Fall 2022

Closed Form Implicitly Integrated Models for Computationally Efficient Simulation of Power Electronics

Andrew Wunderlich

Closed Form Implicitly Integrated Models for Computationally Efficient Simulation of Power Electronics

by

Andrew Wunderlich

Bachelor of Science in Engineering
University of South Carolina, 2017

Submitted in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy in

Electrical Engineering

College of Engineering and Computing

University of South Carolina

2022

Accepted by:

Enrico Santi, Major Professor

Roger Dougal, Committee Member

Adel Nasiri, Committee Member

Jason Bakos, Committee Member

Cheryl L. Addy, Interim Vice Provost and Dean of the Graduate School

DEDICATION

To my parents, Scott and Jenny Wunderlich, who have supported me through every step of my life and have made tremendous sacrifices, both financial and personal, for my education and development.

To my father, whose influence in my life would be difficult to overstate. His guidance—leading me to attend the University of South Carolina, to study electrical engineering and to specialize in power electronics as he did, to pursue a Ph.D., and to diligently work to develop deeper scientific understanding—has profoundly shaped my life. His appreciation for the elegant beauty of God's creation as revealed in the fields of science, mathematics, and engineering has inspired me to follow in his footsteps professionally, personally, and spiritually.

To my mother, who has patiently endured untold hours of tiresome technical discussion between two likeminded electrical engineers without complaint, and who has always selflessly placed the needs of her family above her own.

ACKNOWLEDGEMENTS

The first and most significant acknowledgement is owed to my academic advisor, Dr. Enrico Santi, who has mentored me and provided his expert guidance throughout my entire graduate education. The single most valuable component of that education has been to observe and gradually learn to replicate his approach to engineering and problem solving. His commitment to effectively educating his students in both practical and theoretical concepts is unquestionable. In my estimation, no other faculty member does more to ensure that their students gain valuable understanding and leave as well-qualified and capable leaders in their field.

Additionally, I would like to acknowledge my other committee members—Dr. Adel Nasiri, Dr. Roger Dougal, and Dr. Jason Bakos—for their time and excellent feedback during the development of this dissertation. I also extend my sincere thanks to Jared Cronin for his help running several experiments for this work, and for his friendship and positive attitude working together.

Lastly, I owe a great debt of gratitude to my wife, Shelby, for her kindness, patience, and support throughout my final years of graduate school and during the preparation of this dissertation.

This work was financially supported and made possible by the Office of Naval Research under contracts N00014-20-C-1106 and N00014-22-C-1003, and was approved for public release—Distribution Statement A, DCN# 43-9996-22.

ABSTRACT

This work describes novel closed-form, implicitly integrated (CF-implicit) models of switched-mode power converters which feature implicit integration but require no iterative numerical solving algorithm for evaluation because they are explicitly solved prior to model execution. The derived models capture the large-signal dynamic behavior of the power converters, so their use and accuracy are not limited to any one set of operating conditions. These models can be implemented in any computational environment, including directly on an existing embedded controller as a digital twin. Since no iterative solver is required, the models are highly computationally efficient and have a very predictable worst-case execution time, which makes them especially suitable for real-time modeling applications like hardware-in-the-loop simulation or digital twins. The work includes experimental validation in an embedded environment and a discussion of sources of model error. Numerous applications of the modeling technique are explored on a variety of power electronic converters, and computational expense is analyzed. The method is further extended from switch averaged models to switching models and reduced-order models, and the method is qualitatively and quantitatively compared to recently proposed data-driven alternatives based on dynamic neural networks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Abbreviations

ACG ..... Automated Code Generation

ANN ..... Artificial Neural Network

CF ..... Closed Form

CPU ..... Central Processing Unit

DAB ..... Dual Active Bridge

DT ..... Digital Twin

DQ ..... Direct-Quadrature (axes in a rotating reference frame)

FPGA ..... Field Programmable Gate Array

GPU ..... Graphics Processing Unit

HDL ..... Hardware Description Language

HIL ..... Hardware in the Loop

HVDC ..... High Voltage Direct Current

I/O ..... Inputs and Outputs

LTI ..... Linear Time Invariant

MMC ..... Modular Multilevel Converter

NARX ..... Nonlinear AutoRegressive network with eXogenous inputs

PEBB ..... Power Electronics Building Block

PHIL ..... Power Hardware in the Loop

PHM ..... Prognostics and Health Monitoring

PI ..... Proportional-Integral

PSM.....Phase Shift Modulation

PLECS.....Piecewise Linear Electrical Circuit Simulation

RT .....Real Time

SPICE..... Simulation Program with Integrated Circuit Emphasis

WCET ..... Worst Case Execution Time

XAI .....Explainable Artificial Intelligence

CHAPTER 1

INTRODUCTION

Recently both academia and industry have shown great interest in real-time (RT) modeling and simulation of power electronic systems [1]. RT simulation has several important applications to power electronics. Historically, a key driver of the interest in it has been hardware-in-the-loop (HIL) and power-hardware-in-the-loop (PHIL) testing, which allows feedback controllers and power electronic systems to be tested in a laboratory environment which mimics the complete system even while portions of the system are unavailable. Specifically, HIL simulation is used for validation of embedded control systems, while PHIL simulation is used for validating powertrain hardware prototypes by emulating (arbitrarily complex) system behavior at the power exchanging interfaces of the prototype with a simulated model and a high-bandwidth power amplifier. Both HIL and PHIL testing can reduce the cost of verification testing and accelerate the development and deployment of new technologies, especially in control systems [2], [3], [4].

RT simulation is typically more computationally demanding than offline, non-RT simulation due to the stringent requirements on execution time. This is especially true when the simulated system is large and complex, as computational cost typically grows exponentially with the size of the system being solved. For this reason. many solutions for RT modeling have focused on parallel and high-performance computing resources, FPGAs and GPUs, to achieve RT execution [5], [6], [7]. However, there are a growing number of

applications of RT modeling which are pushing the limits of even the best available computing solutions.

Commercially available HIL systems are specially designed for high performance computing in the real time environment [8]. These systems, which are generally expensive [9], aim to maintain real-time execution while using very small timesteps with an explicitly integrated system. Achievable timesteps depend strongly on the size and properties of the simulated system but are frequently advertised at $1\ \mu s$ or less [10]. Customized FPGA-based solutions have demonstrated even better performance in the research literature, offering timesteps as low as $50\ ns$ in certain cases [11] at the cost of a less convenient and more complicated implementation.

However, even with modern hybrid multiprocessor and FPGA-based computing hardware, device-level HIL emulation of complex systems has presented implementation challenges due to the model complexities and high computational demand [12]. A classic example of a challenging system for modern HIL simulations is the modular multilevel converter (MMC), which poses a challenge for detailed RT simulation due to its inherently large size and high number of inputs, outputs, and states. A plethora of papers have been written to address this topology, with some focusing at the submodule level [13], some at the single converter level [14] [15], and some at the interconnected multi-converter system level [16].

More recently, a new technological trend has reignited interest in RT modeling for power electronic systems: the "digital twin" (DT). The concept of a DT has received much attention in both industry and the research literature [17]. Various definitions have been given, but the general concept of a DT is that a physical device has a virtual replica that

runs alongside the device, providing benefits including system monitoring, scenario and risk assessment, diagnostics, prognostics and health monitoring (PHM), and hardware-in-the-loop (HIL) simulation for adaptive control retuning [18]. DTs can exist at the system level, subsystem level, and/or the individual component level, and have been successfully applied in numerous technological domains. Examples of successful DT implementations include aircraft engines [19], wind turbines [20], microgrids [21], autonomous vehicles [22], and many other assets. At present, the US Navy has conceptualized a digital twin for ships which seeks to improve the "capability, adaptability, scalability, resiliency, safety, security, and usability" of current systems [23], and is actively supporting the development of the digital twin concept for power and energy management in future all-electric warships [24].

Like HIL simulation, digital twin technology also motivates computationally efficient models to enable faster-than-RT simulation. A key capability of the conceived digital twin for the naval power and energy system is the ability to run simulations of the physical system, either in real-time alongside the system or in faster-than-real-time to "look ahead" and predict the outcome of a control action in any given mission scenario. This look-ahead capability will allow an autonomous or semiautonomous decision-maker to control the system by evaluating one or more potential courses of action and selecting the action which leads to the best predicted outcome.

With the advent of digital twin technology, there are a growing number of applications of RT and faster-than-RT modeling which are required to run in resource-constrained computing environments. Expensive commercial HIL simulation systems are useful in laboratory testing environments, but in field-deployed equipment, size, weight,

power, and cost constraints may prevent the full adoption of such high-power computing platforms, especially for small device-level digital twins. The conceived digital twin will therefore require various models at the system and subsystem level which are computationally efficient, platform agnostic, and highly accurate and reliable, such that they can be executed anywhere—perhaps even on a simple microcontroller.

CHAPTER 2

TOWARDS DIGITAL TWINS FOR POWER ELECTRONICS

Power electronics is a mature technology which has been successfully researched, implemented, and refined for decades. As a result, there are a very wide variety of modeling methods and tools which have been developed to represent the technology using various levels of abstraction. Each modeling method and each level of abstraction provides unique capability and value and occupies a unique place in the trade-space between fidelity and computational efficiency.

2.1 LEVELS OF ABSTRACTION IN VARIOUS POWER ELECTRONICS MODEL TYPES

Several modeling types are summarized in order of increasing abstraction in Figure 2.1. On the detailed side, there are ultra-high fidelity electromagnetic modeling tools (e.g., Ansys) which account for the 3-dimensional geometric structure of the device and each component. Then there are detailed circuit models which ignore 3-D field effects, but which can provide very high-fidelity models of semiconductor devices, passive elements, and other components (e.g., SPICE). There are other circuit modeling tools with more abstractions such as the use of ideal switches (e.g., PLECS), and then there are switch-averaged models which aim to emulate the lower frequency electrical dynamics of the power electronics converters without including the faster switching-frequency components associated with switch commutation. Yet more abstract are DQ models, which represent sinusoidal AC systems as pseudo-DC systems from the vantage of a rotating reference frame. Ideal-switched models, average models, and DQ models are popular for control

system development and analysis. Then at the system level there are models which remove electrical converter dynamics entirely, effectively capturing only steady-state phenomena such as losses and operational limitations. Typically, such system-level models will intrinsically include the notional behavior of a closed-loop controller, since there is little value in separately considering a closed-loop feedback control for a model whose controlled dynamics have been neglected by abstraction. Steady-state system-level models can include DC representations of voltages and currents, or they can be abstracted yet further to use only power and energy quantities.

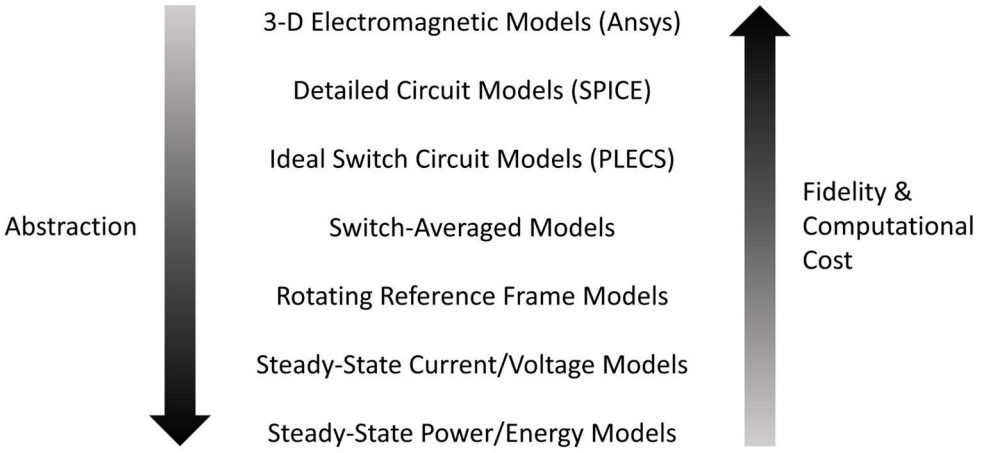

Figure 2.1. An overview of popular model types used for analysis and design of power electronic systems with various levels of abstraction.

2.2 Desirable Model Characteristics For Digital Twins

To create models for digital twins of power electronics converters and systems, we must first select a model type with an appropriate level of abstraction. This selection must consider the goals of the model and should appropriately balance fidelity and computational cost in the context of those goals. Since there are a large variety of possible functions that a digital twin may need to support, there is no single correct answer. In fact,

a single digital twin may bring together multiple models at multiple levels of abstraction and select a given model for evaluation depending on the task it is performing.

Nonetheless, without a total loss of generality, there are several important characteristics of digital twin models which will be desirable in many digital twin applications.

2.2.1 Real-Time Or Faster-Than-Real-Time

The first desirable characteristic is that the model be capable of executing in real-time—or ideally faster than real-time. Real-time execution is a hard requirement of any model which aims to operate in parallel with physical hardware and interact without significant latency. Moreover, if a digital twin model will be used to forecast outcomes of simulated possible futures, the model must execute at a rate faster than real-time so that future outcomes may be predicted before they happen.

These computation time constraints are likely to immediately necessitate some level of abstraction to reduce computational expense. While execution time does depend heavily on the computational platform, we can realistically rule out some of the most high-fidelity, computationally expensive modeling methods anywhere RT execution is required.

2.2.2 Controller Embeddable

Another desirable feature for digital twin models is that they are controller embeddable. This feature would be especially desirable in component-level digital twins which model a single piece of equipment rather than an entire interconnected system; while a system-level digital twin might reasonably be expected to have a dedicated, high-performance platform for execution, a component-level twin cannot reasonably be expected to always have a high-end simulator available for model execution. Optimally,

the digital twin model should be programmed and run directly on the local control platform of the piece of equipment that it represents. This solution is very natural and allows for the use of locally available measurement data in the digital twin without the need for more communication infrastructure. It also reduces the expense of incorporating the digital twin in the real system.

The feature of controller-embeddability places further constraints on the model choice. We must now find a model which is real-time or faster-than-real-time capable not just on any arbitrary platform, but particularly on a platform which offers limited computational capability. Moreover, since embedded platforms do not typically run sophisticated simulation programs but are programmed directly using low-level programming languages, the desirable model must be directly specifiable in such a language. It is not sufficient to define a model using some proprietary state-of-the-art simulation software—the model equations must be known, directly programmable, and evaluable without the aid of sophisticated numerical solvers.

2.2.3 Consistent Worst-Case Execution Time

Another desirable model characteristic of the digital twin model is that it has a reliable and predictable worst-case execution time (WCET). This is closely related to the requirement for real-time execution, as true real-time execution requires that the time-step used always be longer than the WCET of a single model iteration to avoid processing overruns.

In practice, this means that the model should not use an iterative solver like a Newton-Raphson solver, as these solvers can require tens or hundreds of iterations to

compute a single time-step. Since the number of iterations required for the solver to converge is unpredictable, the resulting WCET of such models is also unpredictable.

2.2.4 Reliable And Trusted

Digital twins can be expected to perform mission-critical functions in various mission-critical pieces of equipment. As such, the models the digital twin uses to perform its functions must be trusted and extremely reliable. Simulation models obviously do not suffer from degradation or physical health constraints like physical systems do, so in the modeling context, reliability means that the simulation must not fail to converge or go unstable due to purely numerical phenomena, even when subjected to a large variety of possible simulated scenarios.

Convergence is typically a problem in models which use numerical solvers like the aforementioned Newton-Raphson algorithm. The risk of even the best numerical solvers failing to converge under some circumstances further motivates solutions which remove the need for such solvers entirely. If the model is specified directly by explicitly solved equations, no iterative solver is necessary, and no convergence issues can arise.

2.2.5 Large Signal

To maintain high accuracy in all possible operating scenarios and system configurations, the model should be large signal. Small signal models of nonlinear systems like power electronic systems are useful only at a given operating point, and therefore cannot be relied upon to correctly predict any behavior outside that operating point. Contrarily, large signal models can represent the full non-linear dynamics of the system at all possible operating points. Clearly then, large signal models are preferred for digital

twin applications which must maintain accuracy through all imaginable operating scenarios and system configurations.

2.2.6 HIGHLY ACCURATE

High accuracy for a particular simulation model is reasonably self-explanatory: the digital twin model should closely match the physical twin when subjected to the same inputs. However, what is proposed here is not a particular model *per se* but a new, somewhat-general-purpose modeling method. In this more general context, high accuracy means not just that one single model is accurate, but that the entire proposed modeling technique is flexible enough to allow for the inclusion of all kinds of details so that any number of those details can be added to the model as appropriate. A highly accurate modeling technique should not preclude the insertion of parasitic effects, nonidealities, or other subtle phenomena present in the real system.

2.3 UNIQUENESS OF THE DESIRED MODEL

Many existing modeling methods for power electronics will display some combination of the characteristics above, but existing modeling methods are insufficient to produce a model which perfectly blends all the listed characteristics together in a single model; therefore, sacrifices and trade-offs must typically be made in one or two characteristics. Such sacrifices highlight an opportunity to develop a better model with a superior combination of desired characteristics and fewer shortcomings than the state-of-the-art.

2.3.1 TRADEOFFS WITH INTEGRATION METHODS

Perhaps the biggest trade-off when selecting a dynamic model is the choice of integration method. Each has advantages and drawbacks. For reference, Table 2.1 shows

a summary of the three first-order discretized integration methods—Euler Forward (explicit), Euler Backward (implicit) and Tustin (implicit)—in both derivative form and integral form, and in both the time domain and in the transformed domain (Laplace or Z). Higher-order integration methods are often used in both explicit and implicit solvers, but these are left out of the table for brevity.

Table 2.1. Overview of Continuous and First-Order Discrete Integration Methods

Domain/Type		Continuous	Discrete		
			Euler Forward	Euler Backward	Tustin
Time	$\int$	$x(t) = \int \dot{x}(t)dt$	$x[k] = \Delta t\, \dot{x}[k-1] + x[k-1]$	$x[k] = \Delta t\, \dot{x}[k] + x[k-1]$	$x[k] = \frac{\Delta t}{2}(\dot{x}[k] + \dot{x}[k-1]) + x[k-1]$
	$\frac{d}{dt}$	$\frac{d}{dt}x(t) = \dot{x}(t)$	$\frac{x[k] - x[k-1]}{\Delta t} = \dot{x}[k-1]$	$\frac{x[k] - x[k-1]}{\Delta t} = \dot{x}[k]$	$\frac{x[k] - x[k-1]}{\Delta t} = \frac{\dot{x}[k] + \dot{x}[k-1]}{2}$
Xfrmed (Laplace/Z, neglecting initial conditions)	$\int$	$X(s) = \frac{1}{s}\dot{X}(s)$	$X(z) = \frac{z^{-1}\Delta t}{1 - z^{-1}}\dot{X}(z)$	$X(z) = \frac{\Delta t}{1 - z^{-1}}\dot{X}(z)$	$X(z) = \frac{\Delta t}{2}\frac{1 + z^{-1}}{1 - z^{-1}}\dot{X}(z)$
	$\frac{d}{dt}$	$sX(s) = \dot{X}(s)$	$\frac{1 - z^{-1}}{\Delta t}X(z) = z^{-1}\dot{X}(z)$	$\frac{1 - z^{-1}}{\Delta t}X(z) = \dot{X}(z)$	$\frac{1 - z^{-1}}{\Delta t}X(z) = \frac{1}{2}\left(\dot{X}(z) + z^{-1}\dot{X}(z)\right)$

By definition, explicit integration calculates the state of a system at the next time step as:

$$x[t_k + \Delta t] = func(x[t_k], u[t_k], \Delta t) \tag{2.1}$$

Here x is the vector of state variables of the dynamic system, u is the input vector, t_k is the time at the current step, Δt is the timestep of the discrete model, and $func$ is a system-dependent function which generally is nonlinear. Similarly, implicit integration calculates the state of the system at the next time step as:

$$x[t_k + \Delta t] = func(x[t_k], u[t_k], x[t_k + \Delta t], u[t_k + \Delta t], \Delta t) \tag{2.2}$$

Note that (2.1) is a solved (explicit) equation that can be directly evaluated at each timestep, while (2.2) is an unsolved (implicit) equation which will require numerical solution at each timestep. Thus, explicit integration is usually preferred in RT simulation applications because it significantly reduces the complexity of the computations for each timestep, and because it has a very predictable WCET. However, explicit methods frequently do not perform well for "stiff" systems which have a wide range of time constants. Moreover, all explicit integration methods suffer from stability problems when the simulation timestep is increased [25]. To avoid numerical instability, explicit methods must use extremely small timesteps.

Contrarily, implicit integration methods do not suffer from stability problems and can maintain accuracy with much larger timesteps. However, the requirement to solve (2.2) at each timestep is not trivial. Iterative solvers (e.g., Newton-Raphson solvers) can usually solve such nonlinear equations, but the solution of each timestep can require tens

or hundreds of solver iterations. Thus, these solvers are computationally expensive and have unpredictable WCETs. For these reasons, RT execution using implicit methods can be challenging or even impossible.

2.3.2 Combining Integration Methods Using Hybrid Approaches

Some attempts have been made to combine implicit and explicit methods to create hybrid methods which are suitable for stiff systems and take some of the benefits from both. For example, one approach was proposed in [26] which involves adaptively altering the discretization method between several implicit and explicit options, such that the best algorithm can be selected based on the nature of the simulation inputs. Later, the authors of [27] proposed a hybrid implicit/explicit framework for RT simulation of power electronics which exploits latencies to partition the system, after which some linear subsystems are implicitly integrated while the overall system is explicitly integrated. This framework has been adopted and extended in [28], [11] and [29]. Unfortunately, the framework relies heavily on latencies in the system which are not present in all systems. Additionally, while hybrid methods can achieve better performance than traditional approaches for some circumstances, they invariably also suffer from some drawbacks—in the case of [26] – [29], the partial use of explicit integration is still cause for stability concerns.

CHAPTER 3

CLOSED FORM IMPLICIT MODELS

The simple concept of this proposed work is to discretize the state equations of a power electronic converter or system using implicit integration and then solve the resulting set of discrete-time nonlinear equations analytically using a general-purpose approach. This closed-form analytical solution takes the implicitly integrated system of (2.2) and converts it to the solved form of (2.1) so that it can be directly evaluated as an explicit method would. Thus, the proposed method captures all the benefits of both integration types while eliminating the drawbacks. Since the system is discretized using implicit integration, it retains all the stability and accuracy benefits of implicit methods, and it can use large timesteps. Since the system is explicitly solved, it can be directly evaluated at each time step without any iterative solver or matrix inversion operations. For convenience, the implicitly integrated, explicitly solved modeling approach will hereafter be referred to as the closed form implicit method and abbreviated CF-implicit.

Note that implicit equations like (2.2) are typically nonlinear and not guaranteed to have a closed-form analytical solution, so the modeling technique proposed here is not universally applicable to all systems. On the contrary, the CF-implicit modeling technique applies only to a specific form of nonlinear systems which is observed in power electronics and select other domains. By exploiting the unique form of the nonlinear equations in such systems, we can solve them analytically in closed form and reap the benefits of both implicit and explicit integration.

3.1 A Trivial CF-Implicit Solution for LTI Systems

Before examining nonlinear systems, it is useful to review the process for simpler case of linear, time-invariant (LTI) systems, which have been thoroughly studied and explored. Continuous-time LTI systems are definitionally of the form:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{3.1}$$

As usual, for a system of p states and q inputs, matrix A will have dimensions $p \times p$ and B will have dimensions $p \times q$. Now let us discretize the system with an implicit method, such as the backward Euler (EB) method. Doing so, we have the relationship:

$$x[t_k + \Delta t] = x[t_k] + \Delta t\, \dot{x}[t_k + \Delta t] \tag{3.2}$$

Substituting (3.1) into (3.2), we obtain

$$x[t_k + \Delta t] = x[t_k] + \Delta t(Ax[t_k + \Delta t] + Bu[t_k + \Delta t]) \tag{3.3}$$

Since we used an implicit method, the resulting equation is not solved (implicit)—that is, the term we would like to calculate $x[t_k + \Delta t]$ appears on both sides of the equation. However, for a linear system, this barrier is easily overcome with some simple linear algebra. By isolating the relevant terms on one side of the equation, factoring, and dividing the coefficient, we can see that:

$$x[t_k + \Delta t] = (I - \Delta t A)^{-1}(x[t_k] + Bu[t_k + \Delta t]) \tag{3.4}$$

where I is the identity matrix. Note here that we were able to explicitly solve the system despite using an implicit integration method, because the form of the LTI system allowed us to substitute and obtain a closed-form solution. Furthermore, it is clear from (3.4) that if the timestep Δt is held constant, the matrix inverse can be precomputed before

the model is executed, such that the only calculations which must be performed at each timestep are simple operations of multiplication and addition.

3.2 CF-Implicit Solution of Select Nonlinear Systems

Unfortunately, no such trivial solution exists for the general case of nonlinear dynamic systems, which are described generically as

$$\dot{x}(t) = f\big(x(t), u(t)\big) \tag{3.5}$$

If we again use the implicit backward Euler method, we have that

$$\begin{aligned} x[t_k + \Delta t] &= x[t_k] + \Delta t\, \dot{x}[t_k + \Delta t] \\ &= x[t_k] + \Delta t\, f(x[t_k + \Delta t], u[t_k\ + \Delta t]) \end{aligned} \tag{3.6}$$

Beyond this point, we cannot proceed to solve analytically without further knowledge of the function f. Moreover, in many cases it is not possible to solve (3.6) in closed form. For this reason, general purpose simulation solvers employ sophisticated iterative numerical methods to solve the system at each time step.

