Discrete Time State-Space Modeling Framework for Switched-Mode Power Supplies

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Jared Alexander Baxter

May 2024

© by Jared Alexander Baxter, 2024
All Rights Reserved.

for Mamaw

Acknowledgments

The researcher wishes to express sincere gratitude and appreciation to the members of the doctoral committee: Dr. Daniel J. Costinett, Dr. Nicole McFarlane, Dr. Leon M. Tolbert, Dr. Kevin L. Tomsovic. The support from this committee through their guidance and encouragement contributed enormously to the completion of this dissertation.

The researcher is very appreciative of his colleges while completing the work: Ling, Chongwen, Saeed, Quillen, Jordan, Gabriel, Jie, Ruiyang, Andrew, Spencer, Kamal, Meave, Jingjing, Peter, Joey, Kody, Arka, Doug, as well as many others.

A special and sincere appreciation is extended to Dr. Daniel J. Costinett for his time, encouragement, and extraordinary level of patience throughout the study.

The researcher would also like to thank his parents, Jerry and Deanna, as well as his future wife, Alli, for their support and patience throughout the study.

Abstract

Electrical power consumption has become ever prominent in modern society. Switched-mode power supplies, now more than ever, have become a foundation for residential, commercial, and industrial electrical needs. Increasing diverse electrical demands requires numerous advanced power converters where modeling plays a vital role in the design of these converters. Commonly, modeling is completed using either dedicated hand analysis, which must be completed individually for each topology, or time-stepping circuit simulations, which are insufficiently rapid for broad analysis considering a wide range of potential designs or operating points. Discrete time state-space modeling of switching converters has shown merits in rapid analysis and generality to arbitrary circuit topologies but is hampered by difficulty incorporating nonlinear elements. This thesis provides a framework to correct limitations of discrete time state-space modeling and leverage its utility in the converter design process.

Table of Contents

1 Introduction **1**

2 Literature Review **7**

2.1 History . . . 7

2.2 Analysis and Simulation of Power Converters . . . 9

2.2.1 SPICE . . . 9

2.2.2 PLECS . . . 14

2.2.3 Discrete Time State-Space Modeling . . . 17

2.2.4 Other Analysis Techniques . . . 20

2.3 Optimization of Power Electronics . . . 21

2.3.1 Brute Force . . . 22

2.3.2 Evolution Algorithms . . . 23

2.3.3 Swarm Algorithms . . . 25

2.3.4 Gradient Methods . . . 28

2.3.5 Others . . . 29

2.4 Discussion . . . 29

3 Discrete Time State-Space Modeling **32**

3.1 Derivation . . . 32

3.2 Limitations . . . 34

3.2.1 Singular $\boldsymbol{A}_i$ . . . 34

3.2.2 Linear Circuit Requirement . . . 37

3.3 State-Independent and State-Dependent Switching . . . 38

3.3.1 Steady-State Solution with State-Dependent Switching 41
3.3.2 Example Convergence of Synchronous Buck 45
3.4 Passive Switching Correction 48
3.4.1 Algorithm 50
3.5 Simulation Verification 54
3.6 Experimental Verification 56
3.7 Jacobian Method 57
3.8 Conclusions 59

4 Converter Optimization **60**
4.1 Optimized Component Selection 60
4.2 Optimization Methodology 69
4.3 Example Design Optimizations 74
4.3.1 Battery Charger 74
4.3.2 48-to-1V 87
4.4 Conclusions 102

5 Conclusions **108**
5.1 Summary of Work 108
5.1.1 Publications 110
5.2 Contributions 110

Bibliography **112**

Appendices **127**
A Circuit Parsing 128
A.1 Network Theory 129
A.2 Netlist to State-Space 136
A.3 Normal Tree 138
A.4 Finding Loop and Cutset Matrices 138
A.5 Solving for State Matrices 140
A.6 Transformers 143

B AURA User Guide 146
B.1 Database 146
B.2 Modeling 152
B.3 Optimization 161

Vita **162**

List of Tables

2.1 Advantages, Disadvantages, and Applications of Optimization Methodologies for Power Converters. 27

3.1 Buck Converter Deadtime Solution 43
3.2 Example Synchronous Buck Parameters 46
3.3 Categories of Frequency Dynamics in the Proposed Method 51
3.4 Steady-State Solve Time of Simulation Methods 53
3.5 Steady-State Solve Converter Properties 53

4.1 Buck Converter Deadtime Solution 75
4.2 EPC GaN Devices 75
4.3 Switch Capacitor Fibonacci Converter Parameters 88
4.4 Series Capacitor Buck / Switched Capacitor Parameters 88
4.5 Toroidal Inductor Parameters 99

5.1 Author's Contributions to the Field 111
2 Netlist of a Buck Converter 139
3 Currently known transistor parameters 148
4 Currently known inductor parameters 149
5 Currently known capacitor parameters 154

List of Figures

1.1 Example model of the power converter design process. 3
1.2 Flow diagram of power converter optimization. 5

2.1 SPICE algorithm flow diagram. 11
2.2 Flow diagram of power converter time stepping modeling. 13
2.3 PLECS algorithm flow diagram. 16
2.4 Quasi-Newton steady-state solve breakdown. 18
2.5 Discrete time state-space modeling solve breakdown. 19

3.1 Synchronous buck converter and equivalent transistor representation used in Fig. 1.2 and Fig. 3.2. 40
3.2 Waveform of a synchronous buck converter after the high side device turns off. The initial guessed deadtime length is too long between active switching actions. 42
3.3 Passive switching error for the synchronous buck converter considering only the discrete sampling points. 46
3.4 Example complete waveforms (solid blue lines) and discrete sampling points (green diamonds) for both deadtimes, and two operating points, of the synchronous buck converter. 47
3.5 Passive switching error for the synchronous buck converter considering full continuous-time waveforms. 49
3.6 Non-synchronous buck converter switch node waveform with a small stray loop inductance added in series with transistor. 51

3.7 Flow diagram of how multi-resonant and single-resonant circuits are addressed in the proposed method. 52
3.8 Flyback converter (EVAL-CN0342-EB1Z) schematic (a), experimental waveforms (b), modeled waveforms (c). 55
3.9 Buck-boost converter prototype and schematic (a), experimental waveforms (b), modeled waveforms (c). 55

4.1 Flowchart of schematic-level design optimization. In an inner loop, the discrete time steady-state model is iterated to find a valid converter solution. In an outer loop, the design (including e.g. component selection, switching frequency) is iterated to optimize an objective function. 61
4.2 Datasheet of an EPC2055 GaN FET. 62
4.3 Datasheet of the XGL3520-561ME Coilcraft inductor. 63
4.4 Datasheet of the GRM31CR60J107MEA8 Murata capacitor 64
4.5 Continuous representation of a transistor power loss swept across output capacitance and conduction resistance. 66
4.6 Example iterative transistor selection from discrete devices. Individual FETs are shown with a Voronoi diagram (dashed gray lines) showing the regions closest to each discrete device. Cases of selecting subsequent iteration are shown in. 68
4.7 Waveform of a synchronous buck converter after the high side device turns off. The initial guessed deadtime length is too long between active switching actions. Therefore an additional dependent switching interval $(i, 2)$ is inserted into the independent switching interval i to resolve the error in dependent switching actions. 72
4.8 Power loss values for steady-state topology implementations. 76
4.9 Linearized surface of power loss values taken from Fig. 4.8. 76
4.10 Example weighting function 77
4.11 Combined weighting function and linearized power loss surfaces 77

4.12 Buck-boost converter schematic (a) optimization process (b) modeled results (c). 79
4.13 4V switched capacitor Fibonacci sequence schematic (a) optimization process (b) modeled results (c). 80
4.14 Buck switched capacitor schematic (a) optimization process (b) modeled results (c). 81
4.15 8V switch capacitor Fibonacci sequence schematic (a) optimization process (b) modeled results (c). 82
4.16 Switched capacitor Fibonacci sequence schematic (a) optimization process (b) modeled results (c). 83
4.17 All possible discrete device combinations for the 8V switch capacitor Fibonacci topology. 84
4.18 Brute force iteration of all discrete device combinations for the 8V switch capacitor Fibonacci topology. 84
4.19 Some possible discrete device combinations for the 4V switch capacitor Fibonacci topology. 85
4.20 Brute force iteration of some discrete device combinations for the 4V switch capacitor Fibonacci topology. 85
4.21 Switched capacitor Fibonacci prototype (a) experimental waveforms (b) modeled vs experimental results at 30 W output power (c). 86
4.22 Switched capacitor Fibonacci transistor design space where all converter transistors are the same. The optimal point, shown by the black circle, is the EPC2055 transistor. P_{loss} is a weighted mean of hundreds of simulations for each transistor between the input voltage and output power range 88
4.23 Switch capacitor Fibonacci converter with IC modules indicated by the blue dashed boxes. 89
4.24 Switch capacitor Fibonacci converter IC weighted average power loss (W) for varying device widths. 89
4.25 Switch capacitor Fibonacci converter IC cell area (m^2) for varying device widths. 90
4.26 Switch capacitor Fibonacci IC prototype. 90

4.27 Switch capacitor Fibonacci converter IC results. 91
4.28 GA-based Pareto front for all topologies (a), convergence from multiple initial conditions of the proposed method when optimizing for efficiency only (b) and measured-vs-modeled efficiency of the prototype up to $I_{out} = 10$ A (c). 92
4.29 Over 9000 different solutions to the series cap / switched cap step down topology solved using a genetic algorithm to determine the Pareto front. . . 94
4.30 3-phase series capacitor buck schematic. 94
4.31 5-Phase series capacitor buck schematic. 95
4.32 3-phase series capacitor buck with switched-capacitor step down schematic. . 95
4.33 LEGO-POL schematic. 96
4.34 Multi-Inductor Hybrid schematic. 97
4.35 Dickson^2 schematic. 97
4.36 Prototype PCB photograph (a) and thermal image at $I_{out} = 10$ A (b) 99
4.37 Thermal image of series cap/switched cap at 20 A output current with toroid inductors. 100
4.38 AC series resistance of chosen inductor as shown in Table 4.5 using impedance analyzer. 100
4.39 Prototype PCB photograph (a) and thermal image at $I_{out} = 30$ A (b) using toroid inductors and altered output connection. 101
4.40 Series cap/switched cap at fs = 200 kHz with toroid inductors. 103
4.41 Series cap/switched cap at fs = 500 kHz with toroid inductors. 103
4.42 Series cap/switched cap at fs = 1 MHz with toroid inductors. 103
4.43 Series cap/switched cap at fs = 2 MHz with toroid inductors. 105
4.44 $V_{ds,M4}$ (a) and L_1 (b) modeled and experimental waveform comparison. . . . 106
4.45 Updated Pareto front and prototype selection using inductor from Table 4.5. 107

1 Buck Converter for State-Space Derivation. 130
2 Undirect and direct graphs. 130
3 Path examples 130
4 Connected graph examples 131

5 Loops . . . 131
6 Trees . . . 133
7 Cutsets . . . 133
8 Incidence Matrix . . . 133
9 Reduced Incidence Matrix . . . 135
10 Loop Matrix . . . 135
11 Fundamental Loop Matrix . . . 135
12 Cutset Matrix . . . 137
13 Fundamental Cutset Matrix . . . 137
14 Sorting Incidence Matrix . . . 139
15 Reduced Row Echelon Form of Incidence Matrix . . . 141
16 Buck Converter Fundamental Cutset Matrix . . . 141
17 Loop Matrix of Buck Converter . . . 142
18 KVL and KCL equations separated by tree and element type. . . . 144
19 Comparison of hand calculations and parser results. . . . 145
20 LTSpice example circuit model. . . . 154
21 Solved steady state waveforms. . . . 160
22 Solved steady-state waveforms with correct dioe behavior. . . . 160

Chapter 1

Introduction

Power electronics is growing rapidly in both applications and component technologies. Now more than ever, the everyday activities of society are highly dependent on electrical energy. Current market drivers include consumer electronics, electric vehicles, smart grid, LED lighting, Internet of Things (IoT), aerospace, defense, wireless power transfer, and implantable biomedical devices. In 2005, it was reported that 30% of all electric power generated was processed by power electronics somewhere between generation and load, and as much as 80% of power generated will utilize power electronics before reaching the load by 2030 [1]. Additionally, it is estimated electricity demand will continue to annually increase by 1% through 2050, and the human population without access to electricity will decrease from 25% today to less than 8% by 2030 [2, 3, 4].

With this increased demand in electricity, there is also a demand for increased performance in power electronics applications. One of these emerging applications is vehicle electrification where efficiency, power density, specific power, and cost are high priorities for improvement [5]. This increased dependency and demand for power electronics has further driven new technologies that increase converter performance, such as wide bandgap (WBG) transistors. However, these new WBG devices, while having superior properties to traditional silicon, present new design challenges [6]. The reduced losses associated with WBG devices place a spotlight on passives and parasitic elements in converters as potential boundary points in converter performance improvement.

Emerging markets such as photovoltaics and electric vehicles have driven more demand for the power electronics industry, and new technologies have pushed the boundary of performance. Recently, the design automation and reliability of power electronics converters has received more attention [7]. As the demand for power electronics continues to increase, the need to produce optimal converters with state-of-the-art technology at a faster pace grows. Currently, one of the most time consuming aspects of circuit development is the modeling and design of power converters, specifically in steady-state [8].

Currently the power electronics design process is dominated by initial topology selection, analysis, and optimization. This process for power electronics is a conglomerate of approximations, engineering intuition, trends, and prototype/revision cycles. A simplified V-model diagram of the converter design process is shown in Fig. 1.1. On the left side of the "V", the design process is carried out, transitioning from an application-specific design specification to a complete design suitable for prototyping. On the right side of the "V", the design is tested and validated, often with multiple inner-cycles of revision that may revert back to the initial design stage. As an example, [9] details a limited design process in which multiple candidate designs are assessed using general models, then gradually refined with more accurate, but complex, modeling efforts and empirical measurements to progress toward a final implementation.

A narrower look at the left hand side of the "V" in Fig. 1.2 shows the initial broad-scale optimization before detailed design work is completed. The design starts with a recognition of design objectives and constraints. Constraints are boundary conditions to which the converter design must adhere, while objectives are the goals or metrics that are either being maximized or minimized. Typical converter design constraints might include input and output voltage, current, and/or power values. Contrarily, objectives are metrics such as efficiency, power density, and/or specific power. From these objectives and constraints, a topology, operating method, device selection, and control strategy are optimized. There are many different methods that can be used to perform this optimization depending on the formulation of constraints and objective functions. Also, within the optimization function or method, the evaluation or feasibility of each iteration or step is based on steady-state waveforms, analytical equations, or both. For example, for each topology to be effectively

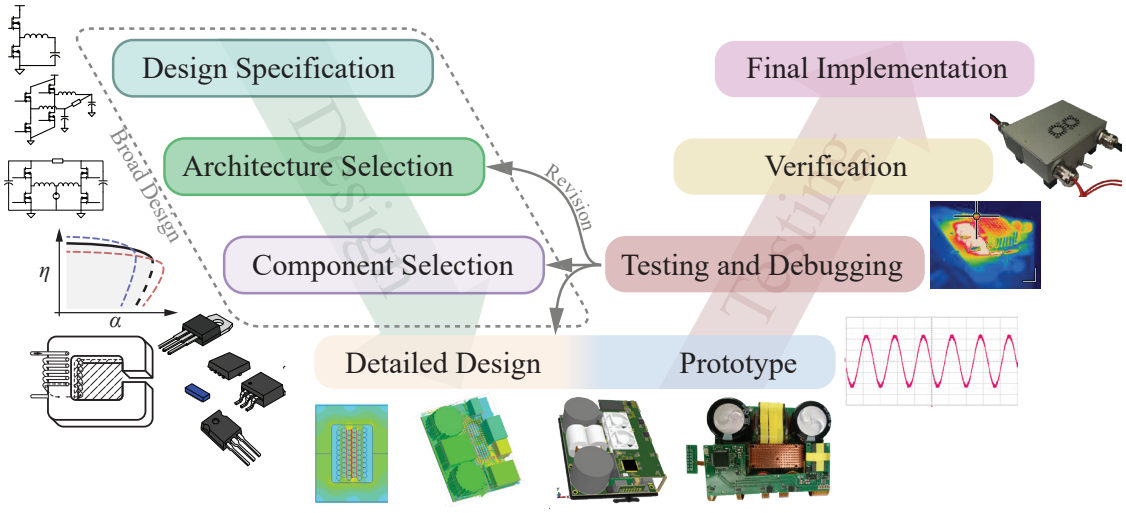


Figure 1.1: Example model of the power converter design process.

compared, they must be equally optimized and simulated. For a large number of converters with complexities such as zero voltage switching (ZVS), resonance, and/or paralleled devices or phases, it can take a long time using current design techniques to find an optimum converter, especially for a multi-objective solution.

In the design stage, analysis of the converter is predominately focused on periodic steady-state operation. Ongoing developments in multi-domain simulation, combined with multi-objective optimization techniques, seek to automate much of the detailed design process [10, 11, 12]. These developments will eventually replace hardware prototype/revision cycles with "virtual prototyping" in simulation, accelerate the design process, and reduce the time-to-market for new technologies [13]. With increasing fidelity, developments in a wide variety of software simulation tools may signal replacing a significant portion of the need for experimental validation in the long-term.

However, to achieve this level of fidelity, increasingly complex and computationally-intensive models are required. In applications requiring high performance power conversion, the scope of candidate-optimal design approaches is too broad to employ high-fidelity, model-based optimization. Even with continued advancement in the computational power available for the design process, a distinction remains between the broad-scale design, which translates a specification into a conceptual paper design, and the detailed design process, which translates this design into a physical implementation.

This thesis focuses on the broad-scale design effort, including tasks such as circuit topology and component selection, component (electrical) sizing, and modulation pattern selection. Within broad-scale design, simplified models are used, either in simulation or derived from dedicated mathematical analysis. Simplified models allow for a broader scope of designs to be considered within a feasible timeframe, narrowing to a single or small number of candidate designs suitable for detailed design and prototyping.

The remainder of this document is organized into the following sections. Chapter 2 provides a literature review of what others have done to fully and partially model and optimize power converters. Chapter 3 discusses discrete time state-space modeling, a method that has shown merits of performing rapid modeling of arbitrary converter designs. The chapter also expands on some of the method's limitations and how to overcome them.

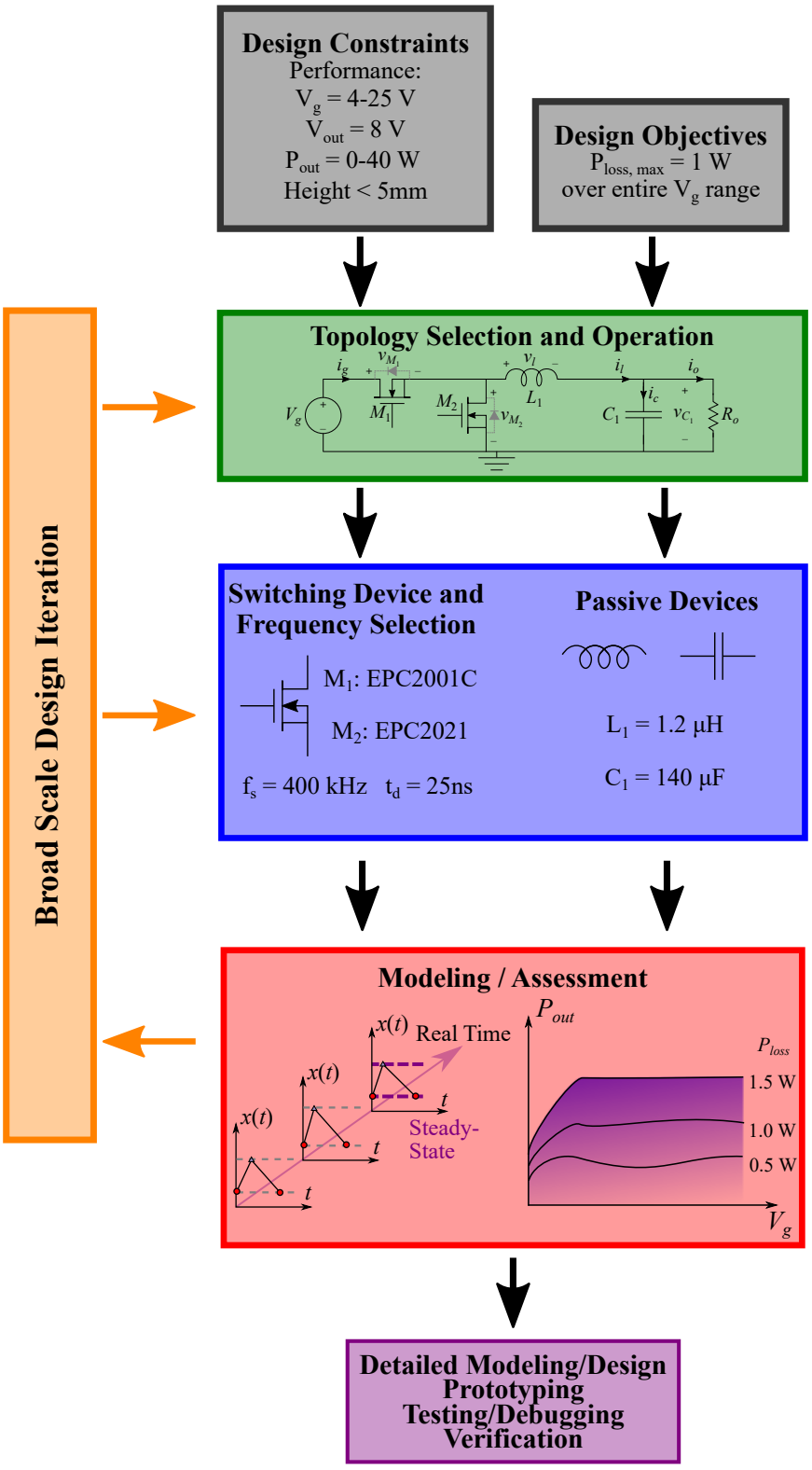


Figure 1.2: Flow diagram of power converter optimization.

Chapter 4 discusses optimization techniques that utilize and take advantage of discrete time state-space modeling. Chapter 5 considers the contributions of the work and author. Appendix A showcases a method to parse a circuit to use state-space modeling quickly. Appendix B provides a brief user guide to utilizing the proposed framework.

Chapter 2

Literature Review

Power converter optimization not only involves the ideal optimization result of having fast convergence to the optimal solution, but also requires a fast and accurate simulator or analysis tool to fully assess the fitness value after each iteration of optimization. Therefore, in this literature review, simulators and analysis tools are reviewed along with optimization methods for power converters. The review is focused on topology-level simulation, but will also include a few examples of thermal and control optimization, as they are popular optimization topics for power converters.

2.1 History

Computer simulation has been utilized for the analysis and design of circuits since the development of power electronics in the early 1950s by R. D. Middlebrook and others. The first use of computer-based circuit modeling was on the Model VI computer at Bell Laboratories in the early 1950s [14]. The Model VI solved numerical functions for basic filtering circuits. Because of this 'rapid' analysis technique, optimization could be implemented to further improve electrical filters. After the Model VI, multiple computer programs were built for private use in the 1960s at various academic and research institutions. Improved formulation and integration schemes were developed to better handle different circuit components and non-linearities. Eventually, two programs stood out in the early 1970s using two different techniques to solve circuits: ASTAP and SPICE [15]. ASTAP, Advanced

Statistical Analysis Program, was developed by IBM using the sparse tableau method that was first published in 1971 [16, 17]. SPICE, Simulation Program with Integrated Circuit Emphasis, was developed at the University of California, Berkeley based upon CANCER, Computer Analysis of Nonlinear Circuits, Excluding Radiation, to be better suited for the public domain [18]. SPICE1, first presented in 1973, used nodal analysis but suffered from limitations due to element representation [19]. SPICE2, published in 1975, utilized modified nodal analysis (MNA) and is the basis for today's SPICE programs [20]. One reason for the development of these early computer programs was to design circuits through the use of faster modeling and analysis. Until this point, equations had to be solved by hand, thus, it was tedious to achieve a true optimization of a circuit. The development of these programs meant circuits could be analyzed quicker and thus nominal and stochastic optimization techniques, such as gradient methods, linear programming capability, quasi-newton methods, and Lagrangian methods, were quickly applied to ASTAP and SPICE through interactive systems for design optimization, APLSTAP and DELIGHT, respectively [21].

Traditional optimization techniques trace back before the 1950s; however, the augmentation of non-linear techniques and algorithms was congruent with the advent of computer-based circuit simulators in the mid 19th century, likely due to the development of 'turning machines' at the time. A framework consisting of the Sequential Unconstrained Minimization Technique (SUMT) is presented using continuous variables to optimize an inductor and a switching regulator [22, 23]. The same authors then implemented an augmented Lagrangian (ALAG) penalty function technique, drew comparisons to SUMT, and found the ALAG method to be superior. ALAG was shown to be more efficient due to only computing the first derivative and required less computation demand than SUMT [24]. Later, the same approach is used in the optimization of boost and buck-boost converters [25, 26] and a half bridge DC-DC [27, 28, 29]. These early techniques were used mainly to minimize weight for applications such as space flight [30]; however, they were typically constrained to passive device optimization, i.e. many assumptions or calculations about converter properties were done outside of the optimization process. All of the aforementioned optimization processes use continuous variables and differentiable constraint equations. Stepping outside of continuous variables could be done using branch and bound algorithms to move from

custom design requirements to discrete values; however, the scope of the problems are well defined to stay within the reasonableness of computation power at the time. For example, the optimization of switching frequency proved to be too computationally demanding and was thus replaced as a constant and then iterated through multiple optimization runs [27, 26].

Simulation and optimization techniques continued to improve and change as design-oriented analysis of power electronics developed over the years. Some simulation programs, such as SPICE, continue to be used today through many simulations platforms. Contrarily, optimization methods, such as particle swarm and genetic algorithms, were developed in the last 50 years and have been applied to power electronics applications within the last 20 years.

2.2 Analysis and Simulation of Power Converters

There are two main approaches to analyzing and simulating power converters. The first is to use broad general circuit simulators such as SPICE or PLECS to understand what transients are occurring within the circuit. The other is dedicated steady-state analysis for power converters using analytical equations that often make use of simplified waveforms. However, programs have been developed that take advantage of both dedicated models and general simulators.

The most accurate method for determining how a power converter will perform is through prototyping; however, computer-based simulation is typically less expensive and easier to make adjustments. While rapid prototyping has the potential to become more feasible due to 3-D printing technologies and other advancements, computer-based simulation is more cost effective and quicker during the early evaluation of a power converter. Therefore, the focus of this literature review will be on computer-based modeling and analysis. The following subsections will first consider some of the most popular circuit simulators and then discuss and review some of the research conducted to analyze steady-state power converters.

2.2.1 SPICE

SPICE is a simulation program that is widely used today to analyze power electronics. The program utilizes Gaussian elimination to solve a MNA matrix representation of a circuit.

Originally, SPICE1 used nodal analysis; however, nodal representation of a circuit cannot model current controlled or voltage sources. Thus, SPICE2 was developed to use MNA and is the method used in SPICE simulators today.

In nodal analysis, Kirchhoff's current law (KCL) is applied to each node to generate a set of independent equations describing the circuit. These equations can be written as

$$\boldsymbol{Y}\boldsymbol{v} = \boldsymbol{j} \tag{2.1}$$

where $\boldsymbol{v}$ is the vector of voltages with respect to the ground node, $\boldsymbol{Y}$ is the node admittance matrix, and $\boldsymbol{j}$ is the nodal current source vector. To deal with the inability to model current controlled sources or voltage sources originally, Norton equivalents were used to replace voltage sources with current sources. Later, the MNA method was developed to incorporate voltage sources into the matrices solved. The MNA matrix can be expressed as

$$\begin{bmatrix} \boldsymbol{Y_R} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{i} \end{bmatrix} = \begin{bmatrix} \boldsymbol{j} \\ \boldsymbol{e} \end{bmatrix} \tag{2.2}$$

where $\boldsymbol{Y_R}$ is the reduced nodal matrix, $\boldsymbol{B}$ is the partial derivative of KCL equations with respect to the added current variables, and $\boldsymbol{C}$ and $\boldsymbol{D}$ complete the current sensing branch elements differentiated by $\boldsymbol{i}$. An example of this process can be found in [31].

The basic block diagram of SPICE is illustrated in Fig. 2.1 [32]. The flowchart for SPICE begins by formulating an initial operating point, and then, if there are non-linear devices in the circuit, linear companion models are constructed. Next, the nodal matrix is formed as shown in (2.2). The model is then solved using numeric integration (Backward Euler, trapezoidal, or gear-2) and tested for convergence. Newton's method is used to adjust an initial operating point if the model does not converge for non-linear elements. Once convergence is met, if a transient simulation is selected, the next time point is calculated and the process starts over until the end time is reached [32].

Newton's method is a gradient-based method used by SPICE and many other simulators to check convergence of non-linear models. Newton's method is used as opposed to steepest descent algorithm, since it takes a more direct route to the objective. For an optimum

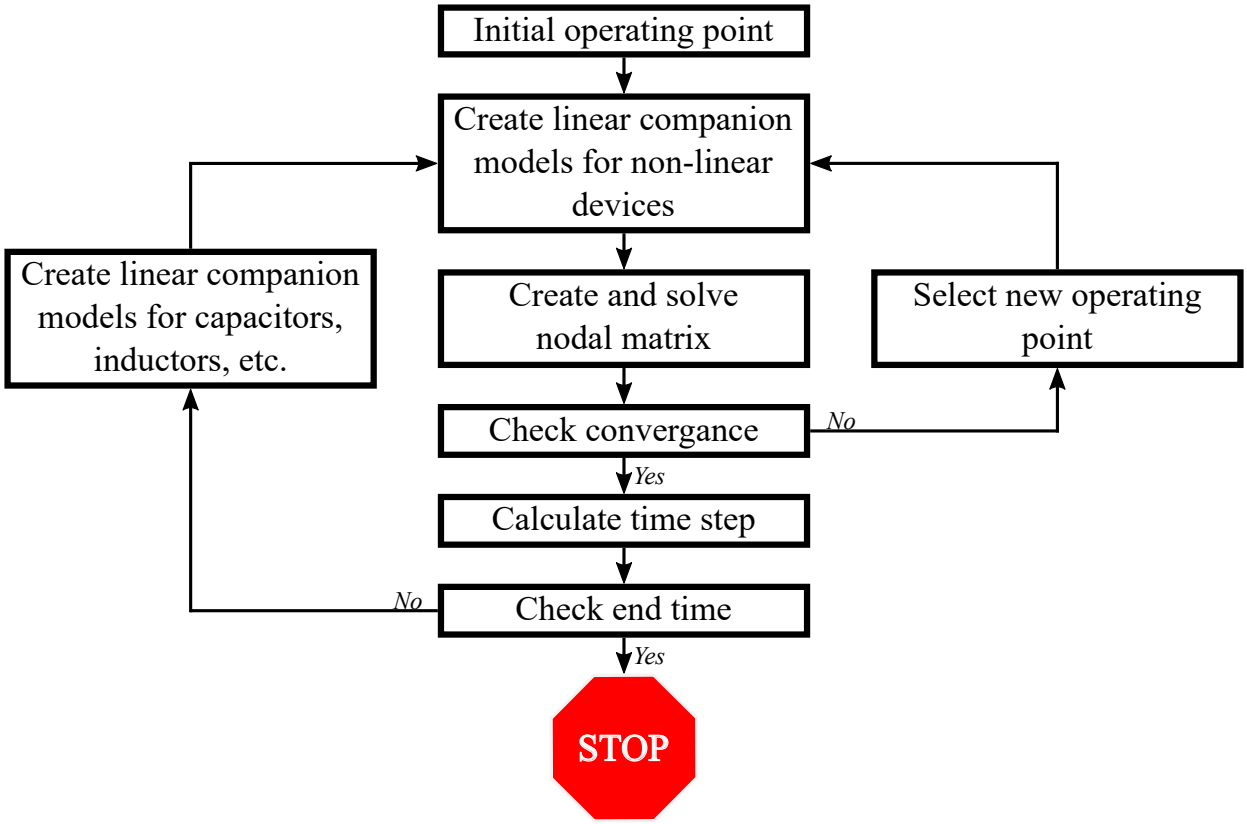


Figure 2.1: SPICE algorithm flow diagram.

solution of zero, $f(x^*) = 0$, the steepest descent algorithm is

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \gamma \nabla \boldsymbol{f}(\boldsymbol{x}_n) \tag{2.3}$$

where γ represents some rate parameter and $\nabla \boldsymbol{f}(\boldsymbol{x}_n)$ is the gradient of $\boldsymbol{f}(\boldsymbol{x}_n)$. Newton's method algorithm is

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \gamma \boldsymbol{H}^{-1} \nabla \boldsymbol{f}(\boldsymbol{x}_n) \tag{2.4}$$

where $\boldsymbol{H}$ is the Hessian and typically $\gamma = 1$. Newton's method can be generalized for one dimension, called the Gauss-Newton method

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \gamma \boldsymbol{J}^{-1} \boldsymbol{f}(\boldsymbol{x}_n) \tag{2.5}$$

where $\boldsymbol{J}$ is the Jacobian matrix. Often when documents refer to the Newton's method they use (2.5) instead of (2.4), since the application is typically only solving one dimension of variables. This simplification saves significant time in calculations, as the full Hessian is not solved after each iteration.

The popularity of SPICE has made it a standard for device and circuit modeling. A major advantage of SPICE over other simulators is the familiarity of academia and industry with creating models and performing analysis using one of its platforms. Also, due to its longevity, there is a legacy of circuits and models that have been created and analyzed using the program. However, one of the reasons that SPICE is so widely popular, its generality, is its limitation for power converter modeling. As described above, SPICE uses nodal analysis, which is preferred for the design of circuits without discontinuities, such as analog circuits and filter design, its originally intended purpose. However, the switching action of power electronic circuits causes the simulation time of SPICE to increase due to the need to model these large transients using time domain simulations. These spikes can be mitigated by adding parasitic loss elements to the circuit; however, this also increases complexity of the circuit.

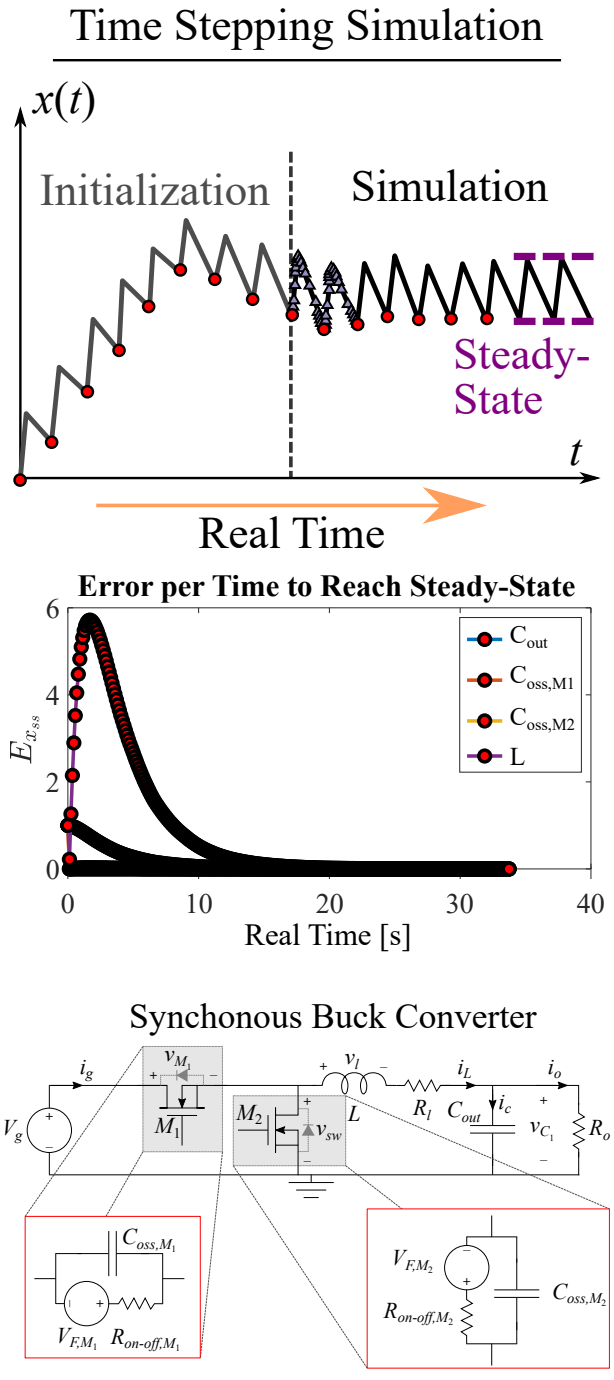


Figure 2.2: Flow diagram of power converter time stepping modeling.

An illustration and simulation to showcase the long initialization time is shown in Fig. 2.2. A synchronous buck converter is simulated until the converter's waveforms reach steady-state. The top figure shows an illustration of the convergence to steady-state with an initialization period to settle the converter waveforms close to the steady-state value. The red dots indicate completion of a full switching period while the triangles indicate the small time stepping that occurs within each period to simulate the waveform. The middle figure of Fig. 2.2 shows a simulated example of the error

$$E_{\boldsymbol{x}_{ss}} = |\boldsymbol{x}(T_s) - \boldsymbol{x}(0)| \tag{2.6}$$

after each switching period for the capacitors and inductors in the circuit. As seen in the figure, there is a large initialization error of C_{out} reaching almost 6 times the steady-state value and requiring almost 20 seconds of real time simulation to reach within 50% error of the final steady-state value.

2.2.2 PLECS

Piecewise Linear Electrical Circuit Simulation (PLECS) implements a set of piece-wise linear state-space equations to represent circuits with ideal linear or switching elements [33]. Each time subinterval of a converter is modeled by a set of passive elements described by

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{2.7}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \tag{2.8}$$

where $\boldsymbol{u}$ is a vector representing input sources, $\boldsymbol{x}$ is a vector representing state variables, $\boldsymbol{y}$ is a vector of output voltages and currents, and $\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{C}$,$\boldsymbol{D}$ are constant real matrices for each switching interval of the circuit.

Since the non-linear dynamics of the switching transition are not included in this formulation, information about the converter is lost and not represented by the state-space equations. However, the loss of energy associated with every switching action can

be reintroduced by providing a transition matrix that takes the switching energy lost out of the system before setting the initial conditions of the next time interval.

The steady-state analysis technique lends itself to power electronics simulation. In PLECS, a 'Switch Manager' is used to determine when to switch from one state to another and when steady-state is reached as shown in Fig. 2.3. After every simulation step, the Switch Manager evaluates the position of each switch and determines if a switching action is necessary. To compute a steady-state analysis, PLECS uses the quasi-Newton method with Broyden's update.

Quasi-Newton methods can be used as an alternative to Newton's method. They are typically used if the computation of the Jacobian or Hessian take too long to compute for each iteration or are unavailable. Generally, for optimization problems, approximations of the Jacobian are based on the secant method. There are many different forms of quasi-Newton methods; however, Broyden's method is used by PLECS.

Maksimovic describes Broyden's method used by PLECS in [8]. The analysis requires the number of state variables plus-one simulations in order to compute the initial Jacobian matrix using the finite-difference quasi-Netwon method. In each simulation, one state variable is slightly perturbed and the calculated difference is used to formulate the Jacobian. After each update, only one simulation is solved using

$$\boldsymbol{J}_{n+1} = \boldsymbol{J}_n + \frac{\big(\boldsymbol{x}_{n+1} - \boldsymbol{F}_T(\boldsymbol{x}_{n+1})\big)(\boldsymbol{x}_{n+1} - \boldsymbol{x}_n)^T}{||\boldsymbol{x}_{n+1} - \boldsymbol{x}_n||^2}. \tag{2.9}$$

This leads to fewer simulations and thus faster convergence of the model to steady-state.

An illustration and simulation to showcase the quickened simulation time is shown in Fig. 2.4. A synchronous buck converter is simulated until the converter's waveforms reach steady-state. The top figure shows an illustration of the convergence to steady-state. The red dots indicate completion of a full switching period while the triangles indicate the small time stepping that occurs within each period to simulate the waveform. There is an initialization stage until a circular topology is found, meaning there is a consistant switching sequence between periods. Then the Jacobian is calculated as indicated by the large green rectangle.

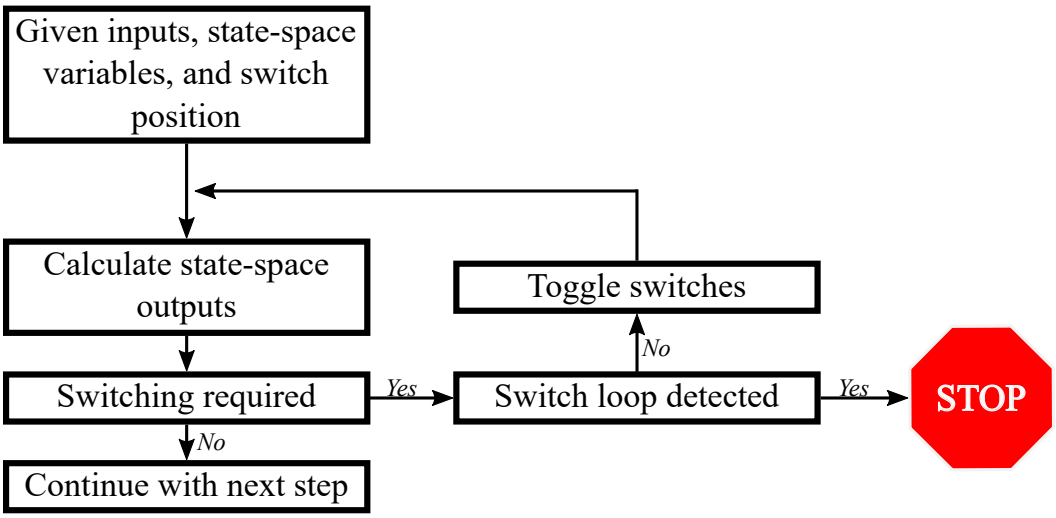


Figure 2.3: PLECS algorithm flow diagram.

