of the hardware functionality is mainly determined by non-functional constraints such as performance, cost, power consumption and reliability [106].

To cope with the complexity of embedded system design, the design abstraction level continues increasing [103]. Many commercial tools are available which directly generate embedded code, e.g. C or HDL, from abstract high-level models. As these tools utilize purely functional models such as Simulink models, it is not possible to seamlessly explore several architectural alternatives [131]. In particular, the complex interaction between digital and analog system components requires a careful low-level design for accomplishing system requirements and constraints. Due to the difficulty in representing the system behavior entirely, design verification is an additional challenge which leads to separated verification of analog and mixed signal systems.

Dealing with the contemporary AMS system design challenges is of central importance for bringing products to market in time with the required functionality and quality. Sangiovanni-Vincentelli speaks about a "novel engineering field" that is emerging for efficiently coping with the needs for a successful system level design [104] [103] [106].

In order to provide the needed efficiency for designing today's embedded systems, both design methodologies and tools must be improved. Design methodologies should capture the design components and constraints at several abstraction levels for a seamless system design refinement [103]. System level design languages and tools must be improved for better supporting emerging design methodologies. They should allow the modeling of heterogeneous analog and digital systems based on formal descriptions for supporting automated analysis and synthesis. Moreover, they should carry out fast model execution at multiple abstraction levels for enabling early design validation and verification.

1.2 Dissertation Goal and Objectives

The goal of the research presented in this dissertation was the development of modeling and computation methods that enable the fast simulation as well as the robustness evaluation of large heterogeneous analog and digital systems.

In order to develop design tools that better support the modeling, design, analysis and verification of analog and mixed signal systems, the impact of the modeling and computation methods on these design activities were considered in the scope of the research activities. The application domain was restricted to the design of AMS components in low power embedded systems. As several abstraction levels are required for designing such components, they provide the needed complexity for the evaluation of the novel system modeling and behavior computation methods.

1.3 Dissertation Outline

This dissertation is organized as shown in Fig 1.1. There are two parts that correspond to the research goals. The main chapters are structured as independent contributions presenting their particular motivation and related work and evaluating their findings to each research topic.

Chapter 2 introduces the state-of-the-art in AMS system design and indicates the support gap of design tools. It presents in section 2.5 a layered model, that captures the relevant system design activities that should be supported by design methodologies and tools.

The first part of this dissertation (chapters 3– 5) copes with the modeling and simulation of AMS systems. Chapter 3 describes the state-of-the-art in analog circuit simulation and introduces the modeling of power electronic circuits using ideal switches. It defines in section 3.6 a graph based method for finding inconsistencies and predicting electrical impulses after topology switching which is explained using a DC-DC power converter. It also proposes SystemC AMS extension for the simulation of power electronic circuits at system level. Chapter 4 presents in section 4.5 a novel mathematical model of signals that efficiently captures continuous time and discrete event component behaviors as well as a signal subdivision method in section 4.6 which is

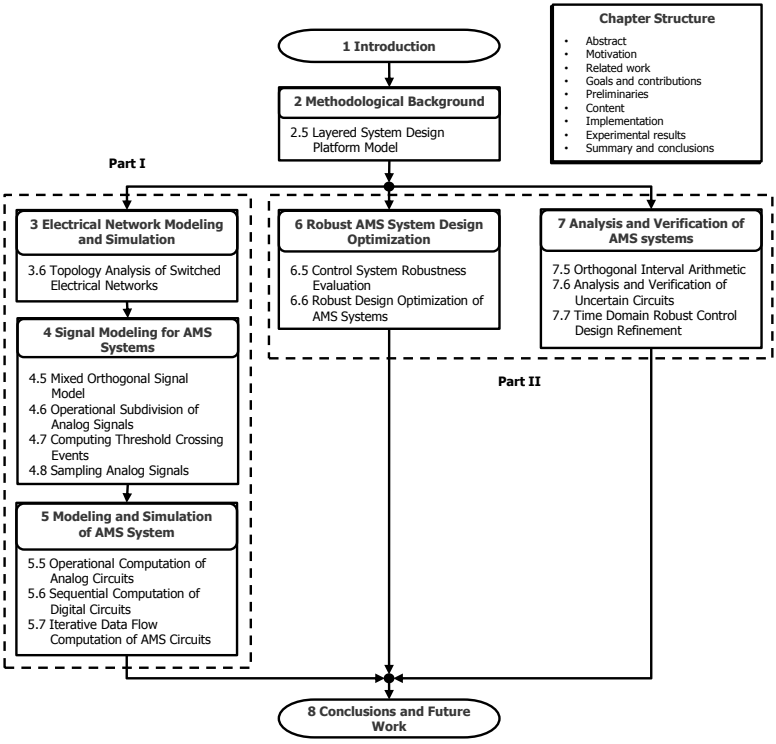


Fig. 1.1: Dissertation structure

utilized for efficiently handling the mapping of continuous time signals into discrete event signals. Section 4.7 and 4.8 explain the application of the signal subdivision method for threshold detection and signal sampling respectively. Chapter 5 utilizes the novel signal model for the efficient simulation of AMS systems. It derives in section 5.5 operational behavior computation methods for analog circuits at different abstraction levels and presents in section 5.6 a state based digital system modeling and behavior computation method. After that, it proposes in section 5.7 an iterative data flow model of computation which relies on the developed computation methods for achieving fast and accurate AMS system simulations in SystemC AMS.

The second part of this dissertation investigates the impact of the novel modeling and computation methods on the design, validation and verification of AMS systems. Chapter 6 proposes in section 6.6 a control system design optimization method that does not impose restrictions on the controller structure and enables the improvement of the control system robustness with respect to disturbances and manufacturing tolerances. Moreover, it defines in section 6.5 a robustness evaluation index which is applicable to linear and nonlinear control systems and allows the simple comparison of several designs as well as the fine tuning of the design parameters. Chapter 7 investigates the properties of several range arithmetic methods in order to carry out faster and more accurate circuit tolerance analysis. It defines in section 7.5 a novel range arithmetic approach that keeps the correlation between parameters in vectorial form. It presents in section 7.6 operational computation methods for analog system design analysis and verification and proposes in section 7.7 a cost function which is well suited for robust control design refinement in the time domain.

Finally, chapter 8 states the main contributions of the dissertation and proposes topics for future work.

1.4 Contributions to AMS System Design

This dissertation contributes to the state-of-the-art in AMS system modeling and simulation by the following:

1. A generic method for predicting topology inconsistencies and electrical impulses in switched electrical networks which enables the fast simulation of power electronic circuits (3.6).
2. A novel formal signal model that efficiently captures the continuous time and discrete event behavior of AMS systems' components (4.5).
3. An operational matrix for arbitrary analog signal subdivision which enables fast and accurate threshold crossing events' detection (4.6).
4. A set of analog circuit behavior computation methods based on the proposed signal model which allow efficient and accurate analog component simulations at several abstraction levels (5.5).
5. A state based digital system modeling and computation method which supports the proposed signal model to enable the fast AMS system simulations (5.7).

Moreover, this dissertation contributes to the state-of-the-art in AMS system design tools by the following:

1. A normalized robustness evaluation index which is applicable to linear and nonlinear control systems and allows the simple comparison of several AMS system designs (6.5).
2. A fast and reliable approach for computing the plant uncertainty characteristics due to parameter tolerances (6.6.1).
3. A novel operational matrix of multiplication which enables the efficient computation of common control system evaluation indexes for fast AMS system design analysis and refinement (7.6.3).

1.5 Paradigm Change in AMS System Simulation

Based on the mentioned contributions, this dissertation proposes a novel way of modeling and computing the AMS system's behavior. As shown in Fig. 1.2, instead of carrying out iterative scalar value or event computations at single time points, the behavior computation takes place on a value matrix or event sequence over large time intervals. The key advantages of the novel AMS system modeling and computation methods are that they enable:

1. fast and accurate simulations
2. multiple concurrent behavior abstraction levels
3. fast analog signal analysis and tracing

These properties lead to a seamless design of AMS systems that together with the novel computation methods for performance and robustness evaluation allow a fast robust design optimization and refinement which consider both disturbances and manufacturing tolerances.

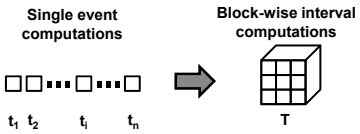


Fig. 1.2: Paradigm change in AMS system simulation

Chapter 2
Methodological Background

Methodologies and tools for embedded system design contribute significantly to the reduction of the time-to-market as well as to the minimization of design errors. The understanding of the system design principles is fundamental for the appropriate selection of system design methodologies and for the development of powerful design and analysis tools. This chapter presents a simple and general model that organizes the essential steps of the *model based system design* into a set of inter-dependent layers. It captures the relevant design activities allowing the representation of existing design methodologies and tools into a unified view. This abstract representation of the system design process allows the systematical reasoning about the key characteristics and dependencies of the design activities and serves as methodological background for deriving modeling and computation methods that better support the system design activities. In particular, some issues that currently limit the support of design tools across several design activities such modeling, analysis and verification for AMS systems are identified. Furthermore, the corresponding contributions of the dissertation chapters to the AMS system design topics are ordered using this model.

This chapter is organized as follows. After the presentation of the research motivation and related work in section 2.1 and 2.2 respectively, the contributions of this chapter are summarized in section 2.3. Section 2.4 explains the current problems of design tools for AMS systems which lead to a productivity gap. Section 2.5 describes the *Layered System Design Platform Model*. Finally, section 2.6 summarizes how the several layers of the presented system design platform model contribute to the development of better design methodologies and tools indicating the weak points of current design tools.

2.1 Motivation

The continuous advances on computer systems in the past decades have contributed to the increasing popularity of model based engineering methods. System specification, design and validation can be carried out at higher abstraction levels using model centered development approaches [100]. In addition, simulation techniques allow to cope with design problems that can not be worked out analytically [129].

The increasing demand for reducing time-to-market, cost and design errors in embedded system design requires that the essential design steps are adequately supported by design methodologies and tools across the whole design process [106]. A seamless design chain based on the system properties at the several design steps is needed for improving design productivity and quality [106].

A fundamental issue in the design of AMS systems is that current design methodologies do not utilize appropriate abstractions for capturing the system properties [103]. In order to achieve correct, efficient, reliable and robust electronic designs, a rigorous specification and characterization of system properties at several abstraction levels is necessary [105]. Capturing the properties of AMS systems' functionality in a formal way (mathematically) is a prerequisite for enabling the efficient design correctness verification and the error-free design refinement towards the final system implementation [105]. It allows the automatic system design refinement that meets design constraints and optimization criteria from one abstraction level to the next [105].

The early validation and verification of AMS systems' functionality and performance requires design tools that allow the heterogeneous behavior specification of analog and mixed signal parts and support the fast model execution. As design languages and tools must be based on formally defined models for effectively supporting design activities, it is fundamental to understand the required model properties at the several steps in order to achieve a better design support.

2.2 Related Work

Focused on the simulation of discrete event systems (DEVS), Zeigler defined an extensive theoretical background for modeling analog and digital systems [130]. In order to understand and compare the properties of the modeling formalisms utilized in embedded system design, Lee proposed a denotational framework called the *tagged signal model* that captures the concurrency and communication behavior of frequently used computational models [70] [69] [68].

Based on a clear definition of model execution and interconnection semantics, Sander et al. proposed a formal system development process for performing design refinement [102]. They organized models of computation with respect to time abstraction and remarked the importance of separating computation and communication for better fit the right model abstraction [54]. They also implemented a design language named ForSyDe which provides design transformations for enabling the automatic design synthesis from embedded system models [101].

For better explanation of a modeling and simulation environment supporting decision making, Zeigler presented a layered representation of the tool functionality which captures the necessary problem solving activities [129]. Dalle et al. extended Zeigler's layered tool architecture representation for promoting the collaborative development of DEVS simulation platforms. The proposed *Open Simulation Architecture* (OSA) defines standardized interfaces for integrating modeling, simulation and analysis tools [20] [19] [21] [22]. Vachoux et al. utilized a similar layered structure for enabling the simple definition and integration of computational models in SystemC AMS [119] [118].

Due to the increasing complexity and heterogeneity of contemporary AMS systems, the selection and integration of design methodologies and tools becomes a challenge. It requires a good understanding of the entire design task. In particular, the impact of the modeling and simulation capabilities on the essential system design activities need to be better considered. An abstract representation of the design task needs to be worked out for the development of more efficient design methodologies and tools.

2.3 Representing a Seamless System Design Chain

The aim of the work presented in this chapter was the definition of a methodological framework that captures the essential model based design steps (from specification to implementation) and the relevant model properties for the development of more efficient AMS system modeling and behavior computation methods. To this end, it was necessary:

- To investigate which are the most relevant properties that should be captured for adequately modeling AMS systems.
- To analyze how design tools supporting heterogeneous analog and digital system modeling and simulation are built.
- To find out which design activities are not well addressed by current design methodologies and tools.
- To capture design activities dependencies and interaction.
- To analyze the properties and limitations of the modeling and simulation approaches for design analysis and verification regarding performance, accuracy and stability.

As result of these activities, a methodological framework which focuses on describing the key model properties required at the several system design activities was elaborated. For better matching design methodologies and tools, the identified model properties were organized in several layers. This system design model extends Zeigler's layered architecture representation for modeling and simulation by:

- Considering further activities that are required for achieving a correct, efficient, reliable and robust embedded system design.
- Providing a unified view of design methodologies and tools.
- Describing the system properties that should be captured at each layer during model based system design.
- Taking into account the interaction of system model with both, the simulation software and the hardware execution platform.

As the dependencies between the design activities and the model properties were captured and analyzed regarding their impact on system design correctness and efficiency, this chapter defines a methodological background for the development of efficient design tools that meet contemporary design support requirements for AMS systems.

2.4 Embedded System Design Methodologies

Design is a *transformation process* that based on a specification creates a product [106]. The *design methodology* describes the way in which the design process is organized. It specifies the steps and checks that must be carried out during the design process [106]. Different methodologies are used for system design [106]. *System-Level Design* (SLD) is a design methodology that assembles components to a whole system [103]. In order to deal with system complexity, *top-down decomposition* into subsystems is usually applied. This widely utilized design technique requires a full understanding of the entire system for achieving an appropriated system partitioning. This is a difficult task because the complexity of today's embedded systems is very high [106]. *Bottom-up system design* is a design method that starting with a set of pre-existing designs, modifies or extends these solutions to respond to a set of (new) application requirements [106]. Developing systems by composing already pre-designed subsystems (vertical design chain) helps to reduce time-to-market and design costs [106]. As already working systems are utilized as initial solution, this design technique reduces the system implementation risks. However, it becomes often difficult to meet all system requirements and to achieve an optimal system design using this design method [106]. *Electronic System Level* (ESL) is a design methodology that emerged to cope with the complexity of emerging system on chip (SoC) designs [35]. This technique describes the electronic system using a set of abstractions (at different levels) that allows the conceptualization of the chip design [107]. Depending on the elements of the design abstraction, several ESL methodologies were proposed. *Function based ESL* describes the embedded systems utilizing formal computational models which represent the relations between input and outputs [132]. *Architecture based ESL* utilizes a set of components (processors, memories, peripherals, etc.) that describe the computer organization [132]. *Function-Architecture based ESL*, refines first the design requirements into a functional model (such as in functional based ESL) and then into an architectural model (such as in the architecture based ESL). Although this co-design methodology is particularly suitable for architecture exploration, the system architecture is top-down designed for a particular function or application. For this reason, it becomes difficult to extend the architecture design to support multiple applications.

In order to reduce design and implementation costs, system-on-chips (SoCs) are typically designed as a platform that provide support for many applications [132]. A *platform* is an abstraction characterized by a set of components. It allows a parametrization of the possible solution space for a given level of abstraction [106]. The purpose of a platform is to provide a common design that can be re-used or adapted to several domain specific applications. *Platform based ESL* design combines top-down and bottom-up design methods [106]. The *design functionality* is mapped (top-down) into a *platform instance* propagating implementation constraints. The *platform instance* is built (bottom-up) by selecting *library components*. For systems designed using a domain-specific computational model (function-based design), the best suited architecture is normally known and represented as *architecture template*. As the architecture instance can be automatically parametrized, the manual design refinement is not necessary [132]. Design reuse is guided "from the perspective of a *reference system architecture*" [107]. *Platform based design* (PBD) defines a *standard layer of abstraction* that separates *system functionality* and *architecture*, hiding unnecessary implementation details [106] [103]. This design technique allows the refinement of the system functionality at several levels of abstraction (using the same workflow) that is the main idea of the Y-chart proposed by Gajski [32].

2.4.1 The Design Productivity Gap for AMS Systems

Electronic design automation (EDA) tools provide a set of processes and guidelines that describe the *system design flow* as well as features for behavioral synthesis and design verification. They enable the early design space exploration of high-level system designs and the increase of the design productivity for high complex integrated circuits [132]. As "design productivity decreases exponentially with respect to technology advances" [105], the efficiency of design methodologies and tools must be continuously improved for addressing the emerging challenges in the system design chain [106]. Metrics describing system properties such as cost, weight, size, power dissipation and performance should be utilized for optimally guiding the design [105]. EDA tools that work efficient and with guaranteed correctness through different abstraction levels from the conceptual model to the silicon implementation and

packaging are needed. As the number of analog and mixed signal components in electronic systems continues increasing, the maturity level of today's design tools is not proper for system integration [106]. In order to reduce design costs, companies designing electronic systems are demanding design methodologies and tools that enable a reliable integration of pre-designed components into large chips (design reuse) as well as a better verification and validation support that considers robustness and reliability requirements [106]. The poor abstraction level in the analog and mixed signal design makes difficult the harmonized system representation and leads to segregated design methodologies for analog and digital hardware [107]. In order to improve system design exploration, refinement, synthesis and verification both, a unified design methodology as well as suitable tool supporting combined analog and digital design are required [106].

The major problems that difficult the analog design at system level are the poor behavior abstraction and the low simulation performance of the modeling and computation methods utilized in today's AMS design tools. In order to evaluate the impact of modeling and computation methods on AMS system design productivity, the hardware component design of a low-power embedded system is considered in the course of this dissertation. It utilizes both Dynamic Voltage Scaling (DVS) and Dynamic Frequency Scaling (DFS) for energy saving as shown in the block diagram in Fig. 2.1. The DVFS system estimates the processor workload and adjusts its supply voltage and operation frequency for minimizing power consumption. The voltage and frequency control parts require a careful design because they are subject to a lot of non-functional implementation requirements and constraints.

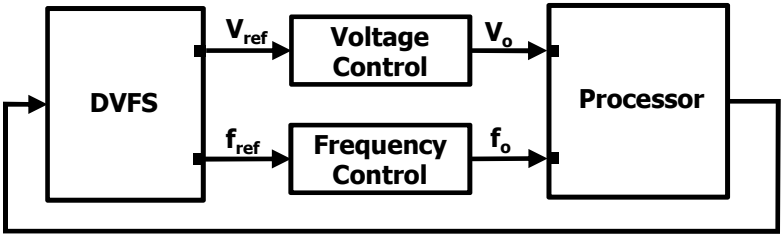


Fig. 2.1: DVFS system block diagram

2.5 Layered System Design Platform Model

In the context of this work, a *model* is an abstract representation of a *physical system* that captures knowledge or information about a system for specification, design, analysis, verification or implementation purposes. A physical system is composed by a set of parts (called components) which jointly interact with the system environment for providing or enabling a self contained functionality. A *design activity* (or design step) is an action performed by a designer during design of a system, possibly following a *design methodology* and ideally utilizing tool support [98].

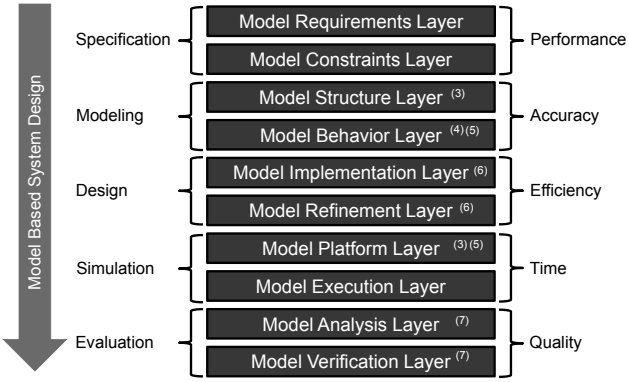


Fig. 2.2: Layered system design platform model

As pointed out by Sangiovanni-Vincentelli [106], design tools may be considered as a platform. This idea corresponds with Zeigler's layered design tool representation and the recently proposed open simulation architecture (see related work in section 2.2). With the aim of capturing the purposes and properties of embedded system models at the different design stages, this work proposes a conceptual design platform representation which describes in 10 layers the relationships between embedded system design methodologies and tools. It is shown in figure 2.2 and may be considered as a more contemporary model of Zeigler's design tool representation. It serves as guide for understanding how the several contributions of this work may impact on the improvement of

design tool's productivity. The corresponding chapters are indicated in brackets. This layered system design platform representation is a conceptual model that aims to represents the main steps which are usually explicitly or implicitly carried out during the system design process. **It does not define nor recommend a particular workflow for designing embedded systems.** The model layers are organized as pairs of complementary activities. Each layer of the model involves different features that are needed for supporting the system design activities and extends or utilizes the functionality of previous layers. The main impact of the several activity pairs on the system development is represented by the properties on the right hand side of the figure 2.2.

Example 2.1. Consider the design of the AMS system shown in Fig. 2.1. The model requirements layer captures the desired behavior of DVFS system, that is: depending on the current task the power-consumption of the processor is minimized by adjusting the power supply and operation frequency. The model constraints layer captures the properties and constraints that guide the design implementation. They are: the voltage and the frequency must be stably adjusted within a defined time slot (performance requirements) as well as the corresponding voltage ripple and frequency jittering must be smaller than a given value. Additional non-functional properties such as thermal dissipation, size and cost must be minimized. The system components must be robust respect to manufacturing variations and external perturbations (design criteria). The model structure layer supports the definition, interface declaration and interconnection of the system parts as well as the top-down hierarchical decomposition. The model behavior layer provides a set of modeling abstractions (transfer function, electrical network, etc.) which enables the representation of the system components' behavior with different levels of accuracy during design. The model implementation layer enables the specification of implementation details such as the behavior computation methods and parameters which allow the model simulation as well as component constraints which allow the system design refinement. In best case, it also includes a library of pre-designed components. The model refinement layer provides tools that select and parametrize library components (if component models are avail-

able) or support the refinement of abstract design models towards the final implementation. The model platform layer defines and implements a simulation platform which enables the system behavior computation during design and analysis. The model execution layer maps the entire system model (including behavior computation algorithms) on the target platform (single or multicore). The model analysis layer allows the fast simulation of a mission profile subset and the evaluation of the system design performance at several operating points. It should support the design optimization in terms of the defined properties. The model verification layer should provide enough support for the verification that system design and implementation models fulfill the imposed requirements and constraints e.g. the specified system performance is reached and the supplied voltage and frequency fulfill the defined ripple and jittering constraints. A deep verification considering parameter tolerances and corner cases should be possible.

The following sections describe the design platform model layers, indicating the state-of-the-art and the contributions of the several dissertation chapters to the open issues.

2.5.1 Model Requirements Layer

A *system requirements specification* is a prerequisite to start the system design. It specifies "the effects that a system is required to achieve" [15] and should completely define "what the system is supposed to do" [105] but any design or implementation detail that unnecessarily constrains system design should be omitted [15]. The *Model Requirements Layer* collects the system design requirements which state the externally observable behavior (system functionality).

A good design methodology as well as development tools that define and support the requirements management activities are very important for the mitigation of requirements ambiguities, inconsistencies and traceability problems during the system design process [105].

2.5.2 Model Constraints Layer

The system requirements specification should be accompanied by a set of properties and constraints that the system design has to fulfill as well as by a description of the environmental and regulatory conditions [105]. The *Model Constraints Layer* collects the set of system properties and constraints which guide the design implementation and optimization. The set of system constraints includes constraints on the system behavior and on the system implementation characteristics.

The *system behavior constraints* should be described as a set of equations and inequalities for simplifying its verification (property assessment and relationship evaluation) [105]. The behavior of the system environment cannot be predicted. This non-determinism should be described in the *environment specification* of the system under design but it is not a design property [105]. A set of *design criteria* such as robustness, reliability, etc. should be also specified for guiding system design [105]. Chapter 6 introduces the robustness specification and evaluation problem and proposes a suitable property set for control systems.

The *constraints on the system implementation characteristics* indicate the components to be used or define requirements on the system implementation that are expressed in terms of physical quantities such as power or timing [105]. The so called *reaction requirements* specify the interaction of the system with its environment [47] [46]. They are usually described in terms of equalities or inequalities that involve the *design variables* and the *implementation characteristics* [105]. As the reaction requirements allow the verification if a given architecture ("how the system does what is supposed to do") is feasible, they guide the system architecture selection (platform instance) [103] [105]. The so called *execution requirements* define constraints on the characteristics of the implementation platform elements such as e.g. the battery capacity or the channel bandwidth [105].

The *system specification*, including system functional requirements, properties, design criteria and constraints as well as the system environment characteristics has a significant impact on the final system performance and quality. Chapters 6 and 7 define behavior properties and design criteria that are suitable for guiding and refining the design of AMS control systems to be robust to external perturbation and parameter tolerances.

2.5.3 Model Structure Layer

The aim of modeling is to describe systems. The model based system design starts with the construction of a *specification model* in a declarative or executable language, according to the design requirements defined in the *system requirements specification* [54] [107]. At the start of system design, the specification model is constructed by interconnecting components which describe the different parts of the system [129]. The interaction of the system with its environment is usually represented as a model part. This *structural model* captures *static information* about the system construction [71]. It is described by a structure that specifies a *set of components* a *set of interfaces*, (including the interfaces to the system environment) and a *set relations* among components and interfaces (interconnection or coupling).

The *Model Structure Layer* provides a functionality for representing the system under design from a physical or logical view point by interconnecting model components i.e. defining components and their interconnection relations. It captures the system architecture. In order to model complex systems, *hierarchical decomposition* is applied to the specification model breaking down the represented system into smaller systems. *Atomic components* are successively replaced by *modular components*. In order to hide details about the system structure, *hierarchical composition* is applied to the specification model. The different components and the relations along their interfaces are encapsulated in a *modular component* (also called *coupled model* or *subsystem*) which also represents a system [129]. For carrying out the composition of the model components, well defined interfaces to the component environment are needed [68]. Hierarchical decomposition and composition are essential features that capture different levels of detail and support top-down and bottom-up design.

A weak point of most modeling tools is that the underlying syntax utilized for analog behavior description does not capture the system structure i.e. systems are represented directly at equation level. Using an *abstract syntax* (which is not bound to a specific interpretation) for representing the structure of a model is becoming necessary in order to capture different behavioral and physical domains [106]. Chapter 3 utilizes a modeling approach based on clustered graphs for the efficient analysis and equation formulation of switched electrical networks.

2.5.4 Model Behavior Layer

The *Model Behavior Layer* aims to capture the system behavior in an appropriate manner for system design analysis, implementation, validation and verification. The modeling approach for representing the system functionality should enable the analysis, processing and transformation of the specification model. As AMS systems are usually reactive systems that respond continuously to their environment, the resulting *behavioral model* must enable the computation of the system state evolution in time (system dynamics) [71]. The behavior of a structural (or coupled) model is computed from the behaviors of its components and the relation among their interfaces (coupling specification) [129]. Every model component is an entity with a given behavior expressed in terms of a *set of variables* and a *set of parameters*. The time constitutes an independent variable for all model components.

There is normally more than one representation for a given system (or modular component). In order to reduce the modeling complexity, the *system design functionality* should be captured at the highest possible abstraction level [105]. This requires that the systems' behavioral characteristics are reduced to a set of *essential properties*. There are many formalisms, that are able to capture a given system design specification. These abstract formalisms of the system behavior are called Models of Computation (MoCs). They govern the interaction across model components in the system design [68].

A *model of computation* (MoC) is a *mathematical description* of a system that has a syntax and specifies the semantics of the computation and communication behavior described by the syntax [105]. This architectural pattern defines abstract components and their interaction (operational relationships between components) [68]. The syntax of a model of computation defines the symbols that may be used for describing a system as well as the valid composition of its parts. It allows to specify structurally correct models. The semantics of a model of computation defines the rules that "give a meaning to the system behavior" [105]. The semantic rules of a model of computation define three fundamental characteristics [105]:

- the component actions (computations)
- the coordination of the component actions (concurrency)
- the exchange of components data (communication)

As the semantics of a model of computation is formally defined, it unambiguously captures the system design specification avoiding interpretation mistakes. A key feature of MoCs is the possibility to execute the design functionality for analysis and verification. This allows the early finding of design errors and the validation of the system functionality [105]. Models of computation have a rich set of intrinsic properties which determines the particular class of systems that they can suitably model, e.g. control-dominated or data-dominated systems. A trade-off between MoC properties is normally needed, e.g. making a MoC more expressive can lead to a very difficult or even to an impossible behavior verification. In particular, the communication semantics of a MoC has a significant impact on its properties [106].

In order to overcome modeling and design limitations, different MoCs are usually combined for system design specification, keeping verification and synthesis separated for the heterogeneous parts of the system [105]. The appropriate MoC selection and the integration of the heterogeneous model components which are part of AMS systems need a lot of system knowledge. Moreover, different MoCs are needed to represent the system properties at several phases in the system design process [54]. Such heterogeneous system models require a clear definition of the execution and interconnection semantics between the different MoCs. This is a challenging task, because the semantic of several MoCs may be fundamentally different [54]. Heterogeneity handling can take place implicitly or needs the manual inclusion of domain interfaces. In order to provide a coherent heterogeneous system modeling, several approaches for the integration of different MoCs are utilized in design tools [54]:

- Common refinement of heterogeneous MoCs.
- Hierarchical composition of heterogeneous MoCs.
- Direct composition of heterogeneous MoCs.

In the common refinement of heterogeneous MoCs (called framework specialization), the different MoCs embedded in a modeling language are restricted to a common MoC, obtaining thus a homogenous MoC model which deals with several dynamic behaviors. Modeling languages such as Simulink and VHDL-AMS utilize this approach to handle continuous time (CT) and discrete event (DE) systems. A common signal model is utilized to interconnect components.

Heterogeneous frameworks, such as Ptolemy II, utilize a finite state machine for the hierarchical composition of heterogeneous MoCs. Each state of the finite state machine may utilize a different MoC [39] [106]. As each level of the hierarchy is a homogenous system, this method is not very suitable for representing mixed signal systems such as e.g. analog-to-digital converters.

Coordination languages provide special interfaces for carrying out a *direct composition* of heterogeneous MoCs. These *domain interfaces* allow a better modeling of heterogeneous systems (mixed signal systems are more naturally described) but they are quite difficult to define. Each model of computation is embedded in a domain. SystemC AMS and SystemC ForSyDe implement domain interfaces for handling heterogeneous MoCs.

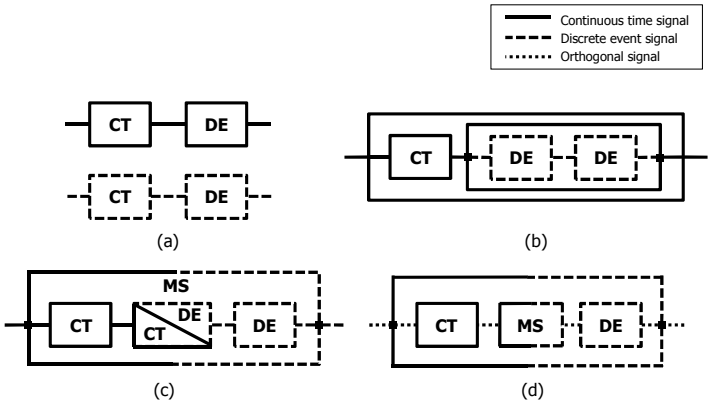


Fig. 2.3: Heterogeneity handling: a) MoC specialization b) Hierarchical MoC composition c) Direct MoC composition d) Orthogonal signals

In order to avoid the challenging task of having to design domain interfaces for each combination of MoCs, Ptolemy II utilizes domain polymorphic components. This modeling method requires the definition of an abstract interface that is as unspecific as possible, e.g. based on non-deterministic automata [68]. Furthermore it is necessary to explicitly define domains for each model of computation. Ptolemy II utilizes special blocks called directors for this purpose.

Jantsch et al. applied *composite data-flow* for integrating data and control flow [53]. This approach captures the timing effects of the domain conversions without resorting to a synchronous or timed MoC. The timing of computational processes is represented at the abstract level of determining if sufficient data is available to start a computation. Thus, the effects of control and timing on data-flow processing are considered at the highest possible abstraction level (i.e. as a data dependency problem) [54].

This dissertation presents in chapter 4 a novel method for signal modeling that enables the representation of heterogeneous behaviors. As shown in Fig. 2.3, orthogonal signals allow the direct composition of several MoCs without the explicit definition of domain interfaces. Avoiding multiple signal representations, this signal modeling method enables the implementation of polymorphic components that change the MoC during design process according to the required properties for refinement or analysis. The redefinition of interfaces is not necessary. Furthermore this approach is significantly more efficient and accurate.

Although a high level of modeling abstraction is important for handling system complexity, the suitability of the MoC primitives for representing the system properties is also crucial in order to efficiently carry out design and analysis tasks. The most suitable system abstraction for coping with a given design task is defined by the design methodology or manually chosen by the developer. According to general behavioral relations or rules which are common to all components of a given model (or modular component), it is possible to define specific behavioral domains which impose variables and inter-component relations. Physical domains like electrical, mechanical or thermal describe systems through *physical variables* such as current, velocity, heat flow, etc. The set of possible variables which may be used to describe the behavior of a given physical component is thus restricted to a subset of *domain variables*. In physical system modeling, the set of relations across system components and their environment must respect physical laws. For example, electrical components must meet Kirchhoffs' laws. Using *domain specific components*, the model correctness can be ensured by construction and the modeling effort reduced. This dissertation focuses on the modeling of power electronic circuits and presents in chapter 3 a modeling abstraction that allows fast and accurate simulations.

2.5.5 *Model Implementation Layer*

As relevant system characteristics are missing, abstract models describing the system functionality are not suitable for implementation purposes [54]. Each possible system implementation must provide the specified functionality but it also has to meet the required performance and implementation constraints [106]. The *Model Implementation Layer* aims to include sufficient modeling details for enabling the stepwise system implementation and verification with respect to all requirements and constraints. It should be possible to realize the desired product as suggested by the *implementation model*. In order to increase the productivity and minimize errors, design tools should provide libraries which include a rich set of component implementation models. Thus, the design space is reduced by specifying a *static configuration* (selection of components and connectivity) and a *dynamic control configuration* [105]. The choice and configuration of predefined system components is called *platform instance*.

The computation of the model behavior for analysis or verification usually needs the configuration of a *simulation platform*. For instance, the simulation of analog components requires the specification of an algorithm which is utilized for the numerical computation of the differential equations that describe the circuit behavior. This algorithm choice is complemented by the specification of execution constraints such as e.g. the integration step size or the allowed computation error. Assuming that the Backward-Euler algorithm is selected, then this platform instance is the functional specification for the next layer which implements the algorithm (the model platform layer). The full implementation of the specified algorithm may require further choice of components such as the linear equation solver, for running on a particular computing platform (carried out by the model execution layer) [106]. Taking into account that each level of the behavior computation algorithm has an impact [1] on the achieved simulation performance and accuracy, this dissertation presents and analyzes in chapters 3, 5 and 7 the characteristics of several simulation platforms instances.

[1] The impact of the algorithm mapping into the available computation resources on the simulation performance is considered in the Model Execution Layer

2.5.6 Model Refinement Layer

The specification model should be sufficiently abstract to enable the implementation of its functionality in many ways, i.e. it should not over-constrain the system implementation [54]. The higher is the abstraction level of the implementation model (fewer implementation details), the larger is the number of possible implementations (design space) that fulfill the behavior defined by the model [54]. This is illustrated in Fig. 2.4.

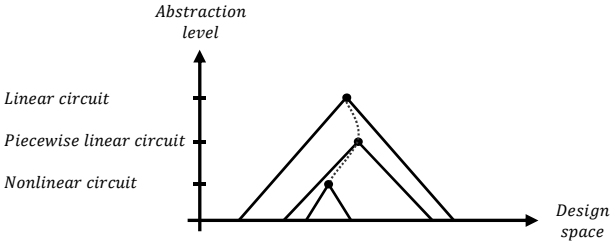


Fig. 2.4: Design space depending on the model abstraction level

As functional requirements and implementation constraints often demand different levels of abstraction, different models (and possibly different MoCs) are usually utilized for system specification and implementation purposes [54]. Starting with a specification model that includes a few implementation details for enabling a fast verification, the design process usually proceeds with a step-wise model refinement which results in an efficient system implementation [54].

Depending on the design and verification goals, different abstraction levels may be used for modeling power electronic circuits:

1. Linear electrical network (transfer function or state space equations).
2. Electrical network with ideal components (piece-wise linear).
3. Electrical network with detailed components (often nonlinear).

For controller design a mathematical description of the electrical network in form of a transfer function (or state space model) is commonly used. This abstract model describes the electrical network as a linear functional system which can be analyzed in the frequency domain

(Laplace transform). This approximation is only valid for small signal behavior. The analysis of the switching response can not be carried out at this abstraction level (no harmonics).

For circuit design and controller optimization tasks, a time domain mathematical description of the power electronic system is required. The electrical network may be represented using linear components and switches which is shown in chapter 3. This abstract model is valid for large signal behavior and it is utilized for the evaluation of the overall system performance. Voltage and current waveforms of different system parts may be analyzed.

For circuits component choice and design verification, a detailed mathematical model of circuit components including manufacturer specific characteristics is required. Nonlinear component models are normally utilized for modeling and analysis of parasitic effects, switching transitions and component stress at the last design stages.

This dissertation proposes in chapters 3 and 5 efficient computation methods for analog circuit simulation at all these abstraction levels.

Although the component selection can be satisfactorily carried out by experienced designers, the refinement and optimization of the implementation model should be done automatically using efficient tools in order to obtain an optimal implementation and to reduce the implementation time [105]. The aim of the *Model Refinement Layer* is to provide design support to automatically achieve a correct and optimal final system design implementation by refining the corresponding specification model. As the models of computation needed for specification and implementation purposes are usually different, the Model Refinement Layer should also provide mapping rules between different models of computation that utilize the constraints and directives defined by the designer for controlling the synthesis. The properties of the implementation model after a synthesis step should be defined by the design methodology. The definition of abstraction levels at which the mapping and refinement processes take place is an important methodology decision. In particular, the performance indexes that characterize the architectural components should be chosen to enable a better component reuse.

The robust control design method proposed in chapter 6 maps the complex IMC controller derived from the plant transfer function into a classical PID controller, keeping the frequency response characteristics. Linear and nonlinear circuits are utilized for design refinement.

Based on performance indexes and constraints, the system design is refined following well defined steps toward the final implementation [103]. This requires the availability of models that capture the system properties at the defined abstraction levels (e.g. logic function, boolean network, electrical circuit, etc) [105].

As there are typically many alternatives for refining a specification model into an implementation model, optimization techniques are needed to find a good trade-off between interrelated implementation criteria such as hardware area, software code size, system performance, power dissipation, etc. [98]. This dissertation proposes in chapter 6 a robust control design optimization method that based on the desired controller bandwidth and the parameter tolerances of the power electronic circuit, reliably refines the control system implementation for obtaining an optimal robust design.

The design methodology should properly capture the platform characteristics and define several platform abstraction levels depending on the size of the design space to be explored and on the accuracy needed for the estimation of the defined characteristics [103]. Using a common description of the system target platform (platform API) the system implementation can be achieved by properly interconnecting library components (mapping onto the platform services) and the performance of several designs can be faster explored [105]. This dissertation considers the exploration of several robust controllers based on the proposed mixed sensitivity robustness index. [2]

2.5.7 *Model Platform Layer*

The mapping of the implementation model into software running on target processors requires an *architectural specification* of the system under design [107]. The *hardware platform model* captures the architectural specification needed for the analysis and verification of the whole system design. For embedded systems, this *hardware platform model* is typically built utilizing hardware components such as processors, memories, buses and non-programmable hardware components. The models of computation used for modeling the system functionality can

[2] The definition of a power supply platform for DVFS is outside of the scope of this work.

be also utilized for the representation of hardware platforms and their instances [105].

For the computation of the system behavior, both the system specification model and the hardware platform model require an interacting engine that executes the (overall) system specification. The *Model Platform Layer* deals with the creation of a *simulation model* for system design analysis and verification. To this end, a *simulation engine* that provides appropriate methods for managing model execution is linked to the system model. It schedules the model execution and synchronizes the simulation data. This *abstract machine* that executes the system model based on the semantics of a computational model defines a *simulation platform*. Given an appropriate input trajectory, the resulting output trajectory (model behavior) is computed by executing the abstract machine [129] [91]. Design languages and simulation programs are not computational models but they have an underlying one [54]. Typically a set of sequential functions define the next execution point that meet the model behavioral relations [68].

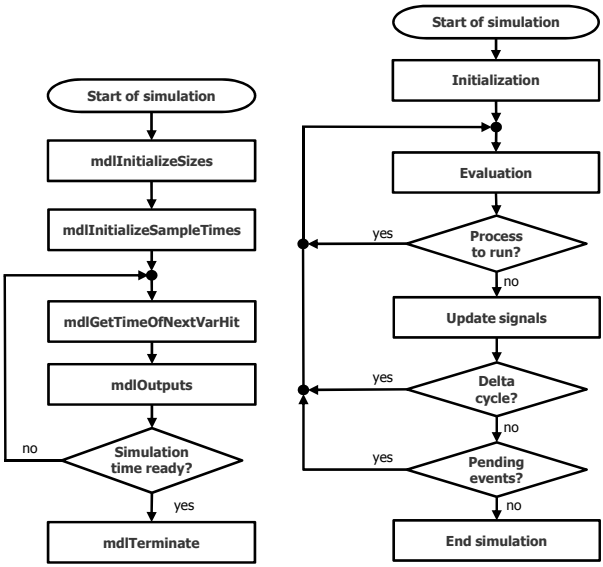


Fig. 2.5: Abstract model execution semantics: a) Simulink b) SystemC

The abstract semantics of the execution engine can be quite different depending on the modeling approach. Fig. 2.5 shows the abstract semantic of the Simulink and SystemC simulation engines. Executing only active processes, the SystemC discrete event execution algorithm achieves fast simulations of digital systems. For continuous time systems the state based simulation engine of Simulink requires less computation steps. As available system design tools extend one of both computation approaches for enabling AMS system simulation, they show a performance and accuracy trade-off for mixed analog and digital system simulation.

The concrete implementation of the set of functions defined in the abstract semantics of execution engine determines the semantics of the computation domains. Chapters 3 and 5 present domain specific implementations for the behavior computation of switched electrical networks and digital systems at register transfer level which enable fast AMS system simulations in SystemC and SystemC AMS respectively.

The architecture of the domain specific execution engine represents a higher level simulation platform and the algorithms provided for model computation (operational specification) such as e.g. the Newton-Raphson nonlinear equations solver utilized for the simulation of analog circuits, constitute the domain specific platform services.

The behavior computation of analog models usually requires that models represented at lower abstraction level (e.g. circuit level) are reduced using a complex computational algorithm to a more abstract functional representation (e.g. state space equations) [91]. Transformation rules for mapping the supported models of computation onto the abstract execution machine are implicitly or explicitly implemented. Chapter 5 derives operational computation methods for analog systems at several abstraction levels that are very efficient and do not require the transformation of circuit equations to a reduced form.

2.5.8 Model Execution Layer

The *Model Execution Layer* runs the *simulation model* on one or more previously defined or dynamically assigned targets which are part of the *computational platform*. The *computational platform* is characterized by the following properties:

- computation power (instructions or tasks per second)
- concurrency (central or single-core/distributed or multi-core)
- heterogeneity (homogeneous/heterogeneous processors)

The utilization of the computational platform determines the time required to carry out the simulation of the system model [129]. Powerful computational platforms are needed to cope with design analysis, optimization and verification for large scale systems. In such engineering problems, the size of the design space or alternative design configurations become a challenge [129]. The optimal utilization of the computational platform resources requires today the implementation of concurrent model execution algorithms as well as the support of hardware acceleration such as GPUs. The intrinsic properties of the behavior computation methods determine the capability of the simulation platform to take advantage from the available computation platform.

2.5.9 Model Analysis Layer

The aim of the *Model Analysis Layer* is to capture and to evaluate the system design properties. An important design activity is to validate that the specification model behaves according to the system design purposes. As model simulation enables the computation of the model behavior, this is the most commonly used method for system design analysis and validation [106]. It requires the definition of an input values sequence (stimuli) under which a certain system behavior (sequence of computed output values) is expected.

Performance and cost metrics that characterize the system and their components are needed for system design analysis and validation [98]. The implementation model that results after a refinement step is validated against the specification model [98]. Design tools that evaluate the results of a refinement step with respect to the performance indexes and constraints are necessary for speeding up model refinement validation. For fast exploring the trade-off amongst different solutions, design tools that enable the instrumentation of behavioral models for performance and cost estimation are also necessary [107] [106]. In particular, the essential characteristics of the final system design such as performance, power, reliability, robustness and cost should be captured by design tools [103].

This dissertation proposes a novel robustness evaluation method as well as operational methods for fast computation of performance indexes which can be applied to common AMS control systems.

2.5.10 Model Verification Layer

The aim of the *Model Verification Layer* is to ensure that the final system implementation fulfills the specified requirements and constraints. This is a key activity and should be always part of system design methodology. The primary goal is the verification that system implementation has no errors. In order to find design errors, the input sequence values utilized for system design simulation are often chosen to cover (all possible) corner cases. The specification and simulation of corner cases becomes particularly difficult for large and complex systems. This increases the probability that design errors in the final system implementation are not detected [106]. For this reason, it is very important that the simulation speed of design tools enables the verification of large systems [54]. System design languages such as SystemC are conceived for achieving a better simulation speed compared to hardware description languages such as VHDL and Verilog.

As the verification effort increases exponentially with the system functionality and complexity, it has usually the higher impact on the total development cost of contemporary embedded systems [54]. Common design verification methodologies take a system model and determine if the set of properties and constraints imposed on the system are fulfilled. The formalism and semantics of the design language utilized for representing the system under design have a considerable impact on the verification efficiency and effectiveness [54].

Design verification should be carried out at high abstraction level to early find design problems [54]. Simulation based system verification is the most common used method for system verification. Due to the fact that the system model is usually not based on a single semantics (different MoCs), the verification of system design and implementation becomes a challenge [54]. The orthogonal signal based modeling method presented in chapter 4 and its computational advantages helps to cope with these issues. It is explained in chapter 7. Moreover, semi-symbolic tolerance analysis is investigated.

2.6 Chapter Summary and Conclusions

As a clear definition of nonfunctional requirements helps to reduce the design space exploration effort and enables the definition of better design methodologies, the presented system design platform model separates functional requirements from design constraints. Design tools which describe the system specification in this way are more suitable for a stepwise system design refinement and optimization.