However, it is possible to consider special cases of (3.5) which are not LTI, but which still allow for explicit closed-form solution of the implicitly discretized system. In particular, let us consider a system of the form

$$\dot{x}(t) = Ax(t) + Bu(t) + d(t)\big(Ex(t) + Fu(t)\big) \tag{3.7}$$

where d is a scalar input to the system which introduces nonlinearity (in that it acts as a coefficient to other states and inputs) and E and F are constant coefficient matrices of the same dimensions as A and B, respectively. This form is widely applicable to power electronic converters and systems—where typically a modulated input d (which is produced by the feedback control) introduces the nonlinearity into the switch-averaged

state equations of the power electronic converter. Assuming this form, we can discretize the system using the implicit backward Euler method by substituting (3.7) into (3.2) and obtain:

$$x[t_k + \Delta t] = x[t_k] + \Delta t\, A\, x[t_k + \Delta t] + \Delta t\, B\, u[t_k + \Delta t] + \Delta t\, d[t_k + \Delta t]\, E\, x[t_k + \Delta t] + \Delta t\, d[t_k + \Delta t]\, F\, u[t_k + \Delta t] \quad (3.8)$$

Grouping the terms containing $x[t_k + \Delta t]$ on the left side and factoring, we have:

$$(I - \Delta t A - \Delta t\, d[t_k + \Delta t]\, E)\, x[t_k + \Delta t] = x[t_k] + \Delta t\, B\, u[t_k + \Delta t] + \Delta t\, d[t_k + \Delta t]\, F\, u[t_k + \Delta t] \quad (3.9)$$

And finally, by matrix inversion we obtain:

$$x[t_k + \Delta t] = (I - \Delta t A - \Delta t\, d[t_k + \Delta t]\, E)^{-1} (x[t_k] + \Delta t\, B\, u[t_k + \Delta t] + \Delta t\, d[t_k + \Delta t]\, F\, u[t_k + \Delta t]) \quad (3.10)$$

Note here that our solution process was quite similar to the trivial LTI case, with just a few added terms. However, there is one major distinction: even with a fixed timestep Δt, the inverted matrix in this solution now changes as a function of time due to the input d. Traditionally, this would mean that the inverse has to be recalculated at each timestep when the input d is updated. However, to avoid the high computational cost of repeatedly performing the matrix inversion, we can instead use a symbolic solver to precalculate the inverse while leaving d as an unknown variable. In this way, we can capture the same benefits as the LTI system and greatly speed up the model execution time while also capturing the full large-signal nonlinear behavior of the system of (3.7).

3.3 CF-Implicit Models For Systems with Multiple Modulated Inputs

The previous derivation assumed a nonlinear system where only one scalar input d was responsible for the nonlinearity in the system. This assumption holds for a wide variety

of single-stage power converters where only one modulated control variable is present in the state equations. However, there are other power converters for which there are multiple modulated control variables present in the state equations which introduce nonlinearity into the system. Notably, any multi-stage power converters and any three-phase converters will have more than one modulated control variable present in the switch-averaged state equations.

Fortunately, the CF-implicit method is easily extended to the case of multiple control inputs just by adding extra terms. For a system featuring n modulated control inputs, designated $d_1, d_2, \dots d_n$, we can formulate the system state equations as

$$\dot{x}(t) = Ax(t) + Bu(t) + \sum_{i=1}^{n} \Big(d_i(t) \big(E_i x(t) + F_i\, u(t) \big) \Big) \tag{3.11}$$

which the reader can see takes the same form as (3.7) but has separate terms and matrices E_i and F_i for each modulated input. While the equation is more complicated upon first inspection, its general form is still quite simple and lends itself to automatic population of the system matrices so that little manual work is required. As in the previous case, the system of (3.11) can be discretized using backward Euler and solved in the same manner, yielding the solution:

$$\begin{aligned} x[t_k + \Delta t] = & \left(I - \Delta t \left(A + \sum_{i=1}^{n} (d_i[t_k + \Delta t]\, E_i) \right) \right)^{-1} \\ & \left(x[t_k] + \Delta t \left(B + \sum_{i=1}^{n} (d_i[t_k + \Delta t]\, F_i) \right) u[t_k + \Delta t] \right) \end{aligned} \tag{3.12}$$

Again, we can use a symbolic solver to invert the matrix leaving each of the modulating inputs $d_1, d_2, \dots d_n$ as unknowns. In this way we obtain a fully-closed-form

solution to the implicitly discretized state equations, and no matrix inversion is required within the simulation loop.

Finally note that while we have assumed the use of backward Euler for the presented derivations, the proposed method can easily be applied to any other implicit discretization techniques as well, including implicit methods of higher order. For example, using Tustin's method (a.k.a. the bilinear transform), and solving in the same manner we obtain:

$$x[t_k + \Delta t] = \left(I - \frac{\Delta t}{2}\left(A + \sum_{i=1}^{n} (d_i[t_k + \Delta t]\, E_i) \right) \right)^{-1} \left(x[t_k] + \frac{\Delta t}{2}\dot{x}[t_k] + \frac{\Delta t}{2}\left(B + \sum_{i=1}^{n} (d_i[t_k + \Delta t]\, F_i) \right) u[t_k + \Delta t] \right) \quad (3.13)$$

3.4 A Highly Generalized CF-Implicit Form

Most power electronic systems have switch-averaged state equations which can be manipulated to fit the form of (3.11). However, it is possible to find special systems which require an even more general form. An example system which requires a more general form is the dual active bridge (DAB) converter, which is operated using phase shift modulation (PSM). The DAB's power transfer equation exhibits a dependence on the square of the phase shift ϕ—something which cannot be made to fit the structure of (3.11). A more thorough analysis of the DAB is provided in Section 5.2.

The most general form to which the CF-implicit method can be applied is

$$\dot{x}(t) = g\big(u(t), d(t)\big)x(t) + h\big(u(t), d(t)\big) \quad (3.14)$$

where g is a $p \times p$ matrix of functions and h is a $p \times 1$ vector of functions. Each element in h and g is a general nonlinear function which can contain any nonlinear

combination of elements from input vectors u and d and produces a scalar output. The key feature of (3.14) which enables the method to be used is that the state variable vector x can be pulled out of the nonlinear functions—which is not necessarily possible in the completely general nonlinear system of (3.5). This form is even more widely applicable to power electronic converters and systems than previous formulations. Specifically, the general form in (3.14) can be used anywhere that state-space averaging applies [30], where the small ripple approximation holds [31] and each switching subinterval is linear, and even in several applications where the small ripple approximation does not hold (i.e., in the dual active bridge converter).

Applying Euler Backward again to (3.14), we have that

$$\begin{aligned} x[t_k + \Delta t] &= x[t_k] + \Delta t\ \dot{x}[t_k + \Delta t] \\ &= x[t_k] + \Delta t\ g(u[t_k + \Delta t], d[t_k + \Delta t])x[t_k + \Delta t] \\ &\quad + \Delta t\ h(u[t_k + \Delta t], d[t_k + \Delta t]) \end{aligned} \tag{3.15}$$

As before, the terms containing x can be grouped on the left side of the equation and x can be factored out, leaving:

$$\big(I - \Delta t\ g(u[t_k + \Delta t], d[t_k + \Delta t])\big)x[t_k + \Delta t] = x[t_k] + \Delta t\ h(u[t_k + \Delta t], d[t_k + \Delta t]) \tag{3.16}$$

And finally by matrix inversion, the generalized form of (3.14) can be solved for EB integration to produce:

$$\begin{aligned} x[t_k + \Delta t] &= \big(I - \Delta t\ g(u[t_k + \Delta t], d[t_k + \Delta t])\big)^{-1} \\ &\quad \big(x[t_k] + \Delta t\ h(u[t_k + \Delta t], d[t_k + \Delta t])\big) \end{aligned} \tag{3.17}$$

As with previous forms, other implicit integration methods can also be used with a similar solution technique. Using Tustin integration, solving (3.14) yields:

$$x[t_k + \Delta t] = \left(I - \frac{\Delta t}{2} g(u[t_k + \Delta t], d[t_k + \Delta t])\right)^{-1} \left(x[t_k] + \frac{\Delta t}{2}\dot{x}[t_k] + \frac{\Delta t}{2} h(u[t_k + \Delta t], d[t_k + \Delta t])\right) \quad (3.18)$$

CHAPTER 4

EXPERIMENTAL VALIDATION OF CF-IMPLICIT MODELS

4.1 MATRIX DECOMPOSITION OF TWO-STAGE CONVERTER STATE EQUATIONS

In this section we demonstrate the proposed CF-implicit method on a two-stage feedback-controlled converter with five state variables and four modulating inputs by formulating the state equations under the structure of (3.11) and creating solved RT models using (3.12) and (3.13). Figure 4.1 shows a schematic diagram of the two-stage converter we are modeling, which consists of a boost step-up converter followed by a three-phase inverter which feeds a wye-connected resistive load. This topology makes a good system for study and is also commonly used in real-world applications such as a inverters interfacing a low-voltage DC element like a battery or photovoltaic panel string to a three-phase AC system.

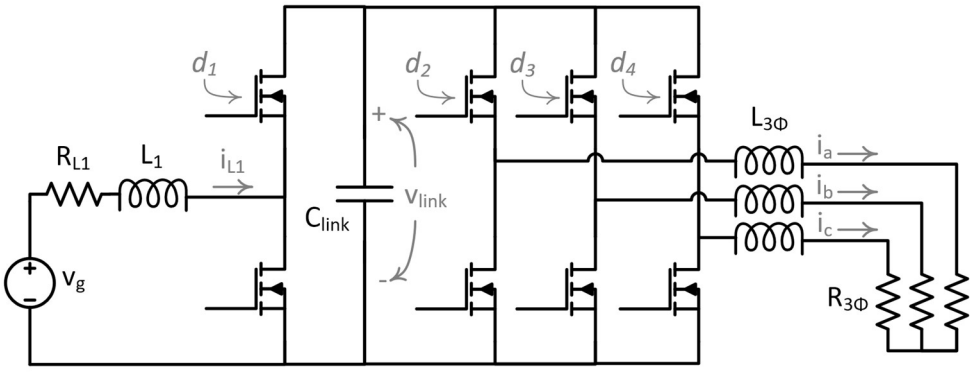

Figure 4.1. Schematic diagram of the modeled system, a two-stage converter with a boost stage and a three-phase inverter.

Note that in the diagram, L_1 represents boost inductor, R_{L1} represents the boost inductor's equivalent series resistance, C represents the DC link capacitance, and $L_{3\Phi}$ represents the inductance of the three inverter phase legs, and $R_{3\Phi}$ represents the resistive load on each phase. The five state variables are the boost inductor current i_{L1}, the line currents i_a, i_b, and i_c, and the DC link voltage v_{link}.

We define d_1, d_2, d_3, d_4 as the duty of the high side switch on each phase leg and consider that the low side switch duty is the complement of the high side duty $d_i' = 1 - d_i$, neglecting any dead time. Doing so, we can develop switch-averaged, continuous-time state equations of the system, which are given below in (4.1) - (4.5). These equations, which constitute the averaged plant model, are nonlinear due to the multiplication of the duty inputs and the system states. Note that while there are five state equations listed—corresponding to the five energy storage elements in the system—the system has only four linearly independent states, because the three-phase inverter currents i_a, i_b, and i_c must always sum to zero.

$$\frac{dv_{link}}{dt} = \frac{1}{C_{link}}(d_1 i_{L1} - d_2 i_a - d_3 i_b - d_4 i_c) \tag{4.1}$$

$$\frac{di_{L1}}{dt} = \frac{1}{L_1}\left(v_g - d_1 v_{link} - i_{L1} R_{L1}\right) \tag{4.2}$$

$$\frac{di_a}{dt} = \frac{1}{L_{3\Phi}}\left(d_2 v_{link} - \frac{1}{3}(d_2 + d_3 + d_4)v_{link} - i_a R_{3\Phi}\right) \tag{4.3}$$

$$\frac{di_b}{dt} = \frac{1}{L_{3\Phi}}\left(d_3 v_{link} - \frac{1}{3}(d_2 + d_3 + d_4)v_{link} - i_b R_{3\Phi}\right) \tag{4.4}$$

$$\frac{di_c}{dt} = \frac{1}{L_{3\Phi}}\left(d_4 v_{link} - \frac{1}{3}(d_2 + d_3 + d_4)v_{link} - i_c R_{3\Phi}\right) \tag{4.5}$$

Since this system has four inputs which introduce nonlinearity via switch modulation, we adopt the structure of (3.11), with $n = 4$. By combining (4.1)-(4.5) into a single matrix equation of this form, we obtain the following vectors and matrices:

$$x = \begin{bmatrix} v_{link} \\ i_{L1} \\ i_a \\ i_b \\ i_c \end{bmatrix} \tag{4.6}$$

$$u = \begin{bmatrix} v_g \end{bmatrix} \tag{4.7}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{R_{L1}}{L_1} & 0 & 0 & 0 \\ 0 & 0 & -\frac{R_{3\Phi}}{L_{3\Phi}} & 0 & 0 \\ 0 & 0 & 0 & -\frac{R_{3\Phi}}{L_{3\Phi}} & 0 \\ 0 & 0 & 0 & 0 & -\frac{R_{3\Phi}}{L_{3\Phi}} \end{bmatrix} \tag{4.8}$$

$$B = \begin{bmatrix} 0 \\ \frac{1}{L_1} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.9}$$

$$E_1 = \begin{bmatrix} 0 & \frac{1}{C_{link}} & 0 & 0 & 0 \\ -\frac{1}{L_1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.10}$$

$$E_2 = \begin{bmatrix} 0 & 0 & -\frac{1}{C_{link}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{2}{3L_{3\Phi}} & 0 & 0 & 0 & 0 \\ -\frac{1}{3L_{3\Phi}} & 0 & 0 & 0 & 0 \\ -\frac{1}{3L_{3\Phi}} & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.11}$$

$$E_3 = \begin{bmatrix} 0 & 0 & 0 & -\frac{1}{C_{link}} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{3L_{3\Phi}} & 0 & 0 & 0 & 0 \\ \frac{2}{3L_{3\Phi}} & 0 & 0 & 0 & 0 \\ -\frac{1}{3L_{3\Phi}} & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.12}$$

$$E_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & -\frac{1}{C_{link}} \\ 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{3L_{3\Phi}} & 0 & 0 & 0 & 0 \\ -\frac{1}{3L_{3\Phi}} & 0 & 0 & 0 & 0 \\ \frac{2}{3L_{3\Phi}} & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.13}$$

$$F_1 = F_2 = F_3 = F_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.14}$$

Note that while d_1, d_2, d_3, and d_4 are inputs to the system, they are not included in the input vector u because they are handled separately in (3.11). While it is somewhat

tedious to populate these matrices by hand, this process could be automated as in many other general-purpose simulation solvers.

4.2 Model Validation as an Embedded Digital Twin

Having formulated the system in this manner, we entered the matrices in MATLAB and defined the duty cycles, states, and other inputs as symbolic variables using the Symbolic Math Toolbox. Then, by entering (3.12) and (3.13), we generated callable functions which calculate $x[t_k + \Delta t]$ at each timestep for a given set of inputs $x[t_k]$, $u[t_k]$, $u[t_k + \Delta t_k]$, and $d[t_k + \Delta t_k]$. We chose a timestep $\Delta t = 50\ \mu s$ to match the switching period of the converter for 20 kHz switching. In this way, the CF-implicit model can be evaluated directly on the converter's digital controller in real time as a digital twin of the physical converter.

Choosing a timestep equal to the controller interrupt period is convenient because the solved model function is then called once per switching period using the same interrupt as the existing feedback controls. Note that using a timestep this large is acceptable because the method uses implicit integration, which does not suffer from stability issues and offers good accuracy with larger timesteps than an explicit method. Using this large timestep with an explicit method would cause the model to go unstable.

The feedback control structure adopted for the two-stage converter is depicted in Figure 4.2. As shown, the boost converter stage is controlled with a nested two-loop compensation scheme which regulates the boost inductor current and the DC link voltage. The three-phase inverter is controlled with PI current compensators in the rotating DQ reference frame.

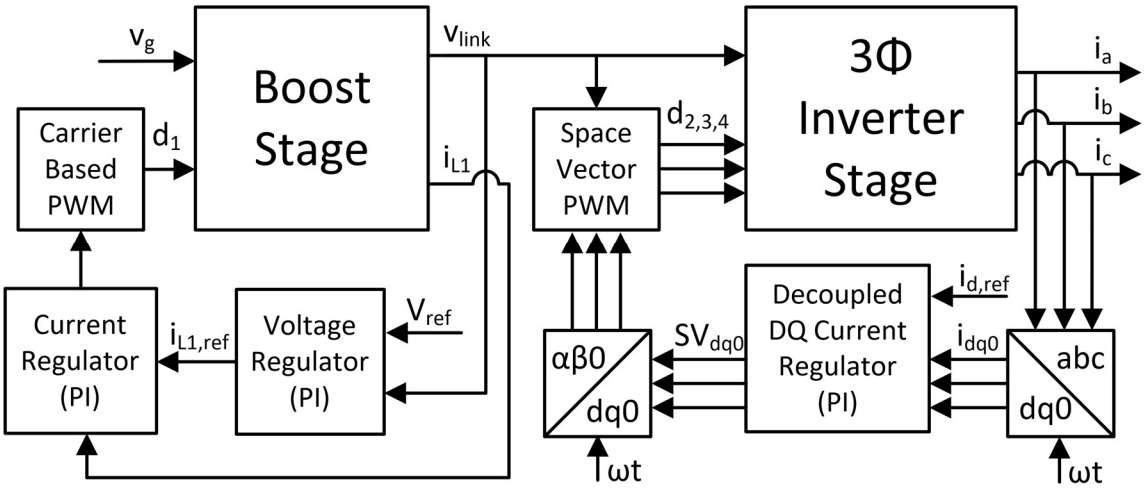

Figure 4.2. The block diagram of the control system implemented on the two-stage converter. The boost converter stage uses a two-loop control structure, with an inner current loop and an outer voltage loop, and the inverter stage uses a DQ current controller. The angle generator which produces ωt is not shown.

To validate the CF-implicit models, we deployed the derived model of the two-stage converter in Figure 4.1 as a real-time digital twin, using the software configuration shown in Figure 4.3. Note that the digital twin has its own unique copy of the control system depicted in Figure 4.2. The physical twin was built in hardware using four integrated half bridge models, inductors, and a three-phase resistive load. A photograph of the setup is shown in Figure 4.4. The nominal parameter values were $L_1 = 1.25\,mH$, $L_{3\Phi} = 2.5\,mH$, $R_{3\Phi} = 10\Omega$, and $C_{link} = 1040\,\mu F$. The assumed ESR of the boost inductor was $R_{L1} = 29.7\,m\Omega$.

We also deployed more traditional, explicitly integrated Euler Forward (EF) RT models to the same platform to compare accuracy, stability, and computational efficiency among the models and prove the value of the proposed models. We ran an experiment inducing a few transients on the physical twin and each of the real-time digital twin models and captured the output data.

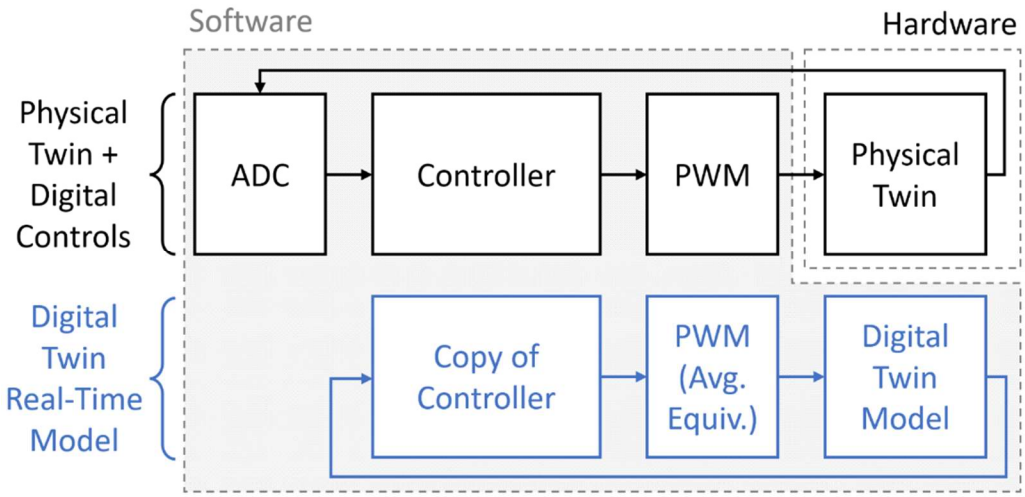

Figure 4.3. A breakdown of hardware and software components for the experimental setup, including the physical twin and digital controls (top, black) and the digital twin real-time model (bottom, blue). All software components are executed in real-time on the digital control platform.

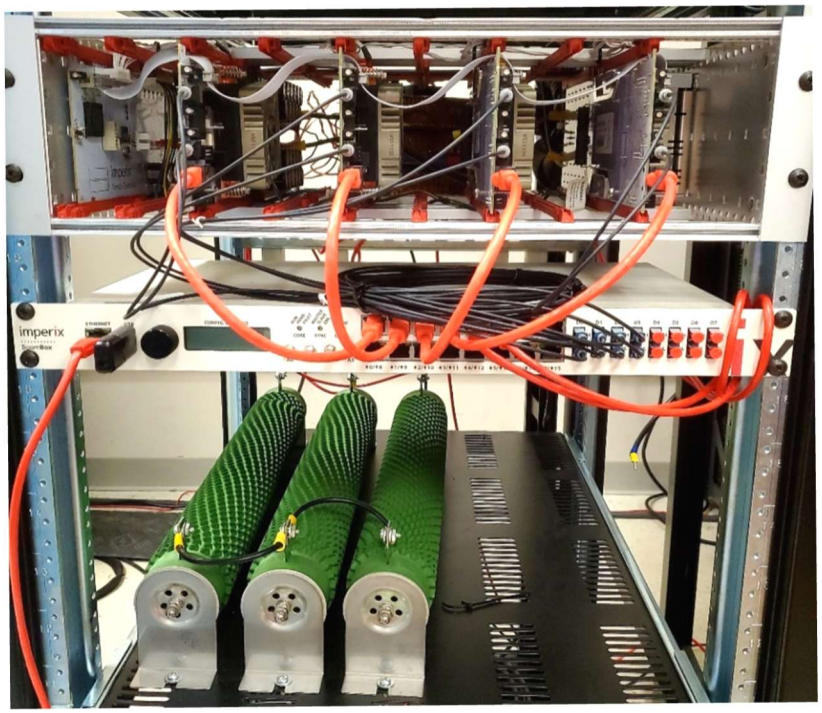

Figure 4.4. The physical twin was a hardware prototype of the converter in Figure 4.1, built with four integrated half-bridge modules (top), four inductors (behind the modules, not pictured), and a three-phase resistive load (bottom). The control platform sits below the modules.

The data captured in these real-time experiments are plotted in Figure 4.5-Figure 4.8. In each test, the experiment involves startup and two subsequent control reference steps. At time $t = 0$, the DC link voltage v_{link} is charged up to the same value as input voltage v_g which is held constant at $125V$ by a laboratory power supply. Note that this is unavoidable, as the high side diode of the boost stage ensures that the DC link voltage cannot fall below the input voltage. The controller then raises v_{link} to the reference value of 400V, which takes some time because the boost inductor current reference saturates at 25A. The startup phase completes around $t = 0.025$ seconds, upon which the boost inductor current drops back down to roughly zero. Then, at $t = 0.05$ seconds, the three-phase inverter current reference $i_{d,ref}$ is stepped from 0A to 10A, and later, and at $t = 0.08$ seconds the DC link voltage reference is stepped from 400V to 420V.

Figure 4.5 shows the experimental results for the CF-implicit model with backward Euler discretization and a timestep of $\Delta t = 50\mu s$, equal to the converter switching period T_{SW}. The DT model outputs clearly show excellent agreement with the PT measured waveforms. Likewise, Figure 4.6 shows the waveforms for the CF-implicit DT using Tustin discretization, with excellent accuracy.

Figure 4.7 shows the same test case with a traditional explicitly integrated DT model which uses forward Euler and the same timestep of $\Delta \mathrm{t} = 50\ \mu s$. Here we observe the serious shortcoming of explicit integration—the explicit DT model is not stable with this large timestep.

Figure 4.8 shows the explicitly integrated forward Euler model of Figure 4.7, but with the timestep reduced to $\Delta t = 1\ \mu s$ to ensure numerical stability. Here again the model

is stable, but at the cost of a much smaller timestep. This smaller timestep drives up the computational cost of the model, requiring more CPU resources for RT operation.

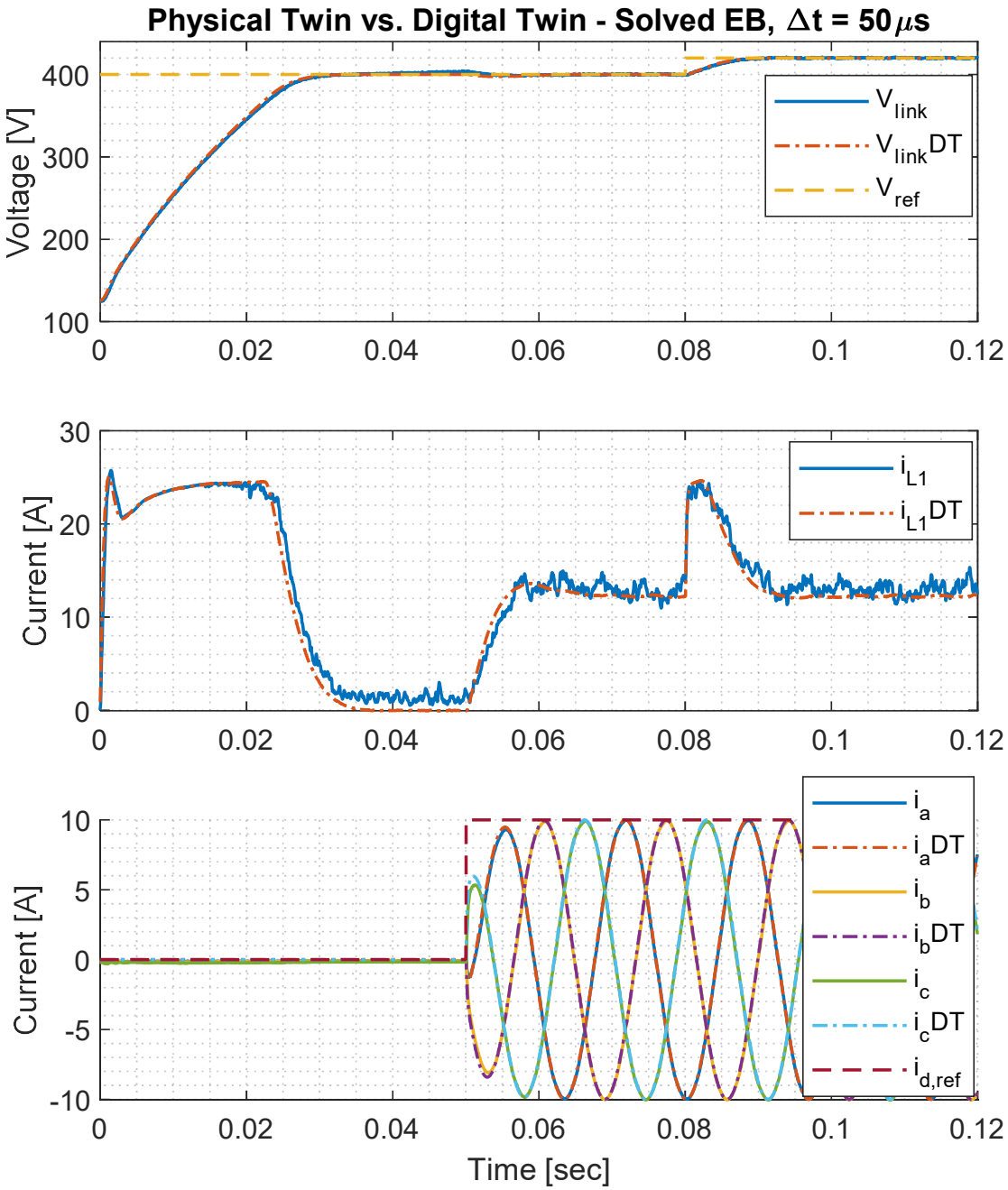

Figure 4.5. Measurements and outputs captured from the real-time control platform for a DT model using the CF-implicit method with Euler Backward discretization at a 50µs timestep. Even with a large timestep, the DT model is stable and accurate.