The next initial condition is then calculated using the Jacobian to 'jump' closer to the steady-state value. The method then alternates between time-domain simulations and quasi-Newton iterations utilizing the updated Jacobian matrix each time.

The middle figure of Fig. 2.4 shows a simulated example of the error after each switching period for the capacitors and inductors in the circuit. Compared to Fig. 2.2, Fig. 2.4 shows a much faster real time to reach steady-state as well as a more inconsistent convergence. The error reduction for the quasi-Newton method is not smooth like the time domain simulation. The error can be seen jumping around until it reaches close to the steady-state value.

Power converter state-space formulation for analysis is advantageous since it is more intuitive to how designers calculate loss in power electronics. The method categorizes a circuit into different switch positions to simulate, essentially expressing each switch position as a different circuit or state-space equation. Then, if a switching action is required, the associated waveforms of the switching action are calculated based on the unique qualities of the switching action in addition to the previous and next state. This method effectively decouples the simulation so each segment can be simulated and analyzed using only the proper analysis techniques for that specific period. Spikes in waveforms can still be seen depending on the initial conditions and total response of a time interval.

2.2.3 Discrete Time State-Space Modeling

Discrete time state-space modeling is a uniform approach to rapidly simulate arbitrary power converter designs. Instead of reducing the error of the initial and final state values, discrete time state-space modeling directly solves for the steady-state vector $x(0) = x(T_s)$ in closed-form. However, the solution requires a predefined switching pattern, which may not be known a priori in the presence of state-dependent switching, such as diode switching or current programmed modulation. In this event, the technique again requires an iterative solution, this time altering the switching times (as opposed to the initial states), as shown in the middle figure of Fig. 2.5 [34]. More detailed analysis of discrete time state-space modeling is given Chapter 3.

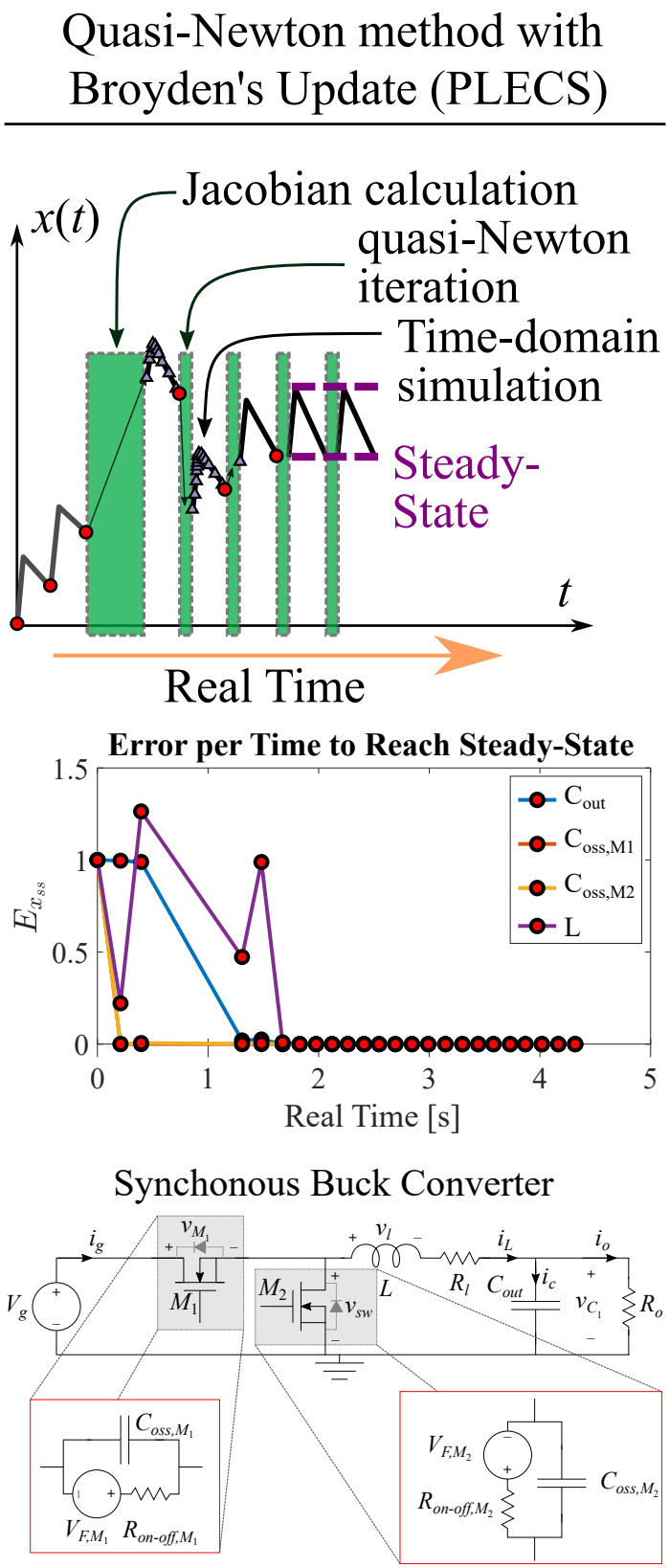


Figure 2.4: Quasi-Newton steady-state solve breakdown.

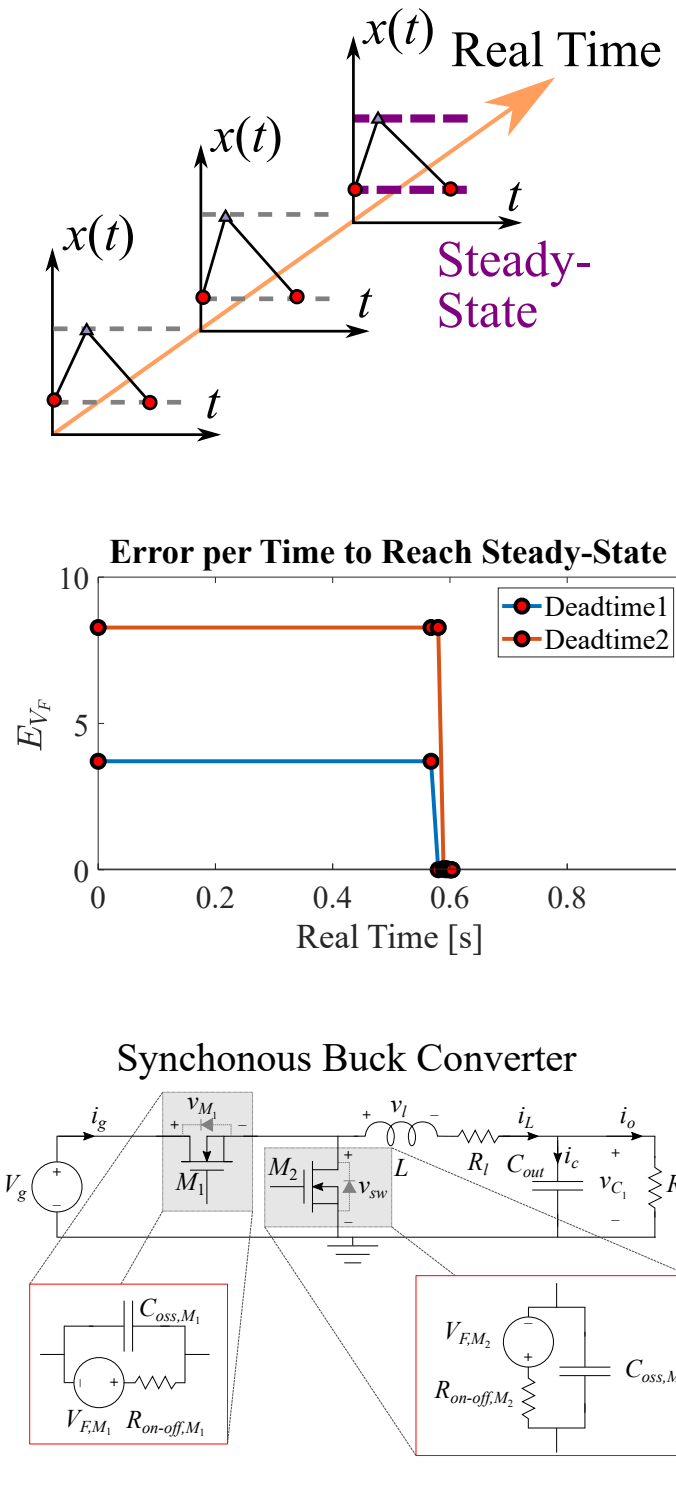


Figure 2.5: Discrete time state-space modeling solve breakdown.

2.2.4 Other Analysis Techniques

There are other analysis techniques other than SPICE and PLCES that can be used to simulate a power converter in steady-state. Other simulators such as Saber and Spectre exist but are based on SPICE and therefore share similar advantages to SPICE outlined previously. While Saber and Spectre are thought of as expanded, more thorough simulators than SPICE, their basic metrics of solving electrical circuits are similar according to their respective user guides [35, 36]. Beyond the scope of commercially available simulators, there are a few publications that try to simplify steady-state analysis of converters both for general use and specific topologies [37].

As stated in Section 2.2.2, Maksimovic describes an automated steady-state analysis tool for switching power converters. The original implementation of this automated tool uses a combination of PSpice or PETS as a circuit simulator and Mathmatica as a state-space solver. The method utilizes a quasi-Newton iteration method via an approximated Jacobian through Broyden's method to reach steady-state and appears to be identical and cited by the steady-state analysis section in the PLECS user manual [8].

A review of steady-state methods was originally presented in [38]. In addition to describing the aforementioned Newton's method with Broyden's update, the paper discusses another form of Newton's method termed globally convergent strategy. This strategy essentially ensures each step of Newton's method reduces the error of the steady-state guess. If the error increases, often due to the initial guess being distant from the solution and 'overshooting,' then the algorithm will reduce the magnitude of the step or 'backtrack' along the Newton step direction until the error reduces. This procedure ensures the method will get closer to the steady-state solution after each iteration; however, the results of the analysis show Newton's method and the Skelboe method [39] to be better. The Skelboe method is further reviewed in [40] and utilizes a linear least squares method to solve for the steady-state values without solving for the Jacobian. The non-derivative method did not match the performance of analytical Newton's method; however, depending on the circuit, the Skelboe method was on par with numerical Newtonian methods.

Others have proposed solving for the switching intervals while solving for the steady-state initial conditions of the converter [41]. This method imposes additional complexities but adds a degree of optimization for the Newton-Raphson method.

Contrary to general methods, there are analysis techniques that are applied to individual converter models to find steady-state behavior such as an interleaved boost [42] or a switch capacitor converter [43].

An optimization based steady-state analysis of power converters was performed in [44] with a focus on the harmonic contribution of each state. The contribution is a quicker steady-state analysis convergence if initial conditions are set by DC or first harmonic averaged model values. A similar technique was used in [45] using MNA instead of state-space.

A waveform converter analysis flow diagram is presented in [46] to achieve ZVS transitions to ensure optimal efficiency. This analysis is very useful to determine diode conduction and switching transitions. However, waveform analysis can be time consuming and was noted as taking longer than the optimization routine in [47, 11].

2.3 Optimization of Power Electronics

The term optimization is often misconstrued in the field of power electronics. Some view optimization as a process of equations that are solved to achieved a result [48]. While these processes are design methodologies or modeling techniques, they are not optimizations. These techniques typically make assumptions and solve a set of equations to reach a design, such as the kg method for inductor design. Sometimes these design techniques provide good standards for a given topology, such as a switch capacitor converter [43]. However, these processes limit the scope of design possibilities.

In general, even optimization techniques that truly optimize some facet of a converter, limit the possible solution set of the optimization. For example, a converter topology or a minimum voltage or current ripple is typically assumed based upon the engineer's expertise. While extreme values of variables are unlikely in certain circumstances, these assumptions theoretically eliminate possible solutions and could prevent a researcher from finding the true optimal topology and control scheme for a given converter.

The following sections describe many techniques used to optimize some aspect of a power converter. The review focuses on methodologies and the scope with which they are implemented. There are many different device modeling techniques that can be used in analytical optimization to calculate a converters performance, such as efficiency. Many of the review papers use many different techniques; however, the scope of this review does not aim to evaluate individual element models for power loss, size, or economic value. A good review of these element models is given in [49].

2.3.1 Brute Force

One method that some have used in place of true optimization is the brute force method. This method uses a set of discrete design variables and then iterates through each possible variable combination. It is the equivalent of using nested for loops in sequential programming. The program does not inherently know anything about the circuit or if it is close to an optimum; it simply tries every possible combination of variables.

Some have used this method to find an optimal solution for an inverter while integrating a SPICE simulation to check feasibility, which took several days to run [50]. There are several papers on using the brute force technique for a variety of power converter optimization problems such as reliability, passives [51], thermal, and control which can take several days to simulate. Others have tried to improve brute force iteration technique by utilizing the apparent independence between some variables. This process effectively optimizes each set of independent variables separately and then combines the solutions together, reducing the total number of iterations. Incorporating parent-child relationships between variables can also reduce iteration time [52]. Others only choose to brute force through variables that are complex and highly dependent on many constraints, such as frequency, while using MATLAB's fmincon function [53, 54].

Recent publications have been presented by ETH Zurick on multi-objective optimization to use brute force techniques to compare efficiency and power density [46]. These techniques often iterate through frequency selection and then move on to solving a system of design equations with modulation scheme and passive values assumed or iterated through. The

optimization is reduced to the best component design given inductance and capacitance values to improve efficiency.

Overall, the advantages of the traditional brute force method are that the optimal solution will be found for the given design space; however, for the design space to be practical for an application, it must be large, and thus, this method takes an extreme amount of time to iterate. One method to reduce the amount of iterations is to separate variables if there is no interdependence between them. For example, if the output capacitance has little effect on the input filter then the two parameters can be iterated separately. This technique can work well for simple, small problems; however, in highly complex converters, this independence could be difficult to obtain. While increased computation capabilities have allowed the brute force method to be used more often in recent years, it is not an optimization technique and is too time consuming for all but small problems.

2.3.2 Evolution Algorithms

Evolutionary optimization is a population-based method where the goal is not to find an optimal solution but to find a population of solutions. Evolution algorithms typically start with a population of solutions that are then altered or passed to the next iteration and then re-evaluated and ranked. A new population is formed from the altered best solutions and the process repeats itself with the new population until a certain stopping criteria is met. More-in depth discussion of these methods as well as state-of-the-art formulations can be found in [55, 56].

The most popular evolutionary algorithm is the Genetic Algorithm (GA). GAs are a type of evolutionary programming that are based on natural selection in biology. The structure of GAs starts with a parent population that is then subject to mutations, random changes in variable values, and cross-overs, taking two or more individuals from the parent population to form a child population. The child and parent populations are then evaluated based on a fitness function that ranks each of the potential solutions. The highest ranking solutions then form the next parent population for the next iteration, thus hopefully improving the overall 'fitness' of the next generation and moving towards an optimal solution.

There are many different GAs that have been applied to power electronics. The non-dominated sorting genetic algorithm (NSGA) has been recently used to solve multi-objective functions with both discrete and continuous variables for the design of an auxiliary railway power supply [57] and a flyback converter [47, 58]. GA frameworks have also been developed such as the "Optimization Workbench" (OWB) [59]. OWB uses GAs as well as parameter sweeps and permutation methods to optimize a boost PFC converter. Another framework called "Optimization of Power Electronic Systems" (OPES) consists of a GA optimizer based on DARWIN and FORTRAN coded steady-state algebraic analysis equations. The authors claim that the DARWIN Plug-In makes the program more general and able to solve a variety of circuits; however, only a boost PFC is considered [60]. Furthermore, DARWIN is used to optimize the passive front end components of a diode rectifier for a voltage source inverter [61]. The use of GAs have also been used to successfully optimize the thermal design of a power converter [11, 62] and the control parameters of a simplified buck [63] and boost [64] converter. A wireless power transfer (WPT) circuit is considered in [65] where GAs are used to adjust the period and switching times to account for stability, using Newton-Raphson, and efficiency modeling.

From the above summary of literature it is apparent there are many different disciplines and scopes that GAs can be used to solve. The advantage of GAs over other optimization techniques is their ability to formulate both ordered or non-ordered variables. This allows properties such as inductor core material, topology, number of devices in parallel, and switching frequency to be represented and optimized [57]. However, the method involves some randomness associated with parent selection, mutations, and crossovers. Due to this impermanence, GAs are not guaranteed to get the 'true' optimal design but are rather designed to get near the optimal or create a Pareto front [66].

Differential Evolution (DE) is a method that is very similar to GAs. DE uses the same basic techniques outlined above for GAs; however, the implementation and control parameters are slightly different. DE is also based on numerical methods, thus, unlike GAs, it cannot optimize unordered indexed variables such as a set of inductor core properties [67]. DE is applied to a feedback network of resistors and capacitors for a buck converter and compared with other stochastic methods such as GAs and brain storm optimization in [68].

If the above method was expanded outside of the continuous domain, branch and bound techniques could be used to determine the optimized discrete variable selection. Therefore, expansion beyond RLC and frequency selection would be difficult to formulate and should be optimized through a GA instead.

From the literature described above, the main limitation present in most work is the narrow design scope. The database of components is insufficient to find a true optimal for their respective solutions. Papers either considered only one part of the converter, such as input passive components, or considered many different variables but limited possibilities of these variables. This aligns with the traditional drawback of GAs; they are good for finding Pareto fronts for medium to small optimization problems but are time consuming for large scale optimizations.

2.3.3 Swarm Algorithms

Swarm algorithms are similar to evolutionary algorithms because they are population and nature based. However, instead of using crossover or mutation to create different solutions that are based on evolution and natural selection, swarm algorithms behave similar to flying geese, bees, ants, or social movements.

Particle Swarm Optimization (PSO) is the most popular swarm algorithm for power electronics optimization, mainly for control parameters [69]. Each iteration updates the velocity of the particles. In general, there are three tendencies in how swarms tend to move: toward the center of the swarm, away from the nearest individual, and toward the best 'solution' found thus far [70]. These tendencies dictate the 'movement' of a possible solution through the variable space.

PSO has been used for a wide variety of optimizations such as power quality disturbance classification [71] and control of DC-DC converters with continuous parameters [72]. PSO also proved to be better than the hill climbing method to dynamically change the duty cycle of a PV solar array using a buck boost converter for maximum power point tracking [73]. The most comprehensive paper on PSO found [74] is the design of an inductor based on number of turns, saturation, and core thickness, NMOS and PMOS channel width, and a driver taper ratio. This study conducted a comparison of PSO, DE and pattern search to

find the best solution was PSO optimization. It had less error than the pattern search and took less than half as many iterations to solve.

Ant Colony Optimization (ACO) is another swarm algorithm being used for power converter optimization. ACO can be thought of as ants moving from their nest to a source of food; the ants tend to choose the shortest known path but some try to expand and find a more optimal path to their target, food. Further explanation of ACO and a comparison of PSO and ACO is the subject of [75]. In literature, ACO is mainly used as a circuit control optimization technique [76, 77, 78]. However, ACO has been shown to optimize feedback and power passives while including steady-state error in the fitness function [79].

Swarm algorithms overall are relatively new compared to other optimization techniques. Another technique called Brain Storm Optimization (BSO) is currently being investigated by [80], and this method could potentially be used in the future for power converter optimization.

Overall, swarm algorithms have mainly been used in a very confined scope of power converter optimization. The literature review only found limited examples that mainly focused on control and power quality optimization. This is due to the limitations of swarm algorithms to a coordinate system representation of variables. In order for the optimization method to be effective, the velocity of a particle through the design space must have direction. For expansion beyond coordinate functions, the velocity parameter loses its value to the optimization. For example, an index value to properties of materials found in [57] cannot be implemented as a swarm optimization problem. This eliminates variable possibilities such as topology, material parameters, and discrete device selection.

Table 2.1: Advantages, Disadvantages, and Applications of Optimization Methodologies for Power Converters.

Optimization Methodology	Can Optimize Unorderd Variables	Convergence	Speed	Application in Power Converter Optimization
Brute Force	-Yes, in all cases	-Guaranteed convergence for given design space -May not be entire design space due to parsing	-Extremely time consuming -Iterate through all possibilities	-Small scale -Topology, component, control, switching frequency, thermal, etc -Good method to observe impact of one variable on system
Genetic Algorithms	-Yes, in all cases	-Not guaranteed optimal solution	-Based on initial parameters given	-Topology, device, control, thermal, etc -Small to medium scale
Differential Evolution	-No, numerical based method	-Faster convergence than GA or PSO (for continuous variables)	-Faster than GAs when close to final value	-Control / feedback networks
Particle Swarm	-No, numerical based method (must have variables on coordinate plane)	-Greater diversity of solution compared to GA due to particle movement -Local minimum	-Momentum of particles can lead to fast convergence	-Control / feedback networks
Neural Networks	-Yes, if trained properly	-Depends on training	-Time consuming to train -Fast response	-Real time control
Gradient Methods	-No, numerical based method -Must be differentiable (or form approximate)	-Quadratic convergence (when close to final value)	-Quadratic convergence (when close to final value)	-Switching frequency, control, passives -(Near) continuous variable problems
Combined Methods e.g. GA and gradient	-Depends on methods used -Initially optimize unordered variables and use continuous numerical methods to complete optimization	-Improved convergence over stochastic methods via numerical iteration	-Sacrifices speed of stochastic method for potential better solution due to additional iterations	-Broad multi-objective optimization -Discrete part selection

2.3.4 Gradient Methods

Gradient based methods calculate some form of the derivative towards the goal and attempt to use this gradient or slope to find an optimal solution. The obvious limitation of this method is falling into local minimum and maximum while trying to find an optimal solution. Therefore, many random conditions are needed to find a true optimum. Additionally, some functions are either difficult or time consuming to calculate the gradient, Jacobian, or Hessian. Therefore, techniques such as the quasi-Newton method that attempt to use a form of the secant method to approximate the gradient of a function are used.

Gradient methods have been used in a variety of applications and optimizations. Gradient based methods have been used to solve a boost PFC converter's EMI filter, inductor, switching frequency, and thermal resistance using the method of feasible directions and sequential quadratic programming in MATLAB linked to Visual DOC (a commercial optimization software) [81, 82]. The results showed an improvement over and hand designed method and took less than a second on an Intel Pentium III microprocessor. Others showed a DC-DC buck converter passive component optimization using CADES (Component Architecture for Design of Engineering System) software [83] and later used the same technique for SiC high frequency design [84]. ProDesign, formerly known as Eden, was further used in optimization of the duty cycle and passives of a 'chopper' feeding a DC machine using an in house simulator [85] and a flyback converter's passive components and switching frequency using hand calculations [86]. Furthermore, a controller and power stage passives were designed by sequential quadratic programming in MATLAB using hand calculations as constraints [87]. One of the last and most complete optimizations of this group initialized using the sequential quadratic programming in MATLAB. Then, understanding the discrete nature of some of their variables, used what appears to be a branch and bound method to reach discrete numbers for core size and inductance [88].

Gradient methods inherently have issues with discrete non-ordered variables. There are different approaches that can be used to approximate the gradient, Jacobean, or Hessian of a function; however, for variables that consider material properties, such as an inductor core, the approximation of a slope can be hard to formulate. As seen by the literature, most

optimizations were for passive components and control feedback networks. In order to find discrete parts, the branch and bound algorithm approach can be used. Many of the gradient based methods found in literature used MATLAB to perform their optimization.

2.3.5 Others

While most of the methods reviewed refer to using a single approach to solve an optimization problem, there are others that have combined methods to try and mitigate the disadvantages of certain techniques. For example, using the conjugate gradient method and DE utilized the fast convergence of gradient methods with an evolution method to avoid local extreme points [89].

Other optimization techniques that were found during the review included neural networks. Neural networks were concentrated on real-time control to mitigate harmonics [90, 91]. Also, spreadsheet design [92] and figure of merit (FOM) [93] methodologies were also presented. FOM attempts to find relationships between known devices to give insight to a designer the relationship between inversely proportional variables. The FOM can then be used to determine the best type of device for a given system. Issues with FOM are that often multiple variables must be set in order to provide insight. Knowing how changing parameters can impact the efficiency of a system is important, but in large systems when relationships are complex between device selection, large scope insights can be difficult to distinguish.

2.4 Discussion

There is a broad array of literature on the techniques and methods used to design power converters. From an overall perspective of the literature, it is apparent that Metaheuristic methods, specifically GAs, are more prevalent than traditional gradient or classical optimization methods as shown in [69]. The characteristics of the literature presented are shown in Table 2.1. It is clear that to deal with unordered variables there must be an index based optimization routine, thus the popularity of GAs was pronounced in the literature. Convergence and speed were issues for different reasons throughout the

literature reviewed. For some GAs, time was a factor due to the simulation time and not the optimization technique itself.

From the steady-state analysis review, techniques based on the Newton-Raphson method dominated commercially available products. Other analysis techniques utilizing non-derivative methods were also analyzed, and some methods were comparable for a few converters. Other key distinctions found in the review are discussed in the following paragraphs.

First, many design methodologies and optimization techniques limited their solution space due to the small sets of component values that were provided to the optimization routine. Additionally, the scope of the problem overall usually does not completely solve the entire topology and operation of the converter. The optimization examples found are used for the feedback network control, passive filtering networks, inductors and transformers, or switching devices of a power converter but never all together.

Second, to limit the possible solution space assumptions are made by the design before the optimization is performed. There are probable boundaries that can be set for variables, e.g., switching at a frequency of 1 Hz is not going to provide an optimal solution for most topologies. However, some researchers assume a topology or switching frequency when setting up an optimization problem. The assumption is typically due to a need to limit the scope of the problem to something that takes less time to compute, especially when considering brute force methods. However, the assumption implies that the researcher knows other known topologies and frequency all fall short of the implemented selection, which is mathematically unknown and unproven.

Third, the interaction between steady-state analysis and optimization of the power converter appears to be absent in many of the optimization methods found. This is a direct result of not attempting to optimize the entire converter. Mostly analytical equations that accurately describe a certain aspect of a power converter were used in optimization problems. Due to the limited scope of the framework, this technique typically was adequate for the design process chosen in literature.

Fourth, there is a general lack of publications on the process of optimization of a power converter. The publications that are out there mainly stress their results and efficiency as

a good metric to judge their work. It is understood that in an applications driven field like power electronics, results such as efficiency and power density drive the progression of research. However, many research publications only cited the methods that they modeled the converters power loss or other metrics and did not provide details on the optimization method itself. Most provided a brief background on the optimization method or cited references on the original formulation of the method, but they meander around details such as number of iterations or time taken to compute that would be useful to compare one process against another.

Finally, from the steady-state analysis review it is apparent that the process and use of state-space representation of a circuit is preferred for switching converters. Steady-state analysis, however, was largely confined to Newton-Raphson methods with only initial conditions varied in order to find steady-state. One possibility that was briefly touched on is altering other variables in the converter to achieve steady-state. Not only altering frequency but also component values could be a possibility with the correct constraints.

Chapter 3

Discrete Time State-Space Modeling

Discrete time state-space modeling [94, 95, 96, 97] has long been considered a suitable modeling framework for generalized analysis of switching circuits in both steady-state and dynamic conditions. The framework natively accommodates periodically switched circuits, given each switching configuration in the circuit can be represented as a linear equivalent circuit. Under this assumption, the framework builds upon the state-space representation of the linear equivalent circuit during each switching interval

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_i \boldsymbol{x}(t) + \boldsymbol{B}_i \boldsymbol{u}(t) \tag{3.1}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}_i \boldsymbol{x}(t) + \boldsymbol{D}_i \boldsymbol{u}(t) \tag{3.2}$$

where the state vector, $\boldsymbol{x}$, contains the n_s converter states and $\boldsymbol{u}$ contains the n_i independent inputs to the system. The matrices $\boldsymbol{A}_i \in \mathbb{R}^{n_s \times n_s}$ and $\boldsymbol{B}_i \in \mathbb{R}^{n_s \times n_i}$ describe the circuit topology. The index $i \in [1...m]$ iterates over different linear equivalent circuits that model the topological changes due to switching.

3.1 Derivation

For any single time interval i, over which the linear equivalent circuit remains a valid approximation to the circuit behavior and approximating all independent inputs as constant during the interval, the solution to (3.1) is

$$\boldsymbol{x}(t) = \mathrm{e}^{\boldsymbol{A}_i t}\boldsymbol{x}(0) + \int_0^t \mathrm{e}^{\boldsymbol{A}_i(t-\tau)}\boldsymbol{B}_i\boldsymbol{u}(\tau)d\tau. \tag{3.3}$$

If $\boldsymbol{u}(t)$ is well-approximated as constant over the entirety of a single switching interval, (3.3) becomes

$$\boldsymbol{x}(t) = \mathrm{e}^{\boldsymbol{A}_i t}\boldsymbol{x}(0) + \boldsymbol{A}_i^{-1}[\mathrm{e}^{\boldsymbol{A}_i t} - \mathbf{I}]\boldsymbol{B}_i\boldsymbol{u}_i. \tag{3.4}$$

By repetitively applying (3.4) over the m linear equivalent switching subcircuits in one period of steady-state operation, the final states at the end of a switching period are

$$\boldsymbol{x}(T_s) = \left(\prod_{i=m}^{1} \mathrm{e}^{\boldsymbol{A}_i t_i}\right)\boldsymbol{x}(0) \sum_{i=1}^{m}\left[\left(\prod_{k=m}^{i+1} \mathrm{e}^{\boldsymbol{A}_k t_k}\right)\boldsymbol{A}_i^{-1}\left(\mathrm{e}^{\boldsymbol{A}_i t_i} - \mathbf{I}\right)\boldsymbol{B}_i\boldsymbol{u}_i\right]. \tag{3.5}$$

Setting $\boldsymbol{x}(T_s) = \boldsymbol{x}(0) = \boldsymbol{X}_{ss}$, the periodic steady-state solution to an arbitrary converter is

$$\boldsymbol{X}_{ss} = \left[\mathbf{I} - \prod_{i=m}^{1} \mathrm{e}^{\boldsymbol{A}_i t_i}\right]^{-1} \sum_{i=1}^{m}\left[\left(\prod_{k=m}^{i+1} \mathrm{e}^{\boldsymbol{A}_k t_k}\right)\boldsymbol{A}_i^{-1}[\mathrm{e}^{\boldsymbol{A}_i t_i} - \mathbf{I}]\boldsymbol{B}_i\boldsymbol{u}_i\right], \tag{3.6}$$

where $\boldsymbol{X}_{ss}$ describes the instantaneous states at the beginning of the period. The only approximation made in the derivation of (3.6) was that $\boldsymbol{u}(t) \approx \boldsymbol{u}_i$ within one switching interval. This approximation is valid for the vast majority of power electronics applications or can be enforced by expanding the scope of the modeled system.

Once the steady-state solution is solved, the value of the states $\boldsymbol{x}(t)$ at any instant within the period can be generated using (3.4), and the value of any other signal can be obtained using the result and (3.2), along with appropriate selection of $\boldsymbol{C}_i$ and $\boldsymbol{D}_i$. Coupled with methods to automatically parse a circuit description into state matrices [98], this approach can rapidly evaluate the steady-state electrical performance of an arbitrary switching circuit without restricting classification or requiring ideal assumptions.

3.2 Limitations

Though relatively approximation-free, there are two key limitations to the application of (3.6) for generalized switching circuit analysis:

1. For direct computation, $\boldsymbol{A}_i$ must be invertible, and
2. Each switching interval must be well-approximated as a linear equivalent circuit over a known duration t_i.

Solutions and implications of these limitations are discussed in the following sections.

3.2.1 Singular $\boldsymbol{A}_i$

The first limitation is solved using the augmented state-vector method of [95]. Although (3.4) contains $\boldsymbol{A}_i^{-1}$, the power series of $(\mathrm{e}^{\boldsymbol{A}_i t} - \mathbf{I})$ eliminates it [99], though neither evaluation of the power series nor direct evaluation is computationally efficient [2]. Instead, [100] shows that for a square matrix

$$\hat{\boldsymbol{A}}_i = \left[\begin{array}{c|c} \boldsymbol{A}_i & \boldsymbol{B}_i \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right], \tag{3.7}$$

the matrix exponential

$$\mathrm{e}^{\hat{\boldsymbol{A}}_i t} = \left[\begin{array}{c|c} \mathrm{e}^{\boldsymbol{A}_i t} & \displaystyle\int_0^t \mathrm{e}^{\boldsymbol{A}_i (t-\tau)} \boldsymbol{B}_i d\tau \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right] \tag{3.8}$$

inherently computes both terms in (3.3). (3.7) presents an efficient method to calculate the necessary terms in the event of a singular matrix $\boldsymbol{A}_i^{-1}$ because the matrix exponential is well-defined for any square matrix and because modern numerical computation tools such as MATLAB can rapidly solve numerical matrix exponentials. Furthermore, the augmented state-vector method [95, 97] can solve the limitation. The augmented state-vector method states that

$$\frac{d}{dt}\hat{\boldsymbol{x}}(t) = \hat{\boldsymbol{A}}_i \hat{\boldsymbol{x}}(t), \tag{3.9}$$

has solution

$$\hat{\boldsymbol{x}}(t) = \left[\begin{array}{c} \boldsymbol{x}(t) \\ \hline \boldsymbol{u}_i \end{array}\right] = \mathrm{e}^{\hat{\boldsymbol{A}}_i t_i}\hat{\boldsymbol{x}}(0). \tag{3.10}$$

Functionally, this method alters the system description so the constant input in each switching interval are instead considered states

$$\frac{d}{dt}\hat{\boldsymbol{x}}(t) = \left[\begin{array}{c|c} \boldsymbol{A}_i & \boldsymbol{B}_i \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right]\left[\begin{array}{c} \boldsymbol{x}(t) \\ \hline \boldsymbol{u}_i \end{array}\right] = \left[\begin{array}{c} \boldsymbol{A}_i\boldsymbol{x}(t) + \boldsymbol{B}_i\boldsymbol{u}_i \\ \hline \mathbf{0} \end{array}\right] = \left[\begin{array}{c} \dot{\boldsymbol{x}}(t) \\ \hline \mathbf{0} \end{array}\right]. \tag{3.11}$$

The new state-space matrix $\hat{\boldsymbol{A}}$ enforces that the time-derivative of the states corresponding to $\boldsymbol{u}_i$ is zero.

Equation (3.10) can be applied in the same manner as in (3.6) to solve the steady-state solution directly using augmented matrices, and finding the solution to [95]

$$\hat{\boldsymbol{X}}_{ss}\left(\mathbf{I} - \prod_{i=m}^{1}\mathrm{e}^{\hat{\boldsymbol{A}}_i t_i}\right) = 0, \tag{3.12}$$

or (3.8) can be used to compute solely $\int_0^t \mathrm{e}^{\boldsymbol{A}_i(t-\tau)}\boldsymbol{B}_i$, which is useful in cases where many evaluations of the same system with varying inputs $\boldsymbol{u}_i$ are needed.

Alternative Augmented State-Space

An alternative to the augmented matrix shown in (3.7) is to include the input vector $\boldsymbol{u}_i$ with $\boldsymbol{B}_i$ since neither will vary within an individual switching interval of the converter

$$\hat{\boldsymbol{A}}_i = \left[\begin{array}{c|c} \boldsymbol{A}_i & \boldsymbol{B}_i\boldsymbol{u}_i \\ \hline 0 & 0 \end{array}\right]. \tag{3.13}$$

The alternative $\hat{\boldsymbol{A}}_i$ approach changes the solution of (3.10) to be

$$\hat{\boldsymbol{x}}(t) = \left[\begin{array}{c} \boldsymbol{x}(t) \\ \hline 1 \end{array}\right] = \mathrm{e}^{\hat{\boldsymbol{A}}_i t_i}\hat{\boldsymbol{x}}(0). \tag{3.14}$$

With the alternative augmented state-space, the solution to (3.12) is to find the basis of the null space

$$\text{null}\left(\mathbf{I} - \prod_{i=m}^{1} \mathrm{e}^{\hat{\boldsymbol{A}}_i t_i}\right) \tag{3.15}$$

where the solution to $\hat{\boldsymbol{X}}_{ss}$ will be the scaling factor which ensures the final row is equal to one as shown by 3.14.

State-Space Averaging

A similar expansion of the state matrices, can be used to find the average value of each state, as detailed in [95] after solving for the steady-state solution. The average values are found by setting the derivatives of the new states equal to the instantaneous value of the original states. By altering the augmented state matrix

$$\check{\boldsymbol{A}}_i = \left[\begin{array}{c|c|c} \boldsymbol{A}_i & \boldsymbol{B}_i & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{I} & \mathbf{0} & \mathbf{0} \end{array}\right], \tag{3.16}$$

the augmented state-vector method states that (3.10) has solution

$$\check{\boldsymbol{x}}(t) = \left[\begin{array}{c} \boldsymbol{x}(t) \\ \hline \boldsymbol{u}_i \\ \hline \displaystyle\int_0^t x(\tau)d\tau \end{array}\right] = \mathrm{e}^{\check{\boldsymbol{A}}_i t_i}\check{\boldsymbol{x}}(0). \tag{3.17}$$

By setting the derivatives of the new states equal to the instantaneous value of the original states, the integral of the states is calculated. Iterating through all time intervals and multiplying by the frequency of the converter gives the average of all the state values

$$\overline{\boldsymbol{x}} = f_s \int_0^{T_s} x(\tau)d\tau. \tag{3.18}$$

3.2.2 Linear Circuit Requirement

Because this modeling approach is built upon the state-space description of (3.1), it is valid only when the system can be modeled within any arbitrarily small interval i, over which the circuit is adequately modeled as a linear equivalent. Though this may require a substantial increase in the number of switching intervals, m, there is no fundamental limit to the capability of the model to adapt to arbitrary nonlinear transfer characteristics of circuit elements or their loss models. The method is only limited by the numerical tolerances of the tools used to solve the state-space. Therefore, the tolerances of the method are similar to current commercial modified nodal analysis [20] and state-space time-stepping [33] simulators which often require iteration and variable time-stepping to ensure that linearized models remain sufficiently accurate in numerical approximations of nonlinear circuit elements.

The incorporation of nonlinear circuit behaviors will increase the modeling complexity and, therefore, analysis time. In some cases, these non-linarities can still be included by broadening the scope of the converter being modeled, such as replacing an AC source with an equivalent DC-AC converter. However, nonlinear loss mechanisms such as core loss or switching loss other than from C_{oss} can be approximated from waveforms retroactively using empirical data after the steady-state is reached to save simulation time.

This approach employs an approximation of high efficiency in the design stage, i.e. it assumes that all nonlinear loss mechanisms are sufficiently small relative to the output power such that they have minimal impact on the converter waveforms. One primary exception to this assumption is the nonlinear state-controlled switching of diodes, including transistor body diodes. Even when modeled as a two-state switch, the switching times of a diode are not externally controlled and can substantially impact converter waveforms even in high efficiency implementations. As such, the following section examines approaches to incorporate this mechanism into the solution of (3.4) through iterative means, while attempting to maintain the benefits of solving steady-state operation directly.

3.3 State-Independent and State-Dependent Switching

Power converters are assumed to be well-modeled as piecewise linear to employ (3.6). Each switching action, either by a transistor or diode, alters the linear-equivalent model of the converter by changing the conduction resistance or forward voltage lumped elements in the linear-equivalent model. The time intervals and their associated linear-equivalent circuits must be known before solving (3.6).

State-independent or active switching actions caused by a controller driving a transistor are set independently by an external signal. Thus, the timing of each of these switching actions are known a priori for a given converter design and steady-state operating point. However, state-dependent or passive switching, such as a diode conducting due to its forward threshold voltage being reached, occurs at a time dictated by the internal waveforms of the converter. The presence of state-dependent devices results in a nonlinear state-space description within a single switching interval,

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_i(\boldsymbol{x})\boldsymbol{x}(t) + \boldsymbol{B}_i(\boldsymbol{x})\boldsymbol{u}(t). \tag{3.19}$$

Alternatively, if the state-dependence can be modeled as piecewise linear, the system can continue to be modeled as linear as in (3.1). In this case, matrices within an interval where a state-dependent switching action occurs will be split into two (or more) subintervals, e.g. for a diode with different forward biased (on) and reverse biased (off) equivalent circuits,

$$\boldsymbol{A}_i = \begin{cases} \boldsymbol{A}_{i,D(on)}, & \text{if } -V_d \geq V_{FW} \\ \boldsymbol{A}_{i,D(off)}, & \text{otherwise} \end{cases} \tag{3.20}$$

and similar for $\boldsymbol{B}_i$. The time-duration of each individual subinterval impacted by state-dependent switching is not known without additional analysis. Because (3.6) always solves for the steady-state of the converter, other states throughout the converter are altered whenever the timing of these switching actions is altered. To maintain the merits of the

direct steady-state solution (3.6), the time-duration of each subinterval must be solved such that (3.20) remains valid at every instant in the steady-state solution waveforms.