The presented design platform model separates the modeling of system structure from behavior to achieve more general and efficient design tools. Most design tools present a poor topology analysis support. Capturing the model architecture is an important modeling feature because structural models allow a better MoC abstraction and analysis than models described only at behavioral level as shown in chapter 3. It enables the faster simulation and efficient verification (corner cases can be automatically derived) of switched electrical networks. Modeling formalisms supporting AMS systems are a further weak point of many design tools. Appropriate MoCs that capture the system properties at several abstraction levels are necessary to ensure a correct system design implementation. In particular, the utilized MoCs should capture the relevant system properties before and after each design refinement step. The set of available MoCs has a considerable impact on the whole design process. Both the availability of MoCs that provide adequate abstractions for system design as well as the efficient implementation of the MoC primitives are very important design tool features that contribute to increasing the system design productivity.

A reason for the poor MoC support is the difficult composition and accurate synchronization of heterogeneous MoCs. The root of this problem lies in the approach for representing signals. The domain mapping or the reduction to a single semantics for completely different dynamical behaviors such as the behaviors of analog and digital systems requires a signal modeling approach that is able to efficiently capture the behavioral properties. A novel signal modeling method with a unified yet orthogonal representation of continuous and discrete dynamic behaviors is proposed in chapter 4 for overcoming the heterogeneous modeling issues. As shown in chapter 5, design tools that provide a better support for heterogeneous system modeling and simulation at several levels of abstraction can be achieved using this orthogonal signal model.

Design tools usually hide the exact structure of the program utilized for carrying out model processing and execution. Domain interfaces are implicitly inserted by the model translation process. The programmer or designer does not known the details of the algorithms utilized for implementing model decomposition, mapping and execution [54]. The selection or modification of these algorithms is normally not possible. With the aim of improving design tool customizing, the presented design platform model emphasizes the importance of a more abstract representation of the simulation platform. The model implementation layer does not only include the system implementation parameters and constraints in the specification model, it also defines an instance of the simulation platform. Moreover, the proposed design platform model decouples the simulation platform from the concrete computational platform and has the potential to support faster model simulations on emerging parallel and high-performance computational platforms[3].

The capabilities of design tools for system design analysis and verification have a significant impact on the system design productivity. For this reason, chapters 6 and 7 propose design methodologies and computation methods that enable the faster achievement of robust AMS system designs based on a reduced set of system properties. The computational advantages of the system behavior modeling and computation methods proposed in chapters 4 and 5 enable faster system design analysis and verification.

As the proposed layered system design platform model captures the entire design process and organizes the design activities in a hierarchical manner, it is very useful to compare and develop design methodologies and tools. The formal description of the several model layers would be an interesting future work towards a quantitative evaluation of design methodologies and tools. In the scope of this dissertation, the presented layered system design model serves as overview for evaluating the impact of the proposed behavior modeling and computation methodologies on the system design activities regarding quality and productivity.

[3] This research topic is not included in the scope of this dissertation

Chapter 3
Electrical Network Modeling and Simulation

Behavior abstraction techniques for analog components modeling at system level are an active research topic in heterogeneous SoC design. In particular, efficient power system modeling is relevant for low power consumption embedded system design. Modeling semiconductor components such as diodes and transistors as ideal instantaneous switches lead to efficient circuit response computations. The resulting switched electrical networks are well suited for the integration of power electronic circuits into system level models. They provide a good accuracy and simulation speed for design and validation purposes. The circuit behavior abstraction based on ideal switches allows fast and stable power control simulations, however requires the handling of voltage and current impulses for determining the correct topology after switching. This chapter presents an efficient impulse analysis method at topology level that allows the prediction and logic representation of voltage and current impulses as well as the fast and reliable topology switching computation. The main advantage of the proposed topological impulse analysis method is the efficient clustering of multiple topologies that produce electrical impulses. It enables the fast identification of such topologies for circuits containing a large number of switches before simulation start. Furthermore, the proposed topological analysis method does not depend on the circuit equations formulation form which is a key feature. In order to validate the proposed methodology, a SystemC AMS extension was implemented and utilized for the modeling and simulation of several power converters.

This chapter is organized as follows. After the presentation of the research motivation and related work in section 3.1 and 3.2 respectively, the research contribution of this chapter is summarized in section 3.3. The traditional circuit modeling and simulation methodology is reviewed in section 3.4. The modeling of switched electrical networks is introduced in section 3.5 and the proposed modeling and analysis method is presented in section 3.6. The SystemC MoC implementation for power electronic modeling is described in section 3.7 and its practical applicability is demonstrated in section 3.8. Finally, section 3.9 states the value of the presented circuit analysis and computation methods.

3.1 Motivation

Analog circuit modeling at higher levels of abstraction is gaining importance to facilitate high-performance system-level simulations. In order to provide support for System-on-Chip (SoC) design, system level modeling and simulation languages and tools such as SystemC were extended in the past years to enable the modeling and simulation of analog and mixed-signal (AMS) systems. The current SystemC AMS standard includes models of computation (MoCs) for continuous and discrete time data flow modeling as well as for electrical network modeling. These behavior abstraction methodologies provide enough support for system design and verification in a wide range of applications. However, for applications requiring large signal behavior and switching mode operation such as driver stages with pulse width modulation (PWM), the existing SystemC AMS modeling capabilities do not achieve the necessary accuracy [44]. Hardware description languages such as VHDL-AMS and Verilog-AMS provide more accurate analog modeling features but they present a poor simulation performance [117].

Modeling methodologies that rely on ideal instantaneous switches represent power electronic systems as internally and externally controlled switched networks providing an appropriate behavior abstraction of power electronic systems. They enable fast, stable and accurate simulations of power electronic circuits coping with the growing need for better models of computation but require the implementation of efficient and reliable computation methods that are able to predict and handle voltage and current impulses as well as to correctly compute and handle circuit topologies after switching. This is a mandatory feature for simulating circuits which contain a large number of internally controlled switches such as high voltage power converters.

3.2 Related Work

In order to provide modeling capabilities for systems with mixed digital and analog behavior, Al-Junaid at al. presented a SystemC extension that uses a general-purpose analog solver for the computation of analog components [1]. A lock-step method was applied for invoking and synchronizing the analog solver with the SystemC kernel.

The language constructs of SystemC-A support a variety of analog abstraction levels, including non-linear components (diode and MOSFET) for electrical circuit modeling. The corresponding circuit equation set was formulated using the modified nodal analysis (MNA) method and the resulting differential algebraic equations (DAEs) were solved using the Newton-Raphson (NR) method. To deal with simulation performance issues, the handling of extremely small and zero time-steps was incorporated into the analog solver [2].

With the aim of filling the gap in heterogeneous SoC modeling and simulation, the SystemC extension for analog and mixed-signal systems includes MoCs that enable the modeling of linear dynamic continuous time systems and linear electrical networks [119] [118]. The continuous time MoCs are embedded into discrete event modules as a cluster of data flow components with a static scheduler. The synchronization between continuous time and discrete event model parts is carried out utilizing a fixed time step. Uhle et al. proposed a model of computation for SystemC AMS that enables the modeling of analog nonlinear parts of cyber-physical systems in similar way to VHDL-AMS and VerilogAMS [117]. They utilized the generic mechanism of SystemC AMS for interfacing the nonlinear continuous time solvers and discrete event system parts during simulation.

Although electrical network modeling is supported by the above mentioned SystemC extensions, the implemented circuit abstractions do not provide the necessary accuracy or simulation performance for the analysis and validation of power electronic circuits at system level. SystemC AMS allows rapid electrical network simulations but it does not offer primitives for modeling internally controlled switches (such as diodes and similar semiconductor components). The proposed extensions based on nonlinear continuous time behavior computation lead to sophisticated models (based on detailed component models with large number of parameters) and slow simulations, restricting their practical application to small power electronic circuits.

Grimm et al. utilized pre-solved parametrized differential equations to enable faster simulations of analog power drivers [44]. The dominant cycle of the network was determined by applying the Dijkstra algorithm. This approach allows very fast simulations but similar to SystemC AMS, it does not fully support the simulation of internally controlled switches. Only a single topological change is permitted at each switching instant.

In order to carry out fast and accurate time domain analysis of internally controlled switched networks, Bedrosian at al. proposed a model of computation that relies on ideal switches and allows several topological changes after switching [12] [13]. Considered impulsive voltages and currents at the switching instants in conjunction with the initial conditions, they determined the correct topology after switching. For the accurate handling of voltage and current impulses at the switching instants, they separated the circuit response into an impulsive and a non-impulsive component. For each circuit topology, the impulsive response part was calculated using a computational method based on the inverse Laplace transform from the network equations (generated utilizing a two-graph modified nodal analysis technique).

Chung et al. presented a simulation method for switching circuits with internally controlled switches that analyzes the circuit as a linear resistive one (energy-storing elements are replaced by independent DC sources) to avoid the inverse Laplace transform calculations [17].

Allmeling et al. developed a toolbox named PLECS which reduces the automatically generated mesh and node equations describing the circuit to a state space form and determines the circuit topology by analyzing the system outputs and the control inputs. In addition to the non-impulsive part of the system output, an output impulse-multiplier which is implicitly associated with a voltage or current impulse, is determined after switching by a switch manager [4].

Massarini et al. proposed a switched network time domain analysis method based on a state space circuit equations formulation that allows the prediction of possible impulsive transitions due to internally controlled switches [80] [79] [81]. For each topology, the state equations and output equations are systematically obtained from the circuit equations and analyzed using a logical representation of impulsive quantities, in order to find the correct topology after switching.

Although the switched network behavior computation methods based on the state space representation of circuit equations allow the systematic prediction and logic description of voltage and current impulses, they require a considerable effort to reduce the circuit equations to the state space form at each switching instant. On the other hand, the prediction and analysis of voltage impulses demands a considerable effort for circuits described using the MNA method. Therefore, a more general switched network analysis method is needed for the efficient behavior computation of power electronic circuits.

3.3 Contribution to Power Electronic Modeling

The aim of the research work presented in this chapter was to derive circuit equations' analysis and reduction methods for an efficient implementation of a switched network simulator that enables the system level time domain response analysis of power electronic circuits. The main research activities carried out to achieve this goal were the following:

- To investigate the suitability of several circuit equations formulation methods to represent switched electrical networks (regarding implementation complexity and computation accuracy).
- To evaluate the impact of several circuit equation formulation methods on the behavior computation efficiency.
- To derive equation reduction methods for switched electrical networks.
- To find out a general method for predicting possible electrical impulses that does not depend on the representation of circuit equations.
- To define an efficient algorithm which finds the correct circuit topology after switching.
- To analyze the accuracy of several algebraic computation methods.
- To develop computational approaches for improving the circuit simulation performance and accuracy.

In order to carry out an efficient analysis and formulation of circuit equations this work captures switched electrical networks using a table representation. This approach enables the prediction of switching impulses and the compact formulation of circuit equations which improves the simulation performance and avoids solvability problems.

As the prediction of switching impulses from network topology properties is not limited to a particular circuit equations formulation method and enables the efficient simulation of switched electrical networks (the impulse analysis after each topology change is not necessary), the proposed method is a valuable contribution to the simulation of large power electronic circuits.

3.4 Circuit Modeling and Simulation

In this section, the traditional algorithm utilized for time domain circuit analysis[4] in popular simulators such as PSpice and Saber is briefly explained. As shown in Fig. 3.1, the simulation algorithm consists of the following main tasks [90]:

1. Formulation of circuit equations
2. Discretization of circuit equations (numerical integration)
3. Linearization of circuit equations
4. Solving linear algebraic circuit equations

Note that the simulation algorithm tasks are carried out sequentially and iteratively.

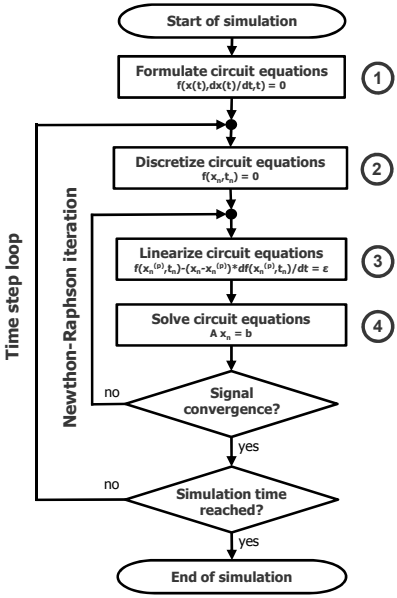


Fig. 3.1: General time domain circuit simulation approach

[4] In this work circuit denotes electrical networks

3.4.1 Formulation of Circuit Equations

There are two approaches usually utilized in practice for the formulation of circuit equations:

- The Modified Nodal Analysis (MNA).
- The reduction to State Space (SS) equations.

The Modified Nodal Analysis (MNA) method is utilized in circuit simulators such as PSpice or Saber for the systematic formulation of circuit equations. It is a simple method that enables the description of a large variety of circuits and leads to sparse system matrices of moderate size. It is considerably more efficient than other general circuit formulation methods such as tableau (for more details see [120]).

The state space representation of the circuit equations requires the reduction of circuit equations to a minimal set (state space form) after their formulation. This equation transformation method is quite sophisticated but leads to dense system matrices. It is utilized by some circuit simulators for achieving a faster time domain circuit analysis [116].

Modified Nodal Analysis (MNA)

In this general formulation method, each circuit element adds its contribution to a system matrix[5] using an element stamp. Frequency independent circuit elements (resistors, etc.) are added to the *admittance matrix* $[G_{MNA}]$ and frequency dependent circuit elements (capacitors, inductors, etc.) are added to the *impedance matrix* $[C_{MNA}]$. Inductors and other circuit elements are added in impedance form i.e. introducing a branch current in the equations. The contribution of voltage and current sources as well as the initial condition of energy storing elements are added to the right-hand-side vector $[w(t)]$. Both matrices are square and have the same size. The first N rows and columns of the matrices correspond to the circuit node voltages and the remaining B rows and columns of the matrices correspond to additional branch currents. This equation construction method is straightforward and eliminates some redundant variables.

[5] In this work, matrices and vectors are denoted in square brackets

After the systematic construction of circuit equations, the following set of differential algebraic equations (DAEs) is obtained:

$$[G_{MNA}]_{(N+B)^2} \cdot [x(t)]_{N+B} + [C_{MNA}]_{(N+B)^2} \cdot \frac{d}{dt}[x(t)]_{N+B} = [w(t)]_{N+B} \quad (3.1)$$

where the real vector $[x(t)]_{N+B}$ consists of N *node voltages* and B *branch currents*. The voltage of all circuit nodes are included in this vector. Square brackets with subscripts are used in this work to explicitly denote the size of matrices and vectors.

Eq. (3.1) is valid for linear electrical networks. Nonlinear elements are modeled as controlled electrical sources. They appear thus in the right-hand-side vector $[w(t)]_{N+B}$.

State Space (SS) Equations

The state space equation formulation is commonly used for the simulation of switched or piece-wise linear circuits. Once the initial circuit equations have been formulated, a procedure to reduce the implicit equations to a set of ordinary differential equations (ODEs) must be applied (e.g. using Gaussian elimination for reducing the equation system to a triangular form) [116]. The circuit equations are thus represented by a minimal set of variables as follows:

$$\frac{d}{dt}[x(t)]_N = [A_{SS}]_{N^2} \cdot [x(t)]_N + [B_{SS}]_{NM} \cdot [u(t)]_M \quad (3.2a)$$

$$[y(t)]_R = [C_{SS}]_{RN} \cdot [x(t)]_N + [D_{SS}]_{RM} \cdot [u(t)]_M \quad (3.2b)$$

where $[x(t)]_N$, $[u(t)]_M$ and $[y(t)]_R$ are the state, input and output vector variables respectively. The *state vector* $[x(t)]_N$ contains continuous signals such as capacitor voltages and inductor currents. The *input vector* $[u(t)]_M$ contains signals that may be discontinuous such as voltages and currents from source circuit elements.

Eq. (3.2) is valid for linear systems. It characterizes any linear system with continuous dynamic behavior. In this chapter, linear electrical networks are considered. There are N independent explicit linear differential equations (know as the *state equations*) to be solved. The remaining R linear algebraic equations, which describe the system outputs, can be computed directly by using algebraic operations.

3.4.2 Numerical Integration

Numerical integration methods transform the differential equations describing a continuous time system into a set of algebraic equations. Discretization methods are numerical integration methods that approximate the time derivative operator with a divided difference operator transforming the continuous time differential equations into *discrete time algebraic equations*. Depending on the time points utilized for computation, discretization methods are classified in two groups [90]:

1. One-step methods.
2. Linear multi-step (LMS) methods.

As shown in Fig. 3.2 one-step methods lead to additional computations at intermediate time points (unfilled circles). Non-stiff integration routines such as Runge-Kutta (RK) are widely utilized for the simulation of continuous time systems e.g. by Simulink. As typical power electronic systems present large spread in time constants, they produce a stiff set of equations which are not handled well by one-step integration methods [116]. Implicit multi-step integration methods such as Backward Differentiation Formulas (BDF) are required for accurate and stable simulations.

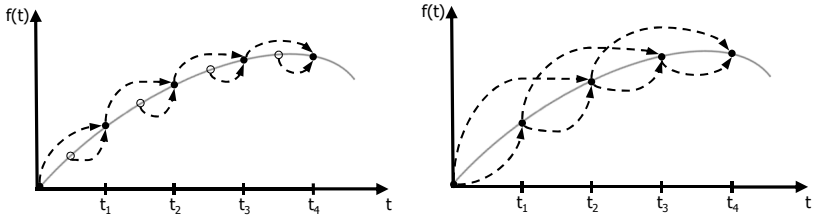


Fig. 3.2: Discretization methods: a) One-step b) Linear multi-step

One-step Discretization Methods

One-step discretization methods make use of the previous computed value x_n and new values at intermediate time points between t_n and t_{n+1} to produce the value x_{n+1} at the time t_{n+1}.

The values at intermediate time points allow to improve the accuracy of the computed new value x_{n+1}. They are discarded after the integration step i.e. they are not reused in the computation of future steps. Commonly used one-step integration methods are Runge-Kutta and Rosenbrock [97]. They are suitable for variable time step simulations.

Linear Multi-step Discretization Methods

Multi-step discretization methods make use of only previously computed values (x_n, x_{n-1}, etc.) in order to produce the new value x_{n+1} at the time t_{n+1}. They are computationally more efficient than one-step methods. They are also more suitable for solving stiff differential equations, such as the equations describing linear electrical circuits. These methods are typically used in traditional circuit simulators such as PSpice. The solution of the ordinary differential equation $dx(t)/dt = f(x(t), t)$ by the linear multi-step methods can be expressed in the following general form:

$$\sum_{j=-1}^{k-1} \alpha_j \cdot x_{n-j} = h \cdot \sum_{j=-1}^{k-1} \beta_j \cdot f(x_{n-j}, t_{n-j}) \tag{3.3}$$

where k defines the number of steps and h is the time step. Depending on the values of α_j and β_j different set of LMS equations are obtained.

In the Adams-Bashforth LMS methods, β_{-1} is equal to zero. The resulting equations are explicit. The next step value can be directly computed using the previous step value and the previous computed derivatives.

$$\begin{aligned} x_{n+1} &= x_n + h \cdot f_n && (k = 1) \\ x_{n+1} &= x_n + \tfrac{h}{2} \cdot (3 \cdot f_n - f_{n-1}) && (k = 2) \\ x_{n+1} &= x_n + \tfrac{h}{12} \cdot (23 \cdot f_n - 16 \cdot f_{n-1} + 5 \cdot f_{n-2}) && (k = 3) \end{aligned} \tag{3.4}$$

The first Adams-Bashforth equation (k=1) is known as *Forward Euler* (FE). It is sometimes used for predicting the new value x_{n+1}.

In the Adams-Moulton LMS methods, β_{-1} is different from zero. The resulting equations are therefore implicit. The next step value depends on the previous step value x_n as well as on the derivative at the next step f_{n+1} and the previous computed derivatives (f_n, f_{n-1}, etc.).

$$\begin{aligned} x_{n+1} &= x_n + \tfrac{h}{2} \cdot (f_{n+1} + f_n) && (k=1) \\ x_{n+1} &= x_n + \tfrac{h}{12} \cdot (5 \cdot f_{n+1} + 8 \cdot f_n + f_{n-1}) && (k=2) \\ x_{n+1} &= x_n + \tfrac{h}{24} \cdot (9 \cdot f_{n+1} + 19 \cdot f_n - 5 \cdot f_{n-1} + f_{n-2}) && (k=3) \end{aligned} \tag{3.5}$$

The first equation (k=1) is known as the *Trapezoidal Rule* (TR). This is one of the most commonly used discretization method in circuit simulation.

In the Backward Differentiation Formulas (BDF), the resulting equations are also implicit. The next step value depend on the previous step values (x_n, x_{n-1}, etc.) as well as on the derivative at the next step f_{n+1}.

$$\begin{aligned} x_{n+1} - x_n &= h \cdot (f_{n+1}) && (k=1) \\ x_{n+1} - \tfrac{4}{3} \cdot x_n + \tfrac{1}{3} \cdot x_{n-1} &= \tfrac{3 \cdot h}{2} \cdot (f_{n+1}) && (k=2) \\ x_{n+1} + \tfrac{18}{11} \cdot x_n - \tfrac{9}{11} \cdot x_{n-1} + \tfrac{2}{11} \cdot x_{n-2} &= \tfrac{6 \cdot h}{11} \cdot (f_{n+1}) && (k=3) \end{aligned} \tag{3.6}$$

The first BDF equation (k=1) is known as *Backward Euler* (BE). The most commonly used BDF equation in circuit simulation is the second order (two-step) BDF, which is often denoted as BDF2. This is known as the *Gear-Shichman formula* or second order Gear formula in the circuit simulation literature. Higher order formulas are often more accurate but any BDF equation with order higher than 6 is zero-unstable. The great advantage of TR, BE, and BDF2 LMS methods is that their stability does not impose restrictions on the time step h. In other words, these methods produce a stable difference equation system for any given stable differential equation system.

3.4.3 Numerical Linearization

If the differential equations describing a continuous time system are nonlinear, the discretized algebraic equations are also nonlinear. Numerical linearization methods transform a set of nonlinear algebraic equations into a set of linear algebraic equations. Practical approaches for solving a set of nonlinear algebraic equations typically consist in an iterative method that generates a sequence of candidate solutions. For circuit simulation (stiff problems), solving nonlinear algebraic equations by any of the standard methods, such as the *fixed point method*, does not work well. For stability reasons, implicit methods are required. The Newton-Raphson (NR) method is instead more appropriate to solve the set of nonlinear algebraic equations describing electronic circuits. This method constructs a local linearized equation using a first order truncated Taylor series expansion. For the one dimensional case, the linear equation around the value x_k is given by:

$$M_k(x) = f(x_k) + J(x_k) \cdot (x - x_k) \tag{3.7a}$$

$$J(x_k) = \left.\frac{\partial f(x)}{\partial x}\right|_{x=x_k} \tag{3.7b}$$

The solution of the implicit equation $f(x) = 0$ is approximated by the solution $M_k(x) = 0$. Thus, the approximate solution x_k is iteratively improved by solving the following linear equation:

$$x_{k+1} = x_k - \frac{f(x_k)}{J(x_k)} \tag{3.8}$$

For the computation of the circuit behavior (multi-dimensional case), Eq. (3.8) is formulated as a linear algebraic equation system.

3.4.4 Numerical Solution of Linear Algebraic Equations

After the discretization and linearization steps, a linear algebraic equation system describing the electrical circuit behavior is obtained. It has the following generic form:

$$[A]_{N^2} \cdot [x_{k+1}]_N = [b_{k+1}]_N \tag{3.9}$$

where N is the number of equations. The final form of the system matrix $[A]_{N^2}$ and of the vector $[b_{k+1}]_N$ depend on the applied numerical methods for circuit equations formulation, integration and linearization.

There are two classes of efficient techniques for solving a linear algebraic equation system:

1. Direct methods.
2. Indirect (or iterative) methods.

For small linear electrical circuits, Eq. (3.9) can be efficiently solved using direct computation methods. For the simulation of large nonlinear circuits, iterative computation methods may provide a better accuracy and performance than direct methods.

Direct methods solve a linear algebraic equation system in a fixed and pre-determined number of steps. The most commonly used method for finding the solution of a system of linear algebraic equations is the *Gaussian elimination* (GE). It is one of the most efficient and robust methods for solving algebraic equation systems. This method is also easy to implement and can be extended for exploiting matrix sparsity. This is useful for circuit simulation that relies on MNA formulation methods. There are many variants of the Gaussian elimination method, such as LU factorization and Cholesky decomposition. Most modern circuit simulators carry out a LU factorization and compute the circuit response at each step applying *backward substitution* and *forward elimination* (for more details see [120] or [90]).

Indirect methods are iterative computation techniques that approach gradually the solution of a linear algebraic equation system. These methods can improve the accuracy of the solution but they converge only if the system matrix $[A]_{N^2}$ has certain properties. The most commonly used methods for the iterative computation of electrical circuit behavior are Gauss-Seidel, Gauss-Jacobi and Conjugate Gradient [90]. As iterative computation methods involve only matrix-vector multiplications, they can be more efficient than direct methods, especially for the simulation of large nonlinear circuits on parallel computers. However, they only work well on a certain classes of circuits e.g. CMOS.

3.5 Power Electronic Modeling with Ideal Switches

As the process during switching is not important at the first design stages, ideal instantaneous switches are an appropriate behavior abstraction for modeling power semiconductor components. An ideal or perfect switch has zero resistance in ON state, zero admittance in OFF state and switches between both states in zero time.

The computational advantages of ideal switching modeling are:

1. Simple numerical integration algorithm (no linearization).
2. Short time switching process (only two computation steps).
3. Reduced equation system.
4. Less solvability problems.
5. More accurate models (no additional components for stability).
6. Faster simulations.

The methods required to solve the differential equations describing the circuit behavior at each circuit topology depend on the circuit element characteristics. Using linear electrical elements, the linearization of the circuit equations is not necessary. Simple integration methods such as Backward Euler (BE) or the Trapezoidal Rule (TR) stably solve the linear circuit equations. Furthermore, the time step can be adjusted once after a switching event (new circuit topology) leading to an efficient and accurate simulation of power electronic circuits. As ideal switches change their state in zero time, only two integration steps need to be computed to handle the discontinuities at each switching event. The numerical integration algorithm is applied again to compute the system state after switching.

The nodes connecting ideal switches can be contracted to a single node when ideal switches are ON. Thus, the size of the network matrices and hence the computation time can be reduced.

As ideal switches do not have very small or very large parameter values (such as resistive switch models), the resulting circuit equation system is normally non-stiff which leads to stable simulations. Moreover, ideal switches do not affect the circuit response. Additional passive components (called snubbers) for simulation convergence are rarely necessary. Such *snubber components* introduced for simulation stability may considerably increase the simulation time and reduce the computation accuracy.

Computational Requirements for Ideal Switches

Ideal switches lead to more stable and faster electronic circuit simulations than other semiconductor modeling approaches but they require the proper handling of *network inconsistencies* after topology changes during simulation. An ideal switch produces a short circuit when it is closed (ON) and an open circuit when it is open (OFF). Depending on the resulting topology after commutation, different *circuit equation inconsistencies* can take place due to:

1. Floating branches.
2. Short-circuited voltage sources.
3. Open-circuited current sources.

If one or more circuit equation inconsistencies are present in a circuit topology, the corresponding circuit equations cannot be numerically solved. Circuit analysis is necessary to find such network inconsistencies. After removing *topological inconsistencies* from circuit equations, the network response can be computed but it may be discontinuous at the switching instants. Voltage or current impulses may occur after topology switching. They can be caused by:

1. Short-circuited capacitors.
2. Open-circuited inductors.
3. Different initial conditions of interconnected circuit elements.

Electrical impulses are produced by *inconsistent initial conditions* or changes in the *topological state* (number of energy storing elements). Impulsive currents or voltages need to be accurately recognized and appropriately handled for correct circuit simulation.

Taking into consideration the network components and topology, switched networks can be classified into:

1. Externally controlled switched networks.
2. Internally controlled switched networks.

In externally controlled switched networks the state of the switches do not depend on the network response. All switches are controlled by external signals. It leads to *forced commutations*.

In internally controlled switched networks one or more switches are controlled by network voltages or currents. The switching time

between different topologies depends on the state of network components. It leads to *natural commutations*. Internally controlled switches may change their state if a current or voltage impulse is applied to them. Taking into account that many topology changes can occur at a given switching instant in internally controlled switched networks, the computation of the correct topology after switching is necessary.

As described in the related work, there are several methods to cope with voltage and current impulses in switched networks as well as for determining the correct topology after switching. In the next section, a general method based on topological analysis is proposed.

3.6 Topology Analysis of Switched Electrical Networks

To carry out an efficient topology analysis of switched electrical networks this work extends graph theory representation methods for the detection of network inconsistencies. Fig. 3.3 shows the graph representation of a Buck converter circuit.

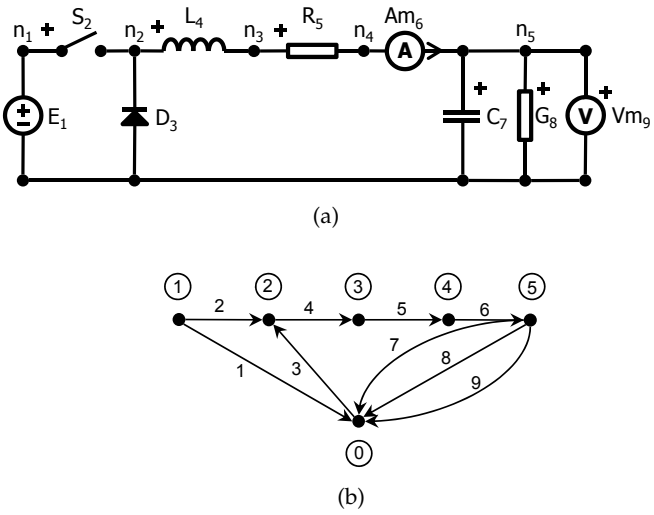


Fig. 3.3: Buck converter: a) Circuit b) Graph representation

A network graph considers the network elements as *terminal components*. It describes the connection between network elements capturing the topological properties of the network in a natural way.

For the construction of the directed graph network representation, each 2 terminal element of the network is numbered and replaced in the circuit by a line called *edge*. An *orientation* corresponding with the assigned current flow direction is associated with each edge in the graph. For passive elements, the node from which the current flows is the positive terminal. For current sources, the direction of the current is defined by its symbol. For voltage sources, the direction of the current flow is from the positive to the negative terminal. The nodes of the network correspond to the *vertexes* of the graph. They are numbered. The vertex numbers in the graph are placed within circles to distinguish them from the edge numbers. Zero is assigned to the ground.

Network graphs can be efficiently described in form of a simple table. The construction of the table representing an electrical network on the computer is straightforward. Each 2 terminal element of the network adds a new column to the table. The input and the output terminals of two ports elements (e.g. a transformer) are represented as separated edges (columns in the table form). For each network element, the edge number, the type as well as the number of the positive and the negative nodes are described in the corresponding table row.

In order to carry out a switched network analysis, this work extends the graph representation by including an additional row that contains the number Z_i assigned to the switching element. It is incremented for each switching element in the network. The following table represents the buck converter circuit.

Edge	1	2	3	4	5	6	7	8	9
Type	E	S	D	L	R	A_m	C	G	V_m
N_p	1	1	0	2	3	4	5	5	5
N_n	0	2	2	3	4	5	0	0	0
Z_i	0	1	2	0	0	0	0	0	0

Table 3.1: Buck converter table

Reduced Network Graph for Inconsistency Analysis

The extended representation of a switched electrical network including the switching element numbers allows the prediction of inconsistent circuit equations as well as the reduction of the circuit equations. The following information about the network nodes and branches is obtained from the *extended network graph*:

- Connected elements which can be reduced to a single element.
- Floating nodes (and their switching dependencies).
- Floating branches (and their switching dependencies).
- Complementary network branches (i.e. diode bridge)

A *floating node* is a node to which only one element is connected. The current through a branch containing a floating node is zero. A *floating branch* is a branch terminated with floating nodes. It is not possible to compute the voltage across its nodes.

Using the extended network graph, the following switched network topology analysis steps are carried out in this work:

1. Group serial connected elements which are represented in impedance form (single current branch containing R-L-A_m elements).
2. Group parallel connected elements which are represented in admittance form (single nodal admittance containing G-C-V_m elements).
3. Identify floating nodes if all switches are open.
4. Identify floating branches if all switches are open.

Applying the steps 1-2, a reduced circuit table is obtained. Table 3.2 shows the reduced circuit table for the Buck converter circuit.

Edge	1	2	3	4-5	7-8-9
Type	E	S	D	L-R-A_m	C-G-V_m
N_p	**1**	**1**	0	**2**	5
N_n	0	**2**	**2**	5	0
Z_i	0	1	2	0	0

Table 3.2: Reduced circuit table for Buck converter

The circuit elements L, R and A_m have the same current and can be grouped into a single edge. The parallel admittance elements G, C

and V_m have the same voltage applied to their nodes and can be also grouped into a single edge.

If all switches are open (step 3), there are two floating nodes in the circuit (node 1 and node 2). As all circuit branches are connected to ground, any floating branch exists that would need to be removed from the network equations when all switches are open (step 4).

Voltage and Current Graphs for Impulse Analysis

An edge on a network graph simultaneously represents the current through the network element and the voltage across the network element. For some elements of the network, one of the constitutive variables may be zero. For example, the current through an ideal switch or the voltage across its terminal are zero when it is open or closed respectively (complementary behavior). For some other elements, one of the constitutive variables is not necessary for the solution of the resulting equation system such as the current through a voltage source and the voltage across the terminal of a current source. Using separated graphs for representing the network voltages and currents, this redundancy can be eliminated (for more details see [120]).

This work utilizes the current and voltage graphs for both reducing network equations as well as for detecting topologies may generate impulses in switched electrical networks. From the *current graph*, the following network information is obtained:

- Open circuited current sources and their switching dependencies.
- Branches which may generate voltage impulses and their switching dependencies.
- Voltage impulses acting on internally controlled switches.

From the *voltage graph*, the following network information is obtained:

- Short circuited nodes and their switching dependencies.
- Branches which may generate current impulses and their switching dependencies.
- Current impulses acting on internally controlled switches.

Using the *extended current graph*, the following switched network topology analysis steps are carried out in this work:

1. Collapse the nodes of the network elements that are not of interest for network equation formulation (voltage sources E and sinks V_m).
2. Collapse the nodes of the network elements that are not of interest for impulse analysis (only switches and branches containing inductors are of interest and remain in the table)
3. Identify open circuited current sources.
4. Identify branches connected to floating nodes and containing inductors (branches that can generate voltage impulses).
5. Identify the switching state dependencies for branches generating voltage impulses.
6. Identify voltage impulses acting on internally controlled switches and their polarity.

Table 3.3 shows the reduced circuit table obtained for the Buck converter circuit after applying the steps 1-2. The current across E and V_m is not of interest for network equation formulation and their nodes were collapsed. The parallel admittance branch G-C is not of interest for impulse analysis and their nodes were also collapsed.

Edge	2	3	4-5-6
Type	S	D	L-R-A_m
N_p	0	0	**2**
N_n	**2**	**2**	0
Z_i	1	2	0

Table 3.3: Current analysis table for Buck converter

Using this reduced table, it is possible to find the states producing a voltage impulse. If both switches are open (Z_1 NOR Z_2), the branch 4-5-6 containing an inductor can generate a voltage impulse (node 2 is open). It is even possible to determine that the produced voltage impulse when the switches are open (and the inductor current is different from zero) is applied to the diode with positive polarity (turning the diode ON).

Using the *extended voltage graph*, the following switched network topology analysis steps are carried out:

1. Collapse the nodes of the network elements that are not of interest for network equation formulation (current sources J).
2. Identify short circuited voltage sources and capacitors if all switches are closed.

3. Identify the switching state dependencies for branches generating current impulses.
4. Identify current impulses acting on internally controlled switches and their polarity.

Table 3.4 shows the reduced circuit table obtained for the Buck converter circuit after applying step 1. The voltage through A_m is zero, and their nodes can be collapsed.

Edge	1	2	3	4-5	7-8-9
Type	E	S	D	L-R	C-G-V_m
N_p	**1**	**1**	0	2	5
N_n	0	**2**	**2**	5	0
Z_i	0	1	2	0	0

Table 3.4: Voltage analysis table for Buck converter

After collapsing all switches, current impulses generated by short circuited voltage sources and capacitors can be easily found (step 2). In the Buck converter example, the battery E is short circuited when the switch S and the diode D are simultaneously closed (Z_1 AND Z_2). The current flow direction is determined by the polarity of the battery.

3.7 Implementation

In order to carry out switched network simulations in SystemC, this work exploits the generic architecture of SystemC AMS for the implementation of a model of computation supporting internally and externally controlled ideal switches. The new MoC, named *Electrical Piecewise-linear Networks* (EPN), follows the same syntax as the currently available MoC ELN (Electrical Linear Networks). Thus, only minimal code modifications are necessary for reusing existing SystemC AMS models.

The EPN MoC implements new primitives for semiconductor modeling and exploits the properties of ideal switched electrical networks for minimizing solvability problems and improving the simulation performance.

Listing 3.1 shows the implementation of the buck converter circuit.

Listing 3.1: Buck converter implementation using EPN MoC

```
SC_MODULE(Circuit)
{
sc_core::sc_in<bool> g;
sc_core::sc_in<double> u;
sc_core::sc_out<double> i_l;
sc_core::sc_out<double> v_c;

sca_epn::sca_de::sca_vsource E;
sca_epn::sca_r R;   sca_epn::sca_l L;
sca_epn::sca_c C;   sca_epn::sca_r G;
sca_epn::sca_switch S;
sca_epn::sca_diode D;
sca_epn::sca_isink Am1;
sca_epn::sca_de::sca_vsink Vm1;

Circuit(sc_core::sc_module_name name, sc_time time_step,
        g("g"), u("u"), i_l("i_l"), v_c("v_c"),
        E("E",1.0), L("L",0.1), R("R", 0.05), C("C",0.1), G("G",0.1),
        S("S"), D("D"), Am1("Am1"), Vm1("Vm1"),
        gnd("gnd"), n1("n1"),  n2("n2"), n3("n3"), n4("n4"), n5("n5")
{
    E.p(n1); E.n(gnd); E.inp(u); E.set_timestep(time_step);
    S.p(n1); S.n(n2); S.ctrl(g); S.set_timestep(time_step);
    D.p(gnd); D.n(n2); D.set_timestep(time_step);
    L.p(n2); L.n(n3); R.p(n3); R.n(n4);
    Am1.p(n5); Am1.n(n4); Am1.outp(i_l); Am1.set_timestep(time_step);
    C.p(n5); C.n(gnd); G.p(n5); G.n(gnd);
    Vm1.p(n5); Vm1.n(gnd); Vm1.outp(v_c); Vm1.set_timestep(time_step);
}
private:
sca_epn::sca_node_ref gnd;
sca_epn::sca_node n1, n2, n3, n4, n5;
};
```

In order to support the specific features of ideal switching modeling with natural commutation, a new analog solver class was implemented. All electrical element primitives (resistors, capacitors, etc.) utilize attributes and methods of this solver class. The namespace **sca_epn** was assigned to the new types and classes. All primitives are derived from the base class **sca_module**. In order to utilize the built-in binding mechanism of SystemC, terminals are instances of a type derived from **sca_port** and nodes are instances of a type derived from **sca_interface**. Circuit elements are also interconnected by binding terminals to nodes. Primitives for modeling ideal switches and diodes (**epn_switch** and **epn_diode** respectively) are provided to enable the novel language capabilities.

During the SystemC AMS initialization phase, the topology analysis method presented in section 3.6 is carried out. After that the circuit equations are formulated from the reduced circuit table representation applying the MNA method. Linear multi-step (LMS) methods are then utilized for the discretization of the linear circuit equations. The BE, TR and BDF2 integration methods were implemented for performance evaluation. The last step of the solver initialization is the system matrix construction. The value of the system matrix coefficients depends on the state of the internally and externally controlled switches. In order to avoid memory reallocation, if the current circuit topology contains inconsistencies, rows and columns associated to non-valid branches and floating nodes are shifted to the border of the system matrix and the solver variables indicating the number of system nodes and branches are respectively reduced. In order to improve simulation performance, the system matrix is factorized using LU decomposition. The system matrix creation and factorization is carried out during circuit simulation only once for each topology, when the corresponding switching conditions are reached. That is, the factorized system matrices are stored in memory and reload if the corresponding topology is being computed.

Figure 3.4 shows the implemented analog solver algorithm. It calls a list of pre-solve methods at the start of each integration step. They are dynamically registered by the circuit elements and managed by the solver. This SystemC AMS functionality is utilized by the ideal switches to read SystemC ports and update the circuit topology if necessary. If state changes are reported, the corresponding system matrix is loaded. The pre-solve methods are called again if inconsistencies are present in the new topology. Forward elimination and backward substitution are applied at each integration step for computing the system response.

After one integration step is done, the solver calls the post-solve methods. The diode class exploits this mechanism for the evaluation of switching conditions. It sets a solver flag if one of its threshold values (voltage or current) was reached. The solver updates the system matrix and repeats the integration step, if a natural switch condition was reported. Because ideal diodes modify their state instantaneously, this loop needs to be executed until no more switching conditions are detected. If a topology occurs twice during the switching process, the electrical network is assumed to be not stable and the simulation is aborted.

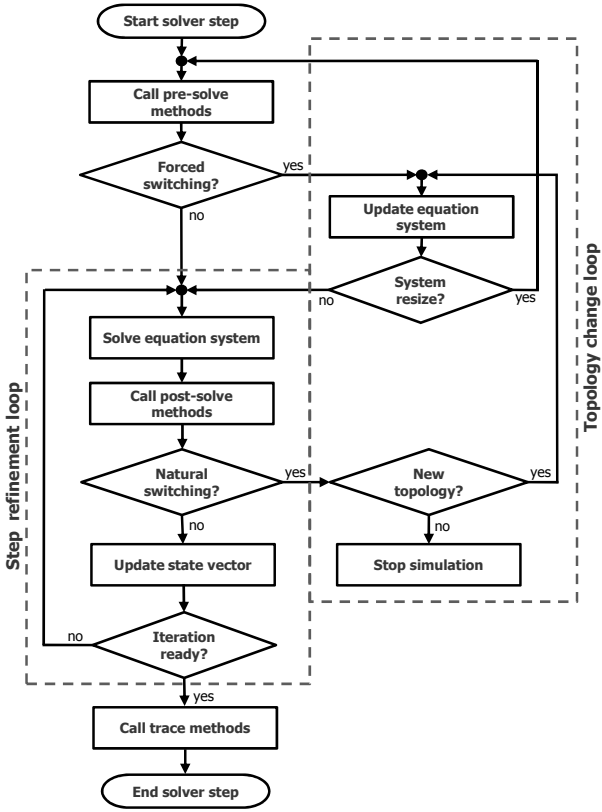


Fig. 3.4: Solver step refinement and topology change loops

The solver time control is implemented in SystemC AMS using a spawn process, which is triggered by events at fixed time steps. Due to the large signal behavior of power electronic circuits, a very short integration step is often required to minimize numerical integration errors. This can notably decrease the simulation performance. As shown in figure 3.4, a step refinement loop was incorporated into the solver algorithm to improve the simulation accuracy results while maintaining a good simulation performance. The solver variable controlling the number of loop iterations may be modified by a circuit element during simulation (calling the corresponding solver interface function in the

pre- or post-solve method). Thus, the time step can be adjusted depending on the current network topology. In order to split the trace methods from the step refinement and topology change loops, they are called by current and voltage sinks after the integration is finished (instead of in the post-solve method, such as in the SystemC ELN MoC).

A set of libraries are additionally provided for enabling the integration of the EPN solver into Simulink models. Thus, electrical circuits described using the SystemC AMS syntax can be utilized for simulation during power control design and validation.

3.8 Experimental Results

The Buck converter circuit shown in Fig. 3.3a was modeled using the presented MoC for performance and accuracy evaluation. A bang-bang controller (on-off controller) that closes the circuit switch when the inductor current is smaller than 9.5 A and opens the circuit switch when the inductor current is greater than 9.5 A was connected to the circuit. Figure 3.5 shows the current inductor and capacitor voltage resulting from the closed loop circuit simulation. Although instantaneous switching takes place many times during the circuit operation, the inductor current does not contain discontinuities. This demonstrates the correct operation of the implemented analog solver. At the beginning of the simulation, the inductor current is zero (initial condition) and the switch is immediately closed by the controller. As no energy is stored in the capacitor (the initial voltage is zero), both the inductor current and the capacitor voltage increase. At time 0.021 s, the upper current limit is reached and the controller opens the switch S. As the inductor current is interrupted (the circuit is open), a voltage impulse is generated by the inductor which turns the diode ON. The energy stored in the inductor is then transferred to the capacitor. The capacitor voltage increases and the inductor current decreases. At time 0.091 s, the lower current limit is reached and the controller closes the switch S again. The negative voltage applied to the diode (due to the connected voltage source) turns it OFF. The short circuit caused by the diode when the switch is closed is properly handled by the proposed algorithm.

In order to evaluate the simulation performance of the implemented MoC, the Buck converter control model was implemented using the

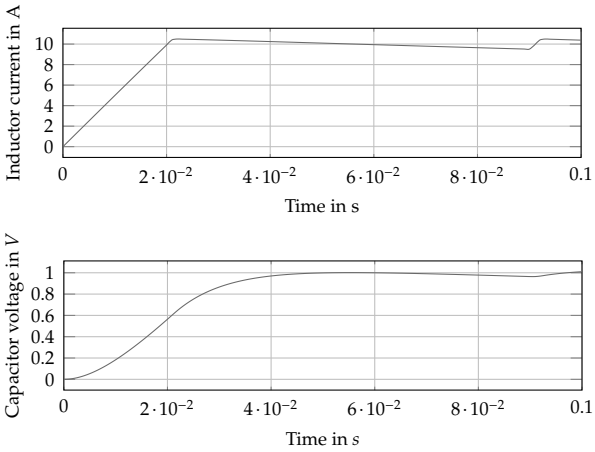


Fig. 3.5: Buck converter simulation results using EPN MoC

ELN MoC in SystemC as well as using the PLECS toolbox in Simulink. As the ELN MoC does not support instantaneous switching, only the first 0.021 *s* (till the lower current threshold is reached) were simulated for comparison. The circuit waveforms were sampled with different sampling times (100 μs and 10 μs) for considering their impact on the simulation performance. The average execution time required for the circuit simulation is presented in Table 3.5. Even for such small model, the presented EPN MoC implementation reached a measurable improvement of the simulation performance.

Simulation execution time in *ms*			Sampling time in μs
SystemC EPN	SystemC ELN	Simulink PLECS	
45.0	50.0	64.5	100
45.0	60.0	68.0	10

Table 3.5: Buck converter simulation execution time

The implemented SystemC AMS extension for power electronic modeling was also validated through the modeling and simulation of a complex high-voltage power converter control utilized in medical machines for x-ray generation. The controlled switched electrical network provides the necessary complexity for a good validation:

- Large number of switches (4 externally controlled switches and 16 internally controlled switches)
- High operation frequency (500 kHz)
- Very large output signal range (30 - 120 kV)

Although the circuit parameter values comprise a very large range, solvability problems were only encountered for one topology (from a total of more than 350 different topologies). The simulation results (signal harmonics) were very close to those obtained with the toolbox PLECS and the achieved simulation performance was even slightly better[6] (around 5%). This is a very good result considering that the reported speed-up of PLECS is greater than $10x$ compared to Saber and $20x$ compared to Simulink/Power System Blockset [4].