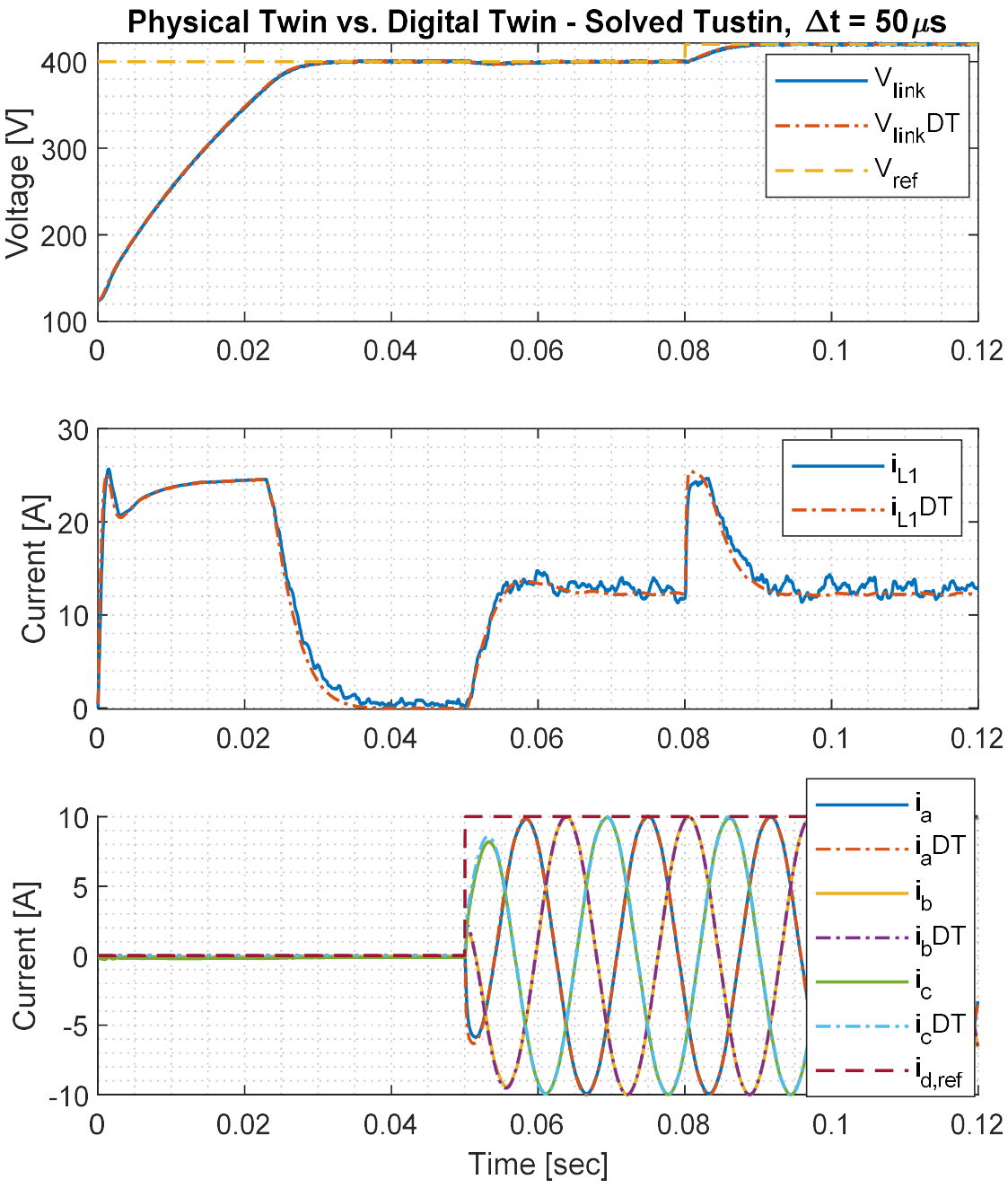

Figure 4.6. Measurements and outputs captured from the real-time control platform for a DT model using the CF-implicit method with Tustin discretization at a 50μs timestep. This DT model is stable and accurate.

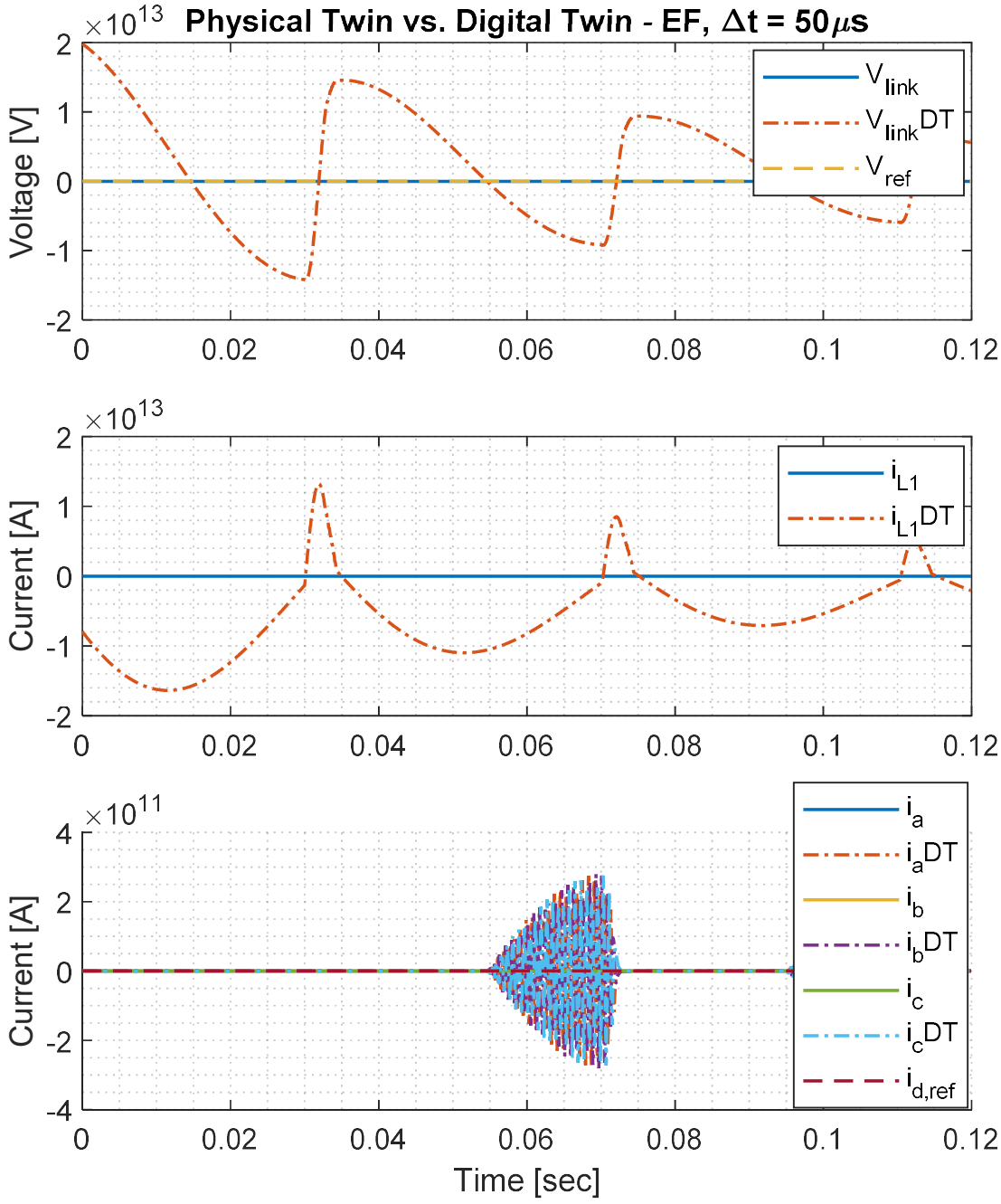

Figure 4.7. Measurements and outputs captured from the real-time control platform for an explicitly integrated DT model featuring Euler Forward discretization and a 50µs timestep. Note the large y-axis scaling. This DT model is numerically unstable.

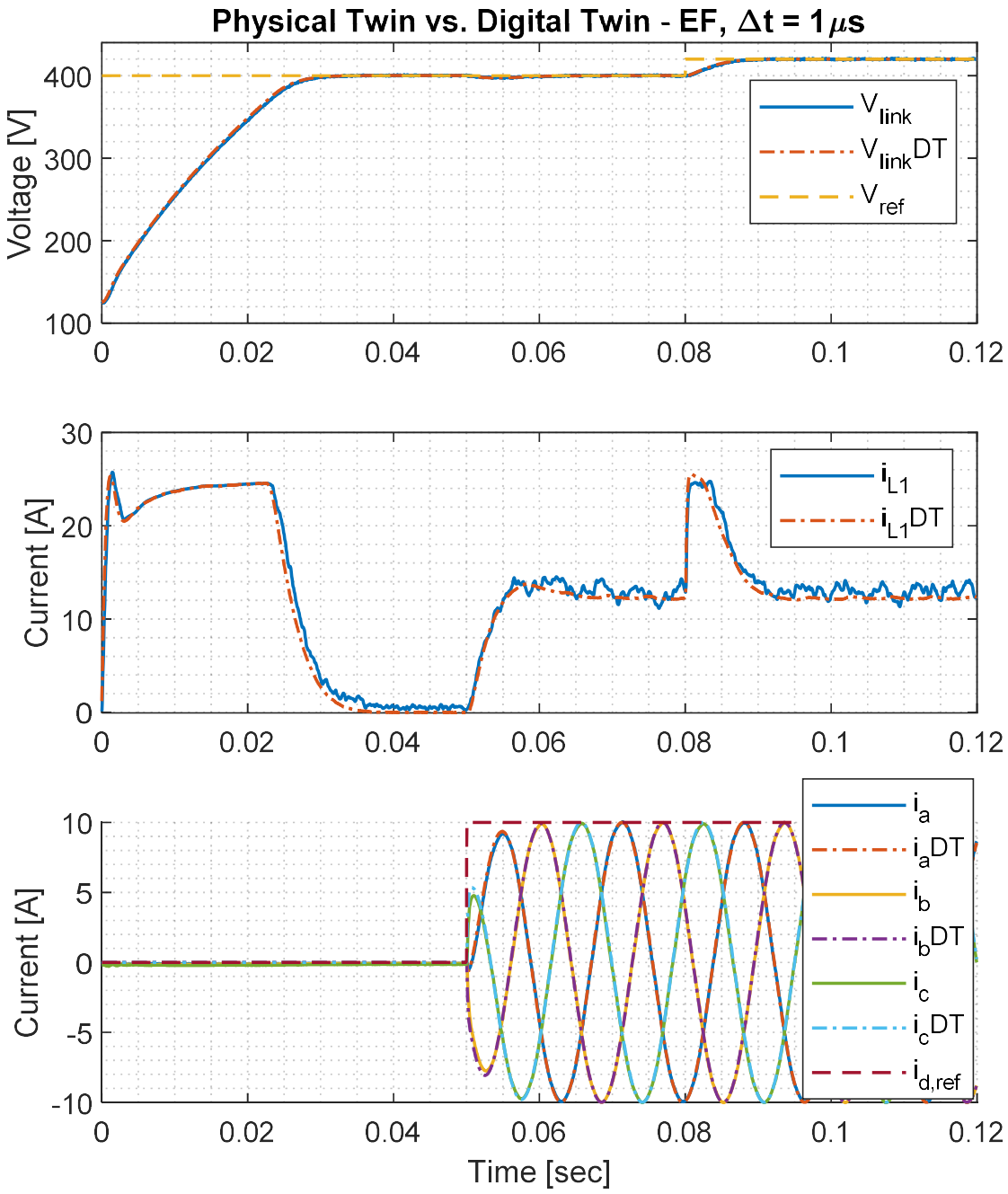

Figure 4.8. Measurements and outputs captured from the real-time control platform for an explicitly integrated DT model featuring Euler Forward discretization and a 1μs timestep. This DT model is stable and accurate.

4.3 Analysis of Computational Expense

It is clear from the simulated waveforms that the CF-implicit models proposed herein offer improved numerical stability, allowing the model to be executed with much larger timesteps than could be used in explicitly integrated models. The use of larger timesteps reduces the computational cost of model execution significantly. To quantify the computational performance improvement provided by the proposed method, we compared the CF-implicit and explicit digital twin models as a function of the number of simulation timesteps per control interrupt. The control platform used in the experiments was an Imperix B-box 3.0, and in each experiment the control code and the digital twin model (as shown in Figure 4.3) were run together on the B-box's CPU: an ARM Cortex A9 with a 1GHz clock frequency and 1GB of DDR3 memory.

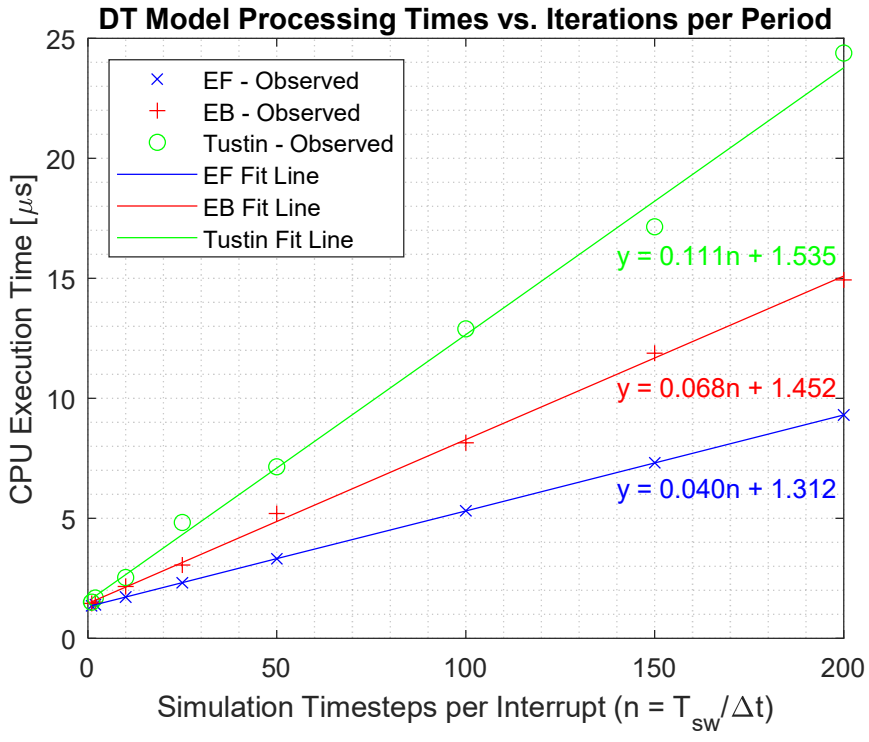

Figure 4.9. A scatter plot showing the DT model processing times for each integration method as observed on the control platform's 1GHz ARM CPU.

Figure 4.9 shows the DT model's CPU processing times for each of the three first-order integration methods as a function of the number of steps per interrupt. Linear regression lines of the data are also shown, along with the corresponding line equations. As expected, each of the methods' computational cost is inversely proportional to the model timestep for each method, growing approximately linearly with the number of steps per interrupt. There is also a small non-zero y-intercept for each of the three regression lines, which represents base CPU load which is present even when the model equations are not evaluated. This base CPU load includes all routine processing tasks that occur on the embedded CPU which are not a part of the DT model code, including terminal PC communications, exception and fault management, watchdog timer management, etc.

Note that while each of the three methods do demonstrate a strong linear dependence on timestep, the observed data points for the two CF-implicit methods do not fall exactly on the regression line—rather, there is a small, somewhat random variance around the regression lines. This variance can be attributed to the symbolic math toolbox which we used to generate the callable functions for (3.12) and (3.13). When the callable function is generated, the toolbox performs an optimization to try to make the function as computationally efficient as possible, but the optimization is not always equally successful, leading to small variations around the expected value predicted by the regression line. In contrast, the observed data points for EF fall directly on the regression line because the model equations are not generated by the symbolic toolbox.

Inspecting Figure 4.9 further, we can clearly see that the explicit EF is the most computationally efficient of the methods *for a given timestep*. However, a fair comparative analysis must account for the fact that explicit integration always requires larger number n

of timesteps per interrupt than implicit methods to maintain numerical stability. Recall that Figure 4.7 showed that EF was unstable for a $50\ \mu s$ timestep (which in Figure 4.9 corresponds to $n = 1$), while both CF-implicit methods were stable and accurate with this timestep. To ensure reliable stability, the explicit EF method required a smaller $1\ \mu s$ timestep ($n = 50$). If we compare the methods on this basis, we can clearly see that the CF-implicit Tustin and EB models are faster at $n = 1$ than the EF model is at $n = 50$. We can also demonstrate this more generically using the regression lines, showing that CF-implicit EB will be more computationally efficient if:

$$t_{CPU,EB} \leq t_{CPU,EF}$$
$$0.068n_{EB} + 1.452 \leq 0.040n_{EF} + 1.312 \tag{4.15}$$

Likewise, CF-implicit Tustin will likely be more computationally efficient if:

$$t_{CPU,Tustin} \leq t_{CPU,EF}$$
$$0.111n_{Tustin} + 1.535 \leq 0.040n_{EF} + 0.312 \tag{4.16}$$

If we use the $50\ \mu s$ timestep for the CF-implicit methods and the $1\ \mu s$ timestep for the explicit method, we obtain that $t_{CPU,EF} = 2.31\ \mu s$ while $t_{CPU,Tustin} = 1.65\ \mu s$ and $t_{CPU,EB} = 1.52\ \mu s$. Thus, the CF-implicit methods also offer a computational performance benefit.

Another possible way to compare the computational cost of different integration methods is to look at the incremental cost to perform the required number of simulation steps for a given method. In this analysis, the fixed overhead cost of each method (represented by the y-intercept of each line in Figure 4.9) is neglected. Table 4.1 shows that the CF-implicit EB and Tustin methods have a computational cost of 68ns and 111ns respectively for the required single timestep, to be compared with a total cost of 2,000ns

for the 50 steps required by the EF method for numerical stability. This is an improvement by a factor of 29 for EB and 18 for Tustin.

Table 4.1. Computational Cost Comparison Based on Figure 4.9, Neglecting Fixed Overhead Costs

Method	# of Steps for Reliable Stability	Cost per Step [ns]	Total Cost [ns]
EF	50	40	2,000
EB	1	68	68
Tustin	1	111	111

4.4 Model Error Analysis

As with any model, it is critical that users understand the inherent limitations of the CF-implicit models, the sources of error which are present, and the operating conditions which are necessary to obtain accurate results. Since the CF-implicit models proposed here are mathematically identical to ordinary implicitly integrated models, they are generally robust and do not induce significant error from any assumptions or approximations. The primary sources of potential error in the proposed models are discussed here.

4.4.1 Unmodeled Parasitic Error

All parasitics such as capacitor and inductor ESR, semiconductor on-resistance and forward drops, etc. that are considered must be included in the system state equations and resultant system matrices. All neglected parasitics are obviously potential sources of error.

Practicing engineers are familiar with parasitics as a source of model error, so an in-depth discussion of this type of error is excluded here. However, it is important that the engineer consider which parasitics to include in a CF-implicit model within the context of the model's intended purpose.

4.4.2 Discretization Error

There are two types of discretization error. Some discretization error is inherent: that is, some information is always lost when a continuous-time system is modeled in a discrete environment. Additionally, some discretization error can be attributed to the chosen integration method, as all practical integration methods are approximations which sacrifice some amount of accuracy for computational efficiency.

Discretization error warrants significant discussion, as a timestep that is too large can induce intolerable error. As a general rule, the discrete model should always be run with a timestep no greater than one tenth the period of highest frequency spectral component that the model is intended to capture. Figure 4.10 shows the error introduced by approximating an ideal integrator with each of the three first-order integration methods. Clearly, each method is an excellent approximation at frequencies which are low compared to the timestep, but the approximations break down at high frequency.

Note that as the discrete sampling frequency increases—i.e., the timestep Δt decreases—discretization error decreases and approaches zero. However, in resource-constrained computing environments, reducing the timestep beyond a certain threshold will induce processing overruns which prevent RT execution, so some discretization error may be unavoidable in the RT model.

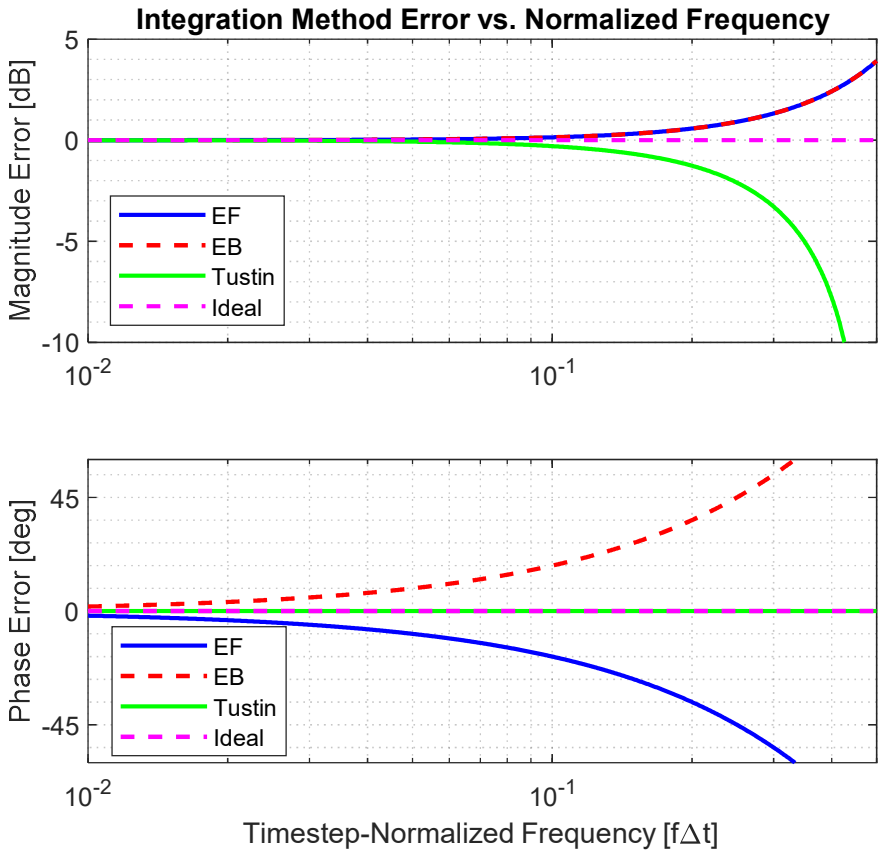

Figure 4.10. Discrete integration methods typically induce appreciable error for any frequency components which exceed one-tenth of the sampling frequency.

4.4.3 Small Ripple Approximation Error

All switch-averaged converter models (including those presented in this work) rely on an approximation that the ripple of the converter waveforms is small with respect to the average value of the waveforms. Typically, this approximation holds well, so the error induced by the approximation is negligible. In any case, it is important to note that this type of error is not improved by reducing the discrete timestep.

4.5 Summary of Experimental Results for Proposed Models

The presented work describes a novel means to obtain an explicitly solved, closed form solution of an implicitly integrated set of state equations. This solution is limited in scope, but it is generally applicable to a wide variety of power electronic systems. The CF-

implicit method delivers unparalleled capability for RT modeling by combining the stability and accuracy of implicit methods with the low computational complexity of explicit methods. The proposed models capture the full nonlinear dynamic behavior of the converter and require no iterative solver or repeated matrix inversion operations for model evaluation.

Due to the low computational cost and the lack of any iterative numerical solver, CF-implicit models are extremely well-suited for real-time applications. They can be evaluated directly on a power converter's embedded controller as a digital twin of the converter and used for many purposes, including online diagnostics, prognostics, optimization, and autonomous decision-making.

CHAPTER 5

SWITCHING MODELS USING THE CF-IMPLICIT METHOD

The CF-implicit models presented in previous chapters are switch-averaged models which assume that feedback controlled inputs (i.e., the converter duty cycles) are bounded continuous variables. Duty cycles in particular can take any value between zero and one; other controlled inputs, such as phase shift in a phase-shift-modulated converter, can similarly be considered as bounded continuous variables. However, the state equations which are manipulated and evaluated using the CF-implicit approach in this method are valid for more than just averaged models. Using a switching function approach, the same state equations can be used to create switching models of the system [32]. In this approach, modulated signals like switch duty cycles can be represented in greater detail by their instantaneous binary values rather than by their average values over a period.

Switching models are not necessary for all applications of simulation in power electronic systems, but they do offer distinct advantages for some applications. For example, switching models may be important for representation and testing of fast analog protection systems, for more accurate representation of sampling in digitally controlled converters and especially in converters with high switching ripple, or for peak current-mode-controlled converters [31]. Switching models may also be useful for cases where an extremely slow modulation strategy diminishes the benefits of switch-averaged models or makes them awkward to apply; for example, multilevel converters like MMCs often use

very slow modulation strategies, with just two switch commutations per 50/60Hz AC period. This is discussed in greater detail in Chapter 6.