An example of passive switching due to the body diodes in the synchronous buck converter of Fig. 3.1 is shown in Fig. 3.2. The piecewise linear model of each MOSFET is implemented as a variable voltage source and resistance,

$$\boldsymbol{V}_{F,M_i} = \begin{cases} V_{FW}, & \text{if } V_{ds,i} \leq -V_{FW} \\ 0, & \text{otherwise} \end{cases} \tag{3.21}$$

$$\boldsymbol{R}_{on-off,M_i} = \begin{cases} R_{ds(on),i}, & \text{if } M_i \text{ on} \\ R_{d,i}, & \text{if } V_{ds,i} \leq -V_{FW} \\ \infty, & \text{otherwise} \end{cases}. \tag{3.22}$$

The diode conduction behavior is unknown prior to solving the steady-state of the converter via (3.6). An initial guess for the switching interval time-durations supposes that the body diodes do not turn on during the deadtime. The steady-state solution under this assumption, given by the red dashed line in Fig. 3.2, contains a violation of (3.20). Namely, the switched-node voltage v_{sw} resonates below $-V_{FW}$, where the diode should turn on. Only observing the discrete time samples, this error can be seen because the voltage v_{sw} at the end of the period is well below the forward voltage. In this case, the discrete time samples are modeled as the state values at the end of each time interval. These can be reconstructed with minimal computation burden by applying recursive application of (3.4),

$$\boldsymbol{X}_{i+1} = \mathrm{e}^{\boldsymbol{A}_i t_i}\boldsymbol{X}_i + \boldsymbol{A}_i^{-1}[\mathrm{e}^{\boldsymbol{A}_i t_i} - \mathbf{I}]\boldsymbol{B}_i\boldsymbol{u}. \tag{3.23}$$

To correct this violation, a new time interval must be introduced to the converter where the body diode of M_2 conducts. This splits the original deadtime interval into two separate subintervals with their own linear-equivalent circuit as shown in Table 3.1. Assuming the correct time-duration of each subinterval is found, this change results in a valid steady-state solution as shown by the solid blue line in Fig. 3.2. Note that, by introducing or changing

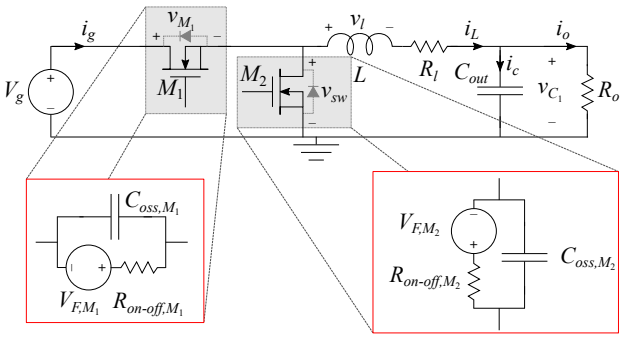


Figure 3.1: Synchronous buck converter and equivalent transistor representation used in Fig. 1.2 and Fig. 3.2.

the duration of switching intervals in the converter, the initial states at the beginning of the deadtime have changed due to the new steady-state solution; the correct $t_{BD(on)}$ is not equal to the amount of time that $v_{sw} < -V_{FW}$ in the initial solution.

Both the initial and final solutions in Fig. 3.2 are in steady-state, i.e. $\boldsymbol{x}(0) = \boldsymbol{x}(T_s)$, as enforced by the solution of (3.6). However, only the final solution exhibits a steady-state waveform with no violations of (3.21)-(3.22).

3.3.1 Steady-State Solution with State-Dependent Switching

To model the presence of state-dependent switching actions, the following notation is used. The index i consistently refers to the switching interval order as determined by only the state-independent (i.e. active, or externally-controlled) switching actions, $i \in \{1, 2, ..., n\}$ for n active switching intervals. The vector $\boldsymbol{p} \in \mathbb{N}^n$ contains the number of distinct subintervals in each active switching interval due to state-dependent switching. The index j refers to switching subintervals within the i^{th} interval that occur due to the presence of state-dependent switching; in interval i, $j \in \{1, ..., p_i\}$. State matrices $\mathbf{A}_{(i,j)}$ and $\mathbf{B}_{(i,j)}$ model the linear-equivalent circuit within the j^{th} passive subinterval of the i^{th} active interval. In all cases $\mathbf{A}_{(i,1)} = \mathbf{A}_i$, as defined by (3.1).

Subinterval time-durations are indexed accordingly, such that

$$t_i = \sum_{j=1}^{p_i} t_{(i,j)}. \tag{3.24}$$

To find the valid steady-state of a power converter using discrete time state-space modeling, each passive switching action must be found within each active switching action. The active switching time intervals are modeled as

$$\boldsymbol{X}_{ss} = \left[\mathbf{I} - \prod_{i=n}^{i} \boldsymbol{\Phi}_i\right]^{-1} \sum_{i=1}^{n} \left[\boldsymbol{\Psi}_i\right] \tag{3.25}$$

where

$$\boldsymbol{\Phi}_i = \prod_{j=p_i}^{1} \mathrm{e}^{\boldsymbol{A}_{(i,j)} t_{(i,j)}} \tag{3.26}$$

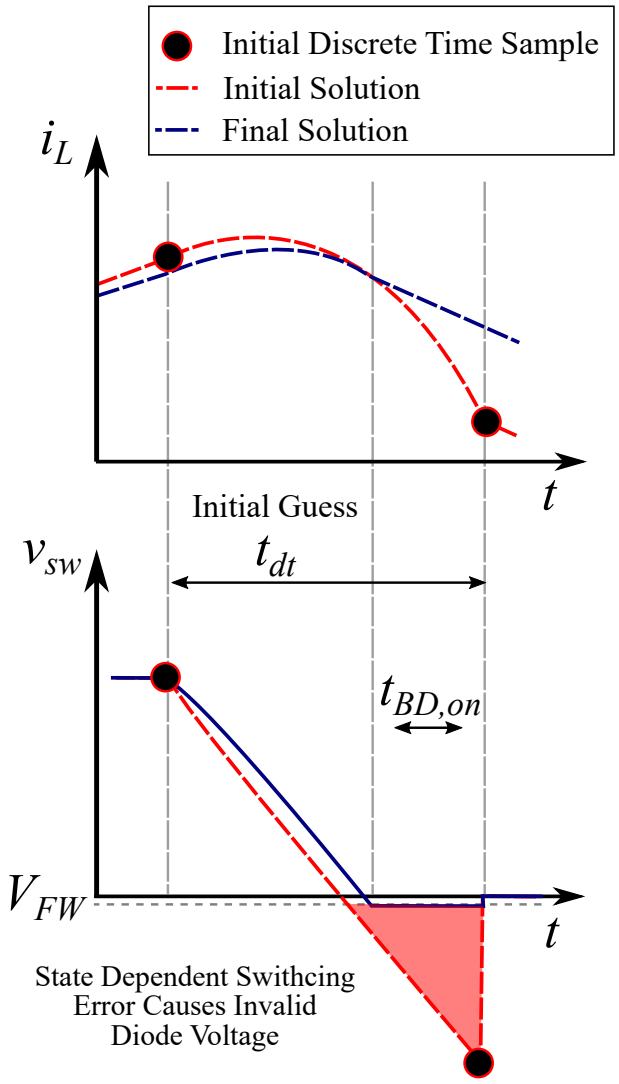


Figure 3.2: Waveform of a synchronous buck converter after the high side device turns off. The initial guessed deadtime length is too long between active switching actions.

Table 3.1: Buck Converter Deadtime Solution

Initial Solution		
Time Interval	t_{dt}	
Circuit Model	$\boldsymbol{A}_{i,BD(off)}$	
Conducting Devices	—	
Final Solution		
Time Interval	t_{dt}	
	$t_{BD(off)}$	$t_{BD(on)}$
Circuit Model	$\boldsymbol{A}_{i,BD(off)}$	$\boldsymbol{A}_{i,BD(on)}$
Conducting Devices	—	$M_{2,BD}$

and

$$\boldsymbol{\Psi}_i = \sum_{j=1}^{p_i} \left[\left(\prod_{k=n}^{i+1} \boldsymbol{\Phi}_k \right) \left(\prod_{k=p_i}^{j+1} \mathrm{e}^{\boldsymbol{A}_{(i,k)} t_{(i,k)}} \right) \times \boldsymbol{A}_{(i,j)}{}^{-1} \left[\mathrm{e}^{\boldsymbol{A}_{(i,j)} t_{(i,j)}} - \mathbf{I} \right] \boldsymbol{B}_{(i,j)} \boldsymbol{u} \right]. \tag{3.27}$$

For each active switching time interval i, there is a set of p_i passive switching time intervals which solves the steady-state of the converter such that all state-dependent switching constraints (e.g. (3.20)) are satisfied. In general, this sequence is solved through numerical iteration.

For the specific case of state-dependence arising from diode conduction, this iteration is employed to ensure the inequality

$$\boldsymbol{E}_{(i,j)} \left(\boldsymbol{y}_D + \boldsymbol{V}_F \right) < 0 \tag{3.28}$$

where $\boldsymbol{E}_{(i,j)}$ is a vector of ± 1 values to indicate whether a diode is conducting or blocking, $\boldsymbol{V}_F$ is a vector of each diode's forward voltage, and $\boldsymbol{y}_D$ is a vector of instantaneous diode voltages during the entirety of the steady-state waveforms,

$$\boldsymbol{y}_D(t) = \boldsymbol{C}_{\boldsymbol{D}(i,j)} \boldsymbol{x}(t) + \boldsymbol{D}_{\boldsymbol{D}(i,j)} \boldsymbol{u}(t) \tag{3.29}$$

for appropriately-derived $\boldsymbol{C}_{\boldsymbol{D}}$ and $\boldsymbol{D}_{\boldsymbol{D}}$.

The goal of the framework is to solve a steady-state solution consisting of interval durations $t_{i,j}$ and the associated conduction ordering of all switching devices constrained by (3.24) and solve the steady-state initial condition for each iteration from (3.25) while ensuring that (3.28) holds for any time $t \in [0, T_s]$ within the switching period.

3.3.2 Example Convergence of Synchronous Buck

To examine the constraints on an iterative solution accounting for passive switching, Fig. 3.3 shows an example of the solution space for the synchronous buck converter of Fig. 3.1. The converter operates in CRM with ZVS and has parameters as listed in Table 3.2.

Fig. 3.3 shows the error in state-dependent conditions at the discrete time sampling points, as the two deadtimes of the buck converter are independently varied from 0.1% to 3.3% of the switching period T_s. In all cases, both transistors are treated as switches as modeled in Fig. 3.1; no attempt is made in this example to account for body diode conduction in order to examine the solution space. The error shown in Fig. 3.3 is

$$\text{Err} = \begin{cases} v_{sw} - (V_g + V_{FW}), & \text{spoif } v_{sw} > (V_g + V_{FW}) \\ v_{sw} + V_{FW}, & \text{if } v_{sw} < -V_{FW} \\ 0, & \text{otherwise} \end{cases} \tag{3.30}$$

and models the magnitude of the spurious over-voltage or under-voltage at the switch node, which would be clamped by body diode conduction in the converter.

When only the discrete sampling points are examined, there are multiple zero-error regions, where only the points near the minimum values of t_{d1} and t_{d2} are valid. Fig. 3.4 shows the waveforms during each deadtime interval of two operating points from Fig. 3.3. Operating point 1 is in the valid zero-error region, with $t_{d1}/Ts = 0.17\%$ and $t_{d2}/Ts = 0.23\%$. Operating point 2 is in an invalid zero-error region, with $t_{d1}/Ts = 1.8\%$ and $t_{d2}/Ts = 1.5\%$. Note that, examining only the discrete time points (green diamonds), both operating points have sampled v_{sw} within the boundaries of diode conduction at the beginning and end of the subinterval. However, when the full waveforms of both points are examined, only operating point 1 is valid at every time instant within the subinterval.

Fig. 3.5 shows the error function over every point in the waveform, rather than solely the discrete time samples. In this case, only valid waveforms show zero error; however, local minima still exist at the invalid operating points from Fig. 3.3.

Data in Fig. 3.5 is obtained by time-stepping through a single period of the steady-state solution from (3.6) with the prescribed deadtimes. Though errors outside of the discrete

Table 3.2: Example Synchronous Buck Parameters

L_1	500 nH	$C_{oss,M1}$	190 pF
C_1	100 µF	$C_{oss,M2}$	880 pF
R_l	1 mΩ	R_{ds}	2.5 mΩ
V_g	48 V	f_s	200 kHz
I_o	1 A	D	$\frac{1}{48}$

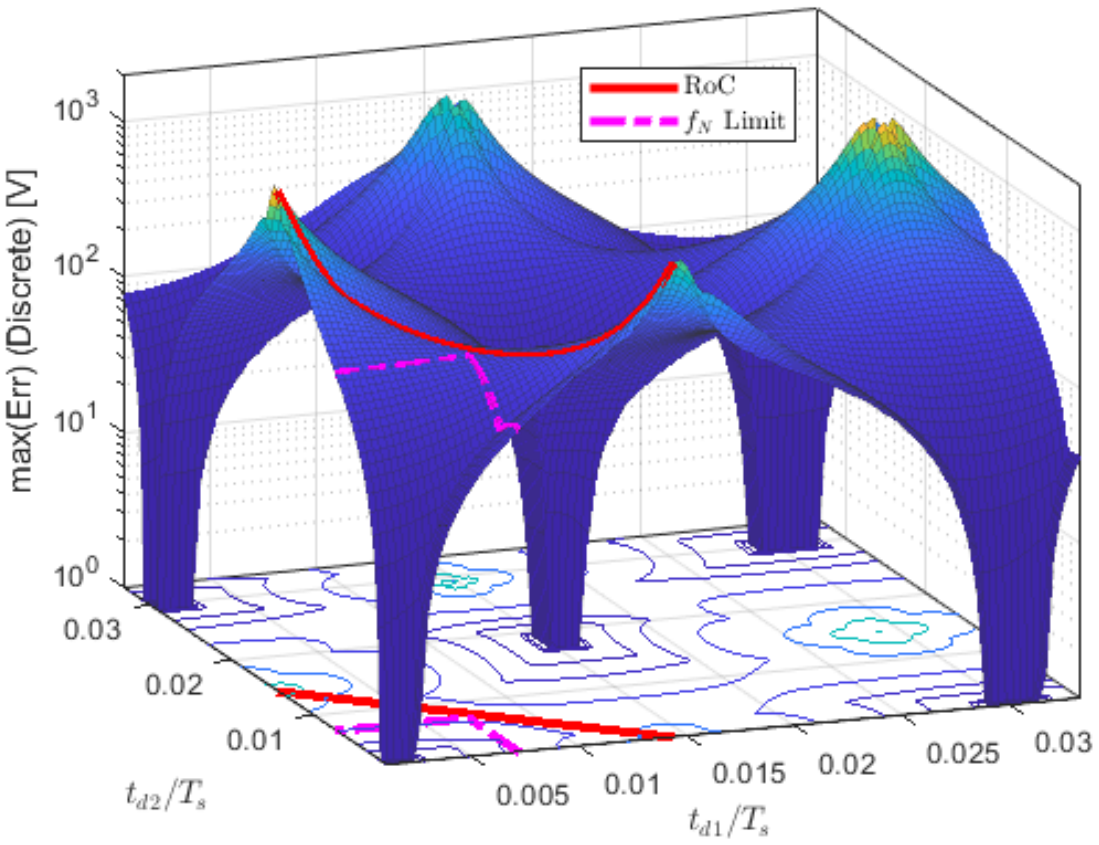


Figure 3.3: Passive switching error for the synchronous buck converter considering only the discrete sampling points.

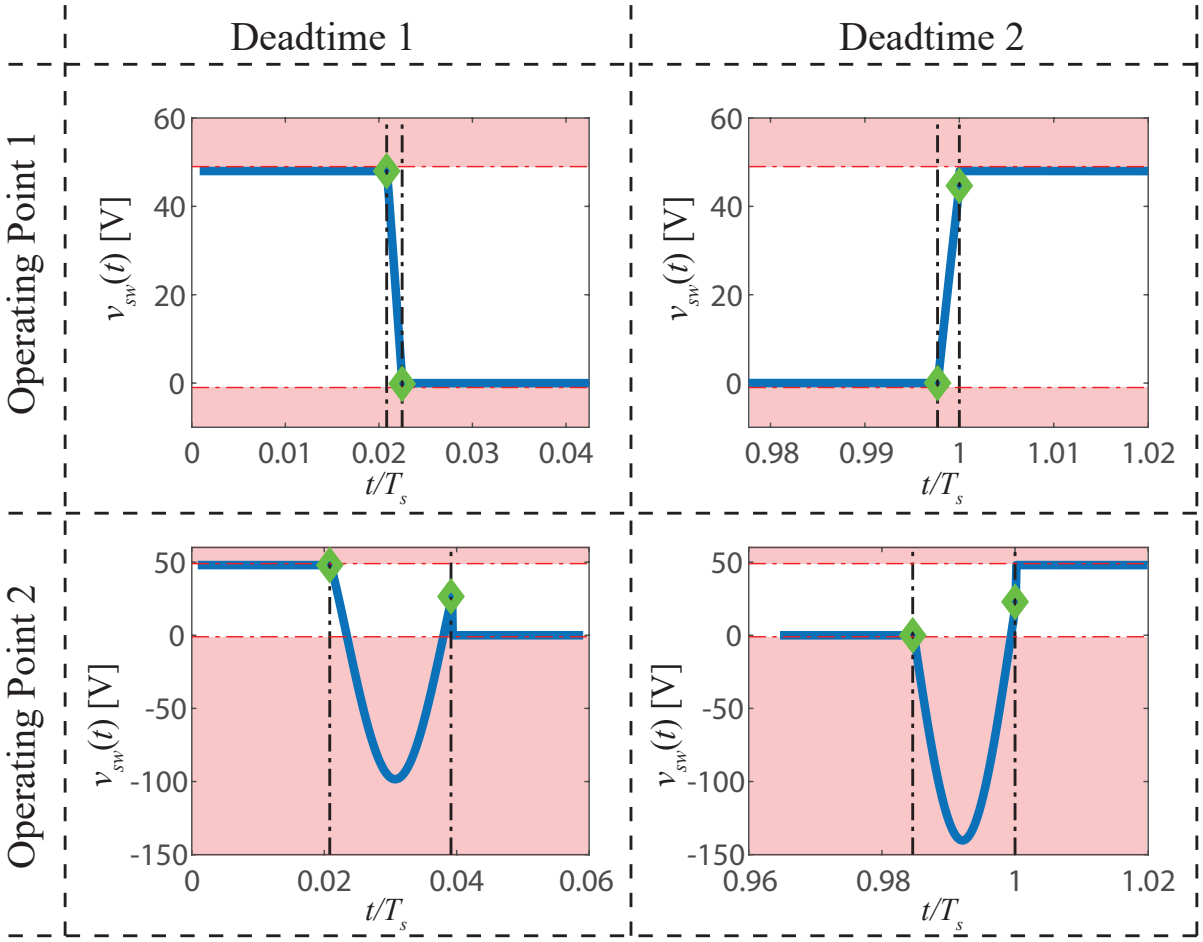


Figure 3.4: Example complete waveforms (solid blue lines) and discrete sampling points (green diamonds) for both deadtimes, and two operating points, of the synchronous buck converter.

sampling points can be identified, this process is computationally intensive and greatly increases the simulation time. Thus, the process should be avoided unless necessary.

The solid red line in both Fig. 3.3 and Fig. 3.5 indicates the region of convergence (RoC) for a theoretical gradient-based error minimization algorithm. As long as the initial guess for the converter deadtimes is within this region, a gradient-descent method will arrive at a valid operating point. The dashed magenta line in both figures represents a boundary based on the natural frequencies of the converter, as discussed in the following section. In this case, limiting both deadtimes according to the natural frequencies of the state matrix guarantees timing durations within the RoC.

The synchronous buck converter presents a relatively simple case for examining the solution space for a passive switching algorithm. Other topologies, which may exhibit multiple passive switching actions within a single interval or where error resulting from passive switching violations has a stronger impact on the complete steady-state waveform, exhibit a higher degree of non-convexity in the solution space.

The following section develops an algorithm to systematically check and address passive switching constraints to eventually converge to a valid steady-state solution.

3.4 Passive Switching Correction

Due to the nonlinearity of the problem, an iterative numerical approach is used to obtain the correct steady-state solution in the presence of state-dependent switching actions. In each iteration, the duration of passive switching intervals may be altered, or new switching states may be inserted or removed from the switching pattern. This section discusses the implementation of a method to determine the iterative changes to the switching pattern and subinterval timing such that the steady-state solution converges to a violation-free result in the presence of state-dependent switching actions.

For subintervals where the slope of the waveform exhibiting a violation is nearly constant, such as v_{sw} in Fig. 3.2, the identification of an invalid steady-state solution due to passive switching is easily solved by evaluating the discrete time points and their derivatives. The length of the body diode time interval in the synchronous buck converter example can be

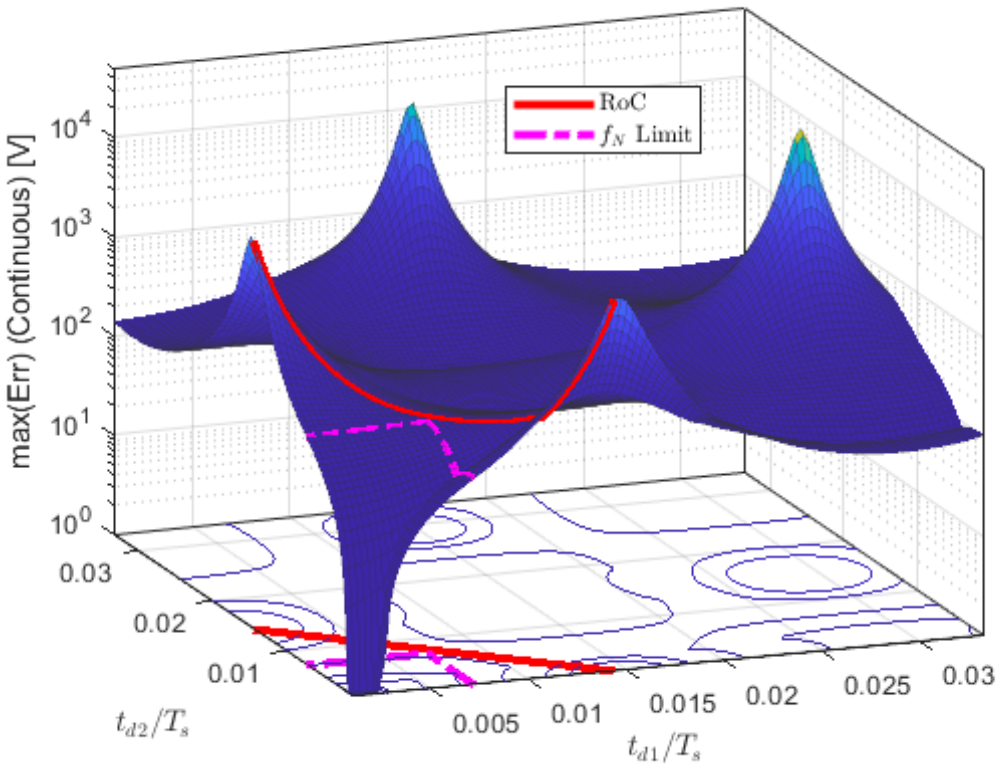


Figure 3.5: Passive switching error for the synchronous buck converter considering full continuous-time waveforms.

solved by taking the partial derivative of the state variables at the discrete switching points with respect to the time interval [101]. Nearly-constant slope subintervals occur when the duration of the subinterval is significantly shorter than the dominant dynamic frequencies of the circuit. When the inverse is true, highly resonant or multi-resonant dynamics are present, and the violations are more difficult to identify and correct.

Eigenvalues of $\boldsymbol{A}_i$ are used to observe the natural frequencies of each sub-interval. The natural frequencies of each state matrix, along with the discrete time sample derivatives, separate each time interval of a converter into three distinct categories, each having their own identification and feasible state-dependent switching correction method.

The first tier, the 'constant slope' category, is similar to the example in Fig. 3.2. The natural frequencies of the circuit are sufficiently small such that, within the subinterval, the derivatives of the state variables associated with the violation are nearly constant. The second tier, the 'single-resonant' category, arises when a single natural frequency has a period which is less than, or nearly equal to, the subinterval time-duration. The third tier, the 'multi-resonant' category, occurs when the eigenvalues indicate multiple natural frequencies are faster than the time interval.

The conditions of these tiers are outlined in Table 3.3, with t_b and t_e representing the beginning and ending times of the time interval, respectively, and an example of the single- and multi-resonant categories is shown for a non-synchronous buck converter with a small power loop inductance in Fig. 3.6. The loop inductance causes a single-resonance while the transistor is turned on and multi-resonance while the transistor is off during the deadtime interval.

When dynamics in the second or third tier occur, the simple slope-projection method of [101] is insufficient to dictate a correction for the ensuing iteration.

3.4.1 Algorithm

The proposed implementation of the eigenvalue and derivative checks for discrete time state-space modeling of power converters is shown in Fig. 3.7. After finding the steady-state of the active switching intervals of a converter, the eigenvalues are first checked and recorded for each subinterval.

Table 3.3: Categories of Frequency Dynamics in the Proposed Method

Tier	Condition
Constant slope	$\text{sgn}(\dot{x}(t_b)) = \text{sgn}(\dot{x}(t_e))$ & $\frac{\pi}{\text{Im}(\text{eig}(A))} > t_e - t_b$
Single-Frequency	One violation of: $\frac{\pi}{\text{Im}(\text{eig}(A))} > t_e - t_b$ $\|\|$ $\text{sgn}(\dot{x}(t_b)) \neq \text{sgn}(\dot{x}(t_e))$
Multi-Frequency	Multiple violations of: $\frac{\pi}{\text{Im}(\text{eig}(A))} > t_e - t_b$

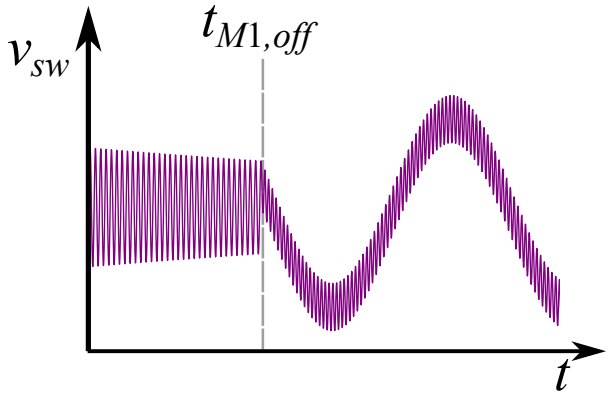


Figure 3.6: Non-synchronous buck converter switch node waveform with a small stray loop inductance added in series with transistor.

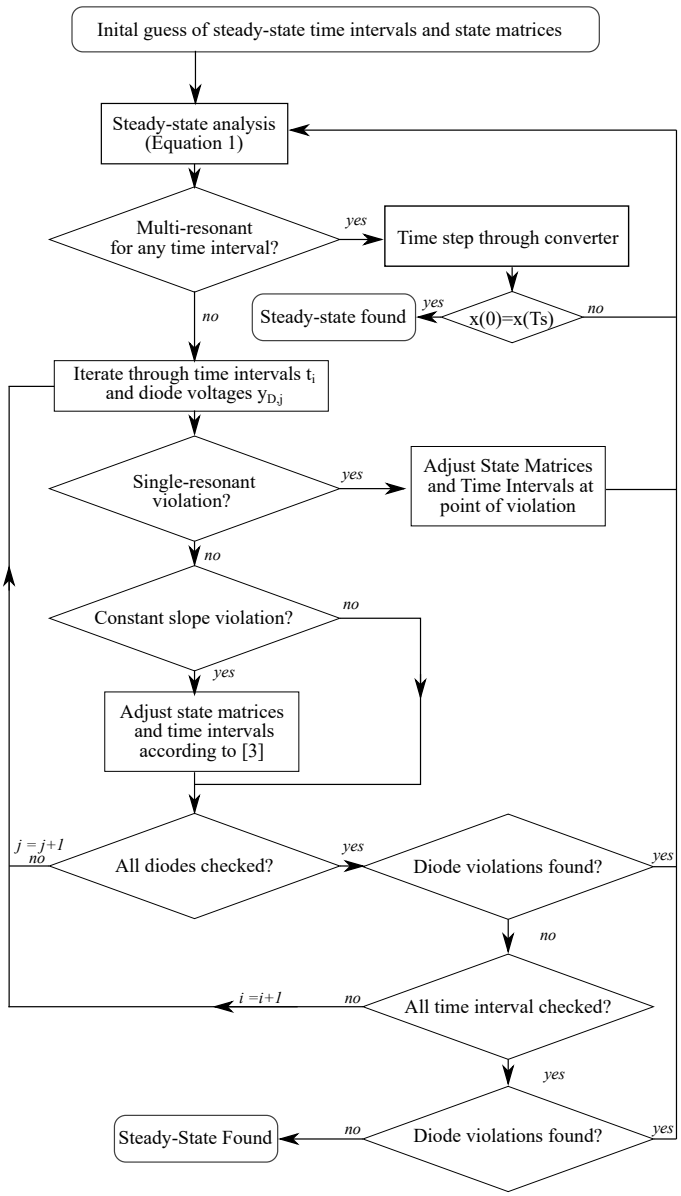


Figure 3.7: Flow diagram of how multi-resonant and single-resonant circuits are addressed in the proposed method.

Table 3.4: Steady-State Solve Time of Simulation Methods

	Convergence Time (sec)[1]				
Simulation	DAB	HDSC	Buck	Flyback	Buck-boost
LTspice	3.78	7.39	0.79	8.33	2.95
PLECS					
-Transient	7.82	3.20	2.20	9.16	4.34
-Steady-State	-	-	21.11	-	-
Proposed Method					
-Total	5.00	2.96	0.52	6.98	3.52
-Solver only	0.87	0.01	0.03	5.85	0.71
-Time-stepping iterations	0	0	0	7	0

[1] All simulations performed on a laptop computer with an Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz processor, 16 GB of RAM, and a 64 bit operating system.

Table 3.5: Steady-State Solve Converter Properties

	Amount				
Parameter	DAB	HDSC	Buck	Flyback	Buck-boost
State Variables	11	13	4	7	9
Time Intervals	8	4	4	2	12

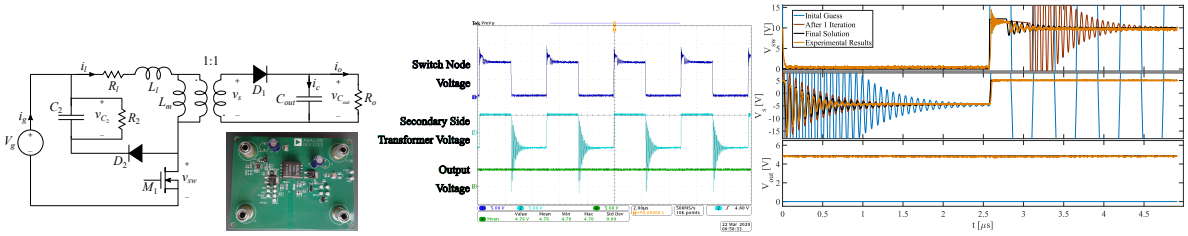
If there is a violation during a multi-resonant interval, the converter is time-stepped for one period to find the approximate location of any state-dependent switching actions. Because the time-stepping begins from an initial state $\boldsymbol{X_{ss}}$, which exhibited passive switching violations, the time-stepped period will, in general, not result in steady-state. Steady-state is then re-solved using (3.6), with the new switching pattern and time intervals found from time-stepping.

If no multi-resonance is detected, diode voltages at each discrete time interval are first examined for violations. If a single-resonance is detected, additional discrete time samples are solved for every quarter period of the fastest natural frequency of the converter. Using these additional discrete time points, a binary search is completed to quickly determine if there is a state-dependent switching violation within the time interval. Intervals where states have semi-constant slopes are identified and corrected using the method from [101].

There are certain cases in which the proposed algorithm still exhibits slow convergence to steady-state. For example, if an initial multi-frequency violation is detected, additional diode conducting time intervals may be placed throughout the steady-state waveform of the converter. If the steady-state solution is not sufficiently close to the final steady-state value, small adjustments in state-dependent switching times can greatly alter the behavior of the converter. Such conditions are likely to occur if multiple extended diode conduction times are required to achieve steady-state or resonant combinations throughout the period. Evaluating and eliminating each violation takes multiple iterations of the algorithm; thus, after 50 iterations, the converter is solved via time-stepping through one period. This resets the diode conduction time intervals at an initial condition closer to the steady-state value.

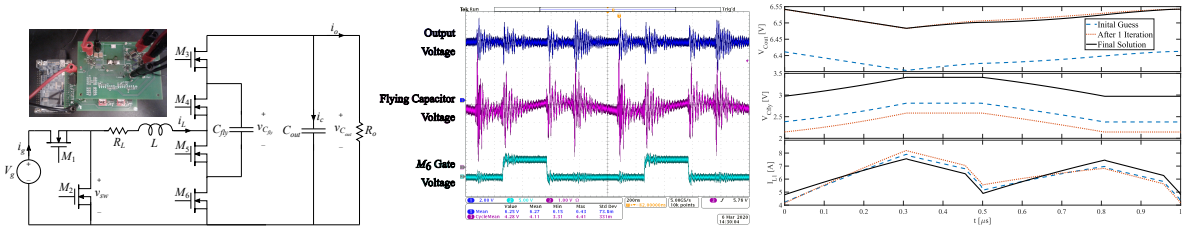
3.5 Simulation Verification

Several different types of converters are used to verify the proposed algorithm against other simulation tools in terms of real-time convergence speed. The DAB, HDSC, and buck converters are the same as detailed in [101]. The flyback and buck-boost are modeled after the experimental prototypes in Fig. 3.8 and Fig. 3.9, respectively. In each case, the transistors are modeled as shown in Fig. 3.1. Additionally, for LTspice and the PLECS transient method,



(a) **(b)** **(c)**

Figure 3.8: Flyback converter (EVAL-CN0342-EB1Z) schematic (a), experimental waveforms (b), modeled waveforms (c).



(a) **(b)** **(c)**

Figure 3.9: Buck-boost converter prototype and schematic (a), experimental waveforms (b), modeled waveforms (c).

waveform data is not saved until the steady-state of the converter is reached. Although this method requires the simulation time to reach steady-state be known before simulation, it provided a fairer comparison across all methods since only steady-state waveforms are being considered. The results of the simulation verification are shown in Table 3.4, with Table 3.5 displaying the number of state variables and time intervals for each converter. The proposed method does require some initialization time, as indicated by the large initial gap in error point in Fig. 1.2. This initialization consists of parsing a circuit netlist to obtain the $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ matrices of the topology. The solver time for the proposed method shows the time to reach steady-state after initialization. Circuit parsing is completed symbolically, such that the process would not be repeated in an optimization routine if there are no changes to the base topology.

The simulation results show the dual-active bridge converter and the hybrid Dickson switch capacitor converter take longer to initialize in the proposed method because of their high number of states. Only the flyback converter takes an extended period of time to solve due to the extended ringing caused by the leakage inductance of the transformer. Regardless, the solver time of the proposed method is well below that of any commercial tool in Table 3.4.

3.6 Experimental Verification

The flyback converter shown in Fig. 3.8 shows the progression of the proposed method through all three possible conditions. The converter is an evaluation board (EVAL-CN0342-EB1Z) with a 1:1 transformer and a RCD snubber network. Initially, only two switching intervals are known, since there is only one active switch in the converter, M_1. During the extended time when D_2 should be on, there is a multi-resonance that requires time-stepping to correct. The steady-state of the converter is then solved using the linear projection method. The simulation timing results are shown in Table 3.4.

The buck-boost converter shown in Fig. 3.9 is an example of the versatility of the proposed method for more than just diode violation correction, but also for other types of state-dependent switching. The converter uses EPC GaN devices with a 4.4 μF flying capacitor and 200 nH inductor. The initial guess of time intervals causes the converter to behave

non-ideally, since the inductor current is unbalanced across $\frac{T_s}{2}$ and the flying capacitor is less than $\frac{V_{out}}{2}$. Other simulators would require a control loop to account for this imbalance [102, 103]. However, in the proposed method, a constraint is placed on the inductor current, similar to the diode forward voltage constraint, to force $i_L = 4.9$ A at $\frac{T_s}{2}$ and T_s, emulating a valley-current programmed mode controller. The proposed method finds the final solution consistently in under 4 seconds.

3.7 Jacobian Method

While the methods shown in Fig. 3.7 are shown to converge to the steady-state solution, a potentially faster method can be used. Once the correct subintervals have been inserted, the subinterval time durations can be adjusted using a Jacobian matrix of all the states and time intervals in the converter instead of moving through and correcting each one at a time [104]. For this example, (3.28-3.29), can be generalized to the form

$$\boldsymbol{E}_i = \boldsymbol{C}_{\boldsymbol{bnd}i}\boldsymbol{X}_i + \boldsymbol{D}_{\boldsymbol{bnd}i}\boldsymbol{u} \leq \boldsymbol{0} \tag{3.31}$$

where $\boldsymbol{E}_i$ should be driven to less than zero for selected boundary constraints $\boldsymbol{C}_{\boldsymbol{bnd}i}$ and $\boldsymbol{D}_{\boldsymbol{bnd}i}$.

A Jacobian is calculated for all time intervals of the converter to determine each time interval's impact on the steady-state as well as each steady-state violation.

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial X_1(t_1)}{\partial t_1} & \frac{\partial X_1(t_2)}{\partial t_1} & \cdots & \frac{\partial X_1(t_n)}{\partial t_1} \\ \frac{\partial X_2(t_1)}{\partial t_1} & \frac{\partial X_2(t_2)}{\partial t_1} & \cdots & \frac{\partial X_2(t_n)}{\partial t_1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial X_m(t_1)}{\partial t_1} & \frac{\partial X_m(t_2)}{\partial t_1} & \cdots & \frac{\partial X_m(t_n)}{\partial t_1} \end{bmatrix}$$

where $\boldsymbol{J} \in \mathbb{R}^{n_s \times m \times m}$ for a system with n_s states and $m = m_i + m_j$ subintervals. Entry j_{ijk} is the response of the i^{th} state, at the j^{th} subinterval, to a perturbation in the duration of the k^{th} subinterval time.

$\boldsymbol{J}$ is evaluated numerically for small perturbations to each time interval. To construct the complete matrix, the steady-state solution (3.6) and discrete time reconstruction (3.23) need to be repeated m times. Because these equations are all closed-form, doing so requires relatively little computational burden.

In order to eliminate the errors present in any iteration of the steady-state solution, $\boldsymbol{J}_{err}$ is evaluated only for the states and times where error with state-dependent switching is present in the steady-state solution. This is combined with equations of the form of (3.24) for every subdivided interval from the controlled switching pattern to give

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{J}^{\pm}_{err,ij} \\ \\ \mathbf{1}_{\boldsymbol{T}} \end{bmatrix}, \tag{3.32}$$

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{D}^{\pm}_{bnd,ij}\boldsymbol{u} \\ \\ \boldsymbol{t_i} \end{bmatrix} \tag{3.33}$$

where $\mathbf{1}_{\boldsymbol{T}}$ is a binary matrix indicating which timing subintervals are part of each controlled interval in $\boldsymbol{t_i}$. The required change to each time subinterval $\Delta\boldsymbol{t_i}$, based on the linearized steady-state Jacobian, is then given by the solution to

$$\boldsymbol{\Phi}(\Delta\boldsymbol{t_i}) + \boldsymbol{\Gamma} = 0 \tag{3.34}$$

where the equation is an equality, rather than inequality, because only the errant signals are included and must be driven back to zero from violating (3.31).

In practice, $\Delta\boldsymbol{t_i}$ must then be checked to ensure no time intervals are driven below zero. If the solution attempts to do so, subintervals inserted in the prior section may need to be

removed, and the algorithm should iterate back and forth between the subinterval insertion and time adjustment steps.

3.8 Conclusions

This chapter introduces discrete time state-space modeling, a method to rapidly model a broad scale of power converters in steady-state. Instead of traditional methods where the converter is always in a valid steady-state, discrete time state-space modeling finds a steady-state and then must determine if the solution is valid for a physical circuit. This work showed that the proposed method is able to find the steady-state of a converter faster than traditional time-stepping circuit solvers. Several limitations with discrete time state-space modeling were overcome to reach this conclusion, the biggest hurdle being state-dependent switching. A comparison to other simulators was shown to validate the method's rapidness over traditional simulation methods, and experimental results were shown to validate the method's accuracy.

Chapter 4

Converter Optimization

Discrete time state-space modeling allows for rapid analysis of switch mode power supplies without the need for additional approximation or dedicated analysis of each topology. With an accelerated modeling method, brute force and metaheuristic methods execute faster; however, there are inherent techniques used in discrete time state-space modeling that can be applied to optimization. A general approach to this is shown in Fig. 4.1. Due to the speed of discrete time state-space modeling, it is feasible to run a multi-loop optimization without sacrificing breadth of analysis. In the inner loop, the solution is iterated to correctly account for all nonlinear state-dependencies. In the outer loop, the hardware design or modulation is altered to optimize the performance.

In order to fully optimize a set of converters, an array of switching and passive devices must be at the disposal of the optimization routine. To accomplish, this a database was set up where both tabular and graphical data is saved into an easily extracted class structure format. A few of the datasheets that can be output are shown in Fig. 4.2-4.4. More information on how to set up and extract data from the database is shown in Appendix B.