3.9 Chapter Summary and Conclusions

The presented SystemC AMS extension for modeling and simulation of power electronic circuits hides switching control details from the models reducing considerably the modeling effort. The experimental results carried out in this work confirm that ideal switched electrical networks are an appropriate behavior abstraction for power electronic circuit modeling at system level. Fast and stable simulations of complex power electronic circuits were achieved with the EPN MoC implementation supporting multiple instantaneous switching. The modeling abstraction of this MoC allows an adequate analysis of signal harmonics for controlled power electronic circuits. Opposite to the current SystemC ELN MoC, the numerical properties of the proposed circuit behavior abstraction minimize solvability issues.

The investigation and analysis activities have shown that:

[6] For more details about the high-voltage power converter simulation see [38].

- Capturing switched networks using a table representation is well suited for circuit analysis and circuit equations reduction.
- Reducing the size of the algebraic equation system that represents the circuit equations leads to more efficient and accurate simulations.
- Matrix pivoting strategies are required to avoid solvability problems.
- Sparse matrix algebraic computation methods are often not appropriate for the behavior computation of power electronic circuits.
- The prediction of switching impulses improves the simulation performance and avoids solvability problems.

Although the modified nodal analysis method introduces some redundant variables in the circuit equations, a fast simulation of switched electrical networks was achieved. The efficient circuit topology computation after switching (prediction of switching impulses) as well as the consistent and compact formulation of circuit equations based on the presented topological analysis significantly contributes to faster and more accurate (less rounding errors) simulations.

The SystemC AMS analog solver utilizes a sparse matrix linear algebraic solver to achieve a good simulation performance. It is not appropriate for the computation of the stiff equations describing power electronic circuits. The ELN MoC suffers from numerical issues which forces the tuning of circuit parameters to achieve stable circuit simulations as shown in [38]. As the algebraic solver implementation of the EPN utilizes matrix pivoting for additionally reducing solvability problems, the observed numerical robustness and performance improvement reinforce the value of this contribution.

Even though an acceptable compromise between accuracy and simulation speed can be reached modeling semiconductor components as ideal switches, linear and nonlinear circuit representations (which avoid behavior details or are more accurate) are also needed for design and verification tasks. Independent of the level of abstraction, the analog circuit behavior computation methods strongly determine the simulation performance. In order to improve the efficiency and accuracy of the analog solver for the circuit behavior computation at several levels of abstractions, computation methods based on orthogonal polynomials were developed in the context of this dissertation. They will be explained after the introduction of a signal abstraction model based on orthogonal functions in the next chapter.

Chapter 4
Signal Modeling for AMS Systems

Signal modeling formalisms are an important research topic to cope with the increasing heterogeneity in embedded system designs. This chapter presents a novel mathematical model of signals for heterogeneous system specification at different abstraction levels which relies on signal coding and signal parameterization. A key advantage of this signal representation is that continuous time signals are efficiently and accurately described by a finite vector of coefficients. Furthermore, a signal subdivision method which can be performed very efficiently on this vector is presented for speeding up AMS system design analysis and verification.

This chapter is organized as follows. After the presentation of the research motivation and related work in section 4.1 and 4.2 respectively, section 4.3 summarizes the research contribution of this chapter. The tagged signal model is reviewed in section 4.4 and the novel vectorial signal model is defined in section 4.5. The general operational approach for analog signal subdivision and its application for fast and accurate crossing event detection as well as for discrete-event and adaptive signal sampling are described in section 4.6, 4.7 and 4.8 respectively. Finally, section 4.10 states the value of the chapter contributions.

4.1 Motivation

In order to obtain a reliable AMS system design, the model abstraction must capture the system properties that describe the design problem. Formal descriptions allow the automatic model analysis and synthesis. In embedded system design, a system is usually represented by a process network. The behavior of a process (computation) and the interaction between processes (communication) is described through signals. As signals are the operands and the results of computation processes, a signal model that formally copes with heterogeneous descriptions at different abstraction levels is essential to allow an efficient implementation of several modeling formalisms (MoCs).

4.2 Related Work

Utilizing set theory, Zeigler formalized the concept of signals (called trajectories) for describing the interaction between different types of dynamical systems [130]. This theoretical framework is utilized as background for the implementation of several DEVS related formalisms.

Lee proposed later a *denotational framework*, called the *tagged signal model*, which considers signals as a collection of events and captures the properties of common modeling formalisms utilized in embedded system design, such as for example synchronous reactive model of computation [70] [69] [68]. Many frameworks for embedded system design, such as Ptolemy and ForSyDe, are based on this signal model.

Jantsch et al. recognized the importance of time representation for the efficient design and validation of complex safety critical embedded systems [54]. Focused on heterogeneous MoCs, Zhu et al. presented a *formal model* of *continuous time signals* that is defined using a *time continuum* and enables the efficient representation of signals in communications systems [133]. Instead of using *time events*, continuous time signals are represented as a *concatenation of sub-signals*. As symbolic integration is not suitable for simulation, SystemC ForSyDe does not take really advantage from the symbolic analog signal representation. It uses a one step integration method (Runge-Kutta) with variable sampling time to cope with dynamical systems such as signal filters which forces the discretization of continuous time signals. The continuous time signal monitoring for model analysis is even carried out using a *fixed sample rate* which lead to slow simulations.

In order to efficiently sample continuous time signals, Andrade et al. presented a time step control mechanism that is based on the slope between the current and the previous signal value and assumes signal continuity for determining the next time step [6]. This simple sampling concept requires many parameters such as the maximum admissible amplitude error, which must be previously defined for each controlled signal and does not provide a good sampling.

Neither the time event signals nor the (symbolic) time continuum signals are able to cope with the trade-off between efficiency and accuracy for both representing and processing continuous time signals. A formal signal description that captures the properties of both analog and digital components is necessary for a suitable modeling and analysis of AMS systems.

4.3 Contribution to Formal Signal Modeling

The aim of the research work presented in this chapter was to find out a suitable signal representation that is able to cope with the challenge of efficient and accurate behavior modeling in heterogeneous embedded systems. The main research activities carried out to achieve this goal have been the following:

- To investigate the efficiency of existing formal methods for signal representation.
- To derive a general signal representation that accurately captures the signal dynamic at several levels of abstraction.
- To evaluate the suitability of several orthogonal functions for the accurate representation of analog and digital signals.
- To find out an analytical solution for the continuous time and value signal subdivision problem.
- To devise an efficient computation algorithm for the quantization of analog signals.
- To derive a simple and accurate computation algorithm for adaptive signal sampling.

This work proposes a novel two-dimensional signal model for representing both analog and digital signals. Using a suitable parameterization of continuous time signals, the trade-off between accuracy and efficiency is reduced to a minimum. The mathematical properties of a given orthogonal polynomials family are exploited for the efficient representation of continuous time signals as well as for the conception of signal processing methods based on algebraic operations. Particularly, a matrix for continuous signal subdivision is presented and utilized for the efficient and reliable sampling of continuous time signals. Thus, the proposed formalism for mixed signal modeling overcomes the practical problems of existing theoretical frameworks.

4.4 The Tagged Signal Model

The *tagged signal model* is a *functional representation* of signals. A signal $s(\tau)$ is represented as a set of events $\{e_i\}$. Each event e_i maps a tag τ_i to a value v_i [70].

$$s(\tau) = \{e_i : \tau_i \rightarrow v_i\} \quad \forall \quad \tau_i \in \mathbb{T}, v_i \in \mathbb{V} \tag{4.1}$$

The set of tags $\{\tau_i\}$ is partially or totally ordered [70].

4.5 The Mixed Orthogonal Signal Model

Although tags capture the main properties of models of computation such as time abstraction, precedence relationships and event synchronization, they require a large number of events to accurately represent continuous time signals. In order to efficiently describe signals at different levels of behavior abstraction, this work applies the following set of transformations for the reduction of the number of events needed to represent timed signals:

1. Representing signals in vectorial form.
2. Coding signals in a signal space.
3. Representing continuous signals in terms of orthogonal functions.

After these transformations continuous time signals are captured by a vector of coefficients. This enable the analysis and transformation of signals using simple algebraic mathematical operations.

4.5.1 Representing Timed Signals in Vector Spaces

Any function $s(\tau)$ describing a *continuous time signal* can be expressed as an ordered set of values $[v_i]$ in an infinite dimensional space $\mathbb{V}^\infty$ such as each element v_i of the vector corresponds to a tag and vice versa $(\tau_i \rightarrow v_i)$.

$$s(\tau) = [v_i]_\infty \quad \forall \tau \in \mathbb{R}, \quad [v_i] \in \mathbb{V}^\infty \tag{4.2}$$

In this vectorial signal representation, a timed signal is considered as a point in an infinite dimensional space.

The domain $\mathbb{V}$ may be any data type. In this chapter real valued signals ($\mathbb{V} = \mathbb{R}$) are considered (time domain) but complex valued signals ($\mathbb{V} = \mathbb{C}$) are also possible (frequency domain).

In order to denote vectors (and matrices) square brackets with subscripts are used. Thus, confusion between scalar numbers and vectors is avoided. Furthermore, the dimension of vectors (and matrices) is explicitly defined to visualize dimension changes on signals.

Example 4.1. Consider the signal $s(\tau)$ resulting from the sampling of a continuous time signal at the fixed times τ_1, τ_2 and τ_3. This signal only contains 3 events $s(\tau) = \{e_1, e_2, e_2\}$. Using the *Dirac delta function* δ this signal can be defined in the time domain as $s(\tau) = v_1 \cdot \delta(\tau - \tau_1) + v_2 \cdot \delta(\tau - \tau_2) + v_3 \cdot \delta(\tau - \tau_3)$. As the basis vectors $[\tau_i]_1$ of the vector space $\mathbb{V}^\infty$ univocal capture each time ordered tag τ_i, the vectorial signal representation is given by $s(\tau) = v_1 \cdot [\tau_1]_1 + v_2 \cdot [\tau_2]_1 + v_3 \cdot [\tau_3]_1$ where the vectors $[\tau_i]_1$ are unitary and orthogonal.

The conceptual difference between the functional and the vectorial signal representation is shown in Fig. 4.1.

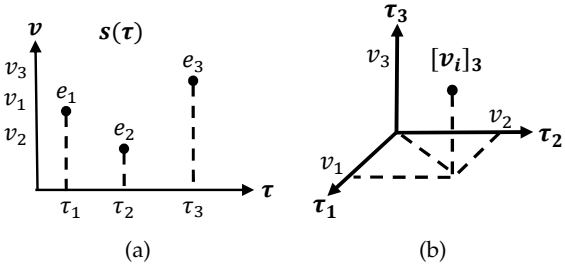


Fig. 4.1: Signal representation: a) Functional b) Vectorial

The *geometric properties* of vector signals allow the systematic definition of the mathematical operations that can be applied to signals. For example, it is possible to define the closeness of two signals as the (euclidean) distance between two points.

In this work, it is considered that vector signals are defined in a *linear vector space*. This means that the multiplication of a signal $s(\tau)$ by a scalar number a and the addition of two arbitrary signals $s_1(\tau)$ and $s_2(\tau)$ are defined.

$$a \cdot s(\tau) = a \cdot [v_i]_\infty = [a \cdot v_i]_\infty, \quad a \in \mathbb{R} \tag{4.3a}$$

$$s_1(\tau) + s_2(\tau) = [v_{i1}]_\infty + [v_{i2}]_\infty = [v_{i1} + v_{i2}]_\infty \tag{4.3b}$$

These properties are utilized in the next section for signal coding.

4.5.2 Coding Signals in a Signal Space

In order to optimize the representation of signals, this work describes a signal $s(\tau)$ by means of a *linear combination* of *elementary signals* $s_e(\tau)$ which are defined in a finite interval $[0, T_e]$ and mapped to a finite interval $[\tau_k, \tau_k + T_k]$ by applying elementary signal operations as follows:

$$s(\tau) = \sum_{k=0}^{\infty} s_k(\tau) = \sum_{k=0}^{\infty} a_k \cdot s_e(\rho_k \cdot \tau - \tau_k) = \sum_{k=0}^{\infty} a_k \cdot [v_{ik}]_\infty \tag{4.4}$$

where a_k and $\rho_k = T_e/T_k$ are scaling factors. Thus, *amplitude scaling*, *time scaling* (time compression or time expansion) and *time shifting* are basic operations allowed on elementary signals.

The set of elementary signals $\{s_e(\tau)\}$ is known as the *signal space*. Representing signals in terms of elementary signals is useful to model signal sources efficiently.

For convenience, the elementary signals $s_e(\tau)$ have a finite duration and the overlapping of elementary signals is allowed.

Example 4.2. Consider now the signal $s(\tau)$ resulting from the sampling of a sine waveform over two periods T. This signal can be represented as $s(\tau) = s_e(\tau) - s_e(\tau - T/2) + s_e(\tau - T) - s_e(\tau - 3 \cdot T/2)$, where the elementary signal $s_e(\tau)$ is defined by the sampling events in the interval [0, 0.5] (See Fig. 4.2).

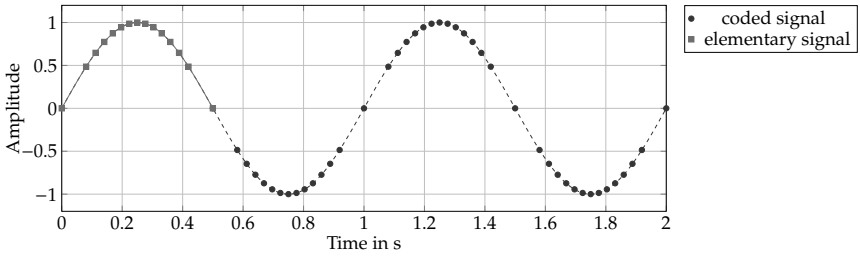


Fig. 4.2: Coding a sampled sine wave signal

4.5.3 Parameterizing Signals in a Vector Space

In order to approximate continuous time elementary signals with a finite set of vector elements, each elementary signal $s_e(\tau)$ is expanded in terms of a *infinite vector space of linearly independent signals* $[f_j(\sigma)]_\infty$ defined in a finite interval $[\sigma_a, \sigma_b]$. This requires that the elementary signals are defined in a *normed inner product space* $\mathbb{V}^\infty$ and implies that:

1. The *norm of a signal* $s(\tau)$ in an interval $[\tau_a, \tau_b]$ is defined by the expression:

$$\| s(\tau) \| = \sqrt{\int_{\tau_a}^{\tau_b} s^2(\tau) d\tau} \tag{4.5}$$

2. The *inner product of two signals* $s_1(\tau)$ and $s_2(\tau)$ in an interval $[\tau_a, \tau_b]$ is defined by the expression:

$$s_1(\tau) \bullet s_2(\tau) = \int_{\tau_a}^{\tau_b} s_1(\tau) \cdot s_2(\tau) d\tau \tag{4.6}$$

Furthermore, it is also assumed that the elementary signals $s_e(\tau)$ have a *finite energy* E_{s_e}. Therefore, the totally of infinite sequences $\{s_k(\tau)\}$ satisfy:

$$E_{s_k} = \int_{\tau_k}^{\tau_k + T_k} s_k^2(\tau) d\tau < +\infty \tag{4.7}$$

Each *basis signal* $f_j(\sigma)$ must satisfy the following equation to be *linear independent*:

$$\int_{\sigma_a}^{\sigma_b} \omega(\sigma) \cdot f_p(\sigma) \cdot f_q(\sigma)\, d\sigma = k_{pq} \cdot \delta_{pq}, \quad k_{pq} \in \mathbb{R} \tag{4.8a}$$

$$\delta_{pq} = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases} \tag{4.8b}$$

where $\omega(\sigma)$ is a *weighting function* and k_{pq} is a constant value.

Each elementary vector signal in the *signal space* $\{s_e(\tau)\}$ is then expanded in terms of the *orthogonal basis signals* $f_j(\sigma)$ by finding a set of coefficients $\{a_{ej}\}$ that satisfy:

$$s_e(\sigma) = \sum_{j=0}^{\infty} a_{ej} \cdot f_j(\sigma) = [a_{ej}]_{\infty}^{T} \cdot [f_j(\sigma)]_{\infty}, \quad a_{ej} \in \mathbb{R} \quad \forall \quad \sigma \in [\sigma_a, \sigma_b] \tag{4.9}$$

Using a suitable vector space of orthogonal basis functions $[f_j(\sigma)]_{\infty}$ each elementary signal $s_e(\sigma)$ can be accurately approximated by limiting the infinite sum Eq. (4.9) to a finite number of terms Q:

$$s_e(\sigma) \approx \sum_{j=0}^{Q} a_{ej} \cdot f_j(\sigma) = [a_{ej}]_{Q}^{T} \cdot [f_j(\sigma)]_{Q}, \quad a_{ej} \in \mathbb{R} \quad \forall \quad \sigma \in [\sigma_a, \sigma_b] \tag{4.10}$$

As the ordered set of orthogonal basis functions $[f_j(\sigma)]_Q$ is known, the representation of the elementary signals can be parameterized by describing each elementary signal using only the *coefficient vector* $[a_{ej}]_Q$.

Expanding Continuous Elementary Vector Signals

A *continuous time and value elementary signal* $\tilde{s}_e(\tau)$ (analog signal) can be expanded over a series of *continuous orthogonal basis signals* $\tilde{f_n}(\sigma)$ by a simple vector-matrix multiplication.

$$\tilde{s}_e(\sigma) = \sum_{n=0}^{+\infty} c_{en} \cdot \tilde{f_n}(\sigma) = [c_{en}]_{\infty}^{T} \cdot [\tilde{f_n}(\sigma)]_{\infty} \quad \forall \quad \sigma \in \mathbb{R}, \sigma \in [\sigma_a, \sigma_b] \tag{4.11}$$

As any *continuous function* (small input changes result in arbitrarily small output changes) on a finite interval can be uniformly approxi-

mated by a polynomial (Weierstrass theorem), this work considers that each *continuous elementary signal* $\tilde{s}_e(\tau)$ and its first derivative are continuous and defined in a finite interval $[\sigma_a, \sigma_b]$. Therefore, any *continuous elementary signal* $\tilde{s}_e(\tau)$ can be approximated with good accuracy (least squares approximation) using a reduced number of *orthogonal polynomials* Q_n as *basis signals*.

$$\tilde{s}_e(\sigma) \approx \sum_{n=0}^{N-1} c_{en} \cdot Q_n(\sigma) = [c_{en}]_N^T \cdot [Q_n(\sigma)]_N \tag{4.12}$$

The most suitable orthogonal polynomials for analog system simulation are *Legendre* $P_n(\sigma)$ and *Chebyshev* $T_n(\sigma)$ polynomials [92][67]. In both cases, the least square error is distributed nearly uniformly in the approximation interval. They are orthogonal in the interval [-1, 1] and defined by the following power expressions:

$$P_n(\sigma) = \frac{1}{2^n} \sum_{j=0}^{\lfloor n/2 \rfloor} \frac{(-1)^j}{j!} \cdot \frac{(2n-2j)!}{(n-j)! \cdot (n-2j)!} \cdot \sigma^{n-2j} \tag{4.13}$$

$$T_n(\sigma) = \frac{n}{2} \sum_{j=0}^{\lfloor n/2 \rfloor} (-1)^j \cdot \frac{(n-j-1)!}{(j)! \cdot (n-2j)!} \cdot (2\sigma)^{n-2j} \tag{4.14}$$

Such *classical orthogonal polynomials* are characterized by a set of *recursion relations* that is useful to efficiently compute numerical operations [96].

Example 4.3. Taylor series are often used for circuit behavior computation. This example shows the superior accuracy of orthogonal polynomials. Consider the polynomial approximation of the continuous signal $s_e(\tau) = 1 - e^{-\tau}$ in the interval [0.0, 1.5] using 3, 4 and 5 terms (coefficients). Fig. 4.3 shows the approximation results a) for a *truncated Taylor power series* and b) for a *Chebyshev polynomials expansion*. The truncated Taylor series requires more than 5 terms to achieve a good accuracy (maximal error ≈ 5%). The 3 terms Chebyshev polynomials expansion achieves a higher accuracy (maximal error ≈ 1%). Furthermore, the error is uniformly distributed in the signal approximation interval. At N Chebyshev points (denoted explicitly in Fig. 4.3), there is no approximation error.

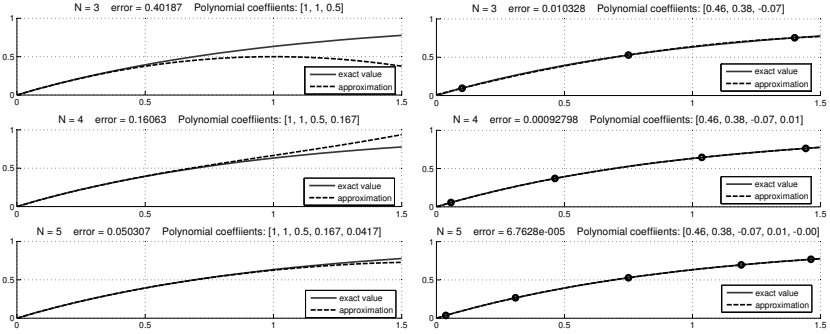


Fig. 4.3: Polynomial approximation: a) Taylor b) Chebyshev

Expanding Discrete Value Elementary Vector Signals

A *discrete value elementary vector signal* $\bar{s}_e(\tau)$ can be expanded using a finite linear combination of *flat orthogonal basis signals* $\bar{f}_m(\sigma)$ such as block pulse, Walsh or Haar-Wavelets which are *piece-wise constant*.

$$\bar{s}_e(\sigma) \cong \sum_{m=0}^{M-1} b_{em} \cdot \bar{f}_m(\sigma) = [b_{em}]_M^T \cdot [\bar{f}_m(\sigma)]_M \quad \forall \quad \sigma \in \mathbb{R}, \sigma \in [\sigma_a, \sigma_b] \quad (4.15)$$

Continuous time digital signals can be accurately expanded over the time interval $[\sigma_a, \sigma_b]$ by defining the discrete orthogonal basis signals in terms of the *generalized block pulse function* $B_m(\sigma)$ which is defined as follows:

$$B_m(\sigma) = \begin{cases} 1 & \text{if } \sigma_m \le \sigma < \sigma_m + T_m \\ 0 & \text{otherwise} \end{cases} \quad (4.16a)$$

$$T_m = \sigma_{m+1} - \sigma_m, \quad \sigma_0 = \sigma_a, \quad \sigma_N = \sigma_b \quad (4.16b)$$

Thus, in this work digital signals in continuous time are represented by the expression:

$$\bar{s}_e(\sigma) = \sum_{m=0}^{M-1} b_{em} \cdot B_m(\sigma) = [b_{em}]_M^T \cdot [B_m(\sigma)]_M \tag{4.17}$$

Example 4.4. Fig. 4.4 shows a) the parameterization of a *continuous time and value elementary signal* using *Chebyshev polynomials* (See previous example) and b) the parameterization of a *discrete value elementary signal* using the *block pulse function*. The coefficients c_i and b_i describing the signals are represented as *vertical segments* on the orthogonal axis. They scale the corresponding orthogonal functions, which are added for the computation of the elementary signal. In both cases, only 3 coefficients are required for the representation of the continuous time signals.

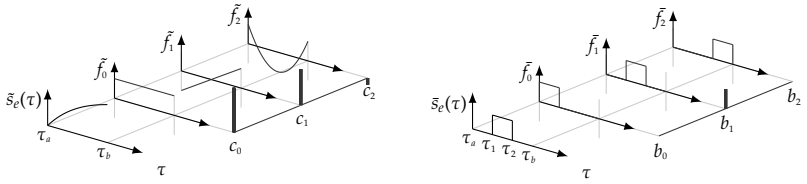


Fig. 4.4: Signal expansion: a) Continuous (analog) b) Discrete (digital)

Expanding Mixed Dynamic Elementary Vector Signals

Equations (4.12) and (4.17) allow the definition of continuous and discrete value elementary vector signals in a finite dimensional vector space $\mathbb{V}^N$ and $\mathbb{V}^M$ respectively. In order to obtain a *general expression* for representing *continuous time signals* (including discontinuities), each elementary signal $s_e(\tau)$ is expanded in this work using an *ordered set of orthogonal piece-wise continuous signals* $[H_{mn}(\sigma)]_{NM}$ as follows:

$$s_e(\sigma) \approx \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} h_{emn} \cdot H_{mn}(\sigma) = [[h_{emn}]_N^T]_M^T \cdot [[H_{mn}(\sigma)]_N]_M \tag{4.18a}$$

$$H_{mn}(\sigma) = \begin{cases} g(\sigma) \cdot Q_n(\rho_m \cdot \sigma) & \text{if } \sigma_m \le \sigma \le \sigma_m + T_m \\ 0 & \text{otherwise} \end{cases} \tag{4.18b}$$

$$\rho_m = (\sigma_b - \sigma_a)/T_m, \quad T_m = \sigma_{m+1} - \sigma_m, \quad \sigma_0 = \sigma_a, \quad \sigma_N = \sigma_b \tag{4.18c}$$

$$[[h_{emn}]_N^T]_M^T = [vec([h_{emn}]_{MN}^T)]^T, \quad [[H_{mn}]_N]_M = vec([H_{mn}]_{MN}^T) \tag{4.18d}$$

where the function $g(\sigma)$ is a *constant value* which depends on the flat orthogonal basis set. In case of the *block pulse function*, $g(\sigma) = 1$.

The coefficients h_{emn} enable the reliable description of analog signal that contain discontinuities. For convenience, they are represented in *matrix form* $[h_{emn}]_{MN}$ (See In Fig. 4.5). Each matrix row defines the coefficients of the polynomial expansion (continuous signal). Signal discontinuities (and concatenation) are captured using multiple matrix rows.

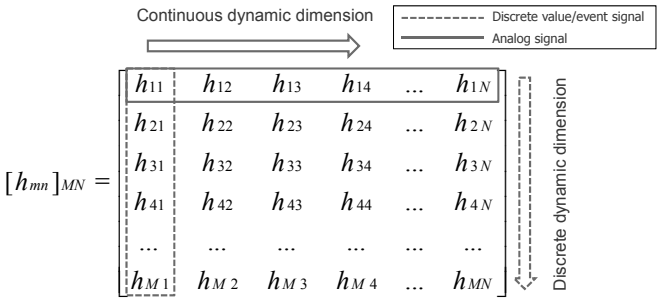


Fig. 4.5: Mixed-signal coefficient matrix

The *coefficient matrix* $[h_{emn}]_{MN}$ and the *matrix of orthogonal piece-wise continuous signals* $[H_{mn}(\sigma)]_{NM}$ are vectorized in Eq. (4.18) for the computation of the mixed dynamic signal expansion. If the matrix $[h_{emn}]_{MN}$ is reduced to a *row vector* (M = 1), the coefficients represent a *continuous time and value signal* in the interval $[\sigma_a, \sigma_b]$. If the matrix $[h_{emn}]_{MN}$ is reduced to a *column vector* (N = 1), the coefficients represent a *discrete value signal* in the interval $[\sigma_a, \sigma_b]$.

Handling Discrete Time Elementary Vector Signals

A *discrete time elementary vector signal* $\hat{s}_e(\tau)$ consists of a *sequence of events* $[e_i]$ in time [71]. It can be considered as a special case of a discrete value elementary signal, in which the values of the signal are only defined at the times σ_m. Thus, a *discrete signal* is not defined over a continuous time but rather at a *set of time instants* at which events occurs (discrete time). They are captured using the Dirac delta function δ. Defining $g(\sigma) = \delta(\sigma - \sigma_m)$ in Eq. (4.18), the previously presented signal model is able to capture such *instantaneous events* in time.

As the time T_m between two events is defined and satisfies $\sigma_m < \sigma_{m+1}$, the *discrete events* e_i representing the signal are *order preserving*. Therefore, *discrete events* can be counted off in *temporal order*. An important observation is that M may be zero. It means, that the signal does not contain any event in the time interval $[\sigma_a, \sigma_b]$ and it is represented by an *empty vector* $[]_0$. As a vector space may not be empty, this special case must be properly handled in the implementation.

Handling Untimed Elementary Vector Signals

A *discrete system* is defined in terms of *event triggered reactions*. Such events are typically clock edges (synchronous) or data tokens (asynchronous). An *untimed elementary vector signal* $\dot{s}_e(n)$ consist only of a *sequence of values* $[v_i]$. The time at which events occur is abstracted. Defining $g(\sigma) = 1$ and $Q_n(\sigma) = 1$ in Eq. (4.18), the previously presented signal model is able to capture such *untimed events*.

In order to cope with *synchronous events* (events occurring at the same time σ_m), it is necessary to capture the *absence* of a signal event. The symbol $\perp$ is used to denote such *empty value event*. It is represented by a *zero dimension row* $[]_{m0}$ in Eq. (4.18). For handling *multi-rate signals* a vector space $\mathbb{V}^k$ with a dimension $k > 1$ is utilized. Thus adaptive rate systems can be captured. Both cases must be properly handled by the implementation. Note that for untimed signals, the columns and the rows of the coefficient matrix $[h_{emn}]_{MN}$ are assigned to synchronous and asynchronous events respectively. Handling *asynchronous events* requires that $M \leq 1$ in Eq. (4.18). Therefore, synchronous signals should capture more than one event ($M > 1$) to be correctly interpreted.

4.6 Operational Subdivision of Analog Signals

As will be shown in chapter 5, the subdivision of a continuous signal derived in this section is very useful for the efficient handling of the interaction between analog and digital systems.

In order to find an operational approach for the subdivision of a polynomial function, the function evaluation points σ are mapped from the interval [-1 , 1] (in which the orthogonal polynomials Q_n are defined) to the sub-interval $[\sigma_a , \sigma_b]$ using the following expression:

$$\sigma_{ab} = \frac{1}{2} \cdot (\sigma_b - \sigma_a) \cdot \sigma + \frac{1}{2} \cdot (\sigma_a + \sigma_b) \tag{4.19}$$

The *Chebyshev polynomials basis matrix* $[B_T(a,b)]_{N^2}$ in the sub-interval $[\sigma_a , \sigma_b]$ is then derived using the recurrence property of Chebyshev polynomials [96]:

$$T_n(\sigma_{ab}) = \sigma_{ab} \cdot T_{n-1}(\sigma_{ab}) - T_{n-2}(\sigma_{ab}) \tag{4.20}$$

Any orthogonal polynomial $Q_n(\sigma)$ can be expressed as follows:

$$Q_n(\sigma) = [\sigma^n]_N^T \cdot [B_Q]_{N^2} \tag{4.21}$$

where the matrix $[B_Q]_{N^2}$ represents the *basis of the polynomial* in terms of the *power vector* $[\sigma^n]_N = [\sigma^N \quad \dots \quad \sigma^2 \quad \sigma \quad 1]$. The *basis matrix* for the Chebyshev polynomials can be computed using equation (4.13). For a 3^{rd} order polynomial the Chebyshev basis matrix $[B_T]_{4^2}$ is:

$$[B_T]_{4^2} = \begin{bmatrix} 0 & 0 & 0 & 4 \\ 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & -3 \\ 1 & 0 & -1 & 0 \end{bmatrix} \tag{4.22}$$

Using equation (4.21) the subdivision matrix $[S_{a,b}]$ can be computed as:

$$[S_Q(a,b)]_{N^2} = [B_Q]_{N^2}^{-1} \cdot [B_Q(a,b)]_{N^2} \tag{4.23}$$

Thus, the coefficients $[c_n(a,b)]_N$ of the polynomial expansion for any signal in a given sub-interval $[\sigma_a , \sigma_b]$, can be computed by a simple matrix multiplication.

$$[c_n(a,b)]_N = [S_Q(a,b)]_{N^2} \cdot [c_n]_N \tag{4.24}$$

For a 4 terms Chebyshev polynomials analog signal expansion, the subdivision matrix $[S_T(a,b)]$ is given by:

$$[S_T(a,b)]_{4^2} = \begin{bmatrix} 1 & \frac{a+b}{2} & \frac{3\cdot(a^2+b^2)+2\cdot a\cdot b-4}{4} & \frac{5\cdot(a^3+b^3)+3\cdot(a^2\cdot b+a\cdot b^2)-6\cdot(a\cdot b)}{4} \\ 0 & \frac{b-a}{2} & \frac{b^2-a^2}{2} & \frac{15\cdot(b^3-a^3)+3\cdot(a\cdot b^2-a^2\cdot b)+12\cdot(a-b)}{8} \\ 0 & 0 & \frac{a^2+b^2-2\cdot a\cdot b}{4} & \frac{3\cdot(a^3+b^3-a^2\cdot b-a\cdot b^2)}{4} \\ 0 & 0 & 0 & \frac{b^3-a^3+3\cdot(a^2\cdot b-a\cdot b^2)}{8} \end{bmatrix} \tag{4.25}$$

For Legendre polynomials, the *subdivision matrix* $[S_P(a,b)]_{N^2}$ can be computed in the same way using the basis matrix $[B_P(a,b)]_{N^2}$ and the following recurrence relation:

$$P_n(\sigma_{ab}) = \frac{1}{n}\cdot\left((2\cdot n-1)\cdot\sigma_{ab}\cdot P_{n-1}(\sigma_{ab}) - (n-1)\cdot P_{n-2}(\sigma_{ab})\right) \tag{4.26}$$

Example 4.5. Consider the binary subdivision of the elementary signal $s_e(\sigma) = [0, -0.5758, 0, 0.7711]\cdot[T_n(\sigma)]_4$ which corresponds to a sine waveform in the interval $[0, \pi]$. The subdivision matrices are:

$$[S_T(-1,0)]_{4^2} = \begin{bmatrix} 1 & 1/2 & -1/4 & -1/4 \\ 0 & 1/2 & 1 & 3/8 \\ 0 & 0 & 1/4 & 3/4 \\ 0 & 0 & 0 & 1/8 \end{bmatrix} \quad [S_T(0,1)]_{4^2} = \begin{bmatrix} 1 & -1/2 & -1/4 & 1/4 \\ 0 & 1/2 & -1 & 3/8 \\ 0 & 0 & 1/4 & -3/4 \\ 0 & 0 & 0 & 1/8 \end{bmatrix}$$

Applying the subdivision 2 times ($S = 1, 2$) to the signal coefficients, the following coefficient matrices are obtained:

$$[h_{emn}]_{2x4} = \begin{bmatrix} 0.4807 & 0.0012 & -0.5783 & 0.0963 \\ -0.4807 & 0.0012 & 0.5783 & 0.0963 \end{bmatrix} \quad [h_{emn}]_{4^2} = \begin{bmatrix} 0.6487 & 0.6151 & -0.2169 & 0.0120 \\ 0.6018 & -0.5416 & -0.0722 & 0.0120 \\ -0.6018 & -0.5416 & 0.0722 & 0.0120 \\ -0.6487 & 0.6151 & 0.2169 & 0.0120 \end{bmatrix}$$

Note that the subdivision matrices can be multiplied for carrying out successive subdivisions. As the last coefficient of the subdivision matrices decreases exponentially in each step $\left(\frac{1}{(2^N)^S} = \frac{1}{8}, \frac{1}{64}, \ldots\right)$, less coefficients are needed after several steps.

4.7 Computing Threshold Crossing Events

The polynomial signal representation introduced in section 4.5 can be exploited for accurate threshold crossing detection by using polynomial root finding methods. As explained in [97], eigenvalue methods can be applied for finding the roots of arbitrary polynomials but they are computationally intensive and not appropriate for simulation purposes. To overcome this limitation, this work takes advantage of the Chebyshev polynomials properties for the identification of crossing event free intervals. As they are ranged between ±1 in the interval $[\sigma_a, \sigma_b]$, the values of the corresponding signal expansion are limited to the range:

$$\left[c_0 - \sum_{n=1}^{N-1} |c_n|, c_0 + \sum_{n=1}^{N-1} |c_n|\right] \tag{4.27}$$

Any other polynomial expansion can be mapped to Chebyshev polynomials using Eq. (4.21).

In order to quickly find the intervals containing crossing events, precomputed subdivision matrices are utilized to divide the signal interval into smaller intervals. Truncating then the Chebyshev polynomial expansion to a 3^{rd} order polynomial, the crossing point is computed analytically. In case that the truncation error is large, the root value is polished using only a few Newton-Raphson iterations.

4.8 Sampling Analog Signals

For many applications such as digital control, analog signals are sampled periodically in time (Riemann sampling). This *time triggered sampling* method simplifies system analysis and design [7].

For some systems, *event based sampling* provides a better performance (e.g. the control action takes place only if it is required). This sampling technique takes samples at the points at which a signal crosses certain predefined values (Lebesgue sampling) [7]. Many digital sensors work in this way.

For signal analysis applications, the reconstruction of a signal from its discrete samples becomes faster if *adaptive sampling* (irregular sampling)

takes place. Sampling non-uniformly a signal depending on its shape enables a better reconstruction accuracy for different algorithms [95].

The previously presented orthogonal signal model allows a very efficient sampling for all theses applications. It is explained in the following sections. An efficient adaptive sampling algorithm based on Chebyshev polynomials is proposed in section 4.8.3.

4.8.1 Periodic Sampling

As the orthogonal polynomials utilized for the parameterization of continuous signals are defined in the interval [-1, 1], the vector $[[H_{mn}(\sigma)]_N]_M$ in Eq. (4.18) does not usually change for a fixed sampling rate. Thus, the resulting *sampling matrix* can be computed once at simulation start. Periodic sampling is therefore performed as a vector matrix multiplication, which can be computed faster than function evaluations, particularly for trigonometric functions.

4.8.2 Event Based Sampling

The *event based sampling* of continuous time signals requires a *time step control*. Applying the threshold crossing detection method presented in section 4.7 for the predefined signal threshold values, event based sampling is carried out fast and accurately.

4.8.3 Adaptive Sampling

The implementation of an accurate *adaptive sampling method* for continuous time signals is an heuristic task and requires a considerable computation effort for a robust algorithm. In order to achieve a faster and more efficient sampling of continuous time and value signals, this work proposes to successively subdivide a signal using the operational method presented in section 4.6. The *recursive subdivision* is stopped

when the error that would be introduced by approximating each sub-signal to a line is smaller than a given tolerance value ϵ.

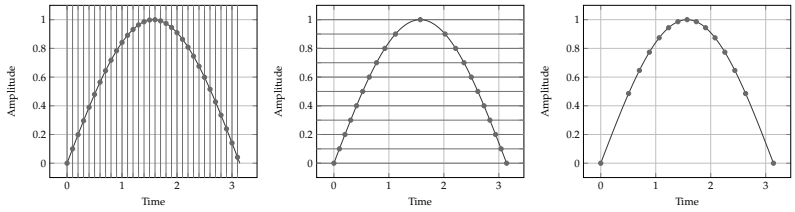
For orthogonal polynomials such as Chebyshev and Legendre, the fast convergence rate enables the approximation of the *truncation error* with a very good accuracy.

The simplest method for *interval partitioning* is the *binary subdivision*. That is, the interval is partitioned into two *equidistant sub-intervals*. In order to distribute the sampling error more uniformly, a *non-symmetric signal subdivision* is necessary. To this end, the signal point at which the perpendicular distance to the line connecting the start and end signal value achieves the maximum value is computed. It corresponds to one of the signal points in which the curve has a maximum or minimum value. Taking profit again of the Chebyshev polynomials properties, this *optimal sub-division point* is computed solving first the following equation:

$$\frac{d}{d\sigma}(12 \cdot c_3 \cdot \sigma^2 + 4 \cdot c_2 \cdot \sigma + c_1 - 3 \cdot c_3 - \bar{c}_1) = 0, \quad \bar{c}_1 = \sum_{n=0}^{N/2} c_{2 \cdot n+1} \tag{4.28}$$

and then finding the maximal value of all inflexion points.

Example 4.6. Fig. 4.6 shows the sampling of a sine waveform analog signal applying the previously described methods. Event based sampling (b) is more efficient than periodic sampling (a) but less accurate near the inflexion point. Using an error tolerance $\epsilon = 0.005$, the adaptive sampling method (c) achieves an optimal sampling.



(a) (b) (c)

Fig. 4.6: Signal sampling: a) Periodic b) Event-based c) Adaptive

4.9 Implementation

As shown in Fig. 4.7 the *orthogonal signal model* was implemented as extension of *SystemC AMS*. The proposed signal representation is domain independent but *polymorphic* with respect to the *orthogonal base functions* used for signal parameterization. The base class **sca_osf_signal_base** is responsible for *signal registration* of a derived signal class in the parent module. The methods of this base class are overloaded by the domain specific implementation. In case of Fig. 4.7, the class **sca_osf_tdf_signal** implements the orthogonal signal model for the timed synchronous data flow model of computation TDF. As the class **sca_osf_signal_expansion** implements a common signal data representation, domain interfaces are not necessary for the interaction of orthogonal signal based domains (e.g. continuous time, discrete event, etc.). Thus, the implementation of domain interfaces can be avoided. In order to support interchangeability of modules in existing SystemC models, domain interfaces to SystemC AMS and SystemC ForSyDe are provided. The application of the signal model for behavior computation and analysis is presented in the next chapters.

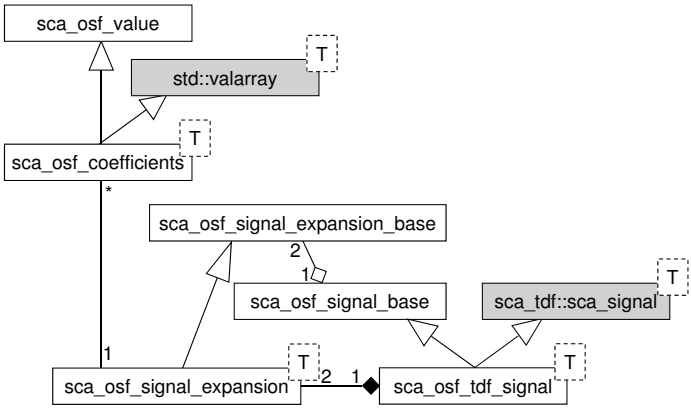


Fig. 4.7: Signal model implementation in SystemC AMS

4.10 Chapter Summary and Conclusions

The presented *mathematical signal model* allows the description of *continuous time signals* through small size *coefficient vectors* and the definition of computational operations on signals in terms of *vector and matrix operations* such as e.g. the signal subdivision presented in section 4.6. Representing *analog signals* using a linear combination of *orthogonal polynomials* significantly contributes to the definition of efficient behavior computation and analysis methods. It relies on the convergence and recurrence properties of the proposed polynomials family. In this chapter efficient computational methods for threshold crossing detection and adaptive sampling of continuous time and value signals were derived. As shown in section 4.5.3 and 4.5.3, discrete event signals and untimed signals can also be represented using the vectorial signal model.

A key advantage of the presented signal model is the suitability to maintain a *strict separation of communication and computation*. Taking profit of the behavior orthogonalization, continuous and discrete behavior as well as synchronous and asynchronous signals are suitably captured. Thus, the presented signal model can be applied for modeling embedded system components at each possible level of behavior abstraction and does not require the modification of module interfaces. The vectorial signal implementation presented in section 4.9 is domain polymorphic reducing considerably the implementation effort for heterogeneous system models. This overcomes the limitation of representing mixed signals with different modeling methodologies such as in *SystemC AMS* or *SystemC ForSyDe* which always require domain interfaces.

As this work focuses on control system applications, only a subset of operations on signals were considered (amplitude and time scaling as well as time shifting). Other useful signal operations such as amplitude shifting, time reversal and time reflection can be considered in future work towards modeling communication systems.

The smart vectorial signal representation based on orthogonal functions constitutes the backbone for efficient and seamless embedded system design activities. As will be presented in the next chapters, system behavior computation, analysis and verification benefit from the presented signal model.

Chapter 5
Modeling and Simulation of AMS Systems

Capturing the behavior of AMS system components at several abstraction levels is fundamental for supporting a seamless design methodology based on system properties. In order to enable the fast simulation of analog system components, efficient, stable and accurate computational methods are required. A current challenge regarding the design of AMS systems is the component heterogeneity. Circuit behavior computation methods that are able to efficiently handle the interaction between the analog and digital parts are needed for fast system level simulations.

This chapter presents operational methods for the efficient and accurate computation of analog circuits' behavior at different abstraction levels. A novel operational method that does not require the linearization of analog circuit equations is proposed for efficient nonlinear circuit simulations. Furthermore, an orthogonal signals based method for the efficient computation of digital circuits' behavior is proposed. The threshold crossing operational computation method presented in chapter 4 is applied for the efficient and accurate handling of the interaction between the continuous and discrete time system parts. The experimental results show that the orthogonal signal based computation methods enable faster simulations of large AMS circuits.

This chapter is organized as follows. After the presentation of the research motivation and related work in section 5.1 and 5.2 respectively, the research contribution of this chapter is summarized in section 5.3. Section 5.4 analyzes analog circuit behavior computation methods utilized for improving the simulation performance and accuracy. Section 5.5 describe the operational method for the solution of state space equations and extends it for the computation of MNA circuit equations. Section 5.5.2 presents the novel operational computation method for nonlinear circuits which avoid the linearization of MNA equations. Section 5.7 explains the efficient computational method for discrete time systems that enables iterative simulations. Section 5.8 describes the details of the implemented systemC AMS MoC and section 5.9 show the experimental results of several circuit simulations. Finally, section 5.10 summarizes the contribution and the most relevant findings of the research activities.

5.1 Motivation

The design and validation of analog and mixed-signal systems require a hierarchical approach which uses different modeling abstractions for accurately capturing the system properties depending on the current analysis goals. In order to capture the interaction of embedded systems with their physical environment, the modeling of the temporal dynamics and concurrency properties of the whole system is necessary [71]. Moreover, System Level Design (SLD) tools that support the fast execution of heterogeneous models at several levels of abstraction are needed to cope with the design verification and validation challenges of today's embedded systems.

5.2 Related Work

SystemC AMS provides features for modeling and simulation of heterogeneous and Analog and Mixed-Signal (AMS) systems at several levels of abstraction. In order to maintain an acceptable simulation performance while modeling the system's behavior with enough accuracy, all SystemC AMS MoCs utilize a fixed time step for computation [42]. Ptolemy and SystemC ForSyDe utilize a DAE solver with time step control for fast simulation of dynamical systems [133] [75]. The reported improvement of the simulation performance was not significant compared to SystemC AMS [9].

Zaum et al. presented a simulation approach for mixed-signal circuits called PRAISE that models circuit behavior assuming piecewise constant (PWC) excitations [128] [126] [127]. Using a symbolic mathematical toolbox, the analog circuit equations were transformed into a state space representation at run-time. This abstract circuit modeling enables the fast transient simulation of linear analog systems. A very good simulation performance was achieved compared with the conventional PSspice circuit simulator. A wrapper approach was utilized for interfacing PRAISE with SystemC during simulation. Hoelldampf et al. extended this framework by including the generation of SystemC events from analog circuit modules to allow the interaction between analog and digital components [51] [50] [49] [72].

The circuit behavior computation method based on piecewise constant excitations requires the inversion and diagonalization of the system matrix. As power electronic systems typically present a stiff set of circuit equations, this computation approach leads often to gross errors. Luciano et al. presented a circuit response computation algorithm for the simulation of switching power converters that utilizes Chebyshev polynomials for the faster truncated Taylor series approximation of the state transition matrix [76]. The proposed algorithm avoids the system matrix inversion and interpolates the input signals for the accurate computation of the forced response. Thus, the approximation error in case of rapidly varying input signals is considerably reduced.