Since the CF-implicit method works directly on the state equations of the system, and since those state equations are valid for fully switched models using switching functions, no adaptation of the CF-implicit method is theoretically needed for switching model creation. However, without special accommodations, accurate evaluation of a switching model may require a significantly smaller timestep from the model so that the switching dynamics can be accurately captured and the resultant converter behavior accurately represented.

In a switching model, it is no longer possible to consider evaluating a model with one timestep per switching period as was used for the CF-implicit average models in Chapter 4. Moreover, even a fixed timestep ten times smaller than the switching period would likely be wildly inaccurate in a traditional switching model, as the low resolution of ten timesteps per switching period effectively quantizes the duty cycle to values of 0%, 10%, 20%, etc. The naturally high sensitivity of the converter to control inputs like duty cycle implies that the resultant voltage gain of the converter would then also be wildly inaccurate, affecting the whole simulated system. Such quantization error is also a problem in digital control design and can lead to sustained limit cycle oscillations and even full control instability in real systems if the digital control signals and PWM carrier waveforms do not have sufficiently high resolution [33].

Thus, to achieve good accuracy in a fixed-timestep switching model, the chosen timestep must be approximately 3 or 4 orders of magnitude smaller than the switching period of the modeled system. This massive reduction in timestep drives up the

computational cost of the model in direct proportion, eliminating the benefits demonstrated in Chapter 4. The small fixed timestep requirement in switching simulations affects both offline and online simulation applications. Online simulations which run in RT have the most inflexible execution time requirements, and thus face the biggest challenge for switching model simulation, especially of large systems. However, even offline simulations which need not run in RT can be slowed down to the point of extreme impracticality.

One obvious solution for offline solvers is to use a variable timestep solver so that large timesteps can be used when no switching action occurs, but smaller timesteps can be used to give good time resolution around switching events. This solution is certainly better than maintaining an extremely small fixed timestep throughout the entire simulation, but even this solution is suboptimal since switching events occur very frequently throughout the simulation and each one requires a very significant reduction of the timestep for accuracy.

Clever solutions have been proposed throughout the years to simulate switching models with larger timesteps. One popular solution which is adopted in Plexim GmbH's PLECS software is to simulate the system in piecewise linear fashion as a series of linear system simulations (one for each unique combination of switch states). Using this strategy, PLECS takes large timesteps between switching events, monitoring boundary conditions at each step using a zero-crossing detector to ensure that no switching events have rendered the presently selected piecewise sub-model invalid for part of the timestep [34]. Unfortunately, this scheme is complicated to implement in embedded environments since the solver runs on and utilizes MATLAB/Simulink's solver infrastructure and zero crossing

detection, and since zero-crossing events may require the solver to step back and recalculate timesteps when boundary conditions are violated, leading to a higher WCET.

5.1 Semiswitched Models Using the CF-Implicit Method

It is also possible to create computationally efficient, partially switched models using the CF-implicit method proposed here, and to circumvent the need for tiny timesteps by exploiting the dual switching and average modeling capabilities of the switching function. The idea is to operate with all switching function inputs as binary for all timesteps except those timesteps in which a switch commutates. However, timesteps which contain a switch commutation use the average model approach for that input, allowing the switching function to take any value between 0 and 1 to correctly capture the ratio of on-time and off-time during that timestep. This eliminates the need for the timestep to land precisely on the instant of switch commutation, allowing the (mostly) switched model to maintain excellent accuracy with significantly larger timesteps. This approach is similar to that proposed in [35] for a voltage source converter in an HIL simulation context.

Since the proposed models do feature non-binary switching function values during timesteps containing switching instants, they may properly be referred to as "semiswitched" models. The extent to which they replicate the behavior of true switching models depends on the time resolution of the model—that is, the ratio $n = T_{sw}/\Delta t$ of the power electronics switching period to the model timestep. A semiswitched model featuring a timestep which is orders of magnitude smaller than the switching period (corresponding to $n \gg 1$) will behave almost exactly like a pure switching model but will have limited computational benefit over the pure switching model. Moreover, as $n \to \infty$, the error resulting from the small ripple approximation discussed in 4.4.3 decreases and

asymptotically approaches zero. Conversely, a model which uses a timestep relatively close to the switching period of the power electronics will begin to deviate from pure switching model and approach the performance of an average model as $n \to 1$, since the timesteps with non-binary values will begin to take a larger share of the switching period. This spectrum is depicted in Figure 5.1

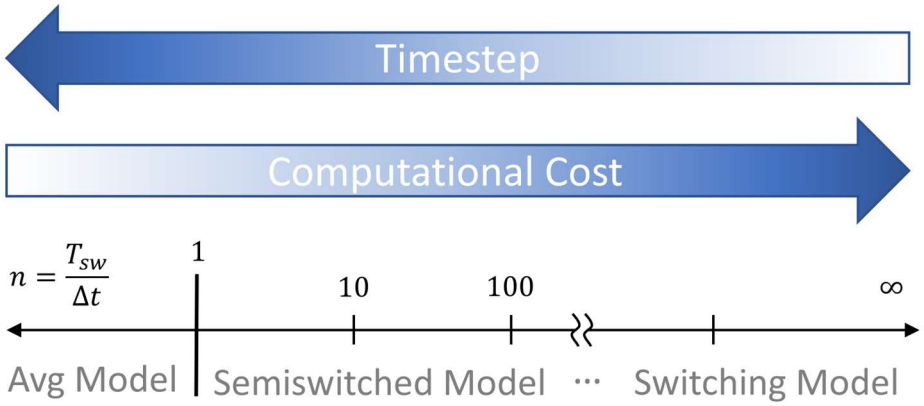

Figure 5.1. Semiswitched models are shown to lie on a spectrum between average models (where typically $\Delta t \geq T_{sw}$) and full switching models (where $\Delta t \ll T_{sw}$).

One outstanding problem that must be addressed to implement semiswitched models is the modulator equivalent model, which is seen to be necessary in the setup of Figure 4.3. This model's output is similar to that of an ordinary PWM modulator, but since the semiswitched modulator must anticipate which timesteps will contain switch commutations and produce an "average" value for that timestep, it requires more complexity than a simple ramp and comparator. A basic structure for the semiswitched equivalent modulator is shown in Figure 5.2.

The concept of the semiswitched modulator in Figure 5.2 is that each time the PWM modulator input changes, a block can forecast and create a vector of its upcoming outputs over the next $n = T_{sw}/\Delta t$ timesteps. The order of the vector elements will change

depending on the carrier waveform of the modulator being emulated, but the average value of the vector elements will always equal the input.

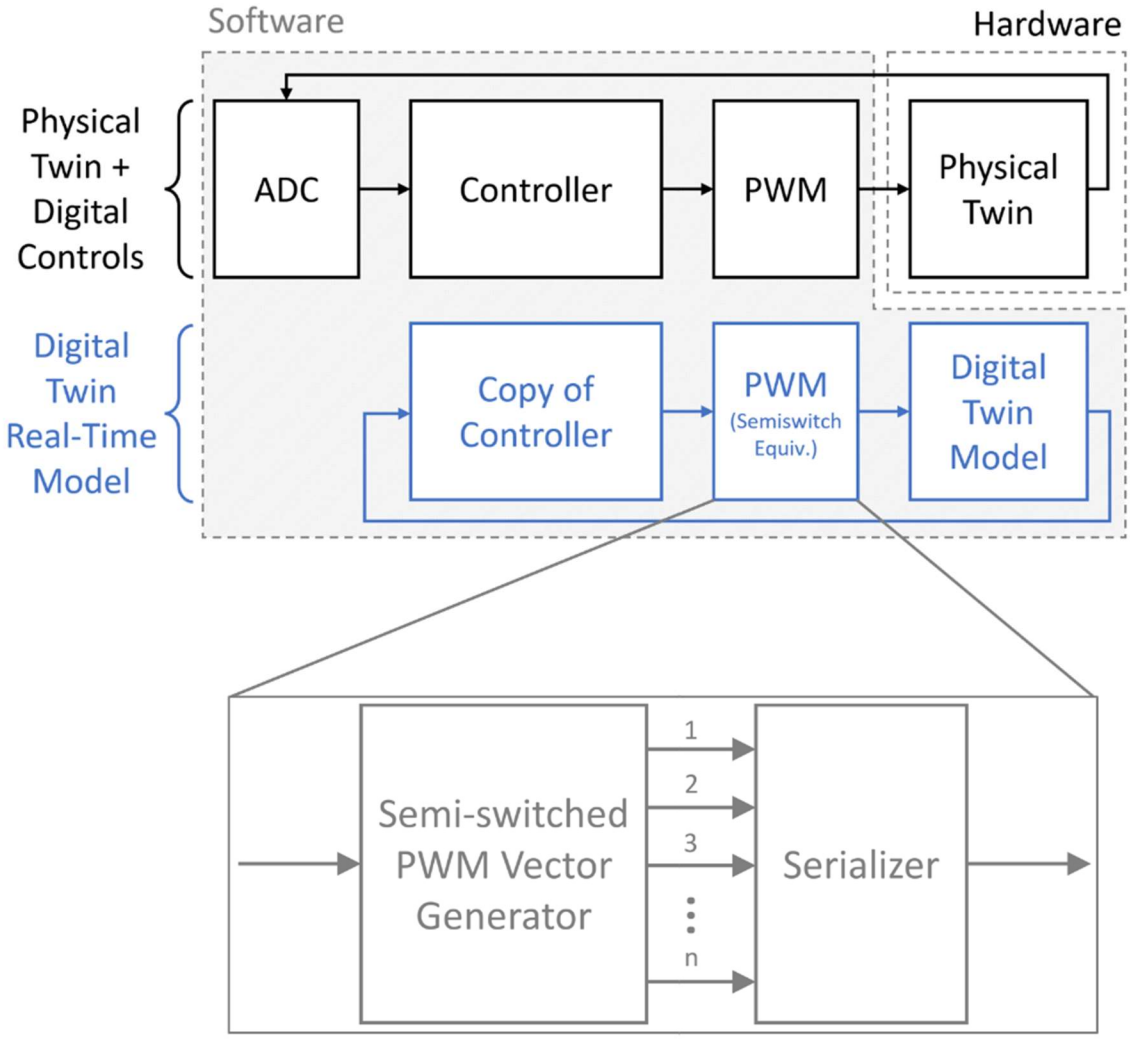

Figure 5.2. An equivalent model for the PWM modulator which is appropriate for semiswitched models.

For example, a sawtooth modulator with an input of 0.65 and $n = 10$ timesteps per switching period would produce a vector

$$p_{sawtooth} = [1\ 1\ 1\ 1\ 1\ 1\ 0.5\ 0\ 0\ 0]^T \tag{5.1}$$

While an inverse sawtooth equivalent modulator would produce

$$p_{invsaw} = [0\ 0\ 0\ 0.5\ 1\ 1\ 1\ 1\ 1\ 1]^T \tag{5.2}$$

And a triangular carrier equivalent modulator would produce

$$p_{triangular} = [1\ 1\ 1\ 0.25\ 0\ 0\ 0.25\ 1\ 1\ 1]^T \tag{5.3}$$

This vectorization is easily programmed for any possible carrier waveform in the "Semi-switched PWM Vector Generator" of Figure 5.2. A subsequent "Serializer" block sequentially feeds through the vector elements, using one each timestep.

5.2 Validation of Semiswitched CF-Implicit Models

The proposed modeling approach is here demonstrated for two values of $n = T_{sw}/\Delta t$ on a boost converter alongside a traditional switching model from the Simulink Specialized Power Systems library blockset. The topology is shown in Figure 5.3. Each converter model has identical parameters as given in Table 5.1 and is subjected to identical input voltages and duty cycles. The results of the simulation are shown in Figure 5.4, and show excellent agreement.

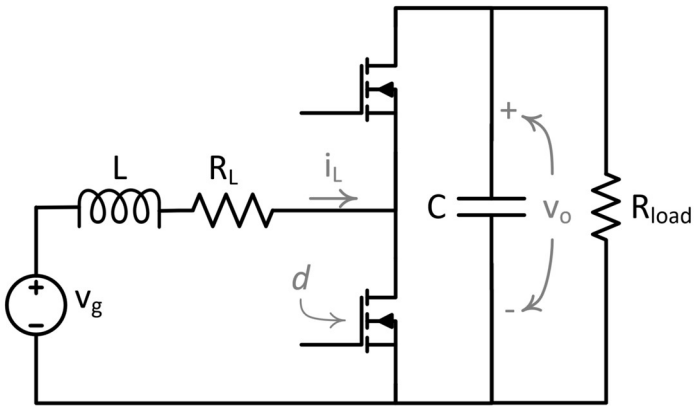

Figure 5.3. The boost converter modeled using the CF-implicit semiswitched methods.

Table 5.1. Parameter Values for Boost Converter

Parameter	Value
C	$260\ \mu F$
L	$2.5\ mH$
R_L	$40\ m\Omega$
R_{load}	$30\ \Omega$

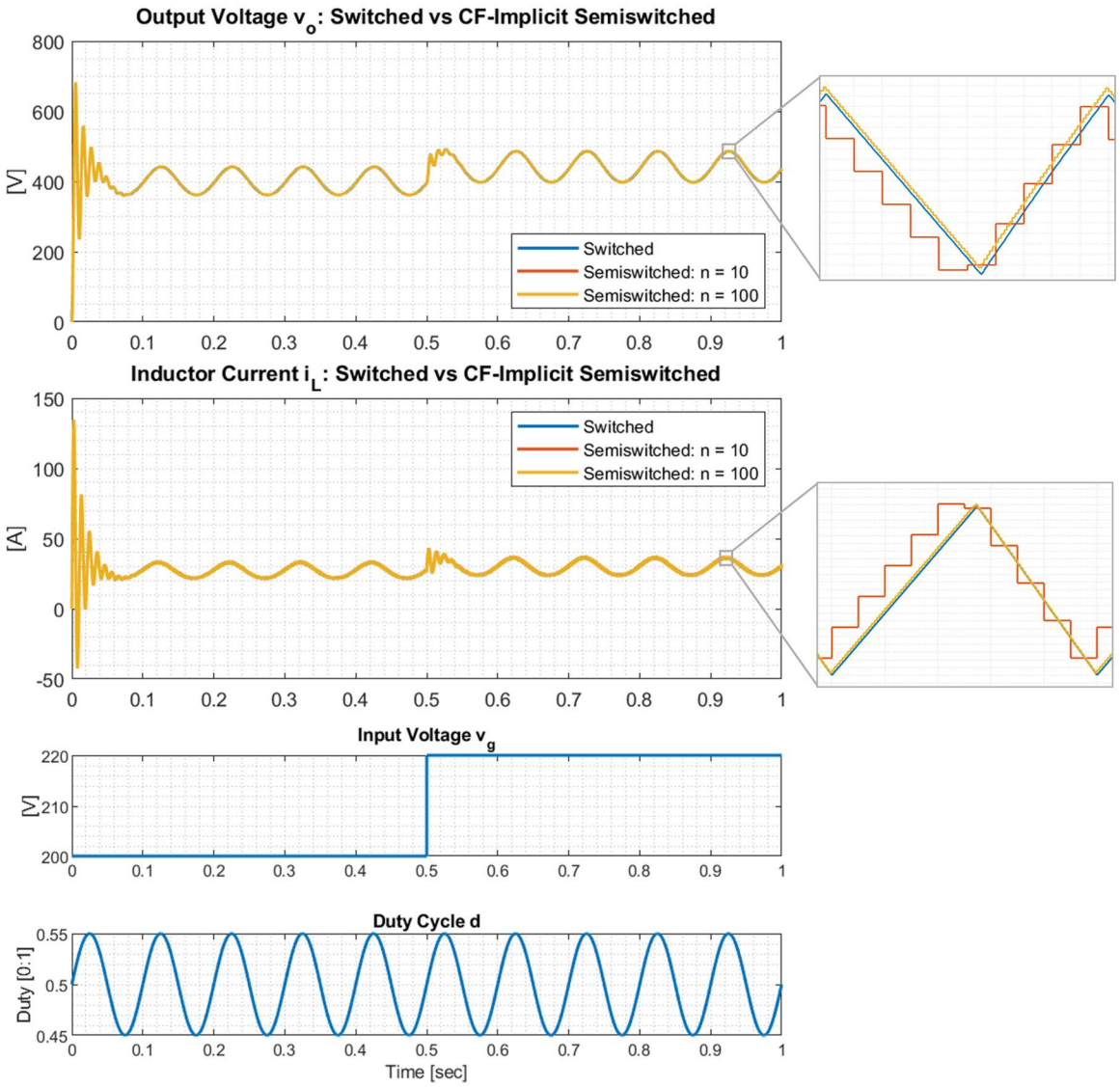

Figure 5.4. Simulation results comparing a traditional fully switched boost converter model to two CF-Implicit semiswitched models with timestep-to-switching-period ratios of $n = 10$ and $n = 100$.

The waveforms of Figure 5.4 also demonstrate the continuum depicted in Figure 5.1; notice that the semiswitched model with $n = 100$ timesteps per switching period more closely approximates the behavior of the pure switching model, and while the semiswitched model with n=10 timesteps per switching period does contain some approximation of switching ripple but does so with lower resolution and fidelity. Simulation models with less than $n = 10$ timesteps per switching period quickly lose enough resolution that they

cease to meaningfully approximate switching behavior, and morph gradually toward a fully switch-averaged model as n approaches 1.

CHAPTER 6

CF-IMPLICIT MODELS FOR TOPOLOGIES OF SPECIAL INTEREST

6.1 MODULAR MULTILEVEL CONVERTER (MMC)

Since currently available power transistors are rated only up to 10kV in a single device, simple two-level converters like the one presented in the previous chapter are only usable in low-voltage applications. Because of this limitation, multilevel converters (including MMCs) are growing in importance as power electronic converters are inserted into more medium/high voltage and high power applications. Multilevel topologies are of interest for utility and industrial applications, as well as high-power naval, maritime, and defense platforms. For example, the US Navy is actively pursuing new designs for an all-electric ship which will use a medium-voltage DC power architecture which would benefit significantly from the use of multilevel converters for machinery drives and energy storage systems [36], [37].

MMCs were first proposed in the early 2000s in a German patent [38] and several subsequent papers [39] [40], and have rapidly grown in popularity for multilevel applications due to advantages like high reliability, inherent redundancy, reasonable cost, and very high (typically ~99%) efficiency [41]. Figure 6.1 shows a generic diagram of a three-phase MMC having n_{SM} total submodules in each armature. An upper and lower armature each connect to inductors L_U and L_L, respectively, together constituting one phase leg of the MMC. Several different types of submodules can be used, but the most

commonly used submodules are half-bridge and full-bridge modules [42], [43]; these are shown in Figure 6.2. The number of submodules per armature n_{SM} varies greatly with the application of the converter; low voltage systems may have only a few submodules per armature, while MMCs in HVDC applications commonly have more than 200 submodules in each armature.

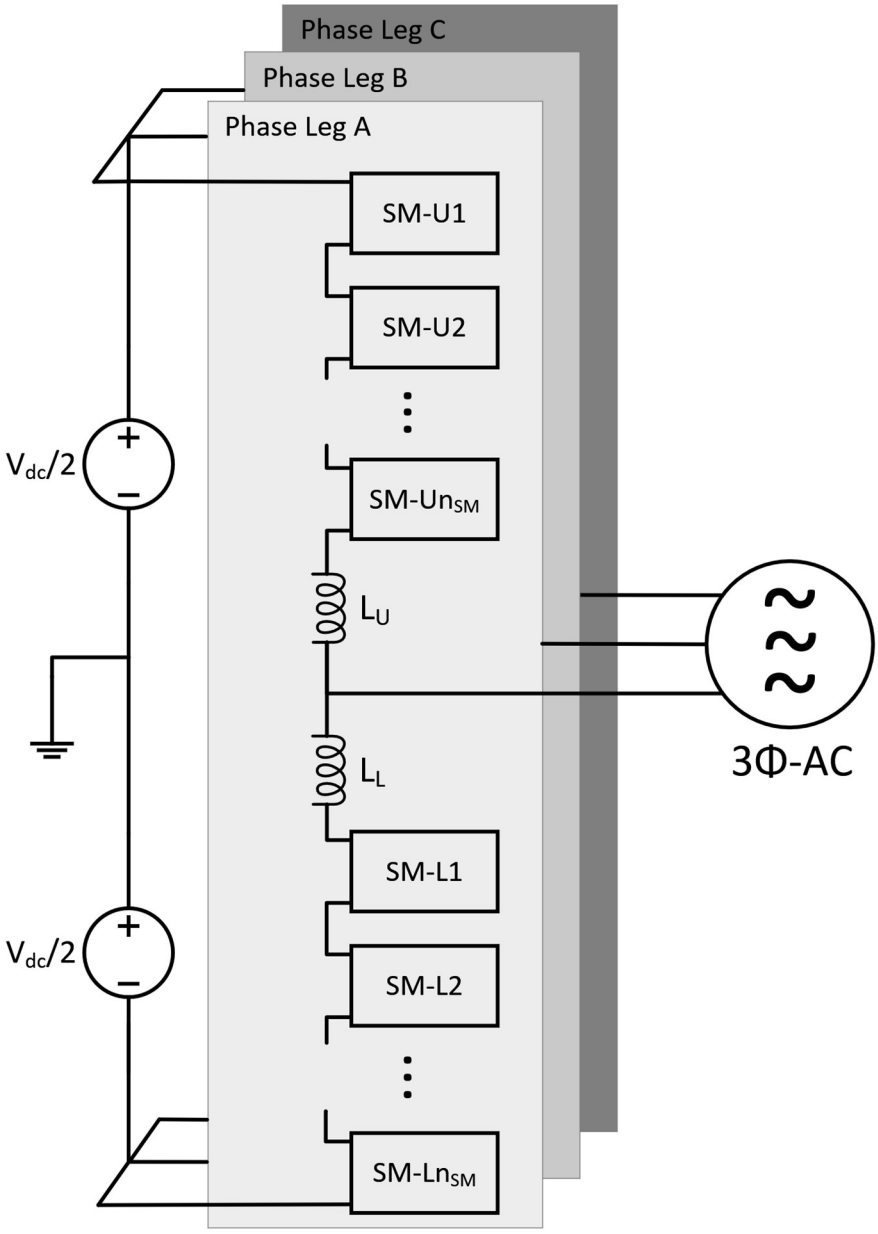

Figure 6.1. A high-level view of a three-phase-leg MMC with n_{SM} submodules in each armature.

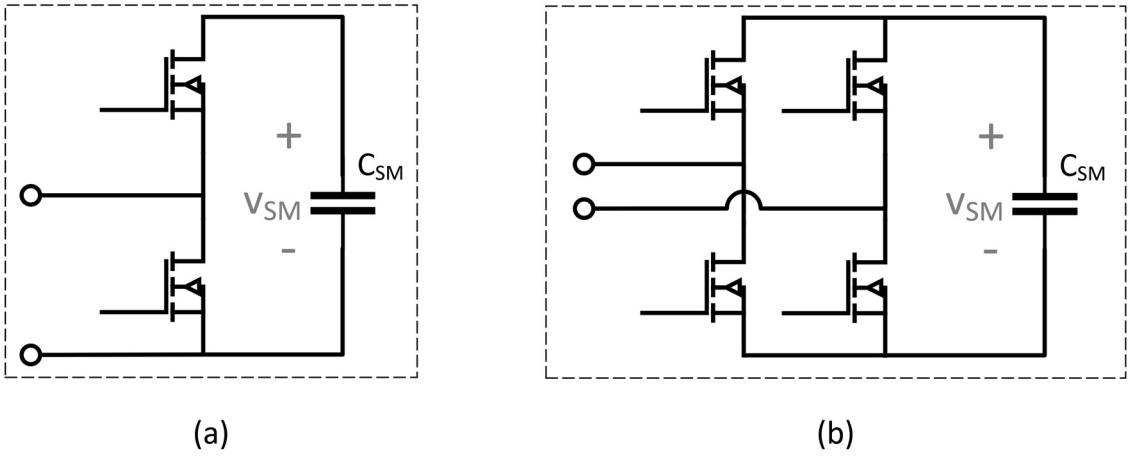

Figure 6.2. The most common submodule circuits used in MMCs: half bridge (a) and full bridge (b).

Multilevel converters like MMCs are of special interest for this work not only because they are necessary for high voltage and high power applications, but also because they are excellent candidates for integration of a digital twin. Multilevel converters tend to have complex multi-objective controllers which must ensure voltage balance among the levels while also ensuring good current/voltage regulation on the output. They also have a large number of control inputs and degrees of freedom available, but there are many potential points of failure and many devices whose health may degrade over time [44]. All of this leads to a system which can be complicated to analyze and manage throughout its life cycle, even though the modular construction lends itself robust performance in the presence of submodule failures [45]. The digital twin concept is well-suited to complex systems like this and promises to provide significant value for control, health management, and optimization. However, simulation models of MMCs can be prohibitively large and complex due to the large size of the converter, leading to very long simulation times [46]; thus, the CF-implicit method could be especially helpful in this application to reduce the computational complexity of the model and allow for RT execution.

6.1.1 Overview of Modulation Strategies for MMCs

There are a wide variety of modulation strategies for MMCs, each of which have benefits and drawbacks. The choice of modulator has a drastic effect on the performance of the converter. Modulation strategies can be classified by switching frequency, with methods for low, medium, and high frequency modulation [47]. Low frequency methods offer significantly lower losses (due to a strong reduction in the number of switch commutations) but suffer from relatively poor performance, with high harmonic distortion and imprecise regulation of the output waveforms [48]. High frequency modulation strategies reduce harmonic distortion and allow for precise regulation of the converter states at the cost of increased switching losses and higher computational complexity.

Low frequency methods can entail as few as one pair of switch commutations per submodule in each fundamental period of the AC waveform. The simplest and most common low frequency modulation strategy is nearest level control (NLC), which produces a staircase waveform approximating a sinusoid by choosing an integer number of modules to be "on" at any given time [49]. Submodule capacitor voltage balance can be maintained in NLC methods by using a sorting algorithm to determine the order in which submodules should turn on based on their measured voltages and the sign of the armature currents [48]. Since the number of levels in the MMC determines the granularity of the achievable output voltage waveform, NLC is generally most feasible for large MMCs with a large number of levels; MMCs with relatively few levels will produce output waveforms with very high harmonic distortion if NLC is used.