4.1 Optimized Component Selection

The rapid converter steady-state modeling enables new modeling approaches to broad-scale optimization of power converters. While the rapid modeling allows brute force methods to

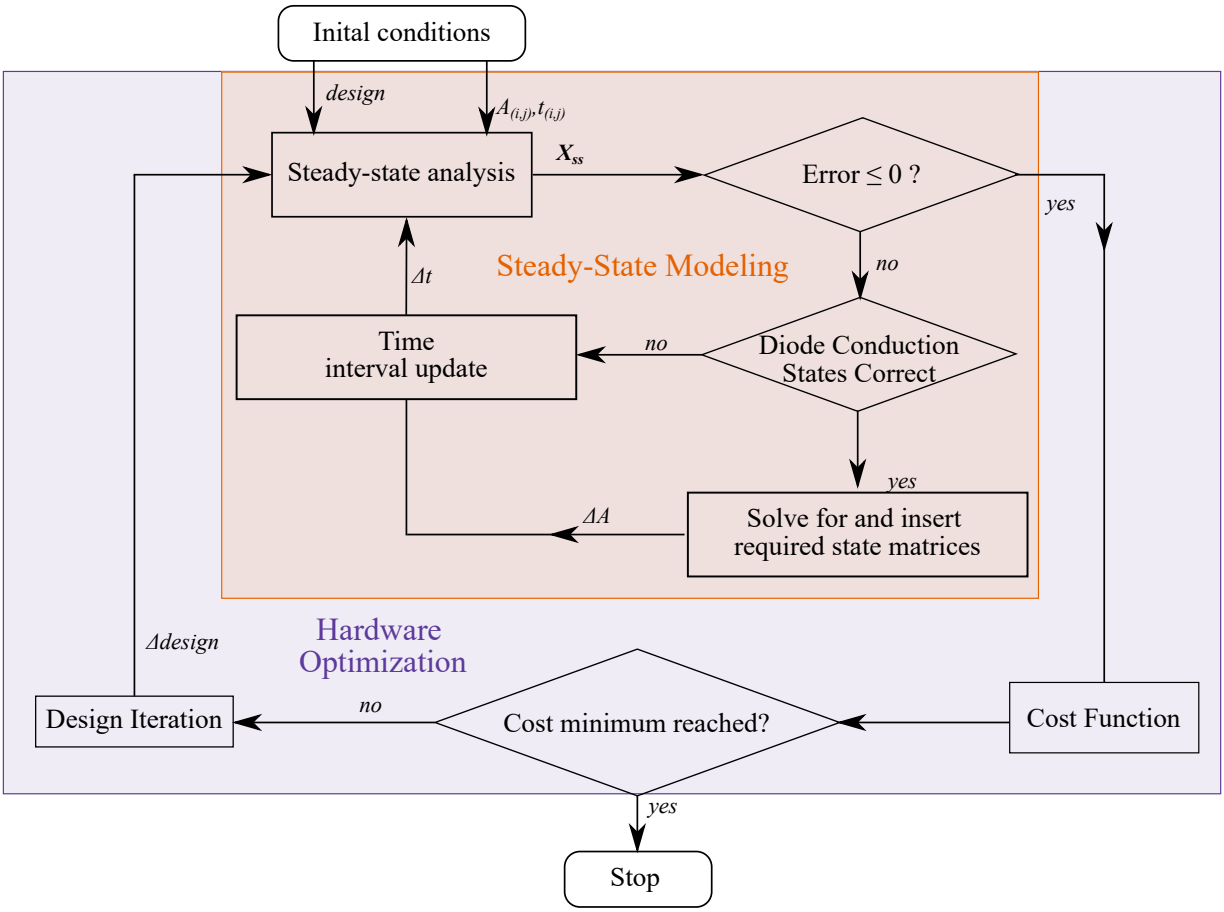


Figure 4.1: Flowchart of schematic-level design optimization. In an inner loop, the discrete time steady-state model is iterated to find a valid converter solution. In an outer loop, the design (including e.g. component selection, switching frequency) is iterated to optimize an objective function.

EPC2055

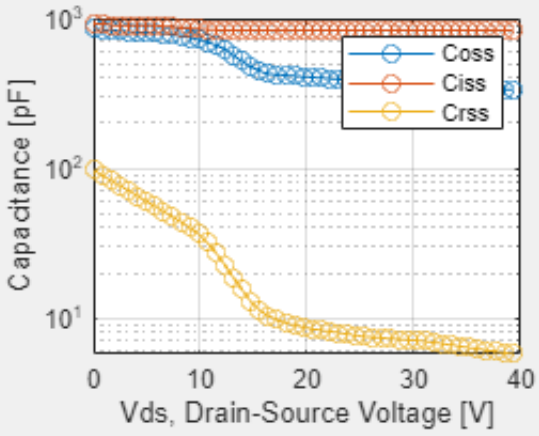


Nameplate Parameters

Parameter	Value	Unit
Vds - Drain-Source Voltage	40.0000	V
Rds(on) - Drain-Source On Resistance	3.6000	mOhm
Ids - Drain-Source Current	29.0000	A

Characteristics

Parameter	Test Conditions	Min	Typ	Max	Ur
Vds	VGS = 0 V, ID = 0.5 mA (Continuous)			40.0000	V
Ids	(Continuous) TA = 25°C)			29.0000	A
Idpulse	(Pulsed) 25°C Tpulse = 300µs)			161.0000	A



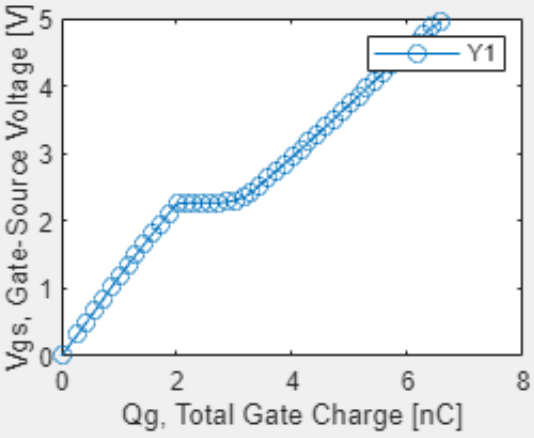


Figure 4.2: Datasheet of an EPC2055 GaN FET.

XGL3520-561ME

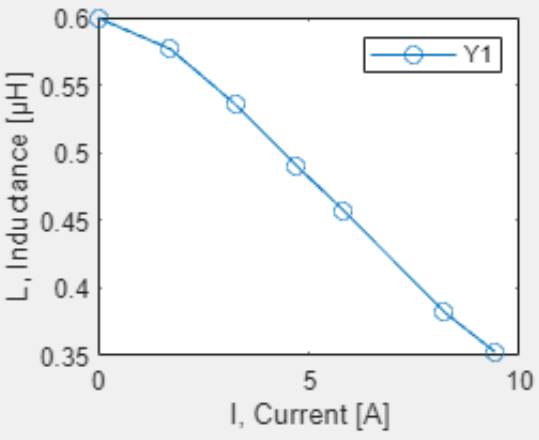


Nameplate Parameters

Parameter	Value	Unit
L - Inductance	0.5600	µH
Rdc - DC Resistance	9.1000	mOhm

Characteristics

Parameter	Test Conditions	Min	Typ	Max	Unit
L	1 MHz, 0.1 Vrms, 0 Adc		0.5600		µH
Rdc			7.9000	9.1000	mOhm
Srf			105.0000		MHz
Isat	10% drop in L		3.2000		A



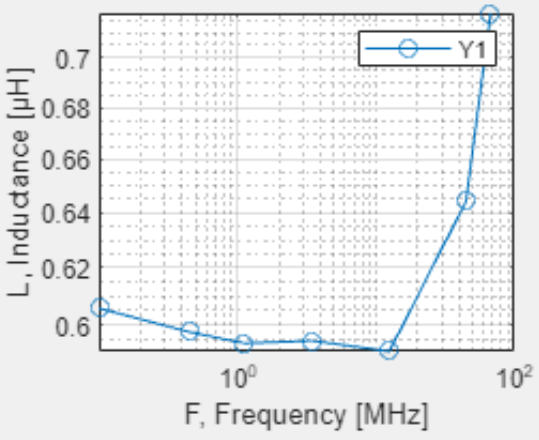


Figure 4.3: Datasheet of the XGL3520-561ME Coilcraft inductor.

GRM31CR60J107MEA8



Nameplate Parameters

Parameter	Value	Unit
C - Capacitance	100.0000	µF
Vdc - DC Rated Voltage	6.3000	V

Characteristics

Parameter	Test Conditions	Min	Typ	Max	Unit
C			100.0000		µF
Vdc			6.3000		V
Size			3216M/1206		One
Temp			X5R		°C

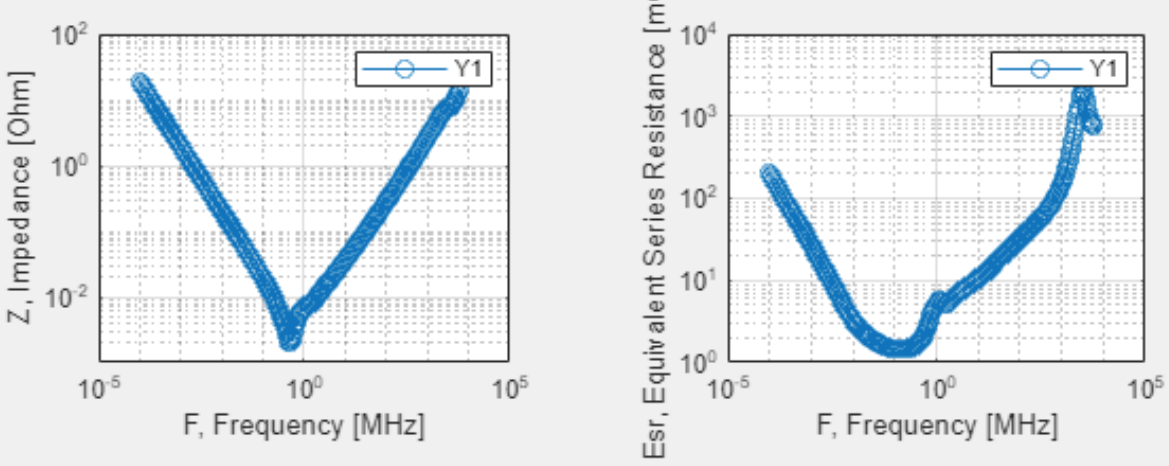


Figure 4.4: Datasheet of the GRM31CR60J107MEA8 Murata capacitor .

cover a broader scope and metaheuristic optimization methods to perform faster, there are formal techniques that take advantage of the modeling framework.

As shown in Fig. 1.2, each of the design objectives used in the optimization are outputs of the converter modeling, such as power loss or power density. The inputs to the modeling process include variables of a converter, such as the selection of semiconductor and passive devices, topology, and switch modulation strategy.

Approaches to optimize converters are already used within the discrete time state-space modeling process itself. For example, in order to account for state-dependent switching, a first-order approximation is used to identify and correct forward voltage or body diode violations iteratively to find the converters steady-state [34]. A similar gradient-based approach is used in this work to examine selection of discrete devices.

An example of a continuous space is shown in Fig. 4.5 for the conduction resistance and output capacitance of a power transistor. For some hypothetical converter, contours of constant power loss are represented by the solid lines with the gradient shown via the purple hue. The arrows point in the direction of steepest descent [105].

While the gradient-based approach can be used, there are additional constraints that must be considered. Many input variables to the modeling are not continuous and/or are unordered, such as the selection of converter topology. The selection of power devices among discrete transistors, in contrast, can be arranged into a pseudo-ordered multivariable space of the individual device parameters where gradient-based methods may be used.

An example of a gradient-based, discrete optimization approach is shown in Fig. 4.6. An initial discrete point is chosen and modeled to assess the converter's efficiency. The parameters of the converter semiconductors are then perturbed by ΔR_{on}, ΔC_{oss} to determine their effect on the converter's performance. The perturbations, however, are not discrete points defined by other semiconductor devices, but rather scaled to accurately estimate the steepest gradient of the converter power loss at the initial condition. The perturbed performance metrics, $\frac{\delta\eta}{\delta R_{on}}$ and $\frac{\delta\eta}{\delta C_{oss}}$, are solved using discrete time state-space modeling. The vector of steepest descent is then approximated by solving for the slope of the output

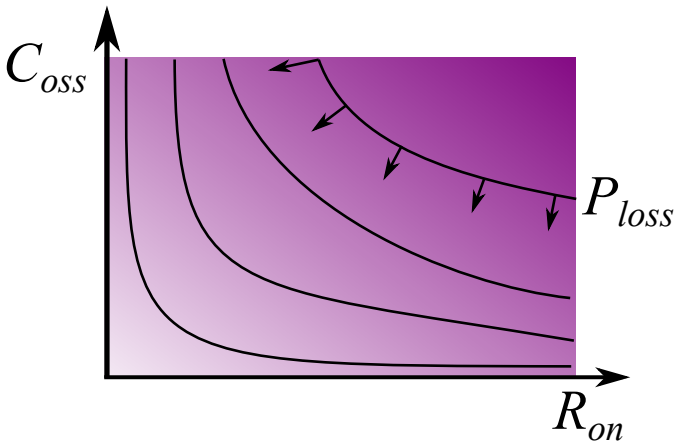


Figure 4.5: Continuous representation of a transistor power loss swept across output capacitance and conduction resistance.

perturbation over the input perturbation,

$$m_{gd} = \left. \frac{\frac{\Delta P_{loss,\Delta R_{on}}}{\Delta R_{on}}}{\frac{\Delta P_{loss,\Delta C_{oss}}}{\Delta C_{oss}}} \right|_{M_{IC}} . \tag{4.1}$$

In this example, the vector of steepest descent, v, is a line with the slope from (4.1) through the initial device M_{IC} with power loss $P_{loss,IC}$. This methodology can be expanded to higher orders to account for multiple input and output modeling variable possibilities.

The next transistor to be modeled must now be selected from the direction of steepest descent. To determine the next iteration, an estimated power loss of the discrete points is calculated using a linear projection from $P_{loss,\Delta R_{on}}$ and $P_{loss,\Delta C_{oss}}$. The discrete device with the lowest projected loss is then confirmed to be an improvement from the initial conditions.

To select a discrete point for a subsequent iteration, the optimization computes the minimum distance d that must be traveled along the vector of steepest descent before each point will be closer than the original. Generally, for an initial point $p_0 = (p_{0,x}, p_{0,y})$ and any other point p_i, the distance is

$$d_i = \frac{1}{2} \frac{(p_{0,x} - p_{i,x})^2 + (p_{0,y} - p_{i,y})^2}{(p_{0,x} - p_{i,x})\frac{\partial \eta}{\partial x} + (p_{0,y} - p_{i,y})\frac{\partial \eta}{\partial y}} . \tag{4.2}$$

Then, the best selection for the next iteration of the optimization is the point with minimum nonnegative d_i.

In order to verify the proposed method two separate examples are completed. First, several topologies are designed to find an optimal converter with a 16-20 V input and up to a 40 W, 8 V output showcasing the discrete time state-space models and the optimization framework for discrete transistors. Second, a 48 V-1 V converter with varying output power is assessed; however, first, optimization methodology utilizing discrete time state-space modeling is introduced.

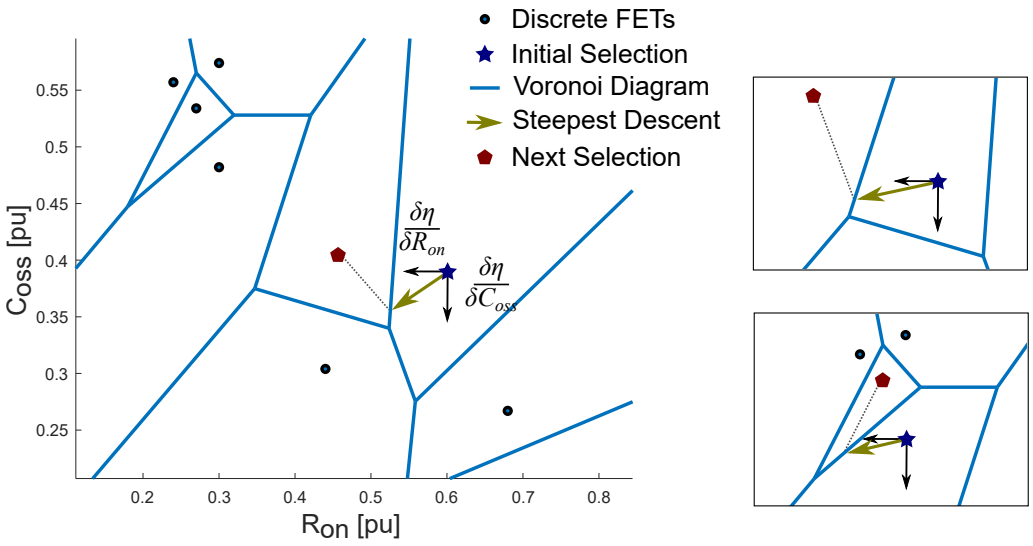


Figure 4.6: Example iterative transistor selection from discrete devices. Individual FETs are shown with a Voronoi diagram (dashed gray lines) showing the regions closest to each discrete device. Cases of selecting subsequent iteration are shown in.

4.2 Optimization Methodology

A switch mode power converter (⧄) can be optimized any number of metrics such as the reduction of power loss, cost, failure rate, weight, and area to list some examples. Any combination of these metrics can be used to evaluate the optimization. For now, this is listed as a function evaluated of the converter $\boldsymbol{f}(⧄)$.

$$\begin{aligned} \min \quad & f\Big(⧄\Big) \\ \text{s.t.} \quad & \{\boldsymbol{h}, \boldsymbol{k}\} \in ⧄ \\ & \{\boldsymbol{\tau}, \boldsymbol{q}, \boldsymbol{d}, \boldsymbol{r}, \boldsymbol{l}, \boldsymbol{c}\} \in \boldsymbol{h} \\ & \{f_s, \boldsymbol{t}_s, \boldsymbol{u}\} \in \boldsymbol{k} \end{aligned} \tag{4.3}$$

The converter, ⧄, contains a set of variables and/or constraints depending on the scope of the optimization. There are two main categories the variables of the converter can be separated into: fixed or hardware variables, $\boldsymbol{h}$, and operation or controlling variables $\boldsymbol{k}$. Fixed variables consist of the hardware used to implement the converter. This includes the topology ($\boldsymbol{\tau}$) and the elements that are used to physically implement the converter such as the transistors ($\boldsymbol{q}$), diodes ($\boldsymbol{d}$), and passive components ($\boldsymbol{r}, \boldsymbol{l}, \boldsymbol{c}$). Controlling variables are elements that are used in the operation of the hardware and are used to change the operating point of the converter. These include switching frequency (f_s), switch timing ($\boldsymbol{t}_s$), and converter inputs ($\boldsymbol{u}$), such as input voltage.

From a hardware perspective, the topology selection determines the number of physical elements that are implemented in the converter. For each topology, there is a fixed arrangement of hardware elements with a set number of devices. For example, a synchronous buck converter topology would have two transistors $\boldsymbol{q} = \{q_1, q_2\}$ while a dual active bridge topology would have eight transistors $\boldsymbol{q} = \{q_1, q_2, \ldots, q_8\}$. For every evaluated hardware implementation $\big|_{\boldsymbol{h}}$, there are an infinite number of operating points that can be achieved,

$$\begin{aligned} \min \quad & f\Big([⧄]\big|_{\boldsymbol{h}}\Big) \\ \text{s.t.} \quad & \{\boldsymbol{h}, \boldsymbol{k}\} \in ⧄ \\ & \{f_s, \boldsymbol{t}_s, \boldsymbol{u}\} \in \boldsymbol{k}. \end{aligned} \tag{4.4}$$

In order to accurately assess each hardware implementation, the controlling variables must be optimized to find a valid operating conditions of the hardware. Examples include finding the optimal switching frequency and deadtimes to reduce power loss. In this work, the focus remains on operating in steady-state utilizing discrete time state-space modeling; therefore, the switch timing $\boldsymbol{t}_s$ must be optimized to find the steady-state point of the converter.

Steady-state Optimization

To find the steady-state of a converter when using time stepping simulators, the simulation is run until the initial and ending state values within a period are equal as shown in Chapter 2.2. However, discrete time state-space modeling is used to solve for the steady-state of the converter; therefore, in addition to optimizing for the topology, discrete transistors, and/or passives implemented, the steady-state modeling can be shown as an optimization of dependent switching actions, j, within the independent switching actions, i. The switch timing $\boldsymbol{t}_s$ is separated into dependent time intervals $\boldsymbol{t}_j$ that occur within independent time intervals $\boldsymbol{t}_i$.

$$
\begin{aligned}
\min_{\boldsymbol{A}_j, \boldsymbol{t}_j} \quad & \boldsymbol{y}_{bnd(i,j)}(t) = \boldsymbol{C}_{bnd(i,j)}\boldsymbol{x}(t) + \boldsymbol{D}_{bnd(i,j)}\boldsymbol{u}(t) < \mathbf{0} \\
\text{s.t.} \quad & \boldsymbol{X}_{ss} = \left[\mathbf{I} - \prod_{i=n}^{i} \boldsymbol{\Phi}_i\right]^{-1} \sum_{i=1}^{n} \left[\boldsymbol{\Psi}_i\right] \\
& \boldsymbol{\Phi}_i = \prod_{j=p_i}^{1} \mathrm{e}^{\boldsymbol{A}_{(i,j)} t_{(i,j)}} \\
& \boldsymbol{\Psi}_i = \sum_{j=1}^{p_i} \left[\left(\prod_{k=n}^{i+1} \boldsymbol{\Phi}_k \right) \left(\prod_{k=p_i}^{j+1} \mathrm{e}^{\boldsymbol{A}_{(i,k)} t_{(i,k)}} \right) \boldsymbol{A}_{(i,j)}{}^{-1} \left[\mathrm{e}^{\boldsymbol{A}_{(i,j)} t_{(i,j)}} - \mathbf{I} \right] \boldsymbol{B}_{(i,j)} \boldsymbol{u} \right]
\end{aligned}
\tag{4.5}
$$

The state-space output equation $\boldsymbol{y}_{bnd(i,j)}(t)$ determines if the converter is operating at a valid steady-state solution. The matrices $\boldsymbol{C}_{bnd(i,j)}$ and $\boldsymbol{D}_{bnd(i,j)}$ are formed as bounded constraint matrices. For each switching interval there are bounds on certain converter voltages and currents that cannot exceed limits set by the physical implementation of the converter. The most common of these limitation is the voltage current relationship of a diode moving between conducting and non-conducting states as discussed in Chapter 3.

The separation of independent, i, and dependent, j, switching actions indexing is showcased in Fig. 4.7 and Table 4.1 from the Chapter 3.3 buck converter example on independent and dependent switching actions. The independent switch action, i, does not change while finding the steady-state of the converter to preserve the duty cycle operating point; however, the state-dependent switching actions, j, are altered to ensure the modeled converter accurately depicts state-dependent behavior such as diode conduction.

State-Space Averaging

One of the most common metrics used to evaluate a power converter is power loss. The power loss of the converter, $p_{loss}(⍰)$, is calculated as the average input voltage, $\overline{v_g}$, and current, $\overline{i_g}$, minus the average output voltage, $\overline{v_o}$, and current, $\overline{i_o}$ while the converter is operating in steady-state,

$$p_{loss}(⍰) = \overline{v_g}\overline{i_g} - \overline{v_o}\overline{i_o}. \tag{4.6}$$

A state-space output equation

$$\begin{bmatrix} \overline{v_g} \\ \overline{i_g} \\ \overline{v_o} \\ \overline{i_o} \end{bmatrix} = \overline{\boldsymbol{y}_{out}} = \boldsymbol{C}_{out}\overline{\boldsymbol{x}} + \boldsymbol{D}_{out}\boldsymbol{u} \tag{4.7}$$

calculates the required input and output voltage and current values but requires the average state values $\overline{\boldsymbol{x}}$ as an input. To find the average of each state, an augmented state matrix is constructed after the steady-state of the converter is reached. The process of how to calculate $\overline{\boldsymbol{x}}$ is introduced in Chapter 3.2 and reiterated here.

By creating an augmented state matrix

$$\check{\boldsymbol{A}}_i = \left[\begin{array}{c|c|c} \boldsymbol{A}_i & \boldsymbol{B}_i & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{I} & \mathbf{0} & \mathbf{0} \end{array}\right], \tag{4.8}$$

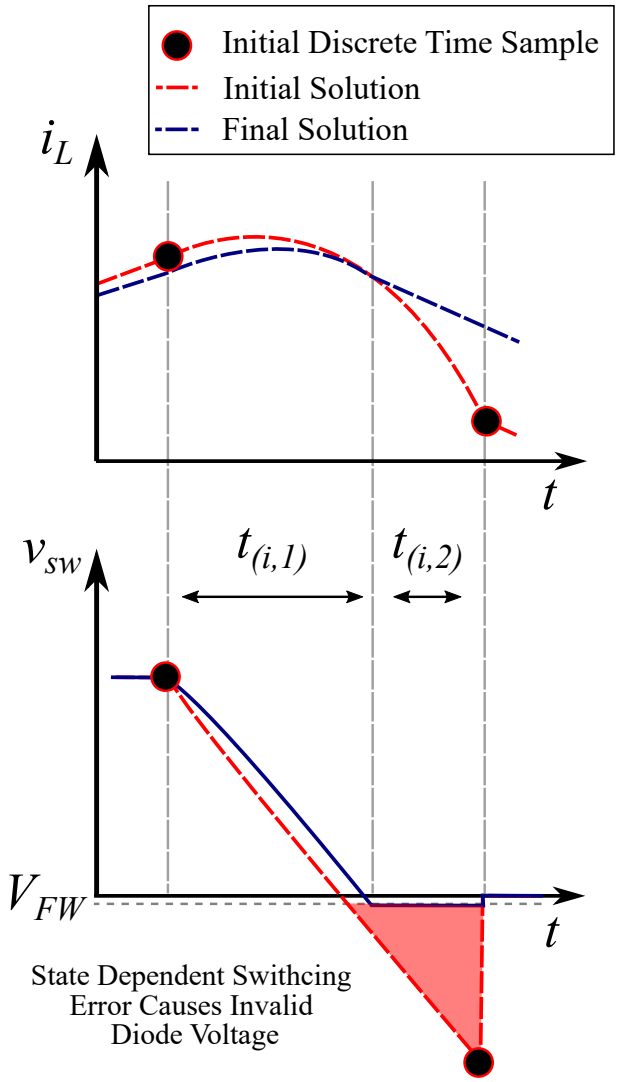


Figure 4.7: Waveform of a synchronous buck converter after the high side device turns off. The initial guessed deadtime length is too long between active switching actions. Therefore an additional dependent switching interval $(i, 2)$ is inserted into the independent switching interval i to resolve the error in dependent switching actions.

the system

$$\frac{d}{dt}\hat{\boldsymbol{x}}(t) = \hat{\boldsymbol{A}}_i\hat{\boldsymbol{x}}(t), \tag{4.9}$$

has solution

$$\check{\boldsymbol{x}}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \hline \boldsymbol{u}_i \\ \hline \int_0^t x(\tau)d\tau \end{bmatrix} = \mathrm{e}^{\check{\boldsymbol{A}}_i t_i}\check{\boldsymbol{x}}(0). \tag{4.10}$$

The integral of the states is calculated by setting the derivatives of the new states equal to the instantaneous value of the original states [95]. Iterating through all time intervals and multiplying by the frequency of the converter gives the average of all the state values

$$\overline{\boldsymbol{x}} = f_s \int_0^{T_s} x(\tau)d\tau. \tag{4.11}$$

Multiple Converter Operating Points

In most applications, a singular operating point is not sufficient to assess a converter's viability. Therefore, an array of controlling variables is used to assess a set hardware implementation at different operating points. For example, if a converter was required to operate over a range of input voltages, an array of operating points for each hardware implementation must be assessed. The set of input voltages, for example, $V_g = \{5, 10, 15, 20\} \in \boldsymbol{u}$, are then each individually assessed at steady-state. This would then provide an array of power loss values

$$\boldsymbol{p}_{loss}\Big([\boxtimes]\Big) = \Big\{p_{loss}\Big([\boxtimes]\big|_{V_g=5}\Big), p_{loss}\Big([\boxtimes]\big|_{V_g=10}\Big), p_{loss}\Big([\boxtimes]\big|_{V_g=15}\Big), p_{loss}\Big([\boxtimes]\big|_{V_g=20}\Big)\Big\}. \tag{4.12}$$

evaluated at each individual input voltage. The array of power loss values can then be multiplied with a weighting function to prioritize a subset of the design space if desired and then averaged to achieve a singular numerical metric to compare converter hardware implementations.

4.3 Example Design Optimizations

4.3.1 Battery Charger

In this example, the following converter optimization is used to showcase the optimization methodology for battery charging applications

$$\begin{aligned} \min_{\boldsymbol{\tau},\boldsymbol{q}} \quad & f\Big(p_{loss}\Big(\boxtimes\Big)\Big) \\ \text{s.t.} \quad & \{\boldsymbol{h},\boldsymbol{k}\} \in \boxtimes \\ & \{\boldsymbol{\tau},\boldsymbol{q},\boldsymbol{d},\boldsymbol{r},\boldsymbol{l},\boldsymbol{c}\} \in \boldsymbol{h} \\ & \{f_s,\boldsymbol{t}_s,\boldsymbol{u}\} \in \boldsymbol{k} \end{aligned} \tag{4.13}$$

where $\boldsymbol{q}$ represents the discrete transistor selection from Table 4.2 within topology $\boldsymbol{\tau}$. The number of transistors to be optimized is dependent on the topology selected. The converter $\boxtimes$ is evaluated at steady-state over the range of input voltage, V_g, and output power, P_{out}, in this example restricted by

$$\begin{aligned} & 16 \text{ V} \leq V_g \leq 20 \text{ V} \\ & 20 \text{ W} \leq P_{out} \leq 40 \text{ W} \\ & V_{out} = 8 \text{ V}. \end{aligned} \tag{4.14}$$

$\boldsymbol{p}_{loss}([\boxtimes])$ finds the power loss of the converter evaluated at a set of valid operating points between valid V_g and P_{out} constraints as shown in Fig. 4.8. The power loss values are then linearized over the design space, Fig. 4.9, and element multiplied with a weighting function to prioritize a subset of the design space as shown in Fig. 4.10. Finally, the mean of the resulting matrix, as shown in Fig. 4.11, is taken to place a single numerical evaluation of topology operating between the design constraints.

Five topologies shown in Fig. 4.12-4.16 showcase the optimization process. Each figure shows the topology schematic, the iteration path of the transistor selection, and the resulting steady-state power loss at a range of power levels and input voltages. For each topology, an initial starting point is selected at random for each transistor. All FETs within the design space are indicated by small blue circles. The converters are optimized as FET selection is

Table 4.1: Buck Converter Deadtime Solution

Initial Solution		
Time Interval	$t_{(i,1)}(t_{dt})$	
Circuit Model	$\boldsymbol{A}_{(i,1)}(\boldsymbol{A}_{i,BD(off)})$	
Conducting Devices	—	
Final Solution		
Time Interval	$t_{(i,1-2)}(t_{dt})$	
	$t_{(i,1)}(t_{BD(off)})$	$t_{(i,2)}(t_{BD(on)})$
Circuit Model	$\boldsymbol{A}_{(i,1)}(\boldsymbol{A}_{i,BD(off)})$	$\boldsymbol{A}_{(i,2)}(\boldsymbol{A}_{i,BD(on)})$
Conducting Devices	—	$M_{2,BD}$

Table 4.2: EPC GaN Devices

Device	Index	R_{on} (mΩ)	C_{oss} (nF)
EPC 2023	1	1.45	1.530
EPC 2015C	2	4.00	0.710
EPC 2055	3	3.60	0.408
EPC 2030	4	2.40	1.120
EPC 2024	5	1.50	1.620
EPC 2031	6	2.60	0.980
EPC 2020	7	2.20	1.020
EPC 2065	8	3.60	0.534
EPC 2029	9	3.20	0.820
EPC 2021	10	2.20	1.100
EPC 2018	11	3.20	0.56
EPC 2302	12	1.80	1.000
EPC 2069	13	2.25	1.044
EPC 2067	14	1.55	1.071
EPC 2066	15	1.10	1.670

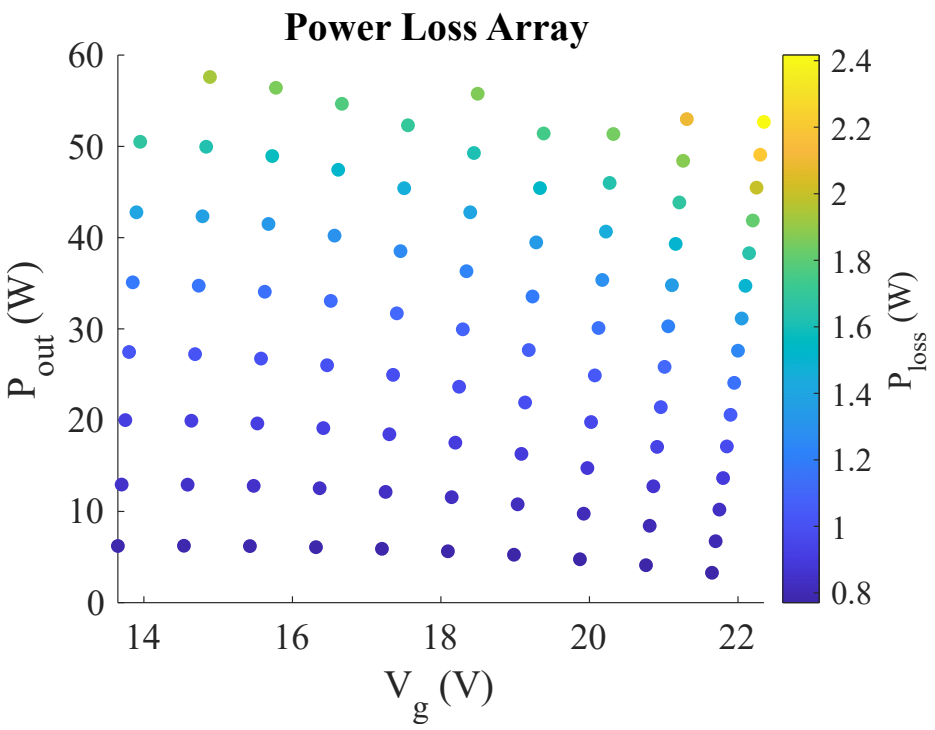


Figure 4.8: Power loss values for steady-state topology implementations.

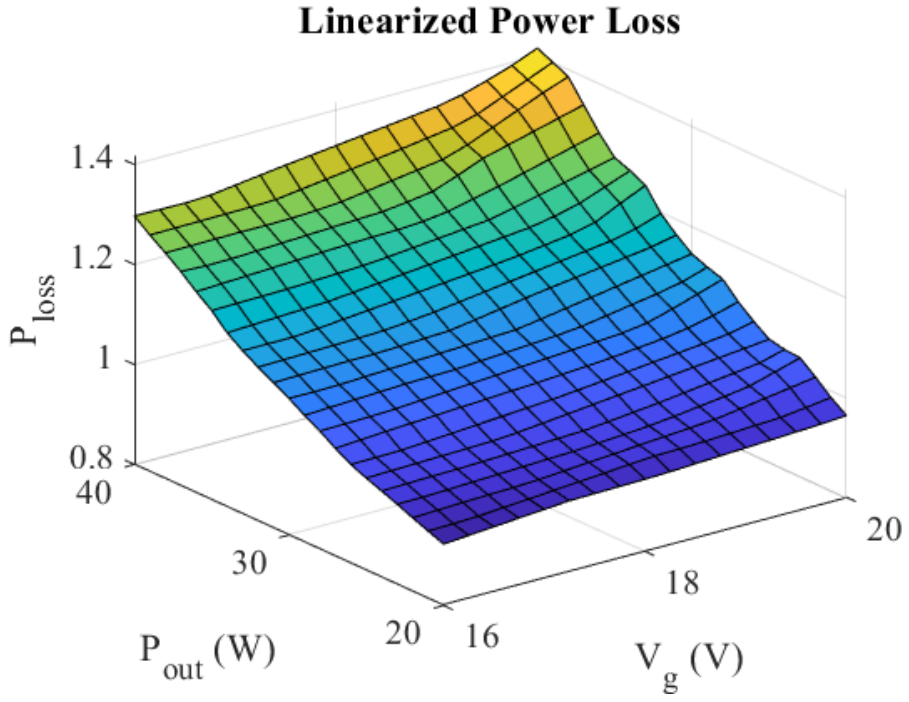


Figure 4.9: Linearized surface of power loss values taken from Fig. 4.8.

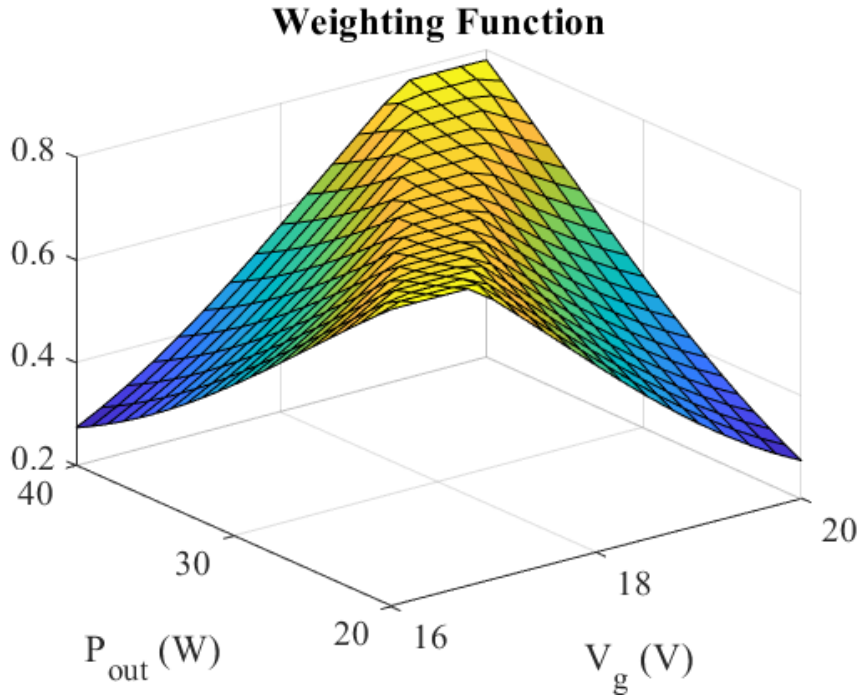


Figure 4.10: Example weighting function

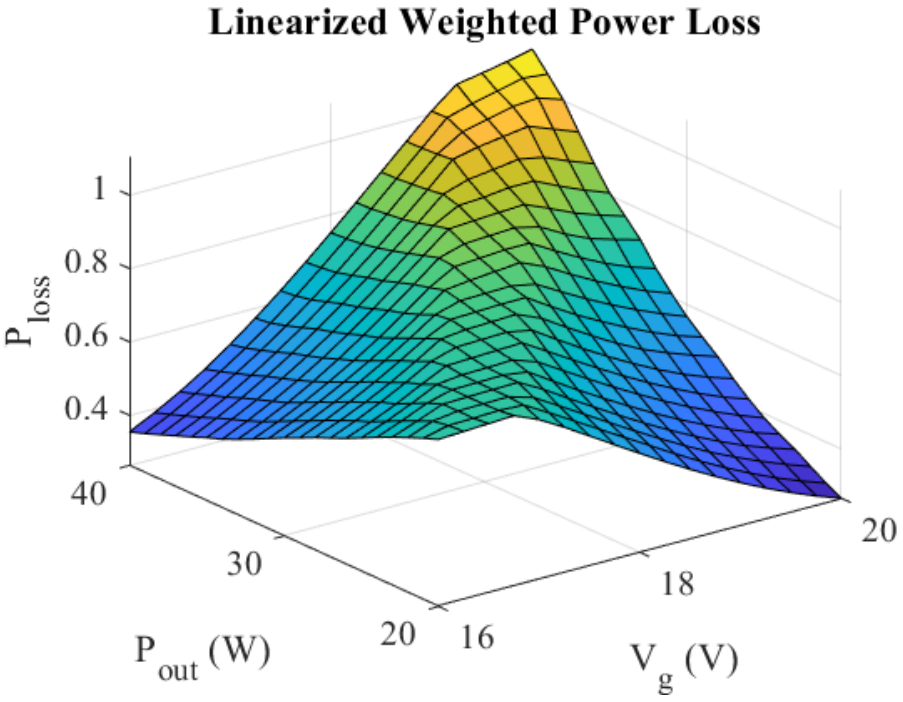


Figure 4.11: Combined weighting function and linearized power loss surfaces

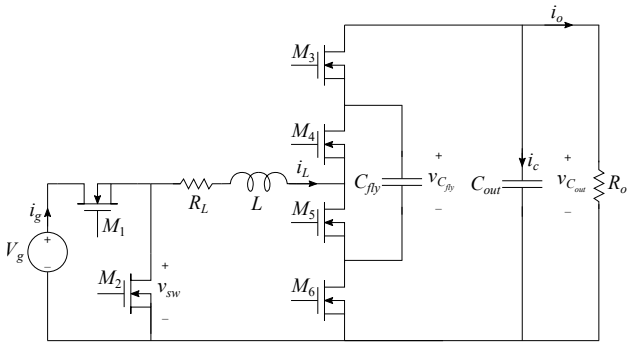
indicated by the lines from one circle to another, until the optimization terminates when there can be no more improvements from the direction of steepest descent. The red circles indicate a final optimized point.

The buck-boost topology in Fig. 4.12 indicates a need for low a R_{on} to account for the large current ripple at low duty cycles. To improve duty cycle, the buck switch capacitor topology, Fig. 4.14, improves upon its buck-boost counterpart, especially when operating where ZVS is reached at high power. A Fibonacci sequence switch capacitor circuit is examined in Fig. 4.15 where power loss is significantly improved over the wide range of input voltages. Fig. 4.16 and Fig. 4.13 examine large circuits capable of wider input voltage range than specified for this example and have dual 4V battery outputs while having slightly greater power loss than the 8V Fibonacci sequence switch capacitor circuit.

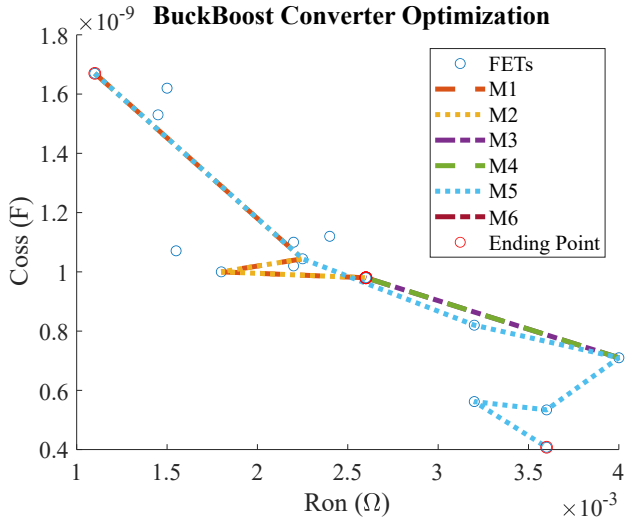
To confirm the proposed optimization method reached the optimal point, a brute force method was used for all possible combinations for the 8V switch capacitor Fibonacci converter as well as the 4V switch capacitor Fibonacci converter. For this example, $M_7 - M_{10}$ are ignored in the 4V switch capacitor Fibonacci converter since they are non-switching FETs for the constrained voltage range, and in both topologies, FETs are combined to form switch cells that are mirrored. All of the possible transistor combinations are assessed and displayed in Fig. 4.17 and Fig. 4.18 for the 8V topology and Fig. 4.19 and Fig. 4.20 for the 4V topology where the devices are indexed according to Table 4.2. Each point changes in size and in color depending on the power loss using the specific transistor combination, and the point of minimum power loss is indicated by a star. For the two topologies tested, the optimum devices matched the proposed optimization methods results with all transistors converging to the EPC 2055 except for M_3, M_6, M_{13}, and M_{16} on the 4V topology that converged to the EPC 2067.