The convergence and accuracy of the series approximation methods depends on the matrix norm. In order to avoid gross errors and to improve the accuracy, Tymerski et al. utilized several techniques for reducing the system matrix norm such as scaling and squaring [116].

In order to achieve faster simulations of nonlinear circuits, Palusinski et al. proposed a behavior computation method based on Chebyshev polynomials [92]. They utilized modified nodal analysis (MNA) for the formulation of the set of nonlinear ordinary differential equations describing the circuit behavior and applied the Newton-Kantorovich [7] approach for the linearization of the circuit equations. Due to the large number of required polynomial approximation terms, the achieved simulation speed-up was moderate.

Li et al. investigated the computation method based on Chebyshev series for the simulation of linear and switched circuits [73] [74]. They proposed an error control algorithm that varies the series expansion order while keeping the step size fixed. Thus, the factorization of the circuit equations remains unaltered. They reported a significant improvement of the simulation accuracy compared with the PSpice circuit simulator but the achieved simulation speed-up was also moderate.

Although several behavior computation methodologies were proposed for the fast simulation of analog and power electronic circuits, the accurate modeling and high performance simulation of analog and mixed signal systems at different levels of abstraction remain a challenge for system level design languages and tools. Behavior computation approaches for analog circuits suffer from stability, accuracy and efficiency issues limiting design, validation and verification tasks.

[7] Kantorovich extended the Newton method for solving nonlinear equations to functional spaces

5.3 Contribution to AMS Circuit Simulation

The aim of the research work presented in this chapter was to tackle the accuracy and efficiency problems of analog circuit behavior computation methods at several levels of abstraction as well as to derive a simulation method for analog and mixed signal systems that take advantage from the signal model presented in chapter 4 for system level design and validation purposes. The main research activities carried out to achieve this goal were the following:

- To investigate the limitations of several analog circuits' behavior computation methods utilized for fast simulations.
- To derive more efficient and accurate behavior computation methods for linear and switched circuits.
- To work out more efficient and accurate behavior computation methods for nonlinear circuits.
- To develop a digital circuit computation method that enables the efficient computation of mixed signal systems.
- To analyze the properties of the proposed operational computational methods for analog circuits.
- To compare the accuracy, stability and performance properties of several computation methods.

To cope with the challenges regarding the design and validation of contemporary low-power embedded systems, several computational methods that rely on orthogonal signals are proposed in this chapter. The proposed computation methods for fast simulation of analog circuits are based on the *operational matrix of integration*. A state based modeling and simulation methodology was developed for digital circuits to allow the fast simulation of AMS systems.

As the proposed operational computation methods allow the modeling of analog system components at different levels of abstraction and their efficiency, stability and accuracy properties enable fast system level simulations, they significantly contribute to a seamless design methodology.

5.4 Efficient Computation of Analog Circuits

Most circuit simulators employed in the industry use stepwise, stiffly stable integration methods to solve the set of differential equations describing the circuit dynamic behavior (see section 3.4.2). In order to overcome the trade-off between simulation performance and accuracy introduced by these integration methods, computation methods based on the *state transition matrix* were often proposed. The advantages and limitations of such computation methods are described in section 5.4.1. Furthermore, the advantages and weak points of the Chebyshev series based computation methods proposed in the literature for improving circuit simulation are explained in section 5.4.2.

5.4.1 State Transition Matrix Based Circuit Computation

As explained in section 3.4.1, the set of equations describing the behavior of linear electrical networks can be represented by a minimal set of variables using *state space equations*. This formulation of circuit equations contains a set of ordinary differential equations (ODEs) which are represented in vectorial form as follows:

$$\frac{d}{dt}[x(t)]_N = [A_{SS}]_{N^2} \cdot [x(t)]_N + [B_{SS}]_{NM} \cdot [u(t)]_M \tag{5.1}$$

where $[x(t)]_N$ and $[u(t)]_M$ are the state and input vector variables respectively. Equation (5.1) captures the dynamic properties of the circuit behavior. As the state variables are expressed in explicit form, the exact solution of the ordinary differential equation can be computed as follows:

$$[x(t)]_N = [\Phi(t)]_{N^2} \cdot [x(t_0)]_N + \int_{t_0}^{t} [\Phi(t-\tau)]_{N^2} \cdot [B_{SS}]_{NM} \cdot [u(\tau)]_M \, d\tau \tag{5.2}$$

where

$$[\Phi(t)]_{N^2} = e^{[A_{SS}]_{N^2} \cdot t} \tag{5.3}$$

is known as the *state transition matrix*.

If the matrix $[B]_{NM}$ and the input vector $[u(t)]_M$ are considered to be constant in a given time interval $[t_a, t_b]$ and the matrix $[A_{SS}]_{N^2}$ is invertible, then the forced system response (integral part in Eq. (5.2)) can be expressed in terms of the state transition matrix as follows:

$$\int_{t_a}^{t_b} [\Phi(t_b - \tau)]_{N^2} \cdot [B]_{NM} \cdot [u(\tau)]_M \, d\tau = [\Psi(t_b - t_a)]_{NM} \cdot [u(t)]_M \tag{5.4}$$

where

$$[\Psi(t_b - t_a)]_{NM} = [A_{SS}]_{N^2}^{-1} \cdot ([\Phi(t_b - t_a)]_{N^2} - [I]_{N^2}) \cdot [B_{SS}]_{NM} \tag{5.5}$$

This simplification can be applied for the computation of system responses in many practical cases. The main advantage of the *analytical system response computation* is that the accuracy of the solution does not depend on the step size (if the above mentioned assumptions hold). Thus, large step sizes can be utilized for simulation. Using this computation method, efficient and accurate simulations can be achieved.

The most popular method for the computation of the state transition matrix is its approximation by a truncated Taylor series expansion.

$$[\Phi(t)]_{N^2} \approx \sum_{k=0}^{P} \frac{1}{k!} ([A_{SS}]_{N^2} \cdot t)^k \tag{5.6}$$

The computation of Eq. (5.6) is straightforward. Furthermore, replacing this representation of the matrix exponential in Eq. (5.5) cancels the matrix inversion $[A_{SS}]_{N^2}^{-1}$. This is a useful property because the norm of the matrix $[A_{SS}]_{N^2}$ is often large for electrical circuits (particularly for power electronic circuits) and therefore, the accurate computation of the matrix inversion $[A_{SS}]_{N^2}^{-1}$ becomes difficult (ill-conditioned problem). To attain a desired accuracy, a sufficiently large number of terms P must be summed. Utilizing orthogonal polynomials, the number of required terms for a given accuracy can be reduced as shown in example 4.3. Luciano et al. utilized Chebyshev polynomials for faster computation of the matrix exponential [76].

The error produced by a matrix series approximation depends not only on the number of added terms, but also on the norm of the matrix. The approximation error increases as the norm of the matrix increases.

The application of any series expansion method for the computation of the *matrix exponential* without matrix preconditioning may lead to gross approximation errors [88]. Preconditioning techniques are necessary for the reliable computation of the matrix exponential. They allow the minimization of the matrix norm. A general approach for matrix norm minimization is *matrix balancing* but its effectiveness is quite limited. The most commonly used approach to minimize the norm of a matrix power is known as *scaling and squaring*[8]. It divides a matrix by a non-zero scalar q and then the q-th power is applied to the series expansion. This method assures that the matrix norm is equal or smaller than 1.

As a very good accuracy is achieved, techniques based on the eigenvalues and eigenvectors decomposition (Jordan decomposition) are often used for the matrix exponential computation. In order to avoid the inversion of the matrix $[A_{SS}]^{-1}_{N^2}$ the natural and the forced system response can be computed simultaneously as proposed by Tymerski et al. [116]. The handling of matrices with repeated eigenvalues (confluent matrices) and matrices with an incomplete number of linearly independent eigenvectors (defective matrices) becomes difficult in Jordan decomposition method.

As explained in [88], there is no method that completely satisfactorily computes the matrix exponential. Furthermore, the efficient computation of the matrix exponential integral is only possible if the forcing function $[u(t)]_M$ is constant (or linearly changes) over the integration interval.

Summarizing, the advantages of the state transition matrix circuit behavior computation method are:

1. Large step sizes may be utilized for fast simulations.
2. No error control is necessary for accurate simulations.

and its limitations are:

1. The formulation of state space equations is required.
2. The computation of the state transition matrix is affected by accuracy and stability issues.
3. This method is not efficient in case of varying input signals e.g. sine waveforms.

[8] This method is utilized by MATLAB for matrix power computation

5.4.2 Chebyshev Series Based Circuit Computation

In order to efficiently solve the circuit equations formulated using the modified nodal analysis method (see section 3.4.1), Palusinski et al. proposed the approximation of each signal of the signal vectors $[x(t)]_{N+B}$ (consisting of N node voltages and B branch currents), $\frac{d}{dt}[x(t)]_{N+B}$ and $[w(t)]_{N+B}$ (containing the contribution of voltage and current sources) by a truncated Chebyshev series expansion of P terms as follows [92]:

$$x_i(\sigma) \approx \frac{1}{2} \cdot c'_{x_i 0} \cdot T_0 + \sum_{n=1}^{P-1} c_{x_i n} \cdot T_n(\sigma) = [c_{x_i n}]_P \cdot [T_n(\sigma)]_P^T \tag{5.7a}$$

$$\frac{d}{d\sigma} x_i(\sigma) \approx \frac{1}{2} \cdot c'_{dx_i 0} \cdot T_0 + \sum_{n=1}^{P-1} c_{dx_i n} \cdot T_n(\sigma) = [c_{dx_i n}]_P \cdot [T_n(\sigma)]_P^T \tag{5.7b}$$

$$w_i(\sigma) \approx \frac{1}{2} \cdot c'_{w_i 0} \cdot T_0 + \sum_{n=1}^{P-1} c_{w_i n} \cdot T_n(\sigma) = [c_{w_i n}]_P \cdot [T_n(\sigma)]_P^T \tag{5.7c}$$

were the symbol ′ denotes the coefficient scaling $c'_0 = 2 \cdot c_0$.

As Chebyshev polynomials are orthogonal in the interval [-1, 1], the following variable transformation is necessary:

$$\sigma = \frac{2 \cdot t - t_a - t_b}{t_b - t_a}, \quad t_a \le t \le t_b \tag{5.8}$$

Due to the recurrence properties of the Chebyshev polynomials, the coefficients $c_{x_i n}$ and $c_{dx_i n}$ meet the following relation:

$$c_{x_i n} = \frac{1}{2 \cdot n} \cdot \left(c_{dx_i (n-1)} - c_{dx_i (n+1)} \right), \quad n > 0 \tag{5.9}$$

In order to compute the coefficient $c'_{x_i 0}$, the following approximation based on the initial condition $x_i(t_a)$ was utilized by Palusinski et al. 5.2:

$$c'_{x_i 0} \approx 2 \cdot x_i(t_a) - 2 \cdot \sum_{n=1}^{P-1} (-1)^n \cdot c_{x_i n} \tag{5.10}$$

Using the previous equations, a vectorial algebraic equation system that allows the computation of the Chebyshev coefficients vector $[c_{x_i n}]_P$ for each signal in the signal vector $[x(t)]_{N+B}$ is expressed in the following form:

$$[A_i]_P \cdot [c_{x_i n}]_P = [B_i]_P \cdot [c_{w_i n}]_P + [c_{u_i n}]_P \tag{5.11}$$

where the matrices $[A_i]_P$ and $[B_i]_P$ are computed from the circuit equation matrices $[G_{MNA}]_{N+B}$ and $[C_{MNA}]_{N+B}$.

With the aim of improving the simulation performance, Palusinski computed the Chebyshev coefficients vector of the corresponding input signal $[c_{w_i n}]_P$ using a Fast Chebyshev Transformation (FCT), which is a numerical computation method similar to the Fast Fourier Transformation (FFT) [92].

The Chebyshev coefficient vector $[c_{u_i n}]_P$ represents the contribution of the initial condition $x_i(t_a)$ as well as the contribution of the remaining $N+B-1$ equations to the considered signal $x_i(t)$. Thus, the set of $N+B$ algebraic equation systems (which are described in generic form by Eq. (5.11)) are iteratively solved using numerical algebraic methods (see section 3.4.4) until convergence to the solution is achieved (waveform relaxation approach).

In order to cope with nonlinear circuit equations, the nonlinear contribution to nodal voltages and branch currents is linearized applying the following approximation (derived from the Newton-Kantorovich method):

$$[g([x(t)]_{N+B}, t)]_{N+B} \approx [M]_{(N+B)^2} \cdot \left([x(t)]_{N+B} - [x_P(t)]_{N+B}\right) + [g([x_P(t)]_{N+B}, t)]_{N+B} \tag{5.12a}$$

$$[M]_{(N+B)^2} = \left.\frac{\partial g}{\partial x}\right|_{[x(t)]_{N+B}, t} \tag{5.12b}$$

where $x_P(t)$ is an initial solution or the result of the previous computation. The matrix $[M]_{N+B}$ is known as *Jacobian matrix*. Note that the presence of the argument t in Eq. (5.12b) indicates the dependence on external voltage and current sources.

After linearization using Eq. (5.12), the matrices $[A_i]_P$ and $[B_i]_P$ are determined for the iterative computation of the circuit behavior using Eq. (5.11) as explained above.

Using the Chebyshev series expansions for solving circuit equations provides several advantages:

1. Circuit equations can be formulated as DAEs or ODEs (the formulation of state space equations is not necessarily required).
2. The sparsity of the system matrices is preserved after the series expansion of voltage and current signals.
3. As the Chebyshev series usually converge rapidly (in particular for smooth signals), only few coefficients are necessary for accurate signal approximations.
4. As the Chebyshev polynomials can be very efficiently computed (recursively), the evaluation of signals at any time point can be carried out very quickly.
5. The Chebyshev series based computation method is efficient and accurate in case of varying input signals e.g. sine waveforms.
6. As the Chebyshev polynomials are bound by ±1, the coefficients of the series expansion are very good indicators of the truncation error, enabling an efficient error control implementation.
7. The Chebyshev series based computation method is well suited for the application of the waveform relaxation approach (parallel computation).

The following weak points of this method need to be considered:

1. The matrix construction procedure for representing the circuit equations in linear form is quite complex and not general (see [92]).
2. Computing the circuit behavior over a long time interval requires a large number of Chebyshev series expansion coefficients for accurate results (see section 5.2).
3. The proposed method for nonlinear equations (Eq. 5.12) requires a very good initial guess for achieving convergence to the solution.
4. The system of linear algebraic equations becomes large (proportional to the number of Chebyshev coefficients) for the direct solution of linear circuit equations.

Numerical analysis carried out in this work confirms that the Chebyshev series method proposed by Palusinski et al. requires a large number of expansion coefficients. An explanation for the poor convergence of this method (even for linear equations) concerns the recurrent formula defined by Eq. (5.9). It may be ill-conditioned, amplifying errors in the smallest coefficients in such way that even the accuracy on the largest coefficients becomes poor [96]. As Eq. (5.10) distributes this errors across all coefficients, it does not seem to be a good approximation for the computation of the initial condition.

5.5 Operational Computation of Analog Circuits

With the aim of take advantage from the orthogonal signal model proposed in chapter 4 (see section 4.5), this section proposes operational computation methods that enable the behavior computation of analog circuits at different abstraction levels (from linear transfer functions to nonlinear circuit equations). They overcome the weak points of the computational method described in section 5.4.2. The analog signal expansions utilized for analog circuit behavior computation are not restricted to be Chebyshev series. Furthermore, the differential equations describing the circuit dynamical behavior are converted to integral equations and then transformed into an algebraic equation system using an *operational matrix of integration* for improving both the computation accuracy and performance. Moreover, the matrix construction procedure is defined in terms of the outer product leading to a direct and simple computation from circuit equation matrices.

5.5.1 Operational Computation of Linear Circuits

Any linear circuit can be described by a state space equation. Fig. 5.1 shows the signal flow model of the corresponding continuous time linear system.

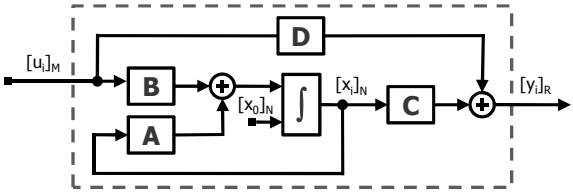


Fig. 5.1: Analog system representation in state space form

In order to apply the operational computation method to the state space equation formulation, the system state and signal vectors are represented by orthogonal signals expressed in matrix-vector multiplication form as follows:

$$[x(\sigma)]_N \approx [c_x]_{PN}^T \cdot [Q_n(\sigma)]_P = [c_x]_{NP} \cdot [Q_n(\sigma)]_P \tag{5.13a}$$

$$[u(\sigma)]_M \approx [c_u]_{PM}^T \cdot [Q_n(\sigma)]_P = [c_u]_{MP} \cdot [Q_n(\sigma)]_P \tag{5.13b}$$

$$[y(\sigma)]_R \approx [c_y]_{PR}^T \cdot [Q_n(\sigma)]_P = [c_y]_{RP} \cdot [Q_n(\sigma)]_P \tag{5.13c}$$

where P is the number of signal coefficients.

In order to simplify the equations, the orthogonal signal model defined by Eq. (4.18) is reduced to the continuous signal form (Eq. (4.12)). Using orthogonal polynomials $Q_n(\sigma)$, the analog signals of any continuous time system are efficiently and accurately described by a reduced number of coefficients.

The integration of a continuous time signal $s(\sigma)$ that is defined by an orthogonal signal expansion in terms of the coefficient vector $[c_s]_P$ can be approximated by the following matrix multiplication:

$$\int_{\sigma_a}^{\sigma_b} s(\sigma)\, d\sigma \approx [c_s]_P^T \cdot [P_Q]_{P^2} \cdot [Q_n(\sigma_b)]_P \tag{5.14}$$

The matrix $[P_Q]_{P^2}$ is known as *operational matrix of integration* and its coefficient values depend on the utilized orthogonal functions. For example, for 3^{rd} order Chebyshev polynomials $T_n(\sigma)$, the corresponding operational matrix of integration $[P_T]_{P^2}$ is given by [94]:

$$[P_T]_{4^2} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ -\frac{1}{4} & 0 & -\frac{1}{4} & 0 \\ -\frac{1}{3} & -\frac{1}{2} & 0 & \frac{1}{6} \\ \frac{1}{8} & 0 & -\frac{1}{4} & 0 \end{bmatrix} \tag{5.15}$$

Legendre polynomials $P_n(\sigma)$ are also well suited for representing and integrating analog signals [93]. In both cases, it is necessary to map the signals to an interval $[\sigma_a, \sigma_b]$ in which the polynomials are orthogonal. This is carried out by a simple variable change (see Eq. (5.8)).

Although Eq. (5.14) is valid for any orthogonal function, Chebyshev and Legendre polynomial approximations of analog signals have certain advantages over others orthogonal functions (e.g. Laguerre or Walsh):

1. The approximation error is almost uniform in the integration interval (due to the least squares approximation properties).

2. Independently of the size of $[P_Q]_{P^2}$, only one non-zero term is truncated (introducing less error than other operational matrices).
3. The value of the truncated term decreases proportional to the size of $[P_Q]_{P^2}$ (few coefficients are needed for a small approximation error).

In order to compute the system behavior, both sides of Eq. (5.1) are integrated. Thus the explicit ordinary differential equation (ODE) is transformed into the following integral equation:

$$[x(t)]_N = \int_{t_0}^{t} \Big([A_{SS}]_{N^2} \cdot [x(\tau)]_N + [B_{SS}]_{NM} \cdot [u(\tau)]_M \Big) d\tau + [x(t_0)]_N \quad (5.16)$$

Replacing the continuous time signals in Eq. (5.16) by the proposed orthogonal signal expansions (Eq. (5.13)) and utilizing Eq. (5.14) for the computation of the signal integrations, the following algebraic equation is obtained:

$$[c_x]_{NP} - [A_{SS\sigma}]_{N^2} \cdot [c_x]_{NP} \cdot [P_Q]_{P^2} = [B_{SS\sigma}]_{NM} \cdot [c_u]_{MP} \cdot [P_Q]_{P^2} + [c_{x_0}]_{NP} \quad (5.17)$$

Note that the orthogonal polynomials vector $[Q_n(\sigma)]_P$ was eliminated in Eq. (5.17). Therefore only signal coefficients are involved in the solution of the differential equation. Due to the variable mapping (from t to σ), the system matrices are modified by a scaling factor (which is denoted by the subscript σ).

Using the Kronecker product properties, the following linear equation system is obtained:

$$[A]_{(NP)^2} \cdot vec([c_x]_{NP}) = vec([c_b]_{NP}) \quad (5.18a)$$

$$[A]_{(NP)^2} = ([I]_{(NP)^2} - [A_{SS\sigma}]_{N^2} \otimes [P_Q]_{P^2}^T) \quad (5.18b)$$

$$[c_b]_{NP} = [B_{SS\sigma}]_{NM} \cdot [c_u]_{MP} \cdot [P_Q]_{P^2} + [c_{x_0}]_{NP} \quad (5.18c)$$

where the matrix $[I]_{(NP)^2}$ is the identity matrix and $vec()$ is a matrix vectorization operator. Note that the size of the previous equation system is proportional to the number P of coefficients. It can be solved using direct computation methods[9]. This work utilizes LU factorization followed by forward and backward substitution (see section 3.4.4).

After the solution of Eq. (5.18), the computation of system response is carried out by evaluating the following algebraic equation at the

[9] For large systems or large number of required signal coefficients, an iterative computation method may provide a better accuracy or performance.

desired time point σ:

$$[y(\sigma)]_R = ([C_{SS\sigma}]_{RN} \cdot [c_x]_{NP} + [D_{SS\sigma}]_{RM} \cdot [c_u]_{MP}) \cdot [Q_n(\sigma)]_P \tag{5.19}$$

In this work, only time invariant systems were considered (i.e. $[A_{SS}]_{N^2}$ and $[B_{SS}]_{N^2}$ are assumed to be constant) but the presented methodology can be extended for time varying systems (see [92]). In such case, the system matrices need to be expressed in terms of orthogonal signals.

For electrical circuits, the circuit equations are usually expressed in terms of a differential algebraic equation (DAE). The procedure carried out above for state space equations can also be applied for the behavior computation of circuit equations formulated using the modified nodal analysis method (see section 3.4.1). After integrating both sides of Eq. (3.1), the electrical circuit is described by the following integral equation:

$$[C_{MNA}]_{(N+B)^2} \cdot [x(t)]_{N+B} = \int_{t_0}^{t} \Big([w(\tau)]_{N+B} - [G_{MNA}]_{(N+B)^2} \cdot [x(\tau)]_{N+B} \Big) d\tau + [x(t_0)]_{N+B} \tag{5.20}$$

where $[w(t)]_{N+B}$ denotes the input signal vector and the signal vector $[x(t)]_{N+B}$ contains the state and output signal vectors. As the matrix $[C_{MNA}]_{(N+B)^2}$ is normally not invertible, the formulation in the form of Eq. (5.16) is not possible. Replacing the continuous time signals in Eq. (5.20) by orthogonal signal expansions, the following algebraic equation is obtained:

$$[G_{MNA\sigma}]_{(N+B)^2} \cdot [c_x]_{(N+B)P} \cdot [P_Q]_{P^2} + [C_{MNA\sigma}]_{(N+B)^2} \cdot [c_x]_{(N+B)P} = [c_b]_{(N+B)P} \tag{5.21a}$$

$$[c_b]_{(N+B)P} := [c_w]_{(N+B)P} \cdot [P_Q]_{P^2} + [c_{x_0}]_{(N+B)P} \tag{5.21b}$$

Using the Kronecker product properties, the following linear equation system is obtained (see [37]):

$$[A]_{((N+B)P)^2} \cdot vec([c_x]_{(N+B)P}) = vec([c_b]_{(N+B)P}) \tag{5.22a}$$

$$[A]_{((N+B)P)^2} := ([I]_{P^2} \otimes [C_{MNA\sigma}]_{(N+B)^2} + [G_{MNA\sigma}]_{(N+B)^2} \otimes [P_Q]_{P^2}^T) \tag{5.22b}$$

$$[c_b]_{(N+B)P} := [c_w]_{(N+B)P} \cdot [P_Q]_{P^2} + [c_{x_0}]_{(N+B)P} \tag{5.22c}$$

After solving Eq. (5.22), the behavior of the system is computed evaluating the orthogonal signals at the desired time point σ.

$$[x(\sigma)]_{N+B} = [c_x]_{(N+B)P} \cdot [Q_n(\sigma)]_P \tag{5.23}$$

5.5.2 Operational Computation of Non-Linear Circuits

Nonlinear circuits are modeled representing the nonlinear circuit elements as externally controlled current or voltage sources. Eq. (5.24) shows the *MNA formulation for nonlinear circuits*.

$$[G_{MNA}]_{(N+B)^2} \cdot [x(t)]_{N+B} + [C_{MNA}]_{(N+B)^2} \cdot \frac{d}{dt}[x(t)]_{N+B} = [w([x(t)]_{N+B}, t)]_{N+B} \tag{5.24}$$

After formulating Eq. (5.24) in terms of orthogonal signals expansions, the following implicit equation is defined for the computation of the circuit behavior:

$$f([c_x]_{(N+B)P}) = [C_{MNA\sigma}]_{(N+B)^2} \cdot [c_x]_{(N+B)P} - h([c_x]_{(N+B)P}) \cdot [P_Q]_{P^2} = 0 \tag{5.25a}$$

$$h([c_x]_{(N+B)P}) := [w([c_x]_{(N+B)P})]_{N+B} - [G_{MNA\sigma}]_{(N+B)^2} \cdot [c_x]_{(N+B)P} \tag{5.25b}$$

The coefficients $[c_x]_{(N+B)P}$ of the system response can be computed using the *Newton-Raphson iteration*:

$$[c_x]^{k+1}_{(N+B)P} = [c_x]^{k}_{(N+B)P} - [M]^{-1}_{((N+B)P)^2} \cdot f([c_x]^{k}_{(N+B)P}) \tag{5.26}$$

However, the computation of the *Jacobian matrix* $[M]_{((N+B)P)^2}$ is quite difficult for the *vector function* $f([c_x]_{(N+B)P})$. Furthermore, the convergence of Eq. (5.26) strongly depends on the initial guess and on the nonlinear circuit characteristics.

In order to simplify the computations and to achieve more reliable results, this work proposes the computation of nonlinear equations based on the *Picard iteration* which is defined as follows:

$$[x(t)^{k+1}]_{N+B} = [x(t_0)]_{N+B} + \int_{t_0}^{t} \left(\frac{d}{d\tau}[x(\tau)^k]_{N+B} \right) d\tau \tag{5.27}$$

Eq. (5.27) avoids the linearization of the system equations. The behavior of nonlinear circuits is computed iteratively by solving the following algebraic equation system (which results from combining Eq. (5.25) and Eq. (5.27)):

$$[C_{MNA\sigma}]_{N+B} \cdot [c_x]^{k+1}_{(N+B)P} = ([c_{x_0}]_{(N+B)P} + h([c_x]^{k}_{(N+B)P}) \cdot [P_Q]_{P^2} \tag{5.28}$$

It shows very good convergence properties for Chebyshev polynomials as will be shown in section 5.9.

5.6 Sequential Computation of Digital Circuits

A digital circuit with memory is usually modeled as a *finite state machine*. The block diagram shown in Fig. 5.2 represents the signal flow model of a sequential digital circuit. The *next-state function* f_q and the output function f_o are combinational networks (switching functions). The *delay function* f_d implements a feedback delay τ_d.

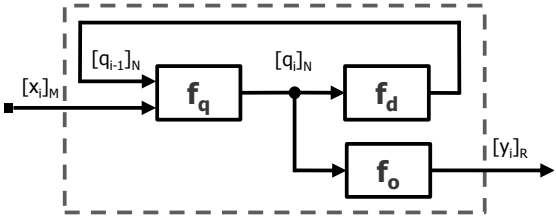


Fig. 5.2: Digital system representation as finite state machine

This abstract representation of a digital circuit is the counterpart of the state space model used for the abstract description of analog circuits. It is considered in this work for computing the digital circuit behavior. The *digital state* $[q_i]_N$ needs to be explicitly modeled for the correct computation of the circuit behavior. It is necessary for coping with discrete event iterations (delta cycles). The input signal $[x_i]_M$ and the output signal $[y_i]_R$ are sequences of values. Note that discretized analog systems correspond to this circuit representation. The value sequences $[x_i]_M$ and $[y_i]_R$ are represented by orthogonal signals to enable the fast digital circuit simulation.

Digital logic circuits are usually modeled based on the axioms of boolean algebra as binary switching functions. The computation of digital logic circuits at gate level [10] can be expressed in transfer function form by expanding the set of input signals into a binary orthogonal basis. This dissertation focuses on the digital circuit modeling at register-transfer level (RTL) as will be shown in the experimental results.

[10] For gate level computation of digital logic circuits see [115]

5.7 Iterative Data Flow Computation of AMS Circuits

Digital and analog circuits are executed in this dissertation using a *synchronous data flow formalism*. The behavior of the several system components is encapsulated into one or more processes. Each *process* reads and/or writes signals represented in vectorial form which include timing information. Before simulation starts, the execution properties (timing and rates) of each computation process must be defined. This property setting takes place during *model initialization*. When it is done, the execution order of the discrete time processes is determined.

After property setting and schedule computation, the simulation starts. In each *evaluation cycle*, the *next execution time* is computed by a *cluster manager* which sequentially executes the model processes. Each process may request the next execution time. When a process is executed, it reads all input signals from the ports and determines the next *local event time*. After updating the *local process time*, the *event processing function* is executed. This step is repeated until all input signal events are read. All signals are defined in the same time interval (according to the *execution time*) but the number of events present on signals is arbitrary (not limited to a constant rate). Thereby, the *event processing function* may be executed several times during process execution. If a process does not have input ports, the *event processing function* is executed once per cycle. The resulting events may be written to the output signals with any arbitrary time delay value.

For systems containing feedback loops, the model evaluation cycle is repeated until signal convergence is achieved. As digital circuits normally contain sequential processes (see finite state machine representation in section 5.6), the *logic state* $[q_{i-1}]_N$ of digital processes is stored in a special process variable. Thus, the initial circuit state can be restored after an evaluation cycle was carried out (if it is necessary).

For analog circuits, the *continuous state* is stored in the analog solver. If an evaluation cycle is repeated to achieve signal convergence, the state value is restored (similar to the computation of digital circuits).

Note that the global model execution is carried out synchronously i.e. the signals of all computation processes share the same start and end time but the events occurring in this time interval are computed asynchronously i.e. digital processes have their own discrete event simulation mechanism and analog processes have their own analog solver and computation settings.

5.8 Implementation

In order to evaluate the accuracy, efficiency and stability of numerical computation methods for analog circuits, a MATLAB toolbox which supports the modeling of analog circuits (described by a net-list or in state space form) was implemented in this work. The analog circuit simulation toolbox contains classes for the formulation of circuit equations (MNA method) and several numerical solvers for time domain circuit response analysis. A key feature of this toolbox is the symbolic solution of circuit equations.

Architecture modeling features (modules, ports and signals) are provided for the simulation of data flow models. Listing 5.1 shows the MATLAB test bench model utilized for circuit simulation in section 5.9. The average Buck converter circuit is connected to a source through a signal and the integration method is defined in the model parameters.

Listing 5.1: Circuit test bench model in MATLAB

```
function model = mdl_converter_buck_avg(par)
   % Create an empty model
   model = core.model('mdl_converter_buck_avg', par);
   % Define model signals
   sig_src = core.signal('sig_src');
   sig_circ = core.signal('sig_circ');
   % Create modules
   src = models.mod_src_cnst('src',par.source);
   circ = models.circ_converter_buck_avg('circ',par);
   % Bind signals
   src.out.bind_signal(sig_src);
   circ.in.bind_signal(sig_src);
   circ.out.bind_signal(sig_circ);
   % Add modules to the model
   model.add_module(src);
   model.add_module(circ);
end
```

A SystemC AMS extension, that defines user analog modules using templates was also implemented. It allows the integration and performance evaluation of computation methods for analog system simulation. As the numerical computation routines implemented in the SystemC extension are limited (compared to those provided by MATLAB), only a small set of analog solvers are currently available. Listing 5.2 shows the SystemC code for the circuit test bench model.

Listing 5.2: Circuit test bench model in SystemC AMS

```
template<typename data_type>
class sca_converter_buck_avg:
public ::sca_usr::sca_analog_system_module<data_type,
   typename sca_type<data_type>::equation_type,
   typename sca_type<data_type>::solver_type>
{  // Define solver and data type for analog module
   typedef ::sca_usr::sca_analog_module<data_type,
   typename sca_type<data_type>::equation_type,
   typename sca_type<data_type>::solver_type>
   base_type;
public:
 sca_converter_buck_avg(::sc_core::sc_module_name _name):
 base_type(_name, _sample_time){}
};
```

MoC implementation for AMS circuit simulation

In order to keep the process modeling and communication abstract and efficient, a model of computation called *orthogonal signal flow (OSF)* was implemented as extension of the timed data flow (TDF) MoC provided by SystemC AMS. Due to the calculation of a static schedule during the model initialization, this synchronous computation MoC enables a good simulation performance.

All main components are defined in the namespace **`sca_osf`**. Module, port and signal classes are derived from the respective TDF classes, supporting thus hierarchical model construction. Input and output port converters are provided for the interaction with SystemC and SystemC AMS model parts. The dynamic features of the TDF MoC allow the time-accurate interaction with discrete event parts. Port functions **`read`**, **`initialize`** and **`write`** are overloaded to provide signal read and writing operations at the *current process time* for a single value or an array of values (orthogonal signal coefficients). An additional class for modeling of *logic states* (which enables digital solver rollback) was implemented.

Fig. 5.3 shows the processing function implementation for OSF modules. The *current process time* can be advanced using the OSF function **`set_event_time`** for synchronous read and write operations on module ports. The process time advance is limited by the *cluster next time step*.

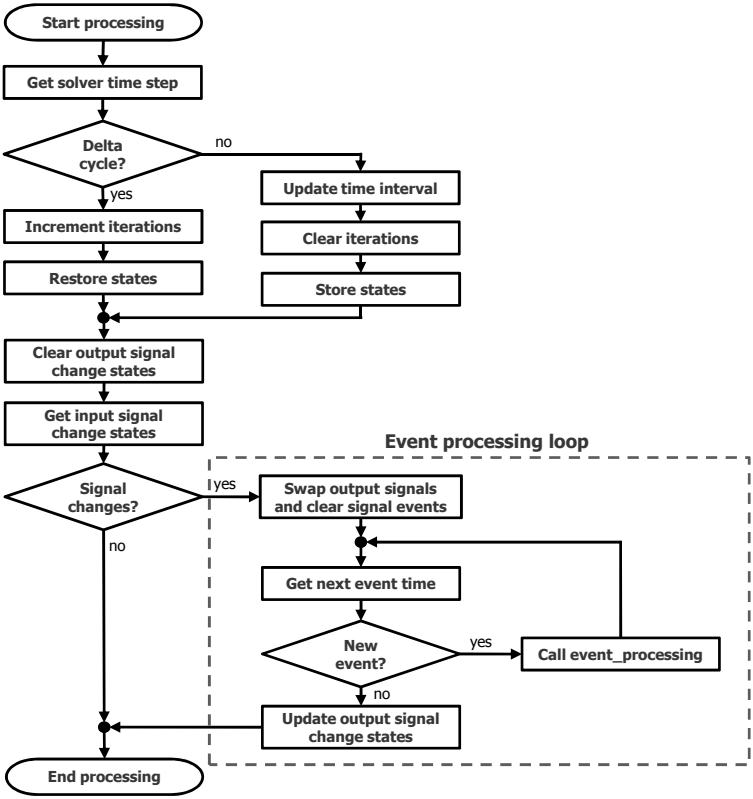


Fig. 5.3: OSF module processing function

The OSF functions **`initialize_event`** and **`write_event`** allow event writing on port signals at arbitrary times, supporting therefore asynchronous signals. During process execution, for each event that is present in the input signals of a module the *current process time* is updated to the next input signal event time and then, the *virtual function* **`event_processing`** is automatically called. Event times are defined relative to the signal start point. The start and the end time points of module output signals can be obtained with the OSF functions **`get_initial_time`** and **`get_final_time`** respectively.

The delay properties of an OSF process are specified in the module output ports using the functions **set_delay** for a deadlock-free TDF model schedule and **set_event_delay** for an arbitrary signal delay time. The simulation accuracy is controlled module-wise using the port functions **set_time_accuracy** and **set_signal_accuracy**. They allow the definition of the maximal admissible time and value deviation for discrete and continuous time signals respectively. The default attribute values for signal accuracy control are 1 *ns* and 1×10^{-6}.

Listing 5.3 shows the implementation of a RS flip-flop with the OSF MoC. It is utilized in the digital phase detector part of the PLL circuit which is proposed in section 5.9 for concept validation. Note that the flip-flop state variables are instances of the OSF class **sca_state**. Thus, the flip-flop state is restored if a delta cycle takes place during simulation.

Listing 5.3: RS Flip-flop implementation using OSF MoC

```
1  class sca_ff_rs: public sca_osf_tdf_module
2  { public:
3   ::sca_osf::sca_tdf::sca_in<bool>  r, s;
4   ::sca_osf::sca_tdf::sca_out<bool> q, qb;
5   ::sca_osf::sca_state<bool> qin, qbin;
6
7   sca_ff_rs(sc_core::sc_module_name _name):
8   sca_osf_tdf_module(_name),
9   r("r"), s("s"), q("q"), qb("qb"){}
10
11  void set_attributes(void) {
12      q.set_timestep(1,SC_NS);  q.set_delay(1);
13      qb.set_timestep(1,SC_NS); qb.set_delay(1);
14      q.set_event_delay(1.0e-9);
15      qb.set_event_delay(1.0e-9); }
16
17  void initialize(void)
18  { q.initialize(false); qb.initialize(false); }
19
20  void event_processing(void) {
21      if((s.read()==1)&&(r.read()==0))
22          { qin  = 1; qbin = 0; }
23      if((s.read()==0)&&(r.read()==1))
24          { qin  = 0; qbin = 1; }
25      if((s.read()==1)&&(r.read()==1))
26          { qin  = 0; qbin = 0; }
27      q.write(qin); qb.write(qbin);}
28 };
```

The analog solver implementing the operational computation methods is embedded into an OSF module for circuit simulation. The corresponding module initialization function carries out the solver setup. During cycle evaluation, the module **`event_processing`** function updates the solver for the computation of the circuit response. Electrical networks are modeling using the EPN MoC presented in chapter 3.

5.9 Experimental Results

As starting point for the experiments, the analog circuit shown in Fig. 5.4 was represented in state space form and simulated using the MATLAB toolbox described in section 5.8. It corresponds to the average model of the Buck converter circuit analyzed in chapter 3. Tab. 5.1 shows the accuracy and performance results obtained using the traditional linear multi-step integration methods (see section 3.4.2), the integration methods based on the state transition matrix (see section 5.4.1) and the proposed operational computation methods (see section 5.5.1).

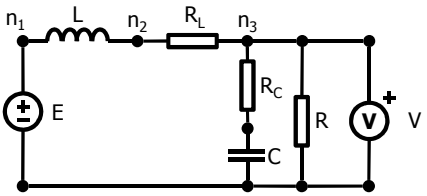


Fig. 5.4: Average Buck converter circuit

The *symbolic solver* (denoted by SYM) is utilized as reference for the accuracy analysis. It evaluates the pre-computed analytical solution at several time points and presents therefore a short execution time. For determining the analytical solution the symbolic solver (based on the MATLAB symbolic toolbox) needs a very long time (around 2 min. for a 3^{rd} order differential equation). As the computation time increases significantly with the number of equations and often no analytical solution can be found, it is not appropriate for practical applications.

Solver	Mean absolute error (MAE)	Mean relative error (MRE)	Execution Time in ms
SYM	0.0	0.0	94.142
LMS (BE)	$1.4561\,e^{-2}$	$2.2772\,e^{-3}$	320.740
LMS (BDF2)	$1.1425\,e^{-4}$	$8.8300\,e^{-5}$	328.070
LMS (TR)	$4,6400\,e^{-5}$	$5.7600\,e^{-5}$	338.360
STM (T)	$2.5655\,e^{-3}$	$4.0051\,e^{-4}$	79.145
STM (TB)	$2.5655\,e^{-3}$	$4.0051\,e^{-4}$	78.283
STM (TBS)	$2.8300\,e^{-11}$	$5.8400\,e^{-12}$	78.172
STM (JD)	$1.6400\,e^{-13}$	$3.2900\,e^{-13}$	77.598
COM (25)	$3.4800\,e^{-7}$	$7.9900\,e^{-8}$	77.390
COM (29)	$1.0500\,e^{-9}$	$2.4000\,e^{-10}$	77.611
COM (35)	$\mathbf{1.8900\,e^{-13}}$	$\mathbf{3.3000\,e^{-13}}$	**78.833**
LOM (25)	$2.8500\,e^{-7}$	$1.0700\,e^{-7}$	81.262
LOM (29)	$8.8200\,e^{-10}$	$3.5200\,e^{-10}$	81.294
LOM (35)	$\mathbf{1.8100\,e^{-13}}$	$\mathbf{3.3700\,e^{-13}}$	**83.121**

Table 5.1: Linear circuit accuracy and performance results

The linear multi-step (LMS) solvers, *Backward Euler* (BE), 2 step *Backward Differentiation Formula* (BDF2) and *Trapezoidal Rule* (TR) (normally used for circuit simulation) require a short time step size (10 times smaller than the signal sampling step) for accurate computations. This leads to large simulation execution times. The 2 step integration methods present a better accuracy than the simple *Backward Euler*, while the impact of the additional step on the execution time is quite small.

The solvers based on the *State Transition Matrix* (denoted by STM) show a very short execution time, but the computational accuracy of this integration method is not better than the accuracy of the 2 steps methods if a *Taylor expansion* (denoted by *T*) without pre-conditioning of the system matrix is utilized for the computation of the matrix exponential. The system matrix norm reduction using *matrix balancing* (denoted by *TB*) does not really improve the accuracy of the Taylor series expansion.

It reduces however the number of required terms and thereby the execution time. *Balancing and scaling* need to be additionally applied to the system matrix for achieving a significant accuracy improvement of the Taylor series expansion (denoted by *TBS*). As the number of needed expansion terms is further reduced, the execution time is also improved (but not significantly). Computing the matrix exponential using the *Jordan Decomposition* (*JD*) method allows an improvement of the solver accuracy and a further reduction of the simulation time compared to the Taylor series expansion method.

The *Chebyshev Operational Matrix* and *Legendre Operational Matrix* computation methods (denoted by COM and LOM respectively) show a short execution time compared with the traditional multi-step methods and their accuracy can be easily controlled by increasing the number of polynomial terms (denoted in brackets). Opposite to the LMS methods, there is no significant trade-off between accuracy and simulation performance. The small impact of the number of coefficients on the execution time indicates that it is dominated by the evaluation of the orthogonal polynomials at the sampling time points. Polynomial series lager than 35 terms do not further reduce the computation error (for the given circuit). This saturation of the accuracy improvement put in evidence that the overall computation accuracy of the operational methods is limited by the accuracy of the linear algebraic solver. This observation is confirmed by the similar accuracy results achieved with the Jordan decomposition method. In this case, the maximal reachable accuracy is limited by the implicit matrix inversion which is also computed using the same linear algebraic solver.

In order to validate the operational computation method for nonlinear systems presented in Sec. 5.5.2, the circuit shown in Fig. 5.5 was modeled and simulated. The resulting diode voltage is shown in Fig. 5.6.

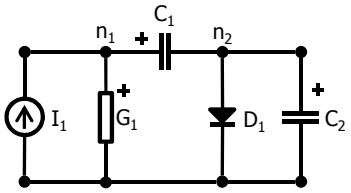


Fig. 5.5: Nonlinear circuit

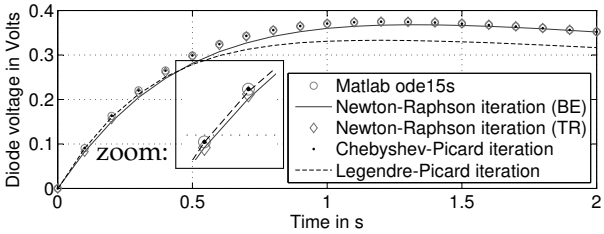


Fig. 5.6: Nonlinear Circuit Response

Furthermore, the implementation of the LMS methods was extended for the computation of nonlinear circuits using the Newton-Raphson algorithm. As the MATLAB symbolic toolbox it is not able to compute an analytical solution for this nonlinear circuit, a numerical solution was obtained utilizing the *MATLAB ode15s function* (denoted by ML) for validation and accuracy analysis purposes. The computation results are shown in Tab. 5.2.

As the *Jacobian matrix* is analytically computed, the simple *Backward Euler* (BE) method shows a good performance. The *Trapezoidal Rule* (TR) method improves the accuracy of the Newton-Raphson iteration but the simulation performance significantly decreases. The presented operational computation method based on the Picard iteration is both, more accurate and faster for Chebyshev polynomials. The convergence properties of Legendre polynomials are poor and they are not suitable for the Picard iteration. It is less accurate than the LMS methods.

Solver	**Mean absolute error (MAE)**	**Mean relative error (MRE)**	**Execution Time in ms**
ML	0.0	0.0	51.346
LMS (BE)	$3.1734\,e^{-3}$	$1.2846\,e^{-2}$	43.374
LMS (TR)	$3.7102\,e^{-4}$	$2.0975\,e^{-3}$	84.759
COM (15)	$\mathbf{9.3460\,e^{-5}}$	$\mathbf{3.9670\,e^{-4}}$	**14.185**
LOM (15)	$2,0555\,e^{-2}$	$7.9829\,e^{-2}$	29.597

Table 5.2: Nonlinear circuit accuracy and performance results

The *analytical method* based on the *state transition matrix* can not be directly applied to nonlinear systems. After linearization this method becomes computationally more expensive than the linear multi-step methods. For this reason, it is rarely utilized in practice and its implementation and analysis was omitted in this work.

In order to evaluate the suitability of the OSF MoC to cope with heterogeneous designs at different abstraction levels, a PLL system (similar to the SystemC AMS PLL model presented in [123]) was modeled. As shown in Fig. 5.7, the PLL consists of a reference signal generator (REF), a phase-frequency detector (PFD), a charge pump (CP), a low-pass filter (LPF), a voltage controlled oscillator (VCO) and a pre-scaler (PSC). The charge pump and the analog filter were modeled at electrical level. The VCO was modeled at functional as well as at architecture level (signal flow). The presented operational methods were utilized for the linear and nonlinear behavior computation. The digital parts (PFD and PSC) were modeled at register transfer level applying the computational model described in section 5.7. The interaction between the analog and digital parts was computed using the method for fast threshold crossing event detection presented in section 4.7.

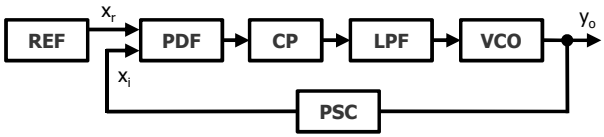


Fig. 5.7: Block diagram of the PLL system

Fig. 5.8 shows the PLL system simulation results. After a transient of 250 μs, the desired output frequency is reached. In order to evaluate the simulation performance achieved by the orthogonal signal based computational methods (OSF MoC), the PLL system was also modeled using the using the SystemC AMS MoCs ELN and LSF as well as TDF for the for the analog and digital parts respectively. Discrete event modeling (DE MoC) was also utilized for digital system modeling. Table 5.3 shows the execution time required by the several modeling abstractions for the PLL system simulation. As expected, the calculation of a static schedule leads to an improvement of the simulation performance for synchronous computational models (TDF and OSF).