Medium and high frequency modulation strategies commonly entail some form of carrier-based pulse width modulation, in which one or more modulated control signals are

compared against n_{SM} separate carrier waveforms. These carrier waveforms can be phase-shifted, level-shifted, or both [43]; however, phase-shifted PWM methods are generally preferred because level-shifted modulation strategies unevenly distribute submodule capacitor voltage ripple and can induce larger loss-inducing circulating currents. Modifications are required for level-shifted methods to ensure capacitor voltage balancing [46].

Space vector modulation can also be used for MMC applications, but its complexity grows exponentially with the number of levels of the MMC [42], which is generally problematic for large MMCs in high voltage applications; nonetheless, some solutions have been proposed to address these shortcomings [50] [51] [52], and optimal control and modulation strategies for MMCs remains an active research topic.

Whatever the choice of modulation strategy is, it should be properly accounted for when creating a CF-implicit digital twin model (or any other type of model) of an MMC. Note from Figure 4.3 that the modulator or some switch-averaged equivalent of it must be included in the digital twin. The appropriate equivalent model of the modulation may vary depending on the chosen strategy.

6.1.2 CF-Implicit Model of an MMC

Figure 6.3 shows a five-level ($n_{SM} = 4$), single-phase-leg, half-bridge MMC that we will use here as a representative system for a CF-implicit model derivation. The modeling method is employed in a scalable way so that systems with more levels and multiple phase-legs can easily be created using the same approach.

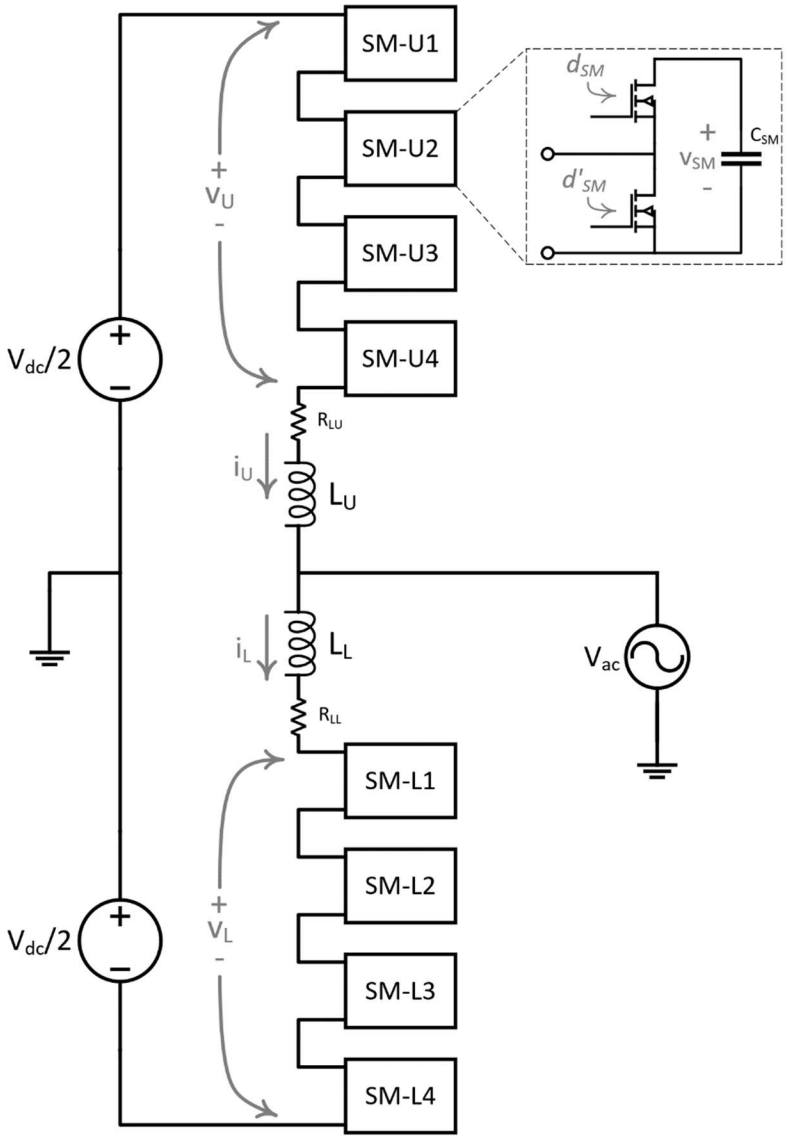

Figure 6.3. A five-level, single-phase-leg MMC with half-bridge modules can be used as a representative system for modeling studies.

The state equations of the upper and lower armature inductor currents are:

$$L_U \frac{di_U}{dt} = \frac{v_{dc}}{2} - i_U R_{LU} - v_{ac} - \sum_{i=1}^{n_{SM}} d_{SM-Ui} v_{SM-Ui} \tag{6.1}$$

$$L_L \frac{di_L}{dt} = \frac{v_{dc}}{2} - i_L R_{LL} + v_{ac} - \sum_{i=1}^{n_{SM}} d_{SM-} \quad v_{SM-L} \tag{6.2}$$

And, for submodules in the upper and lower armatures, the capacitor voltage states are given as:

$$C_{SM-Ui}\frac{dv_{SM-Ui}}{dt} = d_{SM-Ui}i_U, \qquad \forall i \in \{1, \dots, n_{SM}\} \tag{6.3}$$

$$C_{SM-L}\frac{dv_{SM-Li}}{dt} = d_{SM-Li}i_L, \qquad \forall i \in \{1, \dots, n_{SM}\} \tag{6.4}$$

Thus, defining the state vector as

$$x = \begin{bmatrix} i_U \\ i_L \\ v_{SM-U1} \\ \vdots \\ v_{SM-Un} \\ v_{SM-L1} \\ \vdots \\ v_{SM-Ln} \end{bmatrix} \tag{6.5}$$

And the input vector as

$$u = \begin{bmatrix} v_{dc} \\ v_{ac} \\ d_{SM-U1} \\ \vdots \\ d_{SM-Un} \\ d_{SM-L1} \\ \vdots \\ d_{SM-Ln} \end{bmatrix} \tag{6.6}$$

We can formulate the matrix functions g and h as:

$$g(u) = \begin{bmatrix} -\frac{R_{LU}}{L_U} & 0 & -\frac{d_{SM-U1}}{L_U} & \cdots & -\frac{d_{SM-Un}}{L_U} & 0 & 0 & 0 \\ 0 & -\frac{R_{LL}}{L_L} & 0 & 0 & 0 & -\frac{d_{SM-L1}}{L_L} & \cdots & -\frac{d_{SM-Ln}}{L_L} \\ \frac{d_{SM-U1}}{C_{SM-U1}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{d_{SM-Un}}{C_{SM-Un}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{d_{SM-L1}}{C_{SM-L1}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \vdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{d_{SM-L}}{C_{SM}} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.7}$$

$$h(u) = \begin{bmatrix} \frac{v_{dc}}{2L_U} - \frac{v_{ac}}{L_U} \\ \frac{v_{dc}}{2L_L} + \frac{v_{ac}}{L_L} \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{6.8}$$

We elected to modulate the MMC using nearest level control (NLC) to simplify the control structure and to show how semiswitched CF-implicit methods (introduced in Chapter 5) can be employed in a large complex converter. Note that "switch-averaged" modeling methods are useful only when the converter switching frequency is high with respect to the dynamics of the converter and are not useful in the case of NLC modulation, where submodules are commutated only twice per fundamental AC period. Thus, a switching model (or something closely approximating it) is needed for this example. Using a semiswitched CF-implicit model allows us to accurately simulate the model while maintaining a large timestep for speed and computational efficiency.

The block diagram of the open loop NLC modulator implemented in the simulation is shown in Figure 6.4. Moving from left to right in the figure, the modulator operates as follows: First, a perfect sinusoidal reference voltage V_m is created. Then, using the submodule capacitor voltages, the levels and their respective midpoints are calculated. Having defined both the input reference and the midpoints between each level, the intersection times of the midpoints and reference can be calculated to provide the desired switch commutation time instants $t_1 \ldots t_{n_{SM}}$. From these times, a switching function matrix S_{SwFn} is created whose columns correspond to timesteps in the AC period T_{ac} and whose rows correspond each submodule in the armature. For a traditional switching model, the

matrix S_{SwFn} will be purely Boolean, since switches must be fully on or off. Contrarily, for a semiswitched model, the matrix will be mostly Boolean but may contain non-Boolean values at the timestep which contains a switch commutation. Since each row of S_{SwFn} provides the switching function for one submodule over one AC period, the rows can be sorted by the submodule capacitor voltages v_{SM} and ordered to maintain voltage balance. Submodules with the lowest voltage will then receive the most charge for the given AC period, while submodules with the highest voltage will be discharged the most. Finally, the sorted rows are each fed into a serializer which outputs the switching function values sequentially at each timestep in the AC period.

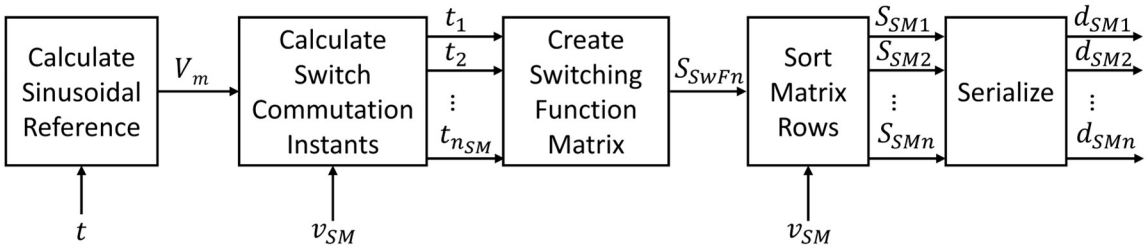

Figure 6.4. A functional block diagram of the components of the semiswitched modulator implemented in the MMC example.

The result of this modulator construction is shown in Figure 6.5 for a traditional switching model (top) and two variations of a semiswitched model (middle and bottom). The plots each show the switching functions for each upper armature submodule in the single phase MMC of Figure 6.3. As in the previously presented semiswitched models, the switching model and the semiswitched model are identical except for those timesteps which contain switch commutations—here the semiswitched models use an average value for one timestep to preserve accuracy while allowing a larger timestep. The middle plot uses a timestep of $\Delta t = T_{ac}/100 \ = 166.7\,\mu s$, and therefore offers much higher time resolution than the lower plot with a timestep of $\Delta t = T_{ac}/10 = 1.667\,ms$.

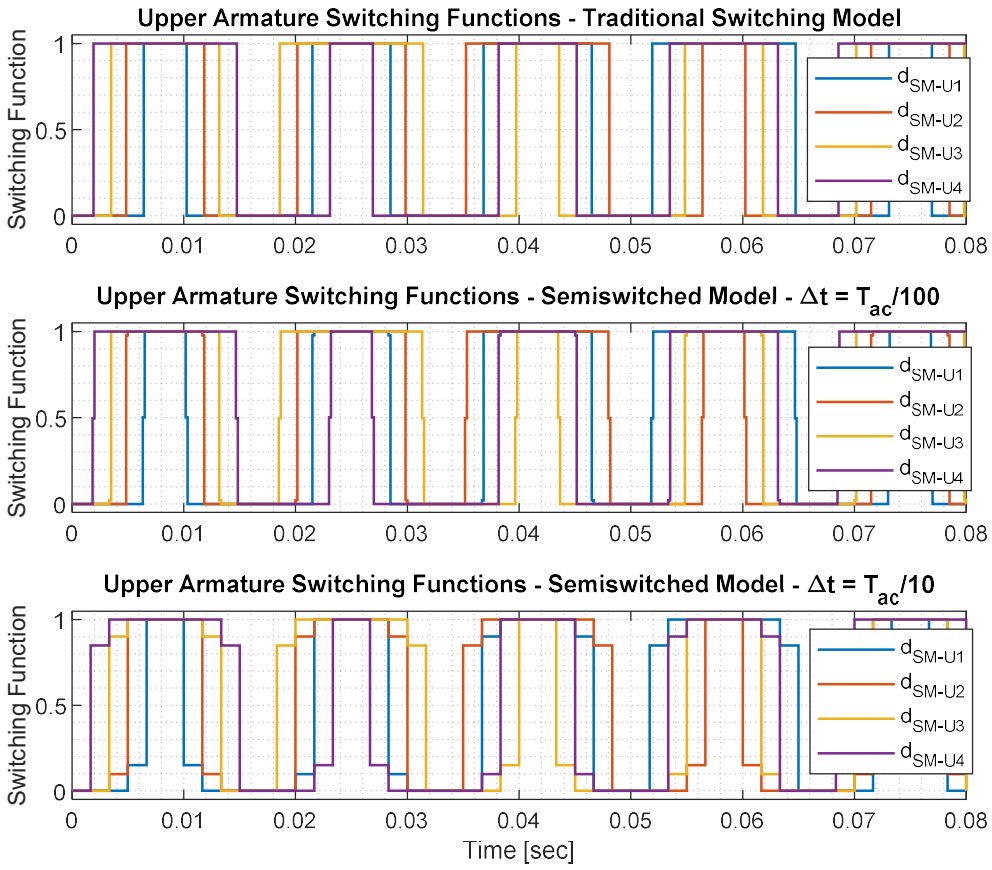

Figure 6.5. The output of the MMC NLC modulator for a pure switching model (top), a semiswitched model with 100 timesteps per AC period (middle) and a semiswitched model with 10 timesteps per AC period.

The switching modulator was paired with a traditional switching model in Simulink's Specialized Power Systems library in Simscape, and the two semiswitched modulators were each paired with CF-implicit models produced by plugging the matrix functions in (6.7) and (6.8) into (3.17). Each model was simulated with a constant input voltage of $v_{dc} = 2000V$ and with the passive component parameters given in Table 6.1.

Table 6.1. Parameter Values for MMC Converter

Parameter	Value
V_{dc}	$2\ kV$
$V_{ac,peak}$	$1\ kV$
f_{ac}	$60\ Hz$
$V_{SM,init}$	$500\ V$
n_{SM}	4
C_{SM}	$10\ mF$
L_U, L_L	$25\ mH$
R_{LU}, R_{LL}	$100\ m\Omega$

The results of the three simulations are overlaid and shown together in Figure 6.6 and Figure 6.7. Specifically, Figure 6.6 shows the armature currents i_U and i_L, and Figure 6.7 shows the upper armature submodule capacitor voltages. The results show excellent agreement between the switching model and the semiswitched model with 100 timesteps per period. However, the semiswitched model with only 10 timesteps per period is significantly less accurate and leads to offsets in the submodule capacitor voltages as well as amplitude errors in the armature and output currents because the timestep is too large even for a semiswitched CF-implicit model. The conclusion is therefore that semiswitched CF-implicit methods are an excellent, computationally efficient alternative for simulating MMCs (and AC converters more broadly) but that the acceptable timestep for such systems must still be more than one order of magnitude below the fundamental AC period of the system.

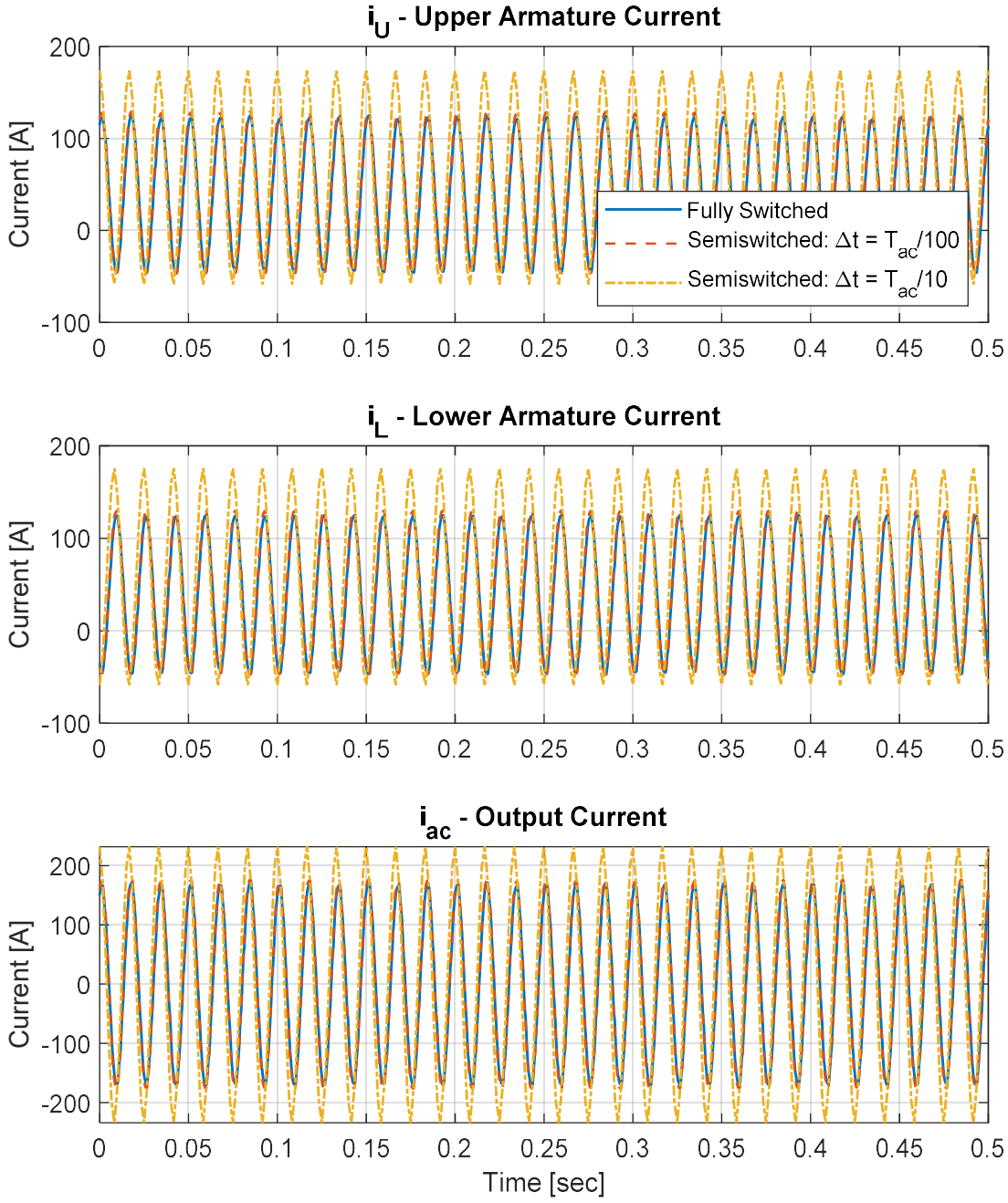

Figure 6.6. Upper armature, lower armature, and output AC waveforms for the simulated single-phase five level MMC are shown for the traditional fully switched model (blue), the semiswitched model with 100 timesteps per AC period (red), and the semiswitched model with 10 timesteps per AC period (yellow).

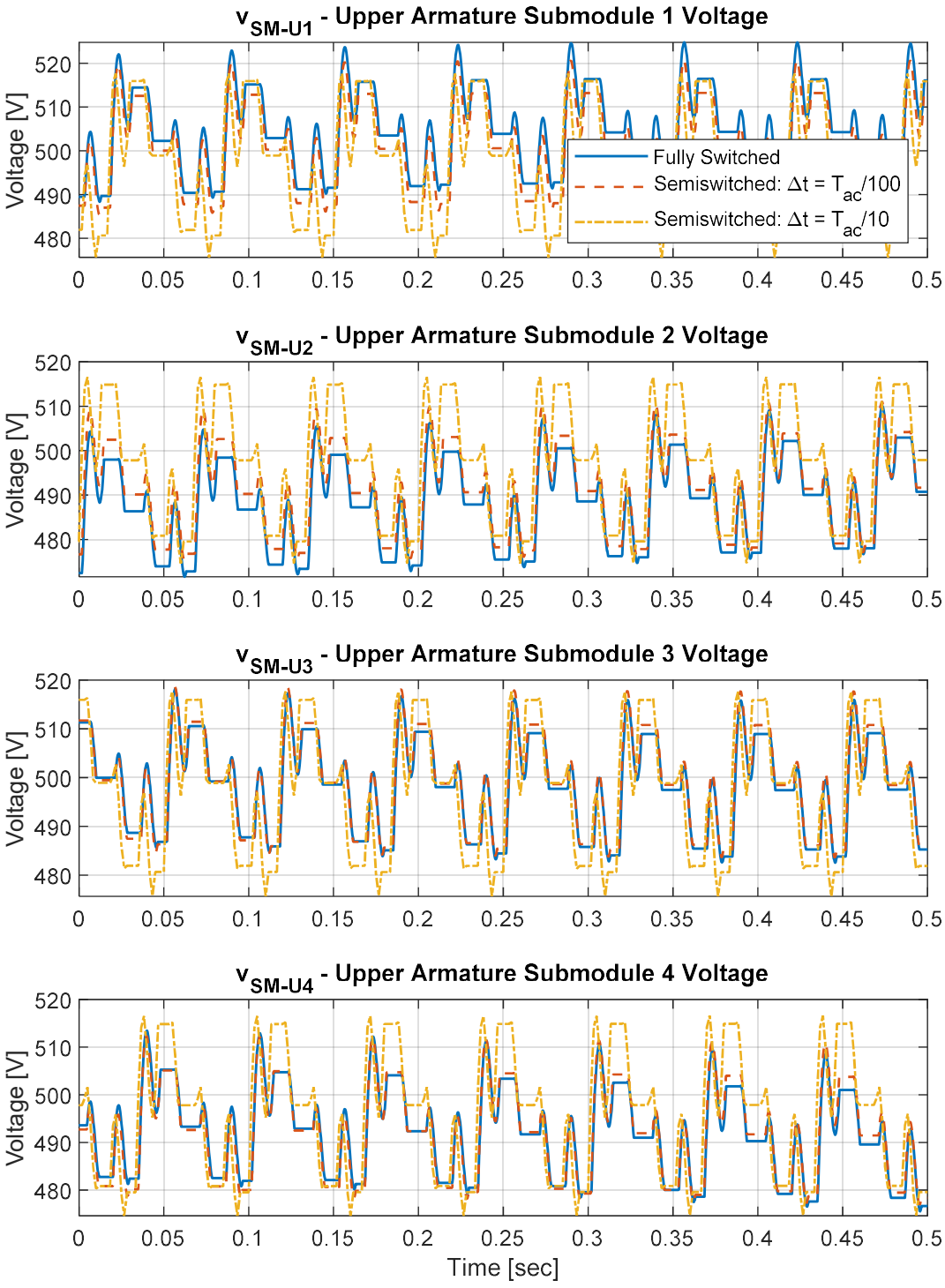

Figure 6.7. Upper armature submodule voltage waveforms for the simulated single-phase five level MMC are shown for the traditional fully switched model (blue), the semiswitched model with 100 timesteps per AC period (red), and the semiswitched model with 10 timesteps per AC period (yellow).

6.2 DUAL ACTIVE BRIDGE (DAB) CONVERTER

Another topology of special interest is the Dual Active Bridge converter, which was first proposed and patented in 1991, but has become quite popular over the last decade [53]. The basic structure is shown in Figure 6.8. The DAB is popular because it offers high power density, high efficiency, guaranteed soft switching on all devices (for some operating conditions), galvanic isolation, and seamless bidirectional power flow due to complete input/output symmetry [54].

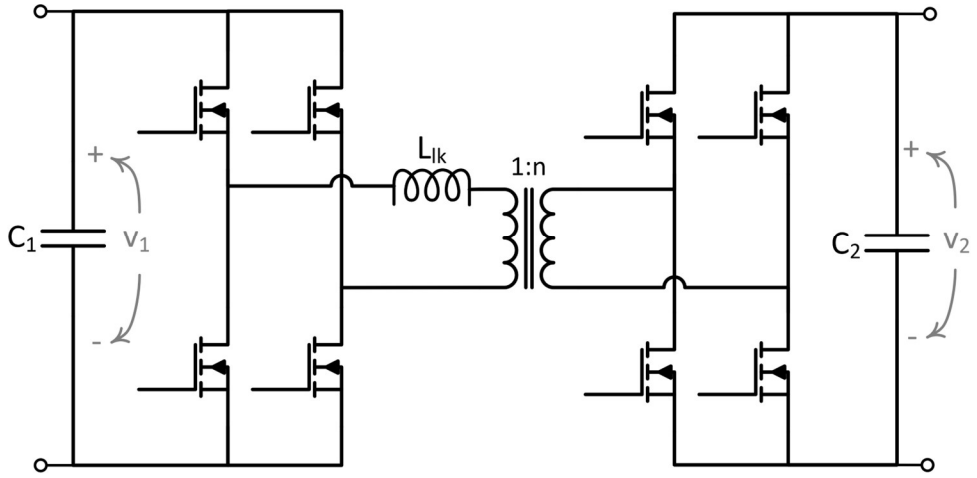

Figure 6.8. The dual active bridge converter topology. In this schematic, L_{lk} represents the total primary-referred leakage inductance of the transformer. The magnetizing inductance and all other parasitics are not shown.

The dual active bridge is unique among DC/DC converters because the power transfer is accomplished entirely via high-frequency AC power transfer across the central transformer. Since there is ideally no DC component at all, any analysis which uses the widely accepted "small ripple approximation" that is used for other converter topologies is wholly unsuitable. Quite a few papers have aimed to address this problem using nontraditional analysis techniques such as generalized average modeling, but these models are typically complicated and unintuitive [55], [56], [57].

The DAB would at first appear to be at least a second-order system (on account of the inductance and capacitance of the transformer and output stage). Surprisingly, however, the generalized average modeling approach has confirmed that the dynamic behavior of the entire converter (e.g., the control-to-output frequency response) is almost perfectly first-order. Effectively, the dynamics of the leakage inductance are not seen at any frequency below the switching frequency, because the operation of the converter ensures that its current is reset to zero each switching cycle. Fundamentally, therefore, the inductor current does not act as a state in a switch-averaged model of the DAB, as shown in [58]. It is therefore possible to create a static equivalent model of the transformer and switches without any serious loss of dynamic accuracy since the switch-averaged dynamics of the system will be fully determined by the capacitors and external sources and loads.

Here we create a much simpler model of the dual active bridge which retains the dominant dynamic features of DAB converters while circumventing the complication of previously proposed approaches. This simpler model retains the high accuracy of previously proposed approaches. Moreover, the simpler model lends itself to implementation using the proposed CF-implicit method and all the associated stability and execution speed benefits.