Experimental Results

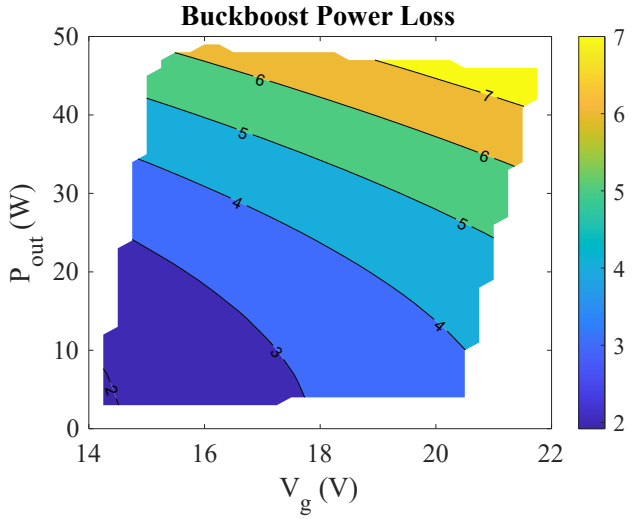
The switched capacitor Fibonacci converter shown in Fig. 4.21 is selected to experimentally validate the modeling method. For the prototype, the design was constrained to have all the same transistors, rather than optimizing each device individually as in the previous section. The EPC2055 was chosen via a sweep of the design space, as shown in Fig. 4.22.



(a)

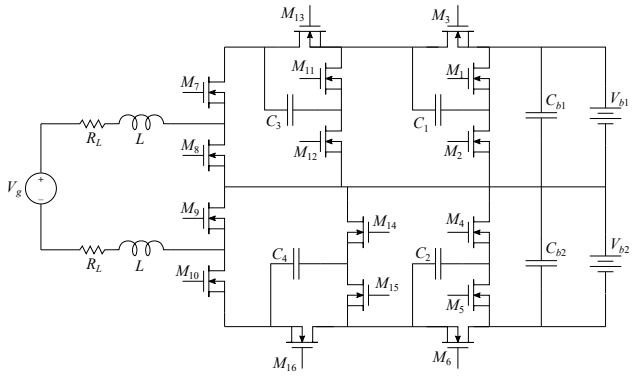


(b)

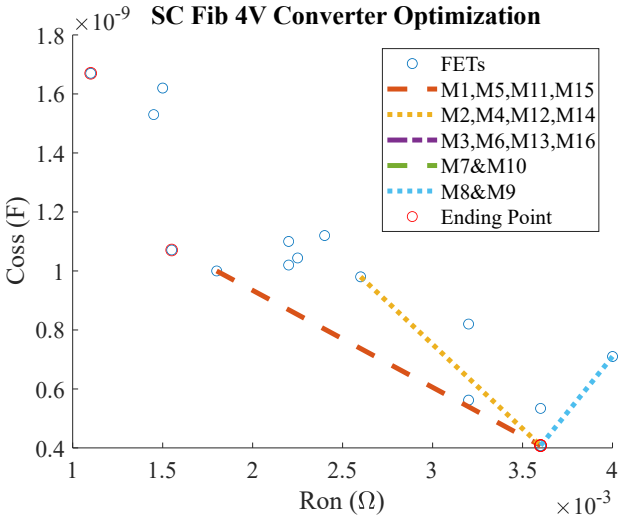


(c)

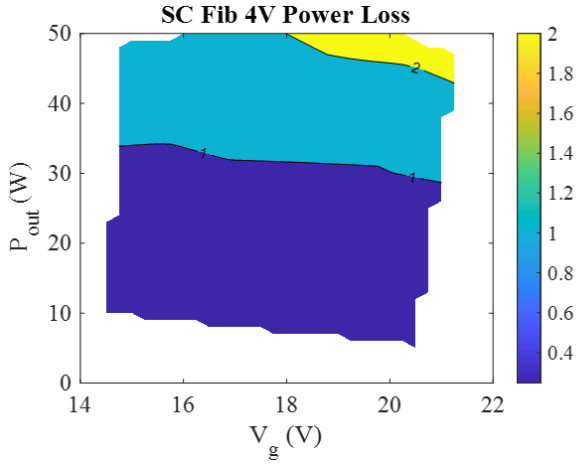
Figure 4.12: Buck-boost converter schematic (a) optimization process (b) modeled results (c).



(a)

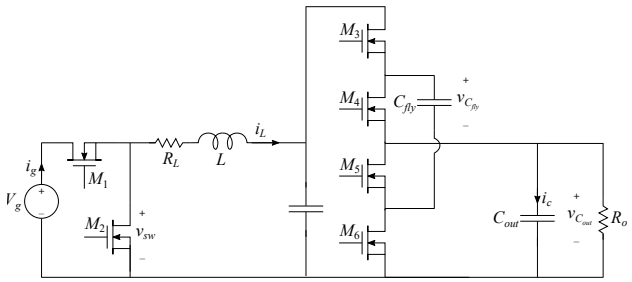


(b)

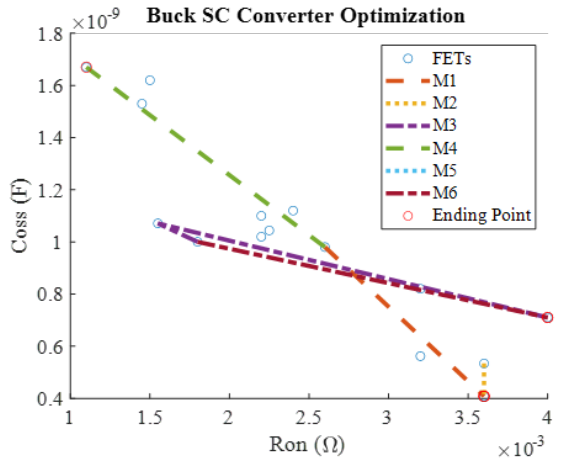


(c)

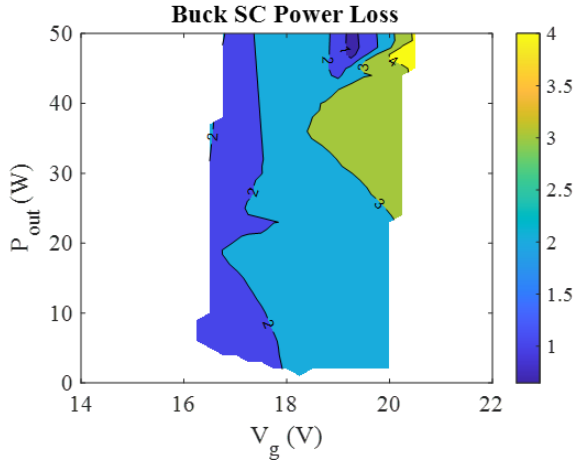
Figure 4.13: 4V switched capacitor Fibonacci sequence schematic (a) optimization process (b) modeled results (c).



(a)

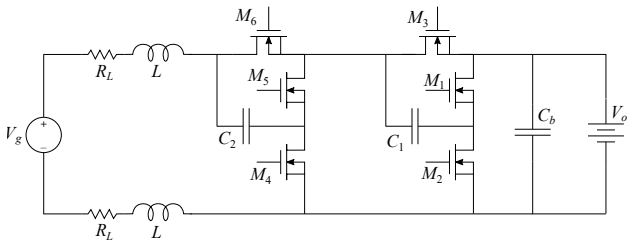


(b)

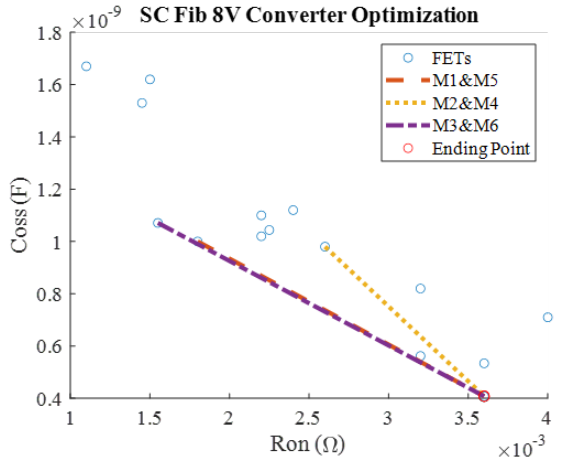


(c)

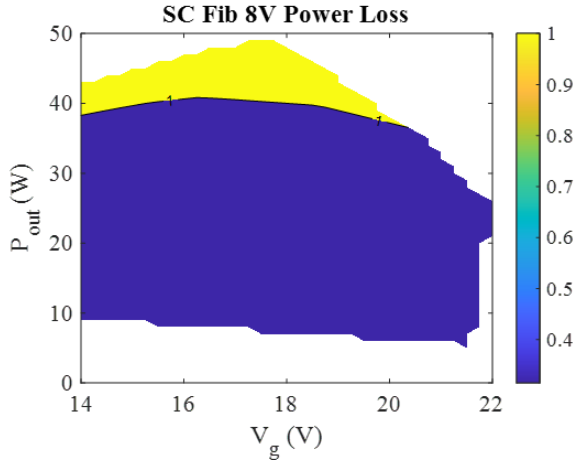
Figure 4.14: Buck switched capacitor schematic (a) optimization process (b) modeled results (c).



(a)

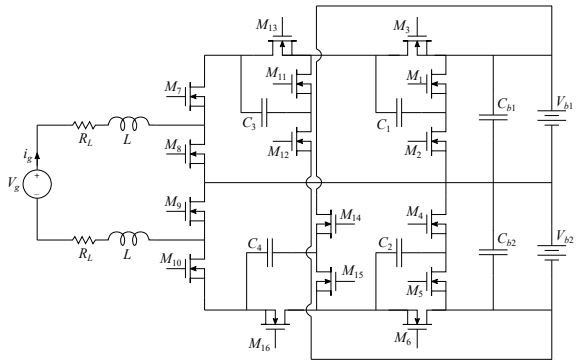


(b)

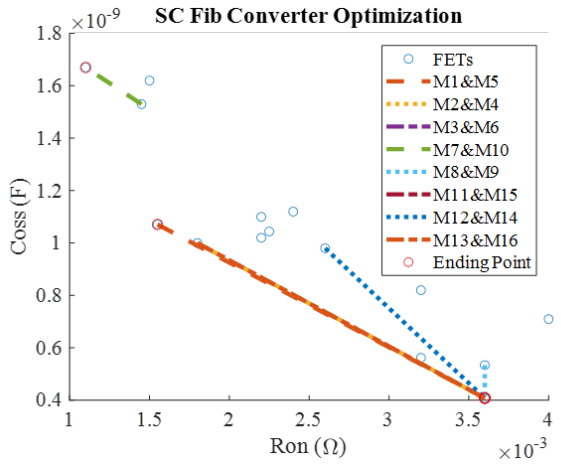


(c)

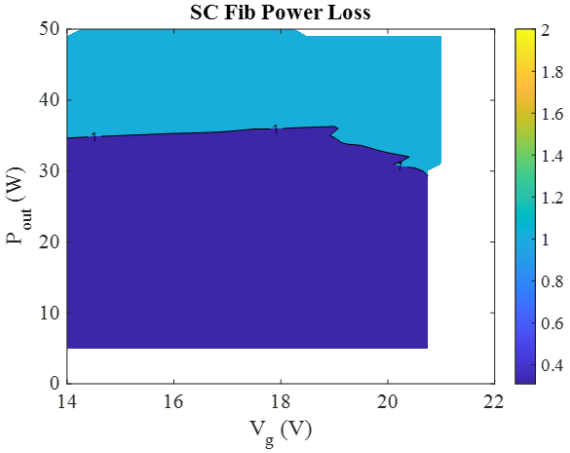
Figure 4.15: 8V switch capacitor Fibonacci sequence schematic (a) optimization process (b) modeled results (c).



(a)



(b)



(c)

Figure 4.16: Switched capacitor Fibonacci sequence schematic (a) optimization process (b) modeled results (c).

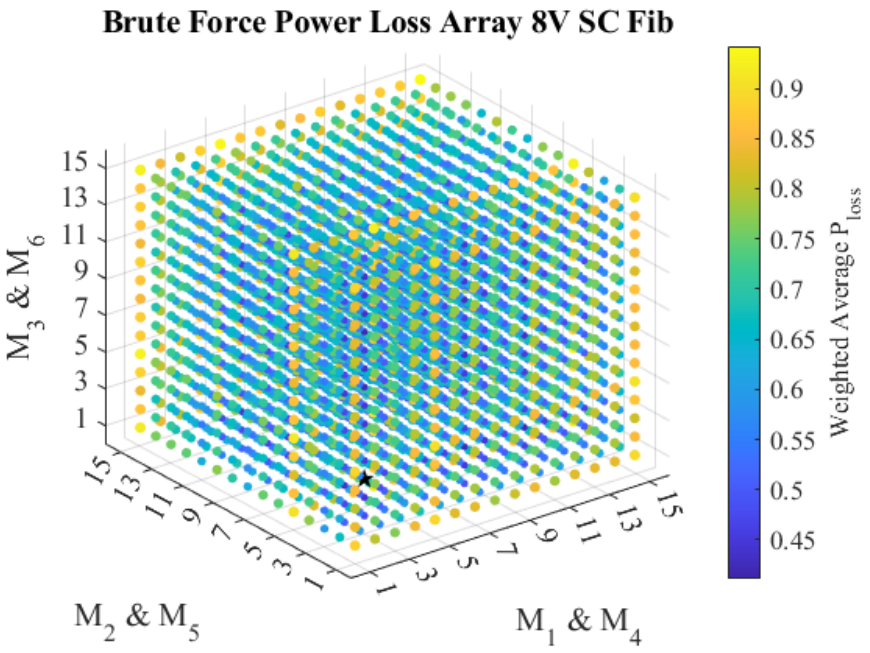


Figure 4.17: All possible discrete device combinations for the 8V switch capacitor Fibonacci topology.

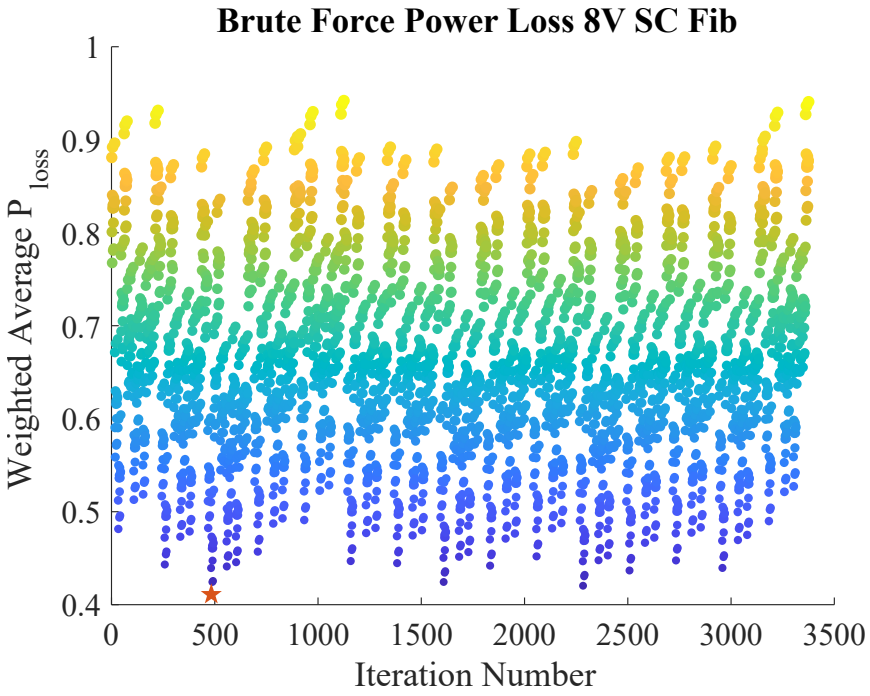


Figure 4.18: Brute force iteration of all discrete device combinations for the 8V switch capacitor Fibonacci topology.

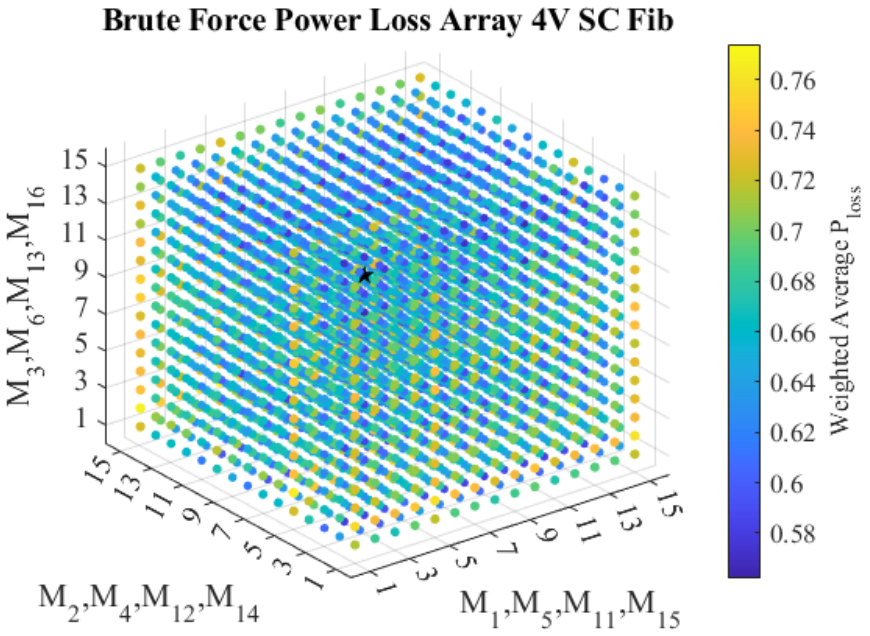


Figure 4.19: Some possible discrete device combinations for the 4V switch capacitor Fibonacci topology.

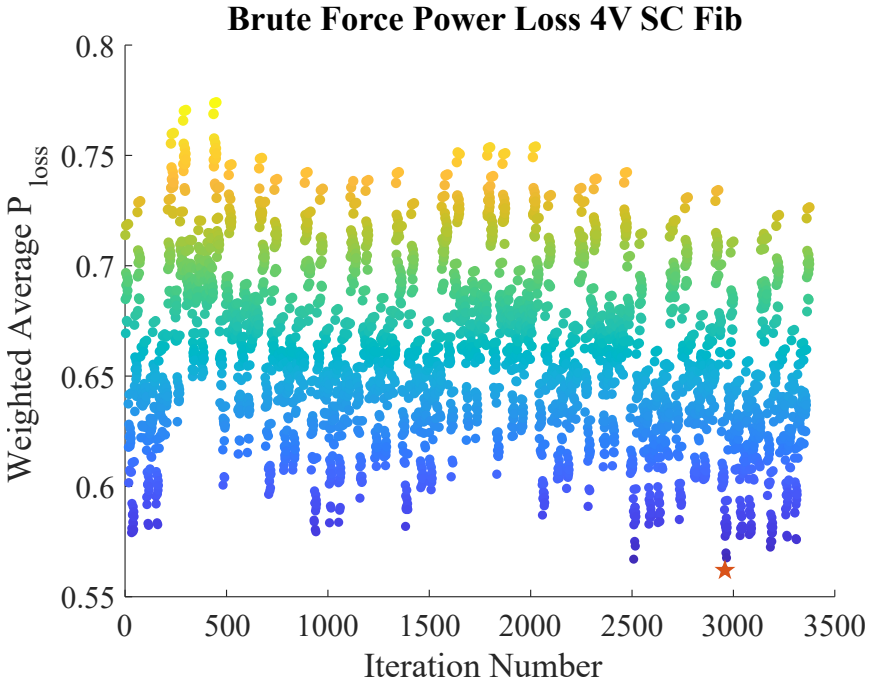
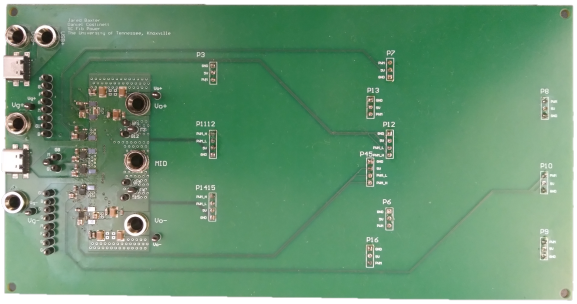
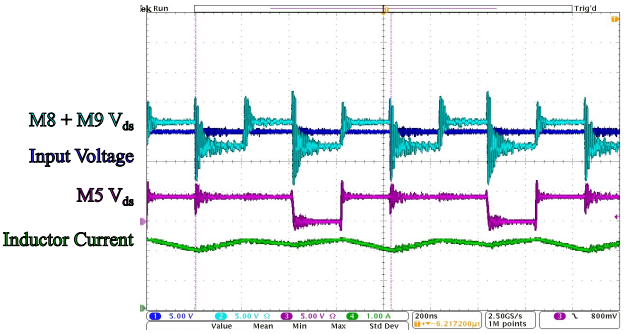


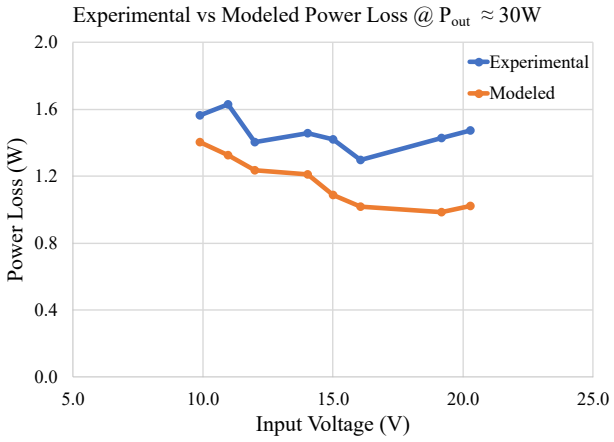
Figure 4.20: Brute force iteration of some discrete device combinations for the 4V switch capacitor Fibonacci topology.



(a)



(b)



(c)

Figure 4.21: Switched capacitor Fibonacci prototype (a) experimental waveforms (b) modeled vs experimental results at 30 W output power (c).

Additional converter parameters are given in Table 4.3. As shown in Fig. 4.21, the layout of the converter requires large decoupling loops across all of the switch capacitor cells. The large decoupling loops lead to increased parasitic inductance throughout the circuit causing increased switching ripple and increasing power loss at higher input voltages compared to modeled results.

Integrated Circuit Prototype

After the discrete switch capacitor Fibonacci converter was completed, an integrated circuit (IC) design of the topology was optimized and experimentally verified. Due to the size constraint of the IC package and the time required for layout, the width's of three transistors were optimized to achieve the highest efficiency modules. Each of the modules is shown in Fig. 4.23 outlined in a purple dashed box. The modules for the buck stage of the converter, transistors $M_{7,8}$ and $M_{9,10}$, only use two of the three transistors in each module while the symmetry of the topology required the top and bottom transistor, $M_{2,3}$ for their respective module, be equal. The optimized range of module area and weighted average power loss (same weighting, input voltage, and output power range as the discrete example) are shown in Fig. 4.24 and 4.25.

The prototype is shown in Fig. 4.26 with an experimental results comparison in Fig. 4.27 where the estimated inductor loss is calculated using I^2R loss of the measured inductor current and resistance. The IC board layout still suffers from the same decoupling issue as the discrete board layout; however, this issue is mitigated, as indicated by the experimental power loss following the trend of modeled power loss.

4.3.2 48-to-1V

In this example, a 48-to-1V converter was designed to verify the proposed optimization method. The optimization database included a set of topologies from literature that has been used for high current, point of load converter. Fig. 4.30 and Fig. 4.31 showcases the 3-phase and 5-phase series capacitor buck, respectively [106]. Fig. 4.32 showcases an 3-phase series capacitor buck with switched-capacitor step down. This additional switch capacitor

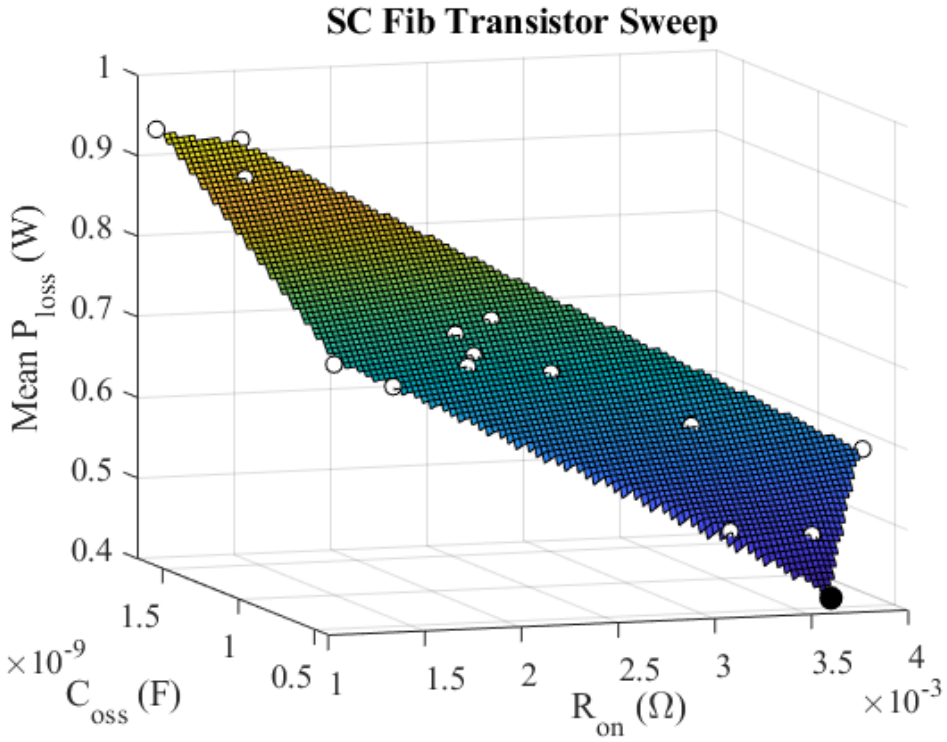


Figure 4.22: Switched capacitor Fibonacci transistor design space where all converter transistors are the same. The optimal point, shown by the black circle, is the EPC2055 transistor. P_{loss} is a weighted mean of hundreds of simulations for each transistor between the input voltage and output power range .

Table 4.3: Switch Capacitor Fibonacci Converter Parameters

L	0.65 µH	P_o	0-40 W
C_{1-4}	10 µF	$C_{oss,M1-16}$	408 pF
R_L	10 mΩ	R_{ds}	3.6 mΩ
V_g	16-20 V	f_s	1.2 MHz

Table 4.4: Series Capacitor Buck / Switched Capacitor Parameters

L	0.251 µH	$M_{1,2,3}$	EPC2044
R_L	0.31 mΩ	$M_{4,5,6}$	EPC2088
$C_{1,2}$	2x10 µF	M_{7-15}	EPC2055
$C_{3,4,5}$	2x47 µF	f_s	500 kHz
V_g	48 V		

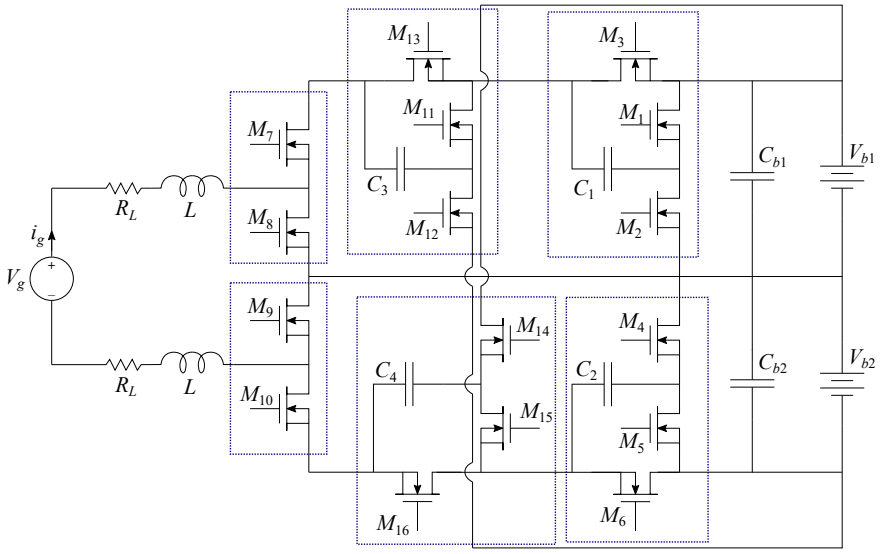


Figure 4.23: Switch capacitor Fibonacci converter with IC modules indicated by the blue dashed boxes.

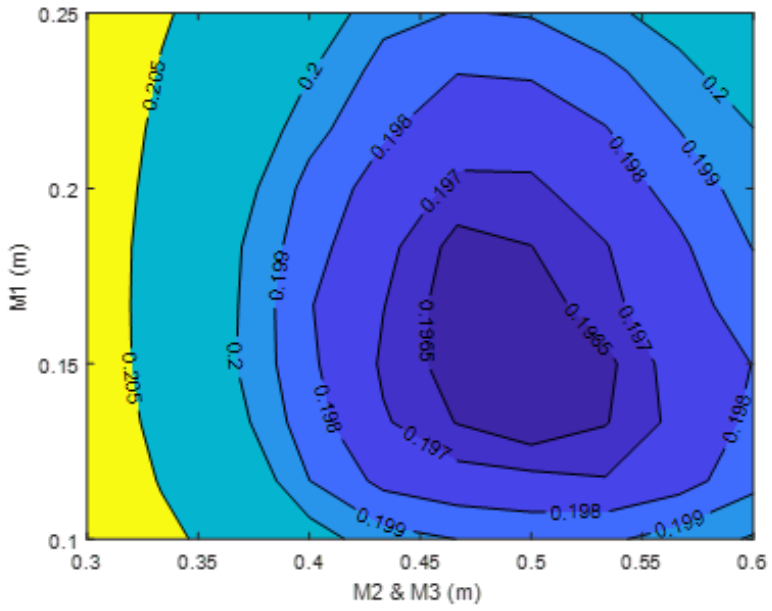


Figure 4.24: Switch capacitor Fibonacci converter IC weighted average power loss (W) for varying device widths.

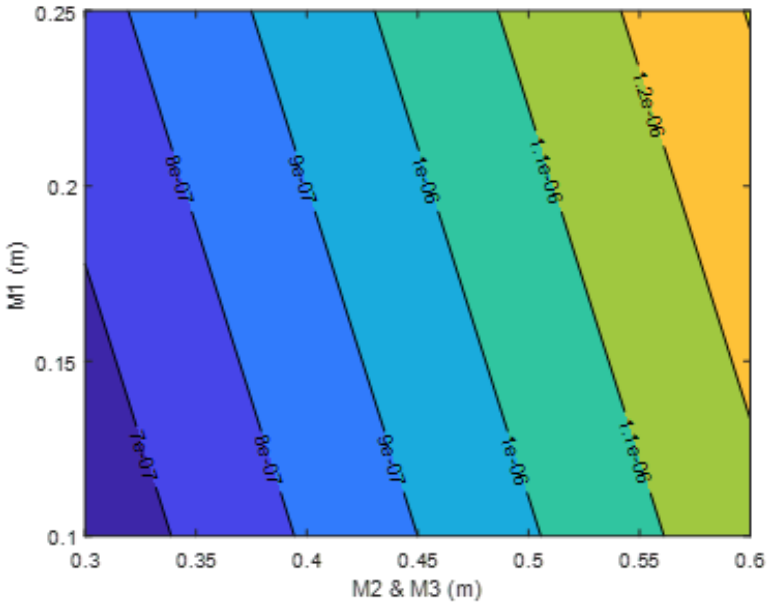


Figure 4.25: Switch capacitor Fibonacci converter IC cell area (m^2) for varying device widths.

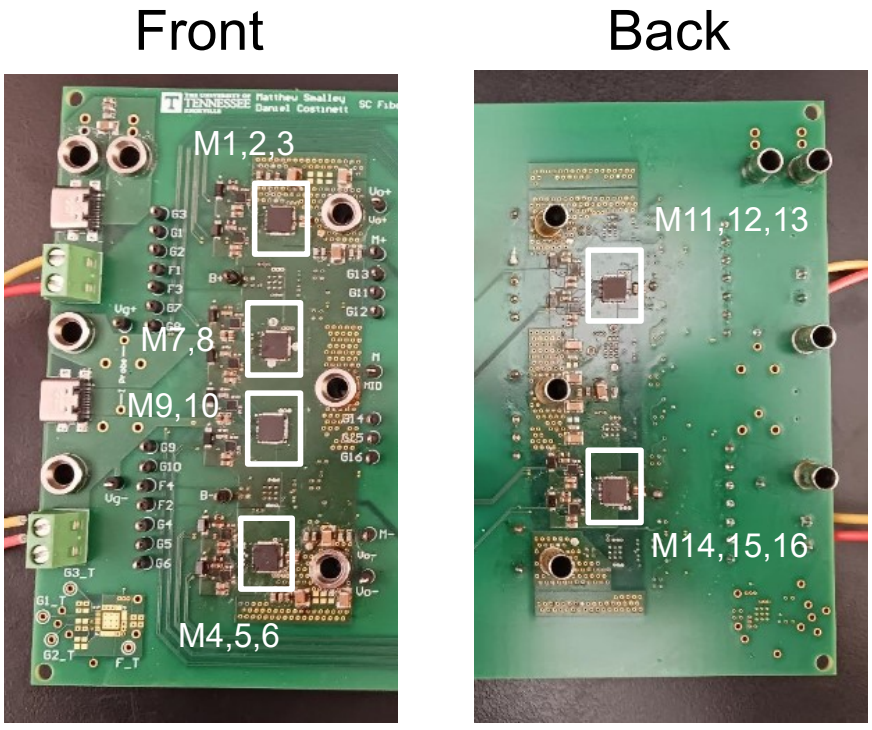


Figure 4.26: Switch capacitor Fibonacci IC prototype.

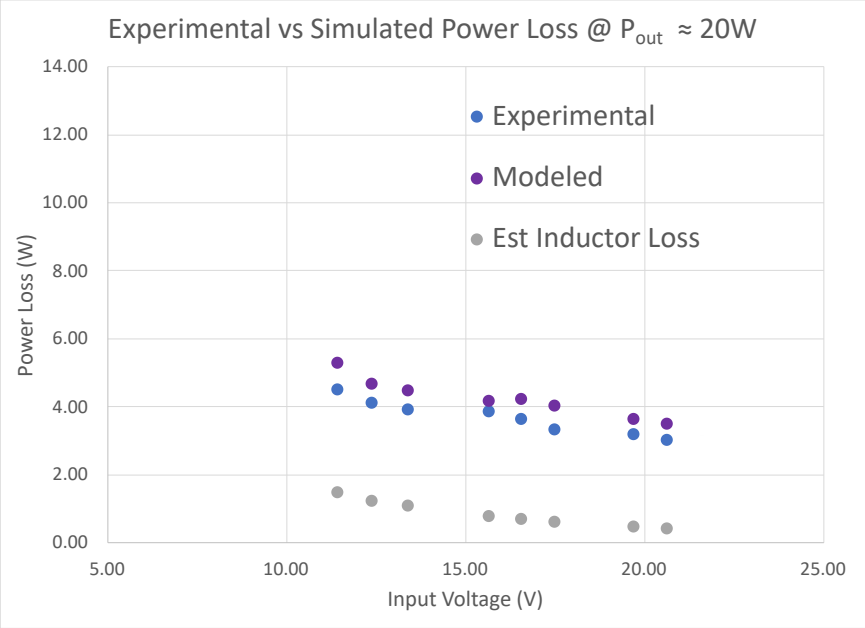
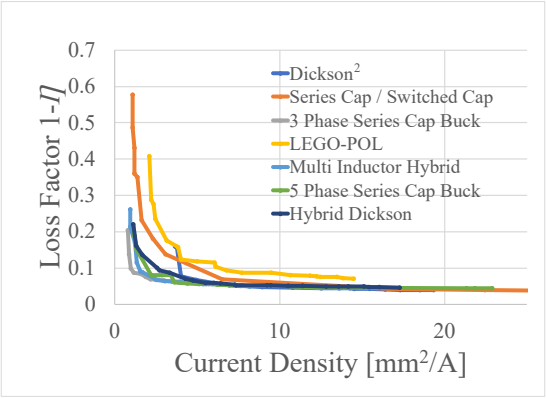
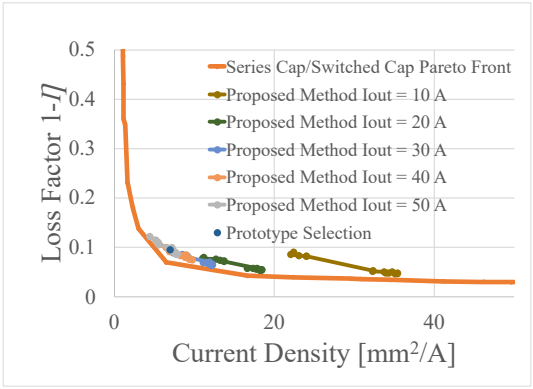


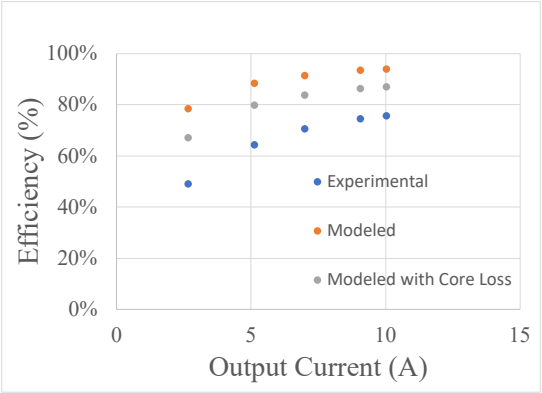
Figure 4.27: Switch capacitor Fibonacci converter IC results.



(a)



(b)



(c)

Figure 4.28: GA-based Pareto front for all topologies (a), convergence from multiple initial conditions of the proposed method when optimizing for efficiency only (b) and measured-vs-modeled efficiency of the prototype up to $I_{out} = 10$ A (c).

output stage is designed to reduce indcutor current loss [107]. Additional topologies included the LEGO-POL Fig. 4.33 [108], multi-inductor hybrid Fig. 4.34 [109], and the Dickson^2 Fig. 4.35 [110]. A database of 20 EPC GaN devices modeled approximately by R_{on} and C_{oss}, 9 Coilcraft and 7 ICE Components discrete inductors modeled approximately by $R_{L,dc}$ and L were used to select implementation of each component. Additionally, the number of components in parallel, switching frequency, and output current were taken as variables for the optimization.

Fig. 4.28 shows the results of the optimization. In Fig. 4.28(a), a genetic algorithm is used to sweep a Pareto front comprised of the efficiency and current density (assessed using the total area of power stage transistors and passives) for each topology. An example of the GA optimization result is shown in Fig. 4.29, overlaid with the proposed method Pareto front. In Fig. 4.28(b), the proposed gradient-based optimization is used to separately select each transistor, inductor, and switching frequency for the series capacitor buck with a switch capacitor step down topology. The optimization examples show the progression of the proposed method after each iteration. In each case, only efficiency is used as the design objective, resulting in higher efficiency but larger footprint designs from the initial conditions. For the examples, the output current and number of transistors and passives in parallel are held constant, resulting in the optimization not fully reaching the Pareto front of all parallel combinations. The transistors and inductors are implemented as single devices, while the series and flying capacitors are implemented as two in parallel. All example optimizations have the same initial conditions except for output current and converge to different transistor and inductor solutions depending on the current loss contribution. By only altering discrete transistor and passive device selection and frequency, the efficiency of the design is improved until paralleling is required for a more efficient design at the operating point.

A prototype using the series capacitor buck with switched capacitor step down topology is fabricated to test the modeling. The prototype uses the components listed in Table 4.4, which were selected from the Pareto front as shown in Fig. 4.28(b) with a selection bias towards high efficiency and high current density designs without paralleled transistors or inductors. The prototype is designed for $I_{out} = 40$ A. Testing results, up to $I_{out} = 10$ A, are shown in Table 4.4(c), and an image of the PCB along with thermal performance at

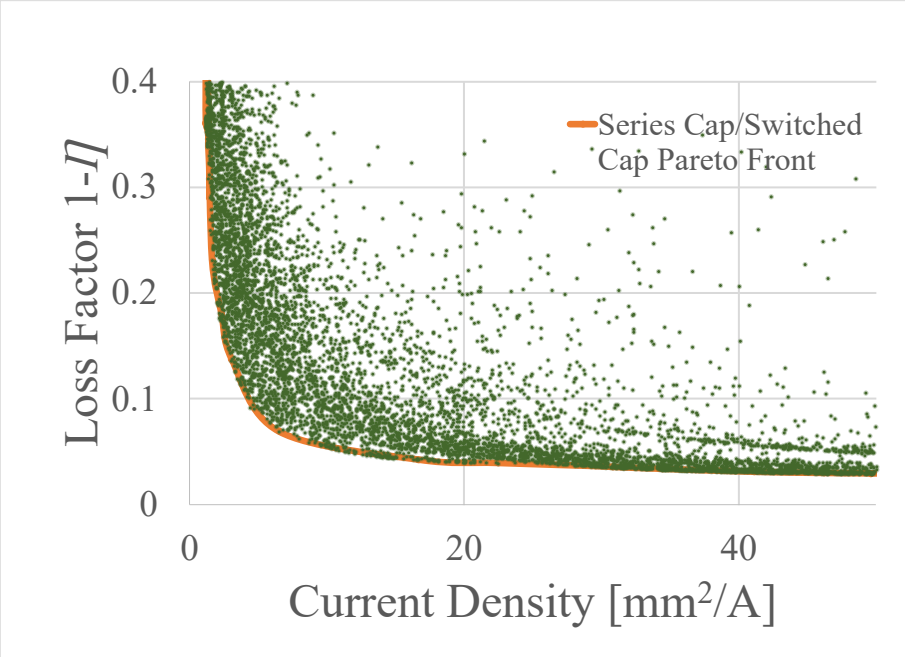


Figure 4.29: Over 9000 different solutions to the series cap / switched cap step down topology solved using a genetic algorithm to determine the Pareto front.