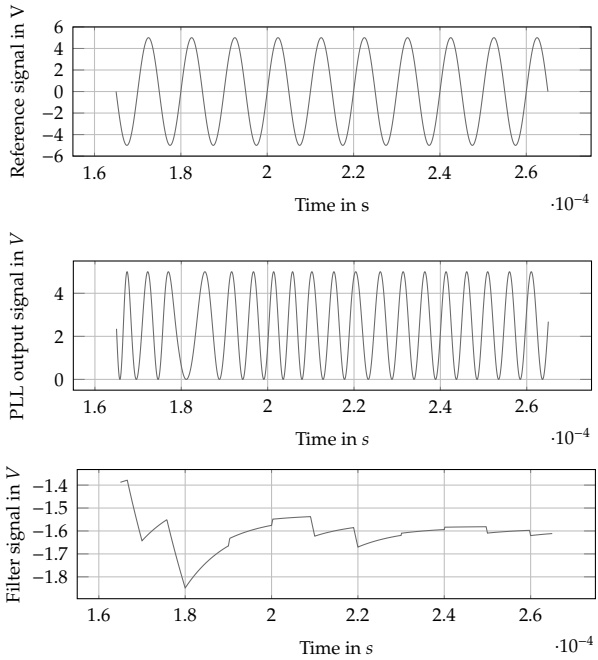


Fig. 5.8: PLL simulation results

The OSF MoC achieves a considerable reduction of the execution time (2x speed-up). In particular, the signal tracing for analysis and validation purposes was significant improved (10x speed-up). As only few coefficients are necessary for representing continuous signals over a large time interval, the orthogonal signal representation enables a very efficient signal tracing.

Signal tracing	DE MoC	TDF MoC	**OSF MoC**
off	1023.0	801.4	**502.1**
on	22819.8	21728.2	**2048.5**

Table 5.3: PLL simulation execution time in *ms*

5.10 Chapter Summary and Conclusions

This chapter presented several behavior computation methods for analog systems that enable faster or more accurate simulations at system and circuit level. The investigation activities have shown that:

- Efficient, accurate and stable integration methods demand sophisticated algebraic methods.
- The proposed operational behavior computation methods only need few signal coefficients for achieving accurate simulation results.
- The proposed nonlinear circuit behavior computation method based on the Picard iteration presents very good convergence properties for orthogonal signals in term of Chebyshev polynomials.
- The convergence properties of Legendre polynomials are not good enough for carrying out nonlinear circuit behavior computation.

The experimental results have shown that the proposed operational computation methods for linear analog circuits are significantly more efficient and accurate than the traditional LMS integration methods and constitute a valuable alternative to the analytical behavior computation methods based on the state transition matrix. As these methods do not impose restrictions on the formulation of circuit equations and do not require the linearization of circuit equations to cope with nonlinear circuits, they are well suited for capturing the behavior of analog circuits at several abstraction levels, enabling a system modeling that is closer to Physics. Relevant system properties such as power consumption can be accurately captured for system design optimization. The operational computation methods also allow the efficient and accurate handling of the interaction between analog and digital circuits. On the other hand, the proposed sequential method for the behavior computation of digital circuits leads to a more efficient process communication.

The experimental results corroborated that the orthogonal signal modeling enables a more efficient computation of AMS systems. A significant reduction of the execution time was obtained with the OSF MoC, in particular if analog signals were traced, which reinforce the importance of this chapter's contribution. In chapter 7, the suitability of the operational computation methods to cope with uncertain analog systems is analyzed and several operational methods for the fast analysis and verification of analog circuits are presented.

Chapter 6
Robust AMS System Design Optimization

Robust embedded system design is becoming more and more important for coping with manufacturing tolerances and external perturbations. There is growing demand for simple and efficient robust design methods that can be easily incorporated into the system design workflow as well as for robustness evaluation methods that support design space exploration leading to robust solutions. This chapter presents a control system design optimization method that does not impose restrictions on the controller structure and enables the improvement of the control system robustness with respect to disturbances and manufacturing tolerances. The proposed AMS control system design method models plant parameter tolerances as structured uncertainties for computing an optimized disturbance weighting function and finds a robust design by tuning the controller parameters to optimize the corresponding mixed sensitivity problem. In order to reduce the design optimization time, internal model control (IMC) is utilized for obtaining a near optimal set of initial controller parameters. Furthermore, a mixed sensitivity index is defined that enables the simple robustness evaluation of both linear and nonlinear control systems. It allows the accurate comparison of several control designs as well as the robust fine tuning of controller parameters.

This chapter is organized as follows. After the presentation of the research motivation and related work in section 6.1 and 6.2 respectively, the research contribution of this chapter is summarized in section 6.3. Section 6.4 introduces first the general robust design problem and then the control system design problem presenting several techniques that lead to robust control designs. The novel robustness evaluation index is defined in section 6.5. The optimized disturbance weighting method which includes the parameter uncertainties in the mixed sensitive control system design problem is explained in section 6.6. In section 6.7 the impact of the controller structure choice and the design method on the robustness of a Buck converter control are investigated. Finally, section 6.8 evaluates the most relevant findings of this chapter.

6.1 Motivation

Robust embedded system design is becoming increasingly important for safety and manufacturing cost reasons in many industry sectors such as automotive and aerospace. As the embedded system size is constantly shrinking, many unknown parameters can significantly affect the overall system performance. In particular, analog circuits may be strongly sensitive to parameter variations due to the manufacturing process as well as to changes on the environmental conditions. For this reason, there is a growing demand for design tools that ensure system performance by including the effects of design uncertainties at several abstraction levels. The robustness evaluation of embedded system designs is a further challenge. As there is normally a very large number of heterogeneous system parameters, this is a quite difficult task.

6.2 Related Work

To assess the robustness of microelectronic circuits and systems, Barke et al. presented a general approach for robustness modeling and several ways to quantify design robustness [11]. The defined *robustness probability* allows system designers to analyze and choose out of different implementations of mixed analog and digital systems. Beyer et al. investigated how to account for design uncertainties in order to improve system robustness [14]. They presented a useful classification of design process uncertainties and summarized methods for robustness measurement as well as for robust design optimization with respect to uncertainty sources. To cope with multi-objective robust optimization problems, Deb et al. presented two procedures for finding Pareto optimal solutions that are less sensitive to small changes in variables (robust solutions) [23]. Wang et al. considered small and large variations in design variables and design environment parameters [122].

Focused on improving the robustness properties of control systems, Alyaqout et al. introduced an approach that combines robust design with robust control [5]. As the computational cost of improving the robustness of the whole system is very high, they considered sequential and iterative robust design optimization strategies.

The robustness properties of control systems are strongly dependent on the controller structure and parameter tuning. Loop shaping is a commonly used method for designing robust controllers in the frequency domain. It attenuates load disturbances and reduces the sensitivity to process [11] variations while keeping measurement noise low. This design method has two disadvantages, the resulting controller is very complex and it is difficult to select appropriate *loop shaping weights*. To mitigate these problems, Chaiya et al. proposed a mixed sensitivity design method based on a fixed-structure controller (approximated PID) that takes into account process parameter uncertainties [16]. They utilized Particle Swarm Optimization (PSO) for tuning the controller parameters. In order to shape the disturbance weighting function they estimated the disturbed plant uncertainty carrying out Monte Carlo simulations.

For many control systems, the mixed sensitivity design optimization method achieves excellent performance and robustness properties if well suited weighting functions are utilized for constraining the optimization problem. As the statistical computation of the worst case disturbed plant demands a long time, faster and more reliable computation methods are need to include the effect of parameter tolerances in the loop shaping weighting functions. A further issue is that the time required for solving the fixed structure control design optimization problem may severely limit the robust tuning of controller parameters. Heuristic global optimization algorithms such as Particle Swarm Optimization (PSO), Simulated Annealing (SA), etc. are frequently used but they are very time demanding and do not guarantee an optimal controller parameter tuning. Therefore, they present a very limited capability to cope with circuit level robust system design optimization problems. As power electronic systems are very susceptible to instability, design refinement methods that consider the impact of unmodeled nonlinear dynamics on the system stability are needed [111] [124]. As robust control design methods may lead to very complex control structures, robustness assessment methods that are suitable to guide the control system design process are also necessary to evaluate the properties of several controllers. In particular, the nonlinear dynamic properties of AMS systems must be considered.

[11] In control system context, ”process” means the process under control or plant

6.3 Contribution to Robust System Design Optimization

The aim of the research work presented in this chapter was to define a *robustness evaluation model* that can be easily applied for the analysis and optimization of embedded control system designs at several abstraction levels as well as to develop a robust control system design optimization method that reduces the controller implementation complexity and the time required for finding optimal controller parameters. The main research activities carried out to this end were the following:

- To investigate which robustness evaluation methods are suitable for the analysis of electronic system designs.
- To work out a robustness evaluation method that is appropriated for the analysis of control system designs at several abstraction levels.
- To evaluate several robust control design methods regarding the effectiveness and simplicity.
- To derive methods for efficiently guiding and constraining the robust design optimization problem through several abstraction levels.

For the evaluation of different controller designs, a mixed sensitivity robustness index was defined which captures the essential control system properties as well as the impact of plant parameter tolerances. To cope with the robustness evaluation of nonlinear control systems, a linear system model is identified from the simulation results at several controller operation working points using the convergence properties of orthogonal polynomials [134].

In order to fastly achieve a robust control system design at circuit level that does not unnecessarily restrict the controller performance, this dissertation proposes a design method that computes plant parametric uncertainty in a novel way (see [36]), utilizes internal model control (IMC) to find a suitable set of initial controller parameters and stepwise optimizes the fixed structure controller parameters for maximizing the the above mentioned mixed sensitivity robustness index.

As a very simple control system design evaluation can be carried out utilizing the proposed mixed sensitivity robustness index, it significantly reduces the analysis time during design exploration. Taking into account that the proposed robust design method speeds up the control system optimization task and is applicable to widely used controllers such as PID, the contribution of this chapter is very significant from a practical point of view.

6.4 Robust System Design

As shown in Fig. 6.1, a system Q which transforms an input signal vector $[r(t)]$ into an output signal vector $[y(t)]$ under unknown operation conditions and external perturbations (denoted by the vector $[w(t)]$) is characterized by a set of performance properties or features (denoted by the vector $[j(t)]$) [11].

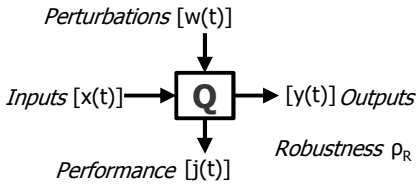


Fig. 6.1: System performance

In order to evaluate the performance properties of a system, nominal operation conditions $[w_{nom}(t)]$ are normally considered. The set of tolerated perturbations or admissible operating conditions considered in the system design is known as *mission profile* $\mathbb{W}_M$ and it is normally specified as an interval vector $[\bar{w}_{spec}]$ in the *perturbation space* $\mathbb{W}$. The set of properties or performance features that the system Q shall fulfill during its operation is known as the *robust performance space* $\mathbb{J}_R$ and it is often specified as an interval $[\bar{j}_{spec}]$ in the *property or performance space* $\mathbb{J}$.

6.4.1 System Robustness Evaluation

A system Q is called *robust* if the performance subspace $\mathbb{J}_M$ resulting from mapping the system inputs under the specified perturbations $\mathbb{W}_M$ is included in the *robust performance space* $\mathbb{J}_R$. This is usually a limited region around the nominal performance point $[j_{nom}(t)]$. As shown in Fig. 6.2, a robust system may tolerate a set of disturbances $\mathbb{W}_R$ which are partially not included in the *mission profile* $\mathbb{W}_M$.

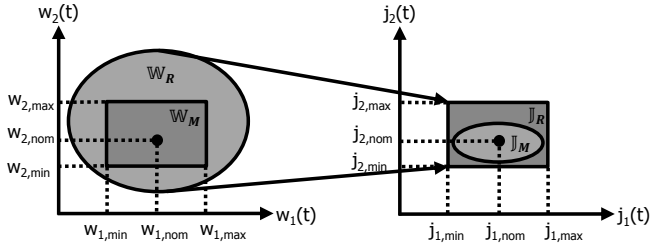


Fig. 6.2: System robustness

The robust implementation of a system strongly depends on the size of performance subspace $\mathbb{J}_M$. The robustness of a system Q can be defined as the probability that the system fulfills the specified properties (*robust performance space* $\mathbb{J}_R$) although the operating conditions are not included in the *mission profile* $\mathbb{W}_M$ [11]. The *robustness probability* ρ_R is given by:

$$\rho_R := P([j(t)] \in \mathbb{J}_R \,|\, [w(t)] \notin \mathbb{W}_M) \tag{6.1}$$

which can be then computed as follows:

$$\rho_R = \frac{P([w(t)] \in \mathbb{W}_R \wedge [w(t)] \notin \mathbb{W}_M)}{1 - P([w(t)] \in \mathbb{W}_M)} \tag{6.2}$$

Computing the system robustness in terms of probability allows the direct comparison between very different system designs. Note that the probability distributions are used in this definition for the normalization of the robustness value.

6.4.2 Robust System Design Optimization

Systems designed to fulfill specification requirements and optimized to improve their performance may be very sensitive to parameter changes. For this reason, a *robustness analysis* should be always part of the design process. In a general way, a system design is considered to be robust if the system performance remains relatively unchanged under uncertain system manufacturing and operation conditions.

In order to consider parameter uncertainties in the system design, a three step system design method was proposed by Taguchi [14]:

1. System design: The system structure is designed and parameterized to fulfill the specification requirements.
2. Parameter design: The parameters are optimized to improve the robustness (sensitivity analysis).
3. Tolerance design: The parameter tolerances are considered for fine tuning design parameters.

Robust design techniques should not be limited to cope with design parameter sensitivities (parameter design step). The optimal operating point should be also considered. Fig. 6.3 shows the resulting performance subspace $\mathbb{J}_M$ of two different robust design solutions under the specified perturbations $\mathbb{W}_M$. Although both system designs are robust, the solutions present a very different robustness. Considering the system robustness in the system design optimization problem is much better. It is known as *robust design optimization*.

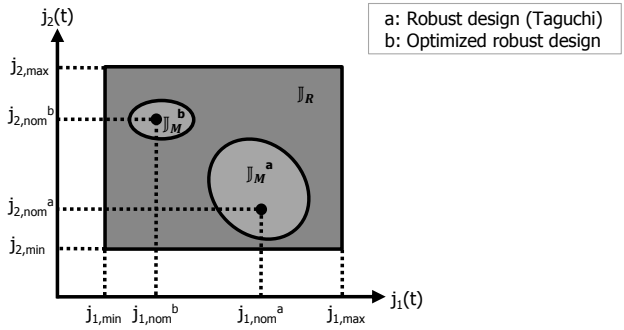


Fig. 6.3: Robust design optimization problem

The objective or cost function utilized for solving the design optimization problem and the constraints imposed to the design parameters mainly determine the achieved system performance and robustness. The optimization algorithm and its parameters have a considerable impact on the optimization time and optimization results. In the next section the most popular robust control design methods in the time and frequency domain are briefly explained.

6.4.3 Robust Control Design

Robust control design methods are well suited to deal with both changes of operating conditions and parameter tolerances. For this reason, this dissertation considers that the system Q under design consists of two parts, a *dynamical process* or *plant* $P([u(t)],[y(t)],[p_p])$ and a *controller* $C([r(t)],[y(t)],[u(t)],[p_c])$ as represented in Fig. 6.4.

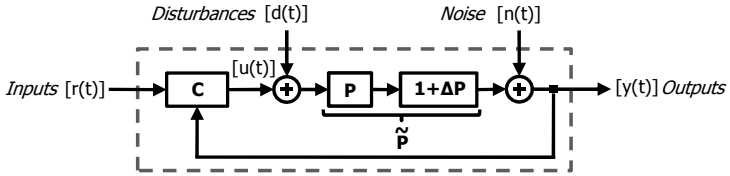


Fig. 6.4: Robust control system block diagram

The *perturbed process* $\tilde{P}$ has three inputs, the *control signal vector* $[u(t)]$, the *disturbance signal vector* $[d(t)]$ and the *noise signal vector* $[n(t)]$. The *process perturbation* ΔP represents variations of the process behavior due to parameter uncertainties and modeling inaccuracy. The *measured output signal vector* $[y(t)]$ is used by the controller to compute the control signal vector $[u(t)]$. The process and controller parameters are represented by the vectors $[p_p]$ and $[p_c]$ respectively. For system design, it is assumed that the controller C is free of disturbances.

Typical requirements for robust control systems are [8]:

1. The system outputs $[y(t)]$ should follow the command signals $[r(t)]$.
2. The load disturbances $[d(t)]$ should be attenuated by the control.
3. The measurement noise $[n(t)]$ should be limited in the loop.
4. The control sensitivity to process variations ΔP should be small.

Considering that the dynamical perturbations $[w(t)]$ acting on the system have different dynamical properties, they are subdivided for convenience into load disturbances $[d(t)]$ and measurement noise $[n(t)]$. The attenuation of load disturbances can be improved by increasing the system bandwidth, but it results in more noise injection (mutually conflicting design requirements). In order to obtain a robust control system design, linear quadratic and H_∞ control are usually applied.

Linear Quadratic Regulator (LQR)

A trade-off between reducing the effect of load disturbances and the injection of measurement noise can be achieved if the control design minimizes the following loss function [8]:

$$J_{LQ} := \frac{1}{T} \cdot \int_0^T (\|[y(t)]\|^2 + \rho \cdot \|[u(t)]\|^2)\, dt \tag{6.3}$$

where ρ is a weighting parameter and ||.|| denotes the vector norm. The *loss function* J_{LQ} balances the control actions against deviations in the output.

H_∞ *Control*

If both the process and the controller are linear systems, the impact of all external influences acting on the system (which are represented by the *generalized disturbance signal vector* $[w(t)]$) can be considered to be independent of the current command signals $[r(t)]$ and the current operating point. The deviation of the control and output signal vectors from their operation values are described by an additional output signal $[z(t)]$ known as the *generalized error* [8]. After applying the Laplace transform to the system equations, the generalized error $z(s)$ can be computed as follows:

$$z(s) = H(s) \cdot w(s) = \begin{bmatrix} S(s) & P(s) \cdot S(s) \\ C(s) \cdot S(s) & T(s) \end{bmatrix} \begin{bmatrix} d(s) \\ n(s) \end{bmatrix} \tag{6.4}$$

where $S(s)$ and T(s) are the sensitivity and complementary sensitivity functions respectively. They are defined as follows:

$$S(s) := \frac{1}{1 + P(s) \cdot C(s)} \tag{6.5a}$$

$$T(s) := \frac{P(s) \cdot C(s)}{1 + P(s) \cdot C(s)} \tag{6.5b}$$

The robust control design problem consists in minimizing the generalized error. The design problem is thus reduced to finding a controller $C(s)$ such that the gain of the transfer function $H(s)$ is small even when the process $P(s)$ has uncertainty (modeling or parametric).

A robust controller that has the same order as the process can be found solving the minimization problem:

$$\|H(P(s), C(s))\|_\infty < \gamma \tag{6.6}$$

where γ is just a design parameter. Both the system performance and robustness are optimized in the H_∞ control design.

Mixed Sensitivity Control

Minimizing the infinity norm of $H(s)$ means that all frequencies of the controller input signals and disturbances are equally important. This is not very realistic because load disturbances typically have low frequencies and measurement noise has high frequencies [8]. Introducing a *weighting filter*, the design problem can be modified so that the disturbances of different frequencies get different emphasis.

The sensitivity function $S(s)$ is a very good indicator of closed loop performance. In order to shape $S(s)$ over the bandwidth frequency ω_B, the following *performance weighting function* $W_P(s)$ is often utilized [109]:

$$W_P(s) = \frac{s/M + \omega_B}{s + \omega_B \cdot A} \tag{6.7}$$

where the peak specification M avoids the amplification of noise at high frequencies. The control design problem is thus reduced to find a controller $C(s)$ that satisfies the relation:

$$\|W_P(s)S(s))\|_\infty < 1 \tag{6.8}$$

The *performance weighting function* $W_P(s)$ specifies a bandwidth lower bound. An upper bound is also required to make sure that the *loop transfer function* $L(s) = C(s)P(s)$ rolls off sufficiently fast at high frequencies. The *disturbance weighting function* $W_T(s)$ is introduced for the roll-off specification of L(s) above the bandwidth. It shapes the complementary sensitivity $T(s)$ and can be computed as follows [109]:

$$W_T(s) = \frac{s + \omega_B/M}{A \cdot s + \omega_B} \tag{6.9}$$

In order to restrict the magnitude of the plant input signals, a *control weighting function* $W_u(s)$ may be additionally introduced. It places an upper bound on the magnitude of $C(s)S(s)$.

The mixed sensitivity specifications are utilized to obtain an optimal H_∞ controller by solving the following optimization problem:

$$\min_{C(s)} \left\| \begin{matrix} W_P(s) \cdot S(s) \\ W_T(s) \cdot T(s) \\ W_u(s) \cdot C(s) \cdot S(s) \end{matrix} \right\|_\infty \tag{6.10}$$

6.5 Control System Robustness Evaluation

The robustness probability index defined in section 6.4 has some drawbacks for evaluating the robustness of several control systems and control design approaches. The controller robustness computed using Eq. (6.2) depends on the performance space J_R and therefore, the robustness evaluation strongly depends on the design method. Several controller specifications may result in completely different robustness characterizations. Furthermore, the denominator of Eq. (6.2) leads to very sensitive computations if it is close to zero.

In order to overcome the practical problems of the existing approaches for robustness evaluation, this dissertation proposes a robustness index ϕ_R which is based on the sensitivity and complementary sensitivity functions. This set of design parameters captures the essence of the control problem [33] [34] [26]. They reflect the stability, robustness and performance of a control system. The sensitivity function $S(s)$ is a very good indicator of closed loop performance. Furthermore, it quantifies the amplification of disturbances at the output of the plant as well as the influence of plant parameter uncertainties [109]. The complementary sensitivity function $T(s)$ is an important performance indicator of the control system response to set-point changes. It determines the overshoot in the system response [109].

The *mixed sensitivity robustness index* ϕ_R is defined as follows:

$$\phi_R := \sum_{k=1}^{K}\sum_{r=1}^{R} P(k)\cdot P(r)\cdot \phi_{R_{nom}}\cdot \phi_{R_\Delta}, \quad \sum_{k=1}^{K} P(k) = 1, \quad \sum_{r=1}^{R} P(r) = 1 \tag{6.11a}$$

$$\phi_{R_{nom}} := \sqrt{\frac{1}{2}\cdot\left(\frac{1}{\|S_{nom}(s)\|_\infty{}^2} + \frac{1}{\|T_{nom}(s)\|_\infty{}^2}\right)} \tag{6.11b}$$

$$\phi_{R_\Delta} := \left(1 - \sqrt{\phi_{SD}^2 + \phi_{TD}^2}\right) \tag{6.11c}$$

$$\phi_{SD} := \sigma_{\frac{1}{\|S(s)\|_\infty}} + \left|\mu_{\frac{1}{\|S(s)\|_\infty}} - \frac{1}{\|S_{nom}(s)\|_\infty}\right| \tag{6.11d}$$

$$\phi_{TD} := \sigma_{\frac{1}{\|T(s)\|_\infty}} + \left|\mu_{\frac{1}{\|T(s)\|_\infty}} - \frac{1}{\|T_{nom}(s)\|_\infty}\right| \tag{6.11e}$$

The robustness of a control system design under nominal parameter values (denoted by $\phi_{R_{nom}}$) is captured by Eq (6.11) as a normalized distance which is defined in terms of the system sensitivities. The impact of the parameter uncertainties on the system robustness (denoted by ϕ_{R_Δ}) is expressed in terms of the standard deviation σ and the expected value μ. As shown in Fig 6.5, the normalized distances ϕ_{SD} and ϕ_{TD} define the perturbation area around the average value.

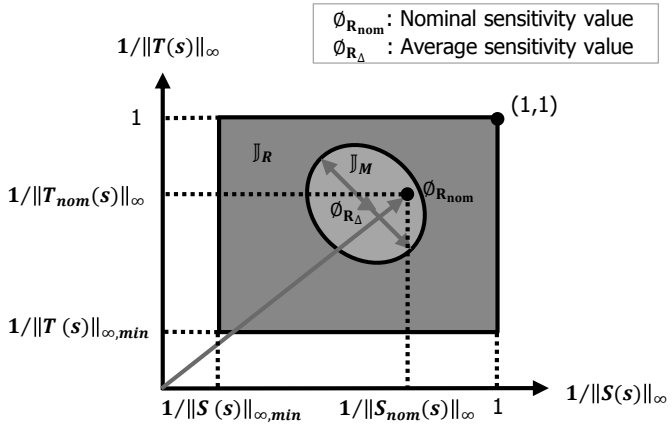


Fig. 6.5: Mixed sensitivity robustness index

As $S(s) + T(s) = 1$, the mixed sensitivity robustness index ϕ_R is constrained to the value range $[0, 1]$ which allows a very simple design evaluation. The value 1 corresponds to an ideal robust control system.

In order to characterize the robustness of nonlinear systems, K different operating point changes are considered for the robustness index computation. A linear system that matches the response to each operating point change is found using identification methods and utilized for the computation of the system sensitivity functions. The corresponding probability that a given operating point change assigned to the index k occurs is denoted by $P(k)$. In a similar way, the parameter ranges are subdivided in R regions and a linear system is identified for each parameter value in the region sample set. $P(r)$ denotes the probability that the system parameters are included in the region r. If the probability distributions for computing $P(k)$ and $P(r)$ are known, an over-conservative robustness evaluation can be avoided. Otherwise, a uniform distribution is assumed.

This dissertation uses a linear system identification method based on orthogonal polynomials which was extended by Zhu for the design of DC-DC step-down converters [134]. It utilizes an operational matrix of integration for the computation of a linear transfer function and can be directly applied to the signal representation proposed in chapter 4 (see appendix A). Thus, this system identification method takes advantage of the behavior computation methods presented in chapter 5.

6.6 Robust Design Optimization of AMS Systems

This dissertation proposes the following 3 general steps to carry out a robust control design of any nonlinear AMS system:

1. To model plant uncertainties such as they can be included in the robust design optimization problem.
2. To select one (or more) suitable controller and plant to solve the design problem and optimize the linearized AMS system to be robust.
3. To fine-tune the controller parameters and to verify AMS system design robustness using an accurate plant model which captures the main system nonlinear characteristics.

This work modifies the mixed sensitivity robust design method explained in section 6.4.3 to fit the proposed robust design method. The following sections present concrete methods for carrying out the design steps. As will be shown in section 6.7, the previously defined robustness index ϕ_R is well suited for both control system evaluation (design space exploration) and controller fine tuning.

6.6.1 Modeling Parametric Uncertainty

It is impossible to exactly model a physical system. To obtain a robust system stability and performance design, the process uncertainty needs to be considered. *Parametric uncertainty* is an unavoidable form of uncertainty in physical systems. The parameters describing the system are unknown but normally defined in a given range. Using a *multiplicative perturbation model*, the uncertain plant transfer function $\tilde{P}(s)$ is represented in the following form:

$$\tilde{P}(s) = (1 + \Delta P(s)) \cdot P(s) \tag{6.12}$$

The idea behind this uncertainty model is that the *normalized plant perturbation* $\Delta P(s)$ provides a plant uncertainty profile [27].

In order to include the plant uncertainty profile in the controller design, this work proposes to obtain a disturbance weighting $W_T(s)$ by solving the following optimization problem in the range of interest:

$$|W_T(jw)| := \frac{1}{M} \cdot \max_{p_p} |\Delta P(s)|_{s=jw} \, , \, p_p \in [p_{p,min}, p_{p,max}] \tag{6.13}$$

Note that it is sufficient to consider only the magnitude of $W_T(jw)$, i.e. the consideration of the weighting function phase is not necessary for determining $W_T(s)$.

As the solutions of Eq. (6.13) for two close frequencies do not differ too much, this work uses the already computed solution at a given frequency jw as start point for the optimization at the next adjacent frequency reducing considerably the computation time. Thus, a novel method called *optimized disturbance weighting* that enables the fast and reliable inclusion of parameter uncertainties in the design process is proposed. It consist of the following steps [36]:

1. Derive a parametric linear model of the plant.
2. Compute the magnitude of the normalized plant perturbation using Eq. (6.13).
3. Find a minimum phase transfer function that matches the computed uncertainty profile.

6.6.2 Fixed Structure Robust Controller Design

To get a design that has both a good reference tracking and a good disturbance rejection, a two degrees-of-freedom controller is necessary. If $W_T = W_u = 1$ in Eq. (6.10), the resulting H_∞ controller is close to a PI controller which shows a small steady state error A. For many applications, a fixed structure controller (such as a classical PID controller) can be tuned to get a robust control design. Therefore, this work reduces the mixed sensitivity design problem to chose a suitable controller and to tune the controller parameters p_c for minimizing the objective function:

$$\min_{p_c} \left\| \begin{matrix} W_P(s) \cdot S(s) \\ W_T(s) \cdot T(s) \end{matrix} \right\|_\infty \tag{6.14}$$

The performance weighting function $W_P(s)$ must be derived from the design specification. This work utilizes Eq. (6.7) to compute $W_P(s)$.

Particle Swarm Optimization (PSO) and Simulated Annealing (SA) were proposed by several authors in order to achieve a robust control that utilizes a fixed structure controller [16] [125] [52]. Even though the weighting filters $W_P(s)$ and $W_T(s)$ significantly reduce the solution space, finding the controller parameters that minimize the mixed sensitivity design problem requires an excessively large number of iterations for obtaining a good solution if such heuristic optimization methods are applied.

A controller design method widely utilized in several industrial areas for controller parameter tuning is Internal Model Control (IMC). As shown in figure 6.6, this control scheme utilizes an internal representation of the plant P_i for building an intrinsic feedback control loop. The design of the internal controller $Q_i(s)$ is straightforward. The internal plant model $P_i(s)$ is factorized as follows:

$$P_i(s) = P_L(s) \cdot P_H(s) \tag{6.15}$$

where the minimum phase transfer function $P_L(s)$ is a low-pass filter that contains the left half plane (LHP) zeros of the plant model $P_i(s)$. The high-pass filter $P_H(s)$ contains right half plane (RHP) zeros and the time delays of the plant model $P_i(s)$.

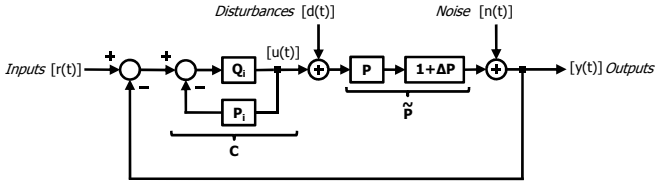


Fig. 6.6: Internal model control

The internal controller $Q_i(s)$ is designed to obtain the inverse system response. In order to obtain a realizable controller $C(s)$, a low-pass filter $F(s)$ is added to the internal controller $Q_i(s)$. The controller $C(s)$ is computed as follows:

$$C(s) = \frac{Q_i(s)}{1 - P_i(s) \cdot Q_i(s)} \tag{6.16a}$$

$$Q_i(s) := \frac{F(s)}{P_L(s)}, \quad F(s) := \frac{1}{(\lambda s + 1)^k}, \quad \lambda \leq \frac{1}{w_0} \tag{6.16b}$$

The order k of the low-pass filter $F(s)$ is chosen to make the internal controller transfer function proper and the filter parameter λ is selected to achieve the desired closed loop bandwidth w_0.

The reduction of the controller parameter λ improves the closed loop performance but decreases the robustness of the control system to external disturbances and parameters variations. In order to find a robust controller design, this dissertation applies sequential quadratic programming (SQP) for solving the reduced mixed sensitivity optimization problem defined by Eq. (6.10). As only the parameter λ needs to be adjusted for finding an optimal robust controller, this fast and deterministic optimization method provides very good results.

In order to approximate the parameters of the selected fixed structure controller, this dissertation applies balanced model order reduction to the optimized internal model controller $C(s)$.

The optimal computation of the initial controller parameters leads to a more efficient design approach than the robust optimization of the fixed structure controller using arbitrary initial parameters.

The following steps are carried out for robust controller design:

1. Design and optimize an IMC controller according to Eq. (6.16) and to Eq. (6.14) respectively (after computing the weighting functions with Eq. (6.7) and Eq. (6.13)).
2. Choose a suitable fixed structure controller (e.g. PID) and determine the initial controller parameters by reducing the IMC controller transfer function to the selected controller structure.
3. Optimize the controller parameters according to Eq. (6.14).

Finally, the following steps are carried out to improve the design robustness considering the intrinsic nonlinearity of AMS systems:

1. Include the nonlinear system characteristics in the plant model.
2. Tune controller parameters for optimizing the mixed sensitivity robustness index Eq. (6.11).
3. Verify controller design robustness.

The verification of the controller robustness is considered in chapter 7.

6.7 Experimental Results

In order to validate the proposed robust design method, the DC-DC step-down power converter control system shown in Fig. 6.7 was designed. It utilizes pulse width modulation (PWM) to control the Buck converter circuit shown in Fig. 3.3a) and is designed to provide a 5 V nominal output voltage.

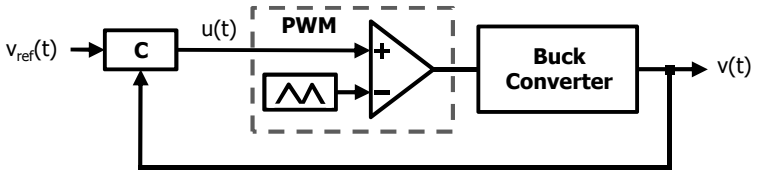


Fig. 6.7: Buck converter control

The nominal circuit parameters[12] are described in Table 6.1. For design and analysis purposes it was assumed that the circuit parameters V, L, R and C have ±20% tolerance.

Inductance L	4.1 μH
Capacitance C	376 μF
Load resistance R	1.0 Ω
Input voltage V	12 V
Inductor resistance R_L	0.08 Ω
Capacitor resistance R_C	0.005 Ω
Switching frequency f_s	100 kHz

Table 6.1: Buck converter parameters

With the aim of demonstrating the effectiveness of the proposed robust design method, a simple approximated PID controller was chosen for design. It is defined by the following transfer function:

$$C(s) := K_p + \frac{K_d}{s} + \frac{K_d.s}{s+\tau_d} \tag{6.17}$$

The derivative control part is implemented as a low-pass filter to obtain a realizable controller. It introduces a small time constant τ_d. Properly tuned, this controller is able to achieve a good set-point tracking and rejection of disturbances. The following constraints were defined for design: $\omega_B = 5.000\ rad/s$, $M = 2$ and $A = 1e^{-4}$.

For comparison purposes, the traditional robust control design techniques presented in section 6.4.3 were additionally applied for finding a robust controller. The H_∞ controller design is straightforward. Using Eq. (6.7), Eq. (6.9) and the MATLAB function *mixsyn*, the following controller was obtained:

$$C(s) = \frac{8.20e^{10} \cdot s^3 + 8.20e^{18} \cdot s^2 + 1.91e^{23} \cdot s + 5.72e^{27}}{s^4 + 9.31e^{12} \cdot s^3 + 1.69e^{19} \cdot s^2 + 6.38e^{24} \cdot s + 6.32e^{24}} \tag{6.18}$$

For the design of the LQR control, a weighting factor $\rho = 0.01$ was chosen. This controller additionally requires the inductor current measurement for computing the plant state as well as an outer PI controller.

[12] The parameters labels correspond to the average circuit shown in Fig. 5.4

For design proposes, the Buck converter circuit was linearized utilizing an average state space model (see section 7.9 for more details). Using this plant model, the plant uncertainty profile was computed solving the optimization problem defined by Eq. (6.13). As shown in Fig. 6.8, this approach is more efficient and accurate than Monte Carlo simulations. A speed-up of 8x was achieved.

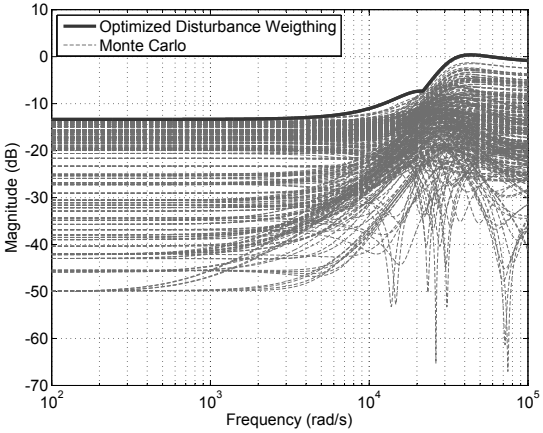


Fig. 6.8: Worst case plant perturbation

Fitting the computed worst case response, the following disturbance weighting transfer function was obtained:

$$W_T(s) = \frac{0.71 \cdot s^2 + 3.06e^4 \cdot s + 3.83e^8}{s^2 + 3.43e^4 \cdot s + 1.68e^9} \tag{6.19}$$

It takes into account the possible plant parameter variations which reduces the robust stability margin introduced by $W_T(s)$ in the mixed sensitivity control design problem.

The proposed PID controller was designed by finding an optimal IMC controller that minimizes Eq (6.14) and then reducing the IMC controller transfer function[13] to the form given by Eq. (6.17). As only the IMC filter parameter λ must be optimized, this method is very fast.

[13] For more details see [99].

The linear plant model utilized in the first design step does not consider that the PWM duty cycle is limited to a range from 0.1 to 0.9 (actuator saturation). This modeling uncertainty leads to an inconsistency between plant and controller states when the control signal saturates. This control saturation may produce performance and stability issues. As shown in Fig. 6.9, there is a performance degradation.

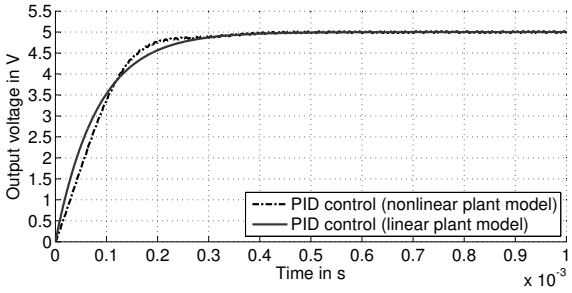
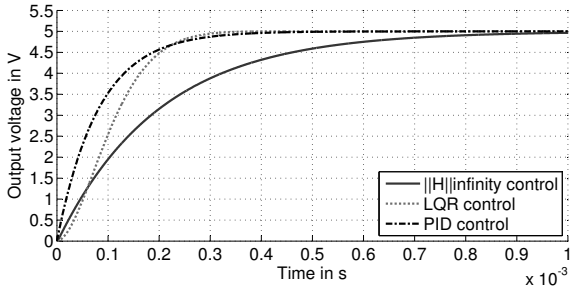


Fig. 6.9: PID control characteristic for linear and nonlinear plant model

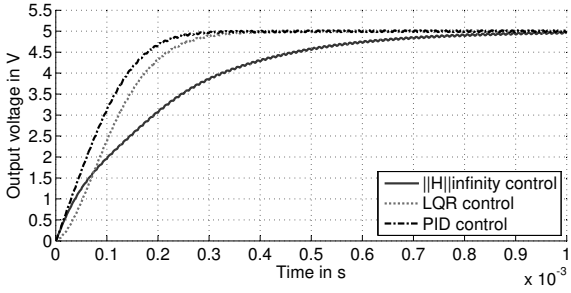
After modeling the nonlinear plant, the initial PID controller design was fine-tuned by optimizing the controller parameters using first Eq. (6.14) and then Eq. (6.11). As the IMC tuning method produces a very good initial design, sequential quadratic programming (SQP) is able to very quickly solve both optimization problems.

Fig. 6.10 shows the set-point-tracking characteristics of the obtained control designs for the average and nonlinear Buck converter models. The nonlinear buck converter model includes both the nonlinear actuator characteristics as well as the switching behavior of the electrical circuit (ideal switched linear network).

All controllers were designed to be robust and therefore, there is no overshoot in the control responses. The H_∞ controller presents the largest settling time. The more sophisticated LQR control which additionally requires the inductor current monitoring and an outer controller presents a very good performance. The simple PID controller is faster than the LQR control. After parameter tuning based on the nonlinear plant model, it is even able to cope with the saturation issues. Thus, applying the proposed design method the simplest controller achieves the best control performance (shortest settling time).



(a)

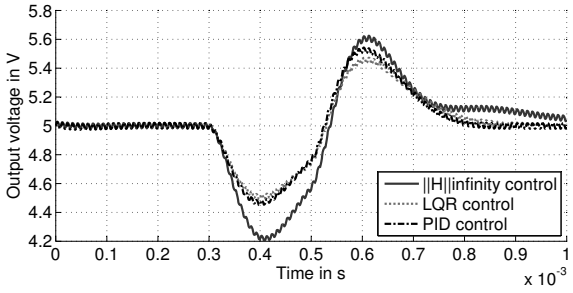


(b)

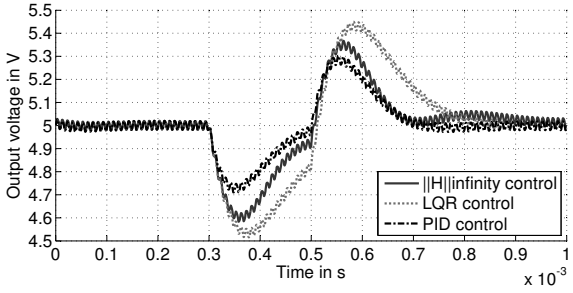
Fig. 6.10: Set-point tracking: a) Linear plant b) Nonlinear plant

In order to analyze the robustness of the several controllers to external disturbances, a 1 V input voltage disturbance and a 1 A load current disturbance were applied to the several control systems. The resulting disturbance rejection characteristics are shown in Fig. 6.11 (a) and (b) respectively. The H_∞ controller shows a quite limited disturbance rejection capability. The LQR control presents the best voltage disturbance rejection but the worst current disturbance rejection. The PID control achieves the best overall disturbance rejection characteristics.

Fig. 6.12 shows the set-point tracking of the several controllers under parameter variations (Monte Carlo simulations). All controllers are robust to parameter variations. In order to better analyze the results, table 6.2 presents the area between the upper and lower response bounds as well as the maximal deviation from the nominal response due to parameter variations.



(a)



(b)

Fig. 6.11: Disturbance rejection: a) Input voltage b) Load current

The area between the upper and lower response bounds (first table column) decreases with the improvement of the control performance. The PID control design shows the smallest impact of parameter variations on the response. On the other hand, there is a correlation between the maximal deviation from the nominal response with the controller stability. The H_∞ control presents the lowest deviation from the nominal response, i.e. it is more stable than the LQR and PID controllers.

Fig 6.13 shows how the sensitivity and complementary sensitivity functions capture the impact of parameter variations on the system robustness. It is derived from the computed Monte Carlo simulations. Small and large parameter variations are plotted in different colors. The nominal controller designs are denoted by filed circles. The triangles denote the average values from all controller design variations.

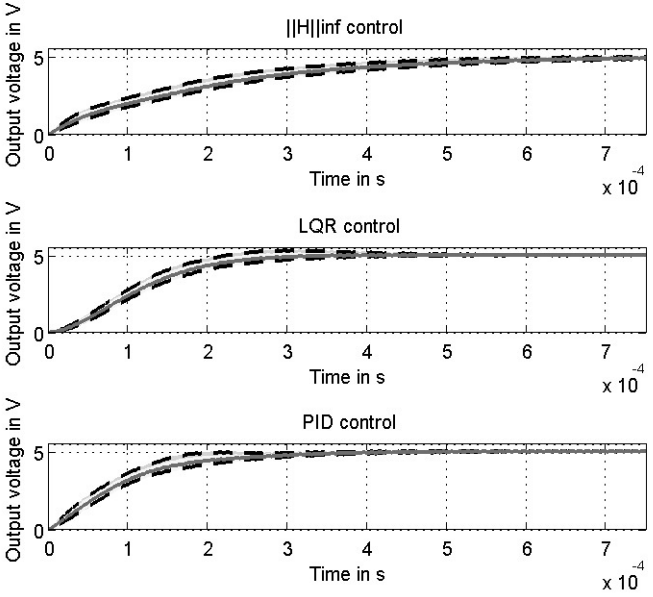


Fig. 6.12: Parameter variations for nonlinear plant model

Controller	Variation area	Maximum error
H_∞	$333.11e^{-6}$	$\mathbf{445.11e^{-3}}$
LQR	$236.28e^{-6}$	$459.56e^{-3}$
PID	$\mathbf{212.00e^{-6}}$	$553.60e^{-3}$

Table 6.2: Impact of parameter variations on system response

It can be see that the PID controller which was designed utilizing the optimized disturbance weighting shows a near optimum nominal design i.e. the robustness value is very close to the point $(1, 1)$. The average robustness value (triangle) is very close to the nominal design (circle) (which is optimal). A very important observation is that the mixed sensitivity robustness analysis captures the poor robustness (current disturbance rejection) of the LQR design which is not visible in the performance and parameter variation analysis.

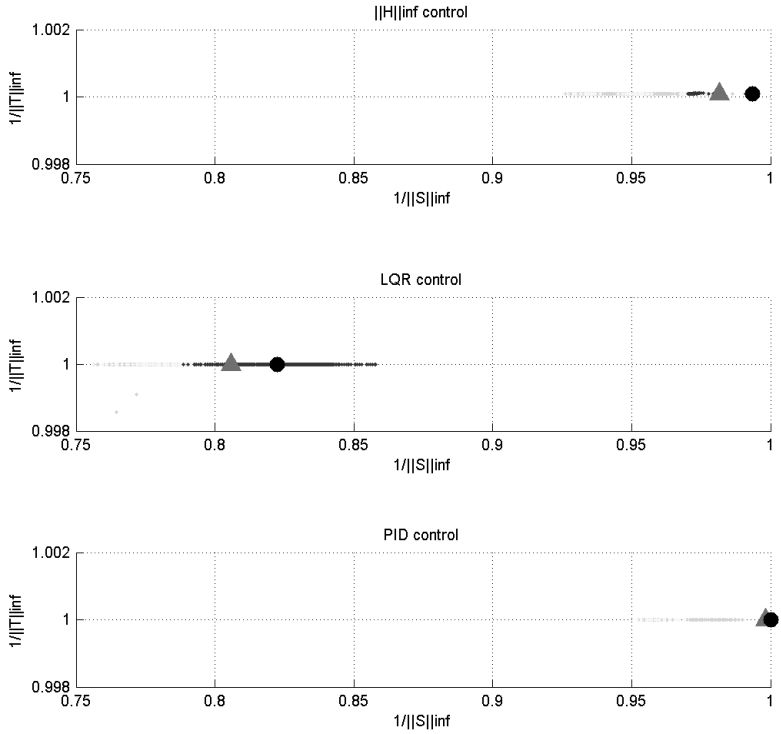
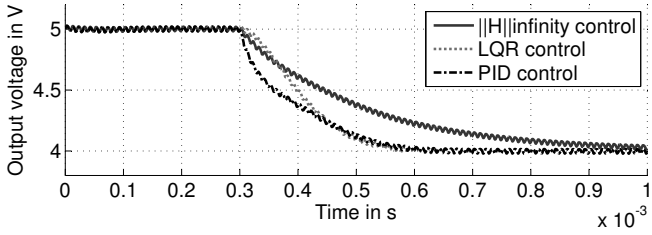
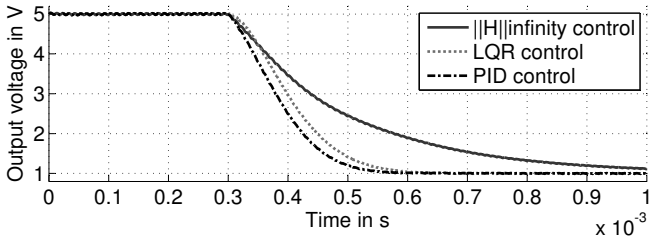


Fig. 6.13: Linear control system design characterization

Due to the nonlinear behavior of the Buck converter, the controller performance depends on the operating point. Fig 6.14 shows the performance of the designed controllers for small and large changes of the reference voltage ($1V$ and $4V$). The PID control shows a significant performance degradation for small changes of the reference voltage. These different control characteristics (performance, robustness and stability) must be included in the control design analysis. Figure 6.15 shows the corresponding changes of the sensitivity and complementary sensitivity functions for both multiple operating points and parameter variations. For the sensitivity computations, the response of the nonlinear closed loop system is linearized using orthogonal polynomials.