6.2.1 Transconductance-Based Dynamic Model of the DAB

Basic power transfer analysis (as shown in [53]) for the ideal DAB converter under the most common modulation strategy—single-phase-shift (SPS) modulation—reveals that the power transferred across the converter is given by:

$$P = \frac{v_1 v_2}{2\pi n f_s L_{lk}} \phi \left(1 - \frac{\phi}{\pi}\right) \tag{6.1}$$

Here v_1 and v_2 are the input and output voltages, n is the transformer turns ratio, ϕ is the phase shift between the primary and secondary full bridges, f_s is the switching frequency, and L_{lk} is the total leakage inductance of the transformer referred to the primary side. It clearly follows that the average current drawn from the primary side full-bridge stage must be:

$$i_1 = \frac{P}{v_1} = \frac{v_2}{2\pi n f_s L_{lk}} \phi \left(1 - \frac{\phi}{\pi}\right) \tag{6.2}$$

Likewise, the current injected into the output capacitor and load by the secondary side full-bridge is:

$$i_2 = \frac{P}{v_2} = \frac{v_1}{2\pi n f_s L_{lk}} \phi \left(1 - \frac{\phi}{\pi}\right) \tag{6.3}$$

These currents can be imposed directly in the model using two controlled current sources which draw and inject current from input and output sides. This approach is similar to that used in [59], but allows for modeling the input-side dynamics and shows explicitly the relationship between the two controlled current sources. In this model, the two controlled sources replace the entire central portion of the DAB converter—including the transformer and all semiconductor switches. The result, shown in Figure 6.9, is a fully switch-averaged model of the DAB which can be executed much faster than a traditional model. Note that the function $g_m(\phi)$ referenced in the figure is a transconductance given by:

$$g_m(\phi) = \frac{1}{2\pi n f_s L_{lk}} \phi \left(1 - \frac{\phi}{\pi}\right) \tag{6.4}$$

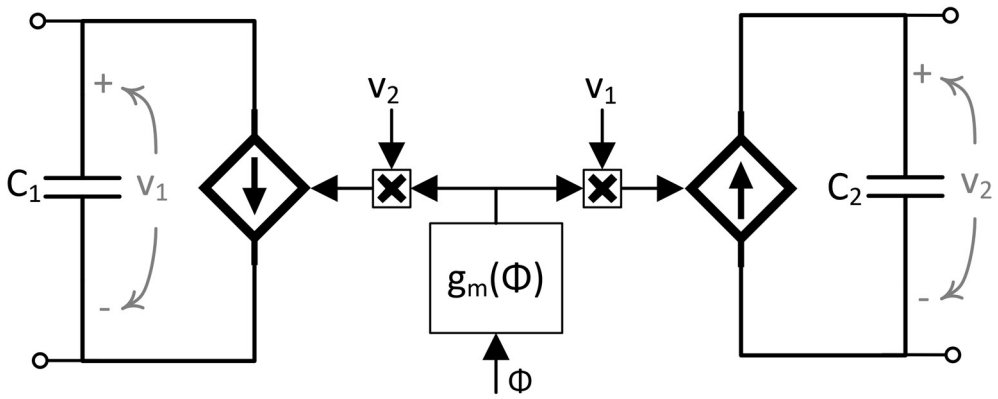

Figure 6.9. The equivalent model of the dual active bridge converter replaces the switches and high-frequency transformer with controlled current sources and a nonlinear function of the phase shift between primary and secondary bridges.

6.2.2 CF-Implicit Implementation of the Transconductance Model

A dynamic analysis of the equivalent circuit model of Figure 6.9 involves writing KCL equations for the input and output stages on the left and right sides, respectively. Focusing on the output stage:

$$C_2 \frac{dv_2}{dt} = i_2 - i_o \tag{6.5}$$

Assuming again a resistive load for purposes of analysis, we can make substitutions for currents i_2 and i_o to obtain:

$$C_2 \frac{dv_2}{dt} = v_1 g_m(\phi) - \frac{v_2}{R_{load}} \tag{6.6}$$

And substituting the definition of $g_m(\phi)$ from (5.4):

$$C_2 \frac{dv_2}{dt} = \frac{\phi\left(1 - \frac{\phi}{\pi}\right)}{2\pi n f_s L_{lk}} v_1 - \frac{v_2}{R_{load}} \tag{6.7}$$

The analysis of the input side is extremely similar, due to the inherent symmetry of the converter. The KCL analysis of the input side yields:

$$C_1 \frac{dv_1}{dt} = i_{in} - i_1 \tag{6.8}$$

Substituting for i_1:

$$C_1 \frac{dv_1}{dt} = i_{in} - v_2 g_m(\phi) \tag{6.9}$$

And substituting for g_m from (5.4):

$$C_1 \frac{dv_1}{dt} = i_{in} - \frac{\phi\left(1 - \frac{\phi}{\pi}\right)}{2\pi n f_s L_{lk}} v_2 \tag{6.10}$$

Note that the input side dynamics appear only when a nonideal source is connected at the input (i.e., a source with a Thevenin impedance, or another converter or circuit). Alternatively, if an ideal voltage source is placed at the input—as is frequently considered in simplified models of power electronics—then capacitor C_1's voltage is fixed, and its dynamics disappear from the model.

Note also that (6.7) and (6.10) do not fit the structure defined in (3.11) if we consider ϕ as the modulated control input in the vector d, because (3.11) does not allow for quadratic dependence on elements of d, and expanding (6.7) and (6.10) clearly produces terms with ϕ^2. Since (3.11) is not sufficient, the more general form of (3.14) can be used in its place.

To create a CF-implicit model of the DAB we must define the source and load connected on each side of the converter. To observe the full dynamics of the converter, we terminate it with a resistive load R_{load} and a Thevenin equivalent voltage source v_s with resistance R_s and inductance L_s. The schematics of the converter and its equivalent transconductance-based model are shown in Figure 6.10 and Figure 6.11, and the parameter values for the model are given in Table 6.2.

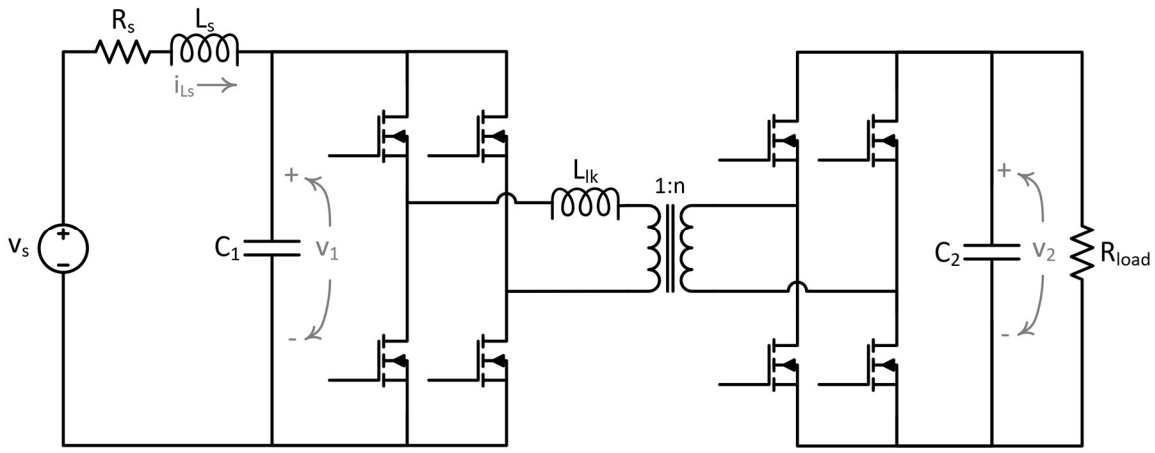

Figure 6.10. The schematic of the DAB converter which is used to validate the accuracy of the proposed modeling approaches.

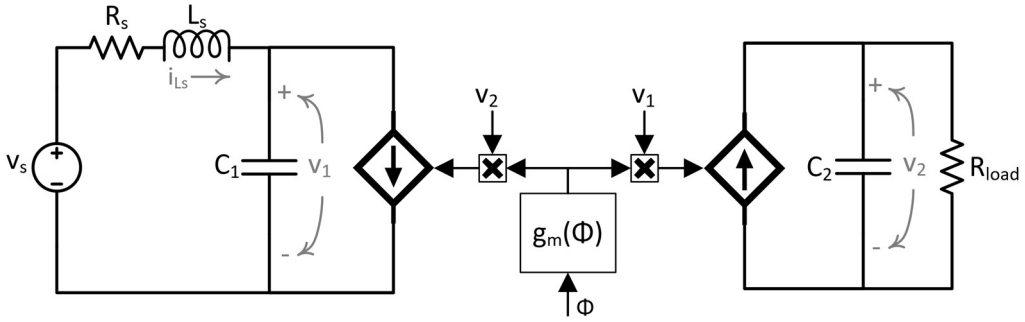

Figure 6.11. The schematic of the transconductance-based equivalent model of the DAB converter replaces the switches and transformer with a pair of controlled current sources.

Table 6.2. Parameter Values for DAB Converter

Parameter	Value
C_1	$500\ \mu F$
C_2	$100\ \mu F$
L_{lk}	$27.2\ \mu H$
L_s	$1\ mH$
R_s	$0.5\ \Omega$
n	1
f_s	$50\ kHz$

As shown, the converter is modeled as third-order system, with state variables i_{Ls}, v_1, and v_2. The current in inductor L_{lk} is not considered as a state variable because the converter operation dictates that the current is reset to zero during each switching cycle;

accordingly, the switch-averaged transconductance-based model and the CF-implicit model have no state associated with the leakage inductance. The derivative-form state equations for the system are:

$$\frac{di_{Ls}}{dt} = \frac{1}{L_s}(v_s - i_{Ls}R_s - v_1) \tag{6.11}$$

$$\frac{dv_1}{dt} = \frac{1}{C_1}\left(i_{Ls} - \frac{\phi\left(1-\frac{\phi}{\pi}\right)}{2\pi n f_s L_{lk}} v_2\right) \tag{6.12}$$

$$\frac{dv_2}{dt} = \frac{1}{C_2}\left(\frac{\phi\left(1-\frac{\phi}{\pi}\right)}{2\pi n f_s L_{lk}} v_1 - \frac{v_2}{R_{load}}\right) \tag{6.13}$$

The state variable vector x therefore is given by:

$$x = \begin{bmatrix} i_{Ls} \\ v_1 \\ v_2 \end{bmatrix} \tag{6.14}$$

And the vectors for system inputs and modulated controller inputs are single element variables given by:

$$u = [v_s] \tag{6.15}$$

$$d = [\phi] \tag{6.16}$$

Combing (6.11)-(6.13) into a single matrix equation of the form of (3.14), the g and h functions are seen to be:

$$g(u,d) = \begin{bmatrix} -\frac{R_s}{L_s} & -\frac{1}{L_s} & 0 \\ \frac{1}{C_1} & 0 & -\frac{1}{C_1}\frac{\phi\left(1-\frac{\phi}{\pi}\right)}{2\pi n f_s L_{lk}} \\ 0 & \frac{1}{C_2}\frac{\phi\left(1-\frac{\phi}{\pi}\right)}{2\pi n f_s L_{lk}} & -\frac{1}{C_2 R_{load}} \end{bmatrix} \tag{6.17}$$

$$h(u,d) = \begin{bmatrix} \frac{v_s}{L_S} \\ 0 \\ 0 \end{bmatrix} \tag{6.18}$$

Using these functions, the CF-implicit model of the system is generated via MATLAB's Symbolic Math Toolbox for EB integration using (3.17). All three models—the CF-implicit model, the transconductance-based model of Figure 6.11, and a switching model of Figure 6.10—are simulated and subjected to identical step function perturbations in the input voltage, load resistance, and phase shift ϕ. The simulation results in Figure 6.12 confirm outstanding agreement between each of the three models, showing that no accuracy was lost in the equivalent transconductance-based model or in the explicitly solved, implicitly integrated model formulation.

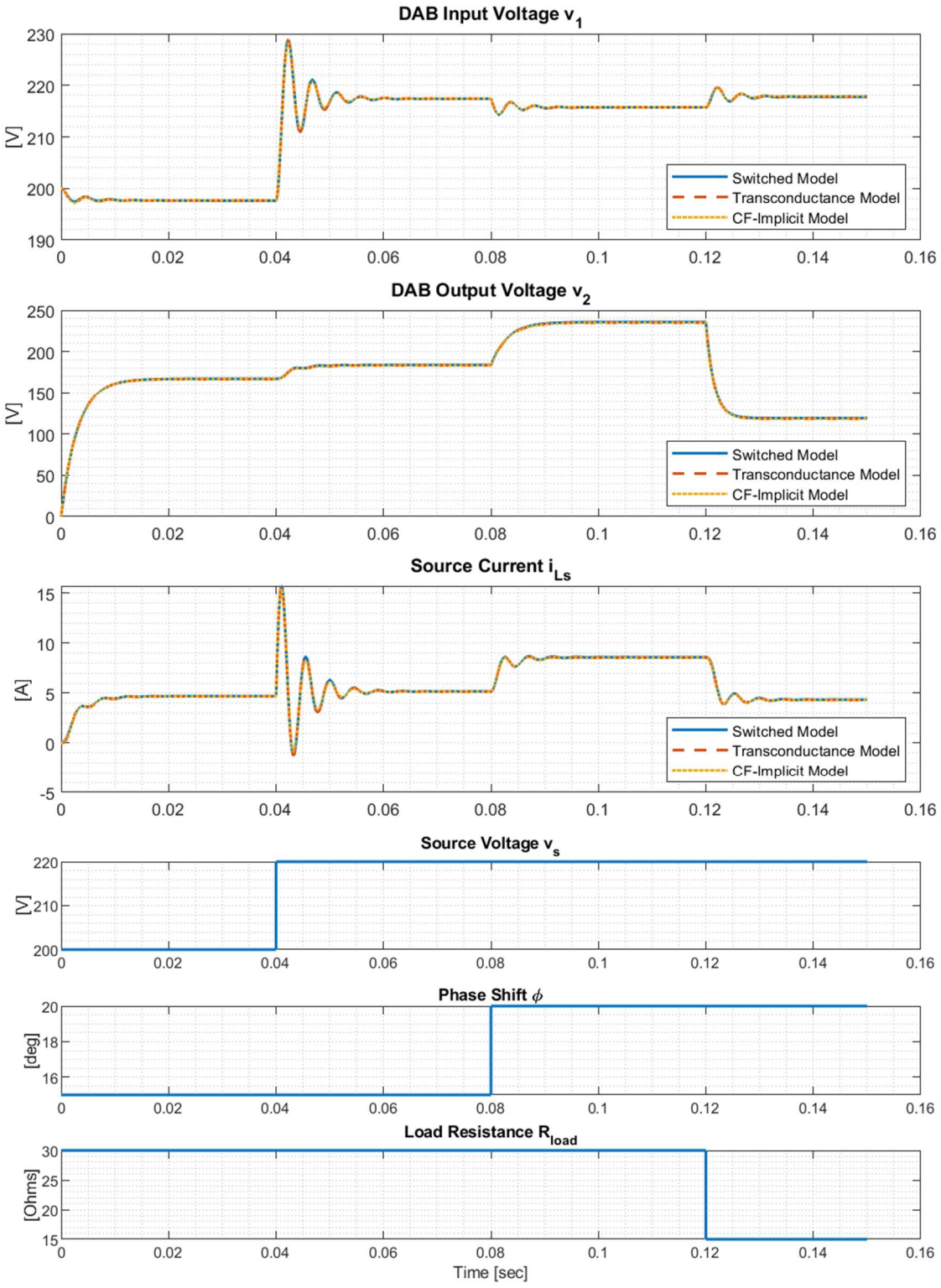

Figure 6.12. Simulation results for equivalent DAB converter models using a traditional switching simulation, the transconductance-based average model, and the explicitly solved model show excellent agreement and no loss of dynamic accuracy.

CHAPTER 7

REDUCED-ORDER CF-IMPLICIT MODELS

One interesting extension of the presented method of Chapter 3 is to define a means for reduced-order models which carry the same benefits as the other CF-implicit models, but which feature a greater degree of abstraction. Reduced-order models aim to sacrifice some accuracy (especially accuracy of fast dynamics and high-frequency content) for greater computational efficiency.

The CF-implicit method presented in Chapter 3 is primarily focused on powertrain hardware modeling, with any simulated feedback controls being included externally to the main DT model, as shown in Figure 4.3. However, it is well-known that feedback control systems are specifically designed to dominate the behavior of the system at steady state and all frequencies up to the bandwidth of the controller. Therefore, when creating reduced-order models which capture only the lower-frequency dynamics of the system, it is natural to include some (or all) of the feedback controls together with the powertrain as a single lumped model rather than as separate entities.

7.1 BOOST CONVERTER EXAMPLE DERIVATION

An example of such a reduced-order CF-implicit approach is presented in this chapter for the boost converter shown in Figure 7.1. Note that, in keeping with the boost converter convention, the duty cycle d is considered as the on-time of the low side switch.

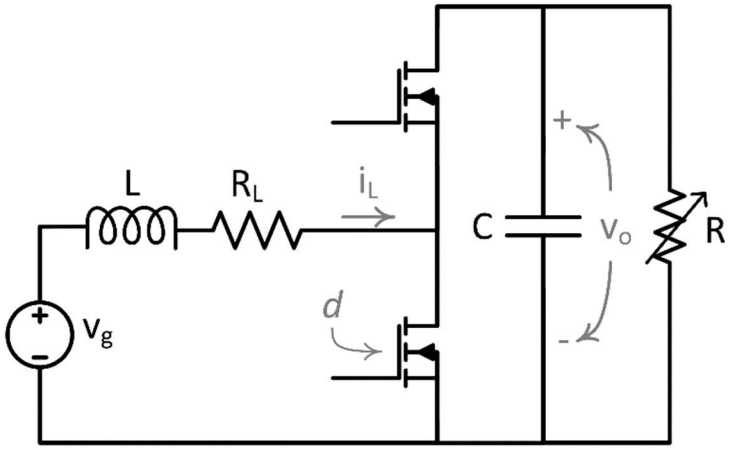

Figure 7.1. Ideal boost converter inductor ESR and a variable resistive load.

A standard, widely used control approach for regulating the output voltage of the converter is the two-loop approach shown in Figure 7.2, in which a fast inner loop regulates inductor current and a slower outer loop regulates output voltage by adjusting the inductor current reference. Since the dynamic behavior of the converter with a well-designed controller depends primarily on the outer control loop, and since the dynamics of the inner loop are necessarily faster than the dynamics of the outer loop, it is common to abstractly model the inner loop as a single lumped feedforward entity when tuning and analyzing the outer loop. This reduced-order approach is shown in Figure 7.3.

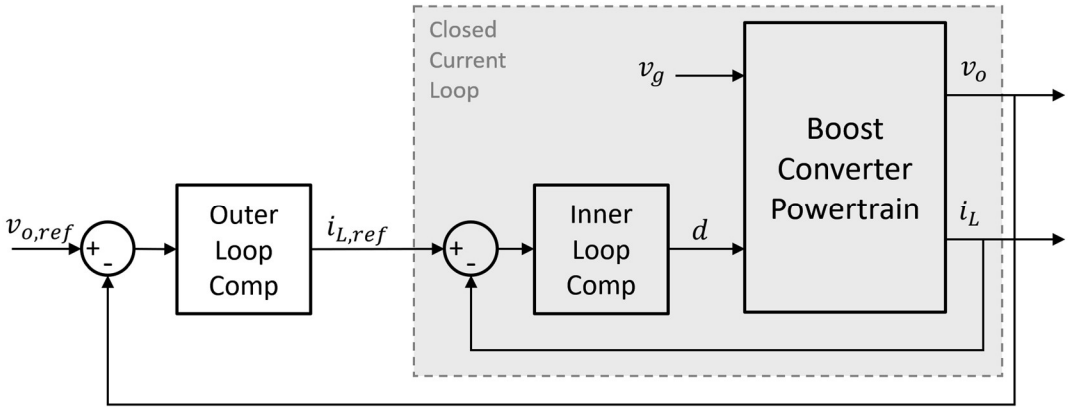

Figure 7.2. Standard two-loop control features a fast inner control loop to regulate inductor current and a slower outer loop to regulate the output voltage.

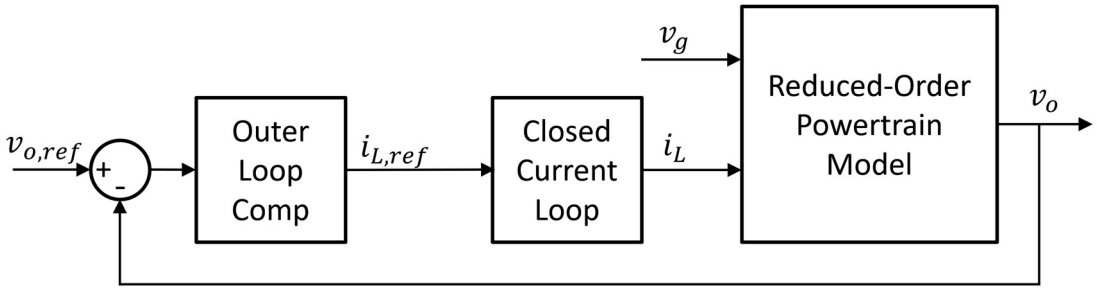

Figure 7.3. A view of the two-loop control system, in which the faster inner current loop has been abstractly modeled as a lumped feedforward entity.

Here the "closed current loop" block approximates the reference tracking properties of the inner feedback loop. The small-signal reference-to-output gain of the current loop having loop gain $T_{CL}(j\omega)$ can be written in the frequency domain as:

$$\frac{\hat{\imath}_L(j\omega)}{\hat{\imath}_{L,ref}(j\omega)} = \frac{T_{CL}(j\omega)}{1 + T_{CL}(j\omega)} \tag{7.1}$$

It is clear from (7.1) that when $T_{CL}(j\omega) \gg 1$, the term $T_{CL}(j\omega)$ dominates the denominator; therefore, when the loop gain is large, the 1 can be neglected and the expression evaluates to approximately unity gain. Contrarily, when $T_{CL}(j\omega) \ll 1$, the term $T_{CL}(j\omega)$ can be neglected in the denominator, and the expression evaluates to approximately the loop gain $T_{CL}(j\omega)$. Therefore, (7.1) behaves essentially as a first order system with a single pole at the crossover frequency of the current loop, designated $\omega_{c,CL}$. That is:

$$\frac{\hat{\imath}_L(j\omega)}{\hat{\imath}_{L,ref}(j\omega)} \approx \frac{1}{1 + \frac{s}{\omega_{c,CL}}} \tag{7.2}$$

Note that the powertrain model required for the reduced-order approach in Figure 7.3 is different than the powertrain model required in the full-order approach. Importantly, the input d is absent from the reduced-order powertrain model because it has been absorbed into the equivalent model of the closed current loop. In its place, the inductor current i_L is

now an input to the powertrain model rather than a state or an output. Hence, we must manipulate the powertrain model to reflect this change in I/O.

As before, we will begin with the switch-averaged state equations of the powertrain, which are:

$$L\frac{di_L}{dt} = v_g - R_L i_L - v_o(1-d) \tag{7.3}$$

$$C\frac{dv_o}{dt} = i_L(1-d) - \frac{v_o}{R} \tag{7.4}$$

Discretizing these equations using Euler Backward (see formula in Table 2.1), we obtain

$$L\frac{i_L[k] - i_L[k-1]}{\Delta t} = v_g[k] - R_L i_L[k] - v_o[k](1-d[k]) \tag{7.5}$$

$$C\frac{v_o[k] - v_o[k-1]}{\Delta t} = i_L[k](1-d[k]) - \frac{v_o[k]}{R} \tag{7.6}$$

Solving (7.5) for the term $(1-d[k])$, we obtain

$$(1-d[k]) = \frac{1}{v_o[k]}\left(v_g[k] - R_L i_L[k] - L\frac{i_L[k] - i_L[k-1]}{\Delta t}\right) \tag{7.7}$$

And then plugging this result into (7.6), we can eliminate d from the equation:

$$C\frac{v_o[k] - v_o[k-1]}{\Delta t} = \frac{i_L[k]}{v_o[k]}\left(v_g[k] - R_L i_L[k] - L\frac{i_L[k] - i_L[k-1]}{\Delta t}\right) - \frac{v_o[k]}{R} \tag{7.8}$$

It is possible to rearrange (7.8) as a solvable standard-form quadratic equation with respect to variable $v_o[k]$. This reformulation yields:

$$\left(\frac{C}{\Delta t} + \frac{1}{R}\right)(v_o[k])^2 - \frac{C}{\Delta t}v_o[k-1]v_o[k] + \left(L\frac{i_L[k] - i_L[k-1]}{\Delta t} + R_L i_L[k] - v_g[k]\right)i_L[k] = 0 \tag{7.9}$$

Therefore, the solution by the quadratic formula is

$$v_o[k] = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$where \quad \left\{ \begin{array}{l} a = \left(\frac{C}{\Delta t} + \frac{1}{R}\right) \\ b = -\frac{C}{\Delta t} v_o[k-1] \\ c = \left(L\frac{i_L[k] - i_L[k-1]}{\Delta t} + R_L i_L[k] - v_g[k]\right) i_L[k] \end{array} \right\} \tag{7.10}$$

The reader can again recognize that this solution represents an implicitly integrated model solved explicitly in closed form; hence, it is certainly a CF-implicit model like those described in previous chapters. As such, it also offers many of the same benefits, including strong numerical stability and high computational efficiency. The traditional benefit of a reduced order model over a full order model is that the simplification has removed some complexity from the model and it may sometimes execute faster than the equivalent full-order model at the partial expense of transient accuracy. A comparison of accuracy between the full-order and reduced-order boost converter model is also included in Section 7.3. An experimental study of computation time for this model in Section 7.4.

7.2 Challenges with Reduced Order Models

However, there are also several key differences and a few complications with the reduced-order approach. One potential difficulty is in automating the solution. The matrix decomposition of the state equations in the method of Chapter 3 has straightforward means of automation via parsing and subsequent "stamping" of terms into the correct coefficient matrix, but the solution process for the reduced-order model proposed here is more complicated, requiring intelligent algebraic manipulation to reach the final solution. It is

not immediately obvious that any formulation exists which could solve the reduced-order modeling equations using a comparable matrix-based approach that is easily automatable.