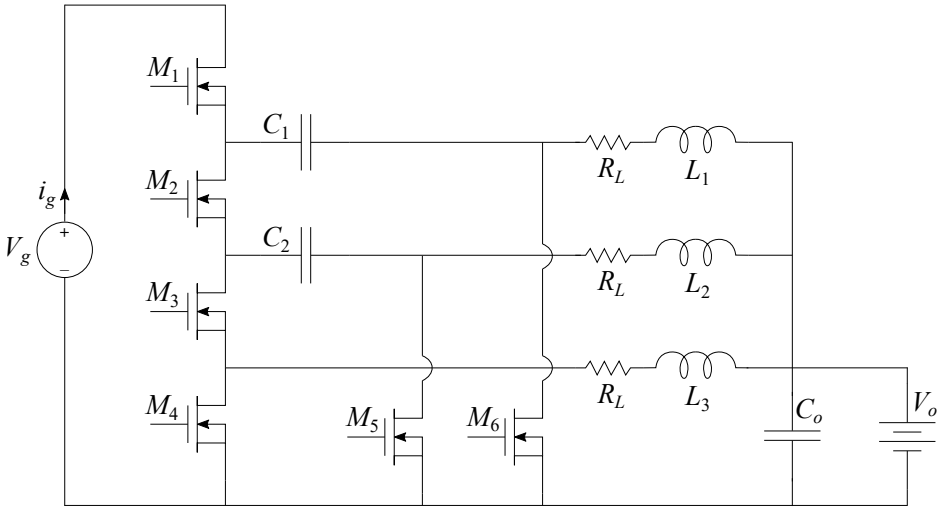


Figure 4.30: 3-phase series capacitor buck schematic.

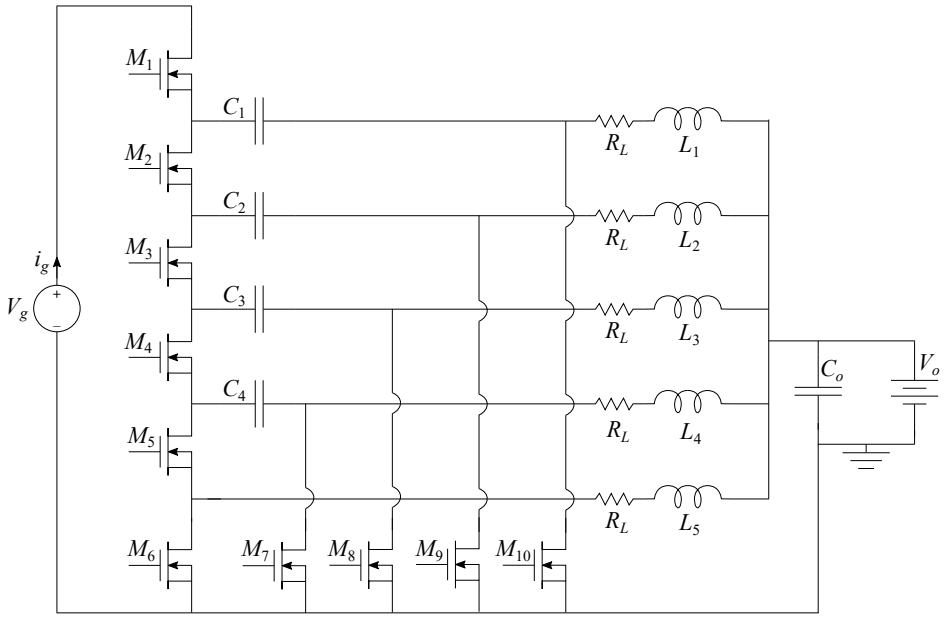


Figure 4.31: 5-Phase series capacitor buck schematic.

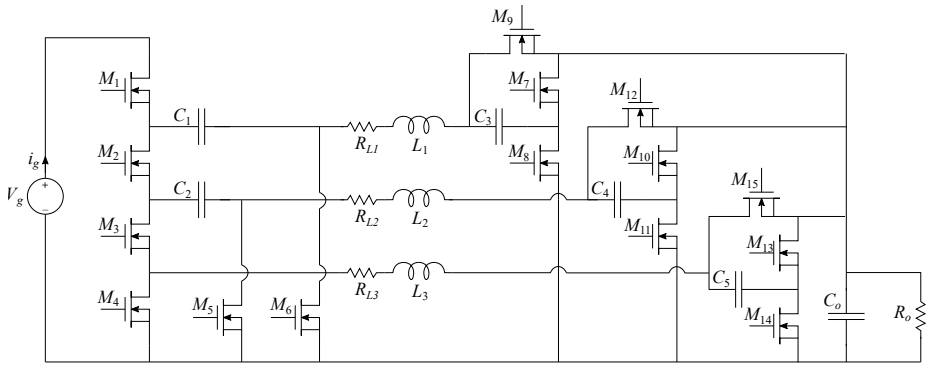


Figure 4.32: 3-phase series capacitor buck with switched-capacitor step down schematic.

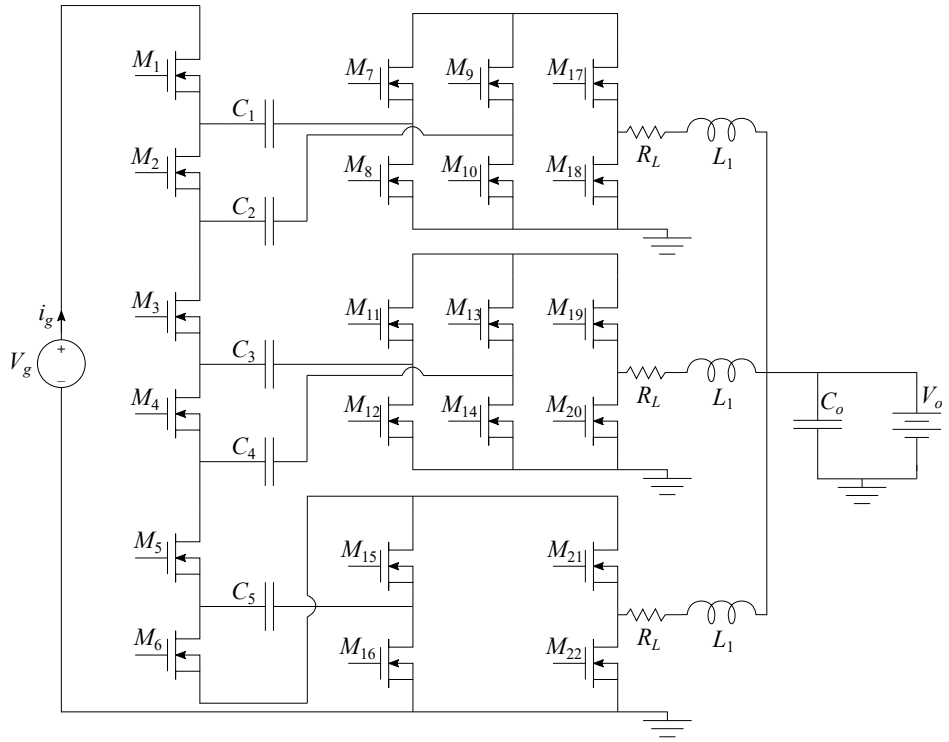


Figure 4.33: LEGO-POL schematic.

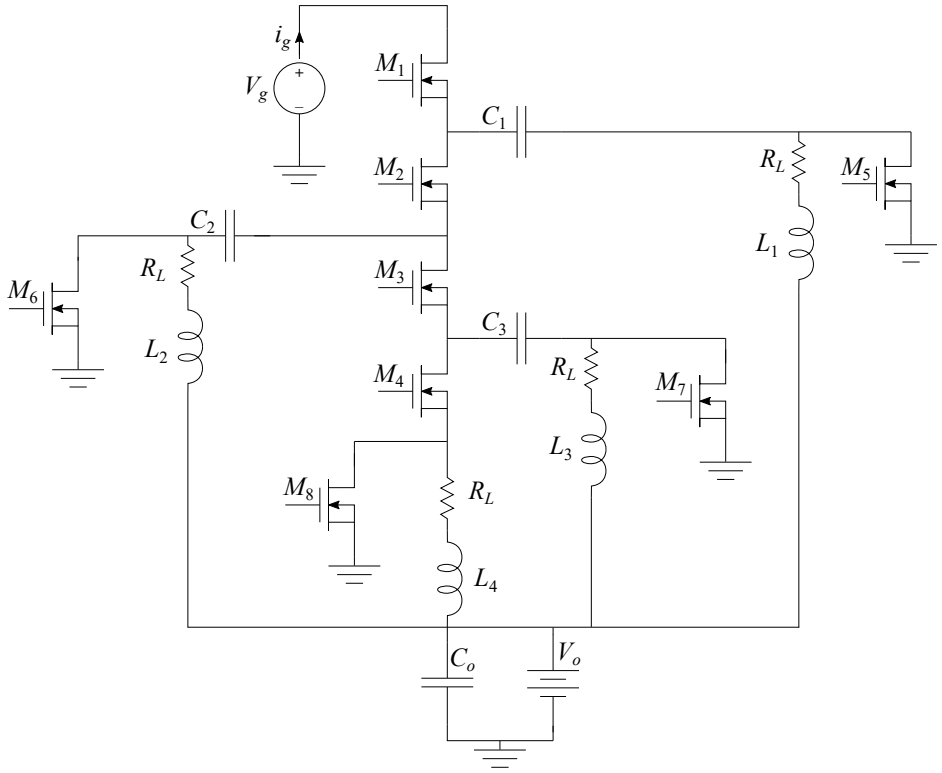


Figure 4.34: Multi-Inductor Hybrid schematic.

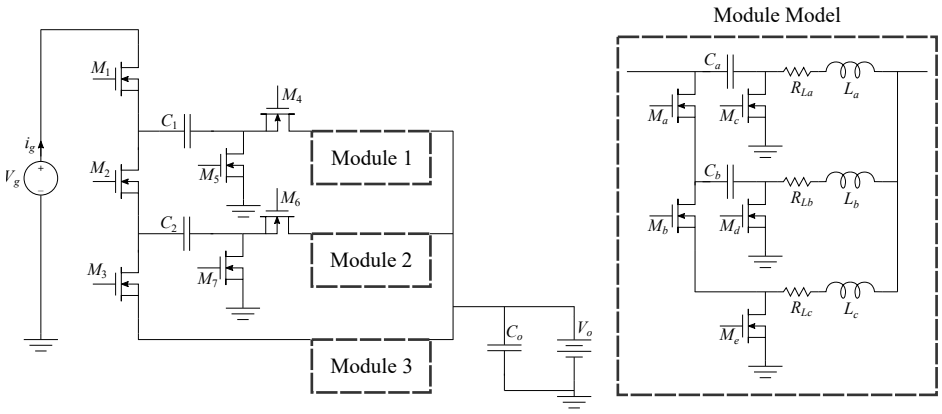


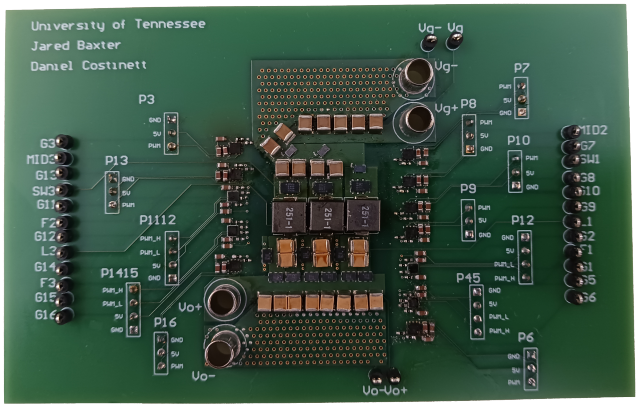
Figure 4.35: Dickson^2 schematic.

10 A are given in Fig. 4.36. From the thermal images and the comparison between modeled and measured efficiency, the prototype appears to exhibit significant inductor ac loss that was not accounted for in the original design or modeled results in Fig. 4.28(c). Core losses calculated using Steinmetz's equation accounted for an approximate additional 0.75 W of ac loss in the inductor, which is included in the core loss modeled results of Fig. 4.28(c). However, there remains an additional 10% of loss unaccounted for in parasitics or other ac losses. From thermal imaging, there appears to be further power loss in the inductors and switching capacitor stage of the topology, which was unaccounted for from the vendor's inductor ac parameters.

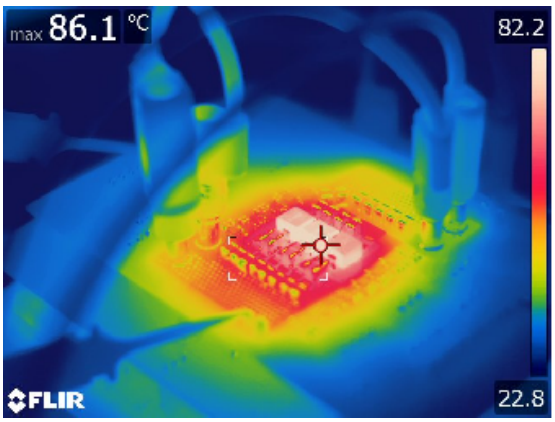
Updated Prototype

In an attempt to alleviate the apparent issues of inductor core and ac loss, an additional inductor was designed. A toroidal iron powder core inductor was constructed to have an increased inductance to confirm the experimental prototype matched modeled projections. The updated inductor parameters are shown in Table 4.5. An updated thermal picture, Fig. 4.37, shows the inductors appear cooler as expected; however, the output terminals of the converter now reach very high temperatures at 20 A output. To once again alleviate a thermal issue, the output terminals of the converter are modified to include three 18AWG wires dipped in solder and placed in the banana jack connection. An updated thermal image at 30 A output current, Fig. 4.39, shows loss primarily occurring in the switch capacitor transistor, which is expected due to the high output current.

To better calculate losses of the inductor, an impedance analyzer was used to find the ac loss at different frequencies as shown in Fig. 4.38. Calculations for ac loss along with core and overlap loss, an efficiency plot of the converter at 0.2 MHz, 0.5 MHz, 1 MHz, and 2 MHz is shown in Fig. 4.40, Fig. 4.41, Fig. 4.42, and Fig. 4.43. respectively. The efficiency results show good matching at lower frequencies. The separation at higher frequencies can be attributed to increased ripple and parasitic losses that are not part of the model and are exaggerated at higher frequencies. Additionally, the increased separation between experimental and modeled efficiencies is expected at higher output currents as parasitic conduction losses rise.



(a)



(b)

Figure 4.36: Prototype PCB photograph (a) and thermal image at $I_{out} = 10$ A (b)

Table 4.5: Toroidal Inductor Parameters

Core Manufacturer	Micrometals
Part Number	T68-2D
A_c	0.179 cm^2
L_m	4.23 cm
Number of Turns	5
Wire Size	18 AWG
L	550 nH
R_L (DC)	6 mΩ

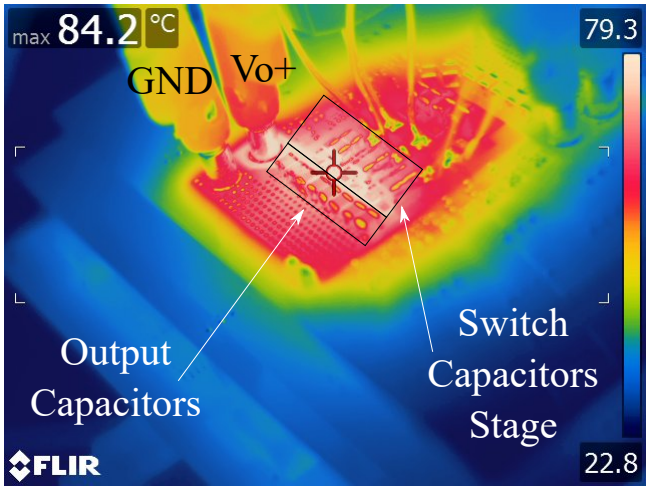


Figure 4.37: Thermal image of series cap/switched cap at 20 A output current with toroid inductors.

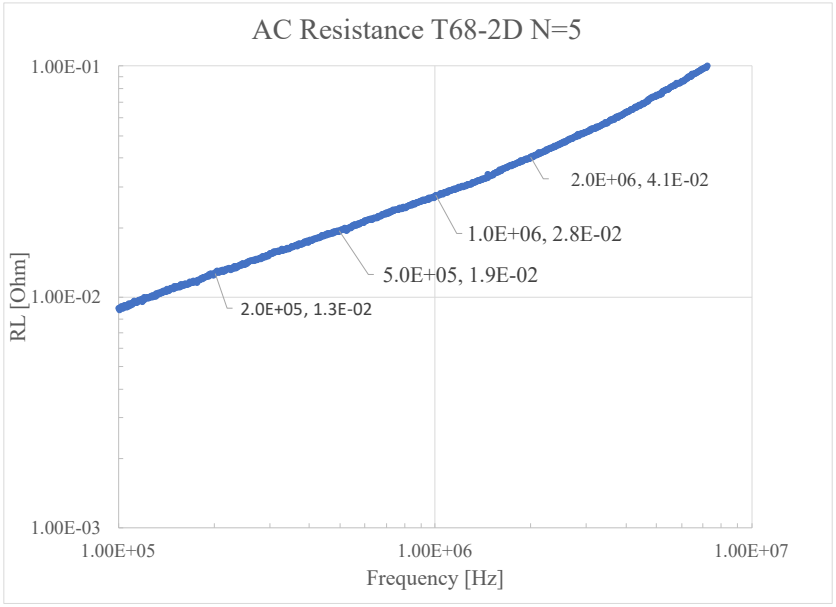
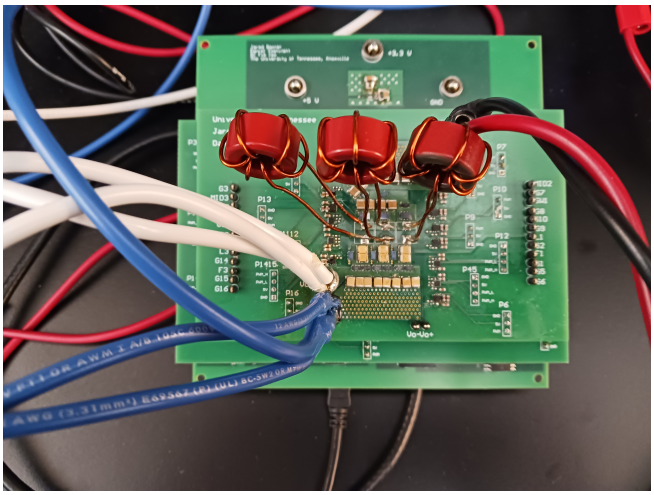
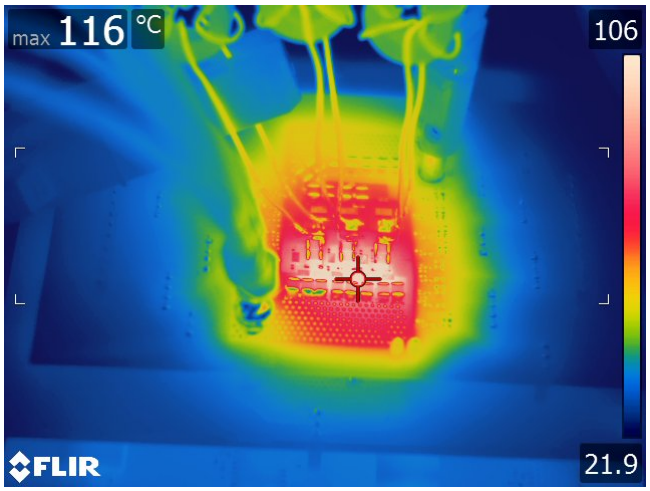


Figure 4.38: AC series resistance of chosen inductor as shown in Table 4.5 using impedance analyzer.



(a)



(b)

Figure 4.39: Prototype PCB photograph (a) and thermal image at $I_{out} = 30$ A (b) using toroid inductors and altered output connection.

A direct comparison of select modeled and experimental waveforms is shown in Fig. 4.44. The ringing due to parasitics and hard switching in Fig. 4.44(a) is the likely cause of increased power loss at higher frequencies trend seen from Fig. 4.40 to Fig. 4.43.

Since the inductor was changed, the results from Fig. 4.28 initally appear to not apply to the final prototype. The Pareto front with the indcutor from Table 4.5 shown in Fig. 4.45 shows the Pareto front has shifted to the right as expected for the large toroidal inductor. All other components and the switching frequency remained constant as the inductor was changed. Therefore, the original optimization is valid for all variables except the inductor since the prototype selection point moves relative to the pareto front as the inductor was updated.

4.4 Conclusions

This chapter discusses methods to take advantage of discrete time state-space modeling in a broad-scale optimization environment. A database has been set up containing discrete components to be optimized with a vast array of tabular and graphical data. Using this data, a gradient based optimization method was used to traverse discrete points to reach optimum converter steady-state. The proposed method projects the best efficiency in a continuous space and then selects the closest discrete component to the direction of steepest descent. Utilizing this method, two optimization example are presented, a battery charger application and a 48-to-1V converter.

In the battery charger application, multiple buck and switch capacitor topology combinations were examined. For the 4V switch capacitor Fibonacci topology, all possible combinations of FETs were brute forced, showcasing the accurate convergence of the optimization method. Additionally, an IC version of the converter was experimentally verified showing the experimental accuracy of the modeling method.

In the 48-to-1V converter, several topologies are assessed along with their respective efficiency and current density relationships. Variables in this example are expanded to include transistor, inductor, capacitor, parallel/series combination of discrete passives and transistors, and switching frequency. Optimization results concurred across GA and the

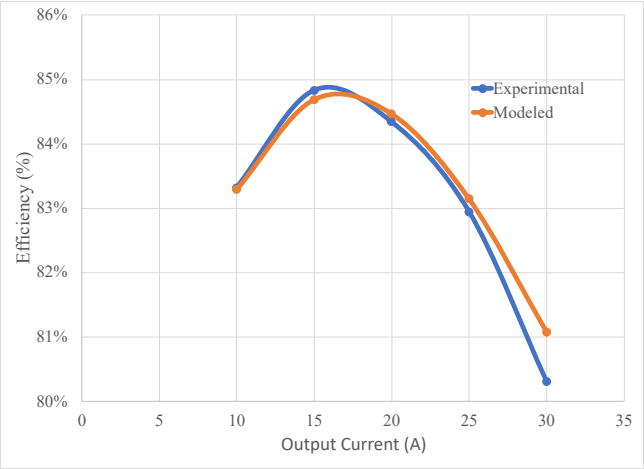


Figure 4.40: Series cap/switched cap at fs = 200 kHz with toroid inductors.

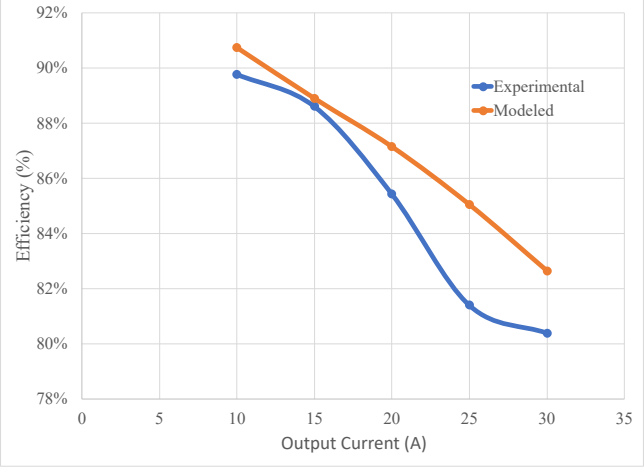


Figure 4.41: Series cap/switched cap at fs = 500 kHz with toroid inductors.

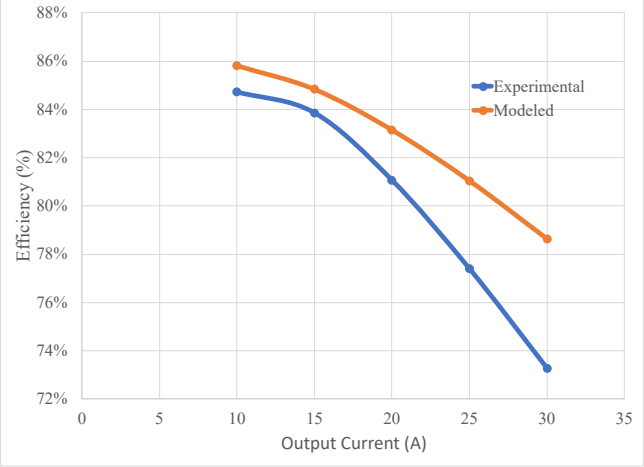


Figure 4.42: Series cap/switched cap at fs = 1 MHz with toroid inductors.

proposed methods, while experimental results initially differed. Discrete inductor ac losses proved to be the major discrepancy between experimental and simulation results, which was alleviated with a hand-wound inductor. Comparison of the Pareto fronts relative to the prototype selection of the chip and toroidal inductor show that the capacitors, transistors and switching frequency were correctly optimized even though the inductor had changed. This shows that even if one variable of the optimization does not have the required information to correctly express experimental results. The trends associated with other variables can still be valid and provide valuable information to the design engineer.

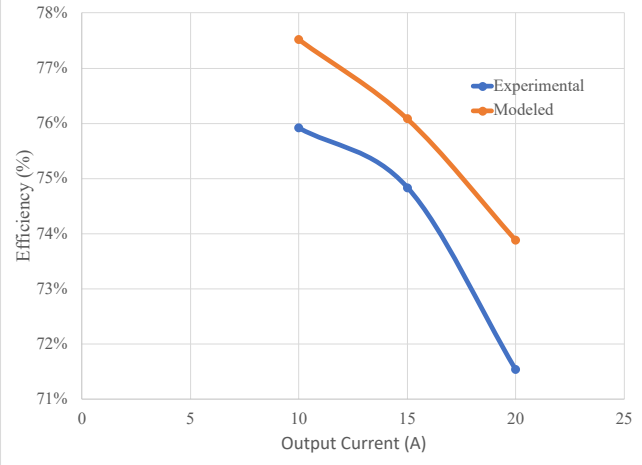
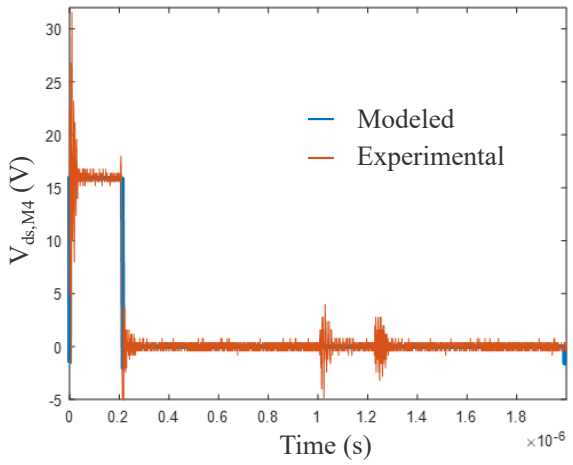
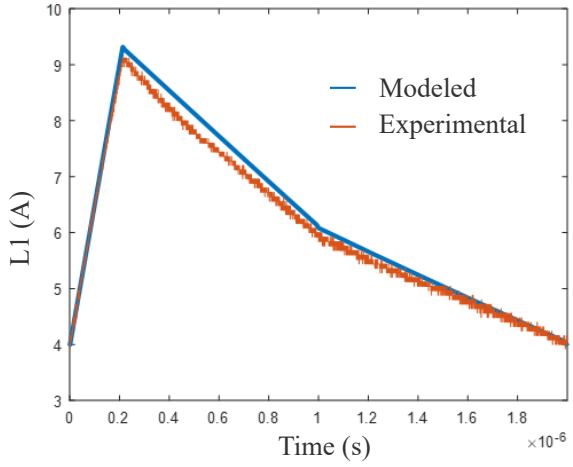


Figure 4.43: Series cap/switched cap at fs = 2 MHz with toroid inductors.



(a)



(b)

Figure 4.44: $V_{ds,M4}$ (a) and L_1 (b) modeled and experimental waveform comparison.

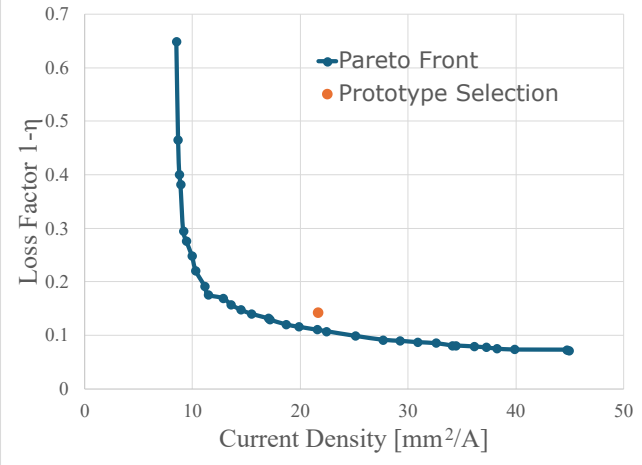


Figure 4.45: Updated Pareto front and prototype selection using inductor from Table 4.5.

Chapter 5

Conclusions

5.1 Summary of Work

This work first investigates discrete time state-space modeling for power electronics design. Background information is first gathered for traditional methods of power converter modeling and optimization. From the background assessment, a gap is found in the methods in which power converters are conventionally modeled. Dedicated converter hand analysis lacks the breadth needed to compare across all converters. Simulation engines are broad enough to consider all converter types; however, they require an extended time to reach steady-state and assess a design.

A framework is developed to rapidly compute the steady-state operation of a converter without the need for dedicated analysis. The approach is general to different classifications of switching converters and scalable to varying levels of model fidelity. Techniques for the inclusion of nonlinearities, such as passive diode commutation, are developed. A simulation comparison is included and shows comparably faster steady-state convergence of test circuits. Experimental verification is performed to confirm model accuracy and simulation results.

Utilizing the rapid modeling approach of discrete time state-space modeling, optimization methods are assessed to rapidly converge to an optimal power converter. A gradient-based approach is chosen, which utilizes a linear projection to optimize discrete transistors. The methodology is supported via a comparison to a brute force method and is experimentally verified.

Additionally, a methodology to quickly retrieve state-space matrices from topology drawings and netlists is developed. The method develops a numerical hybrid matrix that can be quickly calculated for all types of switch-based power supplies.

To fully utilize the effects of discrete time state-space modeling, a design framework is proposed which aims to combine the modeling and optimization techniques already developed, in addition to a database of components and topologies to make rapid assessments within the initial design stage of the switch-based power supply design process.

The final framework provides a database of common topologies, as well as discrete switching and passive devices to be used in optimization. The database includes a uniform structure to be able to handle tabular data such as C_{oss} and R_{on} values given in data sheets as well as graphical data such as C_{oss} vs V_{ds} provided from device manufacturers that can be used to provide a higher level of accuracy to experimental circuits. Additionally, this process allows users to add their own device characteristic or experimental data if needed and parse or sort through these devices for use in optimization. Currently, the framework has a database of over 20 discrete transistors, 20 discrete inductors, and 5 common topologies available to solve power converter design problems.

This work set out to assist switch mode power supply designers with broad-scale initial steady-state converter design by providing a framework to accelerate the design process. The discrete time state-space modeling method is found to be superior for rapid, broad, steady-state, converter-level modeling across all topologies. These features are ideal if a converter model is placed in any optimization routine. However, as demonstrated in the 48-to-1V example, if all the major loss components of a power converter are not accounted for in modeling, optimized solutions can be misleading to potential experimental results. This work shows that while not having all of the major loss components accounted in an experimental design, other discrete elements can still be optimized and used in the next iteration of design.

Detailed design and prototyping are still a required step in the design process. All of the experimental results presented were impacted by parasitics and other loss mechanisms that were not modeled. However, even without detailed modeling, insight can still be gained to narrow the scope of the design.

While this thesis and its works help to create a framework for engineers to work in, it is still up to the engineer to understand and provide the framework with the relevant data to accurately assess and optimize switch mode power supplies.

5.1.1 Publications

Table 5.1 lists several first author publications that are described in this work. Additionally, the Switched Mode Power Supply Toolbox which is based on this framework can be found at https://github.com/costinet/AURA.

5.2 Contributions

The current contributions of the dissertation are described as follows:

- A framework has been developed to rapidly correct state dependent switching within discrete time state-space modeling. The method applies a partial derivative of states with respect to time intervals to correct for 'constant' slope errors and utlizes eigenvalues of the state matrices to determine if additional errors could occur within the switching intervals.
- A framework has been developed to rapidly optimize discrete ordered variables within switch mode power supplies. The method applies a partial derivative of variables with respect to the power loss of the circuit.
- A publicly available switch mode power supply design framework capable of broad scale design optimization and modeling across topology, device selection, and control variables at steady-state operation.

Table 5.1: Author's Contributions to the Field

Year	Title	Publisher	Conference
2019	Converter Analysis Using Discrete Time State-Space Modeling	IEEE	2019 20th Workshop on Control and Modeling for Power Electronics (COMPEL)
2020	Steady-State Convergence of Discrete Time State-Space Modeling with State-Dependent Switching	IEEE	2020 IEEE 21st Workshop on Control and Modeling for Power Electronics (COMPEL)
2022	Broad-Scale Converter Optimization Utilizing Discrete Time State-Space Modeling	IEEE	2022 IEEE Design Methodologies Conference (DMC)
2023	Power Converter and Discrete Device Optimization Utilizing Discrete Time State-Space Modeling	IEEE	2023 IEEE 24th Workshop on Control and Modeling for Power Electronics (COMPEL)

Bibliography

[1] L. M. Tolbert, T. J. King, B. Ozpineci, J. B. Campbell, G. Muralidharan, D. T. Rizy, A. S. Sabau, H. Zhang, W. Zhang, X. Yu, H. F. Huq, and H. Liu, "Power electronics for distributed energy systems and transmission and distribution applications," Oak Ridge National Laboratory, Report ORNL/TM-2005/230, 2005. 1

[2] "Power electronics market - global industry analysis, size, share, growth, trends, regional outlook, and forecast 2021 - 2030," Precedence Research, Report, 2021. 1, 34

[3] "Power electronics market - growth, trends, covid-19 impact, and forecasts (2022 - 2027)," Mordor Intelligence, Report, 2021. 1

[4] "Power electronics for electric vehicle market growing stongly," *EE-Evaluation Engineering*, vol. 59, no. 7, p. 7, 2020/08// 2020. [Online]. Available: https://link.gale.com/apps/doc/A634381073/ITBC?u=tel_a_utl&sid=bookmark-ITBC&xid=e687409b 1

[5] "Electrical and electronics technical team roadmap," U. S. Drive, Report, 2017. 1

[6] E. A. Jones, F. F. Wang, and D. Costinett, "Review of commercial gan power devices and gan-based converter design challenges," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 4, no. 3, pp. 707–719, 2016. 1

[7] A. Andreta, L. F. Lavado Villa, Y. Lembeye, and J. C. Crebier, "A novel automated design methodology for power electronics converters," *Electronics*, vol. 10, no. 3, 2021. 2

[8] D. Maksimovic, "Automated steady-state analysis of switching power converters using a general-purpose simulation tool," in *PESC97. Record 28th Annual IEEE Power Electronics Specialists Conference. Formerly Power Conditioning Specialists Conference 1970-71. Power Processing and Electronic Specialists Conference 1972*, vol. 2, 1997, pp. 1352–1358 vol.2. 2, 15, 20

[9] C. Zhao, B. Trento, L. Jiang, E. A. Jones, B. Liu, Z. Zhang, D. Costinett, F. F. Wang, L. M. Tolbert, J. F. Jansen, R. Kress, and R. Langley, "Design and implementation of

a gan-based, 100-khz, 102-w/in¡sup¿3¡/sup¿ single-phase inverter," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 4, no. 3, pp. 824–840, 2016. 2

[10] T. M. Evans, Q. Le, S. Mukherjee, I. A. Razi, T. Vrotsos, Y. Peng, and A. H. Mantooth, "Powersynth: A power module layout generation tool," *IEEE Transactions on Power Electronics*, pp. 1–1, 2018. 4

[11] T. Wu, B. Ozpineci, M. Chinthavali, W. Zhiqiang, S. Debnath, and S. Campbell, "Design and optimization of 3d printed air-cooled heat sinks based on genetic algorithms," in *2017 IEEE Transportation Electrification Conference and Expo (ITEC)*, 2017, pp. 650–655. 4, 21, 24

[12] A. Bindra and A. Mantooth, "Modern tool limitations in design automation: Advancing automation in design tools is gathering momentum," *IEEE Power Electronics Magazine*, vol. 6, no. 1, pp. 28–33, 2019. 4

[13] J. W. Kolar, "Power electronics design 4.0," *IEEE Design Automation for Power Electronics Workshop (DAPE)*, 2018. 4

[14] R. S. Graham, "Relay computer for network analysis," *Bell Laboratories Record*, vol. 31, no. 4, pp. 152–157, 1953. 7

[15] D. Pederson, "A historical review of circuit simulation," *IEEE Transactions on Circuits and Systems*, vol. 31, no. 1, pp. 103–111, 1984. 7

[16] G. Hachtel, R. Brayton, and F. Gustavson, "The sparse tableau approach to network analysis and design," *IEEE Transactions on Circuit Theory*, vol. 18, no. 1, pp. 101–113, 1971. 8

[17] W. Weeks, A. Jimenez, G. Mahoney, D. Mehta, H. Qassemzadeh, and T. Scott, "Algorithms for astap–a network-analysis program," *IEEE Transactions on Circuit Theory*, vol. 20, no. 6, pp. 628–634, 1973. 8

[18] L. Nagel and R. Rohrer, "Computer analysis of nonlinear circuits, excluding radiation (cancer)," *IEEE Journal of Solid-State Circuits*, vol. 6, no. 4, pp. 166–182, 1971. 8

[19] L. W. Nagel and D. Pederson, "Spice (simulation program with integrated circuit emphasis)," EECS Department, University of California, Berkeley, Report UCB/ERL M382, 1973. 8

[20] L. W. Nagel, "Spice2: A computer program to simulate semiconductor circuits," Ph.D. dissertation, EECS Department, University of California, Berkeley, 1975. 8, 37

[21] R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli, "A survey of optimization techniques for integrated-circuit design," *Proceedings of the IEEE*, vol. 69, no. 10, pp. 1334–1362, 1981. 8

[22] Y. Yu, F. C. Y. Lee, and J. E. Triner, "Power converter design optimization," in *1977 IEEE Power Electronics Specialists Conference*, 1977, pp. 104–112. 8

[23] Y. Yu, F. C. y. Lee, and J. E. Triner, "Power converter design optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-15, no. 3, pp. 344–355, 1979. 8

[24] S. Balachandran and F. C. Y. Lee, "Algorithms for power converter design optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-17, no. 3, pp. 422–432, 1981. 8

[25] S. Rahman and F. C. Lee, "Nonlinear program based optimization of boost and buck-boost converter designs," in *1981 IEEE Power Electronics Specialists Conference*, 1981, pp. 180–191. 8

[26] S. Rahman and F. C. Lee, "Computer simulations of optimum boost and buck-boost converters," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-18, no. 5, pp. 598–608, 1982. 8, 9

[27] C. J. Wu, F. C. Lee, S. Balachandran, and H. L. Goin, "Design optimization for a half-bridge dc-dc converter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-18, no. 4, pp. 497–508, 1982. 8, 9

[28] R. B. Ridley and F. C. Lee, "Practical nonlinear design optimization tool for power converter components," in *1987 IEEE Power Electronics Specialists Conference*, 1987, pp. 314–323. 8

[29] R. B. Ridley, C. Zhou, and F. C. Y. Lee, "Application of nonlinear design optimization for power converter components," *IEEE Transactions on Power Electronics*, vol. 5, no. 1, pp. 29–40, 1990. 8

[30] F. C. Lee, S. Rahman, R. A. Carter, C. H. Wu, Y. Yu, and R. Chang, "Modeling and analysis of power processing systems (mapps)," Report NASA-CR-165538, TRW-32660-6001-RU-01, 1980. 8

[31] H. Chung-Wen, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Transactions on Circuits and Systems*, vol. 22, no. 6, pp. 504–509, 1975. 10

[32] "Spice topics," eCircuit Center, 2003. [Online]. Available: http://www.ecircuitcenter.com/SPICEtopics.htm 10

[33] "Plecs user manual," Plexim, Tech. Rep., 2016. [Online]. Available: https://www.plexim.com/files/plecsmanual.pdf 14, 37

[34] J. A. Baxter and D. J. Costinett, "Steady-state convergence of discrete time state-space modeling with state-dependent switching," in *2020 IEEE 21st Workshop on Control and Modeling for Power Electronics (COMPEL)*, 2020, pp. 1–8. 17, 65

[35] Synopsys, "Saber user guide," 2004. 20

[36] "Spectre circuit simulator user guide," Tech. Rep., 2004. [Online]. Available: https://www.ee.columbia.edu/~harish/uploads/2/6/9/2/26925901/spectre_reference.pdf 20

[37] D. Maksimovic, A. M. Stankovic, V. J. Thottuvelil, and G. C. Verghese, "Modeling and simulation of power electronic converters," *Proceedings of the IEEE*, vol. 89, no. 6, pp. 898–912, 2001. 20

[38] L. Duwang and R. Tymerski, "A comparison of steady state methods for power electronic circuits," in *PESC 98 Record. 29th Annual IEEE Power Electronics Specialists Conference (Cat. No.98CH36196)*, vol. 2, 1998, pp. 1084–1090 vol.2. 20

[39] S. Skelboe, "Computation of the periodic steady-state response of nonlinear networks by extrapolation methods," *IEEE Transactions on Circuits and Systems*, vol. 27, no. 3, pp. 161–175, 1980. 20

[40] L. Duwang and R. Tymerski, "Comparison of simulation algorithms for accelerated determination of periodic steady state of switched networks," *IEEE Transactions on Industrial Electronics*, vol. 47, no. 6, pp. 1278–1285, 2000. 20