(a)



(b)

Fig. 6.14: Nonlinear plant model set-point-tracking: a) Small reference change b) Large reference change

The nonlinearity of the buck converter has a visible impact on the system stability. The robustness region is lager and rounder compared to the linear system case in Fig. 6.13. The corresponding robustness index ϕ_R for the several control designs is shown in Tab. 6.3. It reliably captures the system robustness for both the linear and the nonlinear control systems.

Controller	Linear model	Nonlinear model
H_∞	0.9678	**0.9920**
LQR	0.8815	0.9141
PID	**0.9909**	0.9793

Table 6.3: Robustness index ϕ_R

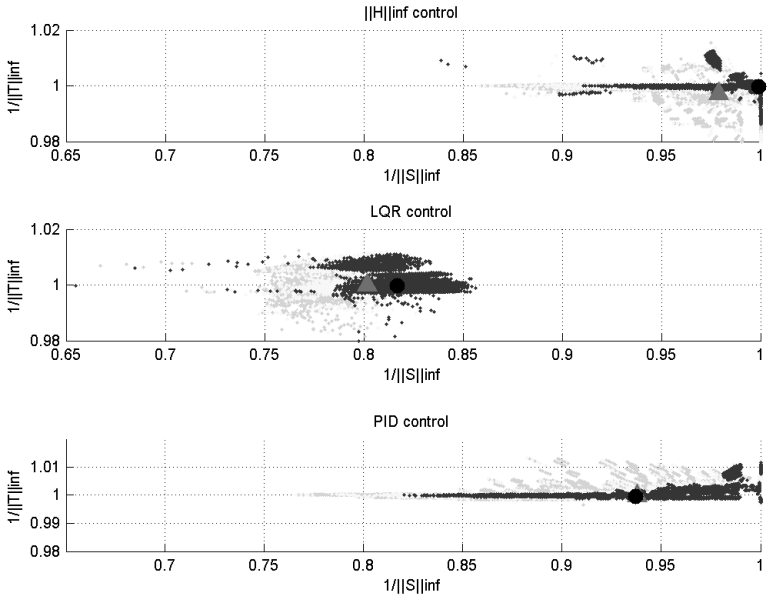


Fig. 6.15: Nonlinear control system design characterization

In the nonlinear case, the degradation of the system performance (tendency to be unstable) leads to a robustness loss for the PID controller (optimized in terms of the parameter uncertainty profile). The H_∞ is more robust over a large number of operating points and copes better with unmodeled plant dynamic. Note that there is an interesting correlation of the robustness index values with the uncertain response values in Tab. 6.2. Not only the variation area between the uncertain response bounds is important for the robustness assessment. The maximum uncertain response deviation from the nominal circuit behavior is relevant for characterizing the stability robustness of AMS control systems.

6.8 Chapter Summary and Conclusions

This chapter introduced the classical robust design techniques (Taguchi) which focus on identifying robust design solutions, explained the drawbacks of separated performance and robustness design optimization and described how robust control design methods minimize the impact of system perturbations and parameter variations. After that, a robust control system design method was proposed which is based on the following principles:

- The impact of parameter tolerances due to the manufacturing process is considered in a non-conservative way during system design.
- It allows the selection of any control system structure.
- It is applicable to the several AMS system abstractions in all design process stages.

Several improvements of the initial work (presented in [36]) were carried out in this dissertation. In order to solve the robust design optimization problem faster and more reliable, the IMC controller design method was considered for finding a near optimal set of initial controller parameters. Heuristic design optimization methods such as simulated annealing are not required for coping with the robust controller parameter optimization. In order to enable the evaluation and optimization of both linear and nonlinear control system designs, the proposed mixed sensitivity robustness index ϕ_R was extended for including the effect of several operating point changes and parameter regions. Furthermore, a weighting factor was also introduced in mixed sensitivity robustness index ϕ_R which avoids over-conservative robustness characterizations due to conditions that are unlikely to occur. Thus, robustness evaluations leading to a system design which presents a poor performance or is significantly more expensive may be avoided.

The investigation and analysis activities have shown that:

- The robustness of control systems strongly depends on the method utilized for controller parameter tuning.
- The optimized disturbance weighting design method works very well for linear systems.
- The modeling of nonlinear system properties is necessary for a reliable robust design refinement.

- The proposed mixed sensitivity robustness index is well suited for the impact evaluation of different controller structures and parameter tuning methods on the control system performance and robustness for linear and nonlinear systems.

The experimental results confirm that the set-point tracking, the disturbance rejection and the sensitivity to parameters variations of a control system can be significantly improved by reducing the control system bandwidth ω_B taking into account the plant response uncertainty bounds.

A very important finding is that the tuning of controller parameters may have a more significant impact on system robustness than the intrinsic controller properties. Even for a simple PID controller, the proposed robust design method achieves very good results.

A further finding is that a careful analysis of system nonlinearity and unmodeled dynamic is necessary for avoiding a poor control performance or stability problems even for well established robust control design methods such H_∞. This is particularly important for power control systems because its limited control action may lead to a strongly nonlinear behavior. This reinforces the value of the modeling methods proposed in chapters 3 and 5 for supporting robust design methodologies which stepwise refine the controller design.

The Buck converter control design analysis carried out in section 6.7 confirms that the robustness of several designs can be reliably characterized in terms of the mixed sensitivity robustness index ϕ_R. Moreover, the proposed index is well suited for the fine tuning of nonlinear control systems. As only few parameters and constraints are necessary for guiding the design, the proposed method is well suited for automatic robust controller synthesis.

A weak point of the mixed sensitivity robustness index ϕ_R is that the statistical computation approach is very time demanding. For this reason, the next chapter investigates range arithmetic based behavior computation methods for uncertain analog systems.

Taking into account that the proposed robust design method leads to very satisfactory results and provides a notable speed-up and reliability compared to heuristic methods (which are often proposed to cope with the control of complex systems), the contribution of this chapter is significant from a practical point of view.

Chapter 7
Analysis and Verification of AMS Systems

Range arithmetic simulation is becoming popular for the inclusion of parameter uncertainties in the analysis and verification of analog system designs. The accuracy and efficiency of current interval and affine arithmetic based circuit response computation methods limit the simulation of uncertain analog systems. The uncertainty analysis of large analog systems requires faster and more accurate computation methods. This chapter presents a novel range arithmetic method called *orthogonal interval arithmetic* that keeps the correlation between parameters in vectorial form and extends the operational computation methods based on orthogonal signals to enable the fast and direct evaluation of the analog system's behavior under parameter variations. In order to carry out faster semi-symbolic robust design optimizations in the time domain, a novel operational method that computes the multiplication of signal expansions is proposed. Thus, the evaluation of signals at many time points for the computation of performance indexes is avoided during the control system design optimization.

This chapter is organized as follows. After the presentation of the research motivation and related work in section 7.1 and 7.2 respectively, the contributions of this chapter are summarized in section 7.3. The existing range arithmetic methods for parameter uncertainty modeling are briefly reviewed in section 7.4. Section 7.5 presents the *orthogonal interval arithmetic*. Section 7.6 introduces the application of the operational methods for time domain analysis, verification and robustness evaluation of analog systems. The novel method for the operational computation of performance indexes is derived in section 7.6.3 and the methodology proposed for robust control design optimization in the time domain is described in section 7.7. The simulation environment developed for design and analysis of range computation methods is briefly described in section 7.8. The evaluation of the accuracy, performance and stability of the presented methods for tolerance analysis is carried out in section 7.9. Finally, section 7.10 summarizes the contribution and the most relevant findings of the research activities.

7.1 Motivation

Semiconductor device scaling allows a higher circuit density and faster devices in electronic chips. This leads to a functionality increase and performance improvement but at the same time, the circuit behavior is affected by processes variations the more the devices are scaled down [10]. Unknown parameters may have a significant impact on the analog circuit performance. Therefore, the inclusion of parameter tolerances in the analysis and verification of analog circuits is a requisite for reliable system design. As measurement based design verification increases the product cost and time-to-market, there is a growing demand for simulation tools that include uncertainty parameters in the system design verification [24]. The behavior verification of large circuits under uncertainty variations is currently a challenge. Monte-Carlo statistical analysis and corner case analysis are utilized to estimate the impact of manufacturing variations on circuit behavior. Circuit simulators such as PSpice provide tools for tolerance analysis based on the Monte-Carlo (MC) method. It represents parameter variations as random processes. Due to the fact that a large number of simulation runs are necessary to obtain accurate results, the simulation time of the MC based tolerance analysis becomes prohibitive for large circuits. Furthermore, both tolerance analysis methods do not guarantee the worst case coverage.

7.2 Related Work

Interval arithmetic methods were introduced for the efficient tolerance analysis of linear [63] [62] and nonlinear electrical circuits [61] [59] [60] [58]. Parameter uncertainty modeling based on affine arithmetic was proposed by Heupke et al. for fast tolerance analysis of analog system designs [48]. This approach allows to keep parameter correlation during system behavior computation and can achieve more accurate results than interval arithmetic. Grimm et al. show that the affine arithmetic based semi-symbolic computations are well suited for coping with the simulation of linear control and signal processing systems [43].

Grabowski et al. extended the Backward Euler multi-step integration method for affine arithmetic based circuit parameter tolerance analysis [40]. They prove that the well-known Newton-Raphson iterative

method for the solution of nonlinear differential algebraic equations can be utilized to compute the response of slightly nonlinear uncertain circuits.

In order to reduce the over-approximation error introduced in non-affine computation operations, Grabowski et al. proposed an extension of affine arithmetic called "quadratic arithmetic" [41]. The observed accuracy improvement was moderate compared to affine arithmetic and the additional terms introduced for the representation of nonlinear uncertainty significantly increased the computation time. A linear dependence between the computation time and the number of parameters was reported.

To tackle the performance issues of semi-symbolic simulations for system level tolerance analysis, Freisfeld et al. modeled circuit equations in a piecewise linear form [31] and solved the resulting Linear Complementary Problem (LCP) by applying the Katzenelson algorithm [57].

Femia et al. [29] utilized averaged state space models to obtain a linear formulation of power electronic circuit equations. They computed the circuit response using a Taylor approximation of the state transition matrix in affine form and utilized genetic algorithms to improve the accuracy of the simulation results.

Although tolerance analysis methods based on range arithmetic computations show a significantly better performance than the Monte Carlo method, over-approximation errors and the large simulation time limit their application for system level analysis. In particular, the accuracy of the range arithmetic methods for circuit tolerance analysis in the frequency domain is strongly limited due to the nonlinear characteristics of circuit transfer functions [25]. Moreover, the accuracy and stability of circuit response computation methods for time domain tolerance analysis were not rigorously investigated. This is an important research topic because the desired system properties are usually specified in the time domain. Furthermore, computation methods that enable the fast analysis of large systems in the time domain are needed to cope with robust design optimization problems.

7.3 Contribution to Uncertain Analog Circuit Analysis

The aim of the research work presented in this chapter was the investigation of the operational computation methods' accuracy and stability properties to carry out range arithmetic based time domain tolerance analysis of linear and nonlinear analog circuits as well as to develop range arithmetic computation methods that reduce the accuracy problems in the frequency domain circuit tolerance analysis. The main research activities carried out to achieve these goals were the following:

- To investigate the limitations of several range arithmetic methods.
- To derive more accurate range computation methods.
- To analyze the sensitivity of the orthogonal signal coefficients to over-approximation errors in range arithmetic computations.
- To examine the stability properties of the range arithmetic based operational computation methods.
- To compare the accuracy and performance properties of several computation methods using range arithmetic computations.
- To develop computational methods for the fast system design robustness and performance analysis.

In order to carry out fast robust analog circuit design optimizations at several abstraction levels, a *parameter sensitivity index* was defined and formulated in terms of uncertain orthogonal signals. Furthermore, a novel method for the *operational multiplication* of polynomial orthogonal signal expansions was derived, which enables the direct computation (using only signal coefficients) of common system performance indexes (nonlinear cost functions) and leads therefore to a considerable improvement of the simulation performance during design optimization.

Moreover, this dissertation proposes a novel range arithmetic modeling and computation method that carries out *linearization in a box* for keeping correlated non-linear terms in range arithmetic computation chains which reduces over-approximation errors.

As both the proposed range arithmetic and the uncertain circuit behavior computation methods reduce the approximation error, performance and stability limitations of existing range arithmetic based computation and simulation approaches, they constitute an important contribution to the field of analog circuit tolerance analysis and verification.

7.4 Parameter Uncertainty Modeling

Parameter uncertainty in analog systems caused by tolerances and variations in the environmental conditions can be efficiently modeled using bounded intervals [48]. Modeling systems with range arithmetic means to replace system variables and parameters by expressions in range form. Interval and affine arithmetic are widely utilized for range computations. The following sections explain their properties and limitations.

7.4.1 Interval Arithmetic

Interval arithmetic allows the computation of the system behavior range. An *interval variable* $\bar{x}$ is defined by its lower x_{lb} and upper x_{ub} bound as follows:

$$\bar{x} = [x_{lb}\,, x_{ub}] = \{x \,|\, x_{lb} \leq x \leq x_{ub}\}\,, \quad x \in \mathbb{R} \tag{7.1}$$

For convenience, real value variables are considered in this chapter. The four basic interval arithmetic operations are defined in terms of the interval bounds [3]:

$$\bar{x} + \bar{y} = [x_{lb} + y_{lb}\,, x_{ub} + y_{ub}] \tag{7.2a}$$

$$\bar{x} - \bar{y} = [x_{lb} - y_{ub}\,, x_{ub} - y_{lb}] \tag{7.2b}$$

$$\begin{aligned}\bar{x} \cdot \bar{y} = [&min(x_{lb} \cdot y_{lb}, x_{lb} \cdot y_{ub}, x_{ub} \cdot y_{lb}, x_{ub} \cdot y_{ub})\,,\\ &max(x_{lb} \cdot y_{lb}, x_{lb} \cdot y_{ub}, x_{ub} \cdot y_{lb}, x_{ub} \cdot y_{ub})]\end{aligned} \tag{7.2c}$$

$$\bar{x}/\bar{y} = [x_{lb}\,, x_{ub}] \cdot [1/y_{ub}\,, 1/y_{lb}]\,, 0 \notin [y_{lb}\,, y_{ub}] \tag{7.2d}$$

The *range* or *width* w of an interval variable is given by:

$$w(\bar{x}) = x_{ub} - x_{lb} \tag{7.3}$$

and the *absolute value* $|\bar{x}|$ is given by:

$$|\bar{x}| = max(|x_{ub}|, |x_{lb}|) \tag{7.4}$$

Other useful properties of an interval variable are the *center* or *midpoint* m defined as follows:

$$m(\bar{x}) = \frac{x_{lb} + x_{ub}}{2} \tag{7.5}$$

and the *radius* r defined as follows:

$$r(\bar{x}) = \frac{x_{ub} - x_{lb}}{2} = \frac{1}{2} \cdot w(\bar{x}) \tag{7.6}$$

Any interval variable $\bar{x}$ can be represented in *centered interval form* by the following expression:

$$\bar{x} = [m(\bar{x}), m(\bar{x})] + [-r(\bar{x}), r(\bar{x})] = m(\bar{x}) \cdot [1, 1] + r(\bar{x}) \cdot [-1, 1] \tag{7.7}$$

Fig. 7.1 shows the centered interval representation.

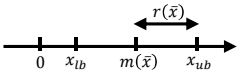


Fig. 7.1: Center and radius of interval variables

The main weakness of interval arithmetic is the loss of correlation in the arithmetic operations. Interval variables are defined as independent intervals which leads to over-estimation of the interval bounds if the variables are related. For example, the subtraction of the same interval variable is different from zero.

Example 7.1. Consider $\bar{x} = [1, 3]$, then $\bar{x} - \bar{x} = [-2, 2] \neq [0, 0]$.

This problem affects all interval arithmetic operations and leads to a bound error explosion in a long computation chain. In such case, the total relative accuracy of the interval bounds tends to be the product of the relative accuracies of each single interval operation in the chain [110]. For this reason, interval arithmetic is not really well suited for the behavior computation of uncertain analog circuits.

7.4.2 *Affine Arithmetic*

In order to overcome the limitations of the interval arithmetic, Stolfi et al. proposed a range arithmetic form called affine arithmetic [110]. An *affine variable* is defined as follows [43]:

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i \cdot \epsilon_i, \quad \epsilon_i \in [-1, 1] \tag{7.8}$$

where x_0 is the *nominal* or *central variable value* and the values x_i, called *noise symbols*, represent the variation of the variable value due to the tolerance of the i^{th} system parameter. The *symbolic real variables* ϵ_i are introduced to capture correlated parameter variations. This *first-degree polynomial representation* is better suited for parameter tolerance modeling in analog systems than interval arithmetic.

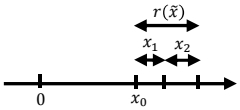


Fig. 7.2: Center and radius of affine variables

As shown in Fig. 7.2, each value x_i contributes to the radius r of an affine variable. It is given by the formula:

$$r(\hat{x}) = \sum_{i=1}^{n} |x_i| \tag{7.9}$$

The *affine operations* are defined as follows [43]:

$$\hat{x} \pm \hat{y} = (x_0 \pm y_0) + \sum_{i=1}^{n} (x_i \pm y_i) \cdot \epsilon_i, \quad \epsilon_i \in [-1, 1] \tag{7.10a}$$

$$c \cdot \hat{x} = c \cdot x_0 + \sum_{i=1}^{n} c \cdot x_i \cdot \epsilon_i, \quad \epsilon_i \in [-1, 1] \tag{7.10b}$$

The symbolic variables ϵ_i allow to keep the the correlation between uncertain parameters. The subtraction of the same affine variable is, as expected, zero.

Example 7.2. Consider $\hat{x} = 2 + 1 \cdot \epsilon_1 \in [1, 3]$, then $\hat{x} - \hat{x} = 0 + 0 \cdot \epsilon_1 \in [0, 0]$. The correct result is obtained using affine arithmetic.

Note that any affine variable can be easily converted to the interval form as follows:

$$\bar{x} = [x_0 - r(\hat{x}), x_0 + r(\hat{x})] \tag{7.11}$$

7.4.2.1 Non-Affine Operations

The nonlinear mathematical operations, such as multiplication, division, square root, etc., are approximated by affine expressions.

The multiplication of two affine variables $\hat{x}$ and $\hat{y}$ is defined as follows:

$$\hat{x} \cdot \hat{y} \approx x_0 \cdot y_0 + \sum_{i=1}^{n} (x_0 \cdot y_i + x_i \cdot y_0) \cdot \epsilon_i + z_{n+1} \cdot \epsilon_{n+1} \tag{7.12a}$$

$$z_{n+1} \coloneqq \Big(\sum_{i=1}^{n} |x_i|\Big) \cdot \Big(\sum_{i=1}^{n} |y_i|\Big) \tag{7.12b}$$

Note that the residual terms in the affine arithmetic multiplication are bound using the following inequality:

$$\Big(\sum_{i=1}^{n} x_i \cdot \epsilon_i\Big) \cdot \Big(\sum_{i=1}^{n} y_i \cdot \epsilon_i\Big) \leq \Big(\sum_{i=1}^{n} |x_i|\Big) \cdot \Big(\sum_{i=1}^{n} |y_i|\Big) \cdot \epsilon_{n+1} \tag{7.13}$$

The cancellation of common symbols in the multiplication of two affine variables may produce more accurate results, than the interval multiplication. It is shown in the following example.

Example 7.3. Consider the multiplication of $\hat{x} = 10 + 2 \cdot \epsilon_1 + 1 \cdot \epsilon_2 \in [7, 13]$ and $\hat{y} = 10 - 2 \cdot \epsilon_1 + 1 \cdot \epsilon_3 \in [7, 13]$.

$\hat{x} \cdot \hat{y} = 100 + 10 \cdot \epsilon_2 + 10 \cdot \epsilon_3 + (2 \cdot \epsilon_1 + \epsilon_2) \cdot (-2 \cdot \epsilon_1 + \epsilon_3)$

$\hat{x} \cdot \hat{y} \simeq 100 + 10 \cdot \epsilon_2 + 10 \cdot \epsilon_3 + 9 \cdot \epsilon_4 \in [71, 129]$

Using interval arithmetic, the upper and lower bounds of the multiplication are over-approximated.
$\bar{x} \cdot \bar{y} = [49, 169]$

The division of two affine variables $\hat{x}$ and $\hat{y}$ is transformed into a multiplication by inverting the denominator.

$$\frac{\hat{x}}{\hat{y}} = \hat{x} \cdot \left(\frac{1}{\hat{y}}\right) \tag{7.14}$$

Therefore, a good affine approximation for the function $1/\hat{y}$ is needed.

$$f(\hat{y}) = \frac{1}{\hat{y}} = \frac{1}{y_0 + \sum_{i=1}^{n} y_i \cdot \epsilon_i} \tag{7.15}$$

Assuming that $f(\hat{y})$ is a bounded twice differentiable function whose second derivative does not change the sign in the interval $[y_{lb}, y_{ub}]$, then the *minimax affine approximation* is given by:

$$f(\hat{y}) \approx \alpha \cdot (y_0 + \sum_{i=1}^{n} y_i \cdot \epsilon_i) + \zeta + \delta \cdot \epsilon_{n+i} \tag{7.16}$$

The resulting affine variable is proportional to the input variable. Thus, this approximation requires a minimum number of parameters.

Note that the decomposition of the affine variable division in two non-affine operations generates two additional independent symbols.

There are several algorithms for computing the coefficients α, ζ and δ. The *Chebyshev approximation*, proposed in [30] for the computation of $f(\hat{y})$, minimizes the error across the affine interval. This approximation is optimal in the sense that it minimizes the area of the polytope defined by the affine terms. The coefficient α is set to be the slope of the line $r(y)$ that interpolates the bounds of $f(\hat{y})$.

$$\alpha = \frac{f(y_{ub}) - f(y_{lb})}{y_{ub} - y_{lb}} \tag{7.17}$$

The independent term ζ is computed as follows:

$$\zeta = \frac{f(y_\alpha) + r(y_\alpha)}{2} - \alpha \cdot y_\alpha \tag{7.18a}$$

$$r(y_\alpha) = \alpha \cdot (y_\alpha - y_{lb}) + f(y_{lb}) \tag{7.18b}$$

$$\frac{d}{dy} f(y_\alpha) = \frac{\text{-}1}{y^2}\bigg|_{y_\alpha} = \alpha \Rightarrow y_\alpha = \sqrt{\frac{y_{lb} - y_{ub}}{f(y_{ub}) - f(y_{lb})}} \tag{7.18c}$$

The *maximum absolute error* δ occurs twice at the end points of the affine variable interval, y_{lb} and y_{ub}. Therefore, it can be computed as follows:

$$\delta = \frac{1}{2} \cdot |f(y_\alpha) - r(y_\alpha)| \tag{7.19}$$

Fig. 7.3a shows the geometrical interpretation of the *Chebyshev approximation*. The terms ζ and $2 \cdot \delta$ correspond to the center and width of the polytope defined by the outer parallel lines with slope α. This approximation encloses the function $f(\hat{y})$ with a minimum error, but introduce an over-estimation of the lower function bound $f(\hat{y}_{ub})$ for the inversion operation. The magnitude of this bound error is twice the value of the error term δ (which corresponds to the polytope width).

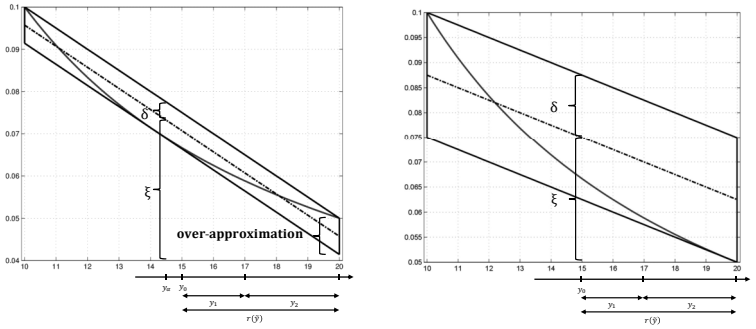
In order to exactly compute the range of the enclosed function, the *minimum range approximation* is often utilized for linearization in non-affine operations [24]. It based on the slope α of the tangent line at the end-point y_{ub} of the interval. This method is easier to compute than the Chebyshev approximation. The slope α is obtained by evaluating the derivative of $f(\hat{y})$ at the end-point y_{ub} of the range variable interval.

$$\alpha = \frac{d}{dy} f(y_{ub}) = \frac{\text{-}1}{y^2}\bigg|_{y_{ub}} = \frac{\text{-}1}{y_{ub}^2} \tag{7.20}$$

The *independent term* ζ and the *maximum absolute error* δ are computed as follows:

$$\zeta = \frac{1}{2} \cdot \left(\frac{1}{y_{lb}} + \frac{1}{y_{ub}} - \alpha \cdot (y_{lb} + y_{ub}) \right) \tag{7.21a}$$

$$\delta = \frac{1}{2} \cdot \left(\frac{1}{y_{lb}} - \frac{1}{y_{ub}} + \alpha \cdot (y_{ub} - y_{lb}) \right) \tag{7.21b}$$



(a) (b)

Fig. 7.3: Affine variable inversion: a) Chebyshev approximation b) Minimum range approximation

Example 7.4. Consider the inversion of $\hat{y} = 15 + 2 \cdot \epsilon_1 + 3 \cdot \epsilon_2 \in [10, 20]$. Using the Chebyshev approximation, the slope α of the line $r(y)$ that interpolates the bounds of the affine variable $\hat{y}$ is $\alpha = -0.005$. At the interior point $y_\alpha = 14.1421$ the slope of the tangent line is also α. The independent term $\zeta = 0.14571$ is the middle value of both curves at the point y_α. The error term $\delta = 0.0042893$ is also computed using the points $f(y_\alpha)$ and $r(y_\alpha)$.

Using the minimum range approximation, the slope α of $1/\hat{y}$ at the end bound y_{ub} is $\alpha = -0.0025$. The independent term $\zeta = 0.1125$ is the middle value of the bounding lines at the point y_0. The error term $\delta = 0.0125$ is also computed using the bounding lines at the point y_0. The obtained upper and lower bounds are the same as those computed using interval arithmetic.

Note that the relative uncertainty introduced by the error term δ is $100 \cdot \delta / (|y_1| + |y_2|) = 100\%$. This value is considerable greater than 17,16% which is relative uncertainty for the Chebyshev approximation.

Note that the interval arithmetic computation shows a smaller range for the inversion operation than the affine arithmetic.

7.4.3 Limitations of Affine Arithmetic

The conservative approximation of the higher order terms in Eq. (7.13) leads to over-approximation errors in the multiplication of two affine variables. It is shown in the following example.

Example 7.5. Consider the multiplication of $\hat{x} = 10 + 1 \cdot \epsilon_1 + 2 \cdot \epsilon_2 \in [7, 13]$ and $\hat{y} = 10 + 1 \cdot \epsilon_1 + 3 \cdot \epsilon_3 \in [6, 14]$.

$\hat{x} \cdot \hat{y} = 100 + 20 \cdot \epsilon_1 + 20 \cdot \epsilon_2 + 30 \cdot \epsilon_3 + (1 \cdot \epsilon_1 + 2 \cdot \epsilon_2) \cdot (1 \cdot \epsilon_1 + 3 \cdot \epsilon_3)$

$\hat{x} \cdot \hat{y} \simeq 100 + 20 \cdot \epsilon_1 + 20 \cdot \epsilon_2 + 30 \cdot \epsilon_3 + 12 \cdot \epsilon_4 \in [18, 182]$

Using interval arithmetic, the following upper and lower bounds are obtained: $\bar{x} \cdot \bar{y} = [42, 182]$

Another issue in non-affine operations is that the multiplication and division operations are not reciprocal operations, i. e. $\hat{x} \cdot (1/\hat{x}) \neq 1$.

Example 7.6. Consider the division of $\hat{x} = 15 + 2 \cdot \epsilon_1 + 3 \cdot \epsilon_2 \in [10, 20]$ by itself. Using the Chebyshev approximation for the inversion of $\hat{x}$, the following result is obtained:

$\hat{x}/\hat{x} = (10 + 2 \cdot \epsilon_1 + 3 \cdot \epsilon_2) \cdot (0.070 - 0.01 \cdot \epsilon_1 - 0.015 \cdot \epsilon_2 + 0.0042 \cdot \epsilon_3)$

$\hat{x}/\hat{x} = 1.06 - 0.0085 \cdot \epsilon_1 - 0.012 \cdot \epsilon_2 + 0.064 \cdot \epsilon_3 + 0.14 \cdot \epsilon_4 \in [0.82, 1.29]$

Using the minimum range approximation for the inversion of $\hat{x}$, a larger bound error is obtained:

$\hat{x}/\hat{x} = (10 + 2 \cdot \epsilon_1 + 3 \cdot \epsilon_2) \cdot (0.075 - 0.005 \cdot \epsilon_1 - 0.0075 \cdot \epsilon_2 + 0.0125 \cdot \epsilon_3)$

$\hat{x}/\hat{x} = 1.12 + 0.075 \cdot \epsilon_1 + 0.18 \cdot \epsilon_2 + 0.12 \cdot \epsilon_3 + 0.14 \cdot \epsilon_4 \in [0.62, 1.62]$

Using interval arithmetic, a worst result is obtained:

$\bar{x}/\bar{x} = [10, 20] \cdot [0.05, 0.1] = [0.5, 2]$

The tracking of dependent terms leads to tighter bounds for affine arithmetic computations.

The additional independent symbol ϵ_{n+1}, introduced for enclosing the residual terms of non-affine operations, increases the uncertainty in computation chains (see example 7.6). An other problem is that the number of independent terms grow at each time step during time domain circuit tolerance analysis which leads to prohibitive simulation times.

Appendix B contains a survey of methods that were proposed for coping with the affine arithmetic limitations. The next section presents the most suitable method for circuit tolerance analysis.

7.4.4 Generalized Interval Arithmetic

The *generalized interval arithmetic* was proposed by Hansen to achieve sharper error bounds in the interval solution of a system of nonlinear equations $f([x_i]_N)$ [45]. He represented the N equation variables x_i using the *centered interval form* as follows:

$$\bar{x}_i = \bar{x}_{mi} + \bar{x}_{ri}\,, \quad \bar{x}_{mi} = [m(\bar{x}_i), m(\bar{x}_i)]\,, \quad \bar{x}_{ri} = [-r(\bar{x}_i), r(\bar{x}_i)] \tag{7.22}$$

In order to improve the convergence of the slope method or the interval Newton method [14] for the solution of the interval equation system $f([\bar{x}_i]_N)$, Hansen proposed the computation of a *linear interval enclosure* $L_j([\bar{x}_i]_N)$ that satisfies the following relation [45]:

$$f_j([x_i]_N) \in L_j([\bar{x}_i]_N)\; \forall\, x_i \in \bar{x}_i \tag{7.23a}$$

$$L_j([\bar{x}_i]_N) = \sum_{i=1}^{n} a_{ji} \cdot \bar{x}_{ri} + \bar{b}_j \tag{7.23b}$$

It is assumed that $f([\bar{x}_i]_N)$ can be represented as a *factorable function*, i.e. a recursively defined function relative to a collection of elementary operations (addition, subtraction, multiplication or division) on other factorable functions $f_j([\bar{x}_i]_N)$. The *affine interval function* $L_j([\bar{x}_i]_N)$ is then defined in terms of a linear combination of *symmetric intervals* $\bar{x}_{ri}$, which are multiplied by the scalar coefficients a_{ji}, and an *independent interval* $\bar{b}_j$, that bounds the higher order terms of $f_j([x_i]_N)$. Thus, the function $f_j([x_i]_N)$ is enclosed in a box $[\bar{x}_i]_N$ by the interval function $L_j([\bar{x}_i]_N)$. This enclosure is denominated optimal if the independent interval $\bar{b}_j$ has a minimum width.

Kolev proposed a *linear interval enclosure*, that can be directly applied to the whole interval equation system $f([\bar{x}_i]_N)$ [61] and presented general computation methods for the *elementary operations* with *generalized intervals* [60]. A *generalized interval* $\tilde{x}$ in *normal form* is defined as follows:

$$\tilde{x} = x_0 + \sum_{i=1}^{n} x_i \cdot \bar{\epsilon}_i + x_b \cdot \bar{\epsilon}_b\,, \quad \bar{\epsilon}_i = \bar{\epsilon}_b = [-1\,,\, 1] \tag{7.24}$$

[14] For more details about the solution of nonlinear interval equations see [89]

The scalar value x_0 determines the *interval center* and the coefficients x_i scale the corresponding interval $\bar{\epsilon}_i$. The *independent interval* $\bar{\epsilon}_b$ that bounds the higher order terms in nonlinear interval computations, is explicitly defined in Eq. (7.24).

The generalized intervals proposed by Kolev utilize *unit symmetrical intervals* $\bar{\epsilon}_i$ and the resulting interval arithmetic can be considered as a generalization of the affine arithmetic. There is however a conceptual difference between both types of range arithmetic. The aim of the affine arithmetic is to minimize the volume of the zonotope enclosing a function while the generalized interval arithmetic tries to minimize the width of the enclosing intervals.

In order to minimize the over-estimation error in the multiplication of dependent generalized intervals, Kolev derived optimal multiplication rules [65] [66]. He computed the multiplication of generalized intervals in the following way:

$$\tilde{x} \cdot \tilde{y} = \frac{f([\epsilon_i^{ub}]) + f([\epsilon_i^{lb}])}{2} + \sum_{i=1}^{n} (x_{lb} \cdot y_i + x_i \cdot y_{lb}) \cdot \bar{\epsilon}_i + (|x_{lb}| \cdot y_b + |y_{lb}| \cdot x_b + z_b) \cdot \bar{\epsilon}_b \quad (7.25a)$$

$$z_b = \frac{f([\epsilon_i^{ub}]) - f([\epsilon_i^{lb}])}{2} - \sum_{i=1}^{n} |x_{lb} \cdot y_i + x_i \cdot y_{lb}| - |x_{lb} \cdot y_b| - |y_{lb} \cdot x_b| \quad (7.25b)$$

$$f([\epsilon_i]) = \left(x_0 + \sum_{i=1}^{n} x_i \cdot \bar{\epsilon}_i + x_b \cdot \bar{\epsilon}_{n+1} \right) \cdot \left(y_0 + \sum_{i=1}^{n} y_i \cdot \bar{\epsilon}_i + y_b \cdot \bar{\epsilon}_{n+2} \right) \quad (7.25c)$$

The vectors $[\epsilon_i^{ub}]$ and $[\epsilon_i^{lb}]$ correspond to the value that maximizes and minimizes the function $f([\epsilon_i])$. Note that the coefficients x_b and y_b are considered independent from one another in the function bound's computation. The linear operations for generalized intervals are [113] [114]:

$$\tilde{x} \pm \tilde{y} = (x_0 \pm y_0) + \sum_{i=1}^{n} (x_i \pm y_i) \cdot \bar{\epsilon}_i + (x_b + y_b) \cdot \bar{\epsilon}_b \quad (7.26a)$$

$$c \cdot \tilde{x} = (c \cdot x_0) + \sum_{i=1}^{n} (c \cdot x_i) \cdot \bar{\epsilon}_i + |c| \cdot x_b \cdot \bar{\epsilon}_b \quad (7.26b)$$

$$c \pm \tilde{x} = (c \pm x_0) \pm \sum_{i=1}^{n} x_i \cdot \bar{\epsilon}_i + |x_b| \cdot \bar{\epsilon}_b \quad (7.26c)$$

Note that Messine's first arithmetic form is utilized for the common error term computations (see appendix B).

7.5 Orthogonal Interval Arithmetic

The modeling of tolerances as affine expressions or generalized intervals instead of simple intervals leads to more accurate results, particularly for the behavior computation of feedback systems. However, the additional term generated in nonlinear operations needs to be handled as a *common error term* or be distributed over the existing affine terms in order to avoid a symbol explosion in long chain computations. As the correlation existing between the linear terms and the higher order terms is not properly considered in both cases, an over or under approximation of the computed system behavior bounds takes always place. With the aim of reducing these accuracy issues, this dissertation proposes a novel range arithmetic that represents uncertain parameters in a *vectorial interval space* $\overline{\mathbb{R}}^N$. Thereby, a geometrical interpretation of the interval arithmetic operations becomes possible. An *orthogonal interval vector* $\overset{\perp}{x}$ is defined as the inner product between the coefficient vector $[x_i]_N$ and the intervals vector $[\bar{\epsilon}_i]_N$:

$$\overset{\perp}{x} = [x_i]_N \bullet [\bar{\epsilon}_i]_N = \sum_{i=0}^{N-1} x_i \cdot \bar{\epsilon}_i = [x_i]_N^T \cdot [\bar{\epsilon}_i]_N, \quad \bar{\epsilon}_i = \begin{cases} [1,1] & \text{if } i = 0 \\ [-1,1] & \text{if } i > 0 \end{cases} \tag{7.27}$$

Fig. 7.1 shows the geometrical representation of orthogonal interval variables. Each value x_i is assigned to a dimension in an orthogonal interval space.

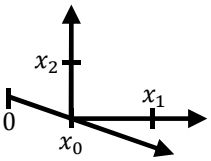


Fig. 7.4: Geometric representation of orthogonal interval variables

The *linear or affine operations* are defined by the vector properties:

$$c \cdot (\overset{\perp}{x} \pm \overset{\perp}{y}) = [c \cdot (x_i \pm y_i)]_N^T \cdot [\bar{\epsilon}_i]_N \tag{7.28}$$

The multiplication of two orthogonal intervals is defined in terms of the outer or Kronecker product as follows:

$$\overset{\perp}{x} \cdot \overset{\perp}{y} = \left([x_i]_N^T \otimes [y_i]_N^T\right) \cdot \left([\bar{\epsilon}_i]_N \otimes [\bar{\epsilon}_i]_N\right) \tag{7.29}$$

In order to avoid the dimension explosion in nonlinear orthogonal interval operations, the concept of *linearization in a box* is applied to Eq. (7.29). Under *monotonicity conditions*, the *linear representation* of the orthogonal interval resulting from Eq. (7.29) is computed as follows:

$$w(\overset{\perp}{x} \cdot \overset{\perp}{y}) = w([p_i + q_i]_N^T \cdot [\bar{\epsilon}_i]_N) \Leftrightarrow \left|\frac{\partial(\overset{\perp}{x} \cdot \overset{\perp}{y})}{\partial \bar{\epsilon}_i}\right| > 0 \,\forall\, x_i, y_i \neq 0 \tag{7.30a}$$

$$p_i = \begin{cases} x_0 \cdot y_0 & \text{if } i = 0 \\ x_0 \cdot y_i + y_0 \cdot x_i & \text{if } i > 0 \end{cases} \tag{7.30b}$$

$$q_i = \begin{cases} \sum\limits_{j>0} (x_j \cdot y_j) \cdot \epsilon_j^m + \sum\limits_{k>j>0} (x_j \cdot y_k + x_k \cdot y_j) \cdot \epsilon_{jk}^m & \text{if } i = 0 \\ |x_i \cdot y_i| \cdot \epsilon_i^c + \frac{1}{2} \cdot \sum\limits_{j \neq i,\, j>0} |x_i \cdot y_j + x_j \cdot y_i| \cdot \epsilon_{ij}^c & \text{if } i > 0 \end{cases} \tag{7.30c}$$

$$\epsilon_j^m = \frac{1}{2} \cdot |\epsilon_j^{ub} - \epsilon_j^{lb}|, \quad \epsilon_{jk}^m = \frac{1}{4} \cdot \left((\epsilon_j^{ub} - \epsilon_j^{lb}) \cdot (\epsilon_k^{ub} - \epsilon_k^{lb})\right) \tag{7.30d}$$

$$\epsilon_i^c = \frac{1}{2} \cdot |\epsilon_i^{ub} + \epsilon_i^{lb}|, \quad \epsilon_{ij}^c = \frac{1}{4} \cdot \left((\epsilon_i^{ub} + \epsilon_i^{lb}) \cdot (\epsilon_j^{ub} + \epsilon_j^{lb})\right) \tag{7.30e}$$

where the operator $w()$ computes the *range between the function bounds in the N-dimensional box*. Otherwise, Eq. (7.25) is applied.

There is a important difference with respect to the *generalized interval arithmetic* proposed by Kolev. The correlation between the quadratic and the linear terms is considered in Eq. (7.30). The novel multiplication rule is derived by seeking the monotonicity conditions associated to the function bounds. If the monotonicity conditions hold, the points ϵ_i^{lb} and ϵ_i^{ub} correspond to the lower and upper multiplication bounds. It reveals that under monotonicity conditions all known affine arithmetic forms always introduce errors in multiplication chains.

The presented methodology based on the concept of linearization in a box and on the consideration of the correlation between linear and higher order terms (vector q_i in Eq. (7.30)) can be applied for the computation of any nonlinear continuous function if the monotonicity conditions hold.

Example 7.7. Consider the following multiplication of orthogonal interval variables: $\hat{\bar{z}} = \hat{\bar{x}} \cdot \hat{\bar{y}} = ([5, 1, 0, -3]_4^T \cdot [\bar{\epsilon}_i]_4) \cdot ([10, -1, 2, 0]_4^T \cdot [\bar{\epsilon}_i]_4)$.

As $\epsilon_i^{ub} = -\epsilon_i^{lb} = [1, 1, -1]$, using Eq. (7.30) the following result is obtained $\hat{\bar{z}} \equiv [54, 5, 10, -30]_4^T \cdot [\bar{\epsilon}_i]_4 \in [9, 99]$.
Using the affine and interval arithmetic the following results are obtained: $\hat{x} \cdot \hat{y} = 50 + 5 \cdot \epsilon_1 + 10 \cdot \epsilon_2 - 30 \cdot \epsilon_3 + 12 \cdot \epsilon_4 \in [-7, 117]$ and $\bar{x} \cdot \bar{y} = [7, 117]$ respectively. The generalized interval arithmetic leads to the following result: $\tilde{x} \cdot \tilde{y} = 54 + 6 \cdot \bar{\epsilon}_1 + 2 \cdot \bar{\epsilon}_2 - 21 \cdot \bar{\epsilon}_3 + 16 \cdot \bar{\epsilon}_b \in [9, 99]$

Note that the range obtained using orthogonal interval arithmetic and generalized interval arithmetic is the same: $w(\hat{\bar{z}}) = 90$ and $w(\tilde{z}) = 90$. It is significantly tighter than the range obtained using the other range arithmetic forms: $w(\hat{z}) = 114$ and $w(\bar{z}) = 110$. The orthogonal interval arithmetic avoids the independent interval $\bar{\epsilon}_b$.

Example 7.8. Consider now the power operation for the orthogonal interval $\hat{\bar{x}} = [0, 3]_2^T \cdot [\bar{\epsilon}_i]_2$. The following result, which has a range $w(\hat{\bar{x}}^2) = 9$, is obtained: $\hat{\bar{x}}^2 = [4.5, 4.5]_2^T \cdot [\bar{\epsilon}_i]_2$. Using generalized interval arithmetic the following result is obtained: $\tilde{x}^2 = 4.5 + 4.5 \cdot \bar{\epsilon}_b$. It has a range ($w(\tilde{x}^2) = 9$). Note that, in both cases, there is no overapproximation but the generalized interval arithmetic produces a new uncorrelated symbol $\bar{\epsilon}_b$.

The division of two orthogonal interval variables $\hat{\bar{x}}$ and $\hat{\bar{y}}$ is transformed into a multiplication by inverting the denominator.

$$\frac{\hat{\bar{x}}}{\hat{\bar{y}}} = \hat{\bar{x}} \cdot \left(\frac{1}{\hat{\bar{y}}}\right) \tag{7.31}$$

To find a linear approximation for the function $1/\hat{\bar{y}}$, the slope method is applied. The center and the radius of the interval function can be computed using the interval bounds y_{lb} and y_{ub}.

$$m\left(\frac{1}{\hat{\bar{y}}}\right) = \frac{1}{2} \cdot \left(\frac{1}{y_{lb}} + \frac{1}{y_{ub}}\right) = \frac{y_0}{y_0^2 - r(\hat{\bar{y}})^2} \tag{7.32}$$

$$r\left(\frac{1}{\hat{\bar{y}}}\right) = \frac{1}{2} \cdot \left(\frac{1}{y_{lb}} - \frac{1}{y_{ub}}\right) = \frac{r(\hat{\bar{y}})}{y_0^2 - r(\hat{\bar{y}})^2} \tag{7.33}$$

Considering Eq. (7.32) and Eq. (7.33), the function $1/\overset{\perp}{y}$ is computed under monotonicity conditions as follows:

$$\frac{1}{\overset{\perp}{y}} = [p_i]_N^T \cdot [\bar{\epsilon}_i]_N\,, \quad p_i = \begin{cases} \frac{y_0}{y_0^2 - r(\overset{\perp}{y})^2} & \text{if } i = 0 \\ \frac{-y_i}{y_0^2 - r(\overset{\perp}{y})^2} & \text{if } i > 0 \end{cases} \tag{7.34}$$

Otherwise, the Chebyshev approximation is utilized for the inversion operation (see section 7.4.2.1).

7.5.1 Considerations for Orthogonal Interval Arithmetic

A relevant property of the orthogonal interval arithmetic is that under monotonicity conditions, the interval bounds of the multiplication and addition operations always correspond to the minimum and maximum function values. This property holds for computation chains and does not depend on the computation order as shown in example 7.9.