Another complication of the reduced-order solution stems from the fact that the solution is quadratic, even for a simple system like the boost converter example in this chapter. Quadratic equations typically have non-unique solutions, and may also yield complex or imaginary results if the term inside the square-root is negative. Fortunately, one of the two solutions can always be ignored or thrown out, and in this circuit topology physical constraints will prevent complex/imaginary solutions from occurring in practice.

However, larger converters may have even more complex systems of equations, requiring the closed form analytical solution of equations which are cubic, quartic, or even quintic. Cubic systems are tedious to solve, while solution of quartic equations require the use of the (rather complicated) Ferrari method, and no general solution in radicals exists for quintic systems [60]. It would therefore appear that the reduced order models presented here are heavily limited in scope to simple systems—likely having just one converter stage.

7.3 Reduced Order Model Accuracy

The simplification of the two-loop system of Figure 7.2 to a single-loop system shown Figure 7.3—upon which the reduced order modeling technique is founded—is based on an approximation that the inner current control loop is much higher bandwidth (generally at least ten times higher, using the traditional rule of thumb for well-designed systems) than the outer voltage loop. However, since the current loop is not infinitely fast in any physical systems, the approximation does introduce some error. Specifically, high frequency dynamics of the model can be lost, since accurate accounting of these dynamics would require the model to consider interactions between the two control loops.

For a complete picture of CF-implicit model accuracy, the boost converter system of Figure 7.1 was simulated with a two-loop PI-compensated digital control using three methods: a traditional model with a very small $100\ ns$ timestep, a full order CF-implicit model with a $50\ \mu s$ timestep, and a reduced order CF-implicit model with a $50\ \mu s$ timestep. The parameters of the converter and the control compensators are given in Table 7.1.

Table 7.1. Parameter Values for Boost Converter & Controller

Parameter	**Value**
C	$550\ \mu F$
L	$125\ \mu H$
R_L	$40\ m\Omega$
R	$Variable — see\ figures$
$Voltage\ loop - K_P$	1
$Voltage\ loop - K_I$	500
$Current\ loop - K_P$	0.005
$Current\ loop - K_I$	2
f_{sw}	$20\ kHz$

Figure 7.4 shows simulation results for each of the three simulations. The simulation shows startup from the steady-state condition where the output capacitor is charged to the input voltage—initially 125V. The output voltage reference is initially set to 200V, so the feedback controls push current quickly through the inductor to raise the output voltage. At time $t = 0.1\ sec$, the variable load resistance R drops from $50\ \Omega$ to $20\ \Omega$, requiring the controller to raise the inductor current significantly to compensate. Then, at $t = 0.2\ sec$, the input voltage steps from $125\ V$ to $150\ V$, and the controls again respond to regulate the output voltage. Finally, at $t = 0.3\ sec$, the output voltage reference steps from $200\ V$ to $250\ V$, and the output voltage quickly rises with another adjustment to the inductor current.

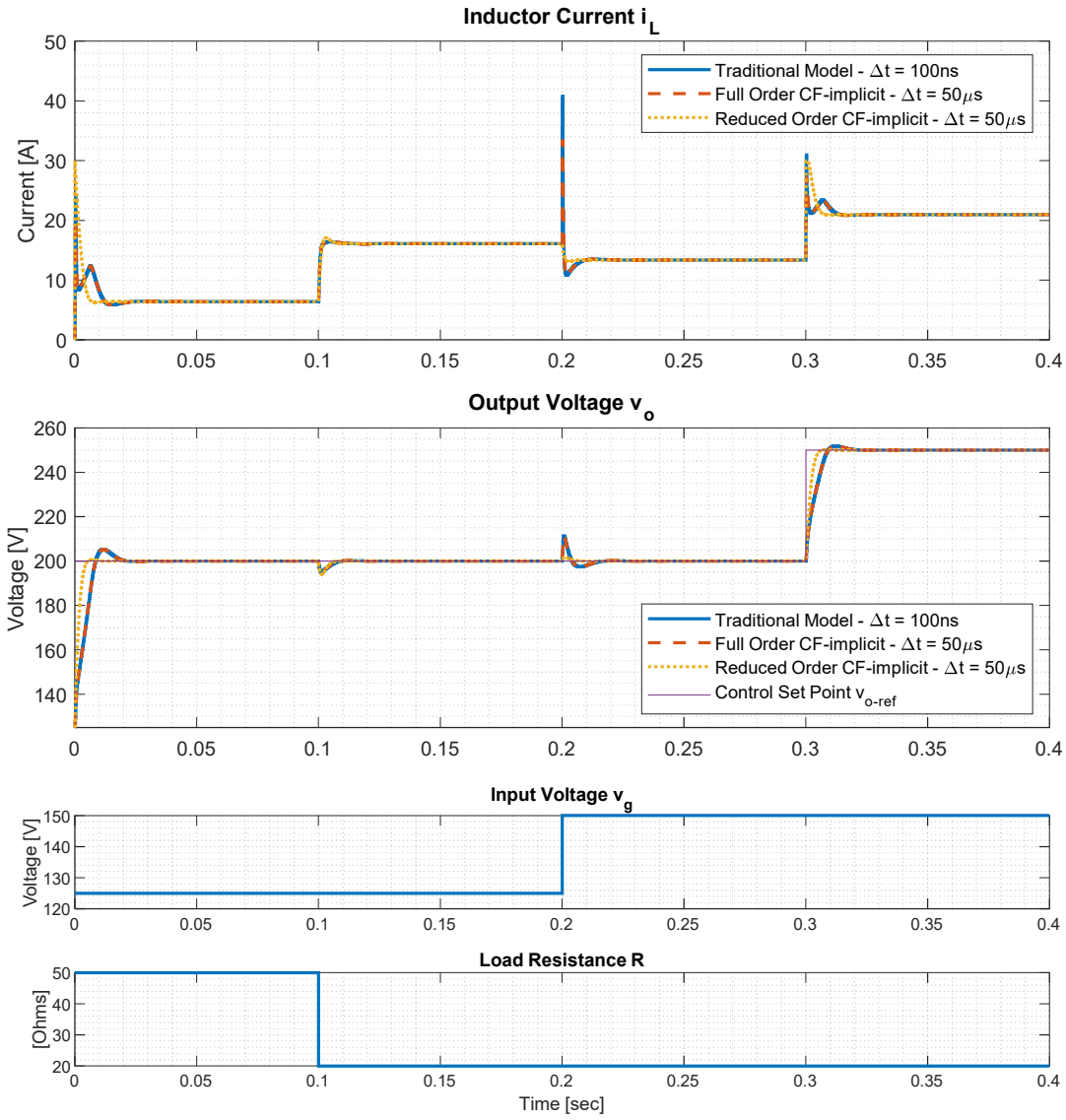

Figure 7.4. Simulation results for a full order and reduced order CF-implicit models of the boost converter with two loop control, with a traditional model featuring a small timestep for comparison.

Clearly, all three models—the traditional model, the full order CF-implicit, and the reduced order CF-implicit—agree perfectly at steady state. However, close inspection of the fast transients at $t = 0, 0.1, 0.2,$ and $0.3\ sec$ reveals that the reduced order model does suffer from some inaccuracy from missing dynamics in the current loop. Figure 7.5 shows a more focused view on of the transient at $t = 0.2\ sec$.

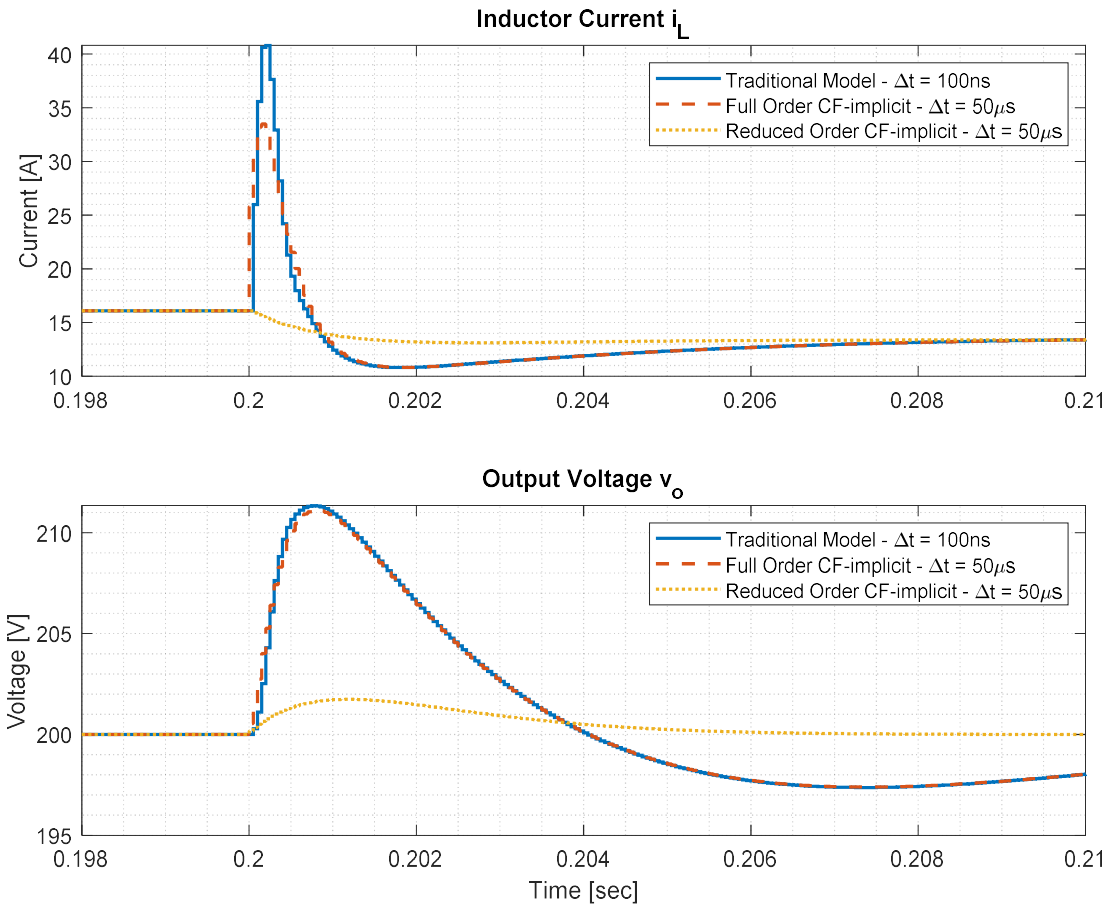

Figure 7.5. A zoomed-in view of the model outputs from Figure 7.4, highlighting error caused by absence of the boost converter's right-half-plane zero in the reduced order model.

The close-up view reveals a nontrivial discrepancy between the traditional model with the small timestep (which in this case is the "correct" waveform) and the full order CF-implicit model. The initial current spike in the traditional model waveform reaches past 40 A, while the full order CF-implicit model reaches only about 33 A. This can be attributed to the large timestep of the CF-implicit model; regardless of integration method, accuracy can always suffer when the dynamics of the transient happen in the scale of just two or three timesteps.

However, the much more significant transient discrepancy here is from the reduced-order model, which eliminates the initial 25 ampere current spike completely and moves

in the opposite direction toward the steady state value at the beginning of the transient. This is unsurprising, since Figure 7.3 shows that the inductor current is simply a low-pass-filtered version of the control reference in the reduced order model; it is, in fact, used as an input to the reduced order converter model rather than computed as an output. Of course, with such a large error in the inductor current, the output voltage transient also tends to be terribly inaccurate; in this case, it reduces the $11V$ overshoot to just about $2V$ before eventually rejoining the more accurate models at the end of the transient.

7.4 Computational Efficiency of the Reduced Order Model

The loss of transient accuracy detailed in the previous section may or may not be acceptable, depending on the application in which the model is deployed. However, since the full-order CF-implicit model is both more accurate and more easily created (via an automatable process), the reduced order must offer greater computational efficiency or faster execution to justify its own use. We recorded computational resource usage on the control platform's ARM Cortex A9 1GHz processor for the example boost converter simulation in this chapter. Data was record for both the full-order and reduced-order CF-implicit models with identical timesteps ranging from $1\ \mu s$ to $50\ \mu s$. The results of the study are shown in Figure 7.6. In the figure, 'x' marks denote observed average CPU execution times per interrupt, while 'o' marks represent recorded minimum and maximum values to show execution time jitter. Linear regression lines from the average execution time data are also shown.

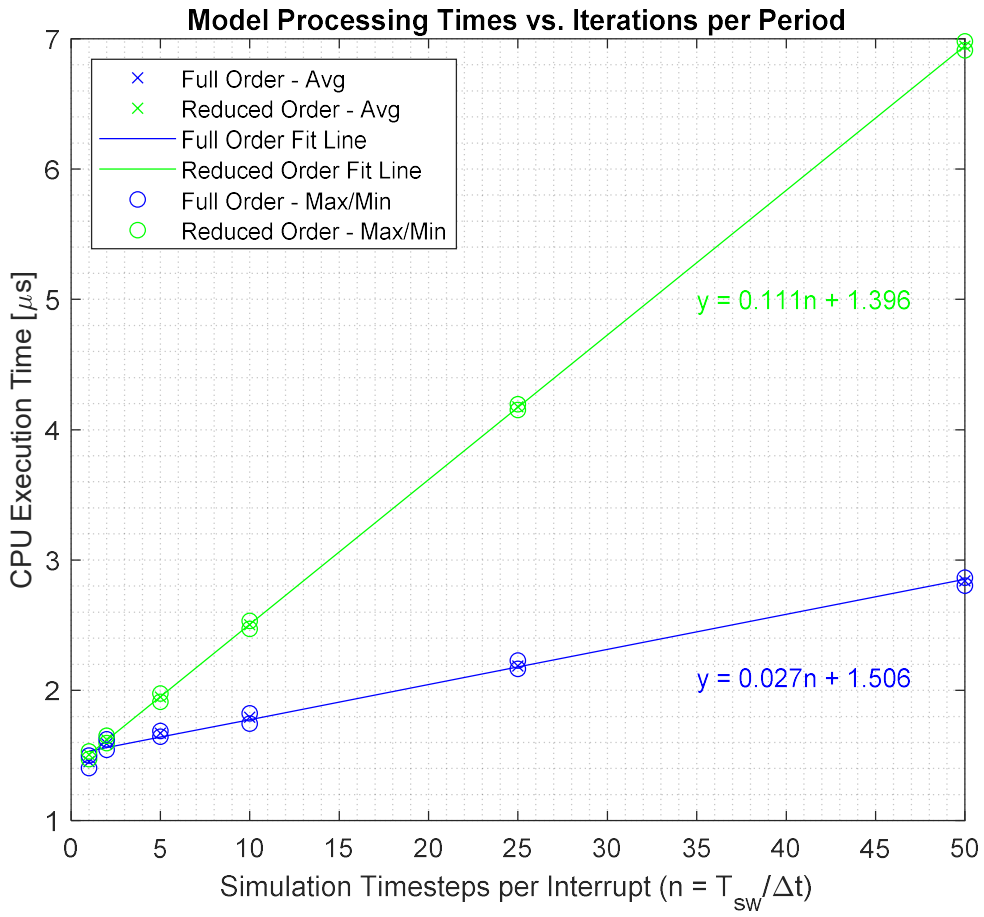

Figure 7.6. A scatter plot showing the processing times for equivalent full- and reduced-order methods as observed on the control platform's 1GHz ARM CPU.

The clear conclusion of the data is that the reduced-order model did not offer any benefit to computational efficiency. On the contrary, on a per-timestep basis it was more than four times more computationally expensive than the full-order model—even despite having removed the inner current control loop and all computational burden associated with it. The increased computational expense can likely be attributed to the square root operation required in the reduced-order model. Square root calculations on floating-point numbers are notoriously costly operations—especially if there is not special dedicated hardware on board for the calculation [61]. Traditionally, square root operations on

floating point numbers require Newton's method to be iteratively applied until a suitably accurate approximation has been reached.

7.5 Conclusions on Reduced-Order CF-Implicit Models

It is possible that the particular data shown in Figure 7.6 may be improved by manipulating compiler-level implementation details. Even so, the improvement would have to be quite dramatic just to get the reduced-order model to compete with the full-order model implementation. Therefore, it is unlikely that the reduced-order model would be implementable in a way that improves upon the equivalent full-order model.

The limited study here of the boost converter example here may not be exactly representative of all possible reduced-order CF-implicit modeling cases. Each implemented reduced-order CF-implicit model will have its own unique set of model equations with its own computational performance. Nevertheless, the example studied in this chapter was sufficiently representative of the general trend to conclude that reduced order models may not be appropriate for most use cases—or perhaps for any use cases. Until a reduced order model can demonstrate significantly reduced computational complexity in the execution of the model, there exists no compelling reason to sacrifice transient accuracy and to go to the trouble of manually deriving it.

CHAPTER 8

COMPARING CF-IMPLICIT MODELS WITH DATA-DRIVEN ALTERNATIVES

The CF-implicit models proposed herein provide a combination of characteristics (as discussed in Chapter 2) which are unique among known analytical modeling approaches. However, recently proposed data-driven modeling approaches using dynamic (i.e., recurrent) neural networks have demonstrated a similar set of features which would make them appropriate for digital twin applications [62]. Like the proposed CF-implicit models, these data-driven ANN models are time-domain, switch-averaged, large-signal, real-time, and embeddable. Of course, since they are based in radically different approaches to model creation, there are also significant differences between the analytical and data-driven modeling approaches which lead to distinct advantages and disadvantages for each method.

8.1 OVERVIEW OF ANALYTICAL AND DATA-DRIVEN MODEL SPECTRUM

One of the unique benefits of the digital twin framework is the availability of both preprogrammed intelligence and the large amount of measurement data which the physical system produces in real time. Taken together, these two can be used to improve performance and decision-making capability of the digital twin. The digital twin may feature some models which were developed by experts prior to system deployment and other models which are developed only from empirical data produced by the physical system. In general, the models that can be used in a digital twin exist on a continuous spectrum from predefined mathematical models to data-driven empirical models, with

hybrid approaches falling somewhere in between. The model spectrum is depicted in Figure 8.1. This section details model types across the spectrum to provide context for the comparison between the models in the remainder of the chapter.

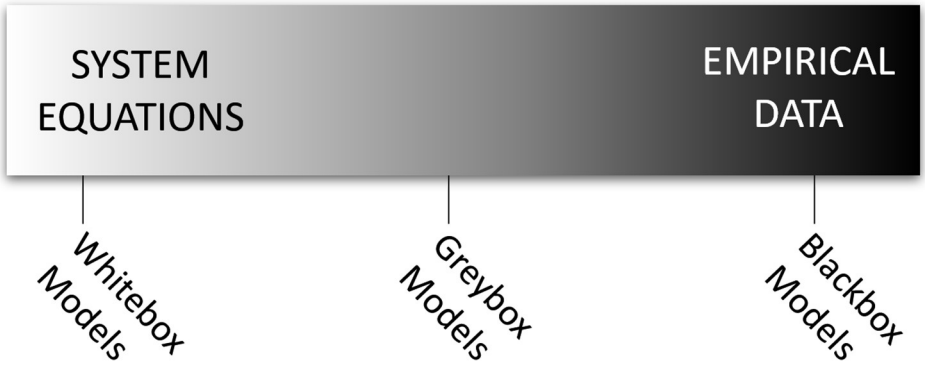

Figure 8.1. Models exist on a spectrum, from rigid predefined whitebox models to data-driven blackbox models.

8.1.1 WHITEBOX MODELS

The CF-implicit models which are the primary focus of this dissertation are pure whitebox models. Whitebox models are predefined analytical models with zero reliance on measured system data. Whitebox models can be behavioral and/or physics based. All mathematical equations and parameters used in the model are known and specified directly by the model developer when the model is created. Most offline simulations conducted during a typical engineering design process would be whitebox models.

8.1.2 BLACKBOX MODELS

Most traditional machine-learning models—including the NARX-ANN models discussed in this chapter—are blackbox models. Occupying the opposite extreme of the spectrum, blackbox models are fully data-driven models which are created by choosing a highly configurable general purpose modeling structure and tuning the model parameters using empirical data. In a blackbox model, no system equations or parameters are dictated

by the model developer—the model's behavior is determined by the empirical data used to create it.

8.1.3 Greybox Models

Falling in between whitebox and blackbox methods, greybox models combine predefined analytical models with some data-driven empirical methods to combine benefits of both. Greybox models exist on a spectrum as depicted in Figure 8.1, and as such they are not as easy to categorically define as the whitebox and blackbox models at the extremes. Generally, however, greybox models include system equations which are directly specified by the model designer in terms of certain system parameters. These parameters may be given nominal values but can be actively reidentified later using empirical data captured online from the physical system. Thus, greybox models rely on *some* measurement data, but usually require less data than would be needed for training a blackbox model.

A detailed discussion of greybox models is outside of the scope of this work. However, since any whitebox model can theoretically be combined with a parameter identification strategy to create a greybox model with online retuning capability [63], the CF-implicit models proposed here could certainly be used in a greybox modeling scheme. This approach is especially useful in digital twin applications where the physical twin's state-of-health may degrade over time—requiring updated model parameters.

8.2 Qualitative Feature Comparison of Blackbox and Whitebox Models

Models across the spectrum of Figure 8.1 can be dramatically different, and the categories of whitebox, blackbox, and greybox are broad. Nevertheless, even using broad categories, many features, advantages, and disadvantages can be surmised from a model's position on the spectrum.

8.2.1 Reproducibility

Traditional whitebox models are extremely predictable, reliable, and reproducible—there is no uncertainty concerning the model's behavior because the model developer explicitly programs the equations and parameters which dictate the model's behavior. However, if the physical system undergoes significant changes throughout its lifetime, it may diverge from the whitebox models because the models are not updated online to reflect the changes.

Blackbox models are significantly more difficult to reliably reproduce. They are typically generated using large datasets—and therefore cannot be perfectly recreated without using the exact same dataset for training. The gradient decent algorithms which adjust blackbox model parameters (e.g., weights in an ANN) during training are also prone to problems with reproducibility because model parameters are often initialized randomly, datasets are frequently shuffled at initialization, and because continuously evolving ML frameworks and hardware configurations have an effect on training success [64] [65].

8.2.2 Risk of Unexpected Model Behavior

Blackbox models carry a much higher degree of uncertainty than whitebox models because the model developer does not explicitly program the behavior of the model. Purely empirical training can lead to bizarre and unexpected behavior in some circumstances, especially if the model training data is poorly chosen. Another significant and well-documented danger is that the model will "overfit" the training dataset by learning coincidental features or noise, which can cause apparently well-trained models to perform poorly when presented with new data. Artificial neural networks are notoriously prone to overfitting.

Since whitebox models are purely deterministic, predefined, and predictable, they pose very little risk of unexpected behavior. The only significant concern about predictability of whitebox models are convergence and numerical stability. Both concerns are addressed and dramatically improved by the CF-implicit method proposed in this work, leading to an even stronger advantage for whitebox models.

8.2.3 Online Model Retuning

In theory, blackbox models offer the benefit of "online" retuning, allowing the user to update their model while the physical system is running and thereby account for any changes that the physical system underwent since the initial deployment. However, blackbox model training generally requires a very large amount of data and can be very computationally expensive.

Pure whitebox models do not offer any online model retuning capabilities. However, as previously discussed, any parameterized whitebox model can be turned into a greybox model by combining the whitebox model with a parameter identification algorithm. This combination arguably offers the best model retuning solution, retaining many benefits of the whitebox model, and allowing for more reliable model retuning with a much smaller dataset than required for retraining a blackbox model.

8.2.4 Suitable for Modeling Unknown I/O Relationships

The chief benefit of blackbox models—and the primary reason they have grown in popularity in recent years—is that they can effectively and accurately model systems whose I/O relationships are unknown or extremely difficult to specify in a whitebox model. Blackbox models are the obvious solution for problems which are too complicated to define explicitly using mathematics. For example, blackbox machine learning models—especially

artificial neural networks—have shown remarkable and unprecedented success in areas such as speech recognition or image classification. Blackbox models have also found some use in modeling power and energy systems [66], [67] but are less common than whitebox models, in part because they do not make use of the widely available knowledge of the mathematics describing these systems.

8.2.5 INTERPRETABILITY

Whitebox models lend themselves to easy interpretation of the results they produce because their input/output relationships are directly specified by the creator of the model. Any output produced by a whitebox model can be traced back through the model to the inputs, leaving no output without a clear mathematical explanation.

Blackbox models usually do not allow for easy interpretation of their results, since the highly flexible computational structures used to create them obtain their generality at the cost of very high complexity. It is usually quite difficult, for example, to trace back the output of a deep neural network through its layers to the inputs and gain meaningful insights about what relationship the neural network has "learned."

Recently the research community has put significant efforts toward the development of methods for explainable artificial intelligence (XAI, also known as interpretable AI) which aim to use the data-driven machine learning approach to produce *whitebox* models whose underlying equations can be analyzed directly and understood by domain experts [68]. XAI shows great promise in increasing the interpretability of data-driven modeling, but currently cannot fully combine the level of accuracy offered by blackbox models with the interpretability of whitebox models; in fact, the best performing ML methods are often the least explainable [69].

8.3 Brief Review of NARX-ANN Converter Modeling Work

The use of dynamic NARX neural networks to model converters was introduced and analyzed in detail in [62]. In that work, the boost converter topology of Figure 8.2 was selected for analysis; note that this topology is nearly identical to the topology studied in Chapter 7 except that the load in this topology is represented by a controlled current source rather than a variable resistor. The model structure and I/O is shown as a signal flow diagram in Figure 8.3.

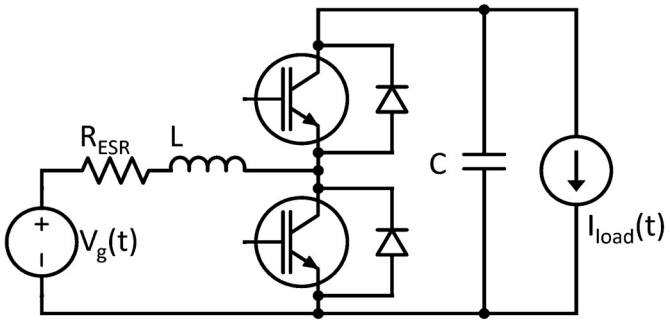

Figure 8.2. The boost converter topology studied in the NARX-ANN paper.