[41] T. Kato and W. Tachibana, "Periodic steady-state analysis of an autonomous power electronic system by a modified shooting method," *IEEE Transactions on Power Electronics*, vol. 13, no. 3, pp. 522–527, 1998. 21

[42] H. B. Shin, J. G. Park, S. K. Chung, H. W. Lee, and T. A. Lipo, "Generalised steady-state analysis of multiphase interleaved boost converter with coupled inductors," *IEE Proceedings - Electric Power Applications*, vol. 152, no. 3, pp. 584–594, 2005. 21

[43] M. D. Seeman and S. R. Sanders, "Analysis and optimization of switched-capacitor dc-dc converters," *IEEE Transactions on Power Electronics*, vol. 23, no. 2, pp. 841–851, 2008. 21

[44] Y. Ji, A. Pfeiffer, and J. Bals, "Optimization based steady-state analysis of switched power electronic systems," in *2010 IEEE 12th Workshop on Control and Modeling for Power Electronics (COMPEL)*, 2010, pp. 1–6. 21

[45] R. Trinchero, I. S. Stievano, and F. G. Canavero, "Steady-state analysis of switching power converters via augmented time-invariant equivalents," *IEEE Transactions on Power Electronics*, vol. 29, no. 11, pp. 5657–5661, 2014. 21

[46] R. M. Burkart, "Advanced modeling and multi-objective optimization of power electronic converter systems," Ph.D. dissertation, ETH Zurich, 2016. 21, 22

[47] D. V. Malyna, J. L. Duarte, M. A. M. Hendrix, and F. B. M. v. Horck, "Multi-objective optimization of power converters using genetic algorithms," in *International Symposium on Power Electronics, Electrical Drives, Automation and Motion, 2006. SPEEDAM 2006.*, 2006, pp. 713–717. 21, 24

[48] Y. Panov, Y. Jang, M. M. Jovanović, and B. T. Irving, "Design optimization and performance evaluation of high-power, high-frequency, bidirectional buck-boost converter with sic mosfets," in *2017 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2017, pp. 259–266. 21

[49] M. Mirjafari and R. S. Balog, "Survey of modelling techniques used in optimisation of power electronic components," *IET Power Electronics*, vol. 7, no. 5, pp. 1192–1203, 2014. 22

[50] K. Rigbers, S. Schroder, T. Durbaum, M. Wendt, and R. W. D. Doncker, "Integrated method for optimization of power electronic circuits," in *2004 IEEE 35th Annual Power Electronics Specialists Conference (IEEE Cat. No.04CH37551)*, vol. 6, 2004, pp. 4473–4478 Vol.6. 22

[51] T. M. Andersen, C. M. Zingerli, F. Krismer, J. W. Kolar, N. Wang, and C. Mathuna, "Modeling and pareto optimization of microfabricated inductors for power supply on chip," *IEEE Transactions on Power Electronics*, vol. 28, no. 9, pp. 4422–4430, 2013. 22

[52] I. D. Laird, J. Scoltock, A. J. Forsyth, and X. Yuan, "A unified framework for computationally efficient power converter design optimisation," in *8th IET International Conference on Power Electronics, Machines and Drives (PEMD 2016)*, 2016, pp. 1–6. 22

[53] R. Yu, B. M. H. Pong, B. W. K. Ling, and J. Lam, "Two-stage optimization method for efficient power converter design including light load operation," *IEEE Transactions on Power Electronics*, vol. 27, no. 3, pp. 1327–1337, 2012. 22

[54] G. Stahl, M. Rodriguez, and D. Maksimovic, "A high-efficiency bidirectional buck-boost dc-dc converter," in *2012 Twenty-Seventh Annual IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2012, pp. 1362–1367. 22

[55] C. A. C. Coello, "Evolutionary multi-objective optimization: a historical view of the field," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 28–36, 2006. 23

[56] K. Deb, *Current trends in evolutionary multi-objective optimization*, 2007, vol. 1. 23

[57] V. Christophe, D. Olivier, and L. Jacques, "A computer-aided design tool dedicated to isolated dc-dc converters based on multiobjective optimization using genetic algorithms," *COMPEL - The international journal for computation and mathematics in electrical and electronic engineering*, vol. 31, no. 2, pp. 583–603, 2012. 24, 26

[58] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002. 24

[59] C. Pohl, L. Zacharias, M. Bodach, S. Slawinski, and T. Barucki, "An open framework for algorithm based multi criteria optimization of power electronic systems," in *Proceedings of PCIM Europe 2015; International Exhibition and Conference for Power Electronics, Intelligent Motion, Renewable Energy and Energy Management*, 2015, pp. 1–8. 24

[60] S. Busquets-monge, G. Soremekun, E. Hefiz, C. Crebier, S. Ragon, D. Boroyevich, Z. Gurdal, M. Arpilliere, and D. K. Lindner, "Power converter design optimization," *IEEE Industry Applications Magazine*, vol. 10, no. 1, pp. 32–38, 2004. 24

[61] F. Wang, G. Chen, D. Boroyevich, S. Ragon, M. Arpilliere, and V. R. Stefanovic, "Analysis and design optimization of diode front-end rectifier passive components for voltage source inverters," *IEEE Transactions on Power Electronics*, vol. 23, no. 5, pp. 2278–2289, 2008. 24

[62] T. Wu, B. Ozpineci, and C. Ayers, "Genetic algorithm design of a 3d printed heat sink," in *2016 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2016, pp. 3529–3536. 24

[63] R. R. Chan, S. D. Sudhoff, Y. Lee, and E. L. Zivi, "Evolutionary optimization of power electronics based power systems," in *APEC 07 - Twenty-Second Annual IEEE Applied Power Electronics Conference and Exposition*, 2007, pp. 449–456. 24

[64] H. P. Ren and T. Zheng, "Optimization design of power factor correction converter based on genetic algorithm," in *2010 Fourth International Conference on Genetic and Evolutionary Computing*, 2010, pp. 293–296. 24

[65] R. Pagano, S. Abedinpour, A. Raciti, and S. Musumeci, "Efficiency optimization of an integrated wireless power transfer system by a genetic algorithm," in *2016 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2016, pp. 3669–3676. 24

[66] K. F. Man, K. S. Tang, and S. Kwong, "Genetic algorithms: concepts and applications [in engineering design]," *IEEE Transactions on Industrial Electronics*, vol. 43, no. 5, pp. 519–534, 1996. 24

[67] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011. 24

[68] Z. H. Zhan and J. Zhang, "Differential evolution for power electronic circuit optimization," in *2015 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, 2015, pp. 158–163. 24

[69] S. E. D. León-Aldaco, H. Calleja, and J. A. Alquicira, "Metaheuristic optimization methods applied to power converters: A review," *IEEE Transactions on Power Electronics*, vol. 30, no. 12, pp. 6791–6803, 2015. 25, 29

[70] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 25–34, 1987. 25

[71] B. Biswal, P. K. Dash, and B. K. Panigrahi, "Power quality disturbance classification using fuzzy c-means algorithm and adaptive particle swarm optimization," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 1, pp. 212–220, 2009. 25

[72] Z. h. Zhan and J. Zhang, "Orthogonal learning particle swarm optimization for power electronic circuit optimization with free search range," in *2011 IEEE Congress of Evolutionary Computation (CEC)*, 2011, pp. 2563–2570. 25

[73] K. Ishaque, Z. Salam, M. Amjad, and S. Mekhilef, "An improved particle swarm optimization (pso)-based mppt for pv with reduced steady-state oscillation," *IEEE Transactions on Power Electronics*, vol. 27, no. 8, pp. 3627–3638, 2012. 25

[74] C. Feeney and N. Wang, "A new electronic design automation tool for the optimization of pwrsoc/pwrsip dc-dc converters," in *2018 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2018, pp. 2905–2909. 25

[75] V. Selvi and D. R. Umarani, "Comparative analysis of ant colony and particle swarm optimization techniques," *International Journal of Computer Applications*, vol. 5, no. 4, 2010. 26

[76] A. K. Tiwari and S. P. Dubey, "Ant colony optimization based hybrid active power filter for harmonic compensation," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 2016, pp. 777–782. 26

[77] C. F. Juang, C. M. Lu, C. Lo, and C. Y. Wang, "Ant colony optimization algorithm for fuzzy controller design and its fpga implementation," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 3, pp. 1453–1462, 2008. 26

[78] A. F. Zobaa and A. Lecci, "Ant colony optimization of controller parameters for power converters," in *8th International Conference on Advances in Power System Control, Operation and Management (APSCOM 2009)*, 2009, pp. 1–5. 26

[79] J. Zhang, H. S. H. Chung, A. W. L. Lo, and T. Huang, "Extended ant colony optimization algorithm for power electronic circuit design," *IEEE Transactions on Power Electronics*, vol. 24, no. 1, pp. 147–162, 2009. 26

[80] C. Shi, S. Yifei, C. Junfeng, Q. Quande, C. Xianghua, L. Xiujuan, and S. Yuhui, "A comprehensive survey of brain storm optimization algorithms," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 1637–1644. 26

[81] S. Busquets-Monge, C. Crebier, S. Ragon, E. Hertz, J. Wei, J. Zhang, D. Boroyevich, Z. Gurda, P. K. Lindner, and A. Arpilliere, "Optimization techniques applied to the design of a boost power factor correction converter," in *2001 IEEE 32nd Annual Power Electronics Specialists Conference (IEEE Cat. No.01CH37230)*, vol. 2, 2001, pp. 920–925 vol.2. 28

[82] S. Busquets-Monge, J. C. Crebier, S. Ragon, E. Hertz, D. Boroyevich, Z. Gurdal, M. Arpilliere, and D. K. Lindner, "Design of a boost power factor correction converter using optimization techniques," *IEEE Transactions on Power Electronics*, vol. 19, no. 6, pp. 1388–1396, 2004. 28

[83] M. Delhommais, G. Dadanema, Y. Avenas, F. Costa, J. L. Schanen, and C. Vollaire, "Design by optimization of power electronics converter including emc constraints," in *2016 International Symposium on Electromagnetic Compatibility - EMC EUROPE*, 2016, pp. 182–187. 28

[84] M. Delhommais, G. Dadanema, Y. Avenas, J. L. Schanen, F. Costa, and C. Vollaire, "Using design by optimization for reducing the weight of a sic switching cell," in *2016 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2016, pp. 1–8. 28

[85] A. Bolopion, J. Fandino, L. Gerbaud, and E. Atienza, "An optimisation process of power electronics applications," *COMPEL - The international journal for computation and mathematics in electrical and electronic engineering*, vol. 20, no. 3, pp. 879–890, 2001. 28

[86] C. Larouci, M. Boukhnifer, and A. Chaibet, "Design of power converters by optimization under multiphysic constraints: Application to a two-time-scale ac/dc-dc converter," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, pp. 3746–3753, 2010. 28

[87] C. Gezgin, B. S. Heck, and R. M. Bass, "Simultaneous design of power stage and controller for switching power supplies," *IEEE Transactions on Power Electronics*, vol. 12, no. 3, pp. 558–566, 1997. 28

[88] M. Delhommais, J. L. Schanen, F. Wurtz, C. Rigaud, and S. Chardon, "First order design by optimization method: Application to an interleaved buck converter and validation," in *2018 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2018, pp. 944–951. 28

[89] S. Pam, Y. Satija, P. Chawda, M. Mansour, R. Hanrahan, and J. Perry, "A web-based tool for compensation design of power converters using hybrid optimization," in *2016 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2016, pp. 3266–3272. 29

[90] B. K. Bose, "Neural network applications in power electronics and motor drives-an introduction and perspective," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 1, pp. 14–33, 2007. 29

[91] F. Filho, H. Z. Maia, T. H. A. Mateus, B. Ozpineci, L. M. Tolbert, and J. O. P. Pinto, "Adaptive selective harmonic minimization based on anns for cascade multilevel inverters with varying dc sources," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 5, pp. 1955–1962, 2013. 29

[92] E. Bond, J. Wang, W. G. Dunford, and K. Mauch, "A simple spreadsheet-based converter performance model reduces analysis and design time," in *Industry Applications Conference, 1996. Thirty-First IAS Annual Meeting, IAS '96., Conference Record of the 1996 IEEE*, vol. 2, 1996, pp. 891–897 vol.2. 29

[93] D. W. Bouler, J. Baxter, and D. Costinett, "Optimization of gan-based ultra-low power boost converter in far-field energy harvesting," in *2016 IEEE PELS Workshop on Emerging Technologies: Wireless Power Transfer (WoW)*, 2016, pp. 231–237. 29

[94] G. C. Verghese, M. E. Elbuluk, and J. G. Kassakian, "A general approach to sampled-data modeling for power electronic circuits," *IEEE Transactions on Power Electronics*, vol. PE-1, no. 2, pp. 76–89, 1986. 32

[95] H. R. Visser and P. P. J. v. d. Bosch, "Modelling of periodically switching networks," in *PESC '91 Record 22nd Annual IEEE Power Electronics Specialists Conference*, 1991, pp. 67–73. 32, 34, 35, 36, 73

[96] F. C. Y. Lee, R. P. Iwens, Y. Yu, and J. E. Triner, "Generalized computer-aided discrete time-domain modeling and analysis of dc-dc converters," *IEEE Transactions on Industrial Electronics and Control Instrumentation*, vol. IECI-26, no. 2, pp. 58–69, 1979. 32

[97] A. Kumar, J. Lu, and K. K. Afridi, "Enhanced-accuracy augmented state-space approach to steady-state modeling of resonant converters," in *2015 IEEE 16th Workshop on Control and Modeling for Power Electronics (COMPEL)*, 2015, pp. 1–6. 32, 34

[98] L. O. Chua and P.-M. Lin, *Computer-aided analysis of electronic circuits : algorithms and computational techniques*, ser. Prentice-Hall series in electrical and computer engineering. Englewood Cliffs, N.J: Prentice-Hall, 1975. 33, 129, 132

[99] D. Costinett, R. Zane, and D. Maksimovic, "Discrete time modeling of output disturbances in the dual active bridge converter," in *2014 IEEE Applied Power Electronics Conference and Exposition - APEC 2014*, 2014, pp. 1171–1177. 34

[100] C. V. Loan, "Computing integrals involving the matrix exponential," *IEEE Transactions on Automatic Control*, vol. 23, no. 3, pp. 395–404, 1978. 34

[101] J. A. Baxter and D. J. Costinett, "Converter analysis using discrete time state-space modeling," in *2019 20th Workshop on Control and Modeling for Power Electronics (COMPEL)*, 2019, pp. 1–8. 50, 54

[102] J. Celikovic, R. Das, H. P. Le, and D. Maksimovic, "Modeling of capacitor voltage imbalance in flying capacitor multilevel dc-dc converters," in *2019 20th Workshop on Control and Modeling for Power Electronics (COMPEL)*, 2019, pp. 1–8. 57

[103] Z. Xia, B. L. Dobbins, and J. T. Stauth, "Natural balancing of flying capacitor multilevel converters at nominal conversion ratios," in *2019 20th Workshop on Control and Modeling for Power Electronics (COMPEL)*, 2019, pp. 1–8. 57

[104] J. A. Baxter and D. J. Costinett, "Power converter and discrete device optimization utilizing discrete time state-space modeling," in *2023 IEEE 24th Workshop on Control and Modeling for Power Electronics (COMPEL)*, 2023, pp. 1–8. 57

[105] J. A. Baxter and D. J. Costinett, "Broad-scale converter optimization utilizing discrete time state-space modeling," in *2022 IEEE Design Methodologies Conference (DMC)*, 2022, pp. 1–6. 65

[106] P. S. Shenoy, M. Amaro, J. Morroni, and D. Freeman, "Comparison of a buck converter and a series capacitor buck converter for high-frequency, high-conversion-ratio voltage regulators," *IEEE Transactions on Power Electronics*, vol. 31, no. 10, pp. 7006–7015, 2016. 87

[107] G. Gabian, J. Gamble, B. Blalock, and D. Costinett, "Hybrid buck converter optimization and comparison for smart phone integrated battery chargers," in *2018 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2018, pp. 2148–2154. 93

[108] J. Baek, Y. Elasser, K. Radhakrishnan, H. Gan, J. P. Douglas, H. K. Krishnamurthy, X. Li, S. Jiang, C. R. Sullivan, and M. Chen, "Vertical stacked lego-pol cpu voltage regulator," *IEEE Transactions on Power Electronics*, vol. 37, no. 6, pp. 6305–6322, 2022. 93

[109] R. Das and H.-P. Le, "A regulated 48v-to-1v/100a 90.9converter for pol applications in data centers and telecommunication systems," in *2019 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2019, pp. 1997–2001. 93

[110] Y. Zhu, T. Ge, Z. Ye, and R. C. Pilawa-Podgurski, "A dickson-squared hybrid switched-capacitor converter for direct 48 v to point-of-load conversion," in *2022 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2022, pp. 1272–1278. 93

[111] Fruko and P. A. Knight, *First Course in Network Theory*. Oxford: Oxford University Press, 2015. 129

Appendices

A Circuit Parsing

To assist in making the discrete state-space steady-state modeling, a circuit parser was created in MATLAB to create the set of state-space and output equations using an LTspice netlist file. Typically forming state-space equations is done by hand calculations utilizing the passive components within the converter and utilizing their relationships to voltage and current, namely $v_L = L\frac{di_L}{dt}$ and $i_C = L\frac{dv_C}{dt}$. The following goes through an example of the hand calculation method with a buck converter shown Figure 1 . The current through the capacitors and voltage across the inductors is found in relation to state variables (capacitor voltages and inductor currents) and constants (DC voltage and current sources)

$$i_C(t) = i_L(t) - \frac{v_C(t)}{R_o} \tag{1}$$

$$v_L(t) = V_g - i_L(t)R_l - v_C(t). \tag{2}$$

Placing these equations into matrix form

$$\begin{bmatrix} i_C(t) \\ v_L(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{R_o} & 1 \\ -1 & R_l \end{bmatrix} \begin{bmatrix} v_C(t) \\ i_L(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} V_g \end{bmatrix}, \tag{3}$$

then dividing by the inductances and capacitances

$$\begin{bmatrix} C & 0 \\ 0 & L \end{bmatrix} \begin{bmatrix} \dot{v_C}(t) \\ \dot{i_L}(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{R_o} & 1 \\ -1 & R_l \end{bmatrix} \begin{bmatrix} v_C(t) \\ i_L(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} V_g \end{bmatrix} \tag{4}$$

results in the state-space description of the circuit

$$\begin{bmatrix} \dot{v_C}(t) \\ \dot{i_L}(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{R_o \times C} & \frac{1}{C} \\ \frac{-1}{L} & \frac{R_l}{L} \end{bmatrix} \begin{bmatrix} v_C(t) \\ i_L(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} V_g \end{bmatrix} \tag{5}$$

This chapter shows an automated method to create these state-space equations utilizing network theory. First, network theory principles and definitions are reviewed. These

principles are then applied to matrices that will be used in solving for the state-space. Then a buck converter is used as an example to explain the automated circuit parser.

A.1 Network Theory

A network, also known as graph, is a set of objects in which some pairs of the objects are related [111]. A graph comprises of a set of vertices (also called nodes) a set of edges (also called branches). In general, there are two types of graphs undirected and directed. Directed graphs are represented by an ordered pair of vertices where each edge has direction between the nodes, often represented by an arrow as shown in Figure 2. Undirected graphs show a connection between vertices through edges that have no direction.

Terminology and Definitions

In order to fully elaborate how network theory intertwines with circuit parsing, some terminology on network theory is first detailed. Formal definitions can be found in [98].

A path is a set of consecutive branches that have a common endpoint where each node is the endpoint for no more than two branches and the starting and ending nodes are not the same. An example of a path is shown in Figure 3. (d,e,f,g) is not a path because node 5 is shared by 3 branches while (b,c,e) is not a set of consecutive branches that share a common endpoint. However, (a,b,d,e) satisfies all of the constraints and is therefore considered a path.

From a set of paths, a connected graph of vertices and edges can then be formed where a path can be found between any two nodes in the graph. In Figure 4, the middle graph is not a connected graph since node 4 does not have a path to reach it. The left and right graphs are connected graphs.

A subgraph, a set of branches and nodes withing the graph, is called a loop if the subgraph is connected and every node of the subgraph has two branches incident upon it. In Figure 5, (h,b,f,g) is not a loop since it is not connected. (d,e,f,g) has three node incident at node 5 within the subgraph and is therefore not a loop. (a,b,d,e,f) surpasses the requirements of being a loop.

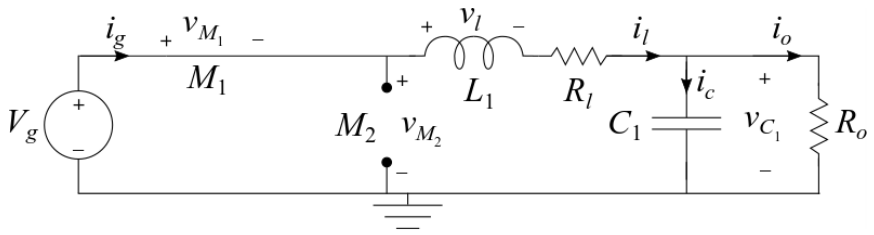


Figure 1: Buck Converter for State-Space Derivation.

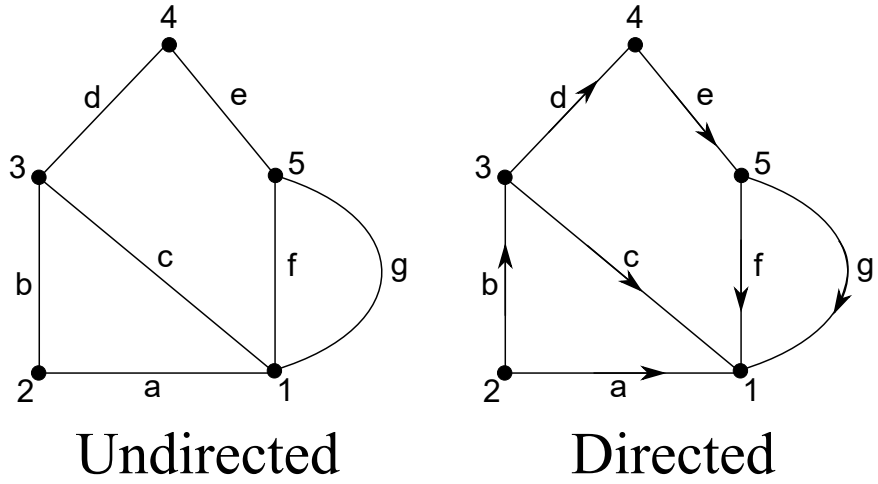


Figure 2: Undirect and direct graphs.

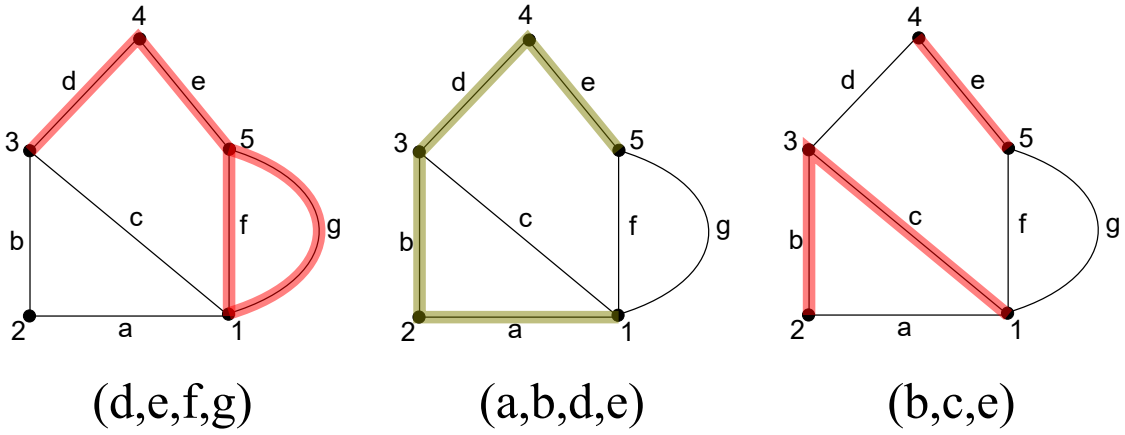


Figure 3: Path examples

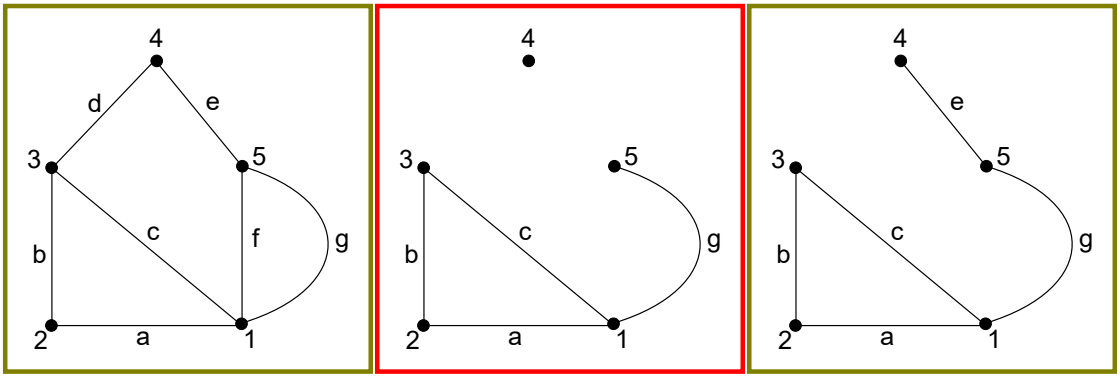


Figure 4: Connected graph examples

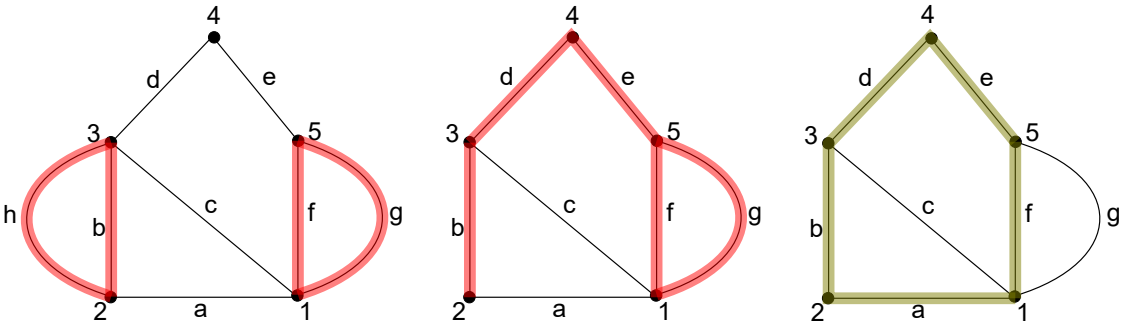


Figure 5: Loops

Using the definition of a loop, trees, can then be formed within the graph. A subgraph is called a tree if it has no loops, is a connected, and contains all nodes of the original graph.Figure 6 shows some examples of what makes a tree. (d,e,f) is not a tree due to violating constraint 2 by not including node 2. (b,c,e) contains all of the nodes but is not connected and therefore violates constraint 1. (b,d,e,f,g) contains all nodes and is connected; however, there is a loop (f,g) and therefore is not a tree.

A subgraph is called a cutsets if the removing all of the branches within the subgraph results in the orignal graph no long being connected and if reinserting one of the removed branches results in a connected graph again. In Figure 7, (b,c,d,g) is not a cutset, replacing branch g will not result in a connected graph again. (e,g) is not a cutset since a non-connected graph is not formed upon the branches removal. Another way of describing cutsets is by drawing a circle around some of the nodes of the graph. If a circle is drawn around a set of nodes, then the branches that intersect the circle form a cutset as shown in Figure 7.

Matrices

Using the definitions for a path, connected graph, loop, tree, and cutset, matrices can be formed which will be used to describe and parse a circuit topology. Formal definitions of these matrices can be found in [98].

In order to describe a graph in matrix form, an incidence matrix is used to show the connection of branches and nodes as shown in Figure 8. Each row is represented by a node and each branch is delineated by a column of the incidence matrix $\boldsymbol{A_a}$. If a branch is not incident at a node then a 0 is placed in the respective column and row of $\boldsymbol{A_a}$. If a branch and row are incident, the arrow direction of the branch determines if a 1 or -1 is placed in the respective row and column of $\boldsymbol{A_a}$, 1 when the arrow is point away from the node and -1 when the arrow is pointing towards the node.

The incidence matrix contains all of the connections between all nodes and branches; however, it is not the most compact form to store the graph information. Any row or node can be removed from the incidence matrix and not information will be lost due to each column needed to be summed to zero. The resulting matrix is then called a reduced incidence matrix, $\boldsymbol{A}$, shown in Figure 9.

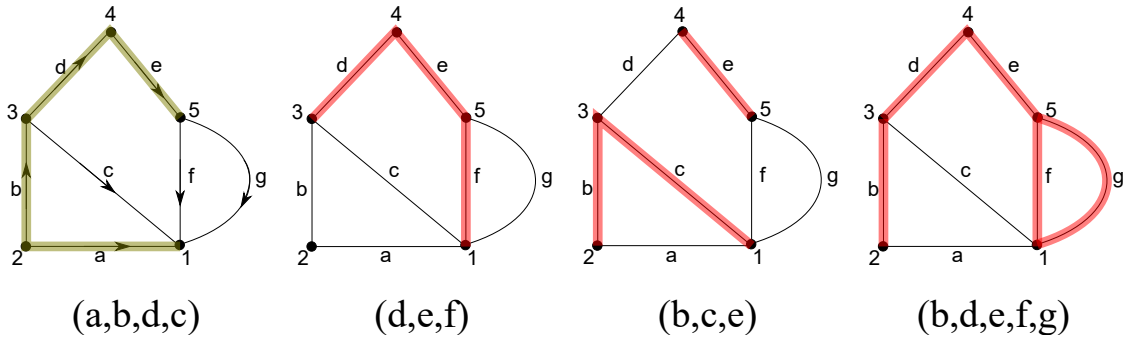


Figure 6: Trees

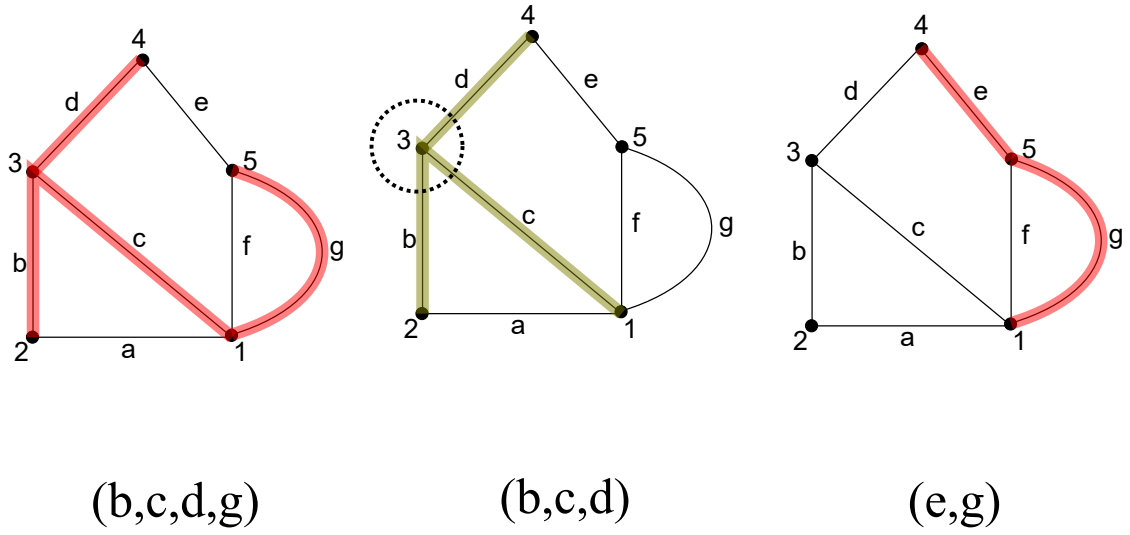


Figure 7: Cutsets

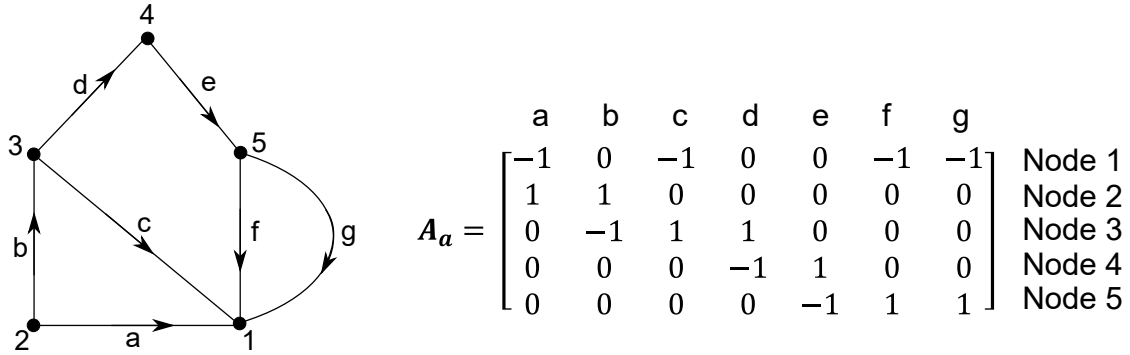


Figure 8: Incidence Matrix

KCL equations using $\boldsymbol{A_a}$ and $\boldsymbol{A}$ can be formed as $\boldsymbol{A_a i} = \boldsymbol{0}$ and $\boldsymbol{A i} = \boldsymbol{0}$, respectively. Additionally, $\boldsymbol{A}$ can be partitioned such that columns of $\boldsymbol{A}_T$ correspond to the tree branches of a chosen tree and $\boldsymbol{A}_L$ correspond to the links. This partition takes the form $\boldsymbol{A} = [\boldsymbol{A}_T\ \boldsymbol{A}_L]$ as shown in Figure 9 by the dotted line.

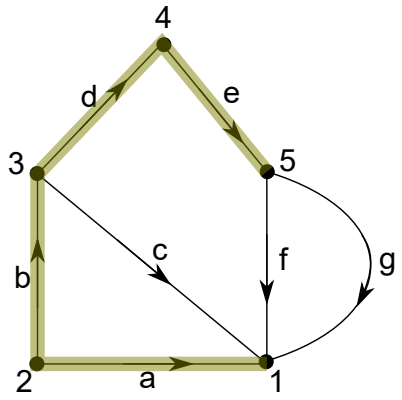
In addition to describing the matrix by its branch and node connections, all of the possibles loops of a graph can also be described in a loop matrix as shown in Figure 10. In the loop matrix each column is represented by the branches of the graph and each row describes a subgraph of branches forming a loop. If the loop and branch direction agree, a 1 is placed in the respective row and column of $\boldsymbol{B}_a$. If their directions opposes, a -1 is placed, and if the loop and branch do not incident, a 0 is placed.

Instead of describing all the loops in a graph, a unique fundamental loop matrix can be formed for every tree that can be defined within a graph as shown in Figure 11. KVL equations using $\boldsymbol{B_a}$ and $\boldsymbol{B}$ can be formed as $\boldsymbol{B_a v} = \boldsymbol{0}$ and $\boldsymbol{B v} = \boldsymbol{0}$, respectively. Additionally, $\boldsymbol{B}$ can be partitioned such that columns of $\boldsymbol{B}_T$ correspond to the tree branches of a chosen tree and the links can be represented by the identity matrix. This partition takes the form $\boldsymbol{B} = [\boldsymbol{B}_T\ \boldsymbol{1}_\mu]$ as shown in Figure 11 by the dotted line. To find the fundamental loop matrix, the incidence matrix can be used after establishing a tree within the graph

$$\boldsymbol{A B}^t = \begin{bmatrix} \boldsymbol{A}_T \boldsymbol{A}_L \end{bmatrix} \begin{bmatrix} \boldsymbol{B}_T^t \\ \boldsymbol{1}_\mu \end{bmatrix} = \boldsymbol{A}_T \boldsymbol{B}_T^t + \boldsymbol{A}_L = \boldsymbol{0} \therefore \boldsymbol{B}_T^t = -\boldsymbol{A}_T^{-1} \boldsymbol{A}_L. \tag{6}$$

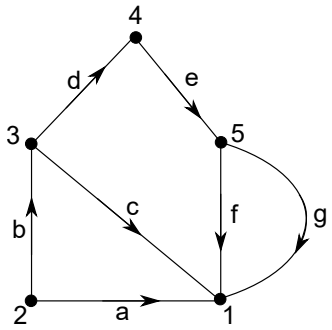
Similar to the loop matrix, a cutset matrix defines all of the cutsets that can be found within a graph. Figure 12 shows an example of a few cutsets within a connected graph. The cutset matrix $\boldsymbol{D_a}$ is defined by columns representing the graphs branches and the cutsets representing the rows. When defining the cutset using a circle, branches are represented by a 1 when their direction goes into the circle and -1 when their direction opposes.

Similar to the loop matrix, there exists a unique fundamental cutset matrix for every tree of a graph where a tree branch together with some combinations of cotree branches forms a cutset. Figure 13 shows an example of a fundamental cutset matrix for the given tree. KCL equations using $\boldsymbol{D_a}$ and $\boldsymbol{D}$ can be formed as $\boldsymbol{D_a i} = \boldsymbol{0}$ and $\boldsymbol{D i} = \boldsymbol{0}$, respectively. Additionally, $\boldsymbol{D}$ can be partitioned such that an identity matrix correspond to the tree



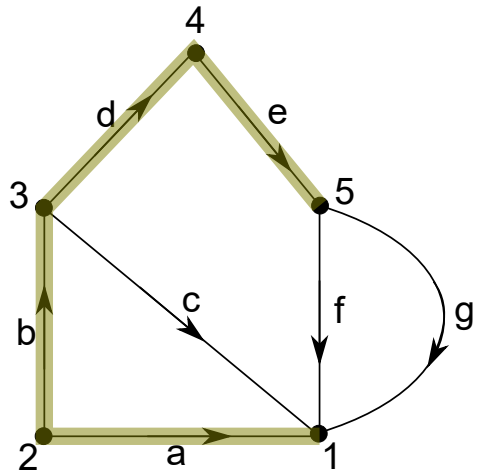
$$\boldsymbol{A} = \begin{array}{c} \begin{matrix} a & b & d & e & c & f & g \end{matrix} \\ \left[\begin{array}{cccc:ccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 1 \end{array}\right] \end{array} \begin{matrix} \\ \text{Node 2} \\ \text{Node 3} \\ \text{Node 4} \\ \text{Node 5} \end{matrix}$$

Figure 9: Reduced Incidence Matrix



$$\boldsymbol{B_a} = \begin{array}{c} \begin{matrix} a & b & c & d & e & f & g \end{matrix} \\ \begin{bmatrix} -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 & -1 & -1 & 0 \\ -1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} \text{Branches in loop} \\ \text{(abc)} \\ \text{(fg)} \\ \text{(fedc)} \\ \text{(abdef)} \\ \text{(cdefg)} \end{matrix}$$

Figure 10: Loop Matrix



$$\boldsymbol{B} = \begin{array}{c} \begin{matrix} a & b & d & e & c & f & g \end{matrix} \\ \left[\begin{array}{cccc:ccc} -1 & 1 & 0 & 0 & 1 & 0 & 0 \\ -1 & 1 & 1 & 1 & 0 & 1 & 0 \\ -1 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right] \end{array}$$

Figure 11: Fundamental Loop Matrix

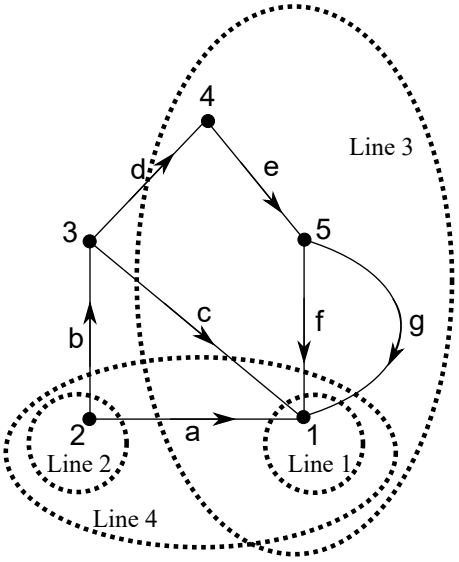
branches of a chosen tree and $\boldsymbol{D}_L$ correspond to the links. This partition takes the form $\boldsymbol{D} = [\mathbf{1}_\rho \ \boldsymbol{D}_L]$ as shown in Figure 13 by the dotted line. The fundamental cutset matrix for a given tree can be solved for by using the fundamental loop matrix

$$\boldsymbol{D}\boldsymbol{B}^t = \begin{bmatrix} \mathbf{1}_\rho \boldsymbol{D}_L \end{bmatrix} \begin{bmatrix} \boldsymbol{B}_T^t \\ \mathbf{1}_\mu \end{bmatrix} = \boldsymbol{B}_T^t + \boldsymbol{D}_L = \mathbf{0} \therefore \boldsymbol{B}_T^t = -\boldsymbol{D}_L \tag{7}$$

which can be solved for via the fundamental incidence matrix using (6). Therefore, for a given tree of a graph, in order to determine the loops and cutsets within a connected graph, the only information required is the incidence matrix and the tree itself. The fundamental loop and cutset matrices can be solved using matrix algebra and will be demonstrated using an example of a buck converter in the next section.