Example 7.9. Consider the multiplication of $\overset{\perp}{x} = [10, 1, 2]_3^T \cdot [\bar{\epsilon}_i]_3$, $\overset{\perp}{y} = [10, 1, 3]_3^T \cdot [\bar{\epsilon}_i]_3$ and $\overset{\perp}{z} = [8, 2, 2]_3^T \cdot [\bar{\epsilon}_i]_3$. Although the multiplication order leads to different results, the interval bounds of the several multiplication results are the same and correspond to the function minimum and maximum values (no approximation error).
$\overset{\perp}{w} = (\overset{\perp}{x} \cdot \overset{\perp}{y}) \cdot \overset{\perp}{z} = ([1176, 384, 624]_3^T \cdot [\bar{\epsilon}_i]_3) \in [168, 2184]$
$\overset{\perp}{w} = (\overset{\perp}{x} \cdot \overset{\perp}{z}) \cdot \overset{\perp}{y} = ([1176, 372, 636]_3^T \cdot [\bar{\epsilon}_i]_3) \in [168, 2184]$
$\overset{\perp}{w} = (\overset{\perp}{y} \cdot \overset{\perp}{z}) \cdot \overset{\perp}{x} = ([1176, 376, 632]_3^T \cdot [\bar{\epsilon}_i]_3) \in [168, 2184]$

If there are no monotonicity conditions in any computation of a chain, a proper approximation of the higher order terms is required for safely bounding the computation results. For this reason, this work handles the vector $[q_i]$ generated in the multiplication of two orthogonal interval variables (see Eq. (7.30)) as a common N-dimensional error term. It allows the handling of the linearized higher order terms as an independent interval. In case of a non-monotone multiplication operation this work utilizes the Chebyshev approximation for the computation of the error term which is then properly distributed across the several interval dimensions (see appendix section B.2).

For the addition and subtraction operations, the handling of the *linearized error vector* $[q_i]_N$ as independent term may be necessary if there is an interval overlapping ($y_{lb} \leq x_{lb} \leq y_{ub}$ or $y_{lb} \leq x_{ub} \leq y_{ub}$) as shown in the following example.

Example 7.10. Consider the uncertain function $\overset{\perp}{w} = (\overset{\perp}{x})^2 - \overset{\perp}{x}$ where $\overset{\perp}{x} = [0.5, 0.5]_2^T \cdot [\bar{e}_i]_2$. As the function operands are defined in the same range $[0, 1]$, a cancellation of linear terms (even function terms) takes place after the subtraction which leads to a change of the power term monotonicity conditions. It demands that the odd part of the linearized error vector (which is represented as offset) is handled as a common error term for obtaining correct results. Thus, the cancellation of the odd error part is avoided and the expected result obtained (instead of zero): $\overset{\perp}{w} = ([\text{-}0.125, 0.125]_2^T \cdot [\bar{e}_i]_2) \in [\text{-}0.25, 0]$

After any non-monotone operation, a redistribution of the independent error terms into the orthogonal dimensions is needed. This introduces a small under-approximation error in a computation chain. Furthermore, the linearization of the higher order terms may produce additional errors during the addition or subtraction of power terms.

Taking into account that the linear representation of the uncertain parameters corresponds to the first two terms of a polynomial expansion, modeling the uncertainty behavior of the circuit parameters as orthogonal signals is a natural extension of the methodology defined in this chapter based on the linearization in a box. Thus, the accuracy of the range computations is determined by the number of polynomial coefficients. Moreover, it enables the representation of time-varying parameters and the computation of the time-varying system behavior using operational approaches. The operational matrix of multiplication defined in section 7.6.3 allows the multiplication of such *nonlinear orthogonal interval variables* in matrix form.

The scope of this dissertation is limited to linear uncertain parameters (affine form). It investigates the impact that uncorrelated higher order terms have on the accuracy of range computations for improving analog circuit tolerance analysis. For this reason, the focus of the next sections is the determination of the influence of different behavior computation methods on the circuit tolerance analysis accuracy.

7.6 Analysis and Verification of Uncertain Circuits

In order to analyze and verify the behavior of analog circuit designs under parameter uncertainties, the time domain analog circuit behavior computation methods presented in chapter 5 may by extended for including uncertain parameters which are modeled using range arithmetic. This allows the computation of all possible system responses in a single simulation run.

As mentioned in section 3.4.1, the most commonly utilized method for the formulation of circuit equations is the modified nodal analysis (MNA). It describes an analog circuit by a set of differential algebraic equations (DAEs). Representing the equation variables and parameters in range form, e.g. using orthogonal intervals, circuit simulators such as PSpice or Saber may be extended for computing the impact of parameter tolerances on the circuit behavior [10]. Linear multi-step (LMS) integration methods are able to satisfactorily solve such uncertain circuit equations (see section 7.2). The implicit tradeoff between accuracy and performance of these stepwise integration methods leads however to slow simulations. As the computational efficiency of range arithmetic methods is relatively low once the number of parameters increases [29], more efficient computation methods are necessary for tolerance analysis of large uncertain circuits.

As explained in section 5.4.1, representing the circuit equations in state space form enables the analytical computation of the circuit behavior. The behavior of the uncertain analog circuit which results from introducing range arithmetic variables into the corresponding set of ordinary differential equations (ODEs) requires the evaluation of the uncertain circuit's state transition matrix $[\dot{\Phi}(t)]$ at the desired time points. If the forced system response (integral part in Eq. (5.2)) is expressed in terms of the state transition matrix, the accuracy of the solution does not depend on the time step size. Therefore, large time step sizes can be utilized for the computation of the uncertain circuit behavior. Although the analytical behavior computation method is able to cope with the performance problems in circuit tolerance analysis, the intrinsic nonlinear characteristics of the state transition matrix produces stability and accuracy issues which limit the reliable computation of uncertain analog circuits. Most methods utilized for the reduction of the system matrix norm such as scaling and squaring do not properly work with range arithmetic (due to the error introduced in non-affine operations).

In order to decrease the above mentioned performance, accuracy and stability issues that take place in the range arithmetic based behavior computation methods proposed in the literature for circuit tolerance analysis, the operational computation methods derived in chapter 5 for the fast simulation of analog circuits (see section 5.5) were extended in this dissertation for the response analysis of uncertain analog circuits. As these methods compute the system behavior using simple algebraic mathematical operations, they can be applied at different abstraction levels (transfer function, state space equations, linear and nonlinear circuit equations). Moreover, as only the orthogonal signal coefficients need to be expressed in range form i.e. the orthogonal polynomials matrix $[Q_n(\sigma)]$ is not affected by range operations, the operational behavior computation methods can be efficiently implemented.

7.6.1 Operational Time Domain Robustness Evaluation

In order to evaluate the robustness of analog circuits to parameter uncertainties in the time domain, the upper and lower boundaries of the uncertain circuit response may be utilized. As shown in chapter 6, the area defined by both curves describes the sensitivity of the circuit to parameter variations. Furthermore, it is a good indicator of the circuit design robustness. For this reason, this dissertation proposes the following *parameter sensitivity index* for solving robust design optimization problems in the time domain:

$$\rho_S = \int_{t_0}^{T} ([y_{ub}(t)]_R - [y_{lb}(t)]_R) \cdot dt \tag{7.35}$$

where $[y_{ub}(t)]_R$ and $[y_{lb}(t)]_R$ are the upper and lower system response signal vectors. Opposite to the method proposed by Barke (see section 6.2), Eq. (7.35) does not depend on the system specification. It can be solved by applying any numerical integration method, e.g.the Simpson rule. As the integration of an orthogonal signal can be approximated by a simple matrix multiplication (see chapter 5), the proposed circuit parameter sensitivity index ρ_S is computed in operational form as follows:

$$\rho_S \simeq ([c_{y_{ub}}]_{RP}^T - [c_{y_{lb}}]_{RP}^T) \cdot [P]_{PP} \cdot [Q(\sigma_b)]_P \tag{7.36}$$

where $[c_{y_{ub}}]$ and $[c_{y_{lb}}]$ are the coefficient vectors of the orthogonal signals corresponding to the upper and lower system response respectively. Note that opposite to the classical computation methods, only a single evaluation at the final time point σ_b is necessary for the operational computation of the parameter sensitivity index ρ_S. This leads to a faster evaluation of the analog circuit robustness in the time domain.

As explained in [37], the orthogonal signal representation enables the direct verification of the system behavior on its signal coefficients. Therefore, the operational computation methods are well suited for the simultaneous robust circuit design optimization and verification.

7.6.2 Behavior Verification of Uncertain Analog Circuits

As the *assertion of signal properties* can be directly carried out on their coefficients, the orthogonal signal representation introduced in section 4.5 can be exploited for the efficient behavior verification of analog circuits. Opposite to the stepwise circuit behavior verification, signal evaluations at a large number of time points are not necessary. For signals expanded in terms of Chebyshev polynomials, it is possible to efficiently verify that the circuit response is enclosed within certain safety or performance bounds (behavior constraints). As explained in section 4.7, the value of the corresponding signal expansion is enclosed in an interval which is defined by the relation:

$$c_0 - \sum_{n=1}^{N-1} |c_n| \leq s(\sigma) \leq c_0 + \sum_{n=1}^{N-1} |c_n| \tag{7.37}$$

The possible maximum and minimum values of the signal $s(\sigma)$ are given by its coefficients c_n. Using polynomial root finding methods, bound crossing points may be accurately computed. In order to enable a faster signal bounds verification, the operational method for signal subdivision presented in chapter 4 is applied for the identification of bound crossing free signal intervals as well as for the reduction of the orthogonal polynomials' order. Thus, after finding small intervals that possibly contain bound crossing points, a low order polynomial approximation is used for the analytical computation of the bound crossing points.

Both the nominal circuit behavior as well as the circuit behavior under parameter variations may be verified using the proposed orthogonal interval arithmetic. Other circuit behavior properties such as the settling time are defined in term of signals and may be also directly computed from the coefficients of orthogonal signals.

7.6.3 Operational Computation of Performance Indexes

Although the analytical behavior computation methods enable slightly faster simulations than the operational computation methods for linear analog circuits, it is not possible to take profit from large simulation steps when the overall system design performance needs to be optimized. As common performance indexes are strongly non-linear, small step sizes or a step error control are required for accurate simulation-based performance index computations. Thus, even if the analytical behavior computation based on the state transition matrix may in many cases be successfully carried out for fast tolerance analysis of some analog circuit, its key advantage disappears when the whole analog system is being optimized.

Widely utilized performance indexes for control system design optimization are the *integral of the squared error* (ISE) and the *integral of the time weighted absolute error* (ITAE). They are given by the following expressions:

$$ISE = \int_{t_i}^{T} e^2(t)\,dt \tag{7.38}$$

$$ITAE = \int_{t_i}^{T} t \cdot |e(t)|\,dt \tag{7.39}$$

where $e(t)$ is the *error signal*. The final time T shall be large enough, i.e. the steady state response must be reached.

The time evaluation of the signals involved in the ISE and ITAE index computations can be avoided if the product of two signals is expressed in an operational way. It is derived in the following. As explained in chapter 4.5, any orthogonal polynomial $Q_n(\sigma)$ can be expressed in matrix form as follows:

$$Q_n(\sigma) = [x^n]_N^T \cdot [B_Q]_{N^2} \cdot [c_n]_N \tag{7.40}$$

The basis of the polynomial is represented by a matrix $[B_Q]_{N^2}$ in terms of the power vector $[x^n]_N = [x^N \quad \dots \quad x^2 \quad x \quad 1]$. This orthogonal polynomial formulation allows the generalization of the operational computations for any orthogonal polynomial set. For simplicity, Chebyshev polynomials are considered in the following. Due to the recurrence properties, the multiplication of two Chebyshev polynomials of order i and j may be computed as follows:

$$T_i(\sigma) \cdot T_j(\sigma) = \frac{1}{2} \cdot T_{i+j}(\sigma) + \frac{1}{2} \cdot T_{i-j}(\sigma) \tag{7.41}$$

Taking profit from equation (7.41) and the Kronecker product, this work proposes the computation of the multiplication of two signal expansions in terms of the Chebyshev polynomials coefficients in the following manner:

$$[c_{nc}]_N = [M_Q]_{(2N-1)(N^2)} \cdot ([c_{na}]_N \otimes [c_{nb}]_N) \tag{7.42}$$

Eq. (7.42) defines an operational computation method for the multiplication of two orthogonal signals without calculating the power functions of the polynomial representation. For 2^{nd} order Chebyshev polynomials ($N = 3$), the *operational matrix of multiplication* M_T is given by:

$$[M_T]_{53^2} = \begin{bmatrix} 1 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 1 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 & 0 & \frac{1}{2} & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix} \tag{7.43}$$

Using eq. (7.42), it is possible to compute the performance indexes ISE and ITAE applying only algebraic operations in the following way:

$$ISE \simeq [M_Q]_{(2P-1)(P^2)} \cdot ([c_e]_{RP}^T \otimes [c_e]_{RP}^T) \cdot [P]_P \cdot [Q(\sigma_b)]_{PP} \tag{7.44}$$

$$ITAE \simeq [M_Q]_{(2P-1)(P^2)} \cdot ([c_t]_{RP}^T \otimes [c_{|e|}]_{RP}^T) \cdot [P]_P \cdot [Q(\sigma_b)]_{PP} \tag{7.45}$$

The coefficients $[c_{|e|}]_{RP}$ are fast and accurately computed using the signal subdivision method presented in chapter 4.5.

7.6.4 Operational Computation of Response Overshoot

The overshoot M_p of the step response is related to the frequency response. Decreasing the overshoot introduces a *robustness margin* and avoids the amplification of noise at high frequencies.

The fast computation of the step response overshoot M_p from orthogonal signal coefficients requires the determination of the system response inflexion points (possible maximum values) i.e. the time points at which the response signal derivative becomes zero. The derivative of the system response signal is obtained multiplying its coefficients by the *operational matrix of derivation* $[D]_{PP}$ as follows:

$$\frac{d}{dt}[\hat{y}(t)]_N \simeq [\hat{c_y}]_{PN}^T \cdot [D]_{PP} \cdot [Q(\sigma)]_P \tag{7.46}$$

For 3^{rd} order Chebyshev polynomials, $[D]_{PP}$ is given by [112]:

$$[D]_{4^2} = \begin{bmatrix} 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 0 \\ 0\ 4\ 0\ 0 \\ 3\ 0\ 6\ 0 \end{bmatrix} \tag{7.47}$$

After transforming the orthogonal polynomials' coefficients obtained using Eq. (7.20) into the corresponding power series coefficients, the signal inflexion points of the system response are determined from the roots of the power polynomial. Evaluating the system response at the computed inflexion points, the peak value y_{max}, that is the maximum value of the system response, is found. The *response overshoot* M_p is finally computed as follows:

$$M_p = \frac{y_{max} - y_{ss}}{y_{ss}} \cdot 100 \tag{7.48}$$

where y_{ss} is the steady state response value. It is determined by evaluating the system response signal at the end time point σ_b.

7.7 Time Domain Robust Control Design Refinement

Although the robust design methodology presented in chapter 6 leads to very good results, it is often necessary to refine the system for fulfilling design requirements specified in the time domain. The system properties defined in section 7.6 are often utilized to this end.

In order to obtain a good command signal tracking and load disturbances attenuation, the ISE and ITAE indexes are included in the optimization cost function. The ISE index penalizes large errors in the system response $y(t)$. This reduces the *rise time* T_r. The ITAE index penalizes response errors that persist for a long time. This reduces the *steady state error* e_{ss}. Minimizing the response overshoot M_p, the controller bandwidth is limited which attenuates the measurement noise injected into the system. The parameter sensitivity index ρ_S should be minimized for increasing the system robustness to external perturbations and parameter variations. This work proposes the following cost function for robust control system design optimization in the time domain:

$$f_d = \sqrt{c_1 \cdot ISE^2 + c_2 \cdot ITAE^2 + c_3 \cdot {M_p}^2 + c_4 \cdot {\rho_S}^2} \tag{7.49}$$

where the coefficients c_i are chosen and adjusted to meet the system requirement specification.

Heuristic optimization methods such as *simulated annealing* (SA) or *Particle Swarm Optimization* (PSO) work well for minimizing Eq. (7.49). They usually need a large number of simulation runs for finding optimal controller design parameters. In order to speed up the optimization process, it is divided in two steps. First, the coefficient c_4 in Eq. (7.49) is set to zero and the reduced cost function f_r, that mainly penalizes performance losses, is computed. If the computed cost function value f_r does not improve the system performance, the parameter sensitivity index ρ_S in Eq. (7.49) is replaced by a large value and the *cost function* f_d is computed afterwards. Thus, the time demanding range simulation that is required for computing ρ_S is avoided. Otherwise, the range system simulation is carried out and the cost function f_d is computed.

7.8 Implementation

The symbolic solver presented in chapter 5 was extended for including uncertain parameters in order to analyze the accuracy and stability of the integration methods described in chapter 3 and 5 for range arithmetic simulations. The MATLAB optimization toolbox is utilized in combination with the symbolic solution for determining the upper and lower uncertain response bounds. Thus, a more reliable method than Monte Carlo simulations is applied for achieving an accurate analysis of the experimental results. Moreover, the range arithmetic methods presented in section 7.4 were implemented as MATLAB classes and added to the developed numerical solvers for time domain tolerance analysis.

The SystemC solver implementation presented in chapter 5 was also extended for the computation of the uncertain circuit response using affine arithmetic. The uncertain circuit parameters are modeled using the open source library libaffa. The affine arithmetic class was extended by a common error term for handling symbol explosion.

As the focus of the research was the accuracy and stability analysis of the behavior computation methods, all the experiments were carried out using MATLAB for reducing the analysis and validation effort. The obtained performance results should be considered as an estimation of the possible performance achievement in several programming languages such as C++.

7.9 Experimental Results

In order to evaluate the accuracy improvement that may be achieved with the presented orthogonal interval arithmetic, the uncertainty profile describing the worst case plant perturbation of the Buck converter circuit presented in section 6.7 was computed using common error term affine arithmetic (AA) and generalized interval arithmetic (GIA). The error term generated in non-affine operations was bounded (in all cases) using interval arithmetic for achieving tighter computations. The circuit parameter values are defined in Tab. 6.1. The upper bound computation results for ±10% parameter tolerance are shown in Fig. 7.5. As the frequency response of the Buck converter is directly computed from the circuit transfer function $H(s)$, it is not affected by the discretization and

linear solving (such as the time domain circuit response). Furthermore, the strongly nonlinear characteristics of this function is very demanding for range computations which ensure the proper validation of the proposed orthogonal interval arithmetic. All basic arithmetic operations are required for the computation of its coefficients (see [134] for more details) which are given by:

$$H(s) = \frac{s^2 + a_1 \cdot s + a_0}{s^2 + b_1 \cdot s + b_0}, \quad a_0 = \frac{R \cdot V}{C \cdot L \cdot (R + R_C)}, \quad a_1 = \frac{R \cdot R_C}{L \cdot (R + R_C)} \tag{7.50a}$$

$$b_0 = \frac{R + R_L}{C \cdot L \cdot (R + R_C)}, \quad b_1 = \frac{R \cdot R_C \cdot C + R \cdot R_L \cdot C + R_C \cdot R_L \cdot C + L}{C \cdot L \cdot (R + R_C)} \tag{7.50b}$$

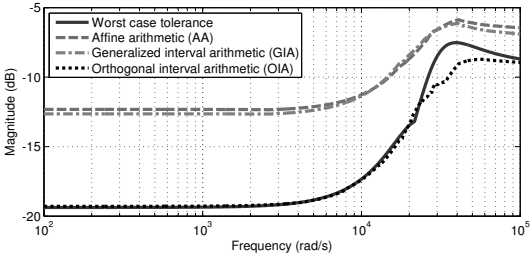


Fig. 7.5: Tolerance analysis upper bound for Buck converter circuit

The over-approximation error introduced by non-affine operations leads to considerably large errors for the affine arithmetic (AA) computations even if the error term is bound using interval arithmetic. The generalized interval arithmetic (GIA) shows a similar accuracy even if Kolev's optimal multiplication rules are applied for minimizing the over approximation error. Considering the correlation of higher order terms, the proposed orthogonal interval arithmetic (OIA) produces a computation result which is really close to the true worst case response (computed using optimization methods). The OIA computation accuracy becomes poor around the response function inflexion point.

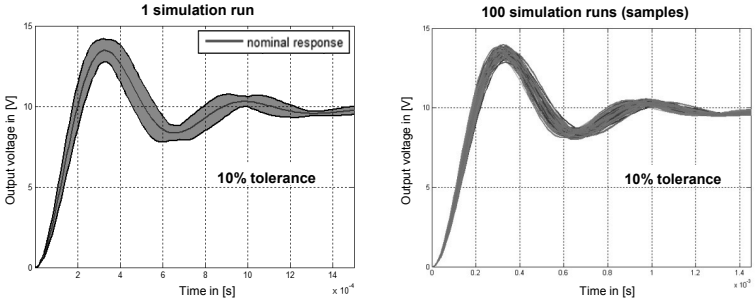
The relative range over-approximation error that the several computation methods produce in the computation of the transfer function coefficients are presented in Tab. 7.1. Although the generalized interval arithmetic (GIA) presents better results than the affine arithmetic (AA), the accuracy of both range computation approaches is poor and shows

a given dependency on the number of computation operations. The larger error occurs in the computation of the coefficient $b1$. The orthogonal interval arithmetic (OIA) computes all circuit coefficients with a relative range error smaller than 10 % and does not show a dependency with the number of operations.

Range arithmetic	**Coefficient**	**Interval bounds**	**Relative range error**
True interval	a_0	$[5.758e^{9}, 1.052e^{10}]$	0.0 %
AA	a_0	$[4.716e^{9}, 1.128e^{10}]$	37.67 %
GIA	a_0	$[4.716e^{9}, 1.110e^{10}]$	33.87 %
OIA	$\mathbf{a_0}$	$\mathbf{[5.555e^{9}, 1.073e^{10}]}$	**8.49** %
True interval	a_1	$[1.191e^{4}, 1.781e^{4}]$	0.0 %
BA	a_1	$[1.091e^{4}, 1.880e^{4}]$	33.73 %
GIA	a_1	$[1.091e^{4}, 1.848e^{4}]$	28.37 %
OIA	$\mathbf{a_1}$	$\mathbf{[1.167e^{4}, 1.815e^{4}]}$	**9.76** %
True interval	b_0	$[5.725e^{8}, 8.672e^{8}]$	0.0 %
AA	b_0	$[5.021e^{8}, 9.131e^{8}]$	39.45 %
GIA	b_0	$[5.021e^{8}, 8.964e^{8}]$	33.78 %
OIA	$\mathbf{b_0}$	$\mathbf{[5.601e^{8}, 8.672e^{8}]}$	**4.21** %
True interval	b_1	$[2.103e^{4}, 2.629e^{4}]$	0.0 %
AA	b_1	$[1.845e^{4}, 2.942e^{4}]$	108.59 %
GIA	b_1	$[1.845e^{4}, 2.837e^{4}]$	88.63 %
OIA	$\mathbf{b_1}$	$\mathbf{[2.103e^{4}, 2.629e^{4}]}$	**0.0** %

Table 7.1: Range arithmetic relative range error

The tolerance analysis in the time domain is less demanding for range computations. Fig. 7.6 shows the time domain uncertain response of the Buck converter circuit computed using bounded affine arithmetic and Monte Carlo simulations for comparison purposes. The parameter tolerances are ±10% (the same tolerance range as before). It validates that for linear circuits, the tolerance analysis carried out using semi-symbolic simulations in a single run is close to the true response bounds.



(a) (b)

Fig. 7.6: Tolerance analysis: a) Affine arithmetic b) Monte Carlo

In order to analyze the impact of the several integration methods investigated in chapter 5 on the accuracy, performance and stability of the range arithmetic based time domain tolerance analysis, the average Buck converter circuit shown in Fig. 5.4 was simulated using orthogonal interval arithmetic for several tolerance values (±5% and ±10%) of the passive components. Tab. 7.2 shows the computation accuracy of the several range arithmetic solver implementations when they are used to compute the area between the upper and lower bounds of the system response (which indicates the sensitivity of the circuit to parameter variations). As the impact of approximation errors was investigated, GIA was utilized (instead of OIA) for all computations.

The computation error increases considerably with the parameter tolerances for all integration methods. The operational computation method using Chebyshev (COM) or Legendre polynomials (LOM) is always more accurate than the LMS and the state transition matrix (STM) integration methods. The operational methods require that the size of the integration interval is kept smaller for achieving convergence to the circuit behavior in range computations. For the proposed circuit, the number of coefficients was reduced to 10 and the integration interval was divided in 5 parts (denoted in brackets in Tab. 7.2). Thus, both a good simulation performance and accuracy was achieved. The performance of the operational methods is also very good for range computations. It is very close to the performance obtained using the analytical method (STM).

Solver	Absolute error (tol=±5%)	Absolute error (tol=±10%)	Execution Time in s
LMS (BE)	$3.0939\,e^{-2}$	$1.0261\,e^{-1}$	30.4203
LMS (BDF2)	$2.7234\,e^{-3}$	$4.5757\,e^{-2}$	40.3107
LMS (TR)	$2.7652\,e^{-3}$	$4.5840\,e^{-2}$	49.6297
STM (B)	$4.3427\,e^{-3}$	$4.3985\,e^{-2}$	10.2291
COM (10/5)	$\mathbf{1.3643\,e^{-4}}$	$\mathbf{2.1686\,e^{-2}}$	**10.6521**
LOM (10/5)	$\mathbf{1.2001\,e^{-4}}$	$\mathbf{2.1892\,e^{-2}}$	**10.9325**

Table 7.2: Accuracy of time domain range computations

In order to validate the Chebyshev polynomials based operational computation method for nonlinear circuit range simulations, the nonlinear circuit shown in Fig. 5.5 was simulated and the simulation results were compared with those obtained applying corner cases simulations. As shown in Fig. 7.7, the operational computation method is also well suited for time domain tolerance analysis of nonlinear circuits.

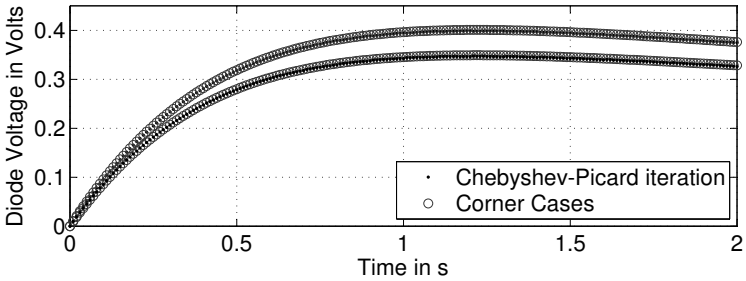


Fig. 7.7: Nonlinear circuit tolerance analysis

Robust Performance Optimization

In order to estimate the performance improvement which may be achieved using the presented operational analysis methods for analog systems, the methodology proposed for robust design refinement in section 7.7 was carried out on a linear control system similar to the Buck converter control presented in chapter 6 (see [36]). An approximated PID controller was initially tuned for obtaining a design that shows a good set-point tracking and disturbance rejection characteristics. The ISE and ITAE indexes were minimized to this end. The controller parameters were then improved using Eq. (7.49). The parameter sensitivity index ρ_S and the response overshoot M_p were properly weighted for considering the impact of parameter variations on the design as well as for improving the closed loop control system stability and robustness. In both cases, simulated annealing was applied. Fig. 7.8 shows the system step response after design optimization (using only the ISE and ITAE indexes) and after robust design optimization (using also the parameter sensitivity index ρ_S).

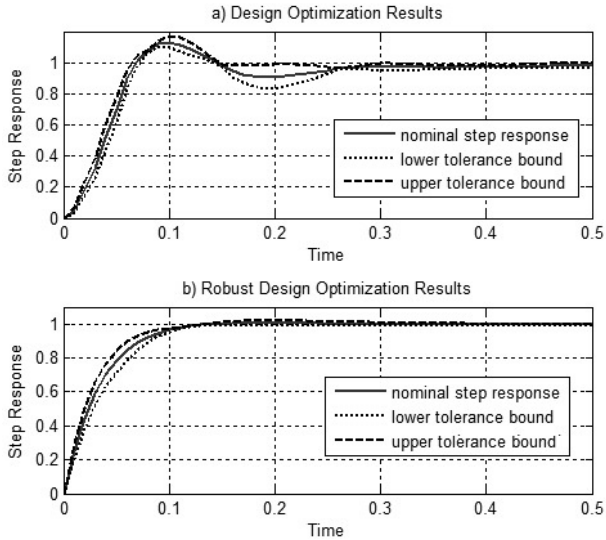


Fig. 7.8: Time domain robust optimization results

After 1000 iterations a very good robust performance design was found. The execution time needed for the controller parameter optimization is presented in Tab. 7.3. The operational method using Chebyshev polynomials (COM) was around 50 times faster than the Trapezoidal Rule (TR) method. Due to stability problems, it was not possible to apply the state transition matrix based behavior computation method in range form.

Solver:	LMS (TR)	COM (35)
Execution time:	11 h. 24 min.	14 min. 34 sec.

Table 7.3: Optimization execution time

7.10 Chapter Summary and Conclusions

This chapter proposed range arithmetic based operational computation methods for the efficient analysis, optimization and verification of uncertain system designs. Furthermore, a novel range computation method that represents uncertain parameters as orthogonal intervals and avoid approximation errors in monotone non-affine computations was presented. The results obtained in the experiments show that:

- Over-approximation errors in range arithmetic computations do not limit the operational behavior computation methods.
- Range arithmetic modifies the stability properties of the operational computation methods (smaller integration intervals are needed).
- The operational behavior computation methods enable more accurate range arithmetic based circuit tolerance analysis than the step-based (LMS) and the analytical (STM) methods.
- The performance of the range arithmetic based operational computation methods is as good as the performance shown by the analytical method for circuit response computations.
- Range arithmetic methods are suitable for the robustness optimization of control system designs.
- The operational computation methods allow a very significant reduction of the robust design optimization time.

The very good accuracy, performance and stability properties together with the possibility of avoiding system response evaluations for system design analysis and verification are important advantages of the operational computation methods. They overcome the performance, accuracy, and stability issues of the LMS and analytical integration methods for range arithmetic based tolerance analysis. The capability of the operational method to carry out nonlinear circuit tolerance analysis using a simple behavior computation algorithm rounds off its advantages leading to a significant contribution to the fast and reliable analog system design analysis and verification.

The proposed orthogonal interval arithmetic shows a high potential for accuracy improving in range arithmetic based circuit tolerance analysis. The experimental results put in evidence that the reduction of the over approximation error in non-affine operations (GIA instead of AA) does not enough improve the accuracy of range arithmetic computation methods for circuit tolerance analysis applications. Furthermore, the loss of higher order terms' correlation in non-affine operations may produce large errors in range arithmetic based circuit behavior computations. Unfortunately, the error term redistribution after non-monotone computations does not enable guaranteed accuracy for the current OIA implementation which limits its application for circuit analysis. The consideration of monotonicity changes in range arithmetic computations is a relevant topic leading to possible future work for minimizing approximation errors in large computation chains.

Although common error term range arithmetic methods always include the worst case uncertainty profile in the frequency domain, this is not true for the dynamic behavior computation in the time domain. For this reason, the extension of the parameter modeling methodology for including orthogonal uncertain signals (see consideration in section 7.5.1) would be an important future contribution for the efficient and accurate uncertain circuit response analysis in the time domain.

Chapter 8
Conclusions and Future Work

Considering that both, system design methodologies and tools, contribute significantly to the reduction of the time-to-market and to the minimization of design errors, the *system design platform model* presented in chapter 2 enables the development of powerful design and analysis tools for AMS systems based on the better understanding of system design principles. This abstract unified view:

1. organizes the essential steps of the model based system design
2. captures the key characteristics of the design activities
3. allows the reasoning about design activities dependencies

Applying a systematical analysis, several aspects were identified and considered in the first and second part of this dissertation to achieve a more efficient and seamless AMS system design chain, among others:

- A careful selection and formal specification of design properties and constraints are essential for guiding the design refinement.
- The system structure should be captured by the modeling approach to enable a more efficient behavior analysis and computation.
- The component behavior representation should enable the efficient computations as well as the simple and direct interaction between heterogeneous components for supporting the fast AMS system simulation at the several stages in the design process.
- A clear and complete specification of the simulation platform as well as the choice and customizing of the methods utilized for behavior computation, mapping between several abstractions or MoCs and optimization of design parameters are necessary to guarantee a high design productivity.
- The automatic computation and estimation of a rich set of metrics which characterize the design performance and implementation characteristics are key tool features needed to define efficient system design refinement and validation methodologies.
- Design tools based on efficient modeling and simulation methods that support several MoCs and levels of abstraction are necessary for the early and complete verification of AMS system designs.

- Methods which include parameter uncertainties in the system design and verification are of central importance for improving the robustness and reliability of AMS system.

8.1 Part I: Efficient System Modeling and Simulation

The MoCs provided by design tools usually present a poor behavior abstraction support for modeling analog components. The available behavior abstractions are too simple or too detailed which make difficult the automatic refinement and the system level design verification. As shown in chapter 6, simplified circuit models do not provide the necessary accuracy for designing robust power systems. For improved low-power embedded system design, chapter 3 introduced a modeling abstraction based on ideal switches which enables the system level simulation of power electronic circuits. This MoC is particularly useful for the modeling of power electronic circuits containing a large number of switches.

The proposed graph based method for finding inconsistencies and predicting electrical impulses after topology switching enables a more efficient and flexible implementation of the switched linear electrical network MoC because it allows the clustering of multiple topologies and it does not require a particular representation of the circuit equations. The evaluation results of the SystemC AMS extension for the simulation of DC-DC power converters at system level encourage the development of new modeling abstractions. Behavior computation methods for switched electrical networks that take advantage of circuit properties and of the inherent repetitive operation mode of power electronic systems could be considered in future work.

The novel signal modeling formalisms proposed in chapter 4 constitute a key contribution to the field of embedded system modeling. As explained in section 2.5.4, the *orthogonal signal model* eliminates the major drawbacks of the existing approaches for modeling and handling heterogeneous embedded systems. As it does not require multiple signal representations for different MoCs, it does not impose restrictions in the way in which heterogeneous MoCs are connected or composed. Both hierarchical and flat heterogeneous models are possible. Furthermore, this mathematical model of signals do not force the mapping

of heterogeneous MoCs into a common MoC, e.g. continuous time or discrete event. As the proposed signal model includes timing (or event order) information, it enables the direct support of different time abstraction levels. Taking advantage of these properties, each AMS system component may be represented with the best suited MoC improving the simulation performance and enabling a better heterogeneous system validation and verification.

The key advantage of the orthogonal signal representation is that continuous time signals are efficiently and accurately described by a finite vector of coefficients. It enables a behavior description accuracy which is near to symbolic expressions while maintaining the simplicity and efficiency of numerical computation methods. As was mentioned in section 1.5, this dissertation proposes a paradigm change in the system behavior modeling which applies the orthogonalization principle to both system signals and parameters for enabling more efficient and accurate behavior computation methods as shown in Fig. 8.1. Taking advantage of the proposed orthogonal signal model demands the development of new behavior computation methods.

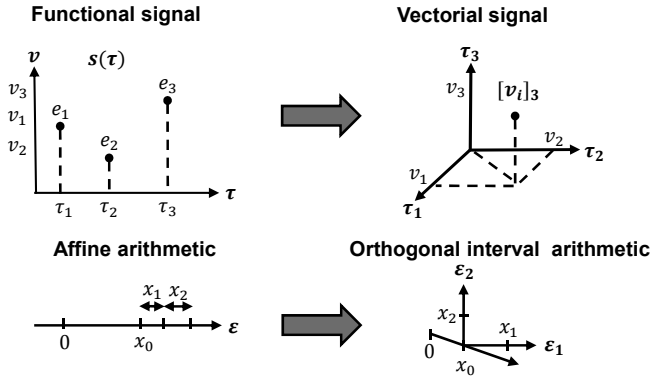


Fig. 8.1: Modeling methods for efficient system behavior computation

The general *operational signal subdivision matrix* presented in section 4.6 is an important contribution that enables the efficient sampling of signals and the fast handling of threshold crossing events. The set of computation methods derived in chapter 5 allow the operational behav-

ior computation at several levels of abstraction for linear and nonlinear analog circuits. As shown by the experimental results, these methods are computationally as accurate as the analytical methods based on the state transition matrix and achieve a similar simulation performance. However, they overcome the practical limitations of the analytical methods. They are directly applicable to any circuit equation representation and do not impose strong restrictions on the input signals (only continuity is necessary). Therefore, the proposed operational computation methods for analog circuits constitute a valuable alternative to the analytical behavior computation methods for speeding up simulations. They present the flexibility of the traditional stepwise behavior computation methods, however are considerably more efficient. Both the integration interval and the number of coefficients may be adjusted for balancing simulation accuracy and performance. As the proposed operational computation methods do not require the linearization of circuit equations to cope with nonlinear circuits, they are well suited for AMS system simulation. They also enable the fast and accurate handling of the interaction between analog and digital circuits. The possibility of extending the analog circuit operational computation methods for a large variety of systems such as time-varying and time-delay systems is an additional relevant advantage.

The state based modeling and computation method for digital systems presented in section 5.6 is the counterpart to the operational computation methods for analog systems and provides the necessary computational properties to enable the efficient simulation of mixed analog and digital systems based on orthogonal signals. This modeling method which is similar to the state space representation of continuous time systems copes efficiently with event sequences and time delays in digital components, enabling efficient process communication (multiple event in one transaction). It contributes thereby to the development of novel system level computation methods such as the proposed iterative computation method for heterogeneous analog and digital circuits. The experimental results corroborate that the proposed behavior computation methods allow a significant improvement of the simulation speed for AMS systems. For the analyzed PLL system which contains several MoCs and abstraction levels, a simulation execution time reduction of 50% was obtained. The simulation speed-up is even more significant if analog signal monitoring is performed for system design analysis and verification purposes (greater than $10x$).

8.2 Part II: Robust System Design and Verification

The proposed set of circuit behavior modeling and computation methods promise a better support for a seamless system level design. As shown in chapter 6, a stepwise analog circuit design refinement from higher to lower abstraction level is necessary for obtaining optimal robust performance designs of AMS systems. Furthermore, a set of suitable metrics are necessary for guiding the design during refinement and exploration. The proposed *mixed sensitivity robustness index* constitutes a significant contribution to this topic because it enables the simple evaluation and comparison of several control system designs. Furthermore, it is applicable to linear and nonlinear analog circuits. This dissertation takes advantage from the orthogonal signal representation to enable the accurate linearization of the nonlinear circuit behavior at different working points (see appendix A). Moreover, several working points and parameter tolerance ranges are characterized by an occurrence probability to avoid that conditions which are unlikely to occur lead to a poor performance or significantly more expensive design (over-conservatism).

As there is always a compromise between the sensitivity to perturbations and the stability properties of a system, designing a system to be robust based on simplified models leads to a poor stability robustness. Modeling accuracy is essential for a correct system robustness evaluation. In particular, system nonlinear characteristics should be properly captured for ensuring a robust design.

The verification of the system design robustness to parameter uncertainty is also necessary. For the efficient computation of the worst-case uncertain circuit behavior, range arithmetic methods were investigated in chapter 7. The experimental results have shown that the affine arithmetic (or generalized interval arithmetic) based operational behavior computation methods are well suited for fast and reliable tolerance analysis of linear and nonlinear analog circuits. They achieve a considerably better accuracy than LMS and analytical integration methods while remaining stable and presenting high computational efficiency.

Range arithmetic behavior computation methods have the drawback that over-approximation errors may unnecessarily introduce larger robustness margins in the system design. The proposed *orthogonal interval arithmetic* considers monotonicity conditions for a proper linearization and handling of higher order terms in the arithmetic operations. It

achieves significantly tighter bounds than affine arithmetic, however a better handling of monotonicity changes is still necessary for avoiding under-approximation errors. This topic will be considered in future work for enabling a range arithmetic based mixed sensitivity index computation.

In order to further exploit the orthogonal signal representation for speeding up system design, a set of operational computation methods were proposed in chapter 7 which enable the faster system design analysis and verification. A novel *operational matrix of multiplication* was presented and utilized for the direct computation from signal coefficients of common control system performance metrics. This was exploited in combination with the previously mentioned range arithmetic based operational behavior computation methods for enabling very fast robust design optimization in the time domain. It is useful for improving the frequency domain system design obtained with the proposed robust design method. Avoiding system response evaluations a speed-up of $50x$ was observed. As both proposed robust system design methods consider manufacturing tolerances and do not impose restrictions on the controller structure, they are helpful for improving circuit design quality and reducing manufacturing costs.

The significant simulation performance improvement for AMS systems as well as the flexibility and stability achieved with the proposed operational computation methods reinforce the value of this dissertation and encourage to continue researching modeling and computation methods that take advantage from the proposed orthogonal signal model. As pointed out in chapter 5, the development of more efficient and stable algebraic methods is mandatory to fully profit from the operational computation methods. In particular, solving poor conditioned equations systems and minimizing rounding errors need to be considered in future work. A very interesting topic for future work is the development of concurrent operational behavior analysis and computation methods which fully exploit the computational power of parallel computer architectures. The intrinsic properties of the orthogonal signals promise novel methods for parallel analog circuit behavior computation which cope with the current scalability issues.

References

[1] H. Al-Aljunaid and T. J. Kazmierski. "SEAMS - a SystemC environment with analog and mixed-signal extensions". In: *2004 IEEE International Symposium on Circuits and Systems*. Vol. 5. 2004, pp. V–281–V–284.

[2] H. Al-Junaid and T. J. Kazmierski. "Analogue and mixed-signal extension to SystemC". In: *IEEE Proceedings - Circuits, Devices and Systems* 152.6 (2005), pp. 682–690.

[3] Goetz Alefeld and Guenter Mayer. "Interval analysis: theory and applications". In: *Journal of Computational and Applied Mathematics* 121.1-2 (2000), pp. 421–464.

[4] J. H. Allmeling and W. P. Hammer. "PLECS-piece-wise linear electrical circuit simulation for Simulink". In: *Power Electronics and Drive Systems, 1999. PEDS '99. Proceedings of the IEEE 1999 International Conference on*. Vol. 1. 1999, 355–360 vol.1.

[5] S. F. Alyaqout, P. Y. Papalambros, and A. G. Ulsoy. "Combined Robust Design and Robust Control of an Electric DC Motor". In: *IEEE/ASME Transactions on Mechatronics* 16.3 (2011), pp. 574–582. ISSN: 1083-4435.

[6] Liliana Andrade et al. "Time Step Control and Threshold Crossing Detection in SystemC AMS 2.0". In: *Actes du huitième colloque du GDR SOC-SIP du CNRS*. CNRS. 2013, p. 3.

[7] K. J. Astrom and B. M. Bernhardsson. "Comparison of Riemann and Lebesgue sampling for first order stochastic systems". In: *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.* Vol. 2. 2002, 2011–2016 vol.2.

[8] Karl Johan Astrom and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, NJ, USA: Princeton University Press, 2010.

[9] S.H. Attarzadeh Niaki et al. "Formal heterogeneous system modeling with SystemC". In: *Specification and Design Languages (FDL), 2012 Forum on*. 2012, pp. 160–167.

[10] Amin Baraka. "USING INTERVAL ARITHMETIC FOR ELECTRONIC CIRCUITS SIMULATION". MA thesis. Cairo: Faculty of Engineering at Cairo University, 2015.

[11] M. Barke et al. "Robustness validation of integrated circuits and systems". In: *Quality Electronic Design (ASQED), 2012 4th Asia Symposium on*. 2012, pp. 145–154.

[12] D. Bedrosian and J. Vlach. "Time-domain analysis of networks with internally controlled switches". In: *1991., IEEE International Sympoisum on Circuits and Systems*. 1991, 846–849 vol.2.

[13] D. Bedrosian and J. Vlach. "Time-domain analysis of networks with internally controlled switches". In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 39.3 (1992), pp. 199–212.

[14] Hans-Georg Beyer and Bernhard Sendhoff. "Robust optimization - A comprehensive survey". In: *Computer Methods in Applied Mechanics and Engineering* 196.33-34 (2007), pp. 3190–3218.

[15] Andrew Butterfield, Gerard Ekembe Ngondi, and Anne Kerr. *A Dictionary of Computer Science*. 7th. Oxford, UK: Oxford University Press, 2016.

[16] Ukrit Chaiya and Somyot Kaitwanidvilai. "Fixed-Structure Robust DC Motor Speed Control". In: *Proceedings of the International Multi Conference of Engineers and Computer Scientists, Hong Kong* (2009).

[17] H. Chung, S. V. Cheong, and A. Ioinovici. "Computer-aided analysis of power electronics converters based on monitoring the internally controlled switches". In: *1993 IEEE International Symposium on Circuits and Systems*. 1993, 2359–2362 vol.4.

[18] A. Cirillo, N. Femia, and G. Spagnuolo. "An interval mathematics approach to tolerance analysis of switching converters". In: *PESC Record. 27th Annual IEEE Power Electronics Specialists Conference*. Vol. 2. 1996, 1349–1355 vol.2.

[19] Olivier Dalle. "Component-based Discrete Event Simulation Using the Fractal Component Model". In: *AI, Simulation and Planning in High Autonomy Systems (AIS)-Conceptual Modeling and Simulation (CMS) Joint Conference*. Buenos Aires, AR, 2007.

[20] Olivier Dalle. "OSA: an Open Component-based Architecture for Discrete-Event Simulation". In: *20th European Conference on Modeling and Simulation (ECMS)*. Bonn, Germany, 2006, pp. 253–259.

[21] Olivier Dalle. "The OSA Project: an Example of Component Based Software Engineering Techniques Applied to Simulation". In: *Proc. of the Summer Computer Simulation Conference (SCSC'07)*. Ed. by H. Vakilzadian. Invited paper. San Diego, CA, USA, 2007, pp. 1155–1162.

[22] Olivier Dalle and Gabriel Wainer. "An Open Issue on Applying Sharing Modeling Patterns in DEVS". In: *Proc. of the DEVS Workshop of the Summer Computer Simulation Conference (SCSC'07)*. San Diego, CA, 2007.

[23] Kalyanmoy Deb and Himanshu Gupta. "Introducing Robustness in Multi-Objective Optimization." In: *Evolutionary Computation* 14.4 (Mar. 21, 2007), pp. 463–494.

[24] T. Ding and Politecnico di Torino. Dipartimento di elettronica e telecomunicazioni. *Worst-case Analysis of Electrical and Electronic Equipment Via Affine Arithmetic: Tesi Di Dottorato*. 2015.

[25] T. Ding et al. "How Affine Arithmetic Helps Beat Uncertainties in Electrical Systems". In: *IEEE Circuits and Systems Magazine* 15.4 (2015), pp. 70–79. ISSN: 1531-636X.

[26] Z. Doulgeri et al. "Robust proportional integral derivative controller tuning with specifications on the infinity-norm of sensitivity functions". In: *IET Control Theory Applications* 1.1 (2007), pp. 263–272.

[27] John Comstock Doyle, Bruce A. Francis, and Allen R. Tannenbaum. *Feedback Control Theory*. Prentice Hall Professional Technical Reference, 1990.

[28] N. Femia and G. Spagnuolo. "Genetic optimization of interval arithmetic-based worst case circuit tolerance analysis". In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 46.12 (1999), pp. 1441–1456.

[29] N. Femia and G. Spagnuolo. “True worst-case circuit tolerance analysis using genetic algorithms and affine arithmetic”. In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 47.9 (2000), pp. 1285–1296.

[30] Luiz Henrique de Figueiredo and Jorge Stolfi. “Affine Arithmetic: Concepts and Applications”. In: *Numerical Algorithms* 37.1 (2004), pp. 147–158.

[31] M. Freisfeld, M. Olbrich, and E. Barke. “Circuit simulations with uncertainties using affine arithmetic and piecewise affine statemodels”. In: *Solid-State and Integrated-Circuit Technology, 2008. ICSICT 2008. 9th International Conference on*. 2008, pp. 496–499.