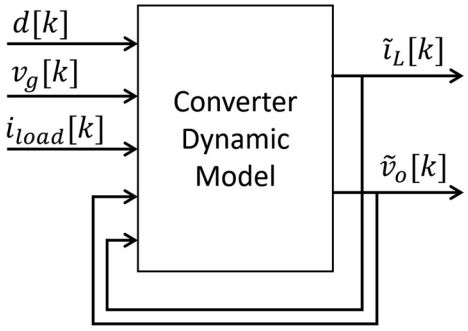

Figure 8.3. The signal-flow equivalent model of the boost converter in Figure 8.2.

Details of model creation—including the selection of the model I/O, the creation of a training dataset, and selection of model hyperparameters are discussed in detail in the work. Empirical analysis revealed that the NARX-ANN models produced accurate results only when the chosen hyperparameters specified a deep neural network with at least three

hidden layers, at least ten neurons per layer, and exactly one delay in the input and feedback tapped delay lines. This NARX-ANN structure is shown in Figure 8.4. The work found that somewhat higher model fidelity could be produced with a larger network—for example, a NARX-ANN with 4 hidden layers and 20 neurons in each layer—at the cost of a slower, more computationally expensive training process. Extremely large network sizes beyond this point were empirically shown to produce diminishing returns on accuracy, while much smaller network sizes offered intolerably poor model accuracy.

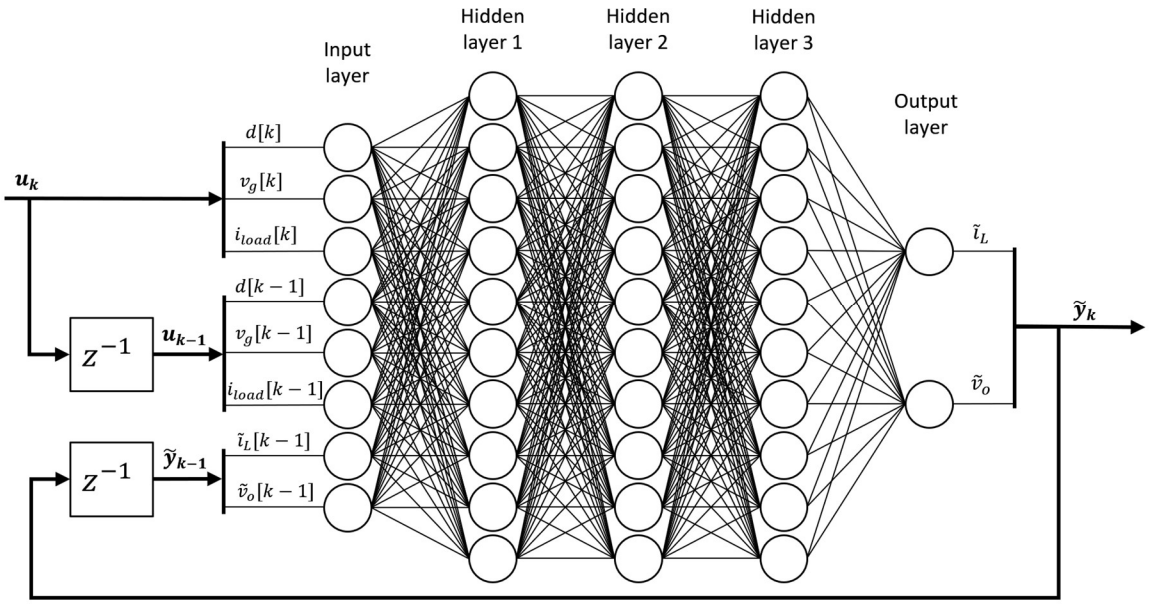

Figure 8.4. The internal structure of the medium-size NARX-ANN used for modeling. This structure provided a good tradeoff between network size and model accuracy.

A novel dual time-domain and frequency-domain validation method is also included in [62]; this dual validation technique was proposed to provide high confidence in the performance of the trained model. The NARX-ANN models are trained in the time-domain, and traditionally they would also be validated exclusively in the time-domain. However, it was shown that the NARX-ANN model can be linearized around a DC operating point and then characterized by obtaining a small-signal frequency response

measurement which can be compared with expected results obtained from the analytically derived converter transfer functions. This dual validation method can provide significantly higher confidence that the model will perform as intended since the model training did not incorporate any frequency domain component. As shown in the paper, the trained dynamic neural network model provided excellent accuracy in the frequency domain testing, matching the expected frequency response very closely at multiple distinct operating points.

Overall, the NARX-ANN converter models appear to be a promising direction for future research. One of the proposed extensions of the NARX-ANN modeling work was to deploy the dynamic neural networks to an embedded environment as a digital twin and evaluate the speed of the model for RT and faster-than-RT simulations. That task is undertaken here to produce a comparison of computational cost between the CF-implicit models and the NARX-ANN models.

8.4 Quantitative Comparison of Computational Cost

For consistency with previous work, the comparison of computational cost uses the same topology for study shown in Figure 8.2. Three distinct models of the converter were created: a full-order CF-implicit model, a medium-size NARX-ANN model having three hidden layers with ten neurons each (exactly as shown in Figure 8.4), and a larger-size NARX-ANN model having four hidden layers with twenty neurons each. Each model was executed in real time with a single $50\ \mu s$ timestep occurring on a regular $50\ \mu s$ interrupt, running on the Imperix B-box 3.0 on a single ARM Cortex A9 CPU with 1GHz clock and 1GB of DDR3 memory. The model execution time per interrupt was measured using the Imperix ACG BB Control Utility tool, which communicates with the B-Box using a

separate on-board CPU to preclude disturbance of the measured CPU utilization. The results of the comparison are shown in Table 8.1.

Table 8.1. Comparison of Measured Model Execution Times on ARM Cortex A9 Processor

Model Type	Percentage of ARM Cortex A9 CPU Utilization	Average Execution Time per $50\ \mu s$ Interrupt	Maximum Execution Time per $50\ \mu s$ Interrupt
Full-Order CF-Implicit Model	4.4%	$1.341\ \mu s$	$1.364\ \mu s$
Medium Size NARX-ANN (3 hidden layers, 10 neurons each)	14.37%	$6.281\ \mu s$	$6.312\ \mu s$
Large Size NARX-ANN (4 hidden layers, 20 neurons each)	33.5%	$15.856\ \mu s$	$15.944\ \mu s$

The results of the comparison are extremely decisive—the CF-implicit models use a small fraction of the processing time required for inference using the NARX-ANN models of both sizes when executed on a CPU processor. Average execution time for the CF-implicit model is reduced by more than 78% from the medium size NARX-ANN and 91% from the large size NARX-ANN. This is perhaps unsurprising, as the ANN models require hundreds (or even thousands) of multiplication and addition operations for each neural inference, and the CPU does not allow for any significant parallelization of the operations. Thus, the CF-implicit model wins the competition purely on account of requiring fewer operations.

Model execution time for the data-driven models is likely to be dramatically reduced on a parallelized computing platform due to the highly parallel structure of neural networks. We expect that the analytical models proposed here may also benefit with some reduction in computational time on a parallel computing architecture, but that the benefit of parallel computing to CF-implicit models would be much less significant than that of

the data-driven NARX-ANN models. It is possible—but unproven—that the data-driven NARX-ANN models could be competitive or even faster on a parallel computing platform. This information would help users make an informed choice when selecting computing hardware for a digital twin application.

Options for a parallel computing platform include graphics processing units (GPUs) and field programmable gate arrays (FPGAs). GPUs have become extremely popular equipment for training and inference in deep neural network applications for their parallelized structure which offers high computational throughput; however, such GPUs consume a large amount of power compared with alternatives and are not always a suitable choice, especially in embedded environments like a digital controller [70]. In these spaces, FPGAs offer a compelling alternative for their excellent flexibility, parallel computing capabilities, and relatively low power consumption. Nevertheless, the tradeoff for the excellent flexibility of FPGAs is that a large amount of detailed work is required for implementation.

Since the Imperix B-box has a user-programmable Xilinx Kintex 7 125K FPGA built in, we did make some quick attempts to execute each of the models on the FPGA by using a high-level synthesis (HLS) tool to convert the models to synthesizable HDL code. The CF-implicit model could be synthesized and successfully executed on the FPGA, but no comparison could be completed because the NARX-ANN models in this work could not be directly programmed on the available FPGA in a fully parallelized manner. Attempts to synthesize even the "medium size" NARX-ANN led to overutilization of every resource on the FPGA, in some cases by a factor of ten or more. Clearly, then, some amount of serialization of the ANN's computations is necessary to execute an ANN of any

reasonable size on an FPGA. One approach for ANN execution is detailed in [71], in which the computations for each neuron are serialized using a multiply-accumulate architecture to sequentially update the output with weighted. Future work may include an implementation of this approach (or some similar approach) to fit the NARX-ANN model on the FPGA and produce a fuller comparison of CF-implicit and NARX-ANN models on an FPGA platform.

8.5 Conclusions of the Comparison

Overall, the proposed CF-implicit models compare very favorably to data-driven alternatives like the NARX-ANN models for a variety of reasons. The CF-implicit models offer important advantages in reproducibility and interpretability, and they avoid the data-driven model's critical risk of unexpected behavior. They are also highly accurate, in part because they directly execute the intended model equations rather than approximating them through a complex network of weights and activations.

If a CPU processor is the intended platform for execution, the CF-implicit models also offer a very strong computational benefit, executing much faster and requiring less resources than equivalent NARX-ANN models. This computational performance advantage may or may not be maintained on parallel computing platforms including GPUs and FPGAs.

REFERENCES

[1] X. Guillaud et al., "Applications of Real-Time Simulation Technologies in Power and Energy Systems," *IEEE Power and Energy Technology Systems Journal,* vol. 2, no. 3, pp. 103-115, 2015.

[2] J. Meng, Y. Want, C. Wang and H. Wang, "Design and Implementation of Hardware-in-the-Loop Simulation System for Testing Control and Operation of DC Microgrid with Multiple DG Units.," *IET Generation, Transmission & Distribution. ,* vol. 11, no. 22, pp. 3065-3072, 2017.

[3] B. Moon, C. A. De La O, H. Ginn and A. Benigni, "Decentralized Model Predictive Control of a Power Electronic Power Distribution System," in *2019 International Conference on Clean Electrical Power (ICCEP)*, Otranto, Italy, 2019.

[4] C. A. De La O, M. Difronzo, A. Benigni and H. L. Ginn, "A Hardware-in-the-Loop Platform for Testing Networked Controllers for Microgrids," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, Washington, DC, 2018.

[5] C. Liu, R. Ma, H. Bai, Z. Li, F. Gechter and F. Gao, "FPGA-Based Real-Time Simulation of High-Power Electronic System With Nonlinear IGBT Characteristics," *IEEE Journal of Emerging and Selected Topics in Power Electronics,* vol. 7, no. 1, pp. 41-51, 2019.

[6] R. Mirzahosseini and R. Iravani, "Small Time-Step FPGA-Based Real-Time Simulation of Power Systems Including Multiple Converters," *IEEE Transactions on Power Delivery,* vol. 34, no. 6, pp. 2089-2099, 2019.

[7] M. D. O. Faruque et al., "Real-Time Simulation Technologies for Power Systems Design, Testing, and Analysis," *IEEE Power and Energy Technology Systems Journal,* vol. 2, no. 2, pp. 63-73, 2015.

[8] M. Lemaire, D. Massicotte and J. Bélanger, "Multi-FPGA Communication Interface for Electric Circuit Co-Simulation," in *2020 IEEE Electric Power and Energy Conference (EPEC)*, Edmonton, CA, 2020.

[9] B. Lu, X. Wu, H. Figueroa and A. Monti, "A Low-Cost Real-Time Hardware-in-the-Loop Testing Approach of Power Electronics Controls," *IEEE Transactions on Industrial Electronics,* vol. 54, no. 2, pp. 919-931, 2007.

[10] C. Dufour, S. Cense and J. Belanger, "An FPGA HIL Reconfigurable Testing Platform for Vehicular Traction Systems," in *2014 IEEE Vehicle Power and Propulsion Conference (VPPC)*, Coimbra, Portugal, 2014.

[11] M. Difronzo, M. Milton, M. Davidson and A. Benigni, "Hardware-in-the-Loop Testing of High Switching Frequency Power Electronics Converters," in *2017 Electric Ship Technologies Symposium (ESTS)*, Arlington, VA, 2017.

[12] T. Liang, Q. Liu and V. R. Dinavahi, "Real-Time Hardware-in-the-Loop Emulation of High-Speed Rail Power System With SiC-Based Energy," *IEEE Access,* vol. 8, pp. 122348 - 122359, 2020.

[13] X. Meng, J. Han, L. M. Bieber, L. Wang, W. Li and J. Belanger, "A Universal Blocking-Module-Based Average Value Model of Modular Multilevel Converters With Different Types of Submodules," *IEEE Transactions on Energy Conversion,* vol. 35, no. 1, pp. 53-66, 2020.

[14] M. Milton, M. Difronzo, D. Chowdhury, H. Ginn and A. Benigni, "A Model of MMCs for Power Electronic System High-Performance Real-Time Simulation," in *2022 Open Source Modelling and Simulation of Energy Systems (OSMSES)*, Aachen, Germany, 2022.

[15] W. Li and J. Bélanger, "An Equivalent Circuit Method for Modelling and Simulation of Modular Multilevel Converters in Real-Time HIL Test Bench," *IEEE Transactions on Power Delivery,* vol. 31, no. 5, pp. 2401-2409, 2016.

[16] M. Difronzo, M. M. Biswas, M. Milton, H. L. Ginn and A. Benigni, "System Level Real-Time Simulation and Hardware-in-the-Loop Testing of MMCs," *Energies,* vol. 14, no. 3046, 2021.

[17] F. Tao, H. Zhang, A. Liu and A. Nee, "Digital Twin in Industry: State-of-the-Art," *IEEE Transactions on Industrial Informatics,* vol. 15, no. 4, pp. 2405-2415, 2019.

[18] A. Rasheed, O. San and T. Kvamsdal, "Digital Twin: Values, Challenges and Enablers From a Modeling Perspective," *IEEE Access,* vol. 8, pp. 21980-22012, 2020.

[19] W. Bellamy III, "Boeing CEO Talks 'Digital Twin' Era of Aviation," 14 September 2018. [Online]. Available: https://www.aviationtoday.com/2018/09/14/boeing-ceo-talks-digital-twin-era-aviation/. [Accessed 22 July 2020].

[20] D. Pomerantz, "The French Connection: Digital Twins From Paris Will Protect Wind Turbines Against Battering North Atlantic Gales," 26 April 2018. [Online]. Available: https://www.ge.com/news/reports/french-connection-digital-twins-paris-will-protect-wind-turbines-battering-north-atlantic-gales. [Accessed 22 July 2020].

[21] W. Danilczyk, Y. Sun and H. He, "ANGEL: An Intelligent Digital Twin Framework for Microgrid Security," in *North American Power Symposium (NAPS)*, Wichita, KS, USA, 2019.

[22] J. Coors-Blankenship, "Taking Digital Twins for a Test Drive with Tesla, Apple," 29 April 2020. [Online]. Available: https://www.industryweek.com/technology-and-iiot/article/21130033/how-digital-twins-are-raising-the-stakes-on-product-development. [Accessed 22 July 2020].

[23] D. Drazen, M. Vandroff and L. Tarasek, "Cyber-Physical Systems: Navy Digital Twin," Carderock Naval Surface Warfare Center, Carderock, MD, 2018.

[24] S. Kanowitz, "How digital twins keep Navy ahead on ship maintenance," 6 May 2020. [Online]. Available: https://defensesystems.com/articles/2020/05/06/navy-digital-twin-ship-maintenance.aspx. [Accessed 22 July 2020].

[25] J. F. Epperson, An Introduction to Numerical Methods and Analysis, 2nd ed., Hoboken, NJ: John Wiley & Sons, Inc., 2013.

[26] M. O. Faruque and V. Dinavahi, "Hardware-in-the-Loop Simulation of Power Electronic Systems Using Adaptive Discretization," *IEEE Transactions on Industrial Electronics,* vol. 57, no. 4, pp. 1146-1158, 2010.

[27] A. Benigni and A. Monti, "A Parallel Approach to Real-Time Simulation of Power Electronics Systems," *IEEE Transactions on Power Electronics,* vol. 30, no. 9, pp. 5192-5206, 2015.

[28] M. Milton, A. Benigni and J. Bakos, "System-Level, FPGA-Based, Real-Time Simulation of Ship Power Systems," *IEEE Transactions on Energy Conversion,* vol. 32, no. 2, pp. 737-747, 2017.

[29] M. Milton, A. Benigni and A. Monti, "Real-Time Multi-FPGA Simulation of Energy Conversion Systems," *IEEE Transactions on Energy Conversion,* vol. 34, no. 4, pp. 2198-2208, 2019.

[30] S. Cuk, Modelling, analysis, and design of switching converters, Ph.D. Dissertation: California Institute of Technology, 1977.

[31] R. W. Erickson and D. Maksimovic, Fundamentals of Power Electronics, 3rd ed., New York: Springer, 2020.

[32] C. C. Marouchos, The Switching Function: Analysis of Power Electronic Circuits, London: The Institution of Engineering and Technology, 2006.

[33] C. Gonzalez-Castano, C. Restrepo, R. Giral, E. Vidal-Idiarte and J. Calvente, "ADC Quantization Effects in Two-Loop Digital Current Controlled DC-DC Power Converters: Analysis and Design Guidelines," *Applied Sciences,* vol. 10, no. 20, p. 7179, 2020.

[34] J. H. Allmeling and W. P. Hammer, "PLECS-piece-wise linear electrical circuit simulation for Simulink," in *Proceedings of the IEEE 1999 International Conference on Power Electronics and Drive Systems. PEDS'99*, Hong Kong, China, 1999.

[35] K. Lian and P. Lehn, "Real-time simulation of voltage source converters based on time average method," *IEEE Transactions on Power Systems,* vol. 20, no. 1, pp. 110-118, 2005.

[36] H. Ravindra, M. Stanovich and M. Steurer, "Documentation for a Notional Two Zone Medium Voltage DC Shipboard Power System Model implemented on the RTDS," ESRDC, Tallahassee, FL, 2016.

[37] K. Sun, D. Soto, M. Steurer and M. O. Faruque, "Experimental verification of limiting fault currents in MVDC systems by using modular multilevel converters," in *2015 IEEE Electric Ship Technologies Symposium (ESTS)*, Alexandria, VA, 2015.

[38] A. L. J. H. Rainer Marquardt, "Stromrichterschaltungen mit verteilten Energiespeichern". Germany Patent DE10103031A1, 24 01 2001.

[39] A. Lesnicar and R. Marquardt, "An innovative modular multilevel converter topology suitable for a wide power range," in *2003 IEEE Bologna Power Tech Conference Proceedings*, Bologna, Italy, 2003.

[40] M. Glinka and R. Marquardt, "A new AC/AC multilevel converter family," *IEEE Transactions on Industrial Electronics,* vol. 52, no. 3, pp. 662-669, 2005.

[41] R. Marquardt, "Modular Multilevel Converters: State of the Art and Future Progress," *IEEE Power Electronics Magazine,* vol. 5, no. 4, pp. 24-31, Dec 2018.

[42] M. A. Perez, S. Ceballos, G. Konstantinou, J. Pou and R. P. Aguilera, "Modular Multilevel Converters: Recent Achievements and Challenges," *IEEE Open Journal of the Industrial Electronics Society,* vol. 2, pp. 224-239, 2021.

[43] M. R. Haddioui, "Control and modulation strategies for MMC-based HVDC," Department of Energy Technology, Aalborg University, Aalborg, 2015.

[44] C. Liu, F. Deng, Q. Yu, Y. Wang, F. Blaabjerg and X. Cai, "Submodule Capacitance Monitoring Strategy for Phase-Shifted Carrier Pulsewidth-Modulation-Based Modular Multilevel Converters," *IEEE Transactions on Industrial Electronics,* vol. 68, no. 9, pp. 8753-8767, 2021.

[45] S. Yang, H. Chen, P. Sun, H. Wang, F. Blaabjerg and P. Wang, "Resilient Operation of an MMC with Communication Interruption in a Distributed Control Architecture," *IEEE Transactions on Power Electronics,* vol. 36, no. 10, pp. 12057-12069, 2021.

[46] M. A. Perez, S. Bernet, J. Rodriguez, S. Kouro and R. Lizana, "Circuit Topologies, Modeling, Control Schemes, and Applications of Modular Multilevel Converters," *IEEE Transactions on Power Electronics,* vol. 30, no. 1, pp. 4-17, 2015.

[47] S. Mikkili, R. Krishna, P. Bonthagorla and T. Senjyu, "Performance Analysis of Modular Multilevel Converter with NPC Sub-Modules in Photovoltaic Grid-Integration," *Applied Sciences,* vol. 12, no. 3, p. 1219, 2022.

[48] A. Marquez, J. I. Leon, S. Vazquez, L. G. Franquelo and M. Perez, "A comprehensive comparison of modulation methods for MMC converters," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, Beijing, 2017.

[49] Ngoc-ThinhQuach, S. H. Chae, S. Lee, H.-C. Kim and E.-H. Kim, "Analyzing Modulation Techniques for the Modular Multilevel Converter," *International Journal of Computer and Electrical Engineering,* vol. 8, no. 4, pp. 259-271, 2016.

[50] Y. Deng, Y. Wang, K. H. Teo and R. G. Harley, "Space vector modulation method for modular multilevel converters," in *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, Dallas, 2014.

[51] R. Chakraborty and A. Dey, "Sample based SVM technique suitable for digital implementation of any level Modular Multilevel converter," in *2018 IEEE 13th International Conference on Industrial and Information Systems (ICIIS)*, Rupnagar, 2018.

[52] Y. Deng, Y. Wang, K. H. Teo, M. Saeedifard and R. G. Harley, "Optimized Control of the Modular Multilevel Converter Based on Space Vector Modulation," *IEEE Transactions on Power Electronics,* vol. 33, no. 7, pp. 5697-5711, 2018.

[53] R. W. A. A. De Doncker, D. M. Divan and M. H. Kheraluwala, "A three-phase soft-switched high-power-density DC/DC converter for high-power applications," *IEEE Transactions on Industry Applications,* vol. 27, no. 1, pp. 63-73, 1991.

[54] B. Zhao, Q. Song, W. Liu and Y. Sun, "Overview of Dual-Active Bridge Isolated Bidirectional DC-DC Converter for High-Frequency-Link Power-Conversion System," *IEEE Transactions on Power Electronics,* vol. 29, no. 8, pp. 4091-4106, 2014.

[55] H. Qin and J. W. Kimball, "Generalized average modeling of dual active bridge dc-dc converter," *IEEE Transactions on Power Electronics,* vol. 27, no. 4, pp. 2078-2084, 2012.

[56] J. A. Mueller and J. W. Kimball, "An improved Generalized Average Model of DC-DC Dual Active Bridge Converters," *IEEE Transactions on Power Electronics,* vol. 33, no. 11, pp. 9975-9988, 2018.

[57] J. A. Mueller and J. W. Kimball, "Modeling Dual Active Bridge Converters in DC Distribution Systems," *IEEE Transactions on Power Electronics,* vol. 34, no. 6, pp. 5867-5879, 2019.

[58] H. Bai, C. Mi, C. Wang and S. Gargies, "The Dynamic Model and Hybrid Phase-Shift Control of a Dual-Active-Bridge Converter," in *2008 24th Annual Conference of IEEE Industrial Electronics*, Orlando, FL, 2008.

[59] H. K. Krishnamurthy and R. Ayyanar, "Building Block Converter Module for Universal (AC-DC, DC-AC, DC-DC) Fully Modular Power Conversion Architecture," in *2007 IEEE Power Electronics Specialists Conference*, Orlando, FL., 2007.

[60] H. Zoladek, "The Topological Proof of Abel-Ruffini Theorem," *Topological Methods in Nonlinear Analysis,* vol. 16, no. 2, pp. 253-265, 2000.

[61] T. F. Hain and D. B. Mercer, "Fast Floating Point Square Root," *sign,* pp. 1-7, 2005.

[62] A. Wunderlich and E. Santi, "Digital Twin Models of Power Electronic Converters Using Dynamic Neural Networks," in *IEEE Applied Power Electronics Conference and Exposition*, 2021.

[63] A. S. Wunderlich, K. Booth and E. Santi, "Hybrid Analytical and Data-Driven Modeling Techniques for Digital Twin Applications," in *2021 IEEE Electric Ship Technologies Symposium (ESTS)*, Arlington, VA, 2021.

[64] J. Villa and Y. Zimmerman, "Reproducibility in ML: why it matters and how to achieve it," 25 May 2018. [Online]. Available: https://www.determined.ai/blog/reproducibility-in-ml. [Accessed 24 March 2022].

[65] C. Liu, C. Gao, X. Xia, D. Lo, J. Grundy and X. Yang, "On the Replicability and Reproducibility of Deep Learning in Software Engineering," *ACM Trans. Softw. Eng. Methodol.,* vol. 1, no. 1, pp. 1-34, 2020.

[66] H. S. Krishnamoorthy and T. N. Aayer, "Machine Learning based Modeling of Power Electronic Converters," in *IEEE Energy Conversion Conference and Exposition (ECCE)*, Baltimore, MD, USA, 2019.

[67] S. Mohagheghi, R. G. Harley, T. G. Habetler and D. Divan, "Condition Monitoring of Power Electronic Circuits Using Artificial Neural Networks," *IEEE Transactions on Power Electronics ,* vol. 24, no. 10, pp. 2363-2367, 2009.

[68] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders and K.-R. Muller, "Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications," *Proceedings of the IEEE,* vol. 109, no. 3, pp. 247-278, 2021.

[69] D. Gunning, M. Stefic, J. Choi, T. Miller, S. Stumpf and G.-Z. Yang, "XAI-Explainable artificial intelligence," *Science Robotics,* vol. 4, no. 37, 2019.

[70] M. Capra, B. Bussolino, A. Marchisio, G. Masera, M. Martina and M. Shafique, "Hardware and Software Optimizations for Accelerating Deep Neural Networks: Survey of Current Trends, Challenges, and the Road Ahead," *IEEE Access,* vol. 8, pp. 225134-225180, 2020.

[71] P. Palangpour, "FPGA implementation of PSO algorithm and neural networks," Missouri University of Science and Technology, Rolla, MO, 2010.