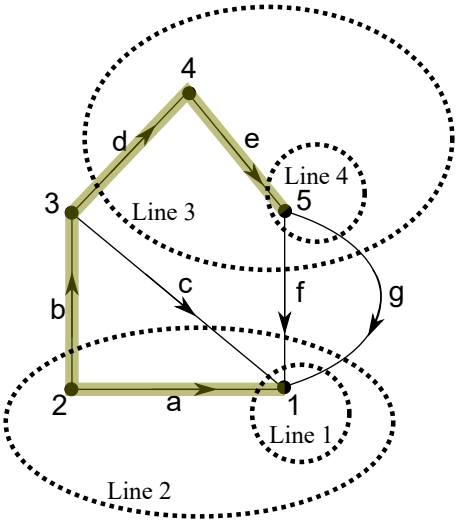
A.2 Netlist to State-Space

This section will go through the process of combining all of the knowledge presented using graph theory to complete a state-space equation for the buck converter shown in Figure 13. The process will be shown for when M_1 is conducting and M_2 is blocking voltage; however, the process can be repeated for the other linear time intervals of a switching buck converter. First, M_1 and M_2 are replaced by their equivalent circuits for their current region of operation. For simplicity of the example, both will be replaced by a resistor M_1 by the conducting resistance R_{on} and M_2 by its blocking resistance R_{off} as shown in Figure 13. Next, the information displayed on the circuit can be represented as a directed graph with each element and its value representing a branch connected between the nodes of the circuit. This information is commonly stored describing each of the lumped elements of the circuit, the nodes the element goes between and a value associated with the element representing its voltage, resistance, etc. in a single table known as a netlist as shown in Table 2. The netlist can then be used to create an directed graph describing the circuit.



$$\boldsymbol{D_a} = \begin{array}{c} \begin{matrix} a & b & c & d & e & f & g \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} \text{Line 1} \\ \text{Line 2} \\ \text{Line 3} \\ \text{Line 4} \end{matrix}$$

Figure 12: Cutset Matrix



$$\boldsymbol{D} = \left[\begin{array}{cccc|ccc} a & b & d & e & c & f & g \\ \hline 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 1 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 1 & 0 & -1 & -1 \end{array}\right] \begin{matrix} \text{Line 1} \\ \text{Line 2} \\ \text{Line 3} \\ \text{Line 4} \end{matrix}$$

Figure 13: Fundamental Cutset Matrix

A.3 Normal Tree

After correctly formatting the circuit's information in a netlist and therefore a incidence matrix, the next step is to determine a tree for the graph. For some applications (such as state-space formulation) there is a preferred order of elements included in the tree. For state-space modeling the preferred order is

1. Independent Voltage Sources
2. Dependent Voltage Sources
3. Capacitors
4. Resistors
5. Inductors
6. Dependent Current Sources
7. Independent Current Sources

The order drives voltage controlled elements such as voltage sources and capacitors to be in the tree while current dependent devices outside of the tree or in the cotree. Elements within the tree will have voltage be their state variable which is conducive for capcitors while element ouside of the tree will have current be the state variable.

A.4 Finding Loop and Cutset Matrices

Having established the desired order for the elements within the tree, the columns of the incidence matrix are then switched so that the most desired elements within the tree are on the left side of the matrix and the most desired cotree elements are on the right side of the incidence matrix as shown in Figure 14.

After sorting the incidence matrix, in order to find the elements that will be in the tree, the reduced row echelon from of incidence matrice is solved. The leading ones, highlighted in Figure 15, are the elements that are selected to be in the tree. Notice $R_{off,M2}$ is not selected to be in the tree due to it forming a loop with V_g and $R_{on,M1}$. The incidence matrix is not

Table 2: Netlist of a Buck Converter

Element	Node 1	Node 2	Value
Vg	Vin	GND	12
Ron,M1	Vin	Vsw	0.001
Roff,M2	Vsw	GND	10000
L	Vsw	VL	20m
RL	VL	Vout	0.02
C	Vout	GND	10μ
Rout	Vout	GND	2

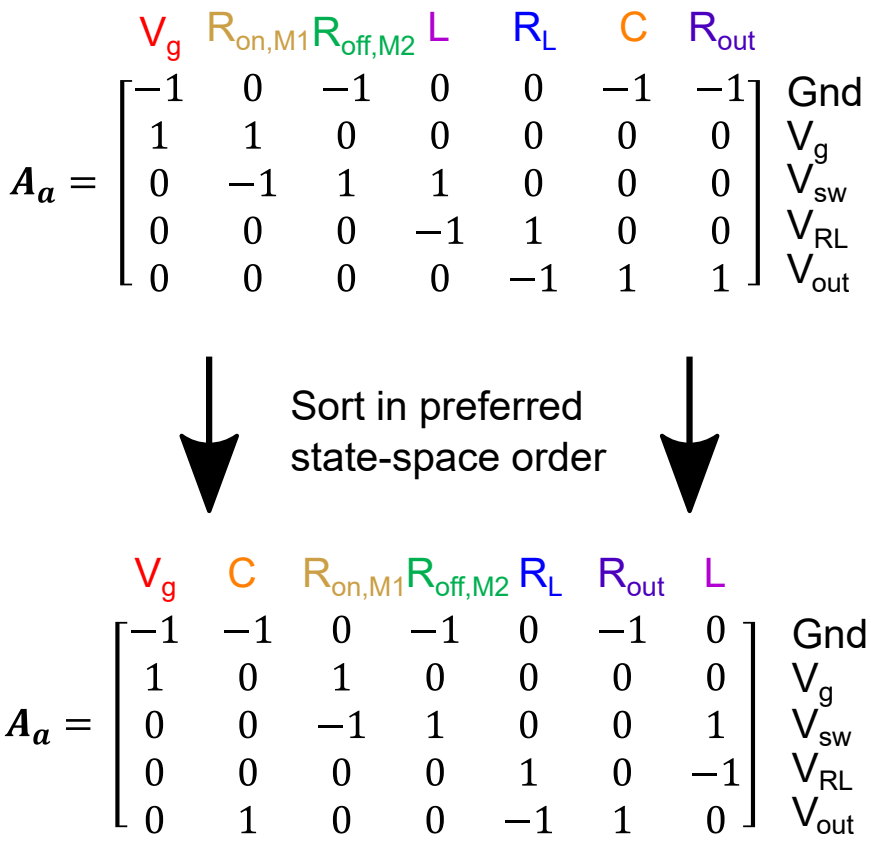


Figure 14: Sorting Incidence Matrix

the most compact form of describing a circuit. Therefore, the last row is zeroed out since it is dependent on the other rows of the matrix.

By rearranging the columns of the reduced row echelon form incidence matrix after removing the zeroed row, the fundamental cutset matrix can be found as shown in Figure 16.

Then by utilizing (7), the fundamental loop matrix of the converter can be solved via reorienting the fundamental cutset matrix as shown in Figure 17.

The resulting loop and cutset matrices can be used as KVL and KCL equations describing the converter,

$$\boldsymbol{D}\boldsymbol{i} = [\mathbf{1}_\rho \ \boldsymbol{D}_L]\boldsymbol{i} = \begin{bmatrix} \mathbf{1}_{EE} & \mathbf{0}_{ER} & \boldsymbol{D}_{EG} & \boldsymbol{D}_{EJ} \\ \mathbf{0}_{RE} & \mathbf{1}_{RR} & \boldsymbol{D}_{RG} & \boldsymbol{D}_{RJ} \end{bmatrix} \begin{bmatrix} \boldsymbol{i}_E \\ \boldsymbol{i}_R \\ \boldsymbol{i}_G \\ \boldsymbol{i}_J \end{bmatrix} = \mathbf{0} \tag{8}$$

$$\boldsymbol{B}\boldsymbol{v} = [\boldsymbol{B}_T \ \mathbf{1}_\mu]\boldsymbol{i} = \begin{bmatrix} -\boldsymbol{D}_{EG}^t & -\boldsymbol{D}_{RG}^t & \mathbf{1}_{GG} & \mathbf{0}_{GJ} \\ -\boldsymbol{D}_{EJ}^t & -\boldsymbol{D}_{RJ}^t & \mathbf{0}_{JG} & \mathbf{1}_{JJ} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_E \\ \boldsymbol{v}_R \\ \boldsymbol{v}_G \\ \boldsymbol{v}_J \end{bmatrix} = \mathbf{0} \tag{9}$$

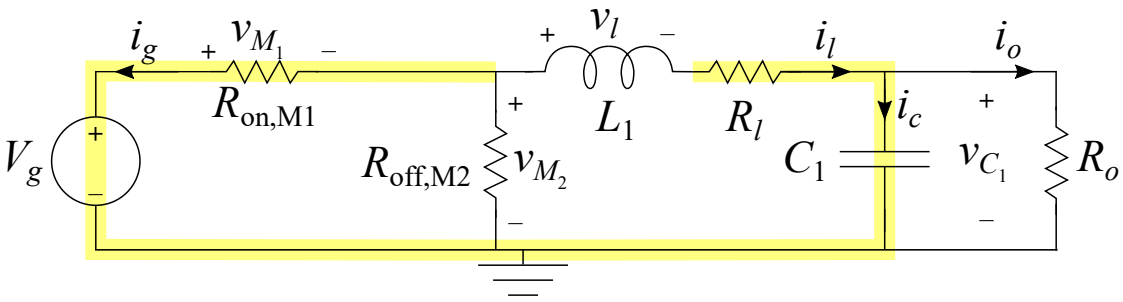
Figure 18 shows how the converter elements are broken down in these equations. E represents all state and constant input variables that are in the tree. R represents all of the resistive elements that are in the tree. G represents all of the resistive elements that are not in the tree and J represents all of the state variables and constants that are not in the tree.

A.5 Solving for State Matrices

After finding the cutset and loop matrices, the hybrid equation can be solved for which relates the state variables voltage and currents to each other that will take the form of

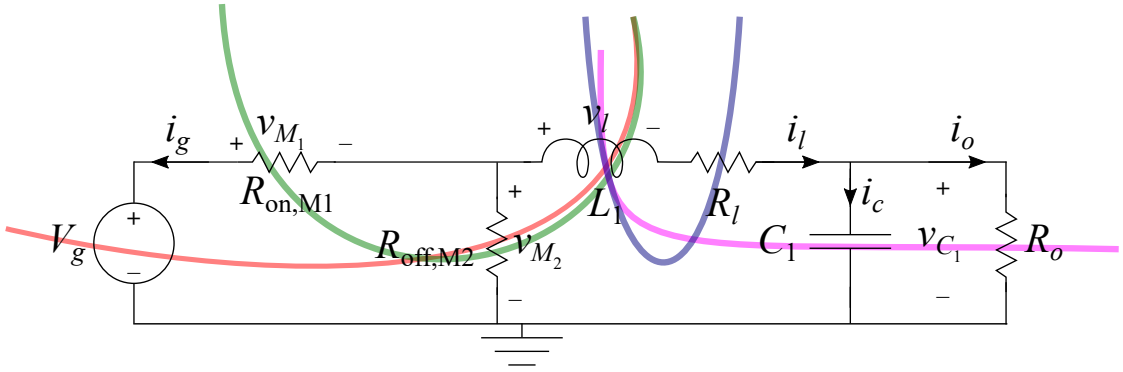
$$\begin{bmatrix} \boldsymbol{i}_E \\ \boldsymbol{v}_J \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}_{EE} & \boldsymbol{H}_{EJ} \\ \boldsymbol{H}_{JE} & \boldsymbol{H}_{JJ} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_E \\ \boldsymbol{i}_J \end{bmatrix}. \tag{10}$$

The following section is a derivation of the hybrid equation. The process begins with the loop and cutset equations formed from (8) and (9)



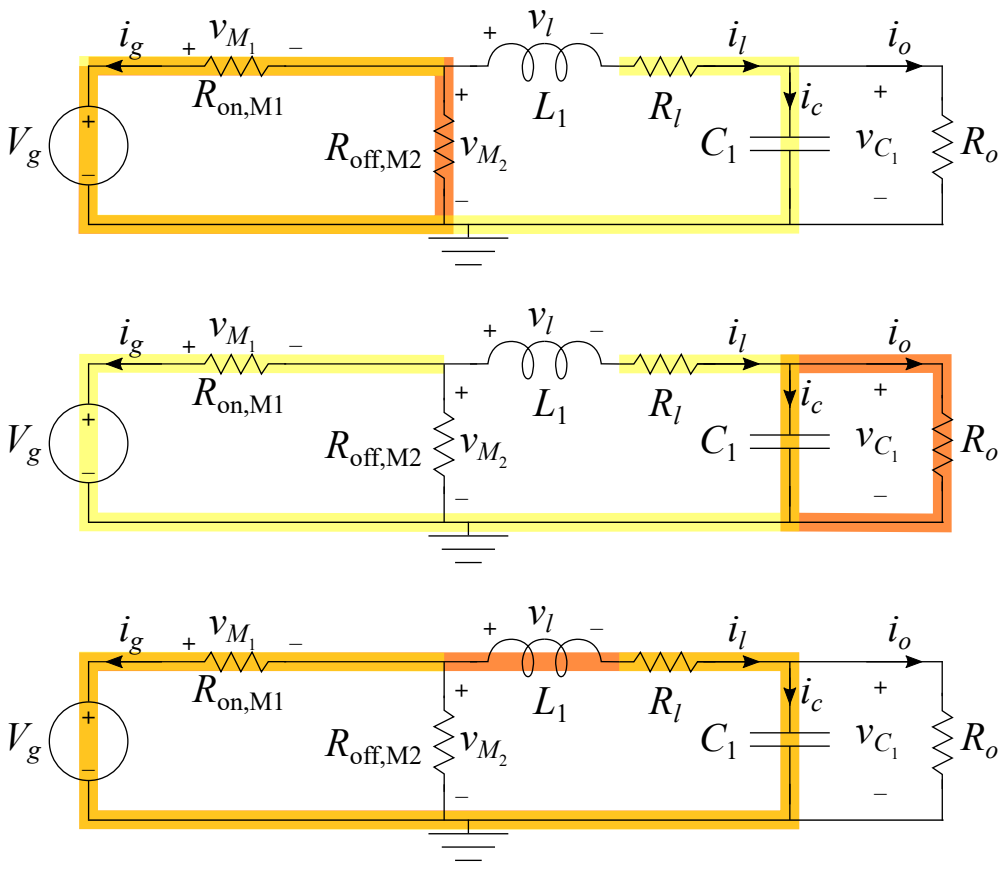
$$rref(\boldsymbol{A_a}) = \begin{array}{c} \begin{matrix} V_g & C & R_{on,M1} & R_{off,M2} & R_L & R_{out} & L \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} \text{Gnd} \\ V_g \\ V_{sw} \\ V_{RL} \\ V_{out} \end{matrix}$$

Figure 15: Reduced Row Echelon Form of Incidence Matrix



$$\boldsymbol{D} = [\mathbf{1} \quad \boldsymbol{D_L}] = \begin{array}{c} \begin{matrix} V_g & C & R_{on,M1} & R_L & R_{off,M2} & R_{out} & L \end{matrix} \\ \left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \end{array}\right] \\ \begin{matrix} \text{Tree} & & & & \text{CoTree} & & \end{matrix} \end{array}$$

Figure 16: Buck Converter Fundamental Cutset Matrix



$$\boldsymbol{B} = [\boldsymbol{B}_T \quad \mathbf{1}] = \begin{array}{c} \begin{matrix} V_g & C & R_{on,M1} & R_L & R_{off,M2} & R_{out} & L \end{matrix} \\ \left[\begin{array}{ccccc|cc} -1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ -1 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right] \end{array}$$

Figure 17: Loop Matrix of Buck Converter

$$\begin{bmatrix} \boldsymbol{i}_E \\ \boldsymbol{v}_J \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}_{EG}\boldsymbol{Z}^{-1}\boldsymbol{D}_{EG}^t & \boldsymbol{D}_{EG}\boldsymbol{Z}^{-1}\boldsymbol{D}_{RG}^t\boldsymbol{Z}_R\boldsymbol{D}_{RJ} - \boldsymbol{D}_{EJ} \\ \boldsymbol{D}_{EJ}^t - \boldsymbol{D}_{RJ}\boldsymbol{Y}^{-1}\boldsymbol{D}_{RG}\boldsymbol{Y}_G\boldsymbol{D}_{EG}^t & \boldsymbol{D}_{RJ}^t\boldsymbol{Y}^{-1}\boldsymbol{D}_{RJ} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_E \\ \boldsymbol{i}_J \end{bmatrix}. \tag{11}$$

Utilizing $v_L = L\frac{di_L}{dt}$ and $i_C = C\frac{dv_C}{dt}$ the state matricies can be formed as shown in Figure 19.

A.6 Transformers

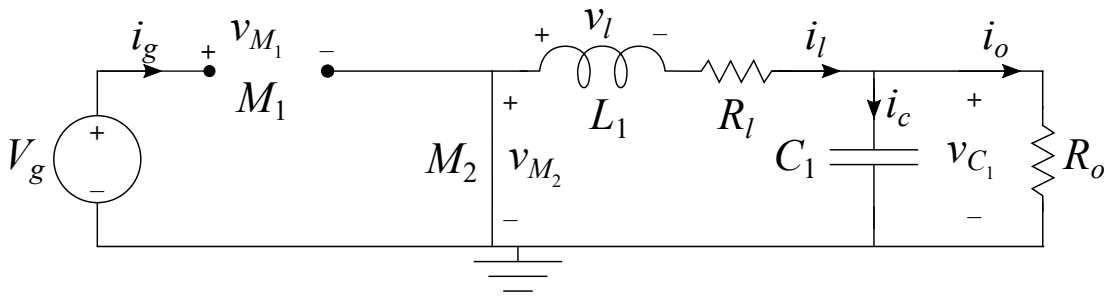
In order to account for transformers in the state-space representation, some additional work is required. Transformers are replaced by their canonical model where dependent voltage and current sources are used to account for turns ratio of the transformer. For the

$$\begin{bmatrix} \boldsymbol{v}_{vv} \\ \boldsymbol{i}_{vc} \\ \boldsymbol{v}_{cv} \\ \boldsymbol{i}_{cc} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{K}_{vv} \\ \boldsymbol{0} & \boldsymbol{K}_{vc} \\ \boldsymbol{K}_{cv} & \boldsymbol{0} \\ \boldsymbol{K}_{cc} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{i}_{sc} \\ \boldsymbol{v}_{sc} \end{bmatrix}. \tag{12}$$

$$\boldsymbol{Di} = \begin{array}{c} \begin{matrix} \text{E} & & \text{R} & & \text{G} & & \text{J} \end{matrix} \\ \left[\begin{array}{cc|cc|cc|c} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & -1 \\ \hline 0 & 0 & 1 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \end{array}\right] \end{array} \begin{bmatrix} i_{Vg} \\ i_C \\ i_{Ron,M1} \\ i_{RL} \\ i_{Roff,M2} \\ i_{Rout} \\ i_L \end{bmatrix}$$

$$\boldsymbol{Bv} = \begin{array}{c} \begin{matrix} \text{E} & & \text{R} & & \text{G} & & \text{J} \end{matrix} \\ \left[\begin{array}{cc|cc|cc|c} -1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ \hline -1 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right] \end{array} \begin{bmatrix} v_{Vg} \\ v_C \\ v_{Ron,M1} \\ v_{RL} \\ v_{Roff,M2} \\ v_{Rout} \\ v_L \end{bmatrix}$$

Figure 18: KVL and KCL equations separated by tree and element type.



Hand Calculations

$$i_C(t) = i_L(t) - {}^{v_C(t)}/_{R_o}$$

$$v_L(t) = V_g - i_L(t)R_l - v_C(t)$$

$$\begin{bmatrix} i_C(t) \\ v_L(t) \end{bmatrix} = \begin{bmatrix} {}^1/_{R_o} & 1 \\ -1 & R_l \end{bmatrix} \begin{bmatrix} v_C(t) \\ i_L(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} [V_g]$$

$$\begin{bmatrix} C & 0 \\ 0 & L \end{bmatrix} \begin{bmatrix} \dot{v_C}(t) \\ \dot{i_L}(t) \end{bmatrix} = \begin{bmatrix} {}^1/_{R_o} & 1 \\ -1 & R_l \end{bmatrix} \begin{bmatrix} v_C(t) \\ i_L(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} [V_g]$$

$$\begin{bmatrix} \dot{v_C}(t) \\ \dot{i_L}(t) \end{bmatrix} = \begin{bmatrix} {}^1/_{C \times R_o} & {}^1/_C \\ -{}^1/_L & {}^{R_l}/_L \end{bmatrix} \begin{bmatrix} v_C(t) \\ i_L(t) \end{bmatrix} + \begin{bmatrix} 0 \\ {}^1/_L \end{bmatrix} [V_g]$$

Circuit Parser

$$\begin{bmatrix} i_{V_g}(t) \\ i_C(t) \\ v_L(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & {}^1/_{R_o} & 1 \\ 1 & -1 & R_l \end{bmatrix} \begin{bmatrix} v_{V_g}(t) \\ v_C(t) \\ i_L(t) \end{bmatrix}$$

$$\begin{bmatrix} i_C(t) \\ v_L(t) \end{bmatrix} = \begin{bmatrix} {}^1/_{R_o} & 1 \\ -1 & R_l \end{bmatrix} \begin{bmatrix} v_C(t) \\ i_L(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} [V_g]$$

Figure 19: Comparison of hand calculations and parser results.

B AURA User Guide

AURA is an optimization, modeling and database framework designed to assist researchers in early stage development of advanced switch mode power converters. This document serves as a how to guide to the framework to set up a database of transistors, inductors, and topologies, understand how to set up a steady-state simulation using discrete time state-space modeling and apply the results to an optimization routine.

- Examples of database set ups are in files Test_###_Database.m
- Example of the modeling is in COMPEL_2023_Buck_D_Sweep_dead.m which is a function called by the optimization code.
- Example of the optimization is in Optim_Buck_D_dead.m

B.1 Database

The database currently consists of a tiered class system where there are different databases for specific components. As of writing, this consist of a component class which handles units and prefixes for the transistor and inductor class and a database class that consists of classes of components.

How form a database/component and add tabular parameters

The constructor for the transistor and inductor databases is similar. For this example we will walk through the transistor database.

To form a new transistor database class where tDB will be the name of the Class use the following code:

```
tDB = transistorDB();
```

To create a transistor class where t will be the name of the class, partNumber will be the part number of the transistor, type is the type of transistor, and material is the material of the transistor

```
t = transistor('partNumber','type','material');
```

The input types must contain one of the following known types of transistors:

```
types  = {'en-nMOS', 'en-pMOS', 'dep-nMOS', 'dep-pMOS',...
          'en-npMOS', 'HEMT', 'HFET', 'PT-IGBT',...
          'NPT-IGBT', 'nJFET', 'pJFET', 'NPN', 'PNP'};
```

The input materials must contain one of the following known materials of transistors:

```
material = {'Si', 'GaN', 'SiC', 'GaAs'}
```

An example of how to construct a transistor class t:

```
t = transistor('EPC2055','en-nMOS','GaN');
```

To add table parameters to the newly formed transistor class use syntax:

```
t.addParameter('parameter',value,'min/typ/max',...
        {'Conditions'});
```

The current parameters each class are shown in Tables 3,4. The spice values are for use on the spice equivalent circuit for Coilcraft inductors. Each table parameter can contain min, max, and/or typ values. Conditions is an area where additional information such as test conditions used to obtain the parameter.

To add a transistor class to the transistor database use the syntax:

```
tDB.add(t)
```

The following example creates a transistor database and a transistor class, defines the properties of the transistor, add a set a parapmeters to the transistor, and adds the transistor to the database.

```
transDB = transistorDB();
EPC2055 = transistor('EPC2055','en-nMOS','GaN');
EPC2055.manufacturer = 'EPC';
EPC2055.material = 'GaN';
EPC2055.addParameter('Vds', 40 ,'max',...
        {'VGS = 0 V, ID = 0.5 mA (Continuous)'} )
EPC2055.addParameter('Ids', 29 ,'max',
```

Table 3: Currently known transistor parameters

Name	Parameter	Name	Parameter
Drain-Source Voltage	Vds	Turn-on Energy	Eon
Source-Drain Voltage	Vsd	Turn-off Energy	Eoff
Gate-Source Voltage	Vgs	Coss Stored Energy	Eoss
Gate Threshold Voltage	Vth	Power Dissipation	Pd
Drain-Source Current	Ids	Transconductance	Gfs
Source-Drain Current	Isd	Rise Time	Tr
Gate Current	Ig	Turn-On Delay Time	Tdon
Pulsed Drain-Source Current	Idpulse	Fall Time	Tf
Pulsed Source-Drain Current	Ispulse	Turn-Off Delay Time	Tdoff
Gate-Source Leakage Current	Igss	Junction Temperature	Tj
Drain-Source Leakage Current	Idss	Case Temperature	Tc
Peak Reverse Recovery Current	Irr	Length	Length
Drain-Source On-State Resistance	Rds	Width	Width
Output Capacitance	Coss	Height	Height
Input Capacitance	Ciss	Internal Gate Resistance	Rg
Reverse Transfer Capacitance	Crss	Total Gate Charge	Qg
Gate-Drain Capacitance	Cgd	Gate-Source Charge	Qgs
Drain-Source Capacitance	Cds	Gate-Drain Charge	Qgd
Gate-Source Capacitance	Cgs	Output Charge	Qoss
Junction-Case Thermal Resistance	Rthjc	Reverse Recovery Charge	Qrr
Junction-Board Thermal Resistance	Rthjb	Reverse Recovery Time	Trr
Junction-Ambient Thermal Resistance	Rthja		

Table 4: Currently known inductor parameters

Name	Parameter
Inductance	L
Frequency	F
Current	I
DC Resistance	Rdc
Self Resonant Frequency	Srf
Saturation Current	Isat
RMS Current	Irms
SPICE Frequency Limit	Fspice
SPICE R1	R1spice
SPICE R2	R2spice
SPICE C	Cspice
SPICE k1	K1spice
SPICE k2	K2spice
SPICE k3	K3spice
SPICE k4	K4spice
SPICE k5	K5spice
Length	Length
Width	Width
Height	Height

```
        {'(Continuous) TA = 25 C'} )
EPC2055.addParameter('Idspulsed', 161, 'max',...
        {'(Pulsed) 25 C Tpulse = 300e-6 s'} )
EPC2055.addParameter('Ron', 0.003 ,'typ',...
        {'VGS = 5 V, ID = 15 A'} )
EPC2055.addParameter('Ron', 0.0036 ,'max',...
        {'VGS = 5 V, ID = 15 A'} )
EPC2055.addParameter('Vth', 0.7 ,'min',...
        {'VDS = VGS, ID = 7 mA'} )
EPC2055.addParameter('Vth', 1.1 ,'typ',...
        {'VDS = VGS, ID = 7 mA'} )
EPC2055.addParameter('Vth', 2.5 ,'max',...
        {'VDS = VGS, ID = 7 mA'} )
EPC2055.addParameter('Vsd', 1.9 ,'typ',...
        {'IS = 0.5 A, VGS = 0 V '} )
EPC2055.addParameter('Coss', 574e-12 ,'typ',...
        {'VDS = 20 V, VGS = 0 V'} )
EPC2055.addParameter('Coss', 612e-12 ,'max',...
        {'VDS = 20 V, VGS = 0 V'} )
EPC2055.addParameter('Rg', 0.4 ,'typ')
EPC2055.addParameter('Qg', 6.6e-9 ,'typ',...
        {'VDS = 20 V, VGS = 5 V, ID = 15 A'} )
EPC2055.addParameter('Qg', 8.5e-9 ,'max',...
        {'VDS = 20 V, VGS = 5 V, ID = 15 A'} )
EPC2055.addParameter('Length', 2.5e-3 ,'typ')
EPC2055.addParameter('Width', 1.5e-3 ,'typ')
EPC2055.addParameter('Height', 0.638e-3 ,'typ')
transDB.add(EPC2055)
transDB.saveDB
```

How form a database/component and add graphical parameters

To add a digitized plot the transistor class, use the digitizedPlot class

```
EPC2207 = transistor('EPC2207','en-nMOS','GaN');
EPC2207DP = digitizedPlot([]);
EPC2207DP.plotData = {[x1, y1; x2, y2; xn, yn ]};
EPC2207DP.logAxes = [0 1];
EPC2207DP.normAxes = [0 0];
EPC2207DP.axisLabels = {'Vds, Drain-Source Voltage',...
        'Coss, Output Capacitance'};
EPC2207DP.dataLabels = {'Y1'};
EPC2207DP.SIUnits = { '' , 'p'};
EPC2207graph = componentPlotData(transistor, EPC2207DP);
EPC2207.addGraph(EPC2207graph);
```

where

- plotData contains the [x,y] digitized data
- logAxes is a binary to determine if the [x y] axis are normalized.
- normAxes is a binary to determine if the [x y] axis are logarithmic.
- axisLabels is the label of each [x y] axis
- dataLabels is the overall data level provided in plotData
- SIUnits sets the unit plotData was entered - currently known prefixes:
  {'f','p','n','µ,'m','','k','M','G','T'}
- componentPlotData(componentClass,digitizedPlotClass) creates a class of the component plot data.
- componentClass is the name of the component class that you are adding the plot data to.
- digitizedPlotClass is the class that you added the plot data to.

How to form a datasheet

For any component in a transistor or inductor class, a datasheet can be displayed to showcase all the components tabular and graphical data. To display the information for a transistor class named EPC2055 use the syntax: EPC2055.datasheet.

The datasheet can also be called from the transistorDB class if the index to the transistor desired is known. For example, if the EPC2055 transistor is 3rd index in a transistorDB class named tDB then tDB(1).datasheet will also display the datasheet

How to extract tabular data from a database class

To pull tabular parameters out of the class use the syntax:

```
tDB(n).parameter.approx/min/typ/max
```

where tDB name of component database, n index of component in component database, parameter is the parameter name, and approx/min/typ/max determines which range of data parameter will be returned. approx will fill in the order of typ, max, min depending on what is filled out for the tabular data. min, typ, or max will find the given value in the tabular data if it exists.

Examples:

```
value = tDB(9).rds.approx
value = tDB(2).Coss.typ
value = tDB(5).Vds.max
```

B.2 Modeling

LTspice model

To provide a schematic framework, an LTspice netlist file is needed. The LTspice netlist file has a .net extension and is created when running a schematic. For the purposes of this program it only acts a method to input a schematic. Create of the circuit to be tested using nmos, diode, independent voltage and current sources as well as RLC values such as the buck converter shown below. Note: a diode should be placed anti-parallel to each nmos to act as

the body diode, and the gates of the nmos should be left unconnected. Also, the values that you place on the components does not matter as they will be input directly into MATLAB later.

The netlist is then produces by running the program and should look like:

```
* G:\My Drive\Graduate Reserach\LTSpice\Buck_Vout_D.asc
V1 N001 N007 12
M1 N001 NC_01 N002 N002 NMOS
M2 N002 NC_02 0 0 NMOS
L1 N003 N004 200n
V2 N004 N006 4
R2 N006 0 5m
C1 N005 0 0.002m
R4 N004 N005 2m
R1 N003 N002 0.0024
D1 N002 N001 D
D2 0 N002 D
R3 N007 0 0.002m
.model D D
.lib C:\Users\jb\Documents\LTspiceXVII\lib\cmp\standard.dio
.model NMOS NMOS
.model PMOS PMOS
.lib C:\Users\jb\Documents\LTspiceXVII\lib\cmp\standard.mos
.tran 1
.backanno
.end
```

This should be added to the topology database as the netlist file.

Table 5: Currently known capacitor parameters

Name	Parameter
Capacitance	C
DC Rated Voltage	Vdc
AC Rated Voltage	Vac
Frequency	F
Size Code	Size
Equivalent Series Resistance	Esr
Impedance	Z
Temperature	Temp
Capacitance Tolerance +/-	Tol
Length	Length
Width	Width
Height	Height

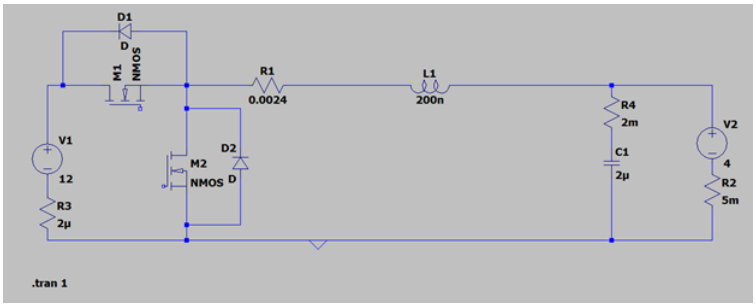


Figure 20: LTSpice example circuit model.

Setting up to model

To model the converter, there are several inputs that must be place within their appropriate class. The first class declaration should be SMPSim which acts as the simulation class of the converter. The simulation class is dependent upon the subclasses within it, the SMPSconverter and SMPStopology classes. To set up the structure correctly the flowing syntax should be used:

```
sim = SMPSim();
conv = sim.converter;
top = sim.topology;
```

There are several variables that must now be set within these classes to simulate the converter.

Element_Properties contains the designator and value of all the components in the converter. As an example:

```
top.Element_Properties = {
        'C1' 5e-6
        'L1' 1e-3
        'M1_C' 10e-9
        'M2_C' 10e-9
        'D1_C' 10e-9
        'D2_C' 10e-3
        'R1' 10e-3
        'R2' 1e-3
        'R3' 1e-3
        'R4' 1e-3
        'M1_R' 5e-3
        'M2_R' 6e-3
        'D1_R' 5e-3
        'D2_R' 9e-3}
```

The first column represents the designator for each component in the LTSpice netlist file. The second column represent the numerical value assigned to that designator. These can be listed as variables of devices that you have in the database. For each nmos and diode that is place in the converter, there must be two separate parameters (1 for the Ron resistance labeled as Designator_R and 1 for the Cds of the semiconductor device labeled as Designator_C). The resistance parameters should be located at the end of the matrix as shown in the example above.

top.Switch_Resistors should be a copy of the semiconductor resistance values located at the end of the top.Element_Properties.

```
top.Switch_Resistors = {
        'M1_R'
        'M2_R'
        'D1_R'
        'D2_R'
};
```

top.Switch_Names provide the name of all the semiconductors in the converter. This should be in the same order as presented above and only contain the designator. This sets the order with which the switching sequence is declared in the next step. It is recommended that all transistor are placed first and diodes second as shown in the example.

```
top.Switch_Names = {
        'M1'
        'M2'
        'D1'
        'D2'
};
```

top.Switch_Sequence sets the switching sequence in the order of the top.Switch_Resistors order with columns representing semiconductor devices and rows representing time intervals. A 1 indicates the semiconductor is on and conducting current. A 0 indicates a semiconductor is off or blocking voltage

```
top.Switch_Sequence = [
 %  M1   M2   D1   D2
          1    0    0    0  % M1 ON
          0    0    0    0  % Deadtime
          0    1    0    0  % M2 ON
          0    0    0    0  % Deadtime
];
```

top.Fwd_Voltage provides the forward voltage for all the transistors. Since moving the declared body diode all transistor forward voltages should be zero with the body diode forward voltage reflected by the anti-parallel diode

```
                    %   M1 M2 D1   D2
top.Fwd_Voltage = [0   0   1.5  1.6]';
```

top.Switch_Resistor_Values is a three row matrix of resistance values that indicate the transition from a semiconductors conduction resistance value being on and off. When a semiconductor is off in the simulation the Rds resistance is not infinite, but the value given in this matrix for each semiconductor The rows should be separated by [R_off; R_on; R_on]. For example:

```
top.Switch_Resistor_Values = [
          % M1    M2    D1    D2
          10e6  10e6  10e6  10e6  %Roff
          5e-3  6e-3  5e-3  9e-3  %Ron
          5e-3  6e-3  5e-3  9e-3  %Ron
          ];
```

conv.u sets the inputs to the converter. The converter has two different types of inputs. The first being the inputs placed in the schematic such as independent voltage and current sources, and the second being diode forward voltages. These should be input to the vector in the following order [(1) Independent voltage sources (in the order they appear in the netlist file; (2) diode forward voltages (top.Fwd_Voltage); (3) Independent current sources (in the order they appear in the netlist file]. Example:

```
conv.u = [ 20 % V1 input voltage
           4   % V2 output voltage
           0   % M1
           0   % M2
           1.5 % D1
           1.6 % D2
           ];
```

top.order should be filled in with how many time intervals are presented in top.Switch_Sequence

```
top.order = [1 2 3 4];
```

Parse the circuit and supply switching pattern

After setting up the model, the circuit can be parsed. top.loadCircuit takes LTSpice netlist and the switching patter and converter the model into a state-space form that is ready to be modelled

```
[] = top.loadCircuit(netlist,swvec,force)
```

where netlist is the .net file from LTspice, swvec is the top.Switch_Sequence values, force is a binary value that should be set to 1 to force the state-space values to update for the switch vector given.

After parsing the circuit, the time duration of each subinterval should be set with conv.setSwitchingPattern(swind, ts) where swind should be equal to top.order and ts is the length in seconds of each time interval

```
conv.setSwitchingPattern( [1 2 3 4],...
          [0.2 0.04 0.7 0.04].*1e-6);
```

Finding the steady-state of the converter

The steady state of the converter is ready to be calculated. To calculate the steady-state waveforms of the converter, use the function Xss = sim.steadyState;.

The result, Xss is a 2D matrix with ns rows and n+1 columns, where ns is thenumber of states in the system and n is the number of switching intervals. The first and last column of Xss should be identical, corresponding to a valid steady-state solution. Each middle column marks the state values that the transition between switching intervals. The state variables are the capacitor voltages and inductor currents within the circuit.

Xss =

29.3758	0.0142	21.2742	20.0063	29.3758
−9.3797	19.9681	−1.2831	−0.0063	−9.3797
4.0411	4.0806	4.0816	4.0414	4.0411
7.7624	17.6958	17.7582	7.8690	7.7624

The row order of the state variables can be found by sim.stateNames These state variables can be visualized by using the sim.plotAllState function

In this case the value of M2_C being below the forward voltage at the end of the second deadtime (-9.3797) indicates that a diode should be conducting. This behavior is common when solving for the converter steady-state using discrete time state-space modeling. To resolve this error, a new time interval should be placed at the end of the second deadtime with D2 conducting to bring the converter into its accurate steady-state. This correction is done via the sim.findValidSteadyState function.

After running the function sim.findValidSteadyState

Xss =

21.5069	0.0149	21.2454	21.5149	20.0070	21.2137	21.5069
−1.5069	19.9665	−1.2547	−1.5149	−0.0070	−1.2181	−1.5069
4.0455	4.0852	4.0861	4.0861	4.0458	4.0458	4.0455
8.6805	18.6064	18.6657	18.6623	8.7472	8.7406	8.6805

The waveforms show there is no longer a diode conduction error. The changes to the time intervals and switching pattern can be observed via conv.swvec to see the canges in switching patter and sim.ts to see the changes in time interval duration

```
conv.swvec =
     1     0     0     0    % M1 Conducting
```

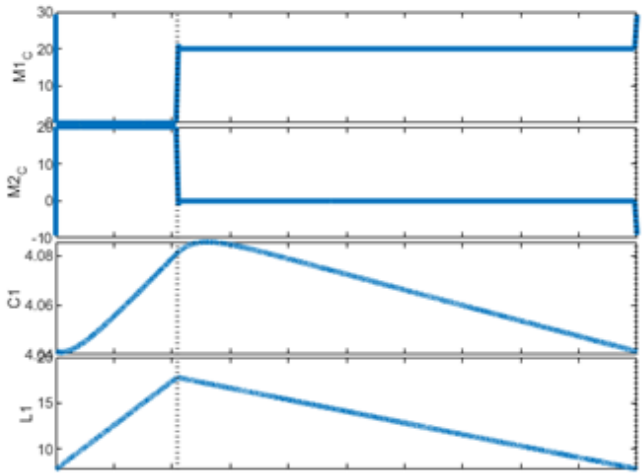


Figure 21: Solved steady state waveforms.

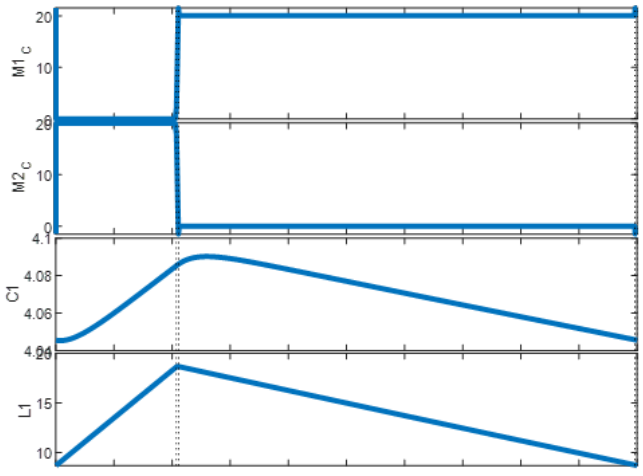


Figure 22: Solved steady-state waveforms with correct dioe behavior.

```
0     0     0     0     % Deadtime
0     0     0     1     % D2 Conducting
0     1     0     0     % M2 Conducting
0     0     0     0     % Deadtime
0     0     0     1     % D2 Conducting
```

Now that the accurate steady state of the converter has been found the average power loss of the converter can now be calculated by finding the average state values via

```
[ avgXs, avgYs ] = sim.ssAvgs(Xss);
```

where Xss is the steady-state values calculated using sim.steadyState, avgXs provides a vector of the average state values, and avgYs provides a vector of the average output values

The average output values by default contains all voltages and currents for all inductors and capacitors in the circuit. With this information the average output and input power can be calculated and reduced using an optimization algorithm. Note: the order of state variables, output variables, switch names and input names can be found via sim.stateNames, sim.ouputNames, sim.switchNames, and sim.inputNames respectively

B.3 Optimization

Now that the converter has been modeled, the transistor, inductor, or frequency selection can be optimized to reduce power loss. The provided optimization method perturbs Ron, Coss, RL and L values on the initial condition to find their effect on Ploss and then projects the expected loss on the other transistors and inductors in the design space. You can also set up one of MATLAB's optimization function such as genetic algorithm to run with the modeling function if you wish.

Vita

Jared Alexander Baxter is the son of Jerry and Deanna Baxter. He was born on January 27, 1995 and is a life-long resident of Sevier County, Tennessee. He attended public school in Sevier County and graduated from Sevier County High School in 2013. Jared entered the University of Tennessee, Knoxville in the fall of 2013 and received a Bachelor of Science degree in Electrical Engineering in 2017. In 2017 Jared was accepted to join the Bredesen Center as a part of Oak Ridge National Laboratory and the University of Tennessee, Knoxville where he received the Energy Science and Engineering Fellowship while pursuing a Doctor of Philosophy in Energy Science Engineering. During this time, he received a congruent Masters of Science in Electrical Engineering in the spring of 2021.