[32] D. D. Gajski and R. H. Kuhn. “Guest Editors’ Introduction: New VLSI Tools”. In: *Computer* 16.12 (1983), pp. 11–14.

[33] D. Garcia, A. Karimi, and R. Longchamp. *PID CONTROLLER DESIGN WITH SPECIFICATIONS ON THE INFINITY-NORM OF SENSITIVITY FUNCTIONS*. 2005.

[34] D. Garcia, A. Karimi, and R. Longchamp. “Robust proportional integral derivative controller tuning with specifications on the infinity-norm of sensitivity functions”. In: *IET Control Theory Applications* 1.1 (2007), pp. 263–272.

[35] A. Gerstlauer et al. “Electronic System-Level Synthesis Methodologies”. In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 28.10 (2009), pp. 1517–1530.

[36] L. Gil and M. Radetzki. “Optimized Disturbance Weighting for Robust System Design under Parameter Uncertainties”. In: *ANALOG 2016; 15. ITG/GMM-Symposium*. 2016, pp. 1–6.

[37] L. Gil and M. Radetzki. “Orthogonal signal modeling and operational computation of AMS circuits for fast and accurate system simulation”. In: *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2016, pp. 499–504.

[38] L. Gil and M. Radetzki. “SystemC AMS power electronic modeling with ideal instantaneous switches”. In: *Specification and Design Languages (FDL), 2014 Forum on*. Vol. 978-2-9530504-9-3. 2014, pp. 1–8.

[39] A. Girault, Bilung Lee, and E. A. Lee. “Hierarchical finite state machines with multiple concurrency models”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 18.6 (1999), pp. 742–760.

[40] D. Grabowski, C. Grimm, and E. Barke. “Semi-symbolic modeling and simulation of circuits and systems”. In: *2006 IEEE International Symposium on Circuits and Systems*. 2006, 4 pp.–986.

[41] D. Grabowski, M. Olbrich, and E. Barke. “Analog circuit simulation using range arithmetics”. In: *2008 Asia and South Pacific Design Automation Conference*. 2008, pp. 762–767.

[42] C. Grimm, A. Barnasconi M. Vachoux, and K Einwich. *An Introduction to Modeling Embedded Analog/Mixed-Signal Systems using SystemC AMS Extensions*. Tech. rep. Open SystemC Initiative, 2008.

[43] C. Grimm, W. Heupke, and K. Waldschmidt. “Refinement of mixed-signals systems with affine arithmetic”. In: *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*. Vol. 1. 2004, 372–377 Vol.1.

[44] Christoph Grimm et al. “AnalogSL: A C++ - Library for Modeling Analog Power Drivers”. In: *Extended Papers: Best of FDL'01 and HDLCon'01*. Kluwer Academic Press, Apr. 2002.

[45] E. R. Hansen. “A generalized interval arithmetic”. In: *Interval Mathematics: Proceedings of the International Symposium Karlsruhe, West Germany, May 20–24, 1975*. Ed. by Karl Nickel. Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 7–18.

[46] Thomas A Henzinger. “Two challenges in embedded systems design: predictability and robustness”. In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 366.1881 (2008), pp. 3727–3736.

[47] Thomas A. Henzinger and Joseph Sifakis. “The Discipline of Embedded Systems Design”. In: *IEEE Computer* 40.10 (2007), pp. 32–40.

[48] Wilhelm Heupke, Christoph Grimm, and Klaus Waldschmidt. “A New Method for Modeling and Analysis of Accuracy and Tolerances in Mixed-Signal Systems”. In: *In Proceedings of the Forum on Specification and Design Languages (FDL'03*. 2003.

[49] S. Hoelldampf et al. “Efficient generation of analog circuit models for accelerated mixed-signal simulation”. In: *2012 IEEE International SOC Conference*. 2012, pp. 104–109.

[50] S. Hoelldampf et al. “Fast mixed-signal simulation using SystemC”. In: *2011 IEEE International Systems Conference*. 2011, pp. 527–530.

[51] S. Hoelldampf et al. “Using analog circuit behavior to generate SystemC events for an acceleration of mixed-signal simulation”. In: *2011 IEEE 29th International Conference on Computer Design (ICCD)*. 2011, pp. 108–112.

[52] M. H. Hung et al. “A Novel Intelligent Multiobjective Simulated Annealing Algorithm for Designing Robust PID Controllers”. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 38.2 (2008), pp. 319–330.

[53] A. Jantsch and P. Bjureus. “Composite signal flow: a computational model combining events, sampled streams, and vectors”. In: *Proceedings Design, Automation and Test in Europe Conference and Exhibition 2000 (Cat. No. PR00537)*. 2000, pp. 154–160.

[54] A. Jantsch and I. Sander. “Models of computation and languages for embedded system design”. In: *Computers and Digital Techniques, IEE Proceedings* 152.2 (2005), pp. 114–129.

[55] Keiichiro Kashiwagi. “An algorithm to reduce the number of dummy variables in affine arithmetic”. In: Magdeburg, Germany, 2012.

[56] Keiichiro Kashiwagi. “Studies on Numerical Verification of Ordinary Differential Equations Using Affine Arithmetic and Mean Value Form”. PhD thesis. 2013.

[57] J. Katzenelson. “An algorithm for solving nonlinear resistor networks”. In: *The Bell System Technical Journal* 44.8 (1965), pp. 1605–1620.

[58] L. Kolev. “Worst-case tolerance analysis of linear DC and AC electric circuits”. In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 49.12 (2002), pp. 1693–1701.

[59] L. Kolev and V. Mladenov. "Worst-Case Tolerance Analysis of Non-Linear Circuits Using an Interval Method". In: *Proc. of the X International Symp. on Theoretical Electrical Eng.* (1999), pp. 621–623.

[60] L. Kolev and I. Nenov. "A General Interval Method for Tolerance Analysis". In: *Proceedings of the ISTET-2001* (2001), pp. 379–382.

[61] L. V. Kolev. "An efficient interval method for global analysis of non-linear resistive circuits". In: *International Journal of Circuit Theory and Applications* 26.1 (1998), pp. 81–92.

[62] L. V. Kolev. "Approximate solution of a transient tolerance problem for linear circuits". In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 39.8 (1992), pp. 666–673.

[63] L. V. Kolev, V. M. Mladenov, and S. S. Vladov. "Interval mathematics algorithms for tolerance analysis". In: *IEEE Transactions on Circuits and Systems* 35.8 (1988), pp. 967–975.

[64] Lubomir V. Kolev. "An Improved Interval Linearization for Solving Nonlinear Problems". In: *Numerical Algorithms* 37.1 (2004), pp. 213–224. ISSN: 1572-9265.

[65] Lubomir V. Kolev. "New Formulae for Multiplication of Intervals". In: *Reliable Computing* 12.4 (2006), pp. 281–292. ISSN: 1573-1340.

[66] Lubomir V. Kolev. "Optimal Multiplication of G-intervals". In: *Reliable Computing* 13.5 (2007), pp. 399–408. ISSN: 1573-1340.

[67] I.I. Lazaro et al. "Analysis of Time Varying Power System Loads via Chebyshev Polynomials". In: *Electronics, Robotics and Automotive Mechanics Conference, 2008. CERMA '08*. 2008, pp. 332–337.

[68] E. A. Lee. "Computing for embedded systems". In: *IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No.01CH 37188)*. Vol. 3. 2001, pp. 1830–1837.

[69] E.A. Lee and A. Sangiovanni-Vincentelli. "A framework for comparing models of computation". In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 17.12 (1998), pp. 1217–1229.

[70] E.A. Lee and A. Sangiovanni-Vincentelli. "Comparing models of computation". In: *Computer-Aided Design, 1996. ICCAD-96. Digest of Technical Papers., 1996 IEEE/ACM International Conference on*. 1996, pp. 234–241.

[71] Edward A. Lee and Sanjit A. Seshia. *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. Second Edition. http://leeseshia.org, 2015.

[72] H. S. L. Lee et al. "Automated generation of hybrid system models for reachability analysis of nonlinear analog circuits". In: *The 20th Asia and South Pacific Design Automation Conference*. 2015, pp. 725–730.

[73] D. Li and R. Tymerski. "Time-domain simulation of switched networks using the Chebyshev series". In: *Power Electronics Specialists Conference, 1995. PESC '95 Record., 26th Annual IEEE*. Vol. 2. 1995, 823–829 vol.2.

[74] Duwang Li, R. Tymerski, and T. Ninomiya. "Chebyshev series integration method for transient simulation of switched networks". In: *IEEE Transactions on Industrial Electronics* 47.2 (2000), pp. 305–314.

[75] Jie Liu et al. “A hierarchical hybrid system model and its simulation”. In: *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*. Vol. 4. 1999, 3508–3513 vol.4.

[76] A. M. Luciano and A. G. M. Strollo. “A fast time-domain algorithm for the simulation of switching power converters”. In: *IEEE Transactions on Power Electronics* 5.3 (1990), pp. 363–370.

[77] J. D. Ma and R. A. Rutenbar. “Fast interval-valued statistical modeling of interconnect and effective capacitance”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25.4 (2006), pp. 710–724.

[78] James D. Ma and Rob A. Rutenbar. “Fast Interval-valued Statistical Interconnect Modeling and Reduction”. In: *Proceedings of the 2005 International Symposium on Physical Design*. ISPD ’05. San Francisco, California, USA: ACM, 2005, pp. 159–166.

[79] A. Massarini and M. K. Kazmierczuk. “A new representation of Dirac impulses in time-domain computer analysis of networks with ideal switches”. In: *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96*. Vol. 1. 1996, 565–568 vol.1.

[80] A. Massarini and U. Reggiani. “Computer-aided time-domain large-signal analysis of networks with switches”. In: *Proceedings of IEEE International Symposium on Industrial Electronics*. Vol. 2. 1996, 567–572 vol.2.

[81] A. Massarini, U. Reggiani, and M. K. Kazimierczuk. “Analysis of networks with ideal switches by state equations”. In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 44.8 (1997), pp. 692–697.

[82] Frederic Messine. *Extensions of Affine Arithmetic: Application to Unconstrained Global Optimization*. 2002.

[83] Frédéric Messine and Ahmed Touhami. “A General Reliable Quadratic Form: An Extension of Affine Arithmetic”. In: *Reliable Computing* 12 (2006), pp. 171–192.

[84] S. Miyajima. “On the Improvement of the Division of the Affine Arithmetic”. Bachelor thesis. Japan: Kashiwagi Laboratory, Waseda University, 2000.

[85] Shinya Miyajima and Masahide Kashiwagi. “A dividing method utilizing the best multiplication in affine arithmetic”. In: *IEICE Electronics Express* 1.7 (2004), pp. 176–181.

[86] Shinya Miyajima, Takatomi Miyata, and Masahide Kashiwagi. “A new dividing method in affine arithmetic”. In: *IEICE Transactions* E86-A.9 (2003), pp. 2192–2196.

[87] Shinya Miyajima, Takatomi Miyata, and Masahide Kashiwagi. “On the best multiplication of the affine arithmetic”. In: *IEICE Transactions* J86-A.2 (2003), pp. 1150–159.

[88] Cleve Moler and Charles Van Loan. “Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later”. In: *SIAM Review* 45.1 (2003), pp. 3–49.

[89] Ramon E Moore, R Baker Kearfott, and Michael J Cloud. *Introduction to interval analysis*. 2009.

[90] Farid N. Najm. *Circuit Simulation*. Wiley-IEEE Press, 2010.

[91] James Joseph Nutaro. *Parallel discrete event simulation with application to continuous systems*. 2003.

[92] O.A. Palusinski et al. "Accelerated simulation of integrated circuits using Chebyshev series". In: *Circuits and Systems, 1992. ISCAS '92. Proceedings., 1992 IEEE International Symposium on*. Vol. 1. 1992, 89–92 vol.1.

[93] P. Paraskevopoulos. "Legendre series approach to identification and analysis of linear systems". In: *IEEE Transactions on Automatic Control* 30.6 (1985), pp. 585–589.

[94] P.N. Paraskevopoulos. "Chebyshev series approach to system identification, analysis and optimal control". In: *Journal of the Franklin Institute* 316.2 (1983), pp. 135–157. ISSN: 0016-0032.

[95] M. Petkovski, S. Bogdanova, and M. Bogdanov. "A simple adaptive sampling algorithm". In: *14th Telecomunications Forum TELFOR 2006*. Belgrade, Serbia, 2006, pp. 329–332.

[96] Roger Peyret. "Chebyshev method". In: *Spectral Methods for Incompressible Viscous Flow*. New York, NY: Springer New York, 2002, pp. 39–100.

[97] William H. Press et al. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3rd ed. New York, NY, USA: Cambridge University Press, 2007.

[98] Martin Radetzki. "Synthesis of digital circuits from object oriented specifications". PhD thesis. University of Oldenburg, Germany, 2000.

[99] Daniel E. Rivera, Manfred Morari, and Sigurd Skogestad. "Internal model control: PID controller design". In: *Industrial & Engineering Chemistry Process Design and Development* 25.1 (1986), pp. 252–265.

[100] I. Sacevski and J. Veseli. "Introduction to Model Driven Architecture (MDA)". In: (June 2007).

[101] I. Sander and A. Jantsch. "System modeling and transformational design refinement in ForSyDe [formal system design]". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 23.1 (2004), pp. 17–32.

[102] I. Sander, A. Jantsch, and Zhonghai Lu. "Development and application of design transformations in ForSyDe [high level synthesis]". In: *2003 Design, Automation and Test in Europe Conference and Exhibition*. 2003, pp. 364–369.

[103] A. Sangiovanni-Vincentelli. "Is a Unified Methodology for System-Level Design Possible?" In: *IEEE Design Test of Computers* 25.4 (2008), pp. 346–357.

[104] A. Sangiovanni-Vincentelli. "Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design". In: *Proceedings of the IEEE* 95.3 (2007), pp. 467–506. ISSN: 0018-9219.

[105] A. Sangiovanni-Vincentelli and G. Martin. "Platform-based design and software design methodology for embedded systems". In: *IEEE Design Test of Computers* 18.6 (2001), pp. 23–33.

[106] A. Sangiovanni-Vincentelli and P. Nuzzo. *System Level Design: a Platform-Based Approach*. UC Berkeley, 2010.

[107] Luiz Santos et al. "Electronic System Level Design". In: *Electronic System Level Design: An Open-Source Approach*. Ed. by Sandro Rigo, Rodolfo Azevedo, and Luiz Santos. Dordrecht: Springer Netherlands, 2011, pp. 3–10.

[108] Iwona Skalna and Milan Hladík. "A new algorithm for Chebyshev minimum-error multiplication of reduced affine forms". In: *Numerical Algorithms* (2017), pp. 1–22.

[109] Sigurd Skogestad and Ian Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. John Wiley & Sons, 2005.

[110] Jorge Stolfi and Luiz Henrique De Figueiredo. *Self-Validated Numerical Methods and Applications*. Brazil: IMPA: Brazilian Mathematics Colloquium Monograph, 1997.

[111] S. Sumsurooah et al. "Robust Stability Analysis of a DC/DC Buck Converter Under Multiple Parametric Uncertainties". In: *IEEE Transactions on Power Electronics* 33.6 (2018), pp. 5426–5441.

[112] K. C. Tam, S. C. Wong, and C. K. Tse. "Fast analytical approach to finding steady-state waveforms for power electronics circuits using orthogonal polynomial basis functions". In: *2006 IEEE International Symposium on Circuits and Systems*. 2006, 4 pp.–.

[113] Balavelan Thanigaivelan, Tara Julia Hamilton, and Adam Postula. "15th Biennial Computational Techniques and Applications Conference CTAC2010". In: (2010).

[114] Balavelan Thanigaivelan, Tara Julia Hamilton, and Adam Postula. "A comparison of interval methods in symbolic circuit analysis applications". In: *ANZIAM Journal* 52.0 (2012), pp. 1084–1101.

[115] M. Thornton. "Simulation and Implication using a Transfer Function Model for Switching Logic". In: *Computers, IEEE Transactions on* PP.99 (2015), pp. 1–1.

[116] R. Tymerski. "A fast time domain simulator for power electronic systems". In: *Proceedings Eighth Annual Applied Power Electronics Conference and Exposition*, 1993, pp. 477–483.

[117] T. Uhle and K. Einwich. "A SystemCAMS extension for the simulation of non-linear circuits". In: *23rd IEEE International SOC Conference*. 2010, pp. 193–198.

[118] A. Vachoux, C. Grimm, and K. Einwich. "Extending SystemC to support mixed discrete-continuous system modeling and simulation". In: *2005 IEEE International Symposium on Circuits and Systems*. 2005, 5166–5169 Vol. 5.

[119] A. Vachoux, C. Grimm, and K. Einwich. "Towards analog and mixed-signal SOC design with systemC-AMS". In: *Proceedings. DELTA 2004. Second IEEE International Workshop on Electronic Design, Test and Applications*. 2004, pp. 97–102.

[120] Jiérâi Vlach and Kishore Singhal. *Computer Methods for Circuit Analysis and Design*. 2nd. New York, NY, USA: John Wiley & Sons, Inc., 1993.

[121] Xuan-Ha Vu. "Rigorous Solution Techniques for Numerical Constraint Satisfaction Problems". PhD thesis. Switzerland: Swiss Federal Institute of Technology in Lausanne (EPFL), 2005.

[122] W. Wang et al. "Multi-Objective Robust Optimization Using a Postoptimality Sensitivity Analysis Technique: Application to a Wind Turbine Design". In: *Journal of Mechanical Design, American Society of Mechanical Engineers* 137 (2015), 011403–1–011403–11.

[123] Tao Xu et al. "A precise SystemC-AMS model for Charge Pump Phase Lock Loop with multiphase outputs". In: *ASIC, 2009. ASICON '09. IEEE 8th International Conference on*. 2009, pp. 50–53.

[124] J. Yang et al. “Optimized Active Disturbance Rejection Control for DC-DC Buck Converters With Uncertainties Using a Reduced-Order GPI Observer”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 65.2 (2018), pp. 832–841.

[125] M. Yasoobi, A. Khosravi, and A. Lari. “Mixed H_2/H_∞ fixed structure speed control of DC motor using IPSO algorithm”. In: *Control, Instrumentation and Automation (ICCIA), 2011 2nd International Conference on*. 2011, pp. 451–456.

[126] D. Zaum et al. “An accelerated mixed-signal simulation kernel for SystemC”. In: *2010 Forum on Specification Design Languages (FDL 2010)*. 2010, pp. 1–6.

[127] D. Zaum et al. “SystemC mixed-signal and mixed-level simulation using an accelerated analog simulation approach”. In: *2010 XIth International Workshop on Symbolic and Numerical Methods, Modeling and Applications to Circuit Design (SM2ACD)*. 2010, pp. 1–4.

[128] D. Zaum et al. “The PRAISE approach for accelerated transient analysis applied to wire models”. In: *2009 IEEE Behavioral Modeling and Simulation Workshop*. 2009, pp. 120–125.

[129] B. P. Zeigler et al. “DEVS-C++: a high performance modelling and simulation environment”. In: *Proceedings of HICSS-29: 29th Hawaii International Conference on System Sciences*. Vol. 1. 1996, 350–359 vol.1.

[130] B.P. Zeigler. *Theory of Modelling and Simulation*. A Wiley-Interscience Publication. John Wiley, 1976.

[131] L. Zhang et al. “Bridging algorithm and ESL design: Matlab/Simulink model transformation and validation”. In: *Proceedings of the 2013 Forum on specification and Design Languages (FDL)*. 2013, pp. 1–8.

[132] Jianwen Zhu and Nikil Dutt. “{CHAPTER} 5 - Electronic system-level design and high-level synthesis”. In: *Electronic Design Automation*. Ed. by Laung-Terng Wang et al. Boston: Morgan Kaufmann, 2009, pp. 235 –297.

[133] Jun Zhu, I. Sander, and A. Jantsch. “HetMoC: Heterogeneous modelling in SystemC”. In: *Specification Design Languages (FDL 2010), 2010 Forum on*. 2010, pp. 1–6.

[134] Yiyu Zhu. “Robust Dynamic Voltage Scaling Design for Low-Power Embedded Systems”. MA thesis. University of Stuttgart, 2017, p. 86.

[135] Yi Zou et al. “Minimum error based affine arithmetic for variational timing analysis”. In: *2005 6th International Conference on ASIC*. Vol. 2. 2005, pp. 978–981.

Appendix A
Parameter Identification Algorithm

Paraskevopoulos has been introduced a method for the identification of linear system parameters that relies on the operational matrix of integration [94]. Zhu extended this algorithm so that the parameters of a linear system whose transfer function has both zeros and poles can be identified [134].

Assuming that the system to be identified is a single-input single-output linear time-invariant system, it is described by the following differential equation (canonical form):

$$x^{(n)}(\tau)+\alpha_{n-1}\cdot x^{(n-1)}(\tau)+\cdots+\alpha_1\cdot x^{(1)}(\tau)+\alpha_0\cdot x(\tau)=\beta_0\cdot r(\tau)+\beta_1\cdot r^{(1)}(\tau)+\cdots+\beta_m\cdot r^{(m)}(\tau) \tag{A.1}$$

where the parameters $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}, \beta_0, \beta_1, \ldots, \beta_m$ are unknowns and the input signal $r(\tau)$ and output signal $x(\tau)$ are constrained to the time interval $\tau \in [\tau_a, \tau_b]$. As the orthogonal polynomials are defined on the time interval $t \in [\sigma_a, \sigma_b]$, a transformation of the independent variable id needed. After variable mapping, the following equation is obtained:

$$y^{(n)}(\sigma)+a_{n-1}\cdot y^{(n-1)}(\sigma)+\cdots+a_1\cdot y^{(1)}(\sigma)+a_0\cdot y(\sigma)=b_0\cdot u(\sigma)+b_1\cdot u^{(1)}(\sigma)+\cdots+b_m\cdot u^{(m)}(\sigma) \tag{A.2}$$

This differential equation is converted into an integral one through multiple integration (n times on both sides of Eq. (A.2)).

$$\begin{aligned}
&\underbrace{\int_{\sigma_a}^{\sigma_b}\cdots\int_{\sigma_a}^{\sigma_b}}_{n\text{ times}} y^{(n)}(\sigma)\cdot(d\sigma)^n + a_{n-1}\cdot\underbrace{\int_{\sigma_a}^{\sigma_b}\cdots\int_{\sigma_a}^{\sigma_b}}_{n\text{ times}} y^{(n-1)}(\sigma)\cdot(d\sigma)^n+\cdots\\
&+a_0\cdot\underbrace{\int_{\sigma_a}^{\sigma_b}\cdots\int_{\sigma_a}^{\sigma_b}}_{n\text{ times}} y(\sigma)\cdot(d\sigma)^n = b_0\cdot\underbrace{\int_{\sigma_a}^{\sigma_b}\cdots\int_{\sigma_a}^{\sigma_b}}_{n\text{ times}} u(\sigma)\cdot(d\sigma)^n\\
&+b_1\cdot\underbrace{\int_{\sigma_a}^{\sigma_b}\cdots\int_{\sigma_a}^{\sigma_b}}_{n\text{ times}} u^{(1)}(\sigma)\cdot(d\sigma)^n+\cdots+b_m\cdot\underbrace{\int_{\sigma_a}^{\sigma_b}\cdots\int_{\sigma_b}^{\sigma_b}}_{n\text{ times}} u^{(m)}(\sigma)\cdot(d\sigma)^n
\end{aligned} \tag{A.3}$$

Expanding the input and output signals in terms of orthogonal polynomials $Q_n(\sigma)$:

$$y(\sigma) \approx \sum_{n=0}^{N-1} y_n \cdot Q_n(\sigma) = [y_i]_N^T \cdot [Q_n(\sigma)]_N \tag{A.4}$$

$$u(\sigma) \approx \sum_{n=0}^{N-1} u_n \cdot Q_n(\sigma) = [u_i]_N^T \cdot [Q_n(\sigma)]_N \tag{A.5}$$

the following relationship can be applied for the conversion of the integral equation into an algebraic one:

$$\underbrace{\int_{\sigma_a}^{\sigma_b} \cdots \int_{\sigma_a}^{\sigma_b}}_{k \text{ times}} [Q_n(\sigma)]_N \cdot (d\sigma)^k \approx \left([P]_{N^2}\right)^k \cdot [Q_n(\sigma_b)]_N \tag{A.6}$$

where $[P]_{N^2}$ is the $N \times N$ operational matrix of integration.

Replacing the previous expressions (Eq. (A.4) to Eq. (**??**)) into equation (A.3), the following algebraic equation is obtained:

$$\begin{aligned} &[y_i]_N^T + a_{n-1} \cdot [y_i]_N^T \cdot [P]_{N^2} + a_{n-2} \cdot [y_i]_N^T \cdot \left([P]_{N^2}\right)^2 + \cdots + a_1 \cdot [y_i]_N^T \cdot \left([P]_{N^2}\right)^{n-1} \\ &\quad + a_0 \cdot [y_i]_N^T \cdot \left([P]_{N^2}\right)^n + f_0 \cdot [g]_N^T \cdot \left([P]_{N^2}\right)^{n-m} + \cdots + f_{m-1} \cdot [g]_N^T \cdot \left([P]_{N^2}\right)^{n-1} \\ &\quad = b_0 \cdot [u_i]_N^T \cdot \left([P]_{N^2}\right)^n + \cdots + b_m \cdot [u_i]_N^T \cdot \left([P]_{N^2}\right)^{n-m} \\ &\quad e_{n-1} \cdot [g]_N^T \cdot \left([P]_{N^2}\right)^{n-1} + \cdots + e_1 \cdot [g]_N^T \cdot [P]_{N^2} + e_0 \cdot [g]_N^T \end{aligned} \tag{A.7}$$

where $[g]_N^T = [1, 0, 0, \ldots, 0]$ and the coefficients $e_0, e_1, \ldots, e_{n-1}, f_0, \ldots, f_{m-1}$ are given by the initial conditions. Eq. (A.8) is finally expressed as a linear system of R equations with $2 \cdot n + 2 \cdot m + 1$ unknowns[1].

$$[Q]_{N \cdot R} \cdot [\theta]_R = [y_i]_N \tag{A.8}$$

where

$$[\theta]_R^T = [a_{n-1}, a_{n-2}, \ldots, a_0, b_m, \ldots, b_0, e_{n-1}, \ldots, e_0, f_{m-1}, \ldots, f_0]$$

$$\begin{aligned} [Q]_{N \cdot R} = \Big[&-[P]_{N^2}^T \cdot [y_i]_N \Big| -\left([P]_{N^2}^T\right)^2 \cdot [y_i]_N \Big| \ldots \Big| -\left([P]_{N^2}^T\right)^n \cdot [y_i]_N \Big| \left([P]_{N^2}^T\right)^n \cdot [u_i]_N \Big| \ldots \Big| \\ &\left([P]_{N^2}^T\right)^{n-m} \cdot [u_i]_N \Big| \left([P]_{N^2}^T\right)^{n-1} \cdot [g]_N \Big| \ldots \Big| [g]_N \Big| -\left([P]_{N^2}^T\right)^{n-1} \cdot [g]_N \Big| \ldots \Big| -\left([P]_{N^2}^T\right)^{n-m} \cdot [g]_N \Big] \end{aligned}$$

[1] It must be ensured that $R \geq 2 \cdot n + 2 \cdot m + 1$

Appendix B
Survey of Affine Arithmetic Modifications and Extensions

B.1 Modified Affine Arithmetic

In order to reduce the over-estimation error that the dependent terms produce in the affine multiplication, Kolev proposed the following correction [64]:

$$\hat{x} \cdot \hat{y} \approx x_0 \cdot y_0 + \frac{1}{2} \cdot \sum_{i=1}^{n} x_i \cdot y_i + \sum_{i=1}^{n} (x_0 \cdot y_i + x_i \cdot y_0) \cdot \epsilon_i + z_{n+1} \cdot \epsilon_{n+1} \tag{B.1a}$$

$$z_{n+1} = \Big(\sum_{i=1}^{n} |x_i|\Big) \cdot \Big(\sum_{i=1}^{n} |y_i|\Big) - \frac{1}{2} \cdot \sum_{i=1}^{n} |x_i \cdot y_i| \tag{B.1b}$$

Since a new term that it is not multiplied by a symbolic variable is added to the affine multiplication, the center of the interval does not coincide with the nominal multiplication value. Furthermore, the error term z_{n+1} becomes smaller if there are common symbols in the affine operands. Thus the over-estimation of the interval bounds can be significantly reduced for power operations.

In order to consider the over-estimation that is introduced by independent terms in the affine multiplication Vu modified the computation of the *error term* z_{n+1} [121]:

$$z_{n+1} = \frac{1}{2} \cdot \sum_{i=1}^{n} |x_i \cdot y_i| + \sum_{1 \le i,j \le n; i \ne j} |x_i \cdot y_j| \tag{B.2}$$

Miyajima presented a computation of the *error term* z_{n+1} which considers the over-estimation that is introduced by both dependent and independent terms in the affine multiplication [87]:

$$z_{n+1} = \frac{1}{2} \cdot \sum_{i=1}^{n} |x_i \cdot y_i| + \sum_{1 \le i < j \le n} |x_i \cdot y_j + x_j \cdot y_i| \tag{B.3}$$

The following two affine multiplication examples illustrate the impact of the *error term* z_{n+1} on the result accuracy.

Example B.1. Consider the multiplication of $\hat{x} = 10 + 1 \cdot \epsilon_1 + 2 \cdot \epsilon_2 \in [7, 13]$ and $\hat{y} = 10 + 1 \cdot \epsilon_1 - 3 \cdot \epsilon_2 \in [6, 14]$. Since the common affine arithmetic multiplication defined in Eq. (7.12) does not take into account dependencies on the variable terms for the computation of the error term, the following range is obtained:
$\hat{x} \cdot \hat{y} = 100 + 20 \cdot \epsilon_1 - 10 \cdot \epsilon_2 + 12 \cdot \epsilon_3 \in [58, 142]$

The modified affine arithmetic (MAA) shifts the center value to cope with dependencies on the higher order affine terms. Computing the error term as proposed by Kolev or Vu, the following range is obtained: $\hat{x} \cdot \hat{y} = 97.5 + 20 \cdot \epsilon_1 - 10 \cdot \epsilon_2 + 8.5 \cdot \epsilon_3 \in [59, 136]$

Note that Eq. B.1 and Eq. B.2 leads to the same results if there are no independent terms in the multiplication. Computing the error term with the formula proposed by Miyajima, the over-estimation is further reduced: $\hat{x} \cdot \hat{y} = 97.5 + 20 \cdot \epsilon_1 - 10 \cdot \epsilon_2 + 4.5 \cdot \epsilon_3 \in [63, 132]$

Example B.2. Consider the multiplication of $\hat{x} = 10 + 1 \cdot \epsilon_1 + 2 \cdot \epsilon_2 \in [7, 13]$ and $\hat{y} = 10 + 1 \cdot \epsilon_1 - 3 \cdot \epsilon_3 \in [6, 14]$. The terms associated with the symbols ϵ_2 and ϵ_3 are now independent.

Using the common affine arithmetic multiplication, the following range is obtained: $\hat{x} \cdot \hat{y} = 100 + 20 \cdot \epsilon_1 + 20 \cdot \epsilon_2 - 30 \cdot \epsilon_3 + 12 \cdot \epsilon_4 \in [18, 182]$

Note that the independence of the terms x_2 and y_3 increase the computed range, compared to the previous example.

The range improvement obtained with Eq.(B.1) is in this case negligible: $\hat{x} \cdot \hat{y} = 100.5 + 20 \cdot \epsilon_1 + 20 \cdot \epsilon_2 - 30 \cdot \epsilon_3 + 11.5 \cdot \epsilon_4 \in [19, 182]$

Computing the error term with the formulas proposed by Vu and Miyajima, the over-estimation is reduced:
$\hat{x} \cdot \hat{y} = 100.5 + 20 \cdot \epsilon_1 + 20 \cdot \epsilon_2 - 30 \cdot \epsilon_3 + 5.5 \cdot \epsilon_4 \in [25, 176]$

Eq. (B.2) and Eq. (B.3) leads in this case to the same results.

Using interval arithmetic, the following upper and lower bounds are obtained: $\bar{x} \cdot \bar{y} = [42, 182]$
Note that the resulting lower bound after the interval multiplication is larger than the previously computed. Therefore, even if the presented modifications in the affine arithmetic multiplication leads to less over-estimation, the over-approximation error caused by the independent terms is not completely eliminated.

For the division in affine form, Kolev [64] and Miyajima [84] [86] [85] formulated the *Chebyshev approximation* as follows:

$$\alpha = \frac{-1}{y_{lb} \cdot y_{ub}} \tag{B.4a}$$

$$\zeta = \frac{1}{2} \cdot \left(\frac{1}{y_{lb}} + \frac{1}{y_s} - \alpha \cdot (y_{lb} + y_{ub}) \right) \tag{B.4b}$$

$$\delta = \frac{1}{2} \cdot \left(\frac{1}{y_{lb}} - \frac{1}{y_s} - \alpha \cdot (y_{lb} - y_s) \right) \tag{B.4c}$$

$$y_s = \begin{cases} \sqrt{y_{lb} \cdot y_{ub}} & \text{if } y_{lb} > 0 \\ -\sqrt{y_{lb} \cdot y_{ub}} & \text{if } y_{ub} < 0 \end{cases} \tag{B.4d}$$

Eq. (B.4) is valid for positive and negative affine variables.

In order to satisfy the relation $\hat{x}/\hat{x} = 1$ Kolev proposed the following formula for the division of two affine expressions [64]:

$$\frac{\hat{x}}{\hat{y}} = \frac{x_0}{y_0} + \left(\sum_{i=1}^{n} (x_i - \frac{x_0}{y_0} \cdot y_i) \cdot \epsilon_i \right) \cdot \frac{1}{\hat{y}} \tag{B.5}$$

Unfortunately, Eq. (B.5) does not avoid the over-approximation error in the affine division. In order to reduce the over-approximation errors in the affine multiplication and division, Miyajima [85] used the *Chebyshev approximation* to linearize the binomial functions. The resulting affine variable has the following form:

$$f(\hat{x}, \hat{y}) \approx \hat{z} = \alpha \cdot \hat{x} + \beta \cdot \hat{y} + \zeta_{xy} + \delta_{xy} \tag{B.6}$$

The coefficients α and β correspond to the nominal values y_0 (or $1/y_0$ for the division) and x_0 respectively. The *maximum approximation error* δ_{xy} in the domain D, which is the joint range of the range of $\hat{x}$ and the range of $\hat{y}$, is given by:

$$\delta_{xy} = \max_{(x,y) \in D} |f(x, y) - \alpha \cdot x + \beta \cdot y + \zeta_{xy}| \tag{B.7}$$

In order to minimize the *maximum approximation error* δ_{xy}, the best linear approximation in the *joint range d*, which is a $2 \cdot n$ polygon symmetrical with respect to the *nominal point* (x_0, y_0), is calculated as follows:

$$\zeta_{xy} = \frac{d_{max} + d_{min}}{2}, \quad \delta_{xy} = \frac{d_{max} - d_{min}}{2} \tag{B.8a}$$

$$d_{max} = \max_{(x,y)\in D} d(x,y), \quad d_{min} = \min_{(x,y)\in D} d(x,y) \tag{B.8b}$$

$$d(x,y) = f(x,y) - \alpha \cdot x + \beta \cdot y \tag{B.8c}$$

Since this method requires to solve two linear program problems for each non-affine operation, the computation time becomes too large for system simulation.

B.2 Handling of Independent Error Terms

As mentioned in section 7.4.3, the additional symbols generated by each non-affine operation continuously grow during dynamical system simulation. To cope with this issue, Kashiwagi proposed an algorithm to reduce the number of independent terms in affine computations but this approach only mitigates the problem [55] [56]. A more efficient approach to handle the error terms is to distribute the *maximum error term* δ into the linear terms, as proposed by Ma in [78] and [77]. The resulting affine variable $\hat{z}$ in non-affine operations is computed as follows:

$$\hat{z} \approx z_0 + z_i \cdot \left(1 + \frac{\delta}{\sum_{i=1}^{n} |z_i|} \cdot \right) \cdot \epsilon_i \tag{B.9}$$

The distribution of the error term does not modify the affine variable range and avoids the generation of additional independent terms.

Eq. (B.9) distributes the error uniformly, which is not necessarily the best distribution. In order to minimize the maximum error in long computation chains, Zou proposed to distribute the error term into the n existing symbols as follows [135]:

$$\hat{z} \approx z_0 + \sum_{i=1}^{n} \left(z_i + h_i \cdot \delta\right) \cdot \epsilon_i, \quad \sum_{i=1}^{n} h_i = 1 \tag{B.10}$$

The coefficients h_i are found solving the error minimization problem for the whole computation chain, which is computationally very expensive.

B.3 Extensions to Affine Arithmetic

In order to avoid the growth of affine terms, Messine introduced a *common error symbol* ϵ_{n+1} for all multiplication operations in a computation chain [82]. Furthermore, he introduced two *additional common error symbols* $\epsilon_{n+2} \in [0, 1]$ and $\epsilon_{n+3} \in [-1, 0]$ which cope with over-estimation errors in power operations. Thus, an *extended affine variable* $\check{x}$ is defined by:

$$\check{x} = x_0 + \sum_{i=1}^{n} x_i \cdot \epsilon_i + x_{n+1} \cdot \epsilon_{n+1} + x_{n+2} \cdot \epsilon_{n+2} + x_{n+3} \cdot \epsilon_{n+3} \tag{B.11}$$

The upper and lower bounds of the corresponding uncertain interval $[x_{lb}, x_{ub}]$ are therefore given by:

$$x_{ub} = x_0 + \sum_{i=1}^{n} |x_i| + x_{n+1} + x_{n+2} \tag{B.12a}$$

$$x_{lb} = x_0 - \sum_{i=1}^{n} |x_i| - x_{n+1} - x_{n+3} \tag{B.12b}$$

Note that the center of the interval $m(\check{x})$ differs from x_0.

$$m(\check{x}) = x_0 + \frac{x_{n+2} - x_{n+3}}{2} \tag{B.13}$$

The *extended affine operations* are defined as follows:

$$\check{x} \pm \check{y} = (x_0 \pm y_0) + \sum_{i=1}^{n} (x_i \pm y_i) \cdot \epsilon_i + \sum_{j=1}^{3} (x_{n+j} + y_{n+j}) \cdot \epsilon_{n+j} \tag{B.14a}$$

$$c \cdot \check{x} = (c \cdot x_0) + \sum_{i=1}^{n} (c \cdot x_i) \cdot \epsilon_i + \sum_{j=1}^{3} (|c| \cdot x_{n+j}) \cdot \epsilon_{n+j} \tag{B.14b}$$

$$c \pm \check{x} = (c \pm c_0) \pm \sum_{i=1}^{n} x_i \cdot \epsilon_i + \sum_{j=1}^{3} x_{n+j} \cdot \epsilon_{n+j} \tag{B.14c}$$

Note that the values x_{n+1}, x_{n+2} and x_{n+3} in the additional common error terms remain always positive. The multiplication of two extended affine variables is defined as follows:

$$\check{x} \cdot \check{y} = (x_0 \cdot y_0) + \sum_{i=1}^{n} (x_0 \cdot y_i + x_i \cdot y_0) \cdot \epsilon_i + \sum_{j=1}^{3} K_j \cdot \epsilon_{n+j} \tag{B.15a}$$

$$K_1 = |x_0| \cdot y_{n+1} + |y_0| \cdot x_{n+1} + \sum_{i=1}^{n+3} \sum_{j=1;\, j \neq i}^{n+3} |x_i \cdot y_j|, \tag{B.15b}$$

$$K_2 = K_2^0 + \sum_{i=1; x_i \cdot y_i > 0}^{n+3} x_i \cdot y_i, \quad K_3 = K_3^0 + \sum_{i=1; x_i \cdot y_i < 0}^{n+3} |x_i \cdot y_i|$$

$$K_2^0 = \begin{cases} x_0 \cdot y_{n+2} + y_0 \cdot x_{n+2} & \text{if } x_0 > 0 \text{ and } y_0 < 0 \\ x_0 \cdot y_{n+2} - y_0 \cdot x_{n+3} & \text{if } x_0 > 0 \text{ and } y_0 > 0 \\ -x_0 \cdot y_{n+3} + y_0 \cdot x_{n+2} & \text{if } x_0 < 0 \text{ and } y_0 > 0 \\ -x_0 \cdot y_{n+3} - y_0 \cdot x_{n+3} & \text{if } x_0 < 0 \text{ and } y_0 < 0 \end{cases} \tag{B.15c}$$

$$K_3^0 = \begin{cases} x_0 \cdot y_{n+3} + y_0 \cdot x_{n+3} & \text{if } x_0 > 0 \text{ and } y_0 < 0 \\ x_0 \cdot y_{n+3} - y_0 \cdot x_{n+2} & \text{if } x_0 > 0 \text{ and } y_0 > 0 \\ -x_0 \cdot y_{n+3} + y_0 \cdot x_{n+2} & \text{if } x_0 < 0 \text{ and } y_0 > 0 \\ -x_0 \cdot y_{n+2} - y_0 \cdot x_{n+2} & \text{if } x_0 < 0 \text{ and } y_0 < 0 \end{cases} \tag{B.15d}$$

Note that the values K_2^0 and K_3^0 depend on the signs of the nominal values. As shown in the following example, the presented formula improve the accuracy of the affine multiplication when the operands have common affine terms, such as in the power operation case.

Example B.3. Consider the multiplication of $\check{x} = 10 + 1 \cdot \epsilon_1 + 2 \cdot \epsilon_2 + 0 \cdot \epsilon_3 + 0 \cdot \epsilon_4 + 0 \cdot \epsilon_5 \in [7, 13]$ and $\check{y} = 10 + 1 \cdot \epsilon_1 - 3 \cdot \epsilon_2 + 0 \cdot \epsilon_3 + 0 \cdot \epsilon_4 + 0 \cdot \epsilon_5 \in [6, 14]$. Using the common affine arithmetic multiplication a range [58, 142] is obtained. Computing the multiplication using Eq. (B.15) the following extended affine value is obtained:

$\hat{x} \cdot \hat{y} = 100 + 20 \cdot \epsilon_1 - 10 \cdot \epsilon_2 - 0 \cdot \epsilon_3 + 1 \cdot \epsilon_4 + 6 \cdot \epsilon_5 \in [64, 131]$

Note that the common error term is zero if there are no common affine terms in the multiplication. Furthermore, Eq. (B.15) result in a tighter range than the modified affine arithmetic formulas proposed by Kolev, Vu and Miyajima.

The following example considers the cases, in which there are independent affine terms in the multiplication operation.

Example B.4. Consider the multiplication of $\check{x} = 10 + 1 \cdot \epsilon_1 + 2 \cdot \epsilon_2 + 0 \cdot \epsilon_3 + 0 \cdot \epsilon_4 + 0 \cdot \epsilon_5 + 0 \cdot \epsilon_6 \in [7, 13]$ and $\check{y} = 10 + 1 \cdot \epsilon_1 + 0 \cdot \epsilon_2 - 3 \cdot \epsilon_3 + 0 \cdot \epsilon_4 + 0 \cdot \epsilon_5 + 0 \cdot \epsilon_6 \in [6, 14]$. The terms associated with the symbols ϵ_2 and ϵ_3 are now independent. Using the common affine arithmetic multiplication a range [18, 182] is obtained. Computing the multiplication using Eq. (B.15) the following extended affine value is obtained:

$\hat{x} \cdot \hat{y} = 100 + 20 \cdot \epsilon_1 + 20 \cdot \epsilon_2 - 30 \cdot \epsilon_3 + 11 \cdot \epsilon_4 + 1 \cdot \epsilon_5 + 0 \cdot \epsilon_6 \in [19, 182]$

The obtained range improvement is in this case negligible. The formulas proposed by Vu and Miyajima are better to cope with the over-estimation problem for independent affine terms.

In order to improve the accuracy in the multiplication of extended affine forms, for the independent affine term case, Skalna et al. extended Eq. (B.3) to handle error term propagation [108].

$$\check{x} \cdot \check{y} = (x_0 \cdot y_0) + \frac{1}{2} \cdot \sum_{i=1}^{n} x_i \cdot y_i + \sum_{i=1}^{n} (x_0 \cdot y_i + x_i \cdot y_0) \cdot \epsilon_i + z_{n+1} \cdot \epsilon_{n+1} \tag{B.16a}$$

$$\begin{aligned} z_{n+1} = {} & x_{n+1} \cdot y_{n+1} + (|x_0| + \sum_{i=1}^{n} |x_i|) \cdot y_{n+1} + (|y_0| + \sum_{i=1}^{n} |y_i|) \cdot x_{n+1} + \\ & \frac{1}{2} \cdot max\left\{ \sum_{i=1; x_i \cdot y_j > 0}^{n} x_i \cdot y_i + \sum_{i=1; x_i \cdot y_i < 0}^{n} |x_i \cdot y_i| \right\} + \sum_{1 \le i < j \le n} |x_i \cdot y_j + x_j \cdot y_i| \end{aligned} \tag{B.16b}$$

They applied the following property to reduce the positive and negative error terms to a single term that is added to the common error term:

$$K_2 \cdot [0, 1] + K_3 \cdot [-1, 0] \subseteq max\{K_2 + K_3\} \cdot [-1, 1] \tag{B.17a}$$

$$K_2 = \sum_{i=1; x_i \cdot y_j > 0}^{n} x_i \cdot y_i, \quad K_3 = \sum_{i=1; x_i \cdot y_j > 0}^{n} |x_i \cdot y_i| \tag{B.17b}$$

B.4 Quadratic Arithmetic

In order to reduce the over-approximation error in computation chains that contain power operations, Messine also proposed to keep the information about the square terms in non-affine computations [82]. The *general quadratic form* is defined as follows [83]:

$$\breve{\breve{x}} = x_0 + \sum_{i=1}^{n} x_i \cdot \epsilon_i + \sum_{i,j=1}^{n} x_{ij} \cdot \epsilon_i \cdot \epsilon_j + x_{n+1} \cdot \epsilon_{n+1} + x_{n+2} \cdot \epsilon_{n+2} + x_{n+3} \cdot \epsilon_{n+3} \quad \text{(B.18)}$$

Grabowski et al. presented later a *simplified quadratic form*, that introduces a new set of symbolic variables ϵ_{ij} defined as follows [41]:

$$\epsilon_{ij} \in \begin{cases} [0, 1] & \text{if } i = j \\ [\text{-}1, 1] & \text{if } i \neq j \end{cases} \quad \text{(B.19)}$$

For simplicity, they neglected the correspondence between ϵ_i and ϵ_{ij}. Since quadratic arithmetic does not provide a way to compute the exact interval bounds, it only achieves a poor accuracy improvement in computation chains. Therefore, it is not worth the increment of the complexity and computation time for circuit tolerance analysis and verification problems.

B.5 Uncertainty Interval Partitioning (UIP)

In order to reduce the over-approximation error in interval [18] [28] and affine arithmetic [29] based circuit tolerance analysis, Femia et al. proposed the *partitioning of tolerance intervals*. This method divides the uncertainty intervals into smaller sub-intervals and computes the bounds of all possible interval combinations in order to find a tighter range. Since the computational cost exponentially increases with the number of partitions, this method is only suitable for the tolerance analysis of circuits that have a small number of uncertainty parameters. Furthermore, a proper algorithm that selects the number of partitions according to the local non-linearity is